MCP Operations Guide

version q4-18

Copyright notice

2025 Mirantis, Inc. All rights reserved.

This product is protected by U.S. and international copyright and intellectual property laws. No part of this publication may be reproduced in any written, electronic, recording, or photocopying form without written permission of Mirantis, Inc.

Mirantis, Inc. reserves the right to modify the content of this document at any time without prior notice. Functionality described in the document may not be available at the moment. The document contains the latest information at the time of publication.

Mirantis, Inc. and the Mirantis Logo are trademarks of Mirantis, Inc. and/or its affiliates in the United States an other countries. Third party trademarks, service marks, and names mentioned in this document are the properties of their respective owners.

Preface

This documentation provides information on how to use Mirantis products to deploy cloud environments. The information is for reference purposes and is subject to change.

Intended audience

This documentation is intended for deployment engineers, system administrators, and developers; it assumes that the reader is already familiar with network and cloud concepts.

Documentation history

The following table lists the released revisions of this documentation:

Revision date	Description
February 8, 2019	Q4`18 GA

Introduction

This guide outlines the post-deployment Day-2 operations for an MCP cloud. It describes how to configure and manage the MCP components, perform different types of cloud verification, and enable additional features depending on your cloud needs. The guide also contains day-to-day maintenance procedures such as how to backup and restore, update and upgrade, or troubleshoot an MCP cluster.

Provision hardware

MCP uses the Ubuntu’s Metal-as-a-Service (MAAS) to provision hardware for an MCP deployment.

MAAS as a bare metal provisioning service requires an IPMI user to manage power state. This should be configured as part of the installation process.

MAAS provides DHCP to the network(s) on which compute nodes reside. Compute nodes then perform PXE boot from the network and you can configure MAAS to provision these PXE booted nodes.

Reinstall MAAS

If your MAAS instance is lost or broken, you can reinstall it. This section describes how to install MAAS from the Ubuntu Server 16.04 qcow2 image.

To reinstall MAAS:

1. Create a cloud-config disk:

    1. For example, create a configuration file named config-drive.yaml:

        ```
        #cloud-config
        debug: True
        ssh_pwauth: True
        disable_root: false
        chpasswd:
          list: |
            root:r00tme
            ubuntu:r00tme
          expire: False
        ```

        Note

        You must change the default password.

    2. Create the configuration drive:

        ```
        export VM_CONFIG_DISK="/var/lib/libvirt/images/maas/maas-config.iso"
        cloud-localds  --hostname maas01 --dsmode local ${VM_CONFIG_DISK}  config-drive.yaml
        ```

2. Create a VM system disk using the preloaded qcow2 image. For example:

    ```
    wget http://cloud-images.ubuntu.com/releases/xenial/release-20180306/ubuntu-16.04-server-cloudimg-amd64-disk1.img -O \
    /var/lib/libvirt/images/maas/maas-system-backend.qcow2

    export VM_SOURCE_DISK="/var/lib/libvirt/images/maas/maas-system.qcow2"
    qemu-img create -b /var/lib/libvirt/images/maas/maas-system-backend.qcow2 -f qcow2 ${VM_SOURCE_DISK} 100G
    ```

3. Create a VM using the predefine-vm script. For example:

    ```
    export MCP_VERSION="master"
    wget https://github.com/Mirantis/mcp-common-scripts/blob/${MCP_VERSION}/predefine-vm/define-cfg01-vm.sh

    chmod 0755 define-vm.sh
    export VM_NAME="maas01.[CLUSTER_DOMAIN]"
    ```

> Note
>
> You may add other optional variables that have default values and set them according to your deployment configuration. These variables include:
>
> - VM_MGM_BRIDGE_NAME="br-mgm"
> - VM_CTL_BRIDGE_NAME="br-ctl"
> - VM_MEM_KB="8388608"
> - VM_CPUS="4"
>
> The br-mgm and br-ctl values are the names of the Linux bridges. See MCP Deployment Guide: Prerequisites to deploying MCP DriveTrain for details. Custom names can be passed to a VM definition using the VM_MGM_BRIDGE_NAME and VM_CTL_BRIDGE_NAME variables accordingly.

4. Boot the created VM.
5. Log in to the VM using the previously defined password.
6. Proceed with installing MAAS:

```
sudo apt-get install maas
```

7. Configure MAAS as required to complete the installation.
8. Verify the installation by opening the MAAS web UI:

```
http://<MAAS-IP-ADDRESS>:5240/MAAS
```

9. If you have installed MAAS from the packages, create an initial (administrative) user first to log in to the MAAS web UI:

```
sudo maas createadmin --username=<PROFILE> --email=<EMAIL_ADDRESS>
```

Add an SSH key

To simplify access to provisioned nodes, add an SSH key that MAAS will deliver when provisioning these nodes.

To add an SSH key:

1. In the MAAS web UI, open the user page by clicking on your user name at the top-right conner.
2. Find the SSH keys section.
3. From the Source drop-down menu, specify the source of an SSH key using one of the options:
    - Launchpad, specifying your Launchpad ID
    - Github, specifying your Github ID
    - Upload, placing an existing public key to the Public key edit box
4. Click Import.

Seealso

MAAS Configuration

Add a boot image

You can select images with appropriate CPU architectures that MAAS will import, regularly sync, and deploy to the managed nodes.

To add a new boot image:

1. In the MAAS web UI, open the Images page.
2. Select the releases you want to make available, as well as any architecture.
3. Click Save selection.

Seealso

MAAS Configuration

Add a subnet

You can add new networking elements in MAAS such as fabrics, VLANs, subnets, and spaces. MAAS should detect new network elements automatically. Otherwise, you can add them manually.

To add a new subnet:

1. In the MAAS web UI, open the Subnets page.
2. Select Subnet in the Add drop-down menu at the top-right corner.
3. Specify Name, CIDR, Gateway IP, DNS servers, Fabric & VLAN, and Space.
4. Click Add subnet.

Seealso

MAAS Networking

Enable DHCP on a VLAN

Before enabling DHCP, ensure that you have the MAAS node network interface properly configured to listen to VLAN.

You can use external DHCP or enable MAAS-managed DHCP. Using an external DHCP server for enlistment and commissioning may work but is not supported. High availability also depends upon MAAS-managed DHCP.

To enable MAAS-managed DHCP on a VLAN:

1. In the MAAS web UI, open the Subnets page.
2. Click on the VLAN you want to enable DHCP on.
3. In the VLAN configuration panel, you find DHCP Disabled.
4. From the Take action drop-down menu at the top-right corner, select the Provide dhcp item.
5. In the Provide DHCP panel, verify or change the settings for Rack controller, Subnet, Dynamic range start IP, Dynamic range end IP.
6. Click Provide dhcp.
7. In the VLAN configuration panel, verify that DHCP is enabled.

Seealso

- Network interface configuration
- DHCP in MAAS

Enable device discovery

MAAS provides passive and active methods of device discovery.

Passive methods include:

- Listening to ARP requests
- DNS advertisements

To enable passive device discovery:

1. In the MAAS web UI, open the MAAS dashboard.
2. On the MAAS dashboard, turn on the Discovery enabled switch.
3. Verify if you can see the discovered devices on the MAAS dashboard.

Active methods include active subnet mapping that forces MAAS to discover nodes on all the subnets with enabled active mapping using an active subnet mapping interval value.

To enable active subnet mapping:

1. In the MAAS web UI, open the Settings page.
2. Go to the Device Discovery section.
3. From the drop-down menu, select the value for Active subnet mapping interval.
4. Open the Subnets page.
5. Click the subnet you want to enable active mapping on.
6. In the Subnet summary section, turn on the Active mapping switch.

Seealso

Device discovery

Add a new node

Using MAAS, you can add new nodes in an unattended way called enlistment or manually, when enlistment does not work.

MAAS enlistment uses a combination of DHCP with TFTP and PXE technologies.

> Note
>
> - To boot a node over PXE, enable netboot or network boot in BIOS.
> - For KVM virtual machines, specify the boot device as network in the VM configuration file and add the node manually. You need to configure the Virsh power type and provide access to the KVM host as described in BMC Power Types.
> - To ignore the already deployed machines and avoid issues with the wait_for_ready Salt state failure, you may need to set ignore_deployed_machines to true in the Reclass model:
>
> ```
> parameters:
>   maas:
>     region:
>       ignore_deployed_machines: true
> ```

To add a new node manually:

1. In the MAAS web UI, open the Nodes page.
2. Click the Add hardware drop-down menu at the top-right corner and select Add machine.
3. Set Machine name, Domain, Architecture, Minimum Kernel, Zone, MAC Address, Power type.

   > Note
   >
   > See Configure power management for more details on power types.

4. Click Save machine.

MAAS will add the new machine to the list of nodes with the status Comissioning.

> Seealso
>
> - Add nodes
> - BMC Power Types

Configure power management

MAAS supports many types of power control, from standard IPMI to non-standard types such as virsh, VMware, Nova, or even completely manual ones that require operator intervention. While most servers may use their own custom vendor management, for example, iLO or DRAC, standard IPMI controls are also supported, and you can use IPMI as shown in the following example.

To configure IPMI node power management type:

1. In the MAAS web UI, open the Nodes page.
2. In the list of nodes, select the one you want to configure.
3. In the machine configuration window, go to the Power section and click the Edit button at the top-right conner.
4. Select IPMI for Power type from the drop-down menu.
5. Specify parameters for the IPMI power type: Power driver, IP address, Power user, Power password, and Power MAC.
6. Click Save changes.

After saving the changes, MAAS will verify that it can manage the node through IPMI.

Seealso

BMC Power Types

Commission a new node

When you add a new node, MAAS automatically starts commissioning the node once configuration is done. Also, you can commission a node manually.

To commision a new node manually:

1. In the MAAS web UI, open the Nodes page.
2. In the list of nodes, click the node you want to configure.
3. In the node configuration window, click Take action at the top-right conner and select Commission.
4. Select additional options by setting the appropriate check boxes to allow SSH access and prevent machine from powering off, retain network and storage configuration if you want to preserve data on the node. For example, if a node comes from an existing cloud with an instance on it.
5. Click Go to start commissioning.
6. Verify that the node status has changed to Ready and hardware summary in the node configuration window has been filled with values other than zeros, which means commisionining was successful.

Note

Use MAAS CLI to commission a group of machines.

Seealso

- MAAS CLI
- Commission nodes
- Add a new node

Deploy a node

Once a node has been commissioned, you can deploy it.

The deployment operation includes installing an operating system and copying the SSH keys imported to MAAS. As a result, you can access the deployed node through SSH using the default user account ubuntu.

To deploy a node:

1. In the MAAS web UI, open the Nodes page.
2. In the list of nodes, verify that the commisioned node is in the Ready state.
3. Click on the node to open the configuration window.
4. In the node configuration window, click the Take action button at the top-right conner and select the Deploy item.
5. Specify the OS, release, and kernel options.
6. Click Go to start the deployment.

Once the deployment is finished, MAAS will change the node status to Deployed.

Seealso

- Deploy nodes
- Add an SSH key

Redeploy a node

To redeploy a node:

1. In the MAAS web UI, open the Nodes page.
2. In the list of nodes, select the node you want to redeploy.
3. In the node configuration window, click Take action at the top-right coner and select Release.
4. Verify that the node status has changed to Ready.
5. Redeploy the node as described in Deploy a node.

Delete a node

Warning

Deleting a node in MAAS is a permanent operation. The node will be powered down, and removed from the MAAS database. All existing configuration done on the node such as name, hardware specs, and power control type will be permanently lost. The node can be readded again, however, it will be unrecognized by MAAS. In such case, you will need to add the node as a new one and reconfigure from scratch.

To delete a node:

1. In the MAAS web UI, open the Nodes page.
2. In the list of nodes, select the node you want to delete.
3. In the node configuration window, click the Take action button at the top-right coner and select Delete.
4. Click Go to confirm the deletion.

SaltStack operations

SaltStack is an orchestration and configuration platform that implements the model-driven architecture (MDA) that you can use to turn services configuration described using Reclass models and Salt Formulas into actual services running on nodes.

Salt Minion operations

Run a command on a node

The Salt integrated cmd.run function is highly flexible and enables the operator to pass nearly any bash command to a node or group of nodes and functions as a simple batch processing tool.

To run a command on a node, execute:

```
salt '[node]' cmd.run '[cmd]'
```

List services on a node

The Salt integrated service.get_all function shows available services on the node.

To list services on a node, run:

```
salt '<NODE_NAME>' service.get_all
```

Restart a service on a node

You can use the Salt integrated service.restart function to restart services.

To restart a service on a node, run:

```
salt '<NODE_NAME>' service.restart <SERVICE_NAME>
```

> Note
>
> If you do not know the name of the service or unsure which services are available, see the List services on a node section.

Verify Minions have joined the Master

If you are not sure whether or not a Minion has been joined the Master, verify the output of salt-key. The output of the command lists all known keys in the following states:

- **Accepted**

  Nodes in this state have successfully joined the Salt Master
- **Denied**

  Nodes in this state have not successfully joined because of a bad, duplicate, or rejected key. Nodes in this state require additional user action to join the Master.
- **Rejected**

  Nodes in this state have been explicitly rejected by an administrator.

To verify Minions have joined the Master, run:

```
salt-key
```

Example of a system response:

```
Accepted Keys:
<NODE_NAME>.domain.local
... [snip] ...
Execute salt-key:
Denied Keys:
Unaccepted Keys:
Rejected Keys:
```

Ping a Minion from the Master

You can ping all properly running Salt Minion nodes from the Salt Master. To verify that you have network availability between Salt Minion nodes and the Salt Master node, use the test.ping command.

To ping a Minion from the Master:

```
salt '<NODE_NAME>.domain.local' test.ping
```

Example of a system response:

```
<NODE_NAME>
 True
```

Salt States operations

Salt State is a declarative or imperative representation of a system state.

List available States of a Minion

A Salt Minion node can have different States.

To list available States of a Minion, execute on a node:

```
salt-call state.show_top
```

Example of a system response:

```
local:
 ----------
 base:
     - linux
     - ntp
     - salt
     - heka
     - openssh
     - nova
     - opencontrail
     - ceilometer
```

Apply a State to a Minion

You can apply changes to a Minion’s State from the Salt Master.

To apply a State to a Minion, run:

```
salt '<NODE_NAME>' state.sls <STATE_NAME>
```

Salt Formula operations

Salt Formula is a declarative or imperative representation of a system configuration.

Verify and validate a Salt Formula

You can verify and validate a new Salt Formula before applying it by running a quick test for invalid Jinja, YAML, and a Salt state.

To verify a SLS file in a Salt Formula, run:

```
salt '*' state.show_sls <SLS_FILE_NAME>
```

To validate the Salt Formula, run in the test-only (dry-run) mode using the test option:

```
salt '*' state.apply test
```

Seealso

Salt Command Line Reference

Apply a Salt Formula

This section covers how you can test and apply a Salt Formula.

To apply all configured states (highstate) from a Salt Formula to all Minions, run on the Salt Master:

```
salt '*' state.apply
```

Note

This command is equal to:

```
salt '*' state.highstate
```

To apply individual SLS files in a Salt Formula, run:

```
salt '*' state.apply <SLS_FILE_1>,<SLS_FILE_2>
```

Warning

Applying Salt Formulas on more than 100 nodes may result in numerous failures.

Note

SaltStack runs new states in parallel leading to temporary out of service that may affect end users. To avoid taking down services on all the nodes at the same time, you can stagger highstates in a batch mode.

To apply a Salt Formula on a big number of nodes, for example, more than 100 nodes, follow one of the approaches below.

- Use the --batch-size or -b flags to specify the number of nodes to have Salt apply a state in parallel:

  ```
  salt --batch-size <NUMBER_OF_NODES> '*' state.apply
  ```

- Specify a percentage of nodes to apply a highstate on:

```
salt -b <PERCENTAGE> '*' state.apply
```

- Use node name conventions in the form of <GROUP>.<NODE_TYPE_NAME><NUM> to run a highstate by a pattern. For example: group1.cmp001:

```
salt 'group1.cmp*' state.highstate
```

- Use Node Groups that you can define in the Salt Master configuration file /etc/salt/master. To run a highstate on nodes within a Node Group, run:

```
salt -N <GROUP_NODE> state.apply
```

- Use Grains for grouping nodes specifying a grain variable in the /etc/salt/grains configuration file and then specify the grain value in the Salt command to apply a highstate for the nodes that have this grain value assigned:

```
salt -G <GRAIN_NAME>:<GRAIN_VALUE> state.apply
```

Note

You can use --batch-size flag together with Node Groups and Grains. For example:

```
salt --batch-size 10% -N computes1 state.apply
salt -b 5 -N compute:compute1 state.apply
```

Seealso

- Salt Node Groups
- Salt Grains
- Salt Command Line Reference

Replace the Salt Master keys

In case your Salt Master keys have been compromised, you can replace both Salt Master CA and RSA SSH keys. The replacement procedure of the Salt Master keys does not affect your cloud environment, only the Salt structure is updated.

Salt Master keys structure

File path	Description
/etc/salt/minion.d/_pki.conf	PKI configuration file pointing to the current Salt Master CA certificate path
/etc/pki/ca/salt_master_ca/	Catalog for the Salt Master CA certificate
/etc/pki/ca/salt_master_ca/ca.crt	Salt Master CA certificate
/etc/pki/ca/salt_master_ca/ca.key	Salt Master CA certificate key
/etc/pki/ca/salt_master_ca/certs	Catalog for the Salt minion certificates signed by the Salt Master CA certificate
/etc/pki/ca/salt_master_ca/certs/XX:XX:XX:XX:XX:XX:XX:XX.crt	Salt minion certificate signed by CA
• /etc/salt/pki/minion/minion.pub • /etc/salt/pki/minion/minion.pem • /etc/salt/pki/minion/minion_master.pub	Salt Master SSH RSA private and public keys for Salt minion
• /etc/salt/pki/master/master.pem • /etc/salt/pki/master/master.pub	Salt Master SSH RSA private and public keys for Salt Master
/etc/salt/pki/master/minions/ctl01.example.int	RSA SSH minion key for communication with Salt Master. Equals to /etc/salt/pki/minion/minion.pub

©2025, Mirantis Inc.

Replace the Salt Master and Salt minions SSH RSA keys

This section provides the instruction of how to replace the Salt Master and the Salt minions SSH RSA keys.

To replace Salt Master and Salt minions SSH RSA keys:

1. Log in to the Salt Master node.
2. Verify that all nodes are available:

   ```
   salt \* test.ping
   ```

3. Create classes/cluster/<cluster-name>/infra/minions-maintenance.yml with the following content:

   ```
   parameters:
     _param:
       char_number_sign: "#"
     linux:
       system:
         file:
           restart-minion.sh:
             name: /usr/local/bin/restart-minion.sh
             user: root
             group: root
             mode: 750
             contents: |
               ${_param:char_number_sign}!/bin/bash
               /usr/sbin/service salt-minion stop
               rm -f /etc/salt/pki/minion/minion*;
               /usr/sbin/service salt-minion start
         job:
           restart-minion:
             enabled: True
             command: /usr/local/bin/restart-minion.sh
             user: root
             minute: '*/5'
   ```

4. Include the minions-maintenance class in the infra/init.yml file:

   ```
   classes:
   ...
   - cluster.<cluster-name>.infra.minions-maintenance
   ```

5. Put all Salt minions into the maintenance mode:

   ```
   salt \* state.sls linux.system.file,linux.system.job
   ```

The command above will cause all Salt minions to remove their keys and restart each 5 minutes.

6. Count your minions:

```
MINIONS_NUMBER=$(ls /etc/salt/pki/master/minions/ -1 | wc -l)
```

7. Verify that all minions are put into the maintenance mode by checking the diff between /master/minions/ and master/minions_denied/:

```
diff <(ls /etc/salt/pki/master/minions/ -1 | wc -l) \
<(ls /etc/salt/pki/master/minions_denied/ -1 | wc -l)
```

   Start the verification at the beginning of the zero or fifth minute to have enough time to purge old minions keys. Proceed only if the diff is empty. If you see the diff for more than 10 minutes, some minions are rejected to execute the cron job. Identify the root cause of the issue and resolve it before proceeding.

8. Stop the Salt Master node:

```
service salt-master stop
```

9. Change directory to the Salt Master key:

```
cd /etc/salt/pki/master
```

10. Remove the Salt Master key:

```
rm -f master.p*
```

11. Generate a new key without a password:

```
ssh-keygen -t rsa -b 4096 -f master.pem
```

12. Remove the RSA public key for the new key as Salt Master does not require it:

```
rm -f master.pem.pub
```

13. Generate the .pem public key for the Salt Master node:

```
openssl rsa -in master.pem -pubout -out master.pub
```

Note

Press Enter for the empty password.

14. Remove the minions list on the Salt Master node:

```
salt-key -y -d '*'
```

15. Start the Salt Master node:

```
service salt-master start
```

16. Verify that the minions are present:

Note

The minions should register on the first or sixth minute.

```
salt-key -L
```

17. Verify that the current minions count is the same as in the step 6:

```
ls /etc/salt/pki/master/minions/ -1 | wc -l
echo $MINIONS_NUMBER
```

18. Disable the maintenance mode for minions by disabling the cron job in classes/cluster/<cluster-name>/infra/minions-maintenance.yml:

```
job:
  restart-minion:
    enabled: False
```

19. Update your minions:

```
salt \* state.sls linux.system.job
```

20. Remove the minions-maintenance class from the infra/init.yml file:

```
classes:
...
# Remove the following line
- cluster.<cluster-name>.minions-maintenance
```

21. Remove the minions-maintenance pillar definition from the Reclass model:

```
rm -f classes/cluster/<cluster-name>/infra/minions-maintenance.yml
```

Replace the Salt Master CA certificates

This section provides the instruction on how to replace the Salt Master CA certificates.

To replace the Salt Master CA certificates:

1. Log in to the Salt Master node.
2. Back up the running Salt configuration in case the rollback is required:

   ```
   tar cf /root/salt-backup.tar /etc/salt /etc/pki/ca/salt_master_ca/
   gzip -9 /root/salt-backup.tar
   ```

3. List all currently issued certificates.

   Currently, the index file for Salt Master CA does not exist. Therefore, you can list all certificates and find the latest ones using the salt_cert_list.py script:

   > Note
   >
   > The script is available within Mirantis from the mcp-common-scripts GitHub repository.

   ```
   ./salt_cert_list.py
   ```

   Example of system response:

   ```
   /etc/pki/ca/salt_master_ca/certs/18:63:9E:A6:F3:7E:10:5F.crt (proxy, 10.20.30.10, horizon.multinode-ha.int)
   /etc/pki/ca/salt_master_ca/certs/EB:51:7C:DF:CE:E7:90:52.crt (10.20.30.10, 10.20.30.10, *.10.20.30.10)
   /etc/pki/ca/salt_master_ca/certs/15:DF:66:5C:8D:8B:CF:73.crt (internal_proxy, mdb01, mdb01.multinode-ha.int, 192.168.2.116, 192.168.2.115, 10.20.30.10)
   /etc/pki/ca/salt_master_ca/certs/04:30:B0:7E:76:98:5C:CC.crt (rabbitmq_server, msg01, msg01.multinode-ha.int)
   /etc/pki/ca/salt_master_ca/certs/26:16:E7:51:E4:44:B4:65.crt (mysql_server, 192.168.2.53, 192.168.2.50, dbs03, dbs03.multinode-ha.int)
   /etc/pki/ca/salt_master_ca/certs/78:26:2F:6E:2E:FD:6A:42.crt (internal_proxy, ctl02, 192.168.2.12, 10.20.30.10, 192.168.2.10)
   ...
   ```

4. Update classes/cluster/<cluster_name>/infra/config.yml with the required values for the Salt Master CA. For example:

   ```
   parameters:
     _param:
       salt_minion_ca_country: us
       salt_minion_ca_locality: New York
       salt_minion_ca_organization: Planet Express
       salt_minion_ca_days_valid_authority: 3650
       salt_minion_ca_days_valid_certificate: 365
   ```

5. Replace the Salt Master CA certificates:

   ```
   rm -f /etc/pki/ca/salt_master_ca/ca*
   salt-call state.sls salt.minion.ca -l debug
   ```

©2025, Mirantis Inc.

6. Publish the Salt Master CA certificates as described in Publish CA certificates.
7. Replace the certificates in your cloud environment according to the list of certificates obtained in the step 3 of this procedure as described in Manage certificates for the affected services.

Seealso

SaltStack components

DriveTrain operations

This section describes the main capabilities of DriveTrain, the MCP lifecycle management engine.

Job configuration history

The DriveTrain Jenkins provides the capability to inspect the history of jobs configuration changes using the Job Configuration History plugin. This plugin captures and stores the changes to all jobs configured in Jenkins and enables the DriveTrain administrator to view these changes. It allows identifying the date of the change, the user that created the change, and the content of the change itself.

To use the Job Configuration History plugin:

1. Log in to the Jenkins web UI as an administrator using the FQDN of your cloud endpoint and port 8081. For example, https://cloud.example.com:8081.
2. Navigate to Job Config History > Show job configs only.
3. Click the required job and review the list of recorded changes.

Alternatively, you can access the history of a job by clicking Job Config History from the particular job view itself.

Seealso

Official plugin documentation

Abort a hung build in Jenkins

This section provides the instruction on how to abort the hung Jenkins build if it does not restore after the Jenkins dedicated node is restared, for example.

To abort a hung build, select from the following options

- Abort the job build from the Jenkins web UI:

    1. Log in to the Jenkins web UI as an Administrator using the FQDN of your cloud endpoint and the 8081 port. For example, https://cloud.example.com:8081.
    2. Navigate to Manage Jenkins > Script Console.
    3. Run the following script setting the job name and number of the hung build accordingly:

    ```
    def build = Jenkins.instance.getItemByFullName("jobName").getBuildByNumber(jobNumber)
    build.doStop()
    build.doKill()
    ```

- Abort the job build from a cid node (if the previous option did not help):

    1. Log in to any cid node.
    2. Run:

    ```
    cd /srv/volumes/jenkins/jobs/<job-name>/builds/
    rm -rf <hung-build-number>
    ```

    3. Log in to the Jenkins web UI as an Administrator using the FQDN of your cloud endpoint and the 8081 port. For example, https://cloud.example.com:8081.
    4. Navigate to Manage Jenkins > Reload Configuration from Disk, click to reload. Or restart the Jenkins instance.

Seealso

Official Jenkins documentation: Aborting a build

Enable Jenkins audit logging

This section instructs you on how to enable the audit logging in Jenkins by enabling the Audit Trail Jenkins plugin. The plugin allows keeping a log of the users who performed particular Jenkins operations, such as managing and using jobs.

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

> Note
>
> If Jenkins is disabled on the Salt Master node, skip the step 3 of the procedure below.

To setup Audit logging in Jenkins:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. In the cicd/control/leader.yml file, configure any of three logger types that include console, file, and syslog.

   > Note
   >
   > By default, only the console output is collected by Fluentd if enabled.

   Pillars examples:

   1. For the console logger:

      ```
      parameters:
        jenkins:
          client:
            audittrail:
              loggers:
                console_logger:
                  type: console
                  output: STD_OUT
                  date_format: "yyyy-MM-dd HH:mm:ss:SSS"
                  log_prefix: ""
      ```

> Note
>
> The date_format and log_prefix parameters in the example above are defaults and can be skipped.

2. For the file logger:

```yaml
parameters:
  jenkins:
    client:
      audittrail:
        loggers:
          file_logger:
            type: file
            log: /var/jenkins_home/file_logger.log
            limit: 100
            count: 10
```

> Note
>
> The limit parameter stands for the file limit size in MB. The count parameter stands for the number files to keep.

3. For the syslog logger:

```yaml
parameters:
  jenkins:
    client:
      audittrail:
        loggers:
          syslog_logger:
            type: syslog
            syslog_server_hostname: 'syslog.host.org'
            syslog_server_port: 514
            syslog_facility: SYSLOG
            app_name: jenkins
            message_hostname: ""
            message_format: RFC_3164
```

4. To configure the audit Logging for Jenkins on the Salt Master node, add the similar pillars to infra/config/jenkins.yml.
5. Refresh pillars:

```
salt -C 'I@jenkins:client' saltutil.refresh_pillar
```

6. Apply the changes:

```
salt -C 'I@jenkins:client:audittrail' state.apply jenkins.client.audittrail
```

Jenkins Matrix-based security authorization

The DriveTrain Jenkins uses Matrix-based security authorization by default. It allows you to grant specific permissions to users and groups. Jenkins uses DriveTrain OpenLDAP server as an identity provider and authentication server.

By default, the Jenkins server includes the following user groups:

- admins

  Contains administrative users with the Jenkins Administer permission.
- Authenticated Users

  Includes all users authenticated through the DriveTrain OpenLDAP server. This group has no permissions configured by default.

The Matrix-based security plugin enables the operator to configure the following types of permissions:

- **Overall**

  Either Administer or Read permissions can be set overall.
- **Credentials**

  Permissions to create, delete, update, and view authentication credentials, and manage domains.
- **Gerrit**

  Permissions to manually trigger and retrigger Gerrit integration plugin to run specific jobs normally initiated by the plugin.
- **Agents**

  Permissions to manage Jenkins agents on worker nodes.
- **Job**

  Permissions for specific operations on Jenkins jobs, including build creation, configuration, and execution.
- **Run**

  Permissions to run and rerun jobs.
- **View**

  Permissions to manage views in the Jenkins UI.
- **SCM**

  Permissions to use SCM tags.
- **Metrics**

  Permissions to view and configure metrics.
- **Lockable resources**

  Permissions to reserve and unlock lockable resources manually.
- **Artifactory**

  Permissions to use the Artifactory integration plugin (only if Artifactory is installed).

©2025, Mirantis Inc.

To configure the Matrix-based Security Authorization:

1. Log in to the Jenkins web UI as an administrator using the FQDN of your cloud endpoint and port 8081. For example, https://cloud.example.com:8081.
2. Navigate to Manage Jenkins > Configure Global Security.
3. Scroll to the Authorization section to view and change the Matrix-based security settings.

Seealso

Official Jenkins Security documentation

Remove executors on Jenkins master

The DriveTrain Jenkins enabled on the Salt Master node enables you to run any job on any Jenkins slave or master. Though, running a job on Jenkins master can lead to job failures since Jenkins master is not intended to be a job executor.

Starting from the MCP 2019.2.4 maintenance update, MCP disables executors on Jenkins master by default to prevent failures of the jobs that run on the Salt Master node. For the MCP versions earlier than 2019.2.4, Mirantis recommends setting zero executors on Jenkins master to disable jobs scheduling on this agent node as described below.

> Note
>
> If Jenkins is disabled on the Salt Master node (for details, refer to MCP Deployment Guide: Deploy CI/CD), you can skip the steps below or simply update your cluster configuration without applying the Salt states.

To set zero executors on Jenkins master on the Salt Master node:

1. Log in to the Salt Master node.
2. In ./classes/cluster/<cluster_name>/infra/config/jenkins.yml, add the following pillar data:

   ```
   parameters:
     _param:
       jenkins:
         client:
           ...
           node:
             master:
               num_executors: 0
             ...
   ```

3. Refresh pillars:

   ```
   salt -C 'I@salt:master' saltutil.refresh_pillar
   ```

4. Apply the changes:

   ```
   salt -C 'I@salt:master' state.apply jenkins.client
   ```

©2025, Mirantis Inc.

Remove anonymous access to Jenkins on the Salt Master node

The DriveTrain Jenkins enabled on the Salt Master node is configured to allow anonymous users to access the Jenkins web UI including the listing of the Jenkins jobs and builds in the web UI.

For security reasons, starting from the MCP 2019.2.4 maintenance update, by default, only authorized users have access to Jenkins on the Salt Master node. For the MCP versions earlier than 2019.2.4, Mirantis recommends configuring Jenkins as described below.

> Note
>
> If Jenkins is disabled on the Salt Master node (for details, refer to MCP Deployment Guide: Deploy CI/CD), you can skip the steps below or simply update your cluster configuration without applying the Salt states.

To remove anonymous access to Jenkins on the Salt Master node:

1. Log in to the Salt Master node.
2. In ./classes/cluster/<cluster_name>/infra/config/jenkins.yml, replace anonymous with authenticated for jenkins_security_matrix_read:

   ```
   parameters:
     _param:
       jenkins_security_matrix_read:
       - authenticated
   ```
3. Refresh pillars:

   ```
   salt -C 'I@salt:master' saltutil.refresh_pillar
   ```
4. Apply the changes:

   ```
   salt -C 'I@salt:master' state.apply jenkins.client
   ```

Use SSH Jenkins slaves

By default, Jenkins uses Java Network Launch Protocol (JNLP) for Jenkins slave connection. Starting from the MCP 2019.2.5 maintenance update, you can set up SSH connection for Jenkins slaves instead of JNLP using the steps below.

> Note
>
> If Jenkins is disabled on the Salt Master node (for details, refer to MCP Deployment Guide: Deploy CI/CD), skip the steps 2 and 3 of the procedure below.

To use SSH connection instead of JNLP for Jenkins slaves:

1. Log in to the Salt Master node.
2. Configure Jenkins Master for the Salt Master node to use SSH Jenkins slaves:
   1. Verify your existing SSH keys for Jenkins admin key:

      ```
      salt-call pillar.get _param:jenkins_admin_public_key_generated
      salt-call pillar.get _param:jenkins_admin_private_key_generated
      ```

      The system output must be not empty.

      If you do not have SSH keys, generate ones:

      ```
      ssh-keygen
      ```

   2. In ./classes/cluster/<cluster_name>/infra/config/jenkins.yml:
      1. Replace the system.docker.swarm.stack.jenkins.slave_single or system.docker.swarm.stack.jenkins.jnlp_slave_single class (the one that is present in model) with the following class:

         ```
         classes:
         ...
         - system.docker.swarm.stack.jenkins.ssh_slave_single
         ```

      2. Remove the following classes if present:

         ```
         classes:
         ...
         - system.docker.client.images.jenkins_master
         - system.docker.client.images.jenkins_slave
         ```

      3. Change the Jenkins slave type to ssh instead of jnlp:

```
parameters:
  ...
  jenkins:
    client:
      node:
        slave01:
          launcher:
            type: ssh
```

4. Add the SSH keys parameters to the parameters section:

    - If you use existing SSH keys:

      ```
      parameters:
        _param:
          ...
          jenkins_admin_public_key: ${_param:jenkins_admin_public_key_generated}
          jenkins_admin_private_key: ${_param:jenkins_admin_private_key_generated}
          ...
      ```

    - If you generated new SSH keys in the step 2.1:

      ```
      parameters:
        _param:
          ...
          jenkins_admin_public_key: <ssh-public-key>
          jenkins_admin_private_key: <ssh-private-key>
          ...
      ```

3. Remove the JNLP slave from Jenkins on Salt Master node:

    1. Log in to Salt Master node Jenkins web UI.
    2. Navigate to Manage Jenkins > Manage nodes.
    3. Select slave01 > Delete agent. Click yes to confirm.

4. Configure Jenkins Master for the cid nodes to use SSH Jenkins slaves:

    1. Verify that the Jenkins SSH key is defined in the Reclass model:

       ```
       salt 'cid01*' pillar.get _param:jenkins_admin_public_key
       salt 'cid01*' pillar.get _param:jenkins_admin_private_key
       ```

    2. In ./classes/cluster/<cluster_name>/cicd/control/leader.yml:

        1. Replace the system.docker.swarm.stack.jenkins class with system.docker.swarm.stack.jenkins.master and the

system.docker.swarm.stack.jenkins.jnlp_slave_multi class with system.docker.swarm.stack.jenkins.ssh_slave_multi if present, or add system.docker.swarm.stack.jenkins.ssh_slave_multi explicitly.

2. Add the system.jenkins.client.ssh_node class right below the system.jenkins.client.node class:

```
classes:
...
- system.jenkins.client.node
- system.jenkins.client.ssh_node
```

5. Remove the JNLP slaves from Jenkins on the cid nodes:

    1. Log in to cid Jenkins web UI.
    2. Navigate to Manage Jenkins > Manage nodes.
    3. Delete slave01, slave02 and slave03 using the menu. For example: slave01 > Delete agent. Click yes to confirm.

6. Refresh pillars:

```
salt -C 'I@jenkins:client' saltutil.refresh_pillar
salt -C 'I@docker:client' saltutil.refresh_pillar
```

7. Pull the ssh-slave Docker image:

```
salt -C 'I@docker:client:images' state.apply docker.client.images
```

8. Apply the changes:

```
salt -C 'I@jenkins:client and I@docker:client' state.apply docker.client
salt -C 'I@jenkins:client' state.apply jenkins.client
```

Enable HTTPS access from Jenkins to Gerrit

By default, Jenkins uses the SSH connection to access Gerrit repositories. This section explains how to set up the HTTPS connection from Jenkins to Gerrit repositories.

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

To enable access from Jenkins to Gerrit through HTTPS:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. In the cicd/control/leader.yml file:
    1. Replace the system.jenkins.client.credential.gerrit class with the system.jenkins.client.credential.gerrit_http class:

        ```
        classes:
        ...
        - system.jenkins.client.credential.gerrit_http
        ```

    2. Redefine the jenkins_gerrit_url parameter as follows:

        ```
        jenkins_gerrit_url: "https://${_param:haproxy_gerrit_bind_host}:${_param:haproxy_gerrit_bind_port}"
        ```

4. Refresh pillars:

    ```
    salt -C 'I@jenkins:client and not I@salt:master' saltutil.refresh_pillar
    ```

5. Apply the changes:

    ```
    salt -C 'I@jenkins:client and not I@salt:master' state.apply jenkins.client
    ```

Manage secrets in the Reclass model

MCP uses the GPG encryption to protect sensitive data in the Git repositories of the Reclass model. The private key from the encrypted data is stored on the Salt Master node and is available to the root user only. Usually, the data stored in the secrets.yml files located in the /srv/salt/reclass/cluster directory is encrypted. The decryption key is located in a keyring in /etc/salt/gpgkeys.

> Note
>
> MCP uses the secrets file name for organizing sensitive data management. If required, you can encrypt data in other files, as well as use unencrypted data in the secrets.yml files.

The secrets encryption feature is not enabled by default. To enable the feature, define secrets_encryption_enabled: 'True' in the Cookiecutter context before the deployment. See MCP Deployment Guide: Infrastructure related parameters: Salt Master for the details.

To change a password:

1. Get the ID of the private key in question:

   ```
   # GNUPGHOME=/etc/salt/gpgkeys gpg --list-secret-keys
   ```

   The machine-readable version of the above command:

   ```
   # GNUPGHOME=/etc/salt/gpgkeys gpg --list-secret-keys --with-colons | awk -F: -e '/^sec/{print $5}'
   ```

2. Encrypt the new password:

   ```
   # echo -ne <new_password> | GNUPGHOME=/etc/salt/gpgkeys gpg --encrypt --always-trust -a -r <key_id>
   ```

3. Add the new password to secrets.yml.

To decrypt the data:

To get the decoded value, pass the encrypted value to the command:

```
# GNUPGHOME=/etc/salt/gpgkeys gpg --decrypt
```

To change the secret encryption private key:

1. Add a new key to keyring in /etc/salt/gpgkeys using one of the following options:

   - Import the existing key:

     ```
     # GNUPGHOME=/etc/salt/gpgkeys gpg --import < <key_file>
     ```

   - Create a new key:

     ```
     # GNUPGHOME=/etc/salt/gpgkeys gpg --gen-key
     ```

2. Replace all encrypted fields in all secrets.yml files with the encrypted value for new key_id.

Seealso

MCP Deployment Guide: Enable all secrets encryption

Configure allowed and rejected IP addresses for the GlusterFS volumes

> Note
>
> This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

This section provides the instruction on how to configure the list of allowed and rejected IP addresses for the GlusterFS volumes.

By default, MCP restricts the access to the control network for all preconfigured GlusterFS volumes.

To configure the GlusterFS authentication:

1. Log in to the Salt Master node.
2. Open your project Git repository with the Reclass model on the cluster level.
3. In the infra/glusterfs.yml file, configure the GlusterFS authentication depending on the needs of your MCP deployment:

   - To adjust the list of allowed and rejected IP addresses on all preconfigured GlusterFS volumes, define the glusterfs_allow_ips and glusterfs_reject_ips parameters as required:

     ```
     parameters:
       _param:
         glusterfs_allow_ips: <comma-seprated list of IPs>
         glusterfs_reject_ips: <comma-seprated list of IPs>
     ```

     > Note
     >
     > You can use the * wildcard to specify the IP ranges.

     Configuration example:

     ```
     parameters:
       _param:
         glusterfs_allow_ips: 10.0.0.1, 192.168.1.*
         glusterfs_reject_ips: 192.168.1.201
     ```

The configuration above allows the access to all GlusterFS volumes from 10.0.0.1 and all IP addresses in the 192.168.1.0/24 network except for 192.168.1.201.

- To change allowed and rejected IP addresses for a single volume:

```
parameters:
  glusterfs:
    server:
      volumes:
        <volume_name>:
          options:
            auth.allow: <comma-seprated_list_of_IPs>
            auth.reject: <comma-seprated_list_of_IPs>
```

- To define the same access-control lists (ACL) as for all preconfigured GlusterFS volumes to a custom GlusterFS volume, define the auth.allow and auth.reject options for the targeted volume as follows:

```
auth.allow: ${_param:glusterfs_allow_ips}
auth.reject: ${_param:glusterfs_reject_ips}
```

4. Apply the changes:

```
salt -I 'glusterfs:server:role:primary' state.apply glusterfs
```

Manage users in OpenLDAP

DriveTrain uses OpenLDAP to provide authentication and metadata for MCP users. This section describes how to create a new user entry in the OpenLDAP service through the Reclass cluster metadata model and grant the user permissions to access Gerrit and Jenkins.

To add a user to an OpenLDAP server:

1. Log in to the Salt Master node.
2. Check out the latest version of the Reclass cluster metadata model from the Git repository for your project.
3. Create a new directory called people in classes/cluster/<CLUSTER_NAME>/cicd/:

   ```
   mkdir classes/cluster/<cluster_name>/cicd/people
   ```

   New user definitions will be added to this directory.
4. Create a new YAML file in the people directory for a new user. For example, joey.yml:

   ```
   touch classes/cluster/<cluster_name>/cicd/people/joey.yml
   ```

5. In the newly created file, add the user definition. For example:

   ```
   parameters:
     _param:
       openldap_pw_joey: "<ENCRYPTED_PASSWORD>"
     openldap:
       client:
         entry:
           people:
             entry:
               jdoe:
                 attr:
                   uid: joey
                   userPassword: ${_param:openldap_pw_joey}
                   uidNumber: 20600
                   gidNumber: 20001
                   gecos: "Joey Tribbiani"
                   givenName: Joey
                   sn: Tribbiani
                   homeDirectory: /home/joey
                   loginShell: /bin/bash
                   mail: joey@domain.tld
                 classes:
                   - posixAccount
                   - inetOrgPerson
   ```

```
          - top
          - shadowAccount
```

Parameters description:

- openldap_pw_joey

  The user password for the joey user that can be created using the following example command:

  ```
  echo "{CRYPT}$(mkpasswd --rounds 500000 -m sha-512 \
    --salt `head -c 40 /dev/random | base64 | sed -e 's/+/./g' \
    |  cut -b 10-25` 'r00tme')"
  ```

  Substitute r00tme with a user encrypted password.

- uid

  The case-sensitive user ID to be used as a login ID for Gerrit, Jenkins, and other integrated services.

- userPassword: ${_param:openldap_pw_joey}

  The password for the joey user, same as the openldap_pw_joey value.

- gidNumber

  An integer uniquely identifying a group in an administrative domain, which a user should belong to.

- uidNumber

  An integer uniquely identifying a user in an administrative domain.

6. Add the new user definition from joey.yml as a class in classes/cluster/<CLUSTER_NAME>/cicd/control/leader.yml:

   ```
   classes:
     ...
     - cluster.<CLUSTER_NAME>.cicd.control
     - cluster.<CLUSTER_NAME>.cicd.people.joey
   ```

   By defining the cluster level parameters of the joey user and including it in the classes section of cluster/<CLUSTER_NANE>/cicd/control/leader.yml, you import the user data to the cid01 node inventory, although the parameter has not been rendered just yet.

7. Commit the change.
8. Update the copy of the model on the Salt Master node:

   ```
   sudo git -C /srv/salt/reclass pull
   ```

9. Synchronize all Salt resources:

```
sudo salt '*' saltutil.sync_all
```

10. Apply the changes:

```
sudo salt 'cid01*' state.apply openldap
```

Example output for a successfully created user:

```
      ID: openldap_client_cn=joey,ou=people,dc=deploy-name,dc=local
Function: ldap.managed
  Result: True
 Comment: Successfully updated LDAP entries
 Started: 18:12:29.788665
Duration: 58.193 ms
 Changes:
          ----------
          cn=joey,ou=people,dc=deploy-name,dc=local:
              ----------
              new:
                  ----------
                  cn:
                      - joey
                  gecos:
                      - Joey Tribbiani
                  gidNumber:
                      - 20001
                  givenName:
                      - Joey
                  homeDirectory:
                      - /home/joey
                  loginShell:
                      - /bin/bash
                  mail:
                      - joey@domain.tld
                  objectClass:
                      - inetOrgPerson
                      - posixAccount
                      - shadowAccount
                      - top
                  sn:
                      - Tribbiani
                  uid:
                      - joey
```

```
                    uidNumber:
                        - 20060
                    userPassword:
                        - {CRYPT}$6$rounds=500000$KaJBYb3F8hYMv.UEHvc0...
                old:
                    None

Summary for cid01.domain.tld
------------
Succeeded: 7 (changed=1)
Failed:    0
------------
Total states run:     7
Total run time: 523.672 ms
```

Disable LDAP authentication on host OS

This section describes how to disable LDAP authentication on a host operating system.

To disable LDAP authentication:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In cluster/<cluster_name>/infra/auth/ldap.yml, disable LDAP:

   ```
   ldap:
     enabled: false
   ```

3. Enforce the linux.system update:

   ```
   salt '<target_node>*' state.sls linux.system
   ```

4. Clean up nodes:

   ```
   salt '<target_node>*' cmd.run 'export DEBIAN_FRONTEND=noninteractive; apt purge -y libnss-ldapd libpam-ldapd; sed -i "s/ ldap//g" /etc/nsswitch.conf'
   ```

©2025, Mirantis Inc.

Enable Gerrit audit logging

This section instructs you on how to enable the audit logging in Gerrit by configuring the httpd requests logger. Fluentd collects the files with the error logs automatically.

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

To set up audit logging in Gerrit:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. In the cicd/control/leader.yml file, add following parameters:

   ```
   parameters:
     _param:
       ...
       gerrit_extra_opts: "-Dlog4j.configuration=file:///var/gerrit/review_site/etc/log4j.properties"
       gerrit_http_request_log: 'True'
       ...
     linux:
       system:
         file:
           "/srv/volumes/gerrit/etc/log4j.properties":
             contents:
               - log4j.logger.httpd_log=INFO,httpd_log
               - log4j.appender.httpd_log=org.apache.log4j.ConsoleAppender
               - log4j.appender.httpd_log.layout=com.google.gerrit.pgm.http.jetty.HttpLogLayout
   ```

4. Refresh pillars:

   ```
   salt -C 'I@gerrit:client' saltutil.refresh_pillar
   ```

5. Create the log4j.properties file:

   ```
   salt -C 'I@gerrit:client' state.apply linux.system.file
   ```

6. Update the Gerrit service:

   ```
   salt -C 'I@gerrit:client' state.apply docker.client
   ```

Configure log rotation using logrotate

Note

This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

This section instructs you on how to configure log rotation for selected services using the logrotate utility.

The services that support the log rotation configuration include:

- **OpenStack services**

  Aodh, Barbican, Ceilometer, Cinder, Designate, Glance, Gnocchi, Heat, Keystone, Neutron, Nova, Octavia, Ironic
- **Other services**

  atop, Backupninja, Ceph, Elasticsearch, Galera (MySQL), GlusterFS, HAProxy, libvirt, MAAS, MongoDB, NGINX, Open vSwitch, PostgreSQL, RabbitMQ, Redis, Salt, Telegraf

MCP supports configuration of the rotation interval and number of rotations. Configuration of other logrotate options, postrotate and prerotate actions, and so on, are not supported.

To configure log rotation:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. Configure the interval and rotate parameters for the target service as required:

   - logrotate:interval

     Define the rotation time interval. Available values daily, weekly, monthly, and yearly.
   - logrotate:rotate

     Define the number of the rotated logs to be kept. The parameter expects the interger value.

   Use the Logrotate configuration table below to determine where to add the log rotation configuration.

Logrotate configuration

Service	Pillar path (target)	File path
Aodh	aodh:server	openstack/telemetry.yml
atop	linux:system:atop	The root file of the component [8]

Backupninja	backupninja:client	infra/backup/client_common.yml
Barbican	barbican:server	openstack/barbican.yml
Ceilometer server [1]	ceilometer:server	openstack/telemetry.yml
Ceilometer agent [1]	ceilometer:agent	openstack/compute/init.yml
Ceph	ceph:common	ceph/common.yml
Cinder controller [2]	cinder:controller	openstack/control.yml
Cinder volume [2]	cinder:volume	openstack/control.yml
Designate	designate:server	openstack/control.yml
Elasticsearch server	elasticsearch:server	stacklight/log.yml
Elasticsearch client	elasticsearch:client	stacklight/log.yml
Galera (MySQL) master	galera:master	openstack/database/master.yml
Galera (MySQL) slave	galera:slave	openstack/database/slave.yml
Glance	glance:server	openstack/control.yml
GlusterFS server [3]	glusterfs:server	The root file of the component [8]
GlusterFS client [3]	glusterfs:client	The root file of the component [8]
Gnocchi server	gnocchi:server	openstack/telemetry.yml
Gnocchi client	gnocchi:client	openstack/control/init.yml
HAProxy	haproxy:proxy	openstack/proxy.yml
Heat	heat:server	openstack/control.yml

Ironic Available since 2019.2.13	ironic:api	openstack/baremetal.yml
Keystone server	keystone:server	openstack/control.yml
Keystone client	keystone:client	openstack/control/init.yml
libvirt	nova:compute:libvirt [4]	openstack/compute/init.yml
MAAS	maas:region	infra/maas.yml
MongoDB	mongodb:server	stacklight/server.yml
Neutron server	neutron:server	openstack/control.yml
Neutron client	neutron:client	openstack/control/init.yml
Neutron gateway	neutron:gateway	openstack/gateway.yml
Neutron compute	neutron:compute	openstack/compute/init.yml
NGINX	nginx:server	openstack/proxy.yml, stacklight/proxy.yml
Nova controller	nova:controller	openstack/control.yml
Nova compute	nova:compute	openstack/compute/init.yml
Octavia manager [5]	octavia:manager	openstack/octavia_manager.yml
Octavia client [5]	octavia:client	openstack/control.yml
Open vSwitch	linux:network:openvswitch	infra/init.yml
PostgreSQL server	postgresql:server (maas:region) [6]	infra/config/postgresql.yml (infra/maas.yml)
PostgreSQL client	postgresql:client (maas:region) [6]	infra/config/postgresql.yml (infra/maas.yml)
RabbitMQ	rabbitmq:server	openstack/message_queue.yml
Redis	redis:server	openstack/telemetry.yml

Salt master [7]	salt:master	infra/config/init.yml
Salt minion [7]	salt:minion	The root file of the component [8]
Telegraf	telegraf:agent	infra/init.yml, stacklight/server.yml

1(1, 2) If Ceilometer server and agent are specified on the same node, the server configuration is prioritized.

2(1, 2) If Cinder controller and volume are specified on the same node, the controller configuration is prioritized.

3(1, 2) If GlusterFS server and client are specified on the same node, the server configuration is prioritized.

4 Use nova:compute:libvirt as pillar path, but only nova:compute as target.

5(1, 2) If Octavia manager and client are specified on the same node, the manager configuration is prioritized.

6(1, 2) PostgreSQL is the dependency of MAAS. Configure PostgreSQL from the MAAS pillar only if the service has been installed as a dependency without the postgresql pillar defined. If the postgresql pillar is defined, configure it instead.

7(1, 2) If the Salt Master and minion are specified on the same node, the master configuration is prioritized.

8(1, 2, 3, 4) Depending on the nodes where you want to change the configuration, select their components' root file. For example, infra/init.yml, openstack/control/init.yml, cicd/init.yml, and so on.

For example, to set log rotation for Aodh to keep logs for the last 4 weeks with the daily rotation interval, add the following configuration to cluster/<cluster_name>/openstack/telemetry.yml:

```
parameters:
  aodh:
    server:
      logrotate:
        interval: daily
        rotate: 28
```

4. Apply the logrotate state on the node with the target service:

```
salt -C 'I@<target>' saltutil.sync_all
salt -C 'I@<target>' state.sls logrotate
```

For example:

```
salt -C 'I@aodh:server' state.sls logrotate
```

Configure remote logging for auditd

> Note
>
> This feature is available starting from the MCP 2019.2.6 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section instructs you on how to configure remote logging for auditd.

To configure remote logging for auditd:

1. Log in to the Salt Master node.
2. In the classes/cluster/<cluster_name>/ directory, open one of the following files:
   - To configure one remote host for auditd for all nodes, use infra/init.yml.
   - To configure a remote host for a set of nodes, use a specific configuration file. For example, openstack/compute/init.yml for all OpenStack compute nodes.
3. Configure the remote host using the following exemplary pillar:

```
parameters:
  audisp:
    enabled: true
    remote:
      remote_server: <ip_address or hostname>
      port: <port>
      local_port: any
      transport: tcp
      ...
      key1: value1
```

4. Refresh pillars on the target nodes:

```
salt <nodes> saltutil.refresh_pillar
```

5. Apply the auditd.audisp state on the target nodes:

```
salt <nodes> state.apply auditd.audisp
```

Configure memory limits for the Redis server

> Note
>
> This feature is available starting from the MCP 2019.2.6 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section instructs you on how to configure the memory rules and limits for the Redis server.

To configure memory limits for the Redis server:

1. Log in to the Salt Master node.
2. In classes/cluster/<cluster_name>/openstack/telemetry.yml, specify the following parameters as required:

   ```
   parameters:
    redis:
     server:
      maxmemory: 1073741824 # 1GB
      maxmemory-policy: <memory-policy>
      maxmemory-samples: 3
   ```

   Supported values for the memory-policy parameter include:

   - volatile-lru - the service removes the key with expiration time set using the Least Recently Used (LRU) algorithm
   - allkeys-lru - the service removes any key according to the LRU algorithm
   - volatile-random - the service removes a random key with an expiration time set
   - allkeys-random - the service removes any random key
   - volatile-ttl - the service removes the key with the nearest expiration time (minor TTL)
   - noeviction - the service does not remove any key but returns an error on write operations
3. Apply the changes:

   ```
   salt -C 'I@redis:server' saltutil.refresh_pillar
   salt -C 'I@redis:server' state.apply redis.server
   ```

Configure multiple NTP servers

> Note
>
> This feature is available starting from the MCP 2019.2.6 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

MCP enables you to configure multiple Network Time Protocol (NTP) servers on new or existing MCP clusters to provide a more flexible and wide NTP support for clustered applications such as Ceph, Galera, and others.

For new MCP clusters, configure multiple NTP servers during the deployment model creation using the ntp_servers parameter passed to Cookiecutter in the following format:

```
server1.ntp.org,server2.ntp.org,server3.ntp.org
```

For details, see Networking deployment parameters in MCP Deployment Guide: Create a deployment metadata model.

For existing MCP clusters, configure multiple NTP servers by updating the NTP configuration for MAAS and all nodes of an MCP cluster.

To configure multiple NTP servers for MAAS:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. In infra/maas.yml, update the MAAS pillars using the example below:

   ```
   parameters:
     maas:
       region:
         ...
         ntp:
           server1:
             enabled: True
             host: ntp.example1.org
           server2:
             enabled: True
             host: ntp.example2.o
   ```

4. Update the MAAS configuration:

   ```
   salt-call saltutil.refresh_pillar
   salt-call state.apply maas.region
   ```

To configure multiple NTP servers for all nodes of an MCP cluster:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. In infra/init.yml, update the MAAS pillars using the example below:

   ```
   parameters:
     ntp:
       client:
       enabled: true
       stratum:
         primary:
           server: primary.ntp.org
         secondary: # if exist
           server: secondary.ntp.org
         srv_3: # if exist
           server: srv_3.ntp.org
   ```

4. Update the NTP configuration:

   ```
   salt '*' saltutil.refresh_pillar
   salt '*' state.apply ntp.client
   ```

Configure kernel crash dumping

> Note
>
> This feature is available starting from the MCP 2019.2.16 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

The Kernel Crash Dump (kdump) mechanism allows you to save the system memory events for later analysis. This section instructs you on how to configure kernel crash dumping for all or particular nodes. The dump will be saved in the /var/crash directory. For more details, see Kernel Crash Dump.

To configure kernel crash dumping:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. In <cluster_name>/infra/init.yml, add the following variable:

   ```
   parameters:
     linux:
       system:
         kernel_crash_dump:
           enabled: true
   ```

4. Apply the changes for the required nodes or for all nodes. To apply the changes for all nodes:

   ```
   salt -C '*' state.sls linux.system.kernel_crash_dump
   ```

5. Manually reboot the nodes for which you have enabled kernel crash dumping. For example, to reboot the OpenStack compute nodes:

   ```
   salt -C 'cmp1*' system.reboot --async
   ```

Once done, you can view the dump in the /var/crash directory.

OpenStack operations

This section includes all OpenStack-related Day-2 operations such as reprovisioning of OpenStack controller and compute nodes, preparing the Ironic service to provision cloud workloads on bare metal nodes, and others.

Manage Virtualized Control Plane

This section describes operations with the MCP Virtualized Control Plane (VCP).

Add a controller node

If you need to expand the size of VCP to handle a bigger data plane, you can add more controller nodes to your cloud environment. This section instructs on how to add a KVM node and an OpenStack controller VM to an existing environment.

The same procedure can be applied for scaling the messaging, database, and any other services.

Additional parameters will have to be added before the deployment.

To add a controller node:

1. Add a physical node using MAAS as described in the MCP Deployment Guide: Provision physical nodes using MAAS.
2. Log in to the Salt Master node.
3. In the /classes/cluster/<cluster_name>/infra/init.yml file, define the basic parameters for the new KVM node:

```
parameters:
  _param:
    infra_kvm_node04_address: <IP ADDRESS ON CONTROL NETWORK>
    infra_kvm_node04_deploy_address: <IP ADDRESS ON DEPLOY NETWORK>
    infra_kvm_node04_storage_address: ${_param:infra_kvm_node04_address}
    infra_kvm_node04_public_address: ${_param:infra_kvm_node04_address}
    infra_kvm_node04_hostname: kvm<NUM>
    glusterfs_node04_address: ${_param:infra_kvm_node04_address}
linux:
  network:
    host:
      kvm04:
        address: ${_param:infra_kvm_node04_address}
        names:
        - ${_param:infra_kvm_node04_hostname}
        - ${_param:infra_kvm_node04_hostname}.${_param:cluster_domain}
```

4. In the /classes/cluster/<cluster_name>/openstack/init.yml file, define the basic parameters for the new OpenStack controller node.

```
openstack_control_node<NUM>_address: <IP_ADDRESS_ON_CONTROL_NETWORK>
openstack_control_node<NUM>_hostname: <HOSTNAME>
openstack_database_node<NUM>_address: <DB_IP_ADDRESS>
openstack_database_node<NUM>_hostname: <DB_HOSTNAME>
openstack_message_queue_node<NUM>_address: <IP_ADDRESS_OF_MESSAGE_QUEUE>
openstack_message_queue_node<NUM>_hostname: <HOSTNAME_OF_MESSAGE_QUEUE>
```

   Example of configuration:

```
kvm04_control_ip: 10.167.4.244
kvm04_deploy_ip: 10.167.5.244
kvm04_name: kvm04
openstack_control_node04_address: 10.167.4.14
openstack_control_node04_hostname: ctl04
```

5. In the /classes/cluster/<cluster_name>/infra/config.yml file, define the configuration parameters for the KVM and OpenStack controller nodes. For example:

```
reclass:
  storage:
    node:
      infra_kvm_node04:
        name: ${_param:infra_kvm_node04_hostname}
        domain: ${_param:cluster_domain}
        classes:
        - cluster.${_param:cluster_name}.infra.kvm
        params:
          keepalived_vip_priority: 103
          salt_master_host: ${_param:reclass_config_master}
          linux_system_codename: xenial
          single_address: ${_param:infra_kvm_node04_address}
          deploy_address: ${_param:infra_kvm_node04_deploy_address}
          public_address: ${_param:infra_kvm_node04_public_address}
          storage_address: ${_param:infra_kvm_node04_storage_address}
      openstack_control_node04:
        name: ${_param:openstack_control_node04_hostname}
        domain: ${_param:cluster_domain}
        classes:
        - cluster.${_param:cluster_name}.openstack.control
        params:
          salt_master_host: ${_param:reclass_config_master}
          linux_system_codename: xenial
          single_address: ${_param:openstack_control_node04_address}
          keepalived_vip_priority: 104
          opencontrail_database_id: 4
          rabbitmq_cluster_role: slave
```

6. In the /classes/cluster/<cluster_name>/infra/kvm.yml file, define new brick for GlusterFS on all KVM nodes and salt:control which later spawns the OpenStack controller node. For example:

```
_param:
  cluster_node04_address: ${_param:infra_kvm_node04_address}
glusterfs:
  server:
    volumes:
```

```
      glance:
        replica: 4
        bricks:
          - ${_param:cluster_node04_address}:/srv/glusterfs/glance
      keystone-keys:
        replica: 4
        bricks:
          - ${_param:cluster_node04_address}:/srv/glusterfs/keystone-keys
      keystone-credential-keys:
        replica: 4
        bricks:
          - ${_param:cluster_node04_address}:/srv/glusterfs/keystone-credential-keys
salt:
  control:
    cluster:
      internal:
        domain: ${_param:cluster_domain}
        engine: virt
        node:
          ctl04:
            name: ${_param:openstack_control_node04_hostname}
            provider: ${_param:infra_kvm_node04_hostname}.${_param:cluster_domain}
            image: ${_param:salt_control_xenial_image}
            size: openstack.control
```

7. In the /classes/cluster/<cluster_name>/openstack/control.yml file, add the OpenStack controller node into existing services such as HAProxy, and others, depending on your environment configuration.

   Example of adding an HAProxy host for Glance:

```
_param:
  cluster_node04_hostname: ${_param:openstack_control_node04_hostname}
  cluster_node04_address: ${_param:openstack_control_node04_address}
haproxy:
  proxy:
    listen:
      glance_api:
        servers:
        - name: ${_param:cluster_node04_hostname}
          host: ${_param:cluster_node04_address}
          port: 9292
          params: check inter 10s fastinter 2s downinter 3s rise 3 fall 3
      glance_registry_api:
        servers:
        - name: ${_param:cluster_node04_hostname}
          host: ${_param:cluster_node04_address}
```

```
        port: 9191
        params: check
```

8. Refresh the deployed pillar data by applying the reclass.storage state:

```
salt '*cfg*' state.sls reclass.storage
```

9. Verify that the target node has connectivity with the Salt Master node:

```
salt '*kvm<NUM>*' test.ping
```

10. Verify that the Salt Minion nodes are synchronized:

```
salt '*' saltutil.sync_all
```

11. On the Salt Master node, apply the Salt linux state for the added node:

```
salt -C 'I@salt:control' state.sls linux
```

12. On the added node, verify that salt-common and salt-minion have the 2017.7 version.

```
apt-cache policy salt-common
apt-cache policy salt-minion
```

> Note
>
> If the commands above show a different version, follow the MCP Deployment guide: Install the correct versions of salt-common and salt-minion.

13. Perform the initial Salt configuration:

```
salt -C 'I@salt:control' state.sls salt.minion
```

14. Set up the network interfaces and the SSH access:

```
salt -C 'I@salt:control' state.sls linux.system.user,openssh,linux.network,ntp
```

15. Reboot the KVM node:

```
salt '*kvm<NUM>*' cmd.run 'reboot'
```

16. On the Salt Master node, apply the libvirt state:

```
salt -C 'I@salt:control' state.sls libvirt
```

17. On the Salt Master node, create a controller VM for the added physical node:

```
salt -C 'I@salt:control' state.sls salt.control
```

> Note
>
> Salt virt takes the name of a virtual machine and registers the virtual machine on the Salt Master node.
>
> Once created, the instance picks up an IP address from the MAAS DHCP service and the key will be seen as accepted on the Salt Master node.

18. Verify that the controller VM has connectivity with the Salt Master node:

```
salt 'ctl<NUM>*' test.ping
```

19. Verify that the Salt Minion nodes are synchronized:

```
salt '*' saltutil.sync_all
```

20. Apply the Salt highstate for the controller VM:

```
salt -C 'I@salt:control' state.highstate
```

21. Verify that the added controller node is registered on the Salt Master node:

```
salt-key
```

22. To reconfigure VCP VMs, run the openstack-deploy Jenkins pipeline with all necessary install parmeters as described in MCP Deployment guide: Deploy an OpenStack environment.

Replace a KVM node

If a KVM node hosting the Virtualized Control Plane has failed and recovery is not possible, you can recreate the KVM node from scratch with all VCP VMs that were hosted on the old KVM node. The replaced KVM node will be assigned the same IP addresses as the failed KVM node.

Replace a failed KVM node

This section describes how to recreate a failed KVM node with all VCP VMs that were hosted on the old KVM node. The replaced KVM node will be assigned the same IP addresses as the failed KVM node.

To replace a failed KVM node:

1. Log in to the Salt Master node.
2. Copy and keep the hostname and GlusterFS UUID of the old KVM node.

   To obtain the UUIDs of all peers in the cluster:

   ```
   salt '*kvm<NUM>*' cmd.run "gluster peer status"
   ```

   > Note
   >
   > Run the command above from a different KVM node of the same cluster since the command outputs other peers only.

3. Verify that the KVM node is not registered in salt-key. If the node is present, remove it:

   ```
   salt-key | grep kvm<NUM>
   salt-key -d kvm<NUM>.domain_name
   ```

4. Remove the salt-key records for all VMs originally running on the failed KVM node:

   ```
   salt-key -d <kvm_node_name><NUM>.domain_name
   ```

   > Note
   >
   > You can list all VMs running on the KVM node using the salt '*kvm<NUM>*' cmd.run 'virsh list --all' command. Alternatively, obtain the list of VMs from cluster/infra/kvm.yml.

5. Add or reprovision a physical node using MAAS as described in the MCP Deployment Guide: Provision physical nodes using MAAS.
6. Verify that the new node has been registered on the Salt Master node successfully:

   ```
   salt-key | grep kvm
   ```

> Note
>
> If the new node is not available in the list, wait some time until the node becomes available or use the IPMI console to troubleshoot the node.

7. Verify that the target node has connectivity with the Salt Master node:

   ```
   salt '*kvm<NUM>*' test.ping
   ```

8. Verify that salt-common and salt-minion have the same version for the new node as the rest of the cluster.

   ```
   salt -t 10 'kvm*' cmd.run 'dpkg -l |grep "salt-minion\|salt-common"'
   ```

   > Note
   >
   > If the command above shows a different version for the new node, follow the steps described in Install the correct versions of salt-common and salt-minion.

9. Verify that the Salt Minion nodes are synchronized:

   ```
   salt '*' saltutil.refresh_pillar
   ```

10. Apply the linux state for the added node:

    ```
    salt '*kvm<NUM>*' state.sls linux
    ```

11. Perform the initial Salt configuration:

    1. Run the following commands:

       ```
       salt '*kvm<NUM>*' cmd.run "touch /run/is_rebooted"
       salt '*kvm<NUM>*' cmd.run 'reboot'
       ```

       Wait some time before the node is rebooted.

    2. Verify that the node is rebooted:

       ```
       salt '*kvm<NUM>*' cmd.run 'if [ -f "/run/is_rebooted" ];then echo \
       "Has not been rebooted!";else echo "Rebooted";fi'
       ```

> Note
>
> The node must be in the Rebooted state.

12. Set up the network interfaces and the SSH access:

    ```
    salt -C 'I@salt:control' state.sls linux.system.user,openssh,linux.network,ntp
    ```

13. Apply the libvirt state for the added node:

    ```
    salt '*kvm<NUM>*' state.sls libvirt
    ```

14. Recreate the original VCP VMs on the new node:

    ```
    salt '*kvm<NUM>*' state.sls salt.control
    ```

    > Note
    >
    > Salt virt takes the name of a VM and registers it on the Salt Master node.
    >
    > Once created, the instance picks up an IP address from the MAAS DHCP service and the key will be seen as accepted on the Salt Master node.

15. Verify that the added VCP VMs are registered on the Salt Master node:

    ```
    salt-key
    ```

16. Verify that the Salt Minion nodes are synchronized:

    ```
    salt '*' saltutil.sync_all
    ```

17. Apply the highstate for the VCP VMs:

    ```
    salt '*kvm<NUM>*' state.highstate
    ```

18. Verify whether the new node has correct IP address and proceed to restore GlusterFS configuration as described in Recover GlusterFS on a replaced KVM node.

Recover GlusterFS on a replaced KVM node

After you replace a KVM node as described in Replace a failed KVM node, if your new KVM node has the same IP address, proceed with recovering GlusterFS as described below.

To recover GlusterFS on a replaced KVM node:

1. Log in to the Salt Master node.
2. Define the IP address of the failed and any working KVM node that is running the GlusterFS cluster services. For example:

   ```
   FAILED_NODE_IP=<IP_of_failed_kvm_node>
   WORKING_NODE_IP=<IP_of_working_kvm_node>
   ```

3. If the failed node has been recovered with the old disk and GlusterFS installed:

   1. Remove the /var/lib/glusterd directory:

      ```
      salt -S $FAILED_NODE_IP file.remove '/var/lib/glusterd'
      ```

   2. Restart glusterfs-server:

      ```
      salt -S $FAILED_NODE_IP service.restart glusterfs-server
      ```

4. Configure glusterfs-server on the failed node:

   ```
   salt -S $FAILED_NODE_IP state.apply glusterfs.server.service
   ```

5. Remove the failed node from the GlusterFS cluster:

   ```
   salt -S $WORKING_NODE_IP cmd.run "gluster peer detach $FAILED_NODE_IP"
   ```

6. Re-add the failed node to the GlusterFS cluster with a new ID:

   ```
   salt -S $WORKING_NODE_IP cmd.run "gluster peer probe $FAILED_NODE_IP"
   ```

7. Finalize the configuration of the failed node:

   ```
   salt -S $FAILED_NODE_IP state.apply
   ```

8. Set the correct trusted.glusterfs.volume-id attribute in the GlusterFS directories on the failed node:

   ```
   for vol in $(salt --out=txt -S $WORKING_NODE_IP cmd.run 'for dir in /srv/glusterfs/*; \
   do echo -n "${dir}@0x"; getfattr  -n trusted.glusterfs.volume-id \
   --only-values --absolute-names $dir | xxd -g0 -p;done' | awk -F: '{print $2}'); \
   do VOL_PATH=$(echo $vol| cut -d@ -f1); TRUST_ID=$(echo $vol | cut -d@ -f2); \
   ```

```
salt -S $FAILED_NODE_IP cmd.run "setfattr -n trusted.glusterfs.volume-id -v $TRUST_ID $VOL_PATH"; \
done
```

9. Restart glusterfs-server:

```
salt -S $FAILED_NODE_IP service.restart glusterfs-server
```

Move a VCP node to another host

To ensure success during moving the VCP VMs running in the cloud environment for specific services, take a single VM at a time, stop it, move the disk to another host, and start the VM again on the new host machine. The services running on the VM should remain running during the whole process due to high availability ensured by Keepalived and HAProxy.

To move a VCP node to another host:

1. To synchronize your deployment model with the new setup, update the /classes/cluster/<cluster_name>/infra/kvm.yml file:

   ```
   salt:
     control:
       cluster:
         internal:
           node:
             <nodename>:
               name: <nodename>
               provider: ${_param:infra_kvm_node03_hostname}.${_param:cluster_domain}
               # replace 'infra_kvm_node03_hostname' param with the new kvm nodename provider
   ```

2. Apply the salt.control state on the new KVM node:

   ```
   salt-call state.sls salt.control
   ```

3. Destroy the newly spawned VM on the new KVM node:

   ```
   virsh list
   virsh destroy <nodename><nodenum>.<domainname>
   ```

4. Log in to the KVM node originally hosting the VM.
5. Stop the VM:

   ```
   virsh list
   virsh destroy <nodename><nodenum>.<domainname>
   ```

6. Move the disk to the new KVM node using, for exmaple, the scp utility, replacing the empty disk spawned by the salt.control state with the correct one:

   ```
   scp /var/lib/libvirt/images/<nodename><nodenum>.<domainname>/system.qcow2 \
   <diff_kvm_nodename>:/var/lib/libvirt/images/<nodename><nodenum>.<domainname>/system.qcow2
   ```

7. Start the VM on the new KVM host:

   ```
   virsh start <nodename><nodenum>.<domainname>
   ```

8. Verify that the services on the moved VM work correctly.

9. Log in to the KVM node that was hosting the VM originally and undefine it:

```
virsh list --all
virsh undefine <nodename><nodenum>.<domainname>
```

Enable host passthrough for VCP

> Note
>
> This feature is available starting from the MCP 2019.2.16 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes how to enable the host-passthrough CPU mode that can enhance performance of the MCP Virtualized Control Plane (VCP). For details, see libvirt documentation: CPU model and topology.

> Warning
>
> Prior to enabling the host passthrough, run the following command to verify that it is applicable to your deployment:
>
> ```
> salt -C "I@salt:control" cmd.run "virsh list | tail -n +3 | awk '{print \$1}' | xargs -I{} virsh dumpxml {} | grep cpu_mode"
> ```
>
> If the output is empty, proceed to enabling host passthrough. Otherwise, first contact Mirantis support.

To enable host passthrough:

1. Log in to a KVM node.
2. Obtain the list of running VMs:

   ```
   virsh list
   ```

   Example of system response:

   ```
   Id  Name                                 State
   ------------------------------------------------
    1  msg01.bm-cicd-queens-ovs-maas.local running
    2  rgw01.bm-cicd-queens-ovs-maas.local running
    3  dbs01.bm-cicd-queens-ovs-maas.local running
    4  bmt01.bm-cicd-queens-ovs-maas.local running
    5  kmn01.bm-cicd-queens-ovs-maas.local running
    6  cid01.bm-cicd-queens-ovs-maas.local running
    7  cmn01.bm-cicd-queens-ovs-maas.local running
    8  ctl01.bm-cicd-queens-ovs-maas.local running
   ```

3. Edit the configuration of each VM using the virsh edit %VM_NAME% command. Add the following lines to the XML configuration file:

```
<cpu mode='host-passthrough'>
  <cache mode='passthrough'/>
</cpu>
```

For example:

```
<domain type='kvm'>
  <name>msg01.bm-cicd-queens-ovs-maas.local</name>
  <uuid>81e18795-cf2f-4ffc-ac90-9fa0a3596ffb</uuid>
  <memory unit='KiB'>67108864</memory>
  <currentMemory unit='KiB'>67108864</currentMemory>
  <vcpu placement='static'>16</vcpu>
  <cpu mode='host-passthrough'>
    <cache mode='passthrough'/>
  </cpu>
  <os>
    <type arch='x86_64' machine='pc-i440fx-bionic'>hvm</type>
    <boot dev='hd'/>
  </os>
.......
```

4. Perform the steps 1-3 on the remaning kvm nodes one by one.
5. Log in to the Salt Master node.
6. Reboot the VCP nodes as described in Scheduled maintenance with a planned power outage using the salt 'nodename01*' system.reboot command. Do not reboot the kvm, apt, and cmp nodes.

> Warning
>
> Reboot nodes one by one instead of rebooting all nodes of the same role at a time. Wait for 10 minutes between each reboot.

Manage compute nodes

This section provides instructions on how to manage the compute nodes in your cloud environment.

Add a compute node

This section describes how to add a new compute node to an existing OpenStack environment.

To add a compute node:

1. Add a physical node using MAAS as described in the MCP Deployment Guide: Provision physical nodes using MAAS.
2. Verify that the compute node is defined in /classes/cluster/<cluster_name>/infra/config.yml.

   > Note
   >
   > Create as many hosts as you have compute nodes in your environment within this file.

   > Note
   >
   > Verify that the count parameter is increased by the number of compute nodes being added.

   Configuration example if the dynamic compute host generation is used:

   ```
   reclass:
     storage:
       node:
         openstack_compute_rack01:
           name: ${_param:openstack_compute_rack01_hostname}<<count>>
           domain: ${_param:cluster_domain}
           classes:
           - cluster.${_param:cluster_name}.openstack.compute
           repeat:
             count: 20
             start: 1
             digits: 3
             params:
               single_address:
                 value: 172.16.47.<<count>>
                 start: 101
               tenant_address:
                 value: 172.16.47.<<count>>
                 start: 101
           params:
   ```

```
        salt_master_host: ${_param:reclass_config_master}
        linux_system_codename: xenial
```

Configuration example if the static compute host generation is used:

```
reclass:
  storage:
    node:
      openstack_compute_node01:
        name: cmp01
        domain: ${_param:cluster_domain}
        classes:
        - cluster.${_param:cluster_name}.openstack.compute
        params:
          salt_master_host: ${_param:reclass_config_master}:
          linux_system_codename: xenial
          single_address: 10.0.0.101
          deploy_address: 10.0.1.101
          tenant_address: 10.0.2.101
```

3. Define the cmp<NUM> control address and hostname in the <cluster>/openstack/init.yml file:

```
_param:
  openstack_compute_node<NUM>_address: <control_network_IP>
  openstack_compute_node<NUM>_hostname: cmp<NUM>

linux:
  network:
    host:
      cmp<NUM>:
        address: ${_param:openstack_compute_node<NUM>_address}
        names:
        - ${_param:openstack_compute_node<NUM>_hostname}
        - ${_param:openstack_compute_node<NUM>_hostname}.${_param:cluster_domain}
```

4. Apply the reclass.storage state on the Salt Master node to generate node definitions:

```
salt '*cfg*' state.sls reclass.storage
```

5. Verify that the target nodes have connectivity with the Salt Master node:

```
salt '*cmp<NUM>*' test.ping
```

6. Apply the following states:

```
salt 'cfg*' state.sls salt.minion.ca
salt '*cmp<NUM>*' state.sls salt.minion.cert
```

7. Deploy a new compute node as described in MCP Deployment Guide: Deploy physical servers.

Caution!

Do not use compounds for this step, since it will affect already running physical servers and reboot them. Use the Salt minion IDs instead of compounds before running the pipelines or deploying physical servers manually.

Incorrect:

```
salt -C 'I@salt:control or I@nova:compute or I@neutron:gateway' \
 cmd.run "touch /run/is_rebooted"
salt --async -C 'I@nova:compute' cmd.run 'salt-call state.sls \
 linux.system.user,openssh,linux.network;reboot'
```

Correct:

```
salt cmp<NUM> cmd.run "touch /run/is_rebooted"
salt --async cmp<NUM> cmd.run 'salt-call state.sls \
 linux.system.user,openssh,linux.network;reboot'
```

Note

We recommend that you rerun the Jenkins Deploy - OpenStack pipeline that runs on the Salt Master node with the same parameters as you have set initially during your environment deployment. This guarantees that your compute node will be properly set up and added.

Reprovision a compute node

Provisioning of compute nodes is relatively straightforward as you can run all states at once. Though, you need to run and reboot it multiple times for network configuration changes to take effect.

> Note
>
> Multiple reboots are needed because the ordering of dependencies is not yet orchestrated.

To reprovision a compute node:

1. Verify that the name of the cmp node is not registered in salt-key on the Salt Master node:

   ```
   salt-key | grep 'cmp*'
   ```

   If the node is shown in the above command output, remove it:

   ```
   salt-key -d cmp<NUM>.domain_name
   ```

2. Add a physical node using MAAS as described in the MCP Deployment Guide: Provision physical nodes using MAAS.
3. Verify that the required nodes are defined in /classes/cluster/<cluster_name>/infra/config.yml.

   > Note
   >
   > Create as many hosts as you have compute nodes in your environment within this file.

   Configuration example if the dynamic compute host generation is used:

   ```
   reclass:
     storage:
       node:
         openstack_compute_rack01:
           name: ${_param:openstack_compute_rack01_hostname}<<count>>
           domain: ${_param:cluster_domain}
           classes:
           - cluster.${_param:cluster_name}.openstack.compute
           repeat:
             count: 20
   ```

©2025, Mirantis Inc.

```
        start: 1
        digits: 3
        params:
          single_address:
            value: 172.16.47.<<count>>
            start: 101
          tenant_address:
            value: 172.16.47.<<count>>
            start: 101
      params:
        salt_master_host: ${_param:reclass_config_master}
        linux_system_codename: xenial
```

Configuration example if the static compute host generation is used:

```
reclass:
  storage:
    node:
      openstack_compute_node01:
        name: cmp01
        domain: ${_param:cluster_domain}
        classes:
        - cluster.${_param:cluster_name}.openstack.compute
        params:
          salt_master_host: ${_param:reclass_config_master}
          linux_system_codename: xenial
          single_address: 10.0.0.101
          deploy_address: 10.0.1.101
          tenant_address: 10.0.2.101
```

4. Apply the reclass.storage state on the Salt Master node to generate node definitions:

```
salt '*cfg*' state.sls reclass.storage
```

5. Verify that the target nodes have connectivity with the Salt Master node:

```
salt '*cmp<NUM>*' test.ping
```

6. Verify that the Salt Minion nodes are synchronized:

```
salt '*cmp<NUM>*' saltutil.sync_all
```

7. Apply the Salt highstate on the compute node(s):

```
salt '*cmp<NUM>*' state.highstate
```

> Note
>
> Failures may occur during the first run of highstate. Rerun the state until it is successfully applied.

8. Reboot the compute node(s) to apply network configuration changes.
9. Reapply the Salt highstate on the node(s):

```
salt '*cmp<NUM>*' state.highstate
```

10. Provision the vRouter on the compute node using CLI or the Contrail web UI. Example of the CLI command:

```
salt '*cmp<NUM>*' cmd.run '/usr/share/contrail-utils/provision_vrouter.py \
    --host_name <CMP_HOSTNAME> --host_ip <CMP_IP_ADDRESS> --api_server_ip <CONTRAIL_VIP> \
    --oper add --admin_user admin --admin_password <PASSWORD> \
    --admin_tenant_name admin --openstack_ip <OPENSTACK_VIP>'
```

> Note
>
> - To obtain <CONTRAIL_VIP>, run salt-call pillar.get _param:keepalived_vip_address on any ntw node.
> - To obtain <OPENSTACK_VIP>, run salt-call pillar.get _param:keepalived_vip_address on any ctl node.

Remove a compute node

This section instructs you on how to safely remove a compute node from your OpenStack environment.

To remove a compute node:

1. Stop and disable the salt-minion service on the compute node you want to remove:

```
systemctl stop salt-minion
systemctl disable salt-minion
```

2. Verify that the name of the node is not registered in salt-key on the Salt Master node. If the node is present, remove it:

```
salt-key | grep cmp<NUM>
salt-key -d cmp<NUM>.domain_name
```

3. Log in to an OpenStack controller node.
4. Source the OpenStack RC file to set the required environment variables for the OpenStack command-line clients:

```
source keystonerv3
```

5. Disable the nova-compute service on the target compute node:

```
openstack compute service set --disable <cmp_host_name> nova-compute
```

6. Verify that Nova does not schedule new instances on the target compute node by viewing the output of the following command:

```
openstack compute service list
```

   The command output should display the disabled status for the nova-compute service running on the target compute node.
7. Migrate your instances using the openstack server migrate command. You can perform live or cold migration.
8. Log in to the target compute node.
9. Stop the nova-compute service:

```
systemctl disable nova-compute
systemctl stop nova-compute
```

10. Log in to the OpenStack controller node.

11. Obtain the ID of the compute service to delete:

    ```
    openstack compute service list
    ```

12. Delete the compute service substituting service_id with the value obtained in the previous step:

    ```
    openstack compute service delete <service_id>
    ```

13. Select from the following options:

    - For the deployments with OpenContrail:

        1. Log in to the target compute node.
        2. Stop the supervisor-vrouter service:

            ```
            service supervisor-vrouter disable
            service supervisor-vrouter stop
            ```

        3. Log in to the OpenContrail UI.
        4. Navigate to Configure > infrastructure > Virtual Routers.
        5. Select the target compute node.
        6. Click Delete.

    - For the deployments with OVS:

        1. Stop the neutron-openvswitch-agent service:

            ```
            systemctl disable neutron-openvswitch-agent.service
            systemctl stop neutron-openvswitch-agent.service
            ```

        2. Obtain the ID of the target compute node agent:

            ```
            openstack network agent list
            ```

        3. Delete the network agent substituting cmp_agent_id with the value obtained in the previous step:

            ```
            openstack network agent delete <cmp_agent_id>
            ```

14. If you plan to replace the removed compute node with a new compute node with the same hostname, you need to manually clean up the resource provider record from the placement service using the curl tool:

    1. Log in to an OpenStack controller node.
    2. Obtain the token ID from the openstack token issue command output. For example:

```
openstack token issue
+------------+------------------------------------+
| Field      | Value                              |
+------------+------------------------------------+
| expires    | 2018-06-22T10:30:17+0000           |
| id         | gAAAAABbLMGpVq2Gjwtc5Qqmp...       |
| project_id | 6395787cdff649cdbb67da7e692cc592   |
| user_id    | 2288ac845d5a4e478ffdc7153e389310   |
+------------+------------------------------------+
```

3. Obtain the resource provider UUID of the target compute node:

```
curl -i -X GET <placement-endpoint-address>/resource_providers?name=<target-compute-host-name> -H \
'content-type: application/json' -H 'X-Auth-Token: <token>'
```

Susbtitute the following parameters as required:

- placement-endpoint-address

  The placement endpoint can be obtained from the openstack catalog list command output. A placement endpoint includes the scheme, endpoint address, and port, for example, http://10.11.0.10:8778. Depending on the deployment, you may need to specify the https scheme rather than http.

- target-compute-host-name

  The hostname of the compute node you are removing. For the correct hostname format to pass, see the Hypervisor Hostname column in the openstack hypervisor list command output.

- token

  The token id value obtained in the previous step.

Example of system response:


```
{
 "resource_providers": [
  {
   "generation": 1,
   "uuid": "08090377-965f-4ad8-9a1b-87f8e8153896",
   "links": [
    {
     "href": "/resource_providers/08090377-965f-4ad8-9a1b-87f8e8153896",
     "rel": "self"
    },
    {
     "href": "/resource_providers/08090377-965f-4ad8-9a1b-87f8e8153896/aggregates",
     "rel": "aggregates"
    },
    {
```

```
        "href": "/resource_providers/08090377-965f-4ad8-9a1b-87f8e8153896/inventories",
        "rel": "inventories"
      },
      {
        "href": "/resource_providers/08090377-965f-4ad8-9a1b-87f8e8153896/usages",
        "rel": "usages"
      }
    ],
    "name": "<compute-host-name>"
  }
 ]
}
```

4. Delete the resource provider record from the placement service substituting placement-endpoint-address, target-compute-node-uuid, and token with the values obtained in the previous steps:

   ```
   curl -i -X DELETE <placement-endpoint-address>/resource_providers/<target-compute-node-uuid> -H \
   'content-type: application/json' -H 'X-Auth-Token: <token>'
   ```

15. Log in to the Salt Master node.

16. Remove the compute node definition from the model in infra/config.yml under the reclass:storage:node pillar.

17. Remove the generated file for the removed compute node under /srv/salt/reclass/nodes/_generated.

18. Remove the compute node from StackLight LMA:

    1. Update and clear the Salt mine:

       ```
       salt -C 'I@salt:minion' state.sls salt.minion.grains
       salt -C 'I@salt:minion' saltutil.refresh_modules
       salt -C 'I@salt:minion' mine.update clear=true
       ```

    2. Refresh the targets and alerts:

       ```
       salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus -b 1
       ```

Reboot a compute node

This section instructs you on how to reboot an OpenStack compute node for a planned maintenance.

To reboot an OpenStack compute node:

1. Log in to an OpenStack controller node.
2. Disable scheduling of new VMs to the node. Optionally provide a reason comment:

   ```
   openstack compute service set --disable --disable-reason \
   maintenance <compute_node_hostname> nova-compute
   ```

3. Migrate workloads from the OpenStack compute node:

   ```
   nova host-evacuate-live <compute_node_hostname>
   ```

4. Log in to an OpenStack compute node.
5. Stop the nova-compute service:

   ```
   service nova-compute stop
   ```

6. Shut down the OpenStack compute node, perform the maintenance, and turn the node back on.
7. Verify that the nova-compute service is up and running:

   ```
   service nova-compute status
   ```

8. Perform the following steps from the OpenStack controller node:

   1. Enable scheduling of VMs to the node:

      ```
      openstack compute service set --enable <compute_node_hostname> nova-compute
      ```

   2. Verify that the nova-compute service and neutron agents are running on the node:

      ```
      openstack network agent list --host <compute_node_hostname>
      openstack compute service list --host <compute_node_hostname>
      ```

      The OpeStack compute service state must be up. The Neutron agent service state must be UP and the alive column must include :-).

      Examples of a positive system response:

      ```
      +----+--------------+------+------+---------+-------+----------------------------+
      | ID | Binary       | Host | Zone | Status  | State | Updated At                 |
      +----+--------------+------+------+---------+-------+----------------------------+
      ```

```
| 70 | nova-compute | cmp1 | nova | enabled | up    | 2020-09-17T08:51:07.000000 |
+----+--------------+------+------+---------+-------+----------------------------+
```

```
+----------+--------------------+------+-------------------+-------+-------+---------------------------+
| ID       | Agent Type         | Host | Availability Zone | Alive | State | Binary                    |
+----------+--------------------+------+-------------------+-------+-------+---------------------------+
| e4256d73 | Open vSwitch agent | cmp1 | None              | :-)   | UP    | neutron-openvswitch-agent |
+----------+--------------------+------+-------------------+-------+-------+---------------------------+
```

3. Optional. Migrate the instances back to their original OpenStack compute node.

Manage gateway nodes

This section describes how to manage tenant network gateway nodes that provide access to an external network for the environments configured with Neutron OVS as a networking solution.

Add a gateway node

The gateway nodes are hardware nodes that provide gateways and routers to the OVS-based tenant networks using network virtualization functions. Standard cloud configuration includes three gateway nodes. Though, you can scale the networking thoughput by adding more gateway servers.

This section explains how to increase the number of the gateway nodes in your cloud environment.

To add a gateway node:

1. Add a physical node using MAAS as described in the MCP Deployment Guide: Provision physical nodes using MAAS.
2. Define the gateway node in /classes/cluster/<cluster_name>/infra/config.yml. For example:

```
parameters:
  _param:
    openstack_gateway_node03_hostname: gtw03
    openstack_gateway_node03_tenant_address: <IP_of_gtw_node_tenant_address>
  reclass:
    storage:
      node:
        openstack_gateway_node03:
          name: ${_param:openstack_gateway_node03_hostname}
          domain: ${_param:cluster_domain}
          classes:
          - cluster.${_param:cluster_name}.openstack.gateway
          params:
            salt_master_host: ${_param:reclass_config_master}
            linux_system_codename: ${_param:linux_system_codename}
            single_address: ${_param:openstack_gateway_node03_address}
            tenant_address: ${_param:openstack_gateway_node03_tenant_address}
```

3. On the Salt Master node, generate node definitions by applying the reclass.storage state:

```
salt '*cfg*' state.sls reclass.storage
```

4. Verify that the target nodes have connectivity with the Salt Master node:

```
salt '*gtw<NUM>*' test.ping
```

5. Verify that the Salt Minion nodes are synchronized:

```
salt '*gtw<NUM>*' saltutil.sync_all
```

6. On the added node, verify that salt-common and salt-minion have the 2017.7 version.

```
apt-cache policy salt-common
apt-cache policy salt-minion
```

Note

If the commands above show a different version, follow the MCP Deployment guide: Install the correct versions of salt-common and salt-minion.

7. Perform the initial Salt configuration:

```
salt '*gtw<NUM>*' state.sls salt.minion
```

8. Set up the network interfaces and the SSH access:

```
salt '*gtw<NUM>*' state.sls linux.system.user,openssh,linux.network,ntp,neutron
```

9. Apply the highstate on the gateway node:

```
salt '*gtw<NUM>*' state.highstate
```

Reprovision a gateway node

If an tenant network gateway node is down, you may need to reprovision it.

To reprovision a gateway node:

1. Verify that the name of the gateway node is not registered in salt-key on the Salt Master node. If the node is present, remove it:

```
salt-key | grep gtw<NUM>
salt-key -d gtw<NUM>.domain_name
```

2. Add a physical node using MAAS as described in the MCP Deployment Guide: Provision physical nodes using MAAS.
3. Verify that the required gateway node is defined in /classes/cluster/<cluster_name>/infra/config.yml.
4. Generate the node definition, by applying the reclass.storage state on the Salt Master node:

```
salt '*cfg*' state.sls reclass.storage
```

5. Verify that the target node has connectivity with the Salt Master node:

```
salt '*gtw<NUM>*' test.ping
```

6. Verify that the Salt Minion nodes are synchronized:

```
salt '*gtw<NUM>*' saltutil.sync_all
```

7. On the added node, verify that salt-common and salt-minion have the 2017.7 version.

```
apt-cache policy salt-common
apt-cache policy salt-minion
```

> Note
>
> If the commands above show a different version, follow the MCP Deployment guide: Install the correct versions of salt-common and salt-minion.

8. Perform the initial Salt configuration:

```
salt '*gtw<NUM>*' state.sls salt.minion
```

9. Set up the network interfaces and the SSH access:

```
salt '*gtw<NUM>*' state.sls linux.system.user,openssh,linux.network,ntp,neutron
```

10. Apply the Salt highstate on the gateway node:

```
salt '*gtw<NUM>*' state.highstate
```

Manage RabbitMQ

A RabbitMQ cluster is sensitive to external factors like network throughput and traffic spikes. When running under high load, it requires special start, stop, and restart procedures.

Restart a RabbitMQ node

Caution!

We recommend that you do not restart a RabbitMQ node on a production environment by executing systemctl restart rabbitmq-server since a cluster can become inoperative.

To restart a single RabbitMQ node:

1. Gracefully stop rabbitmq-server on the target node:

   ```
   systemctl stop rabbitmq-server
   ```

2. Verify that the node is removed from the cluster and RabbitMQ is stopped on this node:

   ```
   rabbitmqctl cluster_status
   ```

   Example of system response:

   ```
   Cluster status of node rabbit@msg01
   [{nodes,[{disc,[rabbit@msg01,rabbit@msg02,rabbit@msg03]}]},
   {running_nodes,[rabbit@msg03,rabbit@msg01]},  # <<< rabbit stopped on msg02
   {cluster_name,<<"openstack">>},
   {partitions,[]},
   {alarms,[{rabbit@msg03,[]},{rabbit@msg01,[]}]}]
   ```

3. Start rabbitmq-server:

   ```
   systemctl start rabbitmq-server
   ```

©2025, Mirantis Inc.

Restart a RabbitMQ cluster

To restart the whole RabbitMQ cluster:

1. Stop RabbitMQ on nodes one by one:

```
salt msg01* cmd.run 'systemctl stop rabbitmq-server'
salt msg02* cmd.run 'systemctl stop rabbitmq-server'
salt msg03* cmd.run 'systemctl stop rabbitmq-server'
```

2. Restart RabbitMQ in the reverse order:

```
salt msg03* cmd.run 'systemctl start rabbitmq-server'
salt msg02* cmd.run 'systemctl start rabbitmq-server'
salt msg01* cmd.run 'systemctl start rabbitmq-server'
```

Restart RabbitMQ with clearing the Mnesia database

To restart RabbitMQ with clearing the Mnesia database:

1. Stop RabbitMQ on nodes one by one:

```
salt msg01* cmd.run 'systemctl stop rabbitmq-server'
salt msg02* cmd.run 'systemctl stop rabbitmq-server'
salt msg03* cmd.run 'systemctl stop rabbitmq-server'
```

2. Remove the Mnesia database on all nodes:

```
salt msg0* cmd.run 'rm -rf /var/lib/rabbitmq/mnesia/'
```

3. Apply the rabbitmq state on the first RabbitMQ node:

```
salt msg01* state.apply rabbitmq
```

4. Apply the rabbitmq state on the remaining RabbitMQ nodes:

```
salt -C "msg02* or msg03*" state.apply rabbitmq
```

Switch to nonclustered RabbitMQ

> Note
>
> This feature is available starting from the MCP 2019.2.13 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

> Note
>
> This feature is available for OpenStack Queens and from the RabbitMQ version 3.8.2.

You can switch clustered RabbitMQ to a nonclustered configuration. Mirantis recommends using such approach only to improve the stability and performance on large deployments in case the clustered configuration causes issues.

Switch to nonclustered RabbitMQ

This section instructs you on how to switch clustered RabbitMQ to a nonclustered configuration.

> **Note**
>
> This feature is available starting from the MCP 2019.2.13 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

> **Caution!**
>
> - This feature is available for OpenStack Queens and from the RabbitMQ version 3.8.2.
> - The procedure below applies only to environments without manual changes in the configuration files of OpenStack services. The procedure applies all OpenStack states to all OpenStack nodes and implies that any state can apply without any errors before starting the maintenance.

To switch RabbitMQ to a nonclustered configuration:

1. Perform the following prerequisite steps:
   1. Log in to the Salt Master node.
   2. Verify that the salt-formula-nova version is 2016.12.1+202101271624.d392d41~xenial1 or newer:

      ```
      dpkg -l |grep salt-formula-nova
      ```

   3. Verify that the salt-formula-oslo-templates version is 2018.1+202101191343.e24fd64~xenial1 or newer.

      ```
      dpkg -l |grep salt-formula-oslo-templates
      ```

   4. Create /root/non-clustered-rabbit-helpers.sh with the following content:

      ```
      #!/bin/bash

      # Apply all known openstack states on given target
      # example: run_openstack_states ctl*
      function run_openstack_states {
        local target="$1"
        all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
        #List of nodes in cloud
      ```

```
  list_nodes=`salt -C "$target" test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
  for node in $list_nodes; do
   #List of applications on the given node
   node_applications=$(salt $node pillar.items __reclass__:applications --out=json | jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
   for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
     echo "Applying state: $component on the $node"
     salt $node state.apply $component
    fi
   done
  done
}

# Apply specified update state for all OpenStack applications on given target
# example: run_openstack_update_states ctl0* upgrade.verify
# will run {nova|glance|cinder|keystone}.upgrade.verify on ctl01
function run_openstack_update_states {
  local target="$1"
  local state="$2"
  all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
  #List of nodes in cloud
  list_nodes=`salt -C "$target" test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
  for node in $list_nodes; do
   #List of applications on the given node
   node_applications=$(salt $node pillar.items __reclass__:applications --out=json | jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
   for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
     echo "Applying state: $component.${state} on the $node"
     salt $node state.apply $component.${state}
    fi
   done
  done
}
```

5. Run simple API checks for ctl01*. The output should not include errors.

```
. /root/non-clustered-rabbit-helpers.sh
run_openstack_update_states ctl01* upgrade.verify
```

6. Open your project Git repository with the Reclass model on the cluster level.
7. Prepare the Neutron server for the RabbitMQ reconfiguration:

    1. In openstack/control.yml, specify the allow_automatic_dhcp_failover parameter as required.

        Caution!

        If set to true, the server reschedules the nets from the failed DHCP agents so that the alive agents catch up the net and serve DHCP. Once the agent reconnects to RabbitMQ, it detects that its net has been rescheduled and removes the DHCP port, namespace, and flows. This parameter is useful if the entire gateway node goes down. In case of an unstable RabbitMQ, agents do not go down and the data plane is not affected. Therefore, we recommend that you set the allow_automatic_dhcp_failover parameter to false. However, consider the risks of a gateway node going down before setting the allow_automatic_dhcp_failover parameter.

        ```
        neutron:
          server:
            allow_automatic_dhcp_failover: false
        ```

    2. Apply the changes:

        ```
        salt -C 'I@neutron:server' state.apply neutron.server
        ```

3. Verify the changes:

```
salt -C 'I@neutron:server' cmd.run "grep allow_automatic_dhcp_failover /etc/neutron/neutron.conf"
```

2. Perform the following changes in the Reclass model on the cluster level:

   1. In infra/init.yml, add the following variable:

   ```
   parameters:
     _param:
       openstack_rabbitmq_standalone_mode: true
   ```

   2. In openstack/message_queue.yml, comment the following class:

   ```
   classes:
   #- system.rabbitmq.server.cluster
   ```

   3. In openstack/message_queue.yml, add the following classes:

   ```
   classes:
   - system.keepalived.cluster.instance.rabbitmq_vip
   - system.rabbitmq.server.single
   ```

   4. If your deployment has OpenContrail, add the following variables:

      1. In opencontrail/analytics.yml, add:

      ```
      parameters:
        opencontrail:
          collector:
            message_queue:
              ~members:
                - host: ${_param:openstack_message_queue_address}
      ```

      2. In opencontrail/control.yml, add:

      ```
      parameters:
        opencontrail:
          config:
            message_queue:
              ~members:
                - host: ${_param:openstack_message_queue_address}
          control:
            message_queue:
              ~members:
                - host: ${_param:openstack_message_queue_address}
      ```

5. To update the cells database when running Nova states, add the following variable to openstack/control.yml:

   ```
   parameters:
     nova:
       controller:
         update_cells: true
   ```

6. Refresh pillars on all nodes:

   ```
   salt '*' saltutil.sync_all; salt '*' saltutil.refresh_pillar
   ```

3. Verify that the messaging variables are set correctly:

> Note
>
> The following validation highlights the output for core OpenStack services only. Validate any additional deployed services appropriately.

1. For Keystone:

   ```
   salt -C 'I@keystone:server' pillar.items keystone:server:message_queue:use_vip_address keystone:server:message_queue:host
   ```

2. For Heat:

   ```
   salt -C 'I@heat:server' pillar.items heat:server:message_queue:use_vip_address heat:server:message_queue:host
   ```

3. For Cinder:

   ```
   salt -C 'I@cinder:controller' pillar.items cinder:controller:message_queue:use_vip_address cinder:controller:message_queue:host
   ```

4. For Glance:

   ```
   salt -C 'I@glance:server' pillar.items glance:server:message_queue:use_vip_address glance:server:message_queue:host
   ```

5. For Nova:

   ```
   salt -C 'I@nova:controller' pillar.items nova:controller:message_queue:use_vip_address nova:controller:message_queue:host
   ```

6. For the OpenStack compute nodes:

   ```
   salt -C 'I@nova:compute' pillar.items nova:compute:message_queue:use_vip_address nova:compute:message_queue:host
   ```

7. For Neutron:

   ```
   salt -C 'I@neutron:server' pillar.items neutron:server:message_queue:use_vip_address neutron:server:message_queue:host
   salt -C 'I@neutron:gateway' pillar.items neutron:gateway:message_queue:use_vip_address neutron:gateway:message_queue:host
   ```

8. If your deployment has OpenContrail:

```
salt 'ntw01*' pillar.items opencontrail:config:message_queue:members opencontrail:control:message_queue:members
salt 'nal01*' pillar.items opencontrail:collector:message_queue:members
```

4. Apply the changes:

1. Stop the OpenStack control plane services on the ctl nodes:

```
. /root/non-clustered-rabbit-helpers.sh
run_openstack_update_states ctl* upgrade.service_stopped
```

2. Stop the OpenStack services on the gtw nodes. Skip this step if your deployment has OpenContrail or does not have gtw nodes.

```
. /root/non-clustered-rabbit-helpers.sh
run_openstack_update_states gtw* upgrade.service_stopped
```

3. Reconfigure the Keepalived and RabbitMQ clusters on the msg nodes:

1. Verify that the rabbitmq:cluster pillars are not present:

```
salt -C 'I@rabbitmq:server' pillar.items rabbitmq:cluster
```

2. Verify that the haproxy pillars are not present:

```
salt -C 'I@rabbitmq:server' pillar.item haproxy
```

3. Remove HAProxy, HAProxy monitoring, and reconfigure Keepalived:

```
salt -C 'I@rabbitmq:server' cmd.run "export DEBIAN_FRONTEND=noninteractive; apt purge haproxy -y"
salt -C 'I@rabbitmq:server' state.apply telegraf
salt -C 'I@rabbitmq:server' state.apply keepalived
```

4. Verify that a VIP address is present on one of the msg nodes:

```
OPENSTCK_MSG_Q_ADDRESS=$(salt msg01* pillar.items _param:openstack_message_queue_address --out json|jq '.[][]')
salt -C 'I@rabbitmq:server' cmd.run "ip addr |grep $OPENSTCK_MSG_Q_ADDRESS"
```

5. Stop the RabbitMQ server, clear mnesia, and reconfigure rabbitmq-server:

```
salt -C 'I@rabbitmq:server' cmd.run 'systemctl stop rabbitmq-server'
salt -C 'I@rabbitmq:server' cmd.run 'rm -rf /var/lib/rabbitmq/mnesia/'
salt -C 'I@rabbitmq:server' state.apply rabbitmq
```

6. Verify that the RabbitMQ server is running in a nonclustered configuration:

```
salt -C 'I@rabbitmq:server' cmd.run "rabbitmqctl --formatter=erlang cluster_status |grep running_nodes"
```

Example of system response:

```
msg01.heat-cicd-queens-dvr-sl.local:
    {running_nodes,[rabbit@msg01]},
msg03.heat-cicd-queens-dvr-sl.local:
    {running_nodes,[rabbit@msg03]},
msg02.heat-cicd-queens-dvr-sl.local:
    {running_nodes,[rabbit@msg02]},
```

4. Reconfigure OpenStack services on the ctl nodes:

   1. Apply all OpenStack states on ctl nodes:

      ```
      . /root/non-clustered-rabbit-helpers.sh
      run_openstack_states ctl*
      ```

   2. Verify transport_url for the OpenStack services on the ctl nodes:

      ```
      salt 'ctl*' cmd.run "for s in nova glance cinder keystone heat neutron; do if [[ -d "/etc/\$s" ]]; then grep ^transport_url /etc/\$s/*.conf; fi; done" shell=/bin/bash
      ```

   3. Verify that the cells database is updated and transport_url has a VIP address:

      ```
      salt -C 'I@nova:controller and *01*' cmd.run ". /root/keystonercv3; nova-manage cell_v2 list_cells"
      ```

5. Reconfigure RabbitMQ on the gtw nodes. Skip this step if your deployment has OpenContrail or does not have gtw nodes.

   1. Apply all OpenStack states on the gtw nodes:

      ```
      . /root/non-clustered-rabbit-helpers.sh
      run_openstack_states gtw*
      ```

   2. Verify transport_url for the OpenStack services on the gtw nodes:

      ```
      salt 'gtw*' cmd.run "for s in nova glance cinder keystone heat neutron; do if [[ -d "/etc/\$s" ]]; then grep ^transport_url /etc/\$s/*.conf; fi; done" shell=/bin/bash
      ```

   3. Verify that the agents are up:

      ```
      salt -C 'I@nova:controller and *01*' cmd.run ". /root/keystonercv3; openstack orchestration service list"
      ```

6. If your deployment has OpenContrail, reconfigure RabbitMQ on the ntw and nal nodes:

   1. Apply the following state on the ntw and nal nodes:

      ```
      salt -C 'ntw* or nal*' state.apply opencontrail
      ```

   2. Verify transport_url for the OpenStack services on the ntw and nal nodes:

      ```
      salt -C 'ntw* or nal*' cmd.run "for s in contrail; do if [[ -d "/etc/\$s" ]]; then grep ^rabbitmq_server_list /etc/\$s/*.conf; fi; done" shell=/bin/bash
      salt 'ntw*' cmd.run "for s in contrail; do if [[ -d "/etc/\$s" ]]; then grep ^rabbit_server /etc/\$s/*.conf; fi; done" shell=/bin/bash
      ```

        3. Verify the OpenContrail status:

        ```
        salt -C 'ntw* or nal*' cmd.run 'doctrail all contrail-status'
        ```

    7. Reconfigure OpenStack services on the cmp nodes:

        1. Apply all OpenStack states on the cmp nodes:

        ```
        . /root/non-clustered-rabbit-helpers.sh
        run_openstack_states cmp*
        ```

        2. Verify transport_url for the OpenStack services on the cmp nodes:

        ```
        salt 'cmp*' cmd.run "for s in nova glance cinder keystone heat neutron; do if [[ -d "/etc/\$s" ]]; then grep ^transport_url /etc/\$s/*.conf; fi; done" shell=/bin/bash
        ```

> Caution!
>
> If your deployment has other nodes with OpenStack services, apply the changes on such nodes as well using the required states.

5. Verify the services:

    1. Verify that the Neutron services are up. Skip this step if your deployment has OpenContrail.

    ```
    salt -C 'I@nova:controller and *01*' cmd.run ". /root/keystonercv3; openstack network agent list"
    ```

    2. Verify that the Nova services are up:

    ```
    salt -C 'I@nova:controller and *01*' cmd.run ". /root/keystonercv3; openstack compute service list"
    ```

    3. Verify that Heat services are up:

    ```
    salt -C 'I@nova:controller and *01*' cmd.run ". /root/keystonercv3; openstack orchestration service list"
    ```

    4. Verify that the Cinder services are up:

    ```
    salt -C 'I@nova:controller and *01*' cmd.run ". /root/keystonercv3; openstack volume service list"
    ```

    5. From the ctl01* node, apply the <app>.upgrade.verify state. The output should not include errors.

    ```
    . /root/non-clustered-rabbit-helpers.sh
    run_openstack_update_states ctl01* upgrade.verify
    ```

6. Perform post-configuration steps:

1. Disable the RabbitMQUnequalQueueCritical Prometheus alert:

    1. In stacklight/server.yml, add the following variable:

        ```
        parameters:
          prometheus:
            server:
              alert:
                RabbitMQUnequalQueueCritical:
                  enabled: false
        ```

    2. Apply the Prometheus state to the mon nodes:

        ```
        salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
        ```

2. Revert the changes in the Reclass model on the cluster level:

    1. In openstack/control.yaml, set allow_automatic_dhcp_failover back to true or leave as is if you did not change the value.
    2. In openstack/control.yaml, remove nova:controller:update_cells:true.

3. Apply the Neutron state:

    ```
    salt -C 'I@neutron:server' state.apply neutron.server
    ```

4. Verify the changes:

    ```
    salt -C 'I@neutron:server' cmd.run "grep allow_automatic_dhcp_failover /etc/neutron/neutron.conf"
    ```

5. Remove the script:

    ```
    rm -f /root/non-clustered-rabbit-helpers.sh
    ```

Seealso

Roll back to clustered RabbitMQ

Roll back to clustered RabbitMQ

This section instructs you on how to roll back RabbitMQ to a clustered configuration after switching it to a nonclustered configuration as described in Switch to nonclustered RabbitMQ.

> **Note**
>
> After performing the rollback procedure, you may notice a number of down heat-engine instances of a previous version among the heat-engine running instances. Such behavior is abnormal but expected. Verify the Updated At field of the running instances of heat-engine. Ignore the stopped(down) instances of heat-engine.

To roll back RabbitMQ to a clustered configuration:

1. If you have removed the non-clustered-rabbit-helpers.sh script, create it again as described in Switch to nonclustered RabbitMQ.
2. Revert the changes performed in the cluster model in the step 2 during Switch to nonclustered RabbitMQ. Use git stash, for example, if you did not commit the changes.
3. From the Salt Master node, refresh pillars on all nodes:

   ```
   salt '*' saltutil.sync_all; salt '*' saltutil.refresh_pillar
   ```

4. Roll back the changes on the RabbitMQ nodes:

   ```
   salt -C 'I@rabbitmq:server' cmd.run 'systemctl stop rabbitmq-server'
   salt -C 'I@rabbitmq:server' cmd.run 'rm -rf /var/lib/rabbitmq/mnesia/'

   salt -C 'I@rabbitmq:server' state.apply keepalived
   salt -C 'I@rabbitmq:server' state.apply haproxy
   salt -C 'I@rabbitmq:server' state.apply telegraf
   salt -C 'I@rabbitmq:server' state.apply rabbitmq
   ```

5. Verify that the RabbitMQ server is running in a clustered configuration:

   ```
   salt -C 'I@rabbitmq:server' cmd.run "rabbitmqctl --formatter=erlang cluster_status |grep running_nodes"
   ```

   Example of system response:

   ```
   msg01.heat-cicd-queens-dvr-sl.local:
       {running_nodes,[rabbit@msg02,rabbit@msg03,rabbit@msg01]},
   msg02.heat-cicd-queens-dvr-sl.local:
       {running_nodes,[rabbit@msg01,rabbit@msg03,rabbit@msg02]},
   msg03.heat-cicd-queens-dvr-sl.local:
       {running_nodes,[rabbit@msg02,rabbit@msg01,rabbit@msg03]},
   ```

©2025, Mirantis Inc.

6. Roll back the changes on other nodes:

   1. Roll back the changes on the ctl nodes:

      ```
      . /root/non-clustered-rabbit-helpers.sh
      run_openstack_states ctl*
      ```

   2. Roll back changes on the gtw nodes. Skip this step if your deployment has OpenContrail or does not have gtw nodes.

      ```
      . /root/non-clustered-rabbit-helpers.sh
      run_openstack_states gtw*
      ```

7. If your environment has OpenContrail, roll back the changes on the ntw and nal nodes:

   ```
   salt -C 'ntw* or nal*' state.apply opencontrail
   ```

8. Roll back the changes on the cmp nodes:

   ```
   . /root/non-clustered-rabbit-helpers.sh
   run_openstack_states cmp*
   ```

Enable queue mirroring

> Note
>
> This feature is available starting from the MCP 2019.2.15 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

Mirroring policy enables RabbitMQ to mirror the queues content to an additional RabbitMQ node in the RabbitMQ cluster. Such approach reduces failures during the RabbitMQ cluster recovery.

> Warning
>
> - This feature is of use only for clustered RabbitMQ configurations.
> - Enabling mirroring for queues and exchanges in RabbitMQ may increase the message passing latency and prolong the RabbitMQ cluster recovery after a network partition. Therefore, we recommend accomplishing the procedure on a staging environment before applying it to production.

To enable queue mirroring:

1. Open your project Git repository with the Reclass model on the cluster level.
2. In <cluster_name>/openstack/message_queue.yml, specify ha_exactly_ttl_120 in classes:

   ```
   classes:
   ...
   - system.rabbitmq.server.vhost.openstack
   - system.rabbitmq.server.vhost.openstack.without_rpc_ha
   - system.rabbitmq.server.vhost.openstack.ha_exactly_ttl_120
   ...
   ```

3. Log in to the Salt Master node.
4. Apply the rabbitmq.server state:

   ```
   salt -C 'I@rabbitmq:server and *01*' state.sls rabbitmq.server
   ```

©2025, Mirantis Inc.

Randomize RabbitMQ reconnection intervals

> Note
>
> This feature is available starting from the MCP 2019.2.15 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

You can randomize RabbitMQ reconnection intervals (or timeouts) for the required OpenStack services. It is helpful for large OpenStack environments where a simultaneous reconnection of all OpenStack services after a RabbitMQ cluster partitioning can significantly prolong the RabbitMQ cluster recovery or cause the cluster to enter the split-brain mode.

Using this feature, the following OpenStack configuration options will be randomized:

- kombu_reconnect_delay - from 30 to 60 seconds
- rabbit_retry_interval - from 10 to 60 seconds
- rabbit_retry_backoff - from 30 to 60 seconds
- rabbit_interval_max - from 60 to 180 seconds

To randomize RabbitMQ reconnection intervals :

1. Open your project Git repository with the Reclass model on the cluster level.
2. Open the configuration file of the required OpenStack service. For example, for the OpenStack Compute service (Nova), open <cluster_name>/openstack/compute/init.yml.
3. Under message_queue, specify rabbit_timeouts_random: True:

```
parameters:
  nova:
    compute:
      message_queue:
        rabbit_timeouts_random: True
```

4. Log in to the Salt Master node.
5. Apply the corresponding OpenStack service state(s). For example, for the OpenStack Compute service (Nova), apply the following state:

```
salt -C 'I@nova:compute' state.sls nova.compute
```

©2025, Mirantis Inc.

> Note
>
> Each service configured with this feature on every node will receive new unique timeouts on every run of the corresponding OpenStack service Salt state.

6. Perform the steps 2-5 for other OpenStack services as required.

Remove a node

Removal of a node from a Salt-managed environment is a matter of disabling the salt-minion service running on the node, removing its key from the Salt Master node, and updating the services so that they know that the node is not available anymore.

To remove a node:

1. Stop and disable the salt-minion service on the node you want to remove:

   ```
   systemctl stop salt-minion
   systemctl disable salt-minion
   ```

2. Verify that the name of the node is not registered in salt-key on the Salt Master node. If the node is present, remove it:

   ```
   salt-key | grep <nodename><NUM>
   salt-key -d <nodename><NUM>.domain_name
   ```

3. Update your Reclass metadata model to remove the node from services. Apply the necessary Salt states. This step is generic as different services can be involved depending on the node being removed.

Seealso

Remove a compute node

Manage certificates

After you deploy an MCP cluster, you can renew your expired certificates or replace them by the endpoint certificates provided by a customer as required. When you renew a certificate, its key remains the same. When you replace a certificate, a new certificate key is added accordingly.

You can either push certificates from pillars or regenerate them as follows:

- Generate and update by salt-minion (signed by salt-master)
- Generate and update by external certificate authorities, for example, by Let’s Encrypt

Certificates generated by salt-minion can be renewed by the salt-minion state. The renewal operation becomes available within 30 days before the expiration date. This is controlled by the days_remaining parameter of the x509.certificate_managed Salt state. Refer to Salt.states.x509 for details.

You can force renewal of certificates by removing old certificates and running salt.minion.cert state on each target node.

Publish CA certificates

If you use certificates issued by Certificate Authorities that are not recognized by an operating system, you must publish them.

To publish CA certificates:

1. Open your project Git repository with the Reclass model on the cluster level.
2. Create the /infra/ssl/init.yml file with the following configuration as an example:

```
parameters:
  linux:
    system:
      ca_certificates:
        ca-salt_master_ca: |
          -----BEGIN CERTIFICATE-----
          MIIGXzCCBEegAwIBAgIDEUB0MA0GCSqGSIb3DQEBCwUAMFkxEzARBgoJkiaJk/Is
          ...
          YqQO
          -----END CERTIFICATE-----
        ca-salt_master_ca_old: |
          -----BEGIN CERTIFICATE-----
          MIIFgDCCA2igAwIBAgIDET0sMA0GCSqGSIb3DQEBCwUAMFkxEzARBgoJkiaJk/Is
          ...
          WzUuf8H9dBW2DPtk5Jq/+QWtYMs=
          -----END CERTIFICATE-----
```

3. To publish the certificates on all nodes managed by Salt, update /infra/init.yml by adding the newly created class:

```
classes:
- cluster.<cluster_name>.infra.ssl
```

4. To publish the certificates on a specific node, update /infra/config.yml. For example:

```
parameters:
  reclass:
    storage:
      node:
        openstack_control_node01:
          classes:
          - cluster.${_param:cluster_name}.openstack.ssl
```

5. Log in to the Salt Master node.
6. Update the Reclass storage:

```
salt-call state.sls reclass.storage -l debug
```

7. Apply the linux.system state on all nodes:

```
salt \* state.sls linux.system.certificate -l debug
```

NGINX certificates

This section describes how to renew or replace the NGINX certificates managed by either salt-minion or self-managed certificates using pillars. For both cases, you must verify the GlusterFS share salt_pki before renewal.

Verify the GlusterFS share salt_pki

Before you proceed with the NGINX certificates renewal or replacement, verify the GlusterFS share salt_pki.

To verify the GlusterFS share salt_pki:

1. Log in to any infrastructure node that hosts the salt_pki GlusterFS volume.
2. Obtain the list of the GlusterFS minions IDs:

   ```
   salt -C 'I@glusterfs:server' test.ping --output yaml | cut -d':' -f1
   ```

   Example of system response:

   ```
   kvm01.multinode-ha.int
   kvm03.multinode-ha.int
   kvm02.multinode-ha.int
   ```

3. Verify that the volume is replicated and is online for any of the minion IDs from the list obtained in the previous step.

   ```
   salt <minion_id> cmd.run 'gluster volume status salt_pki'
   ```

   Example of system response:

   ```
   Status of volume: salt_pki
   Gluster process                          TCP Port  RDMA Port  Online  Pid
   ------------------------------------------------------------------------------
   Brick 192.168.2.241:/srv/glusterfs/salt_pki 49154     0          Y       9211
   Brick 192.168.2.242:/srv/glusterfs/salt_pki 49154     0          Y       8499
   Brick 192.168.2.243:/srv/glusterfs/salt_pki 49154     0          Y       8332
   Self-heal Daemon on localhost               N/A       N/A        Y       6313
   Self-heal Daemon on 192.168.2.242           N/A       N/A        Y       10203
   Self-heal Daemon on 192.168.2.243           N/A       N/A        Y       2068

   Task Status of Volume salt_pki
   ------------------------------------------------------------------------------
   There are no active volume tasks
   ```

4. Log in to the Salt Master node.
5. Verify that the salt_pki volume is mounted on each proxy node and the Salt Master node:

   ```
   salt -C 'I@nginx:server:site:*:host:protocol:https or I@salt:master' \
   cmd.run 'mount | grep salt_pki'
   ```

   Example of system response:

```
prx01.multinode-ha.int:
    192.168.2.240:/salt_pki on /srv/salt/pki type fuse.glusterfs \
    (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
prx02.multinode-ha.int:
    192.168.2.240:/salt_pki on /srv/salt/pki type fuse.glusterfs \
    (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
cfg01.multinode-ha.int:
    192.168.2.240:/salt_pki on /srv/salt/pki type fuse.glusterfs \
    (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
```

6. Proceed with the renewal or replacement of the NGINX certificates as required.

©2025, Mirantis Inc.

Renew or replace the NGINX certificates managed by salt-minion

This section describes how to renew or replace the NGINX certificates managed by salt-minion.

To renew or replace the NGINX certificates managed by salt-minion:

1. Complete the steps described in Verify the GlusterFS share salt_pki.
2. Log in to the Salt Master node.
3. Verify the certificate validity date:

   ```
   openssl x509 -in /srv/salt/pki/*/proxy.crt -text -noout | grep -Ei 'after|before'
   ```

   Example of system response:

   ```
   Not Before: May 30 17:21:10 2018 GMT
   Not After : May 30 17:21:10 2019 GMT
   ```

4. Remove your current certificates from the Salt Master node.

   > Note
   >
   > The following command also removes certificates from all proxy nodes as they use the same GlusterFS share.

   ```
   rm -f /srv/salt/pki/*/*.[pemcrt]*
   ```

5. If you replace the certificates, remove the private key:

   ```
   rm -f /srv/salt/pki/*/proxy.key
   ```

6. Renew or replace your certificates by applying the salt.minion state on all proxy nodes one by one:

   ```
   salt -C 'I@nginx:server:site:*:host:protocol:https' state.sls salt.minion.cert -b 1
   ```

7. Apply the nginx state on all proxy nodes one by one:

   ```
   salt -C 'I@nginx:server:site:*:host:protocol:https' state.sls nginx -b 1
   ```

8. Verify the new certificate validity date:

   ```
   openssl x509 -in /srv/salt/pki/*/proxy.crt -text -noout | grep -Ei 'after|before'
   ```

   Example of system response:

```
Not Before: May 30 17:21:10 2018 GMT
Not After : May 30 17:21:10 2019 GMT
```

Renew the self-managed NGINX certificates

This section describes how to renew the self-managed NGINX certificates.

To renew the self-managed NGINX certificates:

1. Complete the steps described in Verify the GlusterFS share salt_pki.
2. Open your project Git repository with the Reclass model on the cluster level.
3. Update the /openstack/proxy.yml file with the following configuration as an example:

```
parameters:
  _params:
    nginx_proxy_ssl:
      enabled: true
      mode: secure
      key_file:  /srv/salt/pki/${_param:cluster_name}/FQDN_PROXY_CERT.key
      cert_file: /srv/salt/pki/${_param:cluster_name}/FQDN_PROXY_CERT.crt
      chain_file: /srv/salt/pki/${_param:cluster_name}/FQDN_PROXY_CERT_CHAIN.crt
      key: |
        -----BEGIN PRIVATE KEY-----
        MIIJRAIBADANBgkqhkiG9w0BAQEFAASCCS4wggkqAgEAAoICAQC3qXiZiugf6HlR
        ...
        aXK0Fg1hJKu60Oh+E5H1d+ZVbP30xpdQ
        -----END PRIVATE KEY-----
      cert: |
        -----BEGIN CERTIFICATE-----
        MIIHDzCCBPegAwIBAgIDLYclMA0GCSqGSIb3DQEBCwUAMFkxEzARBgoJkiaJk/Is
        ...
        lHfjP1c6iWAL0YEp1IMCeM01l4WWj0ymb7f4wgOzcULfwzU=
        -----END CERTIFICATE-----
      chain: |
        -----BEGIN CERTIFICATE-----
        MIIFgDCCA2igAwIBAgIDET0sMA0GCSqGSIb3DQEBCwUAMFkxEzARBgoJkiaJk/Is
        ...
        UPwFzYlVkwy4ny+UJm9js8iynKro643mXty9vj5TdN1iK3ZA4f4/7kenuHtGBNur
        WzUuf8H9dBW2DPtk5Jq/+QWtYMs=
        -----END CERTIFICATE-----
        -----BEGIN CERTIFICATE-----
        MIIGXzCCBEegAwIBAgIDEUB0MA0GCSqGSIb3DQEBCwUAMFkxEzARBgoJkiaJk/Is
        ...
        /inxvBr89TvbCP2hweGMD6w1mKJU2SWEQwMs7P72dU7VuVqyyoutMWakJZ+xoGE9
        YqQO
        -----END CERTIFICATE-----
        -----BEGIN CERTIFICATE-----
        MIIHDzCCBPegAwIBAgIDLYclMA0GCSqGSIb3DQEBCwUAMFkxEzARBgoJkiaJk/Is
        ...
        lHfjP1c6iWAL0YEp1IMCeM01l4WWj0ymb7f4wgOzcULfwzU=
        -----END CERTIFICATE-----
```

> Note
>
> Modify the example above by adding your certificates and key:
>
> - If you renew the certificates, leave your existing key and update the cert and chain sections.
> - If you replace the certificates, modify all three sections.

> Note
>
> The key, cert, and chain sections are optional. You can select from the following options:
>
> - Store certificates in the file system in /srv/salt/pki/**/ and add the key_file, cert_file, and chain_file lines to /openstack/proxy.yml.
> - Add only the key, cert, and chain sections without the key_file, cert_file, and chain_file lines to /openstack/proxy.yml. The certificates are stored under the /etc directory as default paths in the Salt formula.
> - Use all three sections, as in the example above. All content is available in pillar and is stored in /srv/salt/pki/** as well. This option requires manual upload of the certificates and key files content to the .yml files.

4. Log in to the Salt Master node.
5. Verify the new certificate validity date:

   ```
   openssl x509 -in /srv/salt/pki/*/proxy.crt -text -noout | grep -Ei 'after|before'
   ```

   Example of system response:

   ```
   Not Before: May 30 17:21:10 2018 GMT
   Not After : May 30 17:21:10 2019 GMT
   ```

6. Remove the current certificates.

> Note
>
> The following command also removes certificates from all proxy nodes as they use the same GlusterFS share.

```
rm -f /srv/salt/pki/*/*.[pemcrt]*
```

7. If you replace the certificates, remove the private key:

```
/srv/salt/pki/*/proxy.key
```

8. Apply the nginx state on all proxy nodes one by one:

```
salt -C 'I@nginx:server' state.sls nginx -b 1
```

9. Verify the new certificate validity date:

```
openssl x509 -in /srv/salt/pki/*/proxy.crt -text -noout | grep -Ei 'after|before'
```

   Example of system response:

```
Not Before: May 30 17:21:10 2018 GMT
Not After : May 30 17:21:10 2019 GMT
```

10. Restart the NGINX services and remove the VIP before restart:

```
salt -C 'I@nginx:server' cmd.run 'service keepalived stop; sleep 5; \
service nginx restart; service keepalived start' -b 1
```

HAProxy certificates

This section describes how to renew or replace the HAProxy certificates managed by either salt-minion or self-managed certificates using pillars.

Renew or replace the HAProxy certificates managed by salt-minion

This section describes how to renew or replace the HAProxy certificates managed by salt-minion.

To renew or replace the HAProxy certificates managed by salt-minion:

1. Log in to the Salt Master node.
2. Obtain the list of the HAProxy minions IDs where the certificate should be replaced:

```
salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' \
pillar.get _nonexistent | cut -d':' -f1
```

   Example of system response:

```
cid02.multinode-ha.int
cid03.multinode-ha.int
cid01.multinode-ha.int
```

3. Verify the certificate validity date for each HAProxy minion listed in the output of the above command:

```
for m in $(salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' \
pillar.get _nonexistent | cut -d':' -f1); do for c in $(salt -C ${m} \
pillar.get 'haproxy:proxy:listen' --out=txt | egrep -o "'pem_file': '\S+'" | \
cut -d"'" -f4 | sort | uniq | tr '\n' ' '); do salt -C ${m} \
cmd.run "openssl x509 -in ${c} -text | egrep -i 'after|before'"; done; done;
```

   Example of system response:

```
cid02.multinode-ha.int:
            Not Before: May 29 12:58:21 2018 GMT
            Not After : May 29 12:58:21 2019 GMT
```

4. Remove your current certificates from each HAProxy minion:

```
for m in $(salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' \
pillar.get _nonexistent | cut -d':' -f1); do for c in $(salt -C ${m} \
pillar.get 'haproxy:proxy:listen' --out=txt | egrep -o "'pem_file': '\S+'" | cut -d"'" \
-f4 | sort | uniq | sed s/-all.pem/.crt/ | tr '\n' ' '); \
do salt -C ${m} cmd.run "rm -f ${c}"; done; done; \
for m in $(salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' \
pillar.get _nonexistent | cut -d':' -f1); do for c in $(salt -C ${m} \
pillar.get 'haproxy:proxy:listen' --out=txt | egrep -o "'pem_file': '\S+'" | cut -d"'" \
-f4 | sort | uniq | tr '\n' ' '); do salt -C ${m} cmd.run "rm -f ${c}"; done; done; \
salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' \
cmd.run 'rm -f /etc/haproxy/ssl/salt_master_ca-ca.crt'
```

5. If you replace the certificates, remove the private key:

```
for m in $(salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' \
pillar.get _nonexistent | cut -d':' -f1); do for c in $(salt -C ${m} \
pillar.get 'haproxy:proxy:listen' --out=txt | egrep -o "'pem_file': '\S+'" | cut -d"'" \
-f4 | sort | uniq | sed s/-all.pem/.key/ | tr '\n' ' '); \
do salt -C ${m} cmd.run "rm -f ${c}"; done; done;
```

6. Apply the salt.minion.grains state for all HAProxy nodes to retrieve the CA certificate from Salt Master:

```
salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' state.sls salt.minion.grains
```

7. Apply the salt.minion.cert state for all HAProxy nodes:

```
salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' state.sls salt.minion.cert
```

8. Verify the new certificate validity date:

```
for m in $(salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' \
pillar.get _nonexistent | cut -d':' -f1); do for c in $(salt -C ${m} \
pillar.get 'haproxy:proxy:listen' --out=txt | egrep -o "'pem_file': '\S+'" | cut -d"'" \
-f4 | sort | uniq | tr '\n' ' '); do salt -C ${m} \
cmd.run "openssl x509 -in ${c} -text | egrep -i 'after|before'"; done; done;
```

Example of system response:

```
cid02.multinode-ha.int:
            Not Before: Jun  6 17:24:09 2018 GMT
            Not After : Jun  6 17:24:09 2019 GMT
```

9. Restart the HAProxy services on each HAProxy minion and remove the VIP before restart:

```
salt -C 'I@haproxy:proxy:listen:*:binds:ssl:enabled:true' \
cmd.run 'service keepalived stop; sleep 5; \
service haproxy stop; service haproxy start; service keepalived start' -b 1
```

Renew or replace the self-managed HAProxy certificates

This section describes how to renew or replace the self-managed HAProxy certificates.

To renew or replace the self-managed HAProxy certificates:

1. Log in to the Salt Master node.
2. Verify the certificate validity date:

```
for node in $(salt -C 'I@haproxy:proxy' test.ping --output yaml | cut -d':' -f1); do
  for name in $(salt ${node} pillar.get haproxy:proxy --output=json | jq '.. \
  | .listen? | .. | .ssl? | .pem_file?' | grep -v null | sort | uniq); do
    salt ${node} cmd.run "openssl x509 -in ${name} -text -noout | grep -Ei 'after|before'";
  done;
done;
```

> Note
>
> In the command above, the pem_file value is used to specify the explicit certificate path.

   Example of system response:

```
cid02.multinode-ha.int:
                Not Before: May 25 15:32:17 2018 GMT
                Not After : May 25 15:32:17 2019 GMT
cid01.multinode-ha.int:
                Not Before: May 25 15:29:17 2018 GMT
                Not After : May 25 15:29:17 2019 GMT
cid03.multinode-ha.int:
                Not Before: May 25 15:21:17 2018 GMT
                Not After : May 25 15:21:17 2019 GMT
```

3. Open your project Git repository with Reclass model on the cluster level.
4. For each class file with the HAProxy class enabled, update its pillar values with the following configuration as an example:

```
parameters:
  _params:
    haproxy_proxy_ssl:
      enabled: true
      mode: secure
      key: |
        -----BEGIN RSA PRIVATE KEY-----
        MIIJKAIBAAKCAgEAxSXLtYhzptxcAdnsNy2r8NkgskPm3J/l54hmhuSoL61LpEIi
```

```
      ...
      0z/c5yAddRpU/i6/TH2RlBaSGfmoNw/IuFfLsZI2O6dQo4e+QKX+V3JTeNY=
      -----END RSA PRIVATE KEY-----
    cert: |
      -----BEGIN CERTIFICATE-----
      MIIGEzCCA/ugAwIBAgIILX5kuGcAhw8wDQYJKoZIhvcNAQELBQAwSjELMAkGA1UE
      ...
      /in+Y5Wrl1uGHYeFe0yOdb1uxH+PLxc=
      -----END CERTIFICATE-----
    chain: |
      -----BEGIN RSA PRIVATE KEY-----
      MIIJKAIBAAKCAgEAxSXLtYhzptxcAdnsNy2r8NkgskPm3J/l54hmhuSoL61LpEIi
      ...
      0z/c5yAddRpU/i6/TH2RlBaSGfmoNw/IuFfLsZI2O6dQo4e+QKX+V3JTeNY=
      -----END RSA PRIVATE KEY-----
      -----BEGIN CERTIFICATE-----
      MIIGEzCCA/ugAwIBAgIILX5kuGcAhw8wDQYJKoZIhvcNAQELBQAwSjELMAkGA1UE
      ...
      /in+Y5Wrl1uGHYeFe0yOdb1uxH+PLxc=
      -----END CERTIFICATE-----
      -----BEGIN CERTIFICATE-----
      MIIF0TCCA7mgAwIBAgIJAOkTQnjLz6rEMA0GCSqGSIb3DQEBCwUAMEoxCzAJBgNV
      ...
      M8IfJ5I=
      -----END CERTIFICATE-----
```

Note

Modify the example above by adding your certificates and key:

- If you renew the certificates, leave your existing key and update the cert and chain sections.
- If you replace the certificates, modify all three sections.

5. Remove your current certificates from the HAProxy nodes:

```
for node in $(salt -C 'I@haproxy:proxy' test.ping --output yaml | cut -d':' -f1); do
  for name in $(salt ${node} pillar.get haproxy:proxy --output=json | jq '.. \
  | .listen? | .. | .ssl? | .pem_file?' | grep -v null | sort | uniq); do
    salt ${node} cmd.run "rm -f ${name}";
  done;
done;
```

6. Apply the haproxy.proxy state on all HAProxy nodes one by one:

```
salt -C 'I@haproxy:proxy' state.sls haproxy.proxy -b 1
```

7. Verify the new certificate validity date:

```
for node in $(salt -C 'I@haproxy:proxy' test.ping --output yaml | cut -d':' -f1); do
  for name in $(salt ${node} pillar.get haproxy:proxy --output=json | jq '.. \
  | .listen? | .. | .ssl? | .pem_file?' | grep -v null | sort | uniq); do
    salt ${node} cmd.run "openssl x509 -in ${name} -text -noout | grep -Ei 'after|before'";
  done;
done;
```

Example of system response:

```
cid02.multinode-ha.int:
                Not Before: May 25 15:29:17 2018 GMT
                Not After : May 25 15:29:17 2019 GMT
cid03.multinode-ha.int:
                Not Before: May 25 15:29:17 2018 GMT
                Not After : May 25 15:29:17 2019 GMT
cid01.multinode-ha.int:
                Not Before: May 25 15:29:17 2018 GMT
                Not After : May 25 15:29:17 2019 GMT
```

8. Restart the HAProxy services one by one and remove the VIP before restart:

```
salt -C 'I@haproxy:proxy' cmd.run 'service keepalived stop; sleep 5; \
service haproxy stop; service haproxy start; service keepalived start' -b 1
```

Apache certificates

This section describes how to renew or replace the Apache certificates managed by either salt-minion or self-managed certificates using pillars.

Renew or replace the Apache certificates managed by salt-minion

This section describes how to renew or replace the Apache certificates managed by salt-minion.

> **Warning**
>
> If you replace or renew the Apache certificates after the Salt Master CA certificate has been replaced, make sure that both new and old CA certificates are published as described in Publish CA certificates.

To renew or replace the Apache certificates managed by salt-minion:

1. Log in to the Salt Master node.
2. Verify your current certificate validity date:

   ```
   salt -C 'I@apache:server' cmd.run 'openssl x509 \
   -in /etc/ssl/certs/internal_proxy.crt -text -noout | grep -Ei "after|before"'
   ```

   Example of system response:

   ```
   ctl02.multinode-ha.int:
                   Not Before: May 29 12:58:21 2018 GMT
                   Not After : May 29 12:58:21 2019 GMT
   ctl03.multinode-ha.int:
                   Not Before: May 29 12:58:25 2018 GMT
                   Not After : May 29 12:58:25 2019 GMT
   ctl01.multinode-ha.int:
                   Not Before: Apr 27 12:37:28 2018 GMT
                   Not After : Apr 27 12:37:28 2019 GMT
   ```

3. Remove your current certificates from the Apache nodes:

   ```
   salt -C 'I@apache:server' cmd.run 'rm -f /etc/ssl/certs/internal_proxy.crt'
   ```

4. If you replace the certificates, remove the private key:

   ```
   salt -C 'I@apache:server' cmd.run 'rm -f /etc/ssl/private/internal_proxy.key'
   ```

5. Renew or replace your certificates by applying the salt.minion.cert state on all Apache nodes one by one:

   ```
   salt -C 'I@apache:server' state.sls salt.minion.cert
   ```

6. Refresh the CA chain:

©2025, Mirantis Inc.

```
salt -C 'I@apache:server' cmd.run 'cat /etc/ssl/certs/internal_proxy.crt \
/usr/local/share/ca-certificates/ca-salt_master_ca.crt > \
/etc/ssl/certs/internal_proxy-with-chain.crt; \
chmod 0644 /etc/ssl/certs/internal_proxy-with-chain.crt; \
chown root:root /etc/ssl/certs/internal_proxy-with-chain.crt'
```

7. Verify the new certificate validity date:

```
salt -C 'I@apache:server' cmd.run 'openssl x509 \
-in /etc/ssl/certs/internal_proxy.crt -text -noout | grep -Ei "after|before"'
```

Example of system response:

```
ctl02.multinode-ha.int:
            Not Before: Jun  6 17:24:09 2018 GMT
            Not After : Jun  6 17:24:09 2019 GMT
ctl03.multinode-ha.int:
            Not Before: Jun  6 17:24:42 2018 GMT
            Not After : Jun  6 17:24:42 2019 GMT
ctl01.multinode-ha.int:
            Not Before: Jun  6 17:23:38 2018 GMT
            Not After : Jun  6 17:23:38 2019 GMT
```

8. Restart the Apache services one by one:

```
salt -C 'I@apache:server' cmd.run 'service apache2 stop; service apache2 start; sleep 60' -b1
```

Replace the self-managed Apache certificates

This section describes how to replace the self-managed Apache certificates.

> **Warning**
>
> If you replace or renew the Apache certificates after the Salt Master CA certificate has been replaced, make sure that both new and old CA certificates are published as described in Publish CA certificates.

To replace the self-managed Apache certificates:

1. Log in to the Salt Master node.
2. Verify your current certificate validity date:

   ```
   for node in $(salt -C 'I@apache:server' test.ping --output yaml | cut -d':' -f1); do
     for name in $(salt ${node} pillar.get apache:server:site --output=json | \
     jq '.. | .host? | .name?' | grep -v null | sort | uniq); do
       salt ${node} cmd.run "openssl x509 -in /etc/ssl/certs/${name}.crt -text \
       -noout | grep -Ei 'after|before'";
     done;
   done;
   ```

   Example of system response:

   ```
   ctl02.multinode-ha.int:
               Not Before: May 29 12:58:21 2018 GMT
               Not After : May 29 12:58:21 2019 GMT
   ctl03.multinode-ha.int:
               Not Before: May 29 12:58:25 2018 GMT
               Not After : May 29 12:58:25 2019 GMT
   ctl01.multinode-ha.int:
               Not Before: Apr 27 12:37:28 2018 GMT
               Not After : Apr 27 12:37:28 2019 GMT
   ```

3. Open your project Git repository with Reclass model on the cluster level.
4. For each class file with the Apache server class enabled, update the _param:apache_proxy_ssl value with the following configuration as an example:

   ```
   parameters:
     _params:
       apache_proxy_ssl:
         enabled: true
         mode: secure
         key: |
   ```

```
    -----BEGIN RSA PRIVATE KEY-----
    MIIJKAIBAAKCAgEAxSXLtYhzptxcAdnsNy2r8NkgskPm3J/l54hmhuSoL61LpEIi
    ...
    0z/c5yAddRpU/i6/TH2RlBaSGfmoNw/IuFfLsZI2O6dQo4e+QKX+V3JTeNY=
    -----END RSA PRIVATE KEY-----
  cert: |
    -----BEGIN CERTIFICATE-----
    MIIGEzCCA/ugAwIBAgIILX5kuGcAhw8wDQYJKoZIhvcNAQELBQAwSjELMAkGA1UE
    ...
    /in+Y5Wrl1uGHYeFe0yOdb1uxH+PLxc=
    -----END CERTIFICATE-----
  chain: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIJKAIBAAKCAgEAxSXLtYhzptxcAdnsNy2r8NkgskPm3J/l54hmhuSoL61LpEIi
    ...
    0z/c5yAddRpU/i6/TH2RlBaSGfmoNw/IuFfLsZI2O6dQo4e+QKX+V3JTeNY=
    -----END RSA PRIVATE KEY-----
    -----BEGIN CERTIFICATE-----
    MIIGEzCCA/ugAwIBAgIILX5kuGcAhw8wDQYJKoZIhvcNAQELBQAwSjELMAkGA1UE
    ...
    /in+Y5Wrl1uGHYeFe0yOdb1uxH+PLxc=
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    MIIF0TCCA7mgAwIBAgIJAOkTQnjLz6rEMA0GCSqGSIb3DQEBCwUAMEoxCzAJBgNV
    ...
    M8IfJ5I=
    -----END CERTIFICATE-----
```

Note

Modify the example above by adding your certificates and key:

- If you renew the certificates, leave your existing key and update the cert and chain sections.
- If you replace the certificates, modify all three sections.

5. Remove your current certificates from the Apache nodes:

```
for node in $(salt -C 'I@apache:server' test.ping --output yaml | cut -d':' -f1); do
  for name in $(salt ${node} pillar.get apache:server:site --output=json | \
  jq '.. | .host? | .name?' | grep -v null | sort | uniq); do
    salt ${node} cmd.run "rm -f /etc/ssl/certs/${name}.crt";
  done;
done;
```

6. Apply the apache.server state on all Apache nodes one by one:

```
salt -C 'I@apache:server' state.sls apache.server
```

7. Verify the new certificate validity date:

```
for node in $(salt -C 'I@apache:server' test.ping --output yaml | cut -d':' -f1); do
  for name in $(salt ${node} pillar.get apache:server:site --output=json | \
  jq '.. | .host? | .name?' | grep -v null | sort | uniq); do
    salt ${node} cmd.run "openssl x509 -in /etc/ssl/certs/${name}.crt -text \
    -noout | grep -Ei 'after|before'";
  done;
done;
```

Example of system response:

```
ctl02.multinode-ha.int:
                Not Before: Jun  6 17:24:09 2018 GMT
                Not After : Jun  6 17:24:09 2019 GMT
ctl03.multinode-ha.int:
                Not Before: Jun  6 17:24:42 2018 GMT
                Not After : Jun  6 17:24:42 2019 GMT
ctl01.multinode-ha.int:
                Not Before: Jun  6 17:23:38 2018 GMT
                Not After : Jun  6 17:23:38 2019 GMT
```

8. Restart the Apache services one by one:

```
salt -C 'I@apache:server' cmd.run 'service apache2 stop; service apache2 start' -b 1
```

©2025, Mirantis Inc.

RabbitMQ certificates

This section describes how to renew or replace the RabbitMQ cluster certificates managed by either salt-minion or self-managed certificates using pillars.

Verify that the RabbitMQ cluster uses certificates

This section describes how to determine whether your RabbitMQ cluster uses certificates and identify their location on the system.

To verify that the RabbitMQ cluster uses certificates:

1. Log in to the Salt Master node.
2. Run the following command:

   ```
   salt -C 'I@rabbitmq:server' cmd.run "rabbitmqctl environment | \
   grep -E '/ssl/|ssl_listener|protocol_version'"
   ```

   Example of system response:

   ```
   msg02.multinode-ha.int:
           {ssl_listeners,[{"0.0.0.0",5671}]},
              [{cacertfile,"/etc/rabbitmq/ssl/ca.pem"},
               {certfile,"/etc/rabbitmq/ssl/cert.pem"},
               {keyfile,"/etc/rabbitmq/ssl/key.pem"},
       {ssl,[{protocol_version,['tlsv1.2','tlsv1.1',tlsv1]}]},
   msg01.multinode-ha.int:
           {ssl_listeners,[{"0.0.0.0",5671}]},
              [{cacertfile,"/etc/rabbitmq/ssl/ca.pem"},
               {certfile,"/etc/rabbitmq/ssl/cert.pem"},
               {keyfile,"/etc/rabbitmq/ssl/key.pem"},
       {ssl,[{protocol_version,['tlsv1.2','tlsv1.1',tlsv1]}]},
   msg03.multinode-ha.int:
           {ssl_listeners,[{"0.0.0.0",5671}]},
              [{cacertfile,"/etc/rabbitmq/ssl/ca.pem"},
               {certfile,"/etc/rabbitmq/ssl/cert.pem"},
               {keyfile,"/etc/rabbitmq/ssl/key.pem"},
       {ssl,[{protocol_version,['tlsv1.2','tlsv1.1',tlsv1]}]},
   ```

3. Proceed to renewal or replacement of your certificates as required.

Renew or replace the RabbitMQ certificates managed by salt-minion

This section describes how to renew or replace the RabbitMQ certificates managed by salt-minion.

To renew or replace the RabbitMQ certificates managed by salt-minion:

1. Log in to the Salt Master node.
2. Verify the certificates validity dates:

   ```
   salt -C 'I@rabbitmq:server' cmd.run 'openssl x509 \
   -in /etc/rabbitmq/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
   ```

   Example of system response:

   ```
   Not Before: Apr 27 12:37:14 2018 GMT
   Not After : Apr 27 12:37:14 2019 GMT
   Not Before: Apr 27 12:37:08 2018 GMT
   Not After : Apr 27 12:37:08 2019 GMT
   Not Before: Apr 27 12:37:13 2018 GMT
   Not After : Apr 27 12:37:13 2019 GMT
   ```

3. Remove the certificates from the RabbitMQ nodes:

   ```
   salt -C 'I@rabbitmq:server' cmd.run 'rm -f /etc/rabbitmq/ssl/cert.pem'
   ```

4. If you replace the certificates, remove the private key:

   ```
   salt -C 'I@rabbitmq:server' cmd.run 'rm -f /etc/rabbitmq/ssl/key.pem'
   ```

5. Regenerate the certificates on the RabbitMQ nodes:

   ```
   salt -C 'I@rabbitmq:server' state.sls salt.minion.cert
   ```

6. Verify that the certificates validity dates have changed:

   ```
   salt -C 'I@rabbitmq:server' cmd.run 'openssl x509 \
   -in /etc/rabbitmq/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
   ```

   Example of system response:

   ```
   Not Before: Jun  4 23:52:40 2018 GMT
   Not After : Jun  4 23:52:40 2019 GMT
   Not Before: Jun  4 23:52:41 2018 GMT
   Not After : Jun  4 23:52:41 2019 GMT
   Not Before: Jun  4 23:52:41 2018 GMT
   Not After : Jun  4 23:52:41 2019 GMT
   ```

7. Restart the RabbitMQ services one by one:

```
salt -C 'I@rabbitmq:server' cmd.run 'service rabbitmq-server stop; \
service rabbitmq-server start' -b1
```

8. Verify the RabbitMQ cluster status:

```
salt -C 'I@rabbitmq:server' cmd.run 'rabbitmqctl cluster_status'
```

Example of system response:

```
msg03.multinode-ha.int:
    Cluster status of node rabbit@msg03
    [{nodes,[{disc,[rabbit@msg01,rabbit@msg02,rabbit@msg03]}]},
     {running_nodes,[rabbit@msg01,rabbit@msg02,rabbit@msg03]},
     {cluster_name,<<"openstack">>},
     {partitions,[]},
     {alarms,[{rabbit@msg01,[]},{rabbit@msg02,[]},{rabbit@msg03,[]}]}]
msg01.multinode-ha.int:
    Cluster status of node rabbit@msg01
    [{nodes,[{disc,[rabbit@msg01,rabbit@msg02,rabbit@msg03]}]},
     {running_nodes,[rabbit@msg03,rabbit@msg02,rabbit@msg01]},
     {cluster_name,<<"openstack">>},
     {partitions,[]},
     {alarms,[{rabbit@msg03,[]},{rabbit@msg02,[]},{rabbit@msg01,[]}]}]
msg02.multinode-ha.int:
    Cluster status of node rabbit@msg02
    [{nodes,[{disc,[rabbit@msg01,rabbit@msg02,rabbit@msg03]}]},
     {running_nodes,[rabbit@msg03,rabbit@msg01,rabbit@msg02]},
     {cluster_name,<<"openstack">>},
     {partitions,[]},
     {alarms,[{rabbit@msg03,[]},{rabbit@msg01,[]},{rabbit@msg02,[]}]}]
```

Renew or replace the self-managed RabbitMQ certificates

This section describes how to renew or replace the self-managed RabbitMQ certificates.

To renew or replace the self-managed RabbitMQ certificates:

1. Open your project Git repository with Reclass model on the cluster level.
2. Create the /openstack/ssl/rabbitmq.yml file with the following configuration as an example:

```
classes:
- cluster.<cluster_name>.openstack.ssl
parameters:
  rabbitmq:
    server:
      enabled: true
      ...
      ssl:
        enabled: True
        key: ${_param:rabbitmq_ssl_key}
        cacert_chain: ${_param:rabbitmq_ssl_cacert_chain}
        cert: ${_param:rabbitmq_ssl_cert}
```

> Note
>
> Substitute <cluster_name> with the appropriate value.

3. Create the /openstack/ssl/init.yml file with the following configuration as an example:

```
parameters:
  _param:
    rabbitmq_ssl_cacert_chain: |
      -----BEGIN CERTIFICATE-----
      MIIF0TCCA7mgAwIBAgIJAOkTQnjLz6rEMA0GCSqGSIb3DQEBCwUAMEoxCzAJBgNV
      ...
      RHXc4FoWv9/n8ZcfsqjQCjF3vUUZBB3zdlfLCLJRruB4xxYukc3gFpFLm21+0ih+
      M8IfJ5I=
      -----END CERTIFICATE-----
    rabbitmq_ssl_key: |
      -----BEGIN RSA PRIVATE KEY-----
      MIIJKQIBAAKCAgEArVSJ16ePjCik+6bZBzhiu3enXw8R9Ms1k4x57633IX1sEZTJ
      ...
      0VgM2bDSNyUuiwCbOMK0Kyn+wGeHF/jGSbVsxYI4OeLFz8gdVUqm7olJj4j3xemY
      BlWVHRa/dEG1qfSoqFU9+IQTd+U42mtvvH3oJHEXK7WXzborIXTQ/08Ztdvy
      -----END RSA PRIVATE KEY-----
    rabbitmq_ssl_cert: |
```

```
-----BEGIN CERTIFICATE-----
MIIGIDCCBAigAwIBAgIJAJznLlNteaZFMA0GCSqGSIb3DQEBCwUAMEoxCzAJBgNV
...
MfXPTUI+7+5WQLx10yavJ2gOhdyVuDVagfUM4epcriJbACuphDxHj45GINOGhaCd
UVVCxqnB9qU16ea/kB3Yzsrus7egr9OienpDCFV2Q/kgUSc7
-----END CERTIFICATE-----
```


Note

Modify the example above by adding your certificates and key:

- If you renew the certificates, leave your existing key and update the cert and chain sections.
- If you replace the certificates, modify all three sections.

4. Update the /openstack/message_queue.yml file by adding the newly created class to the RabbitMQ nodes:

```
classes:
- service.rabbitmq.server.ssl
- cluster.<cluster_name>.openstack.ssl.rabbitmq
```

5. Log in to the Salt Master node.
6. Refresh pillars:

```
salt -C 'I@rabbitmq:server' saltutil.refresh_pillar
```

7. Publish new certificates

```
salt -C 'I@rabbitmq:server' state.sls rabbitmq -l debug
```

8. Verify the new certificates validity dates:

```
salt -C 'I@rabbitmq:server' cmd.run 'openssl x509 \
-in /etc/rabbitmq/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
```

Example of system response:

```
Not Before: Apr 27 12:37:14 2018 GMT
Not After : Apr 27 12:37:14 2019 GMT
Not Before: Apr 27 12:37:08 2018 GMT
Not After : Apr 27 12:37:08 2019 GMT
```

```
Not Before: Apr 27 12:37:13 2018 GMT
Not After : Apr 27 12:37:13 2019 GMT
```

9. Restart the RabbitMQ services one by one:

```
salt -C 'I@rabbitmq:server' cmd.run 'service rabbitmq-server stop; \
service rabbitmq-server start' -b1
```

10. Verify the RabbitMQ cluster status:

```
salt -C 'I@rabbitmq:server' cmd.run 'rabbitmqctl cluster_status'
```

Example of system response:

```
msg03.multinode-ha.int:
    Cluster status of node rabbit@msg03
    [{nodes,[{disc,[rabbit@msg01,rabbit@msg02,rabbit@msg03]}]},
     {running_nodes,[rabbit@msg01,rabbit@msg02,rabbit@msg03]},
     {cluster_name,<<"openstack">>},
     {partitions,[]},
     {alarms,[{rabbit@msg01,[]},{rabbit@msg02,[]},{rabbit@msg03,[]}]}]
msg01.multinode-ha.int:
    Cluster status of node rabbit@msg01
    [{nodes,[{disc,[rabbit@msg01,rabbit@msg02,rabbit@msg03]}]},
     {running_nodes,[rabbit@msg03,rabbit@msg02,rabbit@msg01]},
     {cluster_name,<<"openstack">>},
     {partitions,[]},
     {alarms,[{rabbit@msg03,[]},{rabbit@msg02,[]},{rabbit@msg01,[]}]}]
msg02.multinode-ha.int:
    Cluster status of node rabbit@msg02
    [{nodes,[{disc,[rabbit@msg01,rabbit@msg02,rabbit@msg03]}]},
     {running_nodes,[rabbit@msg03,rabbit@msg01,rabbit@msg02]},
     {cluster_name,<<"openstack">>},
     {partitions,[]},
     {alarms,[{rabbit@msg03,[]},{rabbit@msg01,[]},{rabbit@msg02,[]}]}]
```

11. Restart all OpenStack API services and agents.

MySQL/Galera certificates

This section describes how to renew or replace the MySQL/Galera certificates managed by either salt-minion or self-managed certificates using pillars.

Verify that the MySQL/Galera cluster uses certificates

This section describes how to determine whether your MySQL/Galera cluster uses certificates and identify their location on the system.

To verify that the MySQL/Galera cluster uses certificates:

1. Log in to the Salt Master node.
2. Run the following command:

   ```
   salt -C 'I@galera:master' mysql.showglobal | grep -EB3 '(have_ssl|ssl_(key|ca|cert))$'
   ```

   Example of system response:

   ```
   Value:
       YES
   Variable_name:
       have_ssl

   Value:
       /etc/mysql/ssl/ca.pem
   Variable_name:
       ssl_ca

   Value:
       /etc/mysql/ssl/cert.pem
   Variable_name:
       ssl_cert

   Value:
       /etc/mysql/ssl/key.pem
   Variable_name:
       ssl_key
   ```
3. Proceed to renewal or replacement of your certificates as required.

Renew or replace the MySQL/Galera certificates managed by salt-minion

This section describes how to renew or replace the MySQL/Galera certificates managed by salt-minion.

Prerequisites:

1. Log in to the Salt Master node.
2. Verify that the MySQL/Galera cluster is up and synced:

   ```
   salt -C 'I@galera:master' mysql.status | grep -EA1 'wsrep_(local_state_c|incoming_a|cluster_size)'
   ```

   Example of system response:

   ```
   wsrep_cluster_size:
       3

   wsrep_incoming_addresses:
       192.168.2.52:3306,192.168.2.53:3306,192.168.2.51:3306

   wsrep_local_state_comment:
       Synced
   ```

3. Verify that the log files have no errors:

   ```
   salt -C 'I@galera:master or I@galera:slave' cmd.run 'cat /var/log/mysql/error.log |grep ERROR|wc -l'
   ```

   Example of system response:

   ```
   dbs01.multinode-ha.int
       0

   dbs02.multinode-ha.int
       0

   dbs03.multinode-ha.int
       0
   ```

   Any value except 0 in the output indicates that the log files include errors. Review them before proceeding to operations with MySQL/Galera.

4. Verify that the ca-salt_master_ca certificate is available on all nodes with MySQL/Galera:

   ```
   salt -C 'I@galera:master or I@galera:slave' cmd.run 'ls /usr/local/share/ca-certificates/ca-salt_master_ca.crt'
   ```

   Example of system response:

```
dbs01.multinode-ha.int
    /usr/local/share/ca-certificates/ca-salt_master_ca.crt

dbs02.multinode-ha.int
    /usr/local/share/ca-certificates/ca-salt_master_ca.crt

dbs03.multinode-ha.int
    /usr/local/share/ca-certificates/ca-salt_master_ca.crt
```

To renew or replace the MySQL/Galera certificates managed by salt-minion:

1. Log in to the Salt Master node.
2. Obtain the list of the Galera cluster minions:

   ```
   salt -C 'I@galera:master or I@galera:slave' pillar.get _nonexistent | cut -d':' -f1
   ```

   Example of system response:

   ```
   dbs02.multinode-ha.int
   dbs03.multinode-ha.int
   dbs01.multinode-ha.int
   ```

3. Verify the certificates validity dates:

   ```
   salt -C 'I@galera:master' cmd.run 'openssl x509 -in /etc/mysql/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
   salt -C 'I@galera:slave' cmd.run 'openssl x509 -in /etc/mysql/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
   ```

   Example of system response:

   ```
   Not Before: May 30 17:21:10 2018 GMT
   Not After : May 30 17:21:10 2019 GMT
   Not Before: May 30 17:25:24 2018 GMT
   Not After : May 30 17:25:24 2019 GMT
   Not Before: May 30 17:26:52 2018 GMT
   Not After : May 30 17:26:52 2019 GMT
   ```

4. Prepare the Galera nodes to work with old one and new Salt Master CA certificates:

   ```
   salt -C 'I@galera:master or I@galera:slave' cmd.run 'cat /usr/local/share/ca-certificates/ca-salt_master_ca.crt /usr/local/share/ca-certificates/ca-salt_master_ca_old.crt > /etc/mysql/ssl/ca.pem'
   ```

5. Verify that the necessary files are present in the ssl directory:

   ```
   salt -C 'I@galera:master or I@galera:slave' cmd.run 'ls /etc/mysql/ssl'
   ```

   Example of system response:

```
dbs01.multinode-ha.int
    ca.pem
    cert.pem
    key.pem

dbs02.multinode-ha.int
    ca.pem
    cert.pem
    key.pem

dbs03.multinode-ha.int
    ca.pem
    cert.pem
    key.pem
```

6. Identify the Galera nodes minions IDs:

    - For the Galera master node:

      ```
      salt -C 'I@galera:master' test.ping --output yaml | cut -d':' -f1
      ```

      Example of system response:

      ```
      dbs01.multinode-ha.int
      ```

    - For the Galera slave nodes:

      ```
      salt -C 'I@galera:slave' test.ping --output yaml | cut -d':' -f1
      ```

      Example of system response:

      ```
      dbs02.multinode-ha.int
      dbs03.multinode-ha.int
      ```

7. Restart the MySQL service for every Galera minion ID one by one. After each Galera minion restart, verify the Galera cluster size and status. Proceed to the next Galera minion restart only if the Galera cluster is synced.

    - To restart the MySQL service for a Galera minion:

      ```
      salt <minion_ID> service.stop mysql
      salt <minion_ID> service.start mysql
      ```

    - To verify the Galera cluster size and status:

```
salt -C 'I@galera:master' mysql.status | grep -EA1 'wsrep_(local_state_c|incoming_a|cluster_size)'
```

Example of system response:

```
wsrep_cluster_size:
    3

wsrep_incoming_addresses:
    192.168.2.52:3306,192.168.2.53:3306,192.168.2.51:3306

wsrep_local_state_comment:
    Synced
```

8. If you replace the certificates, remove the private key:

```
salt -C 'I@galera:master' cmd.run 'mv /etc/mysql/ssl/key.pem /root'
```

9. Force the certificates regeneration for the Galera master node:

```
salt -C 'I@galera:master' cmd.run 'mv /etc/mysql/ssl/cert.pem /root; mv /etc/mysql/ssl/ca.pem /root'

salt -C 'I@galera:master' state.sls salt.minion.cert -l debug

salt -C 'I@galera:master' cmd.run 'cat /usr/local/share/ca-certificates/ca-salt_master_ca.crt /usr/local/share/ca-certificates/ca-salt_master_ca_old.crt > /etc/mysql/ssl/ca.pem'
```

10. Verify that the certificates validity dates have changed:

```
salt -C 'I@galera:master' cmd.run 'openssl x509 -in /etc/mysql/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
```

Example of system response:

```
Not Before: Jun  4 16:14:24 2018 GMT
Not After : Jun  4 16:14:24 2019 GMT
```

11. Verify that the necessary files are present in the ssl directory on the Galera master node:

```
salt -C 'I@galera:master' cmd.run 'ls /etc/mysql/ssl'
```

Example of system response:

```
dbs01.multinode-ha.int
    ca.pem
    cert.pem
    key.pem
```

12. Restart the MySQL service on the Galera master node:

```
salt -C 'I@galera:master' service.stop mysql
salt -C 'I@galera:master' service.start mysql
```

13. Verify that the Galera cluster status is up. For details, see the step 7.
14. If you replace the certificates, remove the private key:

```
salt -C 'I@galera:slave' cmd.run 'mv /etc/mysql/ssl/key.pem /root'
```

15. Force the certificates regeneration for the Galera slave nodes:

```
salt -C 'I@galera:slave' cmd.run 'mv /etc/mysql/ssl/cert.pem /root; mv /etc/mysql/ssl/ca.pem /root'

salt -C 'I@galera:slave' state.sls salt.minion.cert -l debug

salt -C 'I@galera:slave' cmd.run 'cat /usr/local/share/ca-certificates/ca-salt_master_ca.crt /usr/local/share/ca-certificates/ca-salt_master_ca_old.crt > /etc/mysql/ssl/ca.pem'
```

16. Verify that the necessary files are present in the ssl directory on the Galera slave nodes:

```
salt -C 'I@galera:slave' cmd.run 'ls /etc/mysql/ssl'
```

Example of system response:

```
dbs02.multinode-ha.int
    ca.pem
    cert.pem
    key.pem

dbs03.multinode-ha.int
    ca.pem
    cert.pem
    key.pem
```

17. Verify that the certificates validity dates have changed:

```
salt -C 'I@galera:slave' cmd.run 'openssl x509 -in /etc/mysql/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
```

Example of system response:

```
Not Before: Jun  4 16:14:24 2018 GMT
Not After : Jun  4 16:14:24 2019 GMT
Not Before: Jun  4 16:14:31 2018 GMT
Not After : Jun  4 16:14:31 2019 GMT
```

18. Restart the MySQL service for every Galera slave minion ID one by one. After each Galera slave minion restart, verify the Galera cluster size and status. Proceed to the next Galera slave minion restart only if the Galera cluster is synced. For details, see the step 7.

Renew or replace the self-managed MySQL/Galera certificates

This section describes how to renew or replace the self-managed MySQL/Galera certificates.

To renew or replace the self-managed MySQL/Galera certificates:

1. Log in to the Salt Master node.
2. Create the classes/cluster/<cluster_name>/openstack/ssl/galera_master.yml file with the following configuration as an example:

```
classes:
- cluster.<cluster_name>.openstack.ssl
parameters:
  galera:
    master:
      ssl:
        enabled: True
        cacert_chain: ${_param:galera_ssl_cacert_chain}
        key: ${_param:galera_ssl_key}
        cert: ${_param:galera_ssl_cert}
        ca_file: ${_param:mysql_ssl_ca_file}
        key_file: ${_param:mysql_ssl_key_file}
        cert_file: ${_param:mysql_ssl_cert_file}
```

> Note
>
> Substitute <cluster_name> with the appropriate value.

3. Create the classes/cluster/<cluster_name>/openstack/ssl/galera_slave.yml file with the following configuration as an example:

```
classes:
- cluster.<cluster_name>.openstack.ssl
parameters:
  galera:
    slave:
      ssl:
        enabled: True
        cacert_chain: ${_param:galera_ssl_key}
        key: ${_param:galera_ssl_key}
        cert: ${_param:galera_ssl_key}
        ca_file: ${_param:mysql_ssl_ca_file}
        key_file: ${_param:mysql_ssl_key_file}
        cert_file: ${_param:mysql_ssl_cert_file}
```

Note

Substitute <cluster_name> with the appropriate value.

4. Create the classes/cluster/<cluster_name>/openstack/ssl/init.yml file with the following configuration as an example:

```
parameters:
  _param:
    mysql_ssl_key_file: /etc/mysql/ssl/key.pem
    mysql_ssl_cert_file: /etc/mysql/ssl/cert.pem
    mysql_ssl_ca_file: /etc/mysql/ssl/ca.pem
    galera_ssl_cacert_chain: |
      -----BEGIN CERTIFICATE-----
      MIIF0TCCA7mgAwIBAgIJAOkTQnjLz6rEMA0GCSqGSIb3DQEBCwUAMEoxCzAJBgNV
      ...
      RHXc4FoWv9/n8ZcfsqjQCjF3vUUZBB3zdlfLCLJRruB4xxYukc3gFpFLm21+0ih+
      M8IfJ5I=
      -----END CERTIFICATE-----
    galera_ssl_key: |
      -----BEGIN RSA PRIVATE KEY-----
      MIIJKQIBAAKCAgEArVSJ16ePjCik+6bZBzhiu3enXw8R9Ms1k4x57633IX1sEZTJ
      ...
      0VgM2bDSNyUuiwCbOMK0Kyn+wGeHF/jGSbVsxYI4OeLFz8gdVUqm7olJj4j3xemY
      BlWVHRa/dEG1qfSoqFU9+IQTd+U42mtvvH3oJHEXK7WXzborIXTQ/08Ztdvy
      -----END RSA PRIVATE KEY-----
    galera_ssl_cert: |
      -----BEGIN CERTIFICATE-----
      MIIGIDCCBAigAwIBAgIJAJznLlNteaZFMA0GCSqGSIb3DQEBCwUAMEoxCzAJBgNV
      ...
      MfXPTUI+7+5WQLx10yavJ2gOhdyVuDVagfUM4epcriJbACuphDxHj45GINOGhaCd
      UVVCxqnB9qU16ea/kB3Yzsrus7egr9OienpDCFV2Q/kgUSc7
      -----END CERTIFICATE-----
```

Note

Modify the example above by adding your certificates and key:

- If you renew the certificates, leave your existing key and update the cert and chain sections.
- If you replace the certificates, modify all three sections.

5. Update the classes/cluster/<cluster_name>/infra/config.yml file by adding the newly created classes to the database nodes:

```
openstack_database_node01:
  params:
    linux_system_codename: xenial
    deploy_address: ${_param:openstack_database_node01_deploy_address}
  classes:
  - cluster.${_param:cluster_name}.openstack.database_init
  - cluster.${_param:cluster_name}.openstack.ssl.galera_master
openstack_database_node02:
  params:
    linux_system_codename: xenial
    deploy_address: ${_param:openstack_database_node02_deploy_address}
  classes:
  - cluster.${_param:cluster_name}.openstack.ssl.galera_slave
openstack_database_node03:
  params:
    linux_system_codename: xenial
    deploy_address: ${_param:openstack_database_node03_deploy_address}
  classes:
  - cluster.${_param:cluster_name}.openstack.ssl.galera_slave
```

6. Regenerate the Reclass storage:

```
salt-call state.sls reclass.storage -l debug
```

7. Refresh pillars:

```
salt -C 'I@galera:master or I@galera:slave' saltutil.refresh_pillar
```

8. Verify the certificates validity dates:

```
salt -C 'I@galera:master' cmd.run 'openssl x509 \
-in /etc/mysql/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
salt -C 'I@galera:slave' cmd.run 'openssl x509 \
-in /etc/mysql/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
```

Example of system response:

```
Not Before: May 30 17:21:10 2018 GMT
Not After : May 30 17:21:10 2019 GMT
Not Before: May 30 17:25:24 2018 GMT
Not After : May 30 17:25:24 2019 GMT
Not Before: May 30 17:26:52 2018 GMT
Not After : May 30 17:26:52 2019 GMT
```

9. Force the certificate regeneration on the Galera master node:

```
salt -C 'I@galera:master' state.sls galera -l debug
```

10. Verify the new certificates validity dates on the Galera master node:

```
salt -C 'I@galera:master' cmd.run 'openssl x509 \
-in /etc/mysql/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
```

11. Restart the MySQL service on the Galera master node:

```
salt -C 'I@galera:master' service.stop mysql
salt -C 'I@galera:master' service.start mysql
```

12. Verify that the Galera cluster status is up:

```
salt -C 'I@galera:master' mysql.status | \
grep -EA1 'wsrep_(local_state_c|incoming_a|cluster_size)'
```

Example of system response:

```
wsrep_cluster_size:
    3

wsrep_incoming_addresses:
    192.168.2.52:3306,192.168.2.53:3306,192.168.2.51:3306

wsrep_local_state_comment:
    Synced
```

13. Force the certificate regeneration on the Galera slave nodes:

```
salt -C 'I@galera:slave' state.sls galera -l debug
```

14. Verify that the certificates validity dates have changed:

```
salt -C 'I@galera:slave' cmd.run 'openssl x509 \
-in /etc/mysql/ssl/cert.pem -text -noout' | grep -Ei 'after|before'
```

Example of system response:

```
Not Before: Jun  4 16:14:24 2018 GMT
Not After : Jun  4 16:14:24 2019 GMT
Not Before: Jun  4 16:14:31 2018 GMT
Not After : Jun  4 16:14:31 2019 GMT
```

15. Obtain the Galera slave nodes minions IDs:

```
salt -C 'I@galera:slave' test.ping --output yaml | cut -d':' -f1
```

Example of system response:

```
dbs02.multinode-ha.int
dbs03.multinode-ha.int
```

16. Restart the MySQL service for every Galera slave minion ID one by one. After each Galera slave minion restart, verify the Galera cluster size and status. Proceed to the next Galera slave minion restart only if the Galera cluster is synced.

    - To restart the MySQL service for a Galera slave minion:

    ```
    salt <minion_ID> service.stop mysql
    salt <minion_ID> service.start mysql
    ```

    - To verify the Galera cluster size and status:

    ```
    salt -C 'I@galera:master' mysql.status | \
    grep -EA1 'wsrep_(local_state_c|incoming_a|cluster_size)'
    ```

    Example of system response:

    ```
    wsrep_cluster_size:
        3

    wsrep_incoming_addresses:
        192.168.2.52:3306,192.168.2.53:3306,192.168.2.51:3306

    wsrep_local_state_comment:
        Synced
    ```

Barbican certificates

This section describes how to renew certificates in the Barbican service with a configured Dogtag plugin.

Renew Barbican administrator certificates

This section describes how to renew administrator certificates in the Barbican service with a configured Dogtag plugin.

Prerequisites:

1. Log in to the OpenStack secrets storage node (kmn).
2. Obtain the list of certificates:

   ```
   certutil -L -d /root/.dogtag/pki-tomcat/ca/alias/
   ```

   Example of system response:

   ```
   Certificate Nickname   Trust Attributes
                          SSL,S/MIME,JAR/XPI
   caadmin                u,u,u
   ```

3. Note the nickname and attributes of the administrator certificate to renew:

   ```
   caadmin   u,u,u
   ```

4. Review the certificate validity date and note its serial number:

   ```
   certutil -L -d /root/.dogtag/pki-tomcat/ca/alias/ -n "caadmin" | egrep "Serial|Before|After"
   ```

   Example of system response:

   ```
   Serial Number: 6 (0x6)
       Not Before: Tue Apr 26 12:42:31 2022
       Not After : Mon Apr 15 12:42:31 2024
   ```

To renew the Barbican administrator certificate:

1. Log in to the OpenStack secrets storage node (kmn).
2. Obtain the profile template:

   ```
   pki ca-cert-request-profile-show caManualRenewal --output caManualRenewal.xml
   ```

3. Edit the profile template and add the serial number of the certificate to renew to the highlighted lines of the below template:

   ```
   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <CertEnrollmentRequest>
       <Attributes/>
       <ProfileID>caManualRenewal</ProfileID>
   ```

```
    <Renewal>true</Renewal>
    <SerialNumber>6</SerialNumber>          <!--Insert SerialNumber here-->
    <RemoteHost></RemoteHost>
    <RemoteAddress></RemoteAddress>
    <Input id="i1">
       <ClassID>serialNumRenewInputImpl</ClassID>
       <Name>Serial Number of Certificate to Renew</Name>
       <Attribute name="serial_num">
          <Value>6</Value>          <!--Insert SerialNumber here-->
          <Descriptor>
             <Syntax>string</Syntax>
             <Description>Serial Number of Certificate to Renew</Description>
          </Descriptor>
       </Attribute>
    </Input>
</CertEnrollmentRequest>
```

4. Submit the request and note the request ID:

```
pki ca-cert-request-submit caManualRenewal.xml
```

Example of system response:

```
-----------------------------
Submitted certificate request
-----------------------------
  Request ID: 9
  Type: renewal
  Request Status: pending
  Operation Result: success
```

5. Using the password from /root/.dogtag/pki-tomcat/ca/password.conf, approve the request and note the ID of the new certificate:

> Note
>
> During the first run of a system with self-signed certificates you may get a warning informing of an untrusted issuer. In this case, proceed with importing the CA certificate and accept the default CA server URI.

```
pki -d /root/.dogtag/pki-tomcat/ca/alias/ -c rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -n caadmin ca-cert-request-review 9 --action approve
```

Example of system response:

```
-------------------------------
Approved certificate request 10
-------------------------------
  Request ID: 9
  Type: renewal
  Request Status: complete
  Operation Result: success
  Certificate ID: 0x10
```

6. Download the renewed certificate:

```
pki ca-cert-show 0x10 --output ca_admin_new.crt
```

Example of system response:

```
------------------
Certificate "0x10"
------------------
  Serial Number: 0x10
  Issuer: CN=CA Signing Certificate,O=EXAMPLE
  Subject: CN=PKI Administrator,E=caadmin@example.com,O=EXAMPLE
  Status: VALID
  Not Before: Tue Jun 14 12:24:14 UTC 2022
  Not After: Wed Jun 14 12:24:14 UTC 2023
```

7. Add the renewed certificate to the caadmin and kraadmin users in the LADP database:

```
pki -d /root/.dogtag/pki-tomcat/ca/alias/ -c rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -n caadmin ca-user-cert-add --serial 0x10 caadmin
pki -d /root/.dogtag/pki-tomcat/ca/alias/ -c rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -n caadmin kra-user-cert-add --serial 0x10 kraadmin
```

Example of system response:

```
----------------------------------------------------------------------------------------------------------
Added certificate "2;16;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI Administrator,E=caadmin@example.com,O=EXAMPLE"
----------------------------------------------------------------------------------------------------------
  Cert ID: 2;16;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI Administrator,E=caadmin@example.com,O=EXAMPLE
  Version: 2
  Serial Number: 0x10
  Issuer: CN=CA Signing Certificate,O=EXAMPLE
  Subject: CN=PKI Administrator,E=caadmin@example.com,O=EXAMPLE
```

8. Verify that the new certificate is present in the system:

```
ldapsearch -D "cn=Directory Manager" -b "dc=example,dc=com" -w rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da "uid=caadmin"
ldapsearch -D "cn=Directory Manager" -b "o=pki-tomcat-KRA" -w rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da "uid=kraadmin"
```

Example of system response:

```
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope subtree
# filter: uid=caadmin
# requesting: ALL
#

# caadmin, people, example.com
dn: uid=caadmin,ou=people,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: cmsuser
uid: caadmin
sn: caadmin
cn: caadmin
mail: caadmin@example.com
usertype: adminType
userstate: 1
userPassword:: e1NTSEF9QWY5Mys3a2ZHRUh0cHVyMnhVbDNPcVB2TGZoZHREd2Y3ejRhYnc9PQ=
 =
description: 2;6;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI Administrator,E=ca
 admin@example.com,O=EXAMPLE
description: 2;16;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI Administrator,E=c
 aadmin@example.com,O=EXAMPLE
userCertificate:: MIIDnTCCAoWgAwIBAgIBBjANBgkqhkiG9w0BAQsFADAzMRAwDgYDVQQKDAdF
...
userCertificate:: MIIDnTCCAoWgAwIBAgIBEDANBgkqhkiG9w0BAQsFADAzMRAwDgYDVQQKDAdF
...
```

9. Stop the pki-tomcatd service:

```
systemctl stop pki-tomcatd
```

10. Delete the old certificate using the nickname noted in the prerequisite steps:

```
certutil -D -n "caadmin" -d /root/.dogtag/pki-tomcat/ca/alias/
```

11. Import the renewed certificate using the attributes noted in the prerequisite steps:

```
certutil -A -n "caadmin" -t u,u,u -d /root/.dogtag/pki-tomcat/ca/alias/ -a -i ca_admin_new.crt
```

12. Start the pki-tomcatd service:

```
systemctl start pki-tomcatd
```

13. Verify the new certificate:

```
certutil -L -d /root/.dogtag/pki-tomcat/ca/alias/ -n "caadmin" | egrep "Serial|Before|After"
```

Example of system response:

```
Serial Number: 16 (0x10)
    Not Before: Tue Jun 14 12:24:14 2022
    Not After : Wed Jun 14 12:24:14 2023
```

14. Create new ca_admin_cert.p12 and kra_admin_cert.pem files:

    ```
    openssl pkcs12 -in /root/.dogtag/pki-tomcat/ca_admin_cert.p12 -passin pass:rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -passout pass:1234567 -nocerts -out passPrivateKey.pem
    openssl rsa -in passPrivateKey.pem -out "privateKey.pem" -passin pass:1234567
    openssl pkcs12 -export -in ca_admin_new.crt -inkey privateKey.pem -out ca_admin_new.p12 -clcerts -passout pass:rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da
    openssl pkcs12 -in ca_admin_new.p12 -passin pass:rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -out kra_admin_cert_new.pem -nodes
    ```

    You can change the passout and passin parameters for a stronger password pair.

15. Update kra_admin_cert.pem in the barbican and dogtag folders:

    ```
    cp /etc/barbican/kra_admin_cert.pem ./kra_admin_cert_old.pem
    cp kra_admin_cert_new.pem /etc/barbican/kra_admin_cert.pem
    cp kra_admin_cert_new.pem /etc/dogtag/kra_admin_cert.pem
    systemctl restart barbican-worker.service
    systemctl restart apache2
    ```

> Warning
>
> Once you update the cerificate on the master node, replicate the changes to other nodes. To do so, transfer kra_admin_cert_new.pem from the master node to /etc/barbican/kra_admin_cert.pem on other nodes.

©2025, Mirantis Inc.

Renew Barbican system certificates

This section describes how to renew system certificates in the Barbican service with a configured Dogtag plugin.

Prerequisites:

1. Log in to the OpenStack secrets storage node (kmn).
2. Back up the pki-tomcat configuration files:
   - /etc/pki/pki-tomcat/ca/CS.cfg
   - /etc/pki/pki-tomcat/kra/CS.cfg
3. Obtain the list of used certificates:

   ```
   certutil -L -d /etc/pki/pki-tomcat/alias
   ```

   Example of system response:

   ```
   Certificate Nickname                    Trust Attributes
                                           SSL,S/MIME,JAR/XPI

   ocspSigningCert cert-pki-tomcat CA     u,u,u
   subsystemCert cert-pki-tomcat          u,u,u
   storageCert cert-pki-tomcat KRA        u,u,u
   Server-Cert cert-pki-tomcat            u,u,u
   caSigningCert cert-pki-tomcat CA       CTu,Cu,Cu
   auditSigningCert cert-pki-tomcat CA    u,u,Pu
   transportCert cert-pki-tomcat KRA      u,u,u
   auditSigningCert cert-pki-tomcat KRA   u,u,Pu
   ```

   > Note
   >
   > Server-Cert cert-pki-tomcat certificates are unique for each kmn node.
   >
   > To obtain the serial numbers of these certificates, run the following command on the Salt master node:
   >
   > ```
   > salt 'kmn*' cmd.run "certutil -L -d /etc/pki/pki-tomcat/alias -n 'Server-Cert cert-pki-tomcat' | egrep 'Serial|Before|After'"
   > ```
   >
   > Example of system response:
   >
   > ```
   > kmn01.dogtag.local:
   >         Serial Number: 3 (0x3)
   >             Not Before: Mon Dec 19 16:57:18 2022
   >             Not After : Sun Dec 08 16:57:18 2024
   > ```

```
kmn02.dogtag.local:
        Serial Number: 11 (0xb)
            Not Before: Mon Dec 19 17:02:39 2022
            Not After : Sun Dec 08 17:02:39 2024
kmn03.dogtag.local:
        Serial Number: 10 (0xa)
            Not Before: Mon Dec 19 17:00:40 2022
                      Not After : Sun Dec 08 17:00:40 2024
```

Other certificates are the same for all servers:

```
ocspSigningCert cert-pki-tomcat CA
subsystemCert cert-pki-tomcat
storageCert cert-pki-tomcat KRA
caSigningCert cert-pki-tomcat CA
auditSigningCert cert-pki-tomcat CA
transportCert cert-pki-tomcat KRA
auditSigningCert cert-pki-tomcat KRA
```

4. Note the nickname and attributes of the certificate to renew:

```
transportCert cert-pki-tomcat KRA   u,u,u
```

5. Review the certificate validity date and note its serial number:

```
certutil -L -d /etc/pki/pki-tomcat/alias -n "transportCert cert-pki-tomcat KRA" | egrep "Serial|Before|After"
```

Example of system response:

```
Serial Number: 7 (0x7)
    Not Before: Tue Apr 26 12:42:31 2022
    Not After : Mon Apr 15 12:42:31 2024
```

To renew the Barbican system certificate:

1. Log in to the OpenStack secrets storage node (kmn).
2. Obtain the profile template:

```
pki ca-cert-request-profile-show caManualRenewal --output caManualRenewal.xml
```

3. Edit the profile template and add the serial number of the certificate to renew to the highlighted lines of the below template:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CertEnrollmentRequest>
    <Attributes/>
    <ProfileID>caManualRenewal</ProfileID>
    <Renewal>true</Renewal>
    <SerialNumber>6</SerialNumber>          <!--Insert SerialNumber here-->
    <RemoteHost></RemoteHost>
    <RemoteAddress></RemoteAddress>
    <Input id="i1">
        <ClassID>serialNumRenewInputImpl</ClassID>
        <Name>Serial Number of Certificate to Renew</Name>
        <Attribute name="serial_num">
            <Value>6</Value>          <!--Insert SerialNumber here-->
            <Descriptor>
                <Syntax>string</Syntax>
                <Description>Serial Number of Certificate to Renew</Description>
            </Descriptor>
        </Attribute>
    </Input>
</CertEnrollmentRequest>
```

4. Submit the request and note the request ID:

```
pki ca-cert-request-submit caManualRenewal.xml
```

Example of system response:

```
-----------------------------
Submitted certificate request
-----------------------------
  Request ID: 16
  Type: renewal
  Request Status: pending
  Operation Result: success
```

5. Using the password from /root/.dogtag/pki-tomcat/ca/password.conf, approve the request and note the new cerificate ID:

```
pki -d /root/.dogtag/pki-tomcat/ca/alias/ -c rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -n caadmin ca-cert-request-review 16 --action approve
```

Example of system response:

```
-------------------------------
Approved certificate request 16
-------------------------------
  Request ID: 16
  Type: renewal
```

```
  Request Status: complete
  Operation Result: success
  Certificate ID: 0xf
```

6. Download the renewed certificate:

```
pki ca-cert-show 0xf --output kra_transport.crt
```

   Example of system response:

```
-----------------
Certificate "0xf"
-----------------
  Serial Number: 0xf
  Issuer: CN=CA Signing Certificate,O=EXAMPLE
  Subject: CN=DRM Transport Certificate,O=EXAMPLE
  Status: VALID
  Not Before: Fri Jun 10 13:11:50 UTC 2022
  Not After: Thu May 30 13:11:50 UTC 2024
```

> Note
>
> You can also download an old certificate as a backup measure to revert the changes: pki ca-cert-show 0x7 --output kra_old_transport.crt

7. Stop the pki-tomcatd service:

```
systemctl stop pki-tomcatd
```

8. Delete the old certificate using nickname noted in the prerequisite steps:

```
certutil -D -d /etc/pki/pki-tomcat/alias -n 'transportCert cert-pki-tomcat KRA'
```

9. Import the renewed certificate using the attributes noted in the prerequisite steps:

```
certutil -A -d /etc/pki/pki-tomcat/alias -n 'transportCert cert-pki-tomcat KRA' -i kra_transport.crt -t "u,u,u"
```

10. Update the certificate information in /etc/pki/pki-tomcat/kra/CS.cfg with the base64-encoded data of the new certificate (without the header and footer):

> Note
>
> When updating all certificates, update both the /etc/pki/pki-tomcat/kra/CS.cfg and /etc/pki/pki-tomcat/ca/CS.cfg files.

Examples of files containing data about the certificates:

```
/etc/pki/pki-tomcat/ca/CS.cfg

ca.subsystem.cert <- subsystemCert cert-pki-tomcat
ca.ocsp_signing.cert <- ocspSigningCert cert-pki-tomcat CA
ca.sslserver.cert <- Server-Cert cert-pki-tomcat (unique for each kmn server)
ca.signing.cert <- caSigningCert cert-pki-tomcat CA
ca.audit_signing.cert <- auditSigningCert cert-pki-tomcat CA


/etc/pki/pki-tomcat/kra/CS.cfg

kra.subsystem.cert <- subsystemCert cert-pki-tomcat
kra.storage.cert <- storageCert cert-pki-tomcat KRA
kra.sslserver.cert <- Server-Cert cert-pki-tomcat (unique for each kmn server)
kra.transport.cert <- transportCert cert-pki-tomcat KRA
kra.audit_signing.cert <- auditSigningCert cert-pki-tomcat KRA
```

11. If you are updating a transportCert cert-pki-tomcat KRA certificate, also update Barbican Network Security Services database (NSSDB):

```
certutil -L -d /etc/barbican/alias
certutil -L -d /etc/barbican/alias -n "KRA transport cert" | egrep "Serial|Before|After"
certutil -D -d /etc/barbican/alias -n 'KRA transport cert'
certutil -A -d /etc/barbican/alias -n 'KRA transport cert' -i kra_transport.crt -t ,,
```

12. Start the pki-tomcatd service:

```
systemctl start pki-tomcatd
```

13. Verify that the new certificate is used:

```
certutil -L -d /etc/pki/pki-tomcat/alias -n "transportCert cert-pki-tomcat KRA" | egrep "Serial|Before|After"
certutil -L -d /etc/barbican/alias -n "KRA transport cert" | egrep "Serial|Before|After"
```

14. Replicate newly generated certificates to other nodes. If you have updated the Server-Cert cert-pki-tomcat certificates, verify that each kmn node has a unique updated certificate.

©2025, Mirantis Inc.

15. Upload the renewed certificates to the remaining kmn nodes by repeating steps 7-13.

Add a new LDAP user certificate

You may need to add a new certificate for an LDAP user. For example, if the certificate is outdated and you cannot access the subsystem. In this case, you may see the following error message when trying to reach the Key Recovery Authority (KRA) subsystem:

```
pki -d /root/.dogtag/pki-tomcat/ca/alias/ -c rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -n caadmin kra
PKIException: Unauthorized
```

To add a new LDAP user certificate:

1. Log in to the OpenStack secrets storage node (kmn).
2. Obtain information about the new certificate:

   ```
   pki cert-show 16 --encode
   ```

   Example of system response:

   ```
   ------------------
   Certificate "0x10"
   ------------------
     Serial Number: 0x10
     Issuer: CN=CA Signing Certificate,O=EXAMPLE
     Subject: CN=PKI Administrator,E=caadmin@example.com,O=EXAMPLE
     Status: VALID
     Not Before: Tue Jun 14 12:24:14 UTC 2022
     Not After: Wed Jun 14 12:24:14 UTC 2023

   -----BEGIN CERTIFICATE-----
   MIIDnTCCAoWgAwIBAgIBEDANBgkqhkiG9w0BAQsFADAzMRAwDgYDVQQKDAdFWEFN
   UExFMR8wHQYDVQQDDBZDQSBTaWduaW5nIENlcnRpZmljYXRlMB4XDTIyMDYxNDEy
   MjQxNFoXDTIzMDYxNDEyMjQxNFowUjEQMA4GA1UECgwHRVhBTVBMRTEiMCAGCSqG
   SIb3DQEJARYTY2FhZG1pbkBleGFtcGxlLmNvbTEaMBgGA1UEAwwRUEtJIEFkbWlu
   aXN0cmF0b3IwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCv28DjVwwQ
   LIGkmHgL+ySLY/ja8rKAmL+e7wE1sub6fMFBnSNIi3FbX6850/Nx3GgU+IrwS9lw
   vVXArs7Z7Kw/rm29CDrWlC8fWNYzTmQwhgIlccOiOuaa0QktWUuCUyjhDLyU6VGR
   UIUMz4EG7TU7zg71nYrVjR8elKBDS/ol1jq5qymG0IbKCfL6mNhjTVOy5awbW3ja
   bRp6QgAeRvABzF2R9xVee25/E42351lX76fhnoMvyaMeRfu+l3KVaSHNzupljr0G
   No+l4Wfi2LkxxdX435uv8id0o52KzbofjJMaWdoL70rkL/xng/gaWQ4mW0u0cJyo
   +vVdgIWxUcDBAgMBAAGjgZwwgZkwHwYDVR0jBBgwFoAUQaR5K6VfUXLWk25Zs/x/
   elkXnMQwRwYIKwYBBQUHAQEEOzA5MDcGCCsGAQUFBzABhitodHRwOi8va21uMDEu
   cmwtZG9ndGFnLTIubG9jYWw6ODA4MC9jYS9vY3NwMA4GA1UdDwEB/wQEAwIE8DAd
   BgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHAwQwDQYJKoZIhvcNAQELBQADggEB
   ALmwU2uL1tBl2n2kEUaxyrA+GMmFIZg58hS0Wo2c92lhF1pYypRVy44Bf+iOcdix
   CCy1rV0tpf7qng5VjnFq9aEkbQ14Zg+u6oNopZCKBKFD5lLeEu5wlvuQEsTiTay5
   dzaqdZ1nQ5yobyuTuOOepKTbGzVKh1qPCYLGGX6TUzZB8y8ORqgrm9yo1i9BStUS
   zDhisATkGBoltK8zFeNdXfjd91VsaeiLQz4p38kqv05tCHshJNE7SLwkcGOC3bOQ
   O2EEQJ0U+2QTMX2bg+u41TiPYkFeXvyqXHcmnyGnxhGT18TWH48rxGNh53x5qVFr
   ```

```
T8AoLwQvSnmT7CpSeF9ebWw=
-----END CERTIFICATE-----
```

3. Note the certificate data and serial number in decimal (serial number: 0x10 = 16).
4. Verify that the LDAP user does not have a new certificate:

```
ldapsearch -D "cn=Directory Manager" -b "o=pki-tomcat-KRA" -w rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -h kmn01.rl-dogtag-2.local "uid=kraadmin"
```

Example of system response:

```
# extended LDIF
#
# LDAPv3
# base <o=pki-tomcat-KRA> with scope subtree
# filter: uid=kraadmin
# requesting: ALL
#

# kraadmin, people, pki-tomcat-KRA
dn: uid=kraadmin,ou=people,o=pki-tomcat-KRA
objectClass: top
objectClass: person
# extended LDIF
objectClass: organizationalPerson
objectClass: inetOrgPerson
# extended LDIF
objectClass: cmsuser
uid: kraadmin
sn: kraadmin
# extended LDIF
cn: kraadmin
mail: kraadmin@example.com
usertype: adminType
userstate: 1
userPassword:: e1NTSEF9a2N4aUEvS1BzMWtDZ3VYK1hnaGxNa1QwdDk1emhoZk4yL2xvR2c9PQ=
 =
description: 2;6;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI Administrator,E=ca
 admin@example.com,O=EXAMPLE
userCertificate:: MIIDnTCCAoWgAwIBAgIBBjANBgkqhkiG9w0BAQsFADAzMRAwDgYDVQQKDAdF
 WEFNUExFMR8wHQYDVQQDDBZDQSBTaWduaW5nIENlcnRpZmljYXRlMB4XDTIyMDQyNjEyNDEzN1oXD
 TI0MDQxNTEyNDEzN1owUjEQMA4GA1UECgwHRVhBTVBMRTEiMCAGCSqGSIb3DQEJARYTY2FhZG1pbk
 BleGFtcGxlLmNvbTEaMBgGA1UEAwwRUEtJIEFkbWluaXN0cmF0b3IwggEiMA0GCSqGSIb3DQEBAQU
 AA4IBDwAwggEKAoIBAQCv28DjVwwQLIGkmHgL+ySLY/ja8rKAmL+e7wE1sub6fMFBnSNIi3FbX685
 0/Nx3GgU+IrwS9lwvVXArs7Z7Kw/rm29CDrWlC8fWNYzTmQwhgIlccOiOuaa0QktWUuCUyjhDLyU6
 VGRUIUMz4EG7TU7zg71nYrVjR8elKBDS/ol1jq5qymG0IbKCfL6mNhjTVOy5awbW3jabRp6QgAeRv
 ABzF2R9xVee25/E42351lX76fhnoMvyaMeRfu+l3KVaSHNzupljr0GNo+l4Wfi2LkxxdX435uv8id
 0o52KzbofjJMaWdoL70rkL/xng/gaWQ4mW0u0cJyo+vVdgIWxUcDBAgMBAAGjgZwwgZkwHwYDVR0j
```

```
 BBgwFoAUQaR5K6VfUXLWk25Zs/x/elkXnMQwRwYIKwYBBQUHAQEEOzA5MDcGCCsGAQUFBzABhitod
 HRwOi8va21uMDEucmwtZG9ndGFnLTIubG9jYWw6ODA4MC9jYS9vY3NwMA4GA1UdDwEB/wQEAwIE8D
 AdBgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHAwQwDQYJKoZIhvcNAQELBQADggEBAAvXysrUFQT
 gQqQudT7jzxj/X++gNytno0kWOQeIoJSgp0qiz4RFVF/RIF7zn0jMl6a3hipRBU2nU1Fr4De/xcx4
 gPD/MWJquD6bSNywlYCkhxCwf3Z8xwLlyV1pYQ8YQAkVK0S9qLHLgjZdPRuzW3SGpyOevcY9JaLpX
 qaYJ5Tr9fiAcoD8jvf2w0cRmYVw2RELP3ATTrF1V00WnyVwDyda8eNacBxOd831mQOrA9JJm5c/fQ
 cZr0MovXjyU3ddp3MXS4zmTz4skR3qjvHBSRuUuOAvXhnXtP1OzPeLNSGsXozcL/0mqSEQFrV+TiF
 7hVeYF0IGhvkWQOvKdDgZMF8=

# search result
search: 2
result: 0 Success
```


5. Create a modify_kra_admin.ldif file with information about the new certificate. In userCertificate, insert the certificate data and in description insert the certificate ID with a correct serial number in decimal (serial number: 0x10 = 16).

   For example:

```
root@kmn01:~/1# cat modify_kra_admin.ldif

# extended LDIF
#
# LDAPv3
# base <o=pki-tomcat-KRA> with scope subtree
# filter: uid=kraadmin
# requesting: ALL
#

# kraadmin, people, pki-tomcat-KRA
dn: uid=kraadmin,ou=people,o=pki-tomcat-KRA
changetype: modify
add: userCertificate
userCertificate:: MIIDnTCCAoWgAwIBAgIBEDANBgkqhkiG9w0BAQsFADAzMRAwDgYDVQQKDAdF
 WEFNUExFMR8wHQYDVQQDDBZDQSBTaWduaW5nIENlcnRpZmljYXRlMB4XDTIyMDYxNDEyMjQxNFoXD
 TIzMDYxNDEyMjQxNFowUjEQMA4GA1UECgwHRVhBTVBMRTEiMCAGCSqGSIb3DQEJARYTY2FhZG1pbk
 BleGFtcGxlLmNvbTEaMBgGA1UEAwwRUEtJIEFkbWluaXN0cmF0b3IwggEiMA0GCSqGSIb3DQEBAQU
 AA4IBDwAwggEKAoIBAQCv28DjVwwQLIGkmHgL+ySLY/ja8rKAmL+e7wE1sub6fMFBnSNIi3FbX685
 0/Nx3GgU+IrwS9lwvVXArs7Z7Kw/rm29CDrWlC8fWNYzTmQwhgIlccOiOuaa0QktWUuCUyjhDLyU6
 VGRUIUMz4EG7TU7zg71nYrVjR8eIKBDS/ol1jq5qymG0IbKCfL6mNhjTVOy5awbW3jabRp6QgAeRv
 ABzF2R9xVee25/E42351lX76fhnoMvyaMeRfu+l3KVaSHNzupljr0GNo+l4Wfi2LkxxdX435uv8id
 0o52KzbofjJMaWdoL70rkL/xng/gaWQ4mW0u0cJyo+vVdgIWxUcDBAgMBAAGjgZwwgZkwHwYDVR0j
 BBgwFoAUQaR5K6VfUXLWk25Zs/x/elkXnMQwRwYIKwYBBQUHAQEEOzA5MDcGCCsGAQUFBzABhitod
 HRwOi8va21uMDEucmwtZG9ndGFnLTIubG9jYWw6ODA4MC9jYS9vY3NwMA4GA1UdDwEB/wQEAwIE8D
 AdBgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHAwQwDQYJKoZIhvcNAQELBQADggEBALmwU2uL1tB
 l2n2kEUaxyrA+GMmFIZg58hS0Wo2c92IhF1pYypRVy44Bf+iOcdixCCy1rV0tpf7qng5VjnFq9aEk
 bQ14Zg+u6oNopZCKBKFD5lLeEu5wlvuQEsTiTay5dzaqdZ1nQ5yobyuTuOOepKTbGzVKh1qPCYLGG
 X6TUzZB8y8ORqgrm9yo1i9BStUSzDhisATkGBoltK8zFeNdXfjd91VsaeiLQz4p38kqv05tCHshJN
 E7SLwkcGOC3bOQO2EEQJ0U+2QTMX2bg+u41TiPYkFeXvyqXHcmnyGnxhGT18TWH48rxGNh53x5qVF
 rT8AoLwQvSnmT7CpSeF9ebWw=
```

```
dn: uid=kraadmin,ou=people,o=pki-tomcat-KRA
changetype: modify
add: description
description: 2;16;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI Administrator,E=c
 aadmin@example.com,O=EXAMPLE
```

6. Apply the changes:

```
ldapmodify -D "cn=Directory Manager" -w rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -h kmn01.rl-dogtag-2.local -f ./modify_kra_admin.ldif
```

7. Verify that the LDAP user has a new certificate:

```
ldapsearch -D "cn=Directory Manager" -b "o=pki-tomcat-KRA" -w rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -h kmn01.rl-dogtag-2.local "uid=kraadmin"
```

Example of system response:

```
# extended LDIF
#
# LDAPv3
# base <o=pki-tomcat-KRA> with scope subtree
# filter: uid=kraadmin
# requesting: ALL
#

# kraadmin, people, pki-tomcat-KRA
dn: uid=kraadmin,ou=people,o=pki-tomcat-KRA
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: cmsuser
uid: kraadmin
sn: kraadmin
cn: kraadmin
mail: kraadmin@example.com
usertype: adminType
userstate: 1
userPassword:: e1NTSEF9a2N4aUEvS1BzMWtDZ3VYK1hnaGxNa1QwdDk1emhoZk4yL2xvR2c9PQ=
 =
description: 2;6;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI Administrator,E=ca
 admin@example.com,O=EXAMPLE
description: 2;16;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI Administrator,E=c
 aadmin@example.com,O=EXAMPLE
userCertificate:: MIIDnTCCAoWgAwIBAgIBBjANBgkqhkiG9w0BAQsFADAzMRAwDgYDVQQKDAdF
 WEFNUExFMR8wHQYDVQQDDBZDQSBTaWduaW5nIENlcnRpZmljYXRlMB4XDTIyMDQyNjEyNDEzN1oXD
 TI0MDQxNTEyNDEzN1owUjEQMA4GA1UECgwHRVhBTVBMRTEiMCAGCSqGSIb3DQEJARYTY2FhZG1pbk
 BleGFtcGxlLmNvbTEaMBgGA1UEAwwRUEtJIEFkbWluaXN0cmF0b3IwggEiMA0GCSqGSIb3DQEBAQU
```

```
 AA4IBDwAwggEKAoIBAQCv28DjVwwQLIGkmHgL+ySLY/ja8rKAmL+e7wE1sub6fMFBnSNIi3FbX685
 0/Nx3GgU+IrwS9lwvVXArs7Z7Kw/rm29CDrWlC8fWNYzTmQwhgIlccOiOuaa0QktWUuCUyjhDLyU6
 VGRUIUMz4EG7TU7zg71nYrVjR8eIKBDS/ol1jq5qymG0IbKCfL6mNhjTVOy5awbW3jabRp6QgAeRv
 ABzF2R9xVee25/E42351lX76fhnoMvyaMeRfu+l3KVaSHNzupljr0GNo+l4Wfi2LkxxdX435uv8id
 0o52KzbofjJMaWdoL70rkL/xng/gaWQ4mW0u0cJyo+vVdgIWxUcDBAgMBAAGjgZwwgZkwHwYDVR0j
 BBgwFoAUQaR5K6VfUXLWk25Zs/x/elkXnMQwRwYIKwYBBQUHAQEEOzA5MDcGCCsGAQUFBzABhitod
 HRwOi8va21uMDEucmwtZG9ndGFnLTIubG9jYWw6ODA4MC9jYS9vY3NwMA4GA1UdDwEB/wQEAwIE8D
 AdBgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHAwQwDQYJKoZIhvcNAQELBQADggEBAAvXysrUFQT
 gQqQudT7jzxj/X++gNytno0kWOQeIoJSgp0qiz4RFVF/RIF7zn0jMl6a3hipRBU2nU1Fr4De/xcx4
 gPD/MWJquD6bSNywlYCkhxCwf3Z8xwLlyV1pYQ8YQAkVK0S9qLHLgjZdPRuzW3SGpyOevcY9JaLpX
 qaYJ5Tr9fiAcoD8jvf2w0cRmYVw2RELP3ATTrF1V00WnyVwDyda8eNacBxOd831mQOrA9JJm5c/fQ
 cZr0MovXjyU3ddp3MXS4zmTz4skR3qjvHBSRuUuOAvXhnXtP1OzPeLNSGsXozcL/0mqSEQFrV+TiF
 7hVeYF0IGhvkWQOvKdDgZMF8=
userCertificate:: MIIDnTCCAoWgAwIBAgIBEDANBgkqhkiG9w0BAQsFADAzMRAwDgYDVQQKDAdF
 WEFNUExFMR8wHQYDVQQDDBZDQSBTaWduaW5nIENlcnRpZmljYXRlMB4XDTIyMDYxNDEyMjQxNFoXD
 TIzMDYxNDEyMjQxNFowUjEQMA4GA1UECgwHRVhBTVBMRTEiMCAGCSqGSIb3DQEJARYTY2FhZG1pbk
 BleGFtcGxlLmNvbTEaMBgGA1UEAwwRUEtJIEFkbWluaXN0cmF0b3IwggEiMA0GCSqGSIb3DQEBAQU
 AA4IBDwAwggEKAoIBAQCv28DjVwwQLIGkmHgL+ySLY/ja8rKAmL+e7wE1sub6fMFBnSNIi3FbX685
 0/Nx3GgU+IrwS9lwvVXArs7Z7Kw/rm29CDrWlC8fWNYzTmQwhgIlccOiOuaa0QktWUuCUyjhDLyU6
 VGRUIUMz4EG7TU7zg71nYrVjR8eIKBDS/ol1jq5qymG0IbKCfL6mNhjTVOy5awbW3jabRp6QgAeRv
 ABzF2R9xVee25/E42351lX76fhnoMvyaMeRfu+l3KVaSHNzupljr0GNo+l4Wfi2LkxxdX435uv8id
 0o52KzbofjJMaWdoL70rkL/xng/gaWQ4mW0u0cJyo+vVdgIWxUcDBAgMBAAGjgZwwgZkwHwYDVR0j
 BBgwFoAUQaR5K6VfUXLWk25Zs/x/elkXnMQwRwYIKwYBBQUHAQEEOzA5MDcGCCsGAQUFBzABhitod
 HRwOi8va21uMDEucmwtZG9ndGFnLTIubG9jYWw6ODA4MC9jYS9vY3NwMA4GA1UdDwEB/wQEAwIE8D
 AdBgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHAwQwDQYJKoZIhvcNAQELBQADggEBALmwU2uL1tB
 l2n2kEUaxyrA+GMmFIZg58hS0Wo2c92lhF1pYypRVy44Bf+iOcdixCCy1rV0tpf7qng5VjnFq9aEk
 bQ14Zg+u6oNopZCKBKFD5lLeEu5wlvuQEsTiTay5dzaqdZ1nQ5yobyuTuOOepKTbGzVKh1qPCYLGG
 X6TUzZB8y8ORqgrm9yo1i9BStUSzDhisATkGBoltK8zFeNdXfjd91VsaeiLQz4p38kqv05tCHshJN
 E7SLwkcGOC3bOQO2EEQJ0U+2QTMX2bg+u41TiPYkFeXvyqXHcmnyGnxhGT18TWH48rxGNh53x5qVF
 rT8AoLwQvSnmT7CpSeF9ebWw=

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```


8. Verify that the subsystem is accessible:

```
pki -d /root/.dogtag/pki-tomcat/ca/alias/ -c rCWuvkszR4tbiDmMHfpLqJDtVQbHP1da -n caadmin kra
```

Example of system response:

```
Commands:
 kra-group          Group management commands
 kra-key            Key management commands
 kra-selftest       Selftest management commands
 kra-user           User management commands
```

Certificates for Nova compute services

This section describes how to update certificates required for Nova compute services managed by salt-minion.

Update libvirt certificates

This section describes how to update the libvirt certificates managed by salt-minion.

To update the libvirt certificates managed by salt-minion:

1. Log in to the Salt Master node.
2. Create certificate backups for all compute nodes:

   ```
   salt -C 'I@nova:compute' cmd.run 'cp -pr /etc/pki/libvirt-vnc/server-cert.pem /etc/pki/libvirt-vnc/server-cert.pem_$(date +"%Y_%m_%d").bak'
   ```

3. Remove your current certificates from each compute node:

   ```
   salt -C 'I@nova:compute' cmd.run 'rm -rf /etc/pki/libvirt-vnc/server-cert.pem'
   ```

4. Apply the salt.minion.grains state for all compute nodes to retrieve the CA certificate from Salt Master:

   ```
   salt -C 'I@nova:compute' state.sls salt.minion.grains test=true -b 1
   salt -C 'I@nova:compute' state.sls salt.minion.grains -b 1
   ```

5. Apply the salt.minion.cert state for all compute nodes:

   ```
   salt -C 'I@nova:compute' state.sls salt.minion.cert test=true -b 2
   salt -C 'I@nova:compute' state.sls salt.minion.cert -b 2
   ```

6. Restart the libvrit service on one of the compute nodes:

   ```
   salt '*cmp*' cmd.run 'service libvirtd restart'
   ```

7. Verify that the service has restarted successfully:

   ```
   salt '*cmp*' cmd.run 'service libvirtd status'
   ```

8. Restart the libvirt service and apply the nova state on the remaining nova compute nodes:

   ```
   salt -C 'I@nova:compute' cmd.run 'service libvirtd restart' -b 1
   salt -C 'I@nova:compute'  state.sls nova test=true -b 2
   salt -C 'I@nova:compute'  state.sls nova -b 2
   ```

Update nova certificates

This section describes how to update the Nova NoVNCProxy certificates managed by salt-minion.

To update the Nova NoVNCProxy certificates managed by salt-minion:

1. Log in to the Salt Master node.
2. Create certificate backups for all compute nodes:

```
salt '*ctl*' cmd.run 'cp -pr /etc/pki/nova-novncproxy/client-cert.pem /etc/pki/nova-novncproxy/client-cert.pem_$(date +"%Y_%m_%d").bak'
salt '*ctl*' cmd.run 'cp -pr /etc/pki/nova-novncproxy/server-cert.pem /etc/pki/nova-novncproxy/server-cert.pem_$(date +"%Y_%m_%d").bak'
```

3. Remove your current certificates from each compute node:

```
salt '*ctl*' cmd.run 'rm -rf /etc/pki/nova-novncproxy/client-cert.pem /etc/pki/nova-novncproxy/server-cert.pem'
```

4. Apply the salt.minion.grains state for all compute nodes to retrieve the CA certificate from Salt Master:

```
salt '*ctl*' state.sls salt.minion.grains test=true -b 1
salt '*ctl*' state.sls salt.minion.grains -b 1
```

5. Apply the salt.minion.cert state on all compute nodes:

```
salt '*ctl*' state.sls salt.minion.cert test=true -b 2
salt '*ctl*' state.sls salt.minion.cert -b 2
```

6. Restart the nova-novncproxy service:

```
salt '*ctl*' cmd.run 'service nova-novncproxy restart' -b 1
```

Change the certificate validity period

You can change a certificate validity period by managing the validity period of the signing policy, which is used for certificates generation and is set to 365 days by default.

> Note
>
> The procedure does not update the CA certificates and does not change the signing policy itself.

To change the certificate validity period:

1. Log in to the Salt Master node.
2. In classes/cluster/<cluster_name>/infra/config/init.yml, specify the following pillar:

   ```
   parameters:
     _param:
       salt_minion_ca_days_valid_certificate: <required_value>
       qemu_vnc_ca_days_valid_certificate: <required_value>
   ```

3. Apply the changes:

   ```
   salt '*' saltutil.sync_all
   salt -C 'I@salt:master' state.sls salt.minion.ca
   salt -C 'I@salt:master' state.sls salt.minion
   ```

4. Remove the certificate you need to update.
5. Apply the following state:

   ```
   salt -C '<target_node>' state.sls salt.minion.cert
   ```

6. Verify the end date of the updated certificate:

   ```
   salt -C <taget_node> cmd.run 'openssl x509 -enddate -noout -in <path_to_cert>'
   ```

Enable FQDN on internal endpoints in the Keystone catalog

In the new MCP 2019.2.3 deployments, the OpenStack environments use FQDN on the internal endpoints in the Keystone catalog by default.

In the existing MCP deployments, the IP addresses are used on the internal Keystone endpoints. This section instructs you on how to enable FQDN on the internal endpoints for the existing MCP deployments updated to the MCP 2019.2.3 or newer version.

To enable FQDN on the Keystone internal endpoints:

1. Verify that you have updated MCP DriveTrain to the 2019.2.3 or newer version as described in Update DriveTrain.
2. Log in to the Salt Master node.
3. On the system Reclass level:
    1. Verify that there are classes present under the /srv/salt/reclass/classes/system/linux/network/hosts/openstack directory.
    2. Verify that the following parameters are set in defaults/openstack/init.yml as follows:

        ```
        parameters:
          _param:
            openstack_service_hostname: os-ctl-vip
            openstack_service_host: ${_param:openstack_service_hostname}.${linux:system:domain}
        ```

    3. If you have the extra OpenStack services installed, define the additional parameters in defaults/openstack/init.yml as required:
        - For Manila:

            ```
            parameters:
              _param:
                openstack_share_service_hostname: os-share-vip
                openstack_share_service_host: ${_param:openstack_share_service_hostname}.${linux:system:domain}
            ```

        - For Barbican:

            ```
            parameters:
              _param:
                openstack_kmn_service_hostname: os-kmn-vip
                openstack_kmn_service_host: ${_param:openstack_kmn_service_hostname}.${linux:system:domain}
            ```

        - For Tenant Telemetry:

            ```
            parameters:
              _param:
                openstack_telemetry_service_hostname: os-telemetry-vip
                openstack_telemetry_service_host: ${_param:openstack_telemetry_service_hostname}.${linux:system:domain}
            ```

4. On the cluster Reclass level, configure the FQDN on internal endpoints by editing infra/init.yml:

1. Add the following class for the core OpenStack services:

```
classes:
  - system.linux.network.hosts.openstack
```

2. If you have the extra OpenStack services installed, define the additional classes as required:

   - For Manila:

   ```
   classes:
     - system.linux.network.hosts.openstack.share
   ```

   - For Barbican:

   ```
   classes:
     - system.linux.network.hosts.openstack.kmn
   ```

   - For Tenant Telemetry:

   ```
   classes:
     - system.linux.network.hosts.openstack.telemetry
   ```

5. On the cluster Reclass level, define the following parameters in the openstack/init.yml file:

1. Define the following parameters for the core OpenStack services:

```
parameters:
  _param:
    glance_service_host: ${_param:openstack_service_host}
    keystone_service_host: ${_param:openstack_service_host}
    heat_service_host: ${_param:openstack_service_host}
    cinder_service_host: ${_param:openstack_service_host}
    nova_service_host: ${_param:openstack_service_host}
    placement_service_host: ${_param:openstack_service_host}
    neutron_service_host: ${_param:openstack_service_host}
```

2. If you have the extra services installed, define the following parameters as required:

   - For Tenant Telemetry:

   ```
   parameters:
     _param:
       aodh_service_host: ${_param:openstack_telemetry_service_host}
       ceilometer_service_host: ${_param:openstack_telemetry_service_host}
       panko_service_host: ${_param:openstack_telemetry_service_host}
       gnocchi_service_host: ${_param:openstack_telemetry_service_host}
   ```

- For Manila:

```
parameters:
  _param:
    manila_service_host: ${_param:openstack_share_service_host}
```

- For Designate:

```
parameters:
  _param:
    designate_service_host: ${_param:openstack_service_host}
```

- For Barbican:

```
parameters:
  _param:
    barbican_service_host: ${_param:openstack_kmn_service_host}
```

6. Apply the keystone state:

```
salt -C 'I@keystone:server' state.apply keystone
```

7. Log in to one of the OpenStack controller nodes.
8. Verify that the changes have been applied successfully:

```
openstack endpoint list
```

9. If SSL is used on the Keystone internal endpoints:
   1. If Manila or Telemetry is installed:
      1. Log in to the Salt Master node.
      2. Open the Reclass cluster level of your deployment.
      3. For Manila, edit /openstack/share.yml. For example:

```
parameters:
  _param:
    openstack_api_cert_alternative_names: IP:127.0.0.1,IP:${_param:cluster_local_address},IP:${_param:cluster_vip_address},DNS:${linux:system:name},DNS:${linux:network:fqdn},DNS:${_param:cluster_vip_address},DNS:${_param:openstack_share_service_host}
```

      4. For Tenant Telemetry, edit /openstack/telemetry.yml. For example:

```
parameters:
  _param:
    openstack_api_cert_alternative_names: IP:128.0.0.1,IP:${_param:cluster_local_address},IP:${_param:cluster_vip_address},DNS:${linux:system:name},DNS:${linux:network:fqdn},DNS:${_param:cluster_vip_address},DNS:${_param:openstack_telemetry_service_host}
```

   2. Renew the OpenStack API certificates to include FQDN in CommonName (CN) as described in Manage certificates.

Enable Keystone security compliance policies

In the MCP OpenStack deployments, you can enable additional Keystone security compliance features independently of each other based on your corporate security policy. All available features apply only to the SQL back end for the Identity driver. By default, all security compliance features are disabled.

> Note
>
> This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

This section instructs you on how to enable the Keystone security compliance features on an existing MCP OpenStack deployment. For the new deployments, you can configure the compliance features during the Reclass deployment model creation through Model Designer.

Keystone security compliance parameters

Operation	Enable in Keystone for all SQL back-end users	Override settings for specific users
Force the user to change the password upon the first use	change_password_upon_first_use: True Forces the user to change their password upon the first use	ignore_change_password_upon_first_use: True
Configure password expiration	password_expires_days: <NUM> Sets the number of days after which the password would expire	ignore_password_expiry: True
Set an account lockout threshold	lockout_failure_attempts: <NUM> Sets the maximum number of failed authentication attempts lockout_duration: <NUM> Sets the number of minutes (in seconds) after which a user would be locked out	ignore_lockout_failure_attempts: True
Restrict the user from changing their password	N/A	lock_password: True

Configure password strength requirements	password_regex: <STRING> [9] Sets the strength requirements for the passwords password_regex_description: <STRING> Provides the text that describes the password strength requirements. Required if the password_regex is set.	N/A
Disable inactive users	disable_user_account_days_inactive: <NUM> [10] Sets the number of days after which the user would be disabled	N/A
Configure a unique password history	unique_last_password_count: <NUM> Sets the number of passwords for a user that must be unique before an old password can be reused minimum_password_age: <NUM> Sets the number of days for the password to be used before the user can change it	N/A

Warning

9 When enabled, it may affect all operations with Heat. Heat creates its service users with its own regex, which is 32 characters long and contains uppercase and lowercase letters, digits, and special characters such as !, @, #, %, ^, & and *. Therefore, not to affect the Heat operations, verify that your custom value for this option allows such generated passwords. Currently, you cannot override the password regex enforcement in Keystone for a specific user.

©2025, Mirantis Inc.

10	When enabled, it may affect autoscaling and other operations with Heat that require the deferred authentication. If you need to perform such operations in the Heat stack for the first time after deployment upon the defined termination period and the Heat service user created during the deployment has been inactive during this termination period, the Heat service user will be disabled and not able to authenticate. Currently, you cannot override this parameter in Keystone for a specific user.

To enable the security compliance policies:

1. Log in to the Salt Master node.
2. Open your Git project repository with the Reclass model on the cluster level.
3. Open the openstack/control/init.yml file for editing.
4. Configure the security compliance policies for the OpenStack service users as required.

   - For all OpenStack service users. For example:

     ```
     parameters:
       _param:
         openstack_service_user_options:
           ignore_change_password_upon_first_use: True
           ignore_password_expiry: True
           ignore_lockout_failure_attempts: False
           lock_password: False
     ```

   - For specific OpenStack Queens and newer OpenStack releases service users. For example:

     ```
     keystone:
         client:
           resources:
             v3:
               users:
                 cinder:
                   options:
                     ignore_change_password_upon_first_use: True
                     ignore_password_expiry: False
                     ignore_lockout_failure_attempts: False
                     lock_password: True
     ```

   - For specific OpenStack Pike and older OpenStack releases service users. For example:

     ```
     keystone:
        client:
     ```

```
    server:
      identity:
        project:
          service:
            user:
              cinder:
                options:
                  ignore_change_password_upon_first_use: True
                  ignore_password_expiry: False
                  ignore_lockout_failure_attempts: False
                  lock_password: True
```

5. Enable the security compliance features on the Keystone server side by defining the related Keystone sever parameters as required.

   Example configuration:

```
keystone:
  server:
    security_compliance:
      disable_user_account_days_inactive: 90
      lockout_failure_attempts: 5
      lockout_duration: 600
      password_expires_days: 90
      unique_last_password_count: 10
      minimum_password_age: 0
      password_regex: '^(?=.*\d)(?=.*[a-zA-Z]).{7,}$$'
      password_regex_description: 'Your password must contains at least 1 letter, 1 digit, and have a minimum length of 7 characters'
      change_password_upon_first_use: true
```

6. Apply the changes:

```
salt -C 'I@keystone:client' state.sls keystone.client
salt -C 'I@keystone:server' state.sls keystone.server
```

Restrict the VM image policy

This section instructs you on how to restrict Glance, Nova, and Cinder snapshot policy to only allow Administrators to manage images and snapshots in your OpenStack environment.

To configure Administrator only policy:

1. In the /etc/nova directory, create and edit the policy.json for Nova as follows:

```
{
  "os_compute_api:servers:create_image": "rule:admin_api",
  "os_compute_api:servers:create_image:allow_volume_backed": "rule:admin_api",
}
```

2. In the openstack/control.yml file, restrict managing operations by setting the role:admin value for the following parameters for Glance and Cinder:

```
parameters:
  glance:
    server:
      policy:
        add_image: "role:admin"
        delete_image: "role:admin"
        modify_image: "role:admin"
        publicize_image: "role:admin"
        copy_from: "role:admin"
        upload_image: "role:admin"
        delete_image_location: "role:admin"
        set_image_location: "role:admin"
        deactivate: "role:admin"
        reactivate: "role:admin"
  cinder:
    server:
      policy:
        'volume_extension:volume_actions:upload_image': "role:admin"
```

3. Apply the following states:

```
salt 'ctl*' state.sls glance.server,cinder.controller
```

4. Verify that the rules have changed in the states output.
5. If the Comment: State 'keystone_policy.rule_present' was not found in SLS 'glance.server' error occurs, synchronize Salt modules and re-apply the glance.server state:

```
salt 'ctl*' saltutil.sync_all
salt 'ctl*' state.sls glance.server
```

6. To apply the changes, restart the glance-api service:

©2025, Mirantis Inc.

```
salt 'ctl*' service.restart glance-api
```

Configure Neutron OVS

After deploying an OpenStack environment with Neutron Open vSwitch, you may want to enable some of the additional features and configurations that Neutron provides.

This section describes how to enable and operate supported Neutron features.

Configure Neutron Quality of Service (QoS)

Neutron Quality of Service, or QoS, is a Neutron feature that enables OpenStack administrators to limit and prioritize network traffic through a set of policies for better network bandwidth.

MCP supports QoS policies with the following limitations:

- Bandwidth limit for SR-IOV must be specified in Megabits per second (Mbps) and be dividable by 1000 Kilobits per second (Kbps).

  All values lower than 1000 KB per second are rounded up to 1 MB per second. Since float numbers are not supported, all values that cannot be divided by 1000 Kbps chunks are rounded up to the nearest integer Mbps value.
- QoS rules are supported for the egress traffic only.
- The network interface driver must support minimum transmit bandwidth (min_tx_rate).

  Minimum transmit bandwidth is supported by such drivers as QLogic 10 Gigabit Ethernet Driver (qlcnic), BNXT Poll Mode driver (bnxt), and so on. The Intel Linux ixgbe and i40e drivers do not support setting minimum transmit bandwidth.
- No automatic oversubscription protection.

  Since the minimum transmit bandwidth is supported on the hypervisor level, your network is not protected from oversubscription. Total bandwidth on all ports may exceed maximum available bandwidth in the provider’s network.

This section describes how to configure Neutron Quality of Service.

Enable Neutron Quality of Service

By default, Neutron QoS is disabled. You can enable Neutron QoS before or after deploying an OpenStack environment.

To enable Neutron Quality of Service

1. Log in to the Salt Master node.
2. Open the cluster.<cluster-name>.openstack.init.yml file for editing.
3. Set the neutron_qos_enabled parameter to True.

   ```
   parameters:
     _param:
       neutron_enable_qos: True
       ...
   ```

   This turns on the QoS functionality. Depending on the deployment, the command uploads extensions for the openvswitch or/and sriovnicswitch agents.
4. Re-run Salt configuration on the Salt Master node:

   ```
   salt -C 'I@neutron:server' state.sls neutron
   salt -C 'I@neutron:gateway' state.sls neutron
   salt -C 'I@neutron:compute' state.sls neutron
   ```
5. Proceed to Create a QoS policy.

Create a QoS policy

After you enable the Neutron Quality of Service feature, configure a QoS policy to prioritize one type of traffic over the other. This section describes basic operations. For more information, see: OpenStack documentation.

To create a QoS policy:

1. Log in to an OpenStack controller node.
2. Create a QoS policy:

   ```
   neutron qos-policy-create bw-limiter
   ```

3. Add a rule to the QoSpolicy:

   ```
   neutron qos-bandwidth-limit-rule-create bw-limiter --max-kbps 3000000
   ```

4. Apply the QoS policy:

   - To a new network:

     ```
     neutron net-create <network-name> --qos-policy bw-limiter
     ```

   - To an existing network:

     ```
     neutron net-update test --qos-policy bw-limiter
     ```

   - To a new port:

     ```
     neutron port-create test --name sriov_port --binding:vnic_type direct
     --qos-policy bw-limiter
     ```

   - To an existing port:

     ```
     neutron port-update sriov_port --qos-policy bw-limiter
     ```

> Seealso
>
> Applying changes to a QoS policy

Applying changes to a QoS policy

You can update or remove an existing QoS policy.

To update a QoS policy:

1. Log in to an OpenStack controller node.
2. Update the QoS policy using the neutron qos-bandwidth-limit-rule-update <qos-policy-name> command.

   Example:

   ```
   rule_id=`neutron qos-bandwidth-limit-rule-list bw-limiter -f value -c id`
   neutron qos-bandwidth-limit-rule-update bw-limiter $rule_id --max-kbps 200000
   ```

To remove a QoS policy:

1. Log in to an OpenStack controller node.
2. Remove from:

   - A network:

     ```
     neutron net-update <network-name> --no-qos-policy
     ```

   - A port:

     ```
     neutron port-update <sriov_port> --no-qos-policy
     ```

©2025, Mirantis Inc.

Enable network trunking

The Mirantis Cloud Platform supports port trunking which enables you to attach a virtual machine to multiple Neutron networks using VLANs as a local encapsulation to differentiate traffic for each network as it goes in and out of a single virtual machine network interface (VIF).

Using network trunking is particularly beneficial in the following use cases:

- Some applications require connection to hundreds of Neutron networks. To achieve this, you may want to use a single or a few VIFs and VLANs to differentiate traffic for each network rather than having hundreds of VIFs per VM.
- Cloud workloads are often very dynamic. You may prefer to add or remove VLANs rather than to hotplug interfaces in a virtual machine.
- Moving a virtual machine from one network to another without detaching the VIF from the virtual machine.
- A virtual machine may run many containers. Each container may have requirements to be connected to different Neutron networks. Assigning a VLAN or other encapsulation ID for each container is more efficient and scalable than requiring a vNIC per container.
- Some legacy applications that require VLANs to connect to multiple networks.

The current limitation of network trunking support is that MCP supports only Neutron OVS with DPDK and the Open vSwitch firewall driver enabled. Other Neutron ML2 plugins, such as Linux Bridge and OVN, are not supported. If you use security groups and network trunking, MCP automatically enables the native Open vSwitch firewall driver.

To enable network trunking:

1. Log in to the Salt Master node.
2. Open the cluster.<NAME>.openstack.init.yml file for editing.
3. Set the neutron_enable_vlan_aware_vms parameter to True:

   ```
   parameters:
     _param:
       neutron_enable_vlan_aware_vms: True
       ...
   ```

4. Re-run Salt configuration:

   ```
   salt -C 'I@neutron:server' state.sls neutron
   salt -C 'I@neutron:gateway' state.sls neutron
   salt -C 'I@neutron:compute' state.sls neutron
   ```

Seealso

OpenStack documentation

Enable L2 Gateway support

The L2 Gateway (L2GW) plugin for the Neutron service provides the ability to interconnect a given tenant network with a VLAN on a physical switch. The basic components of L2GW include:

- **L2GW Service plugin**

  Residing on a controller node, the L2GW Service plugin notifies the L2GW agent and normal L2 OVS agents running on compute hosts about network events and distributes the VTEP IP address information between them.

- **L2GW agent**

  Running on a network node, the L2GW agent is responsible for connecting to OVSDB server running on a hardware switch and updating the database based on instructions received from the L2GW service plugin.

Before you proceed with the L2GW enablement, verify that the following requirements are met:

- OVSDB Hardware VTEP physical switch enabled
- L2 population mechanism driver enabled

To enable L2GW support:

1. Log in to the Salt Master node.
2. In the classes/cluster/<cluster_name>/openstack/control.yml file of your Reclass model, configure the OpenStack controller nodes by including the service.neutron.control.services.l2gw class.
3. In the classes/cluster/<cluster_name>/openstack/gateway.yml file of your Reclass model, add the Neutron L2GW agent configuration. For example:

   ```
   neutron:
    gateway:
     l2gw:
       enabled: true
       debug: true
       ovsdb_hosts:
        ovsdb1: 10.164.5.253:6622
        ovsdb2: 10.164.5.254:6622
   ```

   > Note
   >
   > ovsdb{1,2}
   >
   > User-defined identifier of a physical switch, which is a name that will be used in the OpenStack database to identify this switch.

4. Apply the neutron state to the server nodes to install the service plugin packages, enable the L2GW service plugin, and update the Neutron database with the new schema:

```
salt -I 'neutron:server' state.sls neutron -b 1
```

5. Apply the neutron state to the gateway nodes to install the L2GW agent packages and configure the OVSDB parameters that include a switch pointer with the IP address and port:

```
salt -I 'neutron:gateway' state.sls neutron
```

6. Verify that the L2GW Neutron service plugin is enabled in your deployment:

    1. Log in to one of the OpenStack controller nodes.
    2. Verify that the following command is executed without errors:

    ```
    neutron l2-gateway-list
    ```

Seealso

L2 Gateway official documentation

Configure BGP VPN

The Mirantis Cloud Platform (MCP) supports the Neutron Border Gateway Protocol (BGP) VPN Interconnection service. The BGP-based IP VPNs are commonly used in the industry, mainly for enterprises.

You can use the BGP VPN Interconnection service in the following typical use case: a tenant has a BGP IP VPN (a set of external sites) already set up outside the data center and wants to be able to trigger the establishment of a connection between VMs and these VPN external sites.

Enable the BGP VPN Interconnection service

If you have an existing BGP IP VPN (a set of external sites) set up outside the data center, you can enable the BGP VPN Interconnection service in MCP to be able to trigger the establishment of connectivity between VMs and these VPN external sites.

The drivers for the BGP VPN Interconnection service include:

- OVS/BaGPipe driver
- OpenContrail driver
- OpenDaylight driver

To enable the BGP VPN Interconnection service:

1. Log in to the Salt Master node.
2. Open the cluster.<cluster_name>.openstack.init.yml file for editing.
3. Set the neutron_enable_bgp_vpn parameter to True.
4. Set the driver neutron_bgp_vpn_driver parameter to one of the following values: bagpipe, opendaylight, opencontrail. For example:

   ```
   parameters:
     _param:
       neutron_enable_bgp_vpn: True
       neutron_bgp_vpn_driver: bagpipe
       ...
   ```

5. Re-apply the Salt configuration:

   ```
   salt -C 'I@neutron:server' state.sls neutron
   ```

For the OpenContrail and OpenDaylight drivers, we assume that the related SDN controllers are already enabled in your MCP cluster. To configure the BaGPipe driver, see Configure the BaGPipe driver for BGP VPN.

Seealso

OpenStack documentation

Configure the BaGPipe driver for BGP VPN

The BaGPipe driver is a lightweight implementation of the BGP-based VPNs used as a reference back end for the Neutron BGP VPN Interconnection service.

For the instruction below, we assume that the Neutron BGP VPN Interconnection service is already enabled on the OpenStack controller nodes. To enable BGP VPN, see Enable the BGP VPN Interconnection service.

To configure the BaGPipe driver:

1. Log in to the Salt Master node.
2. Open the cluster.<cluster_name>.openstack.compute.yml file for editing.
3. Add the following parameters:

   ```
   parameters:
     ...
     neutron:
       compute:
         bgp_vpn:
           enabled: True
           driver: bagpipe
           bagpipe:
             local_address: <IP address used for BGP peerings>
             peers: <IP addresses of BGP peers>
             autonomous_system: <BGP Autonomous System Number>
             enable_rtc: True # Enable RT Constraint (RFC4684)
         backend:
           extension:
             bagpipe_bgpvpn:
               enabled: True
   ```

4. Re-apply the Salt configuration:

   ```
   salt -C 'I@neutron:compute' state.sls neutron
   ```

> **Note**
>
> If BaGPipe is to be enabled on several compute nodes, set up Route Reflector to interconnect those BagPipe instances. For more information, see BGP and Route Reflection.

Seealso

OpenStack documentation

Enable the Networking NW-ODL ML2 plugin

This section explains how to enable the Networking OpenDaylight (NW-ODL) Modular Layer 2 (ML2) plugin for Neutron in your deployment using the Neutron Salt formula, which can install the networking-odl package and enables Neutron to connect to the OpenDaylight controller.

> Note
>
> The procedure assumes that the OpenDaylight controller is already up and running.

To enable the NW-ODL ML2 plugin:

1. Log in to the Salt Master node.
2. Define the OpenDaylight plugin options in the cluster/<cluster_name>/openstack/init.yml file as follows:

   ```
   _param:
     opendaylight_service_host: <ODL_controller_IP>
     opendaylight_router: odl-router_v2 # default
     opendaylight_driver: opendaylight_v2 # default
     provider_mappings: physnet1:br-floating # default
   ```

3. In the cluster/<cluster_name>/openstack/control.yml file of your Reclass model, configure the Neutron server by including the system.neutron.control.opendaylight.cluster class and setting credentials and port of OpenDaylight REST API. For example:

   ```
   classes
   - system.neutron.control.opendaylight.cluster
   parameters:
     neutron:
       server:
         backend:
           rest_api_port: 8282
           user: admin
           password: admin
   ```

4. In the cluster/<cluster_name>/openstack/gateway.yml file of your Reclass model, include the following class:

   ```
   classes
   - service.neutron.gateway.opendaylight.single
   ```

5. In the classes/cluster/<cluster_name>/openstack/compute.yml file of your Reclass model, include the following class:

```
classes
- service.neutron.compute.opendaylight.single
```

6. Apply the configuration changes by executing the neutron state on all nodes with neutron:server, neutron:gateway, and neutron:compute roles:

```
salt -I 'neutron:server' state.sls neutron
salt -I 'neutron:gateway' state.sls neutron
salt -I 'neutron:compute' state.sls neutron
```

Seealso

- Official OpenStack networking-odl documentation

Enable Cross-AZ high availability for Neutron agents

> Note
>
> This feature is available starting from the MCP 2019.2.9 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

The Mirantis Cloud Platform (MCP) supports HA with availability zones for Neutron. An availability zone is defined as an agent attribute on a network node and groups network nodes that run services like DHCP, L3, and others. You can associate an availability zone with resources to ensure the resources become HA.

Availability zones provide an extra layer of protection by segmenting the Neutron service deployment in isolated failure domains. By deploying HA nodes across different availability zones, the network services remain available in case of zone-wide failures that affect the deployment. For details, see OpenStack documentation.

Enable Cross-AZ high availability for DHCP

> Note
>
> This feature is available starting from the MCP 2019.2.9 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes how to enable Cross-AZ high availability for DHCP. As a result, DHCP services will be created in availability zones selected during the network creation.

To enable Cross-AZ high availability for DHCP:

1. Log in to the Salt Master node.
2. In /srv/salt/reclass/classes/cluster/<cluster_name>/openstack/control.yml, set the following parameters:

   ```
   parameters:
     neutron:
       server:
         dhcp_agents_per_network: '2'
         dhcp_load_type: 'networks'
         network_scheduler_driver: neutron.scheduler.dhcp_agent_scheduler.AZAwareWeightScheduler
   ```

3. In /srv/salt/reclass/classes/cluster/<cluster_name>/infra/config/nodes.yml, set the availability_zone parameter for each network or gateway node as required:

   ```
   parameters:
     reclass:
       storage:
         node:
           ...
           openstack_gateway_node<id>:
             parameters:
               neutron:
                 gateway:
                   availability_zone: <az-name>
           openstack_gateway_node<id+1>:
             parameters:
               neutron:
                 gateway:
                   availability_zone: <az-name>
          ...
   ```

4. Apply the changes:

```
salt -C 'I@salt:master' state.sls reclass.storage
salt -C 'I@neutron:server or I@neutron:gateway' saltutil.refresh_pillar
salt -C 'I@neutron:server or I@neutron:gateway' state.sls neutron
```

Enable Cross-AZ high availability for L3 router

> Note
>
> This feature is available starting from the MCP 2019.2.9 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes how to enable Cross-AZ high availability for an L3 router. As a result, the L3 router services will be created in availability zones selected during the router creation.

To enable Cross-AZ high availability for L3 routers:

1. Log in to the Salt Master node.
2. In /srv/salt/reclass/classes/cluster/<cluster_name>/openstack/control.yml, set the following parameters:

   ```
   parameters:
     neutron:
       server:
         router_scheduler_driver: neutron.scheduler.l3_agent_scheduler.AZLeastRoutersScheduler
         max_l3_agents_per_router: '3'
   ```

3. In /srv/salt/reclass/classes/cluster/<cluster_name>/infra/config/nodes.yml, set the availability_zone parameter for each network or gateway node as required:

   ```
   parameters:
     reclass:
       storage:
         node:
           ...
           openstack_gateway_node<id>:
             parameters:
              neutron:
                 gateway:
                   availability_zone: <az-name>
           openstack_gateway_node<id+1>:
             parameters:
               neutron:
                 gateway:
                   availability_zone: <az-name>
          ...
   ```

4. Apply the changes:

```
salt -C 'I@salt:master' state.sls reclass.storage
salt -C 'I@neutron:server or I@neutron:gateway' saltutil.refresh_pillar
salt -C 'I@neutron:server or I@neutron:gateway' state.sls neutron
```

Seealso

Enable monitoring of the Open vSwitch processes

Ironic operations

Ironic is an Administrators only service allowing access to all API requests only to the OpenStack users with the admin or baremetal_admin roles. However, some read-only operations are also available to the users with the baremetal_observer role.

In MCP, Ironic has not been integrated with the OpenStack Dashboard service yet. To manage and use Ironic, perform any required actions either through the Bare Metal service command-line client using the ironic or openstack baremetal commands, from scripts using the ironicclient Python API, or through direct REST API interactions.

Managing and using Ironic include creating suitable images, enrolling bare metal nodes into Ironic and configuring them appropriately, and adding compute flavors that correspond to the available bare metal nodes.

Prepare images for Ironic

To provision bare metal servers using Ironic, you need to create special images and upload them to Glance.

The configuration of images much depends on an actual hardware. Therefore, they cannot be provided as pre-built images, and you must prepare them after you deploy Ironic.

These images include:

- Deploy image that runs the ironic-python-agent required for the deployment and control of bare metal nodes
- User image based on the hardware used in your non-virtualized environment

> Note
>
> This section explains how to create the required images using the diskimage-builder tool.

Prepare deploy images

A deploy image is the image that the bare metal node is PXE-booted into during the image provisioning or node cleaning. It resides in the node's RAM and has a special agent running that the ironic-conductor service communicates with to orchestrate the image provisioning and node cleaning.

Such images must contain drivers for all network interfaces and disks of the bare metal server.

> Note
>
> This section provides example instructions on how to prepare the required images using the diskimage-builder tool. The steps may differ depending on your specific needs and the builder tool. For more information, see Building or downloading a deploy ramdisk image.

To prepare deploy images:

1. Create the required image by typing:

   ```
   diskimage-create <BASE-OS> ironic-agent
   ```

2. Upload the resulting *.kernel and *.initramfs images to Glance as aki and ari images:

   1. To upload an aki image, type:

      ```
      glance image-create --name <IMAGE_NAME> \
          --disk-format aki \
          --container-format aki \
          --file <PATH_TO_IMAGE_KERNEL>
      ```

   2. To upload an ari image, type:

      ```
      glance image-create --name <IMAGE_NAME> \
          --disk-format ari \
          --container-format ari \
          --file <PATH_TO_IMAGE_INITRAMFS>
      ```

Prepare user images

Ironic understands two types of user images that include:

* Whole disk image

  Image of complete operating system with the partition table and partitions
* Partition image

  Image of root partition only, without the partition table. Such images must have appropriate kernel and initramfs images associated with them.

  The partition images can be deployed using one of the following methods:

  * netboot **(default)**

    The node is PXE-booted over network to kernel and ramdisk over TFTP.
  * local boot

    During a deployment, the image is modified on a disk to boot from a local disk. See Ironic Advanced features for details.

User images are deployed in a non-virtualized environment on real hardware servers. Therefore, they require all necessary drivers for a given bare metal server hardware to be included, that are disks, NICs, and so on.

> Note
>
> This section provides example instructions on how to prepare the required images using the diskimage-builder tool. The steps may differ depending on your specific needs and the builder tool. For more information, see the Create and add images to the Image service.

To prepare whole disk images:

Use standard cloud images as whole disk images if they contain all necessary drivers. Otherwise, rebuild a cloud image by typing:

```
diskimage-create <base system> -p <EXTRA_PACKAGE_TO_INSTALL> [-p ..]
```

To prepare partition images for netboot:

1. Use the images from UEC cloud images that have kernel and initramfs as separate images if they contain all the required drivers.
2. If additional drivers are required, rebuild the standard whole disk cloud image adding the packages as follows:

   ```
   diskimage-create <BASE_SYSTEM>> baremetal -p <EXTRA_PACKAGE_TO_INSTALL> [-p ..]
   ```
3. Upload images to Glance in the following formats:

- For an aki image for kernel, type:

```
glance image-create --name <IMAGE_NAME> \
    --disk-format aki \
    --container-format aki \
    --file <PATH_TO_IMAGE_KERNEL>
```

- For an ari image for initramfs, type:

```
glance image-create --name <IMAGE_NAME> \
    --disk-format ari \
    --container-format ari \
    --file <PATH_TO_IMAGE_INITRAMFS>
```

- For a rootfs or whole disk image in the output format (qcow2 by default) specified during rebuild, type:

```
glance image-create --name <IMAGE_NAME> \
    --disk-format <'QCOW2'_FROM_THE_ABOVE_COMMAND> \
    --container-format <'BARE'_FROM_THE_ABOVE_COMMAND> \
    --kernel-id <UUID_OF_UPLOADED_AKI_IMAGE> \
    --ramdisk-id <UUID_OF_UPLOADED_ARI_IMAGE> \
    --file <PATH_TO_IMAGE_KERNEL>
```

> Note
>
> For rootfs images, set the kernel_id and ramdisk_id image properties to UUIDs of the uploaded aki and ari images respectively.

To prepare partition images for local boot:

1. Use the images from UEC cloud images that have kernel and initramfs as separate images if they contain all the required drivers.
2. If additional drivers are required, rebuild the standard whole disk cloud image adding the packages as follows:

> Caution!
>
> Verify that the base operating system has the grub2 package available for installation. And enable it during the rebuild as illustrated in the command below.

```
diskimage-create <BASE_SYSTEM> baremetal grub2 -p <EXTRA_PACKAGE_TO_INSTALL> [-p ..]
```

Seealso

Diskimage-builder Documentation

Add bare metal nodes

This section describes the main steps to enroll a bare metal node to Ironic and make it available for provisioning.

To enroll and configure bare metal nodes:

1. Enroll new nodes to Ironic using the ironic node-create command:

   ```
   ironic node-create \
     --name <node-name> \
     --driver <driver-name> \
     --driver-info deploy_ramdisk=<glance UUID of deploy image ramdisk> \
     --driver-info deploy_kernel=<glance UUID of deploy image kernel> \
     --driver-info ipmi_address=<IPMI address of the node> \
     --driver-info ipmi_username=<username for IPMI> \
     --driver-info ipmi_password=<password for the IPMI user> \
     --property memory_mb=<RAM size of the node in MiB> \
     --property cpus=<Number of CPUs on the node> \
     --property local_gb=<size of node's disk in GiB> \
     --property cpu_arch=<architecture of node's CPU>
   ```

   Where the local_gb property is the size of the biggest disk of the node. We recommend setting it to a 1 GB smaller size than the actual size to accommodate for the partitions table to be created and the extra configuration drive partition.

2. Add ports for the node that correspond to the actual NICs of the node:

   ```
   ironic port-create --node <UUID_OF_IRONIC_NODE> --address <MAC_ADDRESS>
   ```

   Note

   At least one port for the node must be created for the NIC that is attached to the provisioning network and from which the node can boot over PXE.

3. Alternatively, enroll the nodes by adding them to the Reclass model on the cluster level:

   ```
   parameters:
    ironic:
     client:
      enabled: true
      nodes:
       admin_identity:
        - name: <node-name>
         driver: pxe_ipmitool
   ```

```
properties:
  local_gb: <size of node's disk in GiB>
  cpus: <Number of CPUs on the node>
  memory_mb: <RAM size of the node in MiB>
  cpu_arch: <architecture of node's CPU>
driver_info:
  ipmi_username: <username for IPMI>
  ipmi_password: <password for the IPMI user>
  ipmi_address: <IPMI address of the node>
ports:
  - address: <MAC address of the node port1>
  - address: <MAC address of the node port2>
```

Create compute flavors

The appropriately created compute flavors allows for proper compute service scheduling of workloads to bare metal nodes.

To create nova flavors:

1. Create a flavor using the nova flavor-create command:

   ```
   nova flavor-create <FLAVOR_NAME> <UUID_OR_'auto'> <RAM> <DISK> <CPUS>
   ```

   Where RAM, DISK, and CPUS equal to the corresponding properties set on the bare metal nodes.
2. Use the above command to create flavors for each type of bare metal nodes you need to differentiate.

Provision instances

After Ironic nodes, ports, and flavors have been successfully configured, deploy the nova-compute instances to the bare metal nodes using the nova boot command:

```
nova boot <server name> \
    --image <IMAGE_NAME_OR_ID> \
    --flavor <BAREMETAL_FLAVOR_NAME_OR_ID> \
    --nic net-id=<ID_OF_SHARED_BAREMETAL_NETWORK>
```

Enable SSL on Ironic internal API

> Note
>
> This feature is available starting from the MCP 2019.2.6 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

You can enable SSL for all OpenStack components while generating a deployment metadata model using the Model Designer UI before deploying a new OpenStack environment. You can also enable SSL on Ironic internal API on an existing OpenStack environment.

The example instruction below describes the following Ironic configuration:

- The OpenStack Ironic API service runs on the OpenStack ctl nodes.
- The OpenStack Ironic deploy API and conductor services run on the bmt nodes.

You may need to modify this example configuration depending on the needs of your deployment.

To enable SSL on Ironic internal API on an existing MCP cluster:

1. Open your Git project repository with the Reclass model on the cluster level.
2. Modify ./openstack/baremetal.yml as follows:

```
classes:
- system.salt.minion.cert.openstack_api
- system.apache.server.proxy
- system.apache.server.proxy.openstack.ironic
parameters:
  _param:
    apache_proxy_openstack_api_address: ${_param:cluster_baremetal_local_address}
    apache_proxy_openstack_api_host: ${_param:cluster_baremetal_local_address}
    ironic_conductor_api_url_protocol: https
    openstack_api_cert_alternative_names: IP:127.0.0.1,IP:${_param:cluster_baremetal_local_address},IP:${_param:cluster_baremetal_vip_address},DNS:${linux:system:name},DNS:${linux:network:fqdn},DNS:$  {_param:cluster_baremetal_local_address},DNS:${_param:cluster_baremetal_vip_address}
    apache_ssl:
      enabled: true
      authority: "${_param:salt_minion_ca_authority}"
      key_file: ${_param:openstack_api_cert_key_file}
      cert_file: ${_param:openstack_api_cert_cert_file}
      chain_file: ${_param:openstack_api_cert_all_file}
    apache_proxy_openstack_ironic_host: 127.0.0.1
    haproxy_https_check_options:
    - httpchk GET /
    - httpclose
    - tcplog
    haproxy_ironic_deploy_check_params: check inter 10s fastinter 2s downinter 3s rise 3 fall 3 check-ssl verify none
  haproxy:
    proxy:
      listen:
        ironic_deploy:
          type: None
          mode: tcp
          options: ${_param:haproxy_https_check_options}
  ironic:
    api:
      bind:
        address: 127.0.0.1
```

3. Modify ./openstack/control.yml as follows:

```
classes:
- system.apache.server.proxy.openstack.ironic
parameters:
  _param:
    apache_proxy_openstack_ironic_host: 127.0.0.1
    haproxy_ironic_check_params: check inter 10s fastinter 2s downinter 3s rise 3 fall 3 check-ssl verify none
  haproxy:
    proxy:
      listen:
        ironic:
```

```
      type: None
      mode: tcp
      options: ${_param:haproxy_https_check_options}
ironic:
  api:
    bind:
      address: 127.0.0.1
```

4. Modify ./openstack/control/init.yml as follows:

```
parameters:
  _param:
    ironic_service_protocol: ${_param:cluster_internal_protocol}
```

5. Modify ./openstack/init.yml as follows:

```
parameters:
  _param:
    ironic_service_host: ${_param:openstack_service_host}
    ironic_service_protocol: ${_param:cluster_internal_protocol}
```

6. Modify ./openstack/proxy.yml as follows:

```
parameters:
  _param:
    nginx_proxy_openstack_ironic_protocol: https
```

7. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
```

8. Apply the following Salt states:

```
salt 'bmt*' state.apply salt
salt -C 'I@ironic:api' state.apply apache
salt 'prx*' state.apply nginx
salt -C 'I@ironic:api' state.apply haproxy
salt -C 'I@ironic:api' state.apply ironic
```

Enable the networking-generic-switch driver

Note

This feature is available starting from the MCP 2019.2.6 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

Note

This feature is available as technical preview. Use such configuration for testing and evaluation purposes only.

The networking-generic-switch ML2 mechanism driver in Neutron implements the features required for multitenancy support on the Ironic bare metal nodes. This driver requires the corresponding configuration of the Neutron server service.

To enable the networking-generic-switch driver:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. In openstack/control.yml, add pillars for networking-generic-switch using the example below:

```
parameters:
  ...
  neutron:
    server:
      backend:
        mechanism:
          ngs:
            driver: genericswitch
      n_g_s:
        enabled: true
        coordination: # optional
          enabled: true
          backend_url: "etcd3+http://1.2.3.4:2379"
        devices:
          s1brbm:
            options:
              device_type:
                value: netmiko_ovs_linux
              ip:
                value: 1.2.3.4
              username:
                value: ngs_ovs_manager
```

```
        password:
          value: password
```

4. Apply the new configuration for the Neutron server:

```
salt -C 'I@neutron:server' saltutil.refresh_pillar
salt -C 'I@neutron:server' state.apply neutron.server
```

Troubleshoot Ironic

The most possible and typical failures of Ironic are caused by the following peculiarities of the service design:

- Ironic is sensitive to possible time difference between the nodes that host the ironic-api and ironic-conductor services.

  One of the symptoms of time being out of sync is inability to enroll a bare metal node into Ironic with the error message No conductor service registered which supports driver <DRIVER_NAME> Although, the DRIVER_NAME driver is known to be enabled and is shown in the output of the ironic driver-list command.

  To fix the issue, verify that the time is properly synced between the nodes.

- Ironic requires IPMI access credentials for the nodes to have the admin privilege level. Any lower privilege level, for example, engineer precludes Ironic from functioning properly.

Designate operations

After you deploy an MCP cluster that includes Designate, you can start creating DNS zones and zone records as well as configure auto-generation of records in DNS zones.

Create a DNS zone and record

This section describes how to create a DNS zone and a record in the created DNS zone on the MCP cluster where Designate is deployed.

To create a DNS zone and record:

1. Log in to the Salt Master node.
2. Create a test DNS zone called testdomain.tld. by running the following command against one of the controller nodes where Designate is deployed. For example, ctl01.

   ```
   salt 'ctl01*' cmd.run ". /root/keystonercv3; openstack zone create \
   --email dnsmaster@testdomain.tld testdomain.tld."
   ```

   Once the change is applied to one controller node, the updated distributed database replicates this change between all controller nodes.

   Example of system response:

   ```
   ctl01.virtual-mcp-ocata-ovs.local:
    +----------------+--------------------------------------+
    | Field          | Value                                |
    +----------------+--------------------------------------+
    | action         | CREATE                               |
    | attributes     |                                      |
    | created_at     | 2017-08-01T12:25:33.000000           |
    | description    | None                                 |
    | email          | dnsmaster@testdomain.tld             |
    | id             | ce9836a9-ba78-4960-9c89-6a4989a9e095 |
    | masters        |                                      |
    | name           | testdomain.tld.                      |
    | pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
    | project_id     | 49c11a3aa9534d8b897cf06890871840     |
    | serial         | 1501590333                           |
    | status         | PENDING                              |
    | transferred_at | None                                 |
    | ttl            | 3600                                 |
    | type           | PRIMARY                              |
    | updated_at     | None                                 |
    | version        | 1                                    |
    +----------------+--------------------------------------+
   ```

3. Verify that a DNS zone is successfully created and is in the ACTIVE status:

   ```
   salt 'ctl01*' cmd.run ". /root/keystonercv3; openstack zone list"
   ```

   Example of system response:

```
ctl01.virtual-mcp-ocata-ovs.local:
+------------------------------------+----------------+-------+-----------+------+------+
|id                                  |name            |type   |serial     |status|action|
+------------------------------------+----------------+-------+-----------+------+------+
|571243e5-17dd-49bd-af09-de6b0c175d8c|example.tld.    |PRIMARY| 1497877051|ACTIVE|NONE  |
|7043de84-3a40-4b44-ad4c-94dd1e802370|domain.tld.     |PRIMARY| 1498209223|ACTIVE|NONE  |
|ce9836a9-ba78-4960-9c89-6a4989a9e095|testdomain.tld.|PRIMARY| 1501590333|ACTIVE|NONE  |
+------------------------------------+----------------+-------+-----------+------+------+
```

4. Create a record in the new DNS zone by running the command below. Use any IPv4 address to test that it works. For example, 192.168.0.1.

```
salt 'ctl01*' cmd.run ". /root/keystonercv3; openstack recordset create \
--records '192.168.0.1' --type A testdomain.tld. tstserver01"
```

Example of system response:

```
ctl01.virtual-mcp-ocata-ovs.local:
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2017-08-01T12:28:37.000000           |
| description | None                                 |
| id          | d099f013-460b-41ee-8cf1-3cf0e3c49bc7 |
| name        | tstserver01.testdomain.tld.          |
| project_id  | 49c11a3aa9534d8b897cf06890871840     |
| records     | 192.168.0.1                          |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | ce9836a9-ba78-4960-9c89-6a4989a9e095 |
| zone_name   | testdomain.tld.                      |
+-------------+--------------------------------------+
```

5. Verify that the record is successfully created and is in the ACTIVE status by running the openstack recordset list [zone_id] command. The zone_id parameter can be found in the output of the command described in the previous step.

   Example:

```
salt 'ctl01*' cmd.run ". /root/keystonercv3; openstack recordset list \
ce9836a9-ba78-4960-9c89-6a4989a9e095"

ctl01.virtual-mcp-ocata-ovs.local:
+---+---------------------------+----+---------------------------------------------------------+------+------+
```

```
| id| name                     |type|records                                                   |status|action|
+---+--------------------------+----+----------------------------------------------------------+------+------+
|...|testdomain.tld.           |SOA |ns1.example.org. dnsmaster.testdomain.tld. 1501590517 3598|ACTIVE|NONE  |
|...|testdomain.tld.           |NS  |ns1.example.org.                                          |ACTIVE|NONE  |
|...|tstserver01.testdomain.tld.|A   |192.168.0.1                                               |ACTIVE|NONE  |
+---+--------------------------+----+----------------------------------------------------------+------+------+
```

6. Verify that the DNS record can be resolved by running the nslookup tstserver01.domain.tld [dns server address] command. In the example below, the DNS server address of the Designate back end is 10.0.0.1.

   Example:

```
nslookup tstserver01.testdomain.tld 10.0.0.1

Server:     10.0.0.1
Address:    10.0.0.1#53
Name:   tstserver01.testdomain.tld
Address: 192.168.0.1
```

Seealso

Configure auto-generation of records in a DNS zone

Configure auto-generation of records in a DNS zone

After you create a DNS zone and a record for this zone as described in Create a DNS zone and record, you can configure auto-generation of records in the created DNS zone.

To configure auto-generation of records in the created DNS zone:

1. In your Git project repository, change the directory to classes/cluster/<cluster_name>/openstack/.
2. In init.yml, set the designate_domain_id parameter according to the created DNS zone. For example:

   ```
   designate_domain_id: ce9836a9-ba78-4960-9c89-6a4989a9e095
   ```

3. Refresh pillars on the Salt Minion nodes:

   ```
   salt '*' saltutil.pillar_refresh
   ```

4. Apply the Designate states:

   ```
   salt -C 'I@designate:server and *01*' state.sls designate.server
   salt -C 'I@designate:server' state.sls designate
   ```

5. Using the Nova CLI, boot the VM which you have created a DNS zone for.
6. Verify that the DNS record related to the VM was created by running the salt 'ctl01*' cmd.run "openstack recordset list [zone_id]" command. For example:

   ```
   salt 'ctl01*' cmd.run ". /root/keystonercv3; openstack recordset list \
   ce9836a9-ba78-4960-9c89-6a4989a9e095"
   ```

   Example of system response:

   ```
   ctl01.virtual-mcp-ocata-ovs.local:
   +------------------------------------+---------------------------+-----+-----------+-------+------+
   |id                                  |name                       |type|records    |status|action|
   +------------------------------------+---------------------------+-----+-----------+-------+------+
   |d099f013-460b-41ee-8cf1-3cf0e3c49bc7|tstserver01.testdomain.tld.|A   |192.168.0.1|ACTIVE|NONE  |
   +------------------------------------+---------------------------+-----+-----------+-------+------+
   ```

Ceph operations

Ceph is a storage back end for cloud environments. After you successfully deploy a Ceph cluster, you can manage its nodes and object storage daemons (Ceph OSDs). This section describes how to add Ceph Monitor, Ceph OSD, and RADOS Gateway nodes to an existing Ceph cluster or remove them, as well as how to remove or replace Ceph OSDs.

Prerequisites

Before you proceed to manage Ceph nodes and OSDs, or upgrade Ceph, perform the steps below.

1. Verify that your Ceph cluster is up and running.
2. Log in to the Salt Master node.
3. Add Ceph pipelines to DriveTrain.
    1. Add the following class to the cluster/cicd/control/leader.yml file:

        ```
        classes:
        - system.jenkins.client.job.ceph
        ```

    2. Apply the salt -C 'I@jenkins:client' state.sls jenkins.client state.

Manage Ceph nodes

This section describes how to add Ceph Monitor, Ceph OSD, and RADOS Gateway nodes to an existing Ceph cluster or remove them.

Add a Ceph Monitor node

This section describes how to add a Ceph Monitor node to an existing Ceph cluster.

> Warning
>
> Prior to the 2019.2.10 maintenance update, this feature is available as technical preview only.

> Note
>
> The Ceph Monitor service is quorum-based. Therefore, keep an odd number of Ceph Monitor nodes to establish a quorum.

To add a Ceph Monitor node:

1. In your project repository, add the following lines to the cluster/ceph/init.yml file and modify them according to your environment:

   ```
   _param:
      ceph_mon_node04_hostname: cmn04
      ceph_mon_node04_address: 172.16.47.145
      ceph_mon_node04_ceph_public_address: 10.13.0.4
      ceph_mon_node04_deploy_address: 192.168.0.145
   linux:
     network:
       host:
         cmn04:
           address: ${_param:ceph_mon_node04_address}
           names:
           - ${_param:ceph_mon_node04_hostname}
           - ${_param:ceph_mon_node04_hostname}.${_param:cluster_domain}
   ```

   > Note
   >
   > Skip the ceph_mon_node04_deploy_address parameter if you have DHCP enabled on a PXE network.

2. Define the backup configuration for the new node in cluster/ceph/init.yml. For example:

```
parameters:
  _param:
    ceph_mon_node04_ceph_backup_hour: 4
    ceph_mon_node04_ceph_backup_minute: 0
```

3. Add the following lines to the cluster/ceph/common.yml file and modify them according to your environment:

```
parameters:
  ceph:
    common:
      members:
        - name: ${_param:ceph_mon_node04_hostname}
          host: ${_param:ceph_mon_node04_address}
```

4. Add the following lines to the cluster/infra/config/nodes.yml file:

```
parameters:
  reclass:
    storage:
      node:
        ceph_mon_node04:
          name: ${_param:ceph_mon_node04_hostname}
          domain: ${_param:cluster_domain}
          classes:
          - cluster.${_param:cluster_name}.ceph.mon
          params:
            ceph_public_address: ${_param:ceph_mon_node04_ceph_public_address}
            ceph_backup_time_hour: ${_param:ceph_mon_node04_ceph_backup_hour}
            ceph_backup_time_minute: ${_param:ceph_mon_node04_ceph_backup_minute}
            salt_master_host: ${_param:reclass_config_master}
            linux_system_codename: ${_param:ceph_mon_system_codename}
            single_address: ${_param:ceph_mon_node04_address}
            deploy_address: ${_param:ceph_mon_node04_deploy_address}
            ceph_public_address: ${_param:ceph_mon_node04_public_address}
            keepalived_vip_priority: 104
```

> Note
>
> Skip the deploy_address parameter if you have DHCP enabled on a PXE network.

5. Add the following lines to the cluster/infra/kvm.yml file and modify infra_kvm_node03_hostname depending on which KVM node the Ceph Monitor node should run on:

```
parameters:
  salt:
    control:
      size:
        ceph.mon:
          cpu: 8
          ram: 16384
          disk_profile: small
          net_profile: default
      cluster:
        internal:
          node:
            cmn04:
              name: ${_param:ceph_mon_node04_hostname}
              provider: ${_param:infra_kvm_node03_hostname}.${_param:cluster_domain}
              image: ${_param:salt_control_xenial_image}
              size: ceph.mon
```

6. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
```

7. Log in to the Jenkins web UI.
8. Open the Ceph - add node pipeline.
9. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	Add the Salt target name of the new Ceph Monitor node. For example, cmn04*.
HOST_TYPE Removed since 2019.2.13 update	Add mon as the type of Ceph node that is going to be added.

10. Click Deploy.

The Ceph - add node pipeline workflow:

1. Launch the Ceph Monitor VMs.
2. Run the reclass state.
3. Run the linux, openssh, salt, ntp, rsyslog, ceph.mon states.

4. Update ceph.conf files on all Ceph nodes.
5. Run the ceph.mgr state if the pillar is present.

Add a Ceph OSD node

This section describes how to add a Ceph OSD node to an existing Ceph cluster.

> Warning
>
> Prior to the 2019.2.10 maintenance update, this feature is available as technical preview only.

To add a Ceph OSD node:

1. Connect the Ceph OSD salt-minion node to salt-master.
2. In your project repository, if the nodes are not generated dynamically, add the following lines to cluster/ceph/init.yml and modify according to your environment:

   ```
   _param:
      ceph_osd_node05_hostname: osd005
      ceph_osd_node05_address: 172.16.47.72
      ceph_osd_node05_backend_address: 10.12.100.72
      ceph_osd_node05_public_address: 10.13.100.72
      ceph_osd_node05_deploy_address: 192.168.0.72
      ceph_osd_system_codename: xenial
   linux:
     network:
       host:
         osd005:
           address: ${_param:ceph_osd_node05_address}
           names:
           - ${_param:ceph_osd_node05_hostname}
           - ${_param:ceph_osd_node05_hostname}.${_param:cluster_domain}
   ```

   > Note
   >
   > Skip the ceph_osd_node05_deploy_address parameter if you have DHCP enabled on a PXE network.

3. If the nodes are not generated dynamically, add the following lines to the cluster/infra/config/nodes.yml and modify according to your environment. Otherwise, increase the number of generated OSDs.

   ```
   parameters:
     reclass:
       storage:
   ```

```
    node:
      ceph_osd_node05:
        name: ${_param:ceph_osd_node05_hostname}
        domain: ${_param:cluster_domain}
        classes:
        - cluster.${_param:cluster_name}.ceph.osd
        params:
          salt_master_host: ${_param:reclass_config_master}
          linux_system_codename:  ${_param:ceph_osd_system_codename}
          single_address: ${_param:ceph_osd_node05_address}
          deploy_address: ${_param:ceph_osd_node05_deploy_address}
          backend_address: ${_param:ceph_osd_node05_backend_address}
          ceph_public_address: ${_param:ceph_osd_node05_public_address}
          ceph_crush_parent: rack02
```

> Note
>
> Skip the deploy_address parameter if you have DHCP enabled on a PXE network.

4. Since 2019.2.3, skip this step Verify that the cluster/ceph/osd.yml file and the pillar of the new Ceph OSD do not contain the following lines:

```
parameters:
  ceph:
    osd:
      crush_update: false
```

5. Log in to the Jenkins web UI.
6. Select from the following options:
   - For MCP versions starting from the 2019.2.10 maintenance update, open the Ceph - add osd (upmap) pipeline.
   - For MCP versions prior to the 2019.2.10 maintenance update, open the Ceph - add node pipeline.

> Note
>
> Prior to the 2019.2.10 maintenance update, the Ceph - add node and Ceph - add osd (upmap) Jenkins pipeline jobs are available as technical preview only.

©2025, Mirantis Inc.

Caution!

A large change in the crush weights distribution after the addition of Ceph OSDs can cause massive unexpected rebalancing, affect performance, and in some cases can cause data corruption. Therefore, if you are using Ceph - add node, Mirantis recommends that you add all disks with zero weight and reweight them gradually.

7. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	Add the Salt target name of the new Ceph OSD. For example, osd005*.
HOST_TYPE Removed since 2019.2.3 update	Add osd as the type of Ceph node that is going to be added.
CLUSTER_FLAGS Added since 2019.2.7 update	Add a comma-separated list of flags to check after the pipeline execution.
USE_UPMAP Added since 2019.2.13 update	Use to facilitate the upmap module during rebalancing to minimize impact on cluster performance.

8. Click Deploy.

   The Ceph - add node pipeline workflow prior to the 2019.2.3 maintenance update:

   1. Apply the reclass state.
   2. Apply the linux, openssh, salt, ntp, rsyslog, ceph.osd states.

   The Ceph - add node pipeline workflow starting from 2019.2.3 maintenance update:

   1. Apply the reclass state.
   2. Verify that all installed Ceph clients have the Luminous version.
   3. Apply the linux, openssh, salt, ntp, rsyslog, states.
   4. Set the Ceph cluster compatibility to Luminous.
   5. Switch the balancer module to the upmap mode.
   6. Set the norebalance flag before adding a Ceph OSD.
   7. Apply the ceph.osd state on the selected Ceph OSD node.

8. Update the mappings for the remapped placement group (PG) using upmap back to the old Ceph OSDs.
9. Unset the norebalance flag and verify that the cluster is healthy.

9. If you use a custom CRUSH map, update the CRUSH map:
   1. Verify the updated /etc/ceph/crushmap file on cmn01. If correct, apply the CRUSH map using the following commands:

      ```
      crushtool -c /etc/ceph/crushmap -o /etc/ceph/crushmap.compiled
      ceph osd setcrushmap -i /etc/ceph/crushmap.compiled
      ```

   2. Add the following lines to the cluster/ceph/osd.yml file:

      ```
      parameters:
        ceph:
          osd:
            crush_update: false
      ```

   3. Apply the ceph.osd state to persist the CRUSH map:

      ```
      salt -C 'I@ceph:osd' state.sls ceph.osd
      ```

10. Integrate the Ceph OSD nodes with StackLight:
    1. Update the Salt mine:

       ```
       salt -C 'I@ceph:osd or I@telegraf:remote_agent' state.sls salt.minion.grains
       salt -C 'I@ceph:osd or I@telegraf:remote_agent' saltutil.refresh_modules
       salt -C 'I@ceph:osd or I@telegraf:remote_agent' mine.update
       ```

       Wait for one minute.
    2. Apply the following states:

       ```
       salt -C 'I@ceph:osd or I@telegraf:remote_agent' state.sls telegraf
       salt -C 'I@ceph:osd' state.sls fluentd
       salt 'mon*' state.sls prometheus
       ```

Add a RADOS Gateway node

This section describes how to add a RADOS Gateway (rgw) node to an existing Ceph cluster.

To add a RADOS Gateway node:

1. In your project repository, add the following lines to the cluster/ceph/init.yml and modify them according to your environment:

```yaml
_param:
  ceph_rgw_node04_hostname: rgw04
  ceph_rgw_node04_address: 172.16.47.162
  ceph_rgw_node04_ceph_public_address: 10.13.0.162
  ceph_rgw_node04_deploy_address: 192.168.0.162
linux:
  network:
    host:
      rgw04:
        address: ${_param:ceph_rgw_node04_address}
        names:
        - ${_param:ceph_rgw_node04_hostname}
        - ${_param:ceph_rgw_node04_hostname}.${_param:cluster_domain}
```

   > Note
   >
   > Skip the ceph_rgw_node04_deploy_address parameter if you have DHCP enabled on a PXE network.

2. Add the following lines to the cluster/ceph/rgw.yml file:

```yaml
parameters:
  _param:
    cluster_node04_hostname: ${_param:ceph_rgw_node04_hostname}
    cluster_node04_address: ${_param:ceph_rgw_node04_address}
  ceph:
    common:
      keyring:
        rgw.rgw04:
          caps:
            mon: "allow rw"
            osd: "allow rwx"
  haproxy:
    proxy:
      listen:
        radosgw:
          servers:
```

```
      - name: ${_param:cluster_node04_hostname}
        host: ${_param:cluster_node04_address}
        port: ${_param:haproxy_radosgw_source_port}
        params: check
```

> Note
>
> Starting from the MCP 2019.2.10 maintenance update, the capabilities for RADOS Gateway have been restricted. To update the existing capabilities, perform the steps described in Restrict the RADOS Gateway capabilities.

3. Add the following lines to the cluster/infra/config/init.yml file:

```
parameters:
  reclass:
    storage:
      node:
        ceph_rgw_node04:
          name: ${_param:ceph_rgw_node04_hostname}
          domain: ${_param:cluster_domain}
          classes:
          - cluster.${_param:cluster_name}.ceph.rgw
          params:
            salt_master_host: ${_param:reclass_config_master}
            linux_system_codename:  ${_param:ceph_rgw_system_codename}
            single_address: ${_param:ceph_rgw_node04_address}
            deploy_address: ${_param:ceph_rgw_node04_deploy_address}
            ceph_public_address: ${_param:ceph_rgw_node04_ceph_public_address}
            keepalived_vip_priority: 104
```

> Note
>
> Skip the deploy_address parameter if you have DHCP enabled on a PXE network.

4. Add the following lines to the cluster/infra/kvm.yml file and modify infra_kvm_node03_hostname depending on which KVM node the rgw must be running on:

```
parameters:
  salt:
    control:
      size:
```

```
    ceph.rgw:
      cpu: 8
      ram: 16384
      disk_profile: small
      net_profile: default
  cluster:
    internal:
      node:
        rgw04:
          name: ${_param:ceph_rgw_node04_hostname}
          provider: ${_param:infra_kvm_node03_hostname}.${_param:cluster_domain}
          image: ${_param:salt_control_xenial_image}
          size: ceph.rgw
```

5. Log in to the Jenkins web UI.
6. Open the Ceph - add node pipeline.
7. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	Add the Salt target name of the new RADOS Gateway node. For example, rgw04*.
HOST_TYPE Removed since 2019.2.13 update	Add rgw as the type of Ceph node that is going to be added.

8. Click Deploy.

The Ceph - add node pipeline workflow:

1. Launch RADOS Gateway VMs.
2. Run the reclass state.
3. Run the linux, openssh, salt, ntp, rsyslog, keepalived, haproxy, ceph.radosgw states.

©2025, Mirantis Inc.

Add a Ceph OSD daemon

This section describes how to add new or re-add the existing Ceph OSD daemons on an existing Ceph OSD node.

> Note
>
> This feature is available starting from the MCP 2019.2.13 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

> Note
>
> The pipeline used in this section is a wrapper for Ceph - add node, which simplifies common operations.

To add a new or re-add the existing Ceph OSD daemon:

1. If you are adding a new Ceph OSD daemon, perform the following prerequisite steps. Otherwise, proceed to the next step.
    1. Open your Git project repository with the Reclass model on the cluster level.
    2. In cluster/ceph/osd.yml, add the new Ceph OSD daemon definition. Use the existing definition as a template.
2. Log in to the Jenkins web UI.
3. Select from the following options:
    - For MCP version 2019.2.13, open the Ceph - add osd (upmap) Jenkins pipeline job.
    - For MCP versions starting from 2019.2.14, open the Ceph - add osd Jenkins pipeline job.
4. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	The Salt target name of the host to which the Ceph OSD daemons are going to be added. For example, osd005*.
CLUSTER_FLAGS	A comma-separated list of flags to check after the pipeline execution.

5. Click Deploy.

   The Ceph - add osd pipeline runs Ceph - add node with the following predefined values:

   - OSD_ONLY is set to True to omit enforcing the node configuration because a working node is already configured in the cluster.
   - USE_UPMAP is set to True to gradually add Ceph OSD daemons to the cluster and prevent consuming excessive I/O for rebalancing.

Remove a Ceph Monitor node

This section describes how to remove a Ceph Monitor node from a Ceph cluster.

> Note
>
> The Ceph Monitor service is quorum-based. Therefore, keep an odd number of Ceph Monitor nodes to establish a quorum.

To remove a Ceph Monitor node:

1. In your project repository, remove the following lines from the cluster/infra/config/init.yml file or from the pillar based on your environment:

   ```
   parameters:
     reclass:
       storage:
         node:
           ceph_mon_node04:
             name: ${_param:ceph_mon_node04_hostname}
             domain: ${_param:cluster_domain}
             classes:
             - cluster.${_param:cluster_name}.ceph.mon
             params:
               salt_master_host: ${_param:reclass_config_master}
               linux_system_codename:  ${_param:ceph_mon_system_codename}
               single_address: ${_param:ceph_mon_node04_address}
               keepalived_vip_priority: 104
   ```

2. Remove the following lines from the cluster/ceph/common.yml file or from the pillar based on your environment:

   ```
   parameters:
     ceph:
       common:
         members:
           - name: ${_param:ceph_mon_node04_hostname}
             host: ${_param:ceph_mon_node04_address}
   ```

3. Log in to the Jenkins web UI.
4. Open the Ceph - remove node pipeline.
5. Specify the following parameters:

Parameter	Description and values

SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	Add the Salt target name of the Ceph Monitor node to remove. For example, cmn04*.
HOST_TYPE Removed since 2019.2.13 update	Add mon as the type of Ceph node that is going to be removed.

6. Click Deploy.

   The Ceph - remove node pipeline workflow:

   1. Reconfigure the configuration file on all ceph:common minions.
   2. Destroy the VM.
   3. Remove the Salt Minion node ID from salt-key on the Salt Master node.

7. Remove the following lines from the cluster/infra/kvm.yml file or from the pillar based on your environment:

```
parameters:
  salt:
    control:
      cluster:
        internal:
          node:
            cmn04:
              name: ${_param:ceph_mon_node04_hostname}
              provider: ${_param:infra_kvm_node03_hostname}.${_param:cluster_domain}
              image: ${_param:salt_control_xenial_image}
              size: ceph.mon
```

8. Remove the following lines from the cluster/ceph/init.yml file or from the pillar based on your environment:

```
_param:
   ceph_mon_node04_hostname: cmn04
   ceph_mon_node04_address: 172.16.47.145
linux:
  network:
    host:
      cmn04:
        address: ${_param:ceph_mon_node04_address}
        names:
        - ${_param:ceph_mon_node04_hostname}
        - ${_param:ceph_mon_node04_hostname}.${_param:cluster_domain}
```

Remove a Ceph OSD node

This section describes how to remove a Ceph OSD node from a Ceph cluster.

To remove a Ceph OSD node:

1. If the host is explicitly defined in the model, perform the following steps. Otherwise, proceed to step 2.

    1. In your project repository, remove the following lines from the cluster/ceph/init.yml file or from the pillar based on your environment:

        ```
        _param:
           ceph_osd_node05_hostname: osd005
           ceph_osd_node05_address: 172.16.47.72
           ceph_osd_system_codename: xenial
        linux:
          network:
            host:
              osd005:
                address: ${_param:ceph_osd_node05_address}
                names:
                - ${_param:ceph_osd_node05_hostname}
                - ${_param:ceph_osd_node05_hostname}.${_param:cluster_domain}
        ```

    2. Remove the following lines from the cluster/infra/config/init.yml file or from the pillar based on your environment:

        ```
        parameters:
          reclass:
            storage:
              node:
                ceph_osd_node05:
                  name: ${_param:ceph_osd_node05_hostname}
                  domain: ${_param:cluster_domain}
                  classes:
                  - cluster.${_param:cluster_name}.ceph.osd
                  params:
                    salt_master_host: ${_param:reclass_config_master}
                    linux_system_codename:  ${_param:ceph_osd_system_codename}
                    single_address: ${_param:ceph_osd_node05_address}
                    ceph_crush_parent: rack02
        ```

2. Log in to the Jenkins web UI.
3. Open the Ceph - remove node pipeline.
4. Specify the following parameters:

Parameter	Description and values

SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	Add the Salt target name of the Ceph OSD node to remove. For example, osd005*.
HOST_TYPE Removed since 2019.2.13 update	Add osd as the type of Ceph node that is going to be removed.
OSD Added since 2019.2.13 update	Specify the list of Ceph OSDs to remove while keeping the rest and the entire node as part of the cluster. To remove all, leave empty or set to *.
GENERATE_CRUSHMAP	Select if the CRUSH map file should be updated. Enforce has to happen manually unless it is specifically set to be enforced in pillar.
ADMIN_HOST Removed since 2019.2.13 update	Add cmn01* as the Ceph cluster node with the admin keyring.
WAIT_FOR_HEALTHY	Mandatory since the 2019.2.13 maintenance update. Verify that this parameter is selected as it enables the Ceph health check within the pipeline.
CLEANDISK Added since 2019.2.10 update	Mandatory since the 2019.2.13 maintenance update. Select to clean the data or block partitions.
CLEAN_ORPHANS Added since 2019.2.13 update	Select to clean orphaned disks of Ceph OSDs that are no longer part of the cluster.
FAST_WIPE Added since 2019.2.13 update	Deselect if the entire disk needs zero filling.

5. Click Deploy.

   The Ceph - remove node pipeline workflow:

   1. Mark all Ceph OSDs running on the specified HOST as out. If you selected the WAIT_FOR_HEALTHY parameter, Jenkins pauses the execution of the pipeline until the data migrates to a different Ceph OSD.
   2. Stop all Ceph OSDs services running on the specified HOST.
   3. Remove all Ceph OSDs running on the specified HOST from the CRUSH map.
   4. Remove all Ceph OSD authentication keys running on the specified HOST.
   5. Remove all Ceph OSDs running on the specified HOST from Ceph cluster.
   6. Purge CEPH packages from the specified HOST.
   7. Stop the Salt Minion node on the specified HOST.
   8. Remove all Ceph OSDs running on the specified HOST from Ceph cluster.

9. Remove the Salt Minion node ID from salt-key on the Salt Master node.
10. Update the CRUSHMAP file on the I@ceph:setup:crush node if GENERATE_CRUSHMAP was selected. You must manually apply the update unless it is specified otherwise in the pillar.

6. If you selected GENERATE_CRUSHMAP, check the updated /etc/ceph/crushmap file on cmn01. If it is correct, apply the CRUSH map:

```
crushtool -c /etc/ceph/crushmap -o /etc/ceph/crushmap.compiled
ceph osd setcrushmap -i /etc/ceph/crushmap.compiled
```

Remove a RADOS Gateway node

This section describes how to remove a RADOS Gateway (rgw) node from a Ceph cluster.

To remove a RADOS Gateway node:

1. In your project repository, remove the following lines from the cluster/ceph/rgw.yml file or from the pillar based on your environment:

```
parameters:
  _param:
    cluster_node04_hostname: ${_param:ceph_rgw_node04_hostname}
    cluster_node04_address: ${_param:ceph_rgw_node04_address}
  ceph:
    common:
      keyring:
        rgw.rgw04:
          caps:
            mon: "allow rw"
            osd: "allow rwx"
  haproxy:
    proxy:
      listen:
        radosgw:
          servers:
            - name: ${_param:cluster_node04_hostname}
              host: ${_param:cluster_node04_address}
              port: ${_param:haproxy_radosgw_source_port}
              params: check
```

2. Remove the following lines from the cluster/infra/config/init.yml file or from the pillar based on your environment:

```
parameters:
  reclass:
    storage:
      node:
        ceph_rgw_node04:
          name: ${_param:ceph_rgw_node04_hostname}
          domain: ${_param:cluster_domain}
          classes:
          - cluster.${_param:cluster_name}.ceph.rgw
          params:
            salt_master_host: ${_param:reclass_config_master}
            linux_system_codename:  ${_param:ceph_rgw_system_codename}
            single_address: ${_param:ceph_rgw_node04_address}
            keepalived_vip_priority: 104
```

3. Log in to the Jenkins web UI.

4. Open the Ceph - remove node pipeline.
5. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	Add the Salt target name of the RADOS Gateway node to remove. For example, rgw04*.
HOST_TYPE Removed since 2019.2.13 update	Add rgw as the type of Ceph node that is going to be removed.
CLEANDISK Added since 2019.2.10 update	Mandatory since the 2019.2.13 maintenance update. Select to clean the data or block partitions.

6. Click Deploy.

   The Ceph - remove node pipeline workflow:

   1. Reconfigure HAProxy on the rest of RADOS Gateway nodes.
   2. Destroy the VM.
   3. Remove the Salt Minion node ID from salt-key on the Salt Master node.
7. Remove the following lines from the cluster/infra/kvm.yml file or from the pillar based on your environment:

```
parameters:
  salt:
    control:
      cluster:
        internal:
          node:
            rgw04:
              name: ${_param:ceph_rgw_node04_hostname}
              provider: ${_param:infra_kvm_node03_hostname}.${_param:cluster_domain}
              image: ${_param:salt_control_xenial_image}
              size: ceph.rgw
```

8. Remove the following lines from the cluster/ceph/init.yml file or from the pillar based on your environment:

```
_param:
  ceph_rgw_node04_hostname: rgw04
  ceph_rgw_node04_address: 172.16.47.162
linux:
```

```yaml
network:
  host:
    rgw04:
      address: ${_param:ceph_rgw_node04_address}
      names:
      - ${_param:ceph_rgw_node04_hostname}
      - ${_param:ceph_rgw_node04_hostname}.${_param:cluster_domain}
```

Remove a Ceph OSD daemon

Note

This feature is available starting from the MCP 2019.2.13 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

Note

The pipeline used in this section is a wrapper for Ceph - remove node, which simplifies common operations.

This section describes how to remove a Ceph OSD daemon from the cluster without removing the entire Ceph OSD node.

To remove a Ceph OSD daemon:

1. Log in to the Jenkins web UI.
2. Open the Ceph - remove osd pipeline.
3. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	The Salt target name of the host from which the Ceph OSD daemons are going to be removed. For example, osd005*.
OSD	A comma-separated list of Ceph OSD daemons to remove while keeping the rest and the entire node as part of the cluster. Do not leave this parameter empty.
WAIT_FOR_HEALTHY	Verify that this parameter is selected as it enables the Ceph health check within the pipeline.
CLEAN_ORPHANS	Select to clean orphaned disks of Ceph OSDs that are no longer part of the cluster.
FAST_WIPE	Deselect if the entire disk needs zero filling.

4. Click Deploy.

©2025, Mirantis Inc.

Replace a failed Ceph OSD

This section instructs you on how to replace a failed physical node with a Ceph OSD or multiple OSD nodes running on it using the Ceph - replace failed OSD Jenkins pipeline.

To replace a failed physical node with a Ceph OSD or multiple OSD nodes:

1. Log in to the Jenkins web UI.
2. Open the Ceph - replace failed OSD pipeline.
3. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
HOST	Add the Salt target name of the Ceph OSD node. For example, osd005*.
OSD	Add a comma-separated list of Ceph OSDs on the specified HOST node. For example 1,2.
DEVICE [11]	Add a comma-separated list of failed devices to replace at HOST. For example, /dev/sdb,/dev/sdc.
DATA_PARTITION: [11]	(Optional) Add a comma-separated list of mounted partitions of the failed device. These partitions will be unmounted. We recommend that multiple OSD nodes per device are used. For example, /dev/sdb1,/dev/sdb3.
JOURNAL_BLOCKDB_BLOCKWAL_PARTITION: [11]	Add a comma-separated list of partitions that store journal, block_db, or block_wal of the failed devices on the specified HOST. For example, /dev/sdh2,/dev/sdh3.
ADMIN_HOST	Add cmn01* as the Ceph cluster node with the admin keyring.
CLUSTER_FLAGS	Add a comma-separated list of flags to apply before and after the pipeline.
WAIT_FOR_HEALTHY	Select to perform the Ceph health check within the pipeline.
DMCRYPT [11]	Select if you are replacing an encrypted OSD. In such case, also specify noout,norebalance as CLUSTER_FLAGS.

4. Click Deploy.

The Ceph - replace failed OSD pipeline workflow:

1. Mark the Ceph OSD as out.
2. Wait until the Ceph cluster is in a healthy state if WAIT_FOR_HEALTHY was selected. In this case. Jenkins pauses the execution of the pipeline until the data migrates to a different Ceph OSD.

©2025, Mirantis Inc.

3. Stop the Ceph OSD service.
4. Remove the Ceph OSD from the CRUSH map.
5. Remove the Ceph OSD authentication key.
6. Remove the Ceph OSD from the Ceph cluster.
7. Unmount data partition(s) of the failed disk.
8. Delete the partition table of the failed disk.
9. Remove the partition from the block_db, block_wal, or journal.
10. Perform one of the following depending on the MCP release version:
    - For deployments prior to the MCP 2019.2.3 update, redeploy the failed Ceph OSD.
    - For deployments starting from the MCP 2019.2.3 update:
        1. Wait for the hardware replacement and confirmation to proceed.
        2. Redeploy the failed Ceph OSD on the replaced hardware.

> Note
>
> If any of the steps 1 - 9 has already been performed manually, Jenkins proceeds to the next step.

11(1, 2, 3, 4) The parameter has been removed starting from the MCP 2019.2.3 update.

Restrict the RADOS Gateway capabilities

> Note
>
> This feature is available starting from the MCP 2019.2.10 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

To avoid a potential security vulnerability, Mirantis recommends that you restrict the RADOS Gateway capabilities of your existing MCP deployment to a bare minimum.

To restrict the RADOS Gateway capabilities of an existing MCP deployment:

1. Open your project Git repository with the Reclass model on the cluster level.
2. In cluster/ceph/rgw.yml, modify the RADOS Gateway capabilities as follows:

```
ceph:
  common:
    keyring:
      rgw.rgw01:
        caps:
          mon: "allow rw"
          osd: "allow rwx"
      rgw.rgw02:
        caps:
          mon: "allow rw"
          osd: "allow rwx"
      rgw.rgw03:
        caps:
          mon: "allow rw"
          osd: "allow rwx"
```

3. Log in to the Salt Master node.
4. Apply the changes:

```
salt -I ceph:radosgw state.apply ceph.common,ceph.setup.keyring
```

Enable the Ceph Prometheus plugin

If you have deployed StackLight LMA, you can enhance Ceph monitoring by enabling the Ceph Prometheus plugin that is based on the native Prometheus exporter introduced in Ceph Luminous. In this case, the Ceph Prometheus plugin, instead of Telegraf, collects Ceph metrics providing a wider set of graphs in the Grafana web UI, such as an overview of the Ceph cluster, hosts, OSDs, pools, RADOS gateway nodes, as well as detailed graphs on the Ceph OSD and RADOS Gateway nodes. You can enable the Ceph Prometheus plugin manually on an existing MCP cluster as described below or during the upgrade of StackLight LMA as described in Upgrade StackLight LMA using the Jenkins job.

To enable the Ceph Prometheus plugin manually:

1. Update the Ceph formula package.
2. Open your project Git repository with Reclass model on the cluster level.
3. In classes/cluster/cluster_name/ceph/mon.yml, remove the service.ceph.monitoring.cluster_stats class.
4. In classes/cluster/cluster_name/ceph/osd.yml, remove the service.ceph.monitoring.node_stats class.
5. Log in to the Salt Master node.
6. Refresh grains to set the new alerts and graphs:

   ```
   salt '*' state.sls salt.minion.grains
   ```

7. Enable the Prometheus plugin:

   ```
   salt -C I@ceph:mon state.sls ceph.mgr
   ```

8. Update the targets and alerts in Prometheus:

   ```
   salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus
   ```

9. Update the new Grafana dashboards:

   ```
   salt -C 'I@grafana:client' state.sls grafana
   ```

10. (Optional) Enable the StackLight LMA prediction alerts for Ceph.

    Note

    This feature is available as technical preview. Use such configuration for testing and evaluation purposes only.

> Warning
>
> This feature is available starting from the MCP 2019.2.3 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

1. Open your project Git repository with Reclass model on the cluster level.
2. In classes/cluster/cluster_name/ceph/common.yml, set enable_prediction to True:

   ```
   parameters:
     ceph:
       common:
         enable_prediction: True
   ```

3. Log in to the Salt Master node.
4. Refresh grains to set the new alerts and graphs:

   ```
   salt '*' state.sls salt.minion.grains
   ```

5. Verify and update the alerts thresholds based on the cluster hardware.

   > Note
   >
   > For details about tuning the thresholds, contact Mirantis support.

6. Update the targets and alerts in Prometheus:

   ```
   salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus
   ```

11. Customize Ceph prediction alerts as described in Ceph.

Enable Ceph compression

Note

This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

RADOS Gateway supports server-side compression of uploaded objects using the Ceph compression plugins. You can manually enable Ceph compression to rationalize the capacity usage on the MCP cluster.

To enable Ceph compression:

1. Log in to any rgw node.
2. Run the radosgw-admin zone placement modify command with the --compression=<type> option specifying the compression plugin type and other options as required. The available compression plugins to use when writing a new object data are zlib, snappy, or zstd. For example:

   ```
   radosgw-admin zone placement modify \
     --rgw-zone default \
     --placement-id default-placement \
     --storage-class STANDARD \
     --compression zlib
   ```

   Note

   If you have not previously performed any Multi-site configuration, you can use the default values for the options except compression. To disable compression, set the compression type to an empty string or none.

Seealso

Ceph compression

Enable the ceph-volume tool

Note

This feature is available starting from the MCP 2019.2.7 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

Warning

- Prior to the 2019.2.10 maintenance update, ceph-volume is available as technical preview only.
- Starting from the 2019.2.10 maintenance update, the ceph-volume tool is fully supported and must be enabled prior to upgrading from Ceph Luminous to Nautilus. The ceph-disk tool is deprecated.

This section describes how to enable the ceph-volume command-line tool that enables you to deploy and inspect Ceph OSDs using the Logical Volume Management (LVM) functionality for provisioning block devices. The main difference between ceph-disk and ceph-volume is that ceph-volume does not automatically partition disks used for block.db. However, partitioning is performed within the procedure below.

To enable the ceph-volume tool:

1. Open your Git project repository with the Reclass model on the cluster level.
2. Open ceph/osd.yml for editing.
3. Set the lvm_enabled parameter to True:

```
parameters:
  ceph:
    osd:
      lvm_enabled: True
```

4. For each Ceph OSD, add the definition of a bare block.db partition and define its number in db_partition. For example:

```
parameters:
  ceph:
    osd:
      backend:
        bluestore:
          disks:
            - dev: /dev/vdc
```

```
          block_db: /dev/vdd
          db_partition: 1
linux:
  storage:
    disk:
      /dev/vdd:
        type: gpt
        partitions:
          - size: 10000
```

> Note
>
> Due to a custom disk layout, these definitions may already exist and be configured differently. In this case, verify that the db_partition number is defined.

5. Apply the changes:

   ```
   salt -C 'I@ceph:osd' saltutil.refresh_pillar
   ```

   Once done, all new Ceph OSDs will be deployed using ceph-volume instead of ceph-disk. The existing Ceph OSDs will cause errors during common operations and must be redeployed or defined as legacy_disks as described below.

6. Select from the following options:

   - Redeploy Ceph OSD nodes or daemons:

     > Warning
     >
     > Before redeploying a node, verify that all pools have at least three copies. Redeploy only one node at a time. Redeploying multiple nodes may cause irreversible data loss.

     - Prior to MCP 2019.2.13, redeploy the Ceph OSD nodes:
       1. Remove the OSD nodes as described in Remove a Ceph OSD node.
       2. Add new OSD nodes as described in Add a Ceph OSD node.
     - Starting from MCP 2019.2.13, redeploy the Ceph OSD daemons:
       1. Remove the Ceph OSD daemons as described in Remove a Ceph OSD daemon.
       2. Add new Ceph OSD daemons as described in Add a Ceph OSD daemon.

- Define existing Ceph OSDs as legacy_disks by specifying the legacy_disks pillar for each Ceph OSD in ceph/osd.yml. For example:

```
parameters:
  ceph:
    osd:
      legacy_disks:
        0:
          class: hdd
          weight: 0.048691
          dev: /dev/vdc
```

> Note
>
> Use the legacy_disks option only as a temporary solution for common management of the cluster during the transition period.

Shut down a Ceph cluster for maintenance

This section describes how to properly shut down an entire Ceph cluster for maintenance and bring it up afterward.

To shut down a Ceph cluster for maintenance:

1. Log in to the Salt Master node.
2. Stop the OpenStack workloads.
3. Stop the services that are using the Ceph cluster. For example:
    * Manila workloads (if you have shares on top of Ceph mount points)
    * heat-engine (if it has the autoscaling option enabled)
    * glance-api (if it uses Ceph to store images)
    * cinder-scheduler (if it uses Ceph to store images)
4. Identify the first Ceph Monitor for operations:

    ```
    CEPH_MON=$(salt -C 'I@ceph:mon' --out=txt test.ping | sort | head -1 | \
        cut -d: -f1)
    ```

5. Verify that the Ceph cluster is in healthy state:

    ```
    salt "${CEPH_MON}" cmd.run 'ceph -s'
    ```

    Example of system response:

    ```
    cmn01.domain.com:
        cluster e0b75d1b-544c-4e5d-98ac-cfbaf29387ca
         health HEALTH_OK
         monmap e3: 3 mons at {cmn01=192.168.16.14:6789/0,cmn02=192.168.16.15:6789/0,cmn03=192.168.16.16:6789/0}
                election epoch 42, quorum 0,1,2 cmn01,cmn02,cmn03
         osdmap e102: 6 osds: 6 up, 6 in
                flags sortbitwise,require_jewel_osds
          pgmap v41138: 384 pgs, 6 pools, 45056 kB data, 19 objects
                798 MB used, 60575 MB / 61373 MB avail
                    384 active+clean
    ```

6. Set the following flags to disable rebalancing and restructuring and to pause the Ceph cluster:

    ```
    salt "${CEPH_MON}" cmd.run 'ceph osd set noout'
    salt "${CEPH_MON}" cmd.run 'ceph osd set nobackfill'
    salt "${CEPH_MON}" cmd.run 'ceph osd set norecover'
    salt "${CEPH_MON}" cmd.run 'ceph osd set norebalance'
    salt "${CEPH_MON}" cmd.run 'ceph osd set nodown'
    salt "${CEPH_MON}" cmd.run 'ceph osd set pause'
    ```

7. Verify that the flags are set:

```
salt "${CEPH_MON}" cmd.run 'ceph -s'
```

Example of system response:

```
cmn01.domain.com:
       cluster e0b75d1b-544c-4e5d-98ac-cfbaf29387ca
        health **HEALTH_WARN**
              **pauserd**,**pausewr**,**nodown**,**noout**,**nobackfill**,**norebalance**,**norecover** flag(s) set
        monmap e3: 3 mons at {cmn01=192.168.16.14:6789/0,cmn02=192.168.16.15:6789/0,cmn03=192.168.16.16:6789/0}
              election epoch 42, quorum 0,1,2 cmn01,cmn02,cmn03
        osdmap e108: 6 osds: 6 up, 6 in
              flags **pauserd**,**pausewr**,**nodown**,**noout**,**nobackfill**,**norebalance**,**norecover**,sortbitwise,require_jewel_osds
         pgmap v41152: 384 pgs, 6 pools, 45056 kB data, 19 objects
              799 MB used, 60574 MB / 61373 MB avail
                   384 active+clean
```

8. Shut down the Ceph cluster.

> Warning
>
> Shut down the nodes one by one in the following order:
>
> 1. Service nodes (for example, RADOS Gateway nodes)
> 2. Ceph OSD nodes
> 3. Ceph Monitor nodes

Once done, perform the maintenance as required.

To start a Ceph cluster after maintenance:

1. Log in to the Salt Master node.
2. Start the Ceph cluster nodes.

> Warning
>
> Start the Ceph nodes one by one in the following order:
>
> 1. Ceph Monitor nodes
> 2. Ceph OSD nodes
> 3. Service nodes (for example, RADOS Gateway nodes)

3. Verify that the Salt minions are up:

```
salt -C "I@ceph:common" test.ping
```

4. Verify that the date is the same for all Ceph clients:

```
salt -C "I@ceph:common" cmd.run date
```

5. Identify the first Ceph Monitor for operations:

```
CEPH_MON=$(salt -C 'I@ceph:mon' --out=txt test.ping | sort | head -1 | \
   cut -d: -f1)
```

6. Unset the following flags to resume the Ceph cluster:

```
salt "${CEPH_MON}" cmd.run 'ceph osd unset pause'
salt "${CEPH_MON}" cmd.run 'ceph osd unset nodown'
salt "${CEPH_MON}" cmd.run 'ceph osd unset norebalance'
salt "${CEPH_MON}" cmd.run 'ceph osd unset norecover'
salt "${CEPH_MON}" cmd.run 'ceph osd unset nobackfill'
salt "${CEPH_MON}" cmd.run 'ceph osd unset noout'
```

7. Verify that the Ceph cluster is in healthy state:

```
salt "${CEPH_MON}" cmd.run 'ceph -s'
```

Example of system response:

```
cmn01.domain.com:
        cluster e0b75d1b-544c-4e5d-98ac-cfbaf29387ca
         health HEALTH_OK
         monmap e3: 3 mons at {cmn01=192.168.16.14:6789/0,cmn02=192.168.16.15:6789/0,cmn03=192.168.16.16:6789/0}
                election epoch 42, quorum 0,1,2 cmn01,cmn02,cmn03
         osdmap e102: 6 osds: 6 up, 6 in
                flags sortbitwise,require_jewel_osds
          pgmap v41138: 384 pgs, 6 pools, 45056 kB data, 19 objects
                798 MB used, 60575 MB / 61373 MB avail
                     384 active+clean
```

Seealso

How to do a Ceph cluster maintenance/shutdown

Back up and restore Ceph

This section describes how to back up and restore Ceph OSD nodes metadata and Ceph Monitor nodes.

> Note
>
> This documentation does not provide instructions on how to back up the data stored in Ceph.

Create a backup schedule for Ceph nodes

This section describes how to manually create a backup schedule for Ceph OSD nodes metadata and for Ceph Monitor nodes.

By default, the backing up functionality enables automatically for the new MCP OpenStack with Ceph deployments in the cluster models generated using Model Designer. Use this procedure in case of manual deployment only or if you want to change the default backup configuration.

> Note
>
> The procedure below does not cover the backup of the Ceph OSD node data.

To create a backup schedule for Ceph nodes:

1. Log in to the Salt Master node.
2. Decide on which node you want to store the backups.
3. Get <STORAGE_ADDRESS> of the node from point 2.

```
cfg01:~\# salt NODE_NAME grains.get fqdn_ip4
```

4. Configure the ceph backup server role by adding the cluster.deployment_name.infra.backup.server class to the definition of the target storage node from step 2:

```
classes:
- cluster.deployment_name.infra.backup.server
 parameters:
   _param:
      ceph_backup_public_key: <generate_your_keypair>
```

   By default, adding this include statement results in Ceph keeping five complete backups. To change the default setting, add the following pillar to the cluster/infra/backup/server.yml file:

```
parameters:
  ceph:
    backup:
      server:
        enabled: true
        hours_before_full: 24
        full_backups_to_keep: 5
```

5. To back up the Ceph Monitor nodes, configure the ceph backup client role by adding the following lines to the cluster/ceph/mon.yml file:

Note

Change <STORAGE_ADDRESS> to the address of the target storage node from step 2

```
classes:
- system.ceph.backup.client.single
parameters:
  _param:
    ceph_remote_backup_server: <STORAGE_ADDRESS>
    root_private_key: |
      <generate_your_keypair>
```

6. To back up the Ceph OSD nodes metadata, configure the ceph backup client role by adding the following lines to the cluster/ceph/osd.yml file:

Note

Change <STORAGE_ADDRESS> to the address of the target storage node from step 2

```
classes:
- system.ceph.backup.client.single
parameters:
  _param:
    ceph_remote_backup_server: <STORAGE_ADDRESS>
    root_private_key: |
      <generate_your_keypair>
```

By default, adding the above include statement results in Ceph keeping three complete backups on the client node. To change the default setting, add the following pillar to the cluster/ceph/mon.yml or cluster/ceph/osd.yml files:

Note

Change <STORAGE_ADDRESS> to the address of the target storage node from step 2

```
parameters:
  ceph:
```

```
backup:
  client:
    enabled: true
    full_backups_to_keep: 3
    hours_before_full: 24
    target:
      host: <STORAGE_ADDRESS>
```

7. Refresh Salt pillars:

```
salt -C '*' saltutil.refresh_pillar
```

8. Apply the salt.minion state:

```
salt -C 'I@ceph:backup:client or I@ceph:backup:server' state.sls salt.minion
```

9. Refresh grains for the ceph client node:

```
salt -C 'I@ceph:backup:client' saltutil.sync_grains
```

10. Update the mine for the ceph client node:

```
salt -C 'I@ceph:backup:client' mine.flush
salt -C 'I@ceph:backup:client' mine.update
```

11. Apply the following state on the ceph client node:

```
salt -C 'I@ceph:backup:client' state.sls openssh.client,ceph.backup
```

12. Apply the linux.system.cron state on the ceph server node:

```
salt -C 'I@ceph:backup:server' state.sls linux.system.cron
```

13. Apply the ceph.backup state on the ceph server node:

```
salt -C 'I@ceph:backup:server' state.sls ceph.backup
```

Seealso

- Restore a Ceph Monitor node
- Restore the metadata of a Ceph OSD node

Create an instant backup of a Ceph OSD node metadata or a Ceph Monitor node

After you create a backup schedule as described in Create a backup schedule for Ceph nodes, you may also need to create an instant backup of a Ceph OSD node metadata or a Ceph Monitor node.

> **Note**
>
> The procedure below does not cover the backup of the Ceph OSD node data.

To create an instant backup of a Ceph node:

1. Verify that you have completed the steps described in Create a backup schedule for Ceph nodes.
2. Log in to a Ceph node. For example, to cmn01.
3. Run the following script:

```
/usr/local/bin/ceph-backup-runner-call.sh
```

4. Verify that a complete backup was created locally:

```
ls /var/backups/ceph/full
```

5. Verify that the complete backup was rsynced to the ceph backup server node from the Salt Master node:

```
salt -C 'I@ceph:backup:server' cmd.run 'ls /srv/volumes/backup/ceph/full'
```

> **Seealso**
>
> - Restore a Ceph Monitor node
> - Restore the metadata of a Ceph OSD node

Restore a Ceph Monitor node

You may need to restore a Ceph Monitor node after a failure. For example, if the data in the Ceph-related directories disappeared.

To restore a Ceph Monitor node:

1. Verify that the Ceph Monitor instance is up and running and connected to the Salt Master node.
2. Log in to the Ceph Monitor node.
3. Synchronize Salt modules and refresh Salt pillars:

   ```
   salt-call saltutil.sync_all
   salt-call saltutil.refresh_pillar
   ```

4. Run the following Salt states:

   ```
   salt-call state.sls linux,openssh,salt,ntp,rsyslog
   ```

5. Manually install Ceph packages:

   ```
   apt install ceph-mon -y
   ```

6. Remove the following files from Ceph:

   ```
   rm -rf /etc/ceph/* /var/lib/ceph/*
   ```

7. From the Ceph backup, copy the files from /etc/ceph/ and /var/lib/ceph to their original directories:

   ```
   cp -r /<etc_ceph_backup_path>/* /etc/ceph/
   cp -r /<var_lib_ceph_backup_path>/* /var/lib/ceph/
   ```

8. Change the files ownership:

   ```
   chown -R ceph:ceph /var/lib/ceph/*
   ```

9. Run the following Salt state:

   ```
   salt-call state.sls ceph
   ```

   If the output contains an error, rerun the state.

Restore the metadata of a Ceph OSD node

You may need to restore the metadata of a Ceph OSD node after a failure. For example, if the primary disk fails or the data in the Ceph-related directories, such as /var/lib/ceph/, on the OSD node disappeared.

To restore the metadata of a Ceph OSD node:

1. Verify that the Ceph OSD node is up and running and connected to the Salt Master node.
2. Log in to the Ceph OSD node.
3. Synchronize Salt modules and refresh Salt pillars:

   ```
   salt-call saltutil.sync_all
   salt-call saltutil.refresh_pillar
   ```

4. Run the following Salt states:

   ```
   salt-call state.sls linux,openssh,salt,ntp,rsyslog
   ```

5. Manually install Ceph packages:

   ```
   apt install ceph-osd -y
   ```

6. Stop all ceph-osd services:

   ```
   systemctl stop ceph-osd@<num>
   ```

7. Remove the following files from Ceph:

   ```
   rm -rf /etc/ceph/* /var/lib/ceph/*
   ```

8. From the Ceph backup, copy the files from /etc/ceph/ and /var/lib/ceph to their original directories:

   ```
   cp -r /<path>/* /etc/ceph/
   cp -r /<path>/* /var/lib/ceph/
   ```

9. Change the files ownership:

   ```
   chown -R ceph:ceph /var/lib/ceph/*
   ```

10. Restart the services for all Ceph OSDs:

   ```
   systemctl restart ceph-osd@<osd_num>
   ```

Migrate the Ceph back end

Ceph uses FileStore or BlueStore as a storage back end. You can migrate the Ceph storage back end from FileStore to BlueStore and vice versa using the Ceph - backend migration pipeline.

> Note
>
> Starting from the 2019.2.10 maintenance update, this procedure is deprecated and all Ceph OSDs should use LVM with BlueStore. Back-end migration is described in Enable the ceph-volume tool.
>
> For earlier versions, if you are going to upgrade Ceph to Nautilus, also skip this procedure to avoid a double migration of the back end. In this case, first apply the 2019.2.10 maintenance update and then enable ceph-volume as well.

To migrate the Ceph back end:

1. In your project repository, open the cluster/ceph/osd.yml file for editing:

    1. Change the back end type and block_db or journal for every OSD disk device.
    2. Specify the size of the journal or block_db device if it resides on another device than the storage device. The device storage will be divided equally by the number of OSDs using it.

    Example:

    ```
    parameters:
      ceph:
        osd:
          bluestore_block_db_size: 10073741824
    #      journal_size: 10000
          backend:
    #        filestore:
            bluestore:
              disks:
              - dev: /dev/sdh
                block_db: /dev/sdj
    #            journal: /dev/sdj
    ```

    Where the commented lines are the example lines that must be replaced and removed if migrating from FileStore to BlueStore.

2. Log in to the Jenkins web UI.
3. Open the Ceph - backend migration pipeline.
4. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
ADMIN_HOST	Add cmn01* as the Ceph cluster node with the admin keyring.
TARGET	Add the Salt target name of the Ceph OSD node(s). For example, osd005* to migrate on one OSD HOST or osd* to migrate on all OSD hosts.
OSD	Add * to target all OSD disks on all TARGET OSD hosts or comma-separated list of Ceph OSDs if targeting just one OSD host by TARGET For example 1,2.
WAIT_FOR_HEALTHY	Verify that this parameter is selected as it enables the Ceph health check within the pipeline.
PER_OSD_CONTROL	Select to verify the Ceph status after migration of each OSD disk.
PER_OSD_HOST_CONTROL	Select to verify the Ceph status after the whole OSD host migration.
CLUSTER_FLAGS	Add a comma-separated list of flags to apply for the migration procedure. Tested with blank.
ORIGIN_BACKEND	Specify the Ceph back end before migration.

Note

The PER_OSD_CONTROL and PER_OSD_HOST_CONTROL options provide granular control during the migration to verify each OSD disk after its migration. You can decide to continue or abort.

5. Click Deploy.

The Ceph - upgrade pipeline workflow:

1. Set back-end migration flags.
2. Perform the following for each targeted OSD disk:
    1. Mark the Ceph OSD as out.
    2. Stop the Ceph OSD service.
    3. Remove the Ceph OSD authentication key.
    4. Remove the Ceph OSD from the Ceph cluster

    5. Remove block_db, block_wal, or journal of the OSD.

3. Run the ceph.osd state to deploy the OSD with a desired back end.
4. Unset the back-end migration flags.

> Note
>
> During the pipeline execution, a check is performed to verify whether the back end type for an OSD disk differs from the one specified in ORIGIN_BACKEND. If the back end differs, Jenkins does not apply any changes to that OSD disk.

Migrate the management of a Ceph cluster

You can migrate the management of an existing Ceph cluster deployed by Decapod to a cluster managed by the Ceph Salt formula.

To migrate the management of a Ceph cluster:

1. Log in to the Decapod web UI.
2. Navigate to the CONFIGURATIONS tab.
3. Select the required configuration and click VIEW.
4. Generate a new cluster model with Ceph as described in MCP Deployment Guide: Create a deployment metadata model using the Model Designer. Verify that you fill in the correct values from the Decapod configuration file displayed in the VIEW tab of the Decapod web UI.
5. In the <cluster_name>/ceph/setup.yml file, specify the right pools and parameters for the existing pools.

   > Note
   >
   > Verify that the keyring names and their caps match the ones that already exist in the Ceph cluster deployed by Decapod.

6. In the <cluster_name>/infra/config.yml file, add the following pillar and modify the parameters according to your environment:

   ```
   ceph:
     decapod:
       ip: 192.168.1.10
       user: user
       pass: psswd
       deploy_config_name: ceph
   ```

7. On the node defined in the previous step, apply the following state:

   ```
   salt-call state.sls ceph.migration
   ```

   > Note
   >
   > The output of this state must contain defined configurations, Ceph OSD disks, Ceph File System ID (FSID), and so on.

©2025, Mirantis Inc.

8. Using the output of the previous command, add the following pillars to your cluster model:
    1. Add the ceph:common pillar to <cluster_name>/ceph/common.yml.
    2. Add the ceph:osd pillar to <cluster_name>/ceph/osd.yml.
9. Examine the newly generated cluster model for any occurrence of the ceph keyword and verify that it exists in your current cluster model.
10. Examine each Ceph cluster file to verify that the parameters match the configuration specified in Decapod.
11. Copy the Ceph cluster directory to the existing cluster model.
12. Verify that the ceph subdirectory is included in your cluster model in <cluster_name>/infra/init.yml or <cluster_name>/init.yml for older cluster model versions:

```
classes:
- cluster.<cluster_name>.ceph
```

13. Add the Reclass storage nodes to <cluster_name>/infra/config.yml and change the count variable to the number of OSDs you have. For example:

```
classes:
- system.reclass.storage.system.ceph_mon_cluster
- system.reclass.storage.system.ceph_rgw_cluster # Add this line only if
# RadosGW services run on separate nodes than the Ceph Monitor services.
parameters:
  reclass:
    storage:
      node:
        ceph_osd_rack01:
          name: ${_param:ceph_osd_rack01_hostname}<<count>>
          domain: ${_param:cluster_domain}
          classes:
            - cluster.${_param:cluster_name}.ceph.osd
          repeat:
            count: 3
            start: 1
            digits: 3
            params:
              single_address:
                value: ${_param:ceph_osd_rack01_single_subnet}.<<count>>
                start: 201
              backend_address:
                value: ${_param:ceph_osd_rack01_backend_subnet}.<<count>>
                start: 201
```

14. If the Ceph RADOS Gateway service is running on the same nodes as the Ceph monitor services:

1. Add the following snippet to <cluster_name>/infra/config.yml:

```
reclass:
  storage:
    node:
      ceph_mon_node01:
        classes:
        - cluster.${_param:cluster_name}.ceph.rgw
      ceph_mon_node02:
        classes:
        - cluster.${_param:cluster_name}.ceph.rgw
      ceph_mon_node03:
        classes:
        - cluster.${_param:cluster_name}.ceph.rgw
```

2. Verify that the parameters in <cluster_name>/ceph/rgw.yml are defined correctly according to the existing Ceph cluster.

15. From the Salt Master node, generate the Ceph nodes:

```
salt-call state.sls reclass
```

16. Run the commands below.

Warning

If the outputs of the commands below contain any changes that can potentially break the cluster, change the cluster model as needed and optionally run the salt-call pillar.data ceph command to verify that the Salt pillar contains the correct value. Proceed to the next step only once you are sure that your model is correct.

- From the Ceph monitor nodes:

```
salt-call state.sls ceph test=True
```

- From the Ceph OSD nodes:

```
salt-call state.sls ceph test=True
```

- From the Ceph RADOS Gateway nodes:

```
salt-call state.sls ceph test=True
```

- From the Salt Master node:

```
salt -C 'I@ceph:common' state.sls ceph test=True
```

17. Once you have verified that no changes by the Salt Formula can break the running Ceph cluster, run the following commands.

    - From the Salt Master node:

    ```
    salt -C 'I@ceph:common:keyring:admin' state.sls ceph.mon
    salt -C 'I@ceph:mon' saltutil.sync_grains
    salt -C 'I@ceph:mon' mine.update
    salt -C 'I@ceph:mon' state.sls ceph.mon
    ```

    - From one of the OSD nodes:

    ```
    salt-call state.sls ceph.osd
    ```

    > Note
    >
    > Before you proceed, verify that the OSDs on this node are working fine.

    - From the Salt Master node:

    ```
    salt -C 'I@ceph:osd' state.sls ceph.osd
    ```

    - From the Salt Master node:

    ```
    salt -C 'I@ceph:radosgw' state.sls ceph.radosgw
    ```

Enable RBD monitoring

> Warning
>
> This feature is available as technical preview starting from the MCP 2019.2.10 maintenance update and requires Ceph Nautilus. Use such configuration for testing and evaluation purposes only. Before using the feature, follow the steps described in Apply maintenance updates.

If required, you can enable RADOS Block Device (RBD) images monitoring introduced with Ceph Nautilus. Once done, you can view RBD metrics using the Ceph RBD Overview Grafana dashboard. For details, see Ceph dashboards.

To enable RBD monitoring:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In classes/cluster/<cluster_name>/ceph/setup.yml add the rbd_stats flag for pools serving RBD images to enable serving RBD metrics:

```
parameters:
  ceph:
    setup:
      pool:
        <pool_name>:
          pg_num: 8
          pgp_num: 8
          type: replicated
          application: rbd
          rbd_stats: True
```

3. In classes/cluster/<cluster_name>/ceph/common.yml, set the rbd_monitoring_enabled parameter to True to enable the Ceph RBD Overview Grafana dashboard:

```
ceph:
  common:
    public_network: 10.13.0.0/16
    cluster_network: 10.12.0.0/16
    rbd_monitoring_enabled: True
```

4. Log in to the Salt Master node.
5. Apply the changes:

```
salt "*" saltutil.refresh_pillar
salt "*" state.apply salt.minion.grains
salt "*" saltutil.refresh_grains
```

```
salt -C "I@ceph:mgr" state.apply 'ceph.mgr'
salt -C 'I@grafana:client' state.apply 'grafana.client'
```

Enable granular distribution of Ceph keys

> Note
>
> This feature is available starting from the MCP 2019.2.14 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes how to enable granular distribution of Ceph keys on an existing deployment to avoid keeping the Ceph keys for the services that do not belong to a particular node.

To enable granular distribution of Ceph keys:

1. Open your Git project repository with the Reclass model on the cluster level.
2. Create a new ceph/keyrings folder.
3. Open the ceph/common.yml file for editing.
4. Move the configuration for each component from the parameters:ceph:common:keyrings section to a corresponding file in the newly created folder. For example, the following configuration must be split to four different files.

   ```
   ceph:
    common:
     keyring:
      glance:
       name: ${_param:glance_storage_user}
       caps:
        mon: 'allow r, allow command "osd blacklist"'
        osd: "profile rbd pool=images"
      cinder:
       name: ${_param:cinder_storage_user}
       caps:
        mon: 'allow r, allow command "osd blacklist"'
        osd: "profile rbd pool=volumes, profile rbd-read-only pool=images, profile rbd pool=${_param:cinder_ceph_backup_pool}"
      nova:
       name: ${_param:nova_storage_user}
       caps:
        mon: 'allow r, allow command "osd blacklist"'
        osd: "profile rbd pool=vms, profile rbd-read-only pool=images"
      gnocchi:
       name: ${_param:gnocchi_storage_user}
       caps:
        mon: 'allow r, allow command "osd blacklist"'
        osd: "profile rbd pool=${_param:gnocchi_storage_pool}"
   ```

   In this case, each file must have its own component keyring. For example:

   - In ceph/keyrings/nova.yml, add:

     ```
     parameters:
       ceph:
         common:
     ```

```yaml
      keyring:
        nova:
          name: ${_param:nova_storage_user}
          caps:
            mon: 'allow r, allow command "osd blacklist"'
            osd: "profile rbd pool=vms, profile rbd-read-only pool=images"
```

- In ceph/keyrings/cinder.yml, add:

```yaml
parameters:
  ceph:
    common:
      keyring:
        cinder:
          name: ${_param:cinder_storage_user}
          caps:
            mon: 'allow r, allow command "osd blacklist"'
            osd: "profile rbd pool=volumes, profile rbd-read-only pool=images, profile rbd pool=${_param:cinder_ceph_backup_pool}"
```

- In ceph/keyrings/glance.yml, add:

```yaml
parameters:
  ceph:
    common:
      keyring:
        glance:
          name: ${_param:glance_storage_user}
          caps:
            mon: 'allow r, allow command "osd blacklist"'
            osd: "profile rbd pool=images"
```

- In ceph/keyrings/gnocchi.yml, add:

```yaml
parameters:
  ceph:
    common:
      keyring:
        gnocchi:
          name: ${_param:gnocchi_storage_user}
          caps:
            mon: 'allow r, allow command "osd blacklist"'
            osd: "profile rbd pool=${_param:gnocchi_storage_pool}"
```

5. In the same ceph/keyrings folder, create an init.yml file and add the newly created keyrings:

```yaml
classes:
- cluster.<cluster_name>.ceph.keyrings.glance
- cluster.<cluster_name>.ceph.keyrings.cinder
```

```
- cluster.<cluster_name>.ceph.keyrings.nova
- cluster.<cluster_name>.ceph.keyrings.gnocchi
```

> Note
>
> If Telemetry is disabled, Gnocchi may not be present in your deployment.

6. In openstack/compute/init.yml, add the Cinder and Nova keyrings after class cluster.<cluster_name>.ceph.common:

   ```
   - cluster.<cluster_name>.ceph.keyrings.cinder
   - cluster.<cluster_name>.ceph.keyrings.nova
   ```

7. In openstack/control.yml, add the following line after cluster.<cluster_name>.ceph.common:

   ```
   - cluster.<cluster_name>.ceph.keyrings
   ```

8. In openstack/telemetry.yml add the Gnocchi keyring after class cluster.<cluster_name>.ceph.common:

   ```
   - cluster.<cluster_name>.ceph.keyrings.gnocchi
   ```

9. Log in to the Salt Master node.
10. Synchronize the Salt modules and update mines:

    ```
    salt "*" saltutil.sync_all
    salt "*" mine.update
    ```

11. Drop the redundant keyrings from the corresponding nodes and verify that the keyrings will not change with the new Salt run:

    > Note
    >
    > If ceph:common:manage_keyring is enabled, modify the last state for each component using the following template:
    >
    > ```
    > salt "<target>" state.sls ceph.common,ceph.setup.keyring,ceph.setup.managed_keyring test=true
    > ```

    - For the OpenStack compute nodes, run:

```
salt "cmp*" cmd.run "rm /etc/ceph/ceph.client.glance.keyring"
salt "cmp*" cmd.run "rm /etc/ceph/ceph.client.gnocchi.keyring"
salt "cmp*" state.sls ceph.common,ceph.setup.keyring test=true
```

- For the Ceph Monitor nodes, run:

```
salt "cmn*" cmd.run "rm /etc/ceph/ceph.client.glance.keyring"
salt "cmn*" cmd.run "rm /etc/ceph/ceph.client.gnocchi.keyring"
salt "cmn*" cmd.run "rm /etc/ceph/ceph.client.nova.keyring"
salt "cmn*" cmd.run "rm /etc/ceph/ceph.client.cinder.keyring"
salt "cmn*" state.sls ceph.common,ceph.setup.keyring test=true
```

- For the RADOS Gateway nodes, run:

```
salt "rgw*" cmd.run "rm /etc/ceph/ceph.client.glance.keyring"
salt "rgw*" cmd.run "rm /etc/ceph/ceph.client.gnocchi.keyring"
salt "rgw*" cmd.run "rm /etc/ceph/ceph.client.nova.keyring"
salt "rgw*" cmd.run "rm /etc/ceph/ceph.client.cinder.keyring"
salt "rgw*" state.sls ceph.common,ceph.setup.keyring test=true
```

- For the Telemetry nodes, run:

```
salt "mdb*" cmd.run "rm /etc/ceph/ceph.client.glance.keyring"
salt "mdb*" cmd.run "rm /etc/ceph/ceph.client.nova.keyring"
salt "mdb*" cmd.run "rm /etc/ceph/ceph.client.cinder.keyring"
salt "mdb*" state.sls ceph.common,ceph.setup.keyring test=true
```

12. Apply the changes for all components one by one:

- If ceph:common:manage_keyring is disabled:

```
salt "<target>" state.sls ceph.common,ceph.setup.keyring
```

- If ceph:common:manage_keyring is enabled:

```
salt "<target>" state.sls ceph.common,ceph.setup.keyring,ceph.setup.managed_keyring
```

Seealso

Update Ceph Upgrade Ceph

Glance operations

This section describes the OpenStack Image service (Glance) operations you may need to perform after the deployment of an MCP cluster.

Enable uploading of an image through Horizon with self-managed SSL certificates

By default, the OpenStack Dashboard (Horizon) supports direct uploading of images to Glance. However, if an MCP cluster is deployed using self-signed certificates for public API endpoints and Horizon, uploading of images to Glance through the Horizon web UI may fail. While accessing the Horizon web UI of such MCP deployment for the first time, a warning informs that the site is insecure and you must force trust the certificate of this site. However, when trying to upload an image directly from the web browser, the certificate of the Glance API is still not considered by the web browser as a trusted one since host:port of the site is different. In this case, you must explicitly trust the certificate of the Glance API.

To enable uploading of an image through Horizon with self-managed SSL certificates:

1. Navigate to the Horizon web UI.
2. On the page that opens, configure your web browser to trust the Horizon certificate if you have not done so yet:
   - In Google Chrome or Chromium, click Advanced > Proceed to <URL> (unsafe).
   - In Mozilla Firefox, navigate to Advanced > Add Exception, enter the URL in the Location field, and click Confirm Security Exception.

   > Note
   >
   > For other web browsers, the steps may vary slightly.

3. Navigate to Project > API Access.
4. Copy the Service Endpoint URL of the Image service.
5. Open this URL in a new window or tab of the same web browser.
6. Configure your web browser to trust the certificate of this site as described in the step 2.

   As a result, the version discovery document should appear with contents depending on the OpenStack version. For example, for OpenStack Ocata:


```
{"versions": [{"status": "CURRENT", "id": "v2.5", "links": \
[{"href": "http://cloud-cz.bud.mirantis.net:9292/v2/", "rel": "self"}]}, \
{"status": "SUPPORTED", "id": "v2.4", "links": \
[{"href": "http://cloud-cz.bud.mirantis.net:9292/v2/", "rel": "self"}]}, \
{"status": "SUPPORTED", "id": "v2.3", "links": \
[{"href": "http://cloud-cz.bud.mirantis.net:9292/v2/", "rel": "self"}]}, \
{"status": "SUPPORTED", "id": "v2.2", "links": \
[{"href": "http://cloud-cz.bud.mirantis.net:9292/v2/", "rel": "self"}]}, \
{"status": "SUPPORTED", "id": "v2.1", "links": \
[{"href": "http://cloud-cz.bud.mirantis.net:9292/v2/", "rel": "self"}]}, \
```

```
{"status": "SUPPORTED", "id": "v2.0", "links": \
[{"href": "http://cloud-cz.bud.mirantis.net:9292/v2/", "rel": "self"}]}]}
```


Once done, you should be able to upload an image through Horizon with self-managed SSL certificates.

Telemetry operations

This section describes the Tenant Telemetry service (Ceilometer) operations you may need to perform after the deployment of an MCP cluster.

©2025, Mirantis Inc.

Enable the Gnocchi archive policies in Tenant Telemetry

The Gnocchi archive policies allow you to define the aggregation and storage policies for metrics received from Ceilometer.

Each archive policy definition is set as the number of points over a timespan. The default archive policy contains two definitions and one rule. It allows you to store metrics for seven days with granularity of one minute and for 365 days with granularity of one hour. It is applied to any metrics sent to Gnocchi with the metric pattern *. You can customize all parameters on the cluster level of your Reclass model.

To enable the Gnocchi archive policies:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In /openstack/telemetry.yml, verify that the following class is present:

   ```
   classes:
   ...
   - system.ceilometer.server.backend.gnocchi
   ```

3. In /openstack/control/init.yml, add the following classes:

   ```
   classes:
   ...
   - system.gnocchi.client
   - system.gnocchi.client.v1.archive_policy.default
   ```

   The parameters of system.gnocchi.client.v1.archive_policy.default are as follows:

   ```
   parameters:
     _param:
       gnocchi_default_policy_granularity_1: '0:01:00'
       gnocchi_default_policy_points_1: 10080
       gnocchi_default_policy_timespan_1: '7 days'
       gnocchi_default_policy_granularity_2: '1:00:00'
       gnocchi_default_policy_points_2: 8760
       gnocchi_default_policy_timespan_2: '365 days'
       gnocchi_default_policy_rule_metric_pattern: '"*"'
     gnocchi:
       client:
         resources:
           v1:
             enabled: true
             cloud_name: 'admin_identity'
             archive_policies:
               default:
                 definition:
                   - granularity: "${_param:gnocchi_default_policy_granularity_1}"
                     points: "${_param:gnocchi_default_policy_points_1}"
   ```

```
          timespan: "${_param:gnocchi_default_policy_timespan_1}"
        - granularity: "${_param:gnocchi_default_policy_granularity_2}"
          points: "${_param:gnocchi_default_policy_points_2}"
          timespan: "${_param:gnocchi_default_policy_timespan_2}"
      rules:
        default:
          metric_pattern: "${_param:gnocchi_default_policy_rule_metric_pattern}"
```

4. Optional. Specify additional archive policies as required. For example, to aggregate the CPU and disk-related metrics with the timespan of 30 days and granularity 1, add the following parameters to /openstack/control/init.yml under the default Gnocchi archive policy parameters:

```
parameters:
  _param:
  ...
  gnocchi:
    client:
      resources:
        v1:
          enabled: true
          cloud_name: 'admin_identity'
          archive_policies:
            default:
            ...
            cpu_disk_policy:
              definition:
                - granularity: '0:00:01'
                  points: 2592000
                  timespan: '30 days'
              rules:
              cpu_rule:
                metric_pattern: 'cpu*'
              disk_rule:
                metric_pattern: 'disk*'
```

> **Caution!**
>
> Rule names defined across archive policies must be unique.

5. Log in to the Salt Master node.
6. Apply the following states:

```
salt -C 'I@gnocchi:client and *01*' saltutil.pillar_refresh
salt -C 'I@gnocchi:client and *01*' state.sls gnocchi.client
salt -C 'I@gnocchi:client' state.sls gnocchi.client
```

7. Verify that the archive policies are set successfully:

    1. Log in to any OpenStack controller node.
    2. Boot a test VM:

    ```
    source keystonercv3
    openstack server create --flavor <flavor_id> \
    --nic net-id=<net_id> --image <image_id>  test_vm1
    ```

    3. Run the following command:

    ```
    openstack metric list | grep <vm_id>
    ```

    Use the vm_id parameter value from the output of the command that you run in the previous step.

    Example of system response extract:

    ```
    +---------+------------------+------------------------------+-------+-----------+
    | id      |archive_policy/name| name                        | unit |resource_id|
    +---------+------------------+------------------------------+-------+-----------+
    | 0ace... | cpu_disk_policy  | disk.allocation              | B    | d9011...  |
    | 0ca6... | default          | perf.instructions            | None | d9011...  |
    | 0fcb... | default          | compute.instance.booting.time | sec  | d9011...  |
    | 10f0... | cpu_disk_policy  | cpu_l3_cache                 | None | d9011...  |
    | 2392... | default          | memory                       | MB   | d9011...  |
    | 2395... | cpu_disk_policy  | cpu_util                     | %    | d9011...  |
    | 26a0... | default          | perf.cache.references        | None | d9011...  |
    | 367e... | cpu_disk_policy  | disk.read.bytes.rate         | B/s  | d9011...  |
    | 3857... | default          | memory.bandwidth.total       | None | d9011...  |
    | 3bb2... | default          | memory.usage                 | None | d9011...  |
    | 4288... | cpu_disk_policy  | cpu                          | ns   | d9011...  |
    +---------+------------------+------------------------------+-------+-----------+
    ```

In the example output above, all metrics are aggregated using the default archive policy except for the CPU and disk metrics aggregated by cpu_disk_policy. The cpu_disk_policy parameters were previously customized in the Reclass model.

Add availability zone to Gnocchi instance resource

> Note
>
> This feature is available starting from the MCP 2019.2.7 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes how to add availability zones to a Gnocchi instance and consume the consuming instance.create.end events.

Add an availability zone to a Gnocchi instance resource:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In /openstack/telemetry.yml, set the create_resources parameter to True:

   ```
   ceilometer:
     server:
       publisher:
         gnocchi:
           enabled: True
           create_resources: True
   ```

3. From the Salt Master node, apply the following state:

   ```
   salt -C 'I@ceilometer:server' saltutil.refresh_pillar
   salt -C 'I@ceilometer:server' state.apply ceilometer.server
   ```

Migrate from GlusterFS to rsync for fernet and credential keys rotation

By default, the latest MCP deployments use rsync for fernet and credential keys rotation. Though, if your MCP version is 2018.8.0 or earlier, GlusterFS is used as a default rotation driver and credential keys rotation driver. This section provides an instruction on how to configure your MCP OpenStack deployment to use rsync with SSH instead of GlusterFS.

To migrate from GlusterFS to rsync:

1. Log in to the Salt Master node.
2. On the system level, verify that the following class is included in keystone/server/cluster.yml:

   ```
   - system.keystone.server.fernet_rotation.cluster
   ```

   > Note
   >
   > The default configuration for the system.keystone.server.fernet_rotation.cluster class is defined in keystone/server/fernet_rotation/cluster.yml. It includes the default list of nodes to synchronize fernet and credential keys that are sync_node01 and sync_node02. If there are more nodes to synchronize fernet and credential keys, expand this list as required.

3. Verify that the crontab job is disabled in the keystone/client/core.yml and keystone/client/single.yml system-level files:

   ```yaml
   linux:
     system:
       job:
         keystone_job_rotate:
           command: '/usr/bin/keystone-manage fernet_rotate --keystone-user keystone --keystone-group keystone >> /var/log/key_rotation_log 2>> /var/log/key_rotation_log'
           enabled: false
           user: root
           minute: 0
   ```

4. Apply the Salt orchestration state to configure all required prerequisites like creating an SSH public key, uploading it to mine and secondary control nodes:

   ```
   salt-run state.orchestrate keystone.orchestrate.deploy
   ```

5. Apply the keystone.server state to put the Keystone rotation script and run it in the sync mode hence fernet and credential keys will be synchronized with the Keystone secondary nodes:

   ```
   salt -C 'I@keystone:server:role:primary' state.apply keystone.server
   salt -C 'I@keystone:server' state.apply keystone.server
   ```

6. Apply the linux.system state to add crontab jobs for the Keystone user:

```
salt -C 'I@keystone:server' state.apply linux.system
```

7. On all OpenStack Controller nodes:

   1. Copy the current credential and fernet keys to temporary directories:

      ```
      mkdir /tmp/keystone_credential /tmp/keystone_fernet
      cp /var/lib/keystone/credential-keys/* /tmp/keystone_credential
      cp /var/lib/keystone/fernet-keys/* /tmp/keystone_fernet
      ```

   2. Unmount the related GlusterFS mount points:

      ```
      umount /var/lib/keystone/credential-keys
      umount /var/lib/keystone/fernet-keys
      ```

   3. Copy the keys from the temporary directories to var/lib/keystone/credential-keys/ and /var/lib/keystone/fernet-keys/:

      ```
      mkdir -p /var/lib/keystone/credential-keys/ /var/lib/keystone/fernet-keys/
      cp /tmp/keystone_credential/* /var/lib/keystone/credential-keys/
      cp /tmp/keystone_fernet/* /var/lib/keystone/fernet-keys/
      chown -R keystone:keystone /var/lib/keystone/credential-keys/*
      chown -R keystone:keystone /var/lib/keystone/fernet-keys/*
      ```

8. On a KVM node, stop and delete the keystone-credential-keys and keystone-keys volumes:

   1. Stop the volumes:

      ```
      gluster volume stop keystone-credential-keys
      gluster volume stop keystone-keys
      ```

   2. Delete the GlusterFS volumes:

      ```
      gluster volume delete keystone-credential-keys
      gluster volume delete keystone-keys
      ```

9. On the cluster level model, remove the following GlusterFS classes included in the openstack/control.yml file by default:

```
- system.glusterfs.server.volume.keystone
- system.glusterfs.client.volume.keystone
```

Disable the Memcached listener on the UDP port

Starting from the Q4‘18 MCP release, to reduce the attack surface and increase the product security, Memcached on the controller nodes listens on TCP only. The UDP port for Memcached is disabled by default. This section explains how to disable the UDP listeners for the existing OpenStack environments deployed on top of the earlier MCP versions.

To disable the Memcached listener on the UDP port:

1. Log in to the Salt Master node.
2. Update your Reclass metadata model.
3. Verify the memcached:server pillar:

   ```
   salt ctl01* pillar.get memcached:server
   ```

   The memcached:server:bind:proto pillar should be available after update of the Reclass metadata model and set to False for proto:udp:enabled for all Memcached server instances.

   Example of system response:

   ```
   -- start output --
     ----------
     bind:
         ----------
         address:
             0.0.0.0
         port:
             11211
         proto:
             ----------
             tcp:
                 ----------
                 enabled:
                     True
             udp:
                 ----------
                 enabled:
                     False
         protocol:
             tcp
     enabled:
         True
     maxconn:
         8192
   -- end output --
   ```

4. Run the memcached.server state to apply the changes to all memcached instances:

```
salt -C 'I@memcached:server' state.sls memcached.server
```

Configuring rate limiting with NGINX

MCP enables you to limit the number of HTTP requests that a user can make in a given period of time for your OpenStack deployments. The rate limiting with NGINX can be used to protect an OpenStack environment against DDoS attacks as well as to protect the community application servers from being overwhelmed by too many user requests at the same time.

For rate limiting configuration, MCP supports the following NGINX modules:

- ngx_http_geo_module
- ngx_http_map_module
- ngx_http_limit_req_module
- ngx_http_limit_conn_module

This section provides the related NGINX directives description with the configuration samples which you can use to enable rate limiting in your MCP OpenStack deployment.

Starting from the MCP 2019.2.20 maintenance update, you can also configure request limiting for custom locations.

NGINX rate limiting configuration sample

This section includes the configuration sample of NGINX rate limiting feature that enables you to limit the number of HTTP requests a user can make in a given period of time.

In the sample, all clients except for 10.12.100.1 are limited to 1 request per second. More specifically, the sample illustrates how to:

- Create a geo instance that will match the IP address and set the limit_action variable where 0 stands for unlimited and 1 stands for limited.
- Create global_geo_limiting_map that will map ip_limit_key to ip_limit_action.
- Create a global limit_req_zone zone called global_limit_zone that limits the number of requests to 1 request per second.
- Apply global_limit_zone globally to all requests with 5 requests burst and nodelay.

Configuration sample:

```
nginx:
  server:
    enabled: true
    geo:
      enabled: true
      items:
        global_geo_limiting:
          enabled: true
          variable: ip_limit_key
          body:
            default:
              value: '1'
            unlimited_client1:
              name: '10.12.100.1/32'
              value: '0'
    map:
      enabled: true
      items:
        global_geo_limiting_map:
          enabled: true
          string: ip_limit_key
          variable: ip_limit_action
          body:
            limited:
              name: 1
              value: '$binary_remote_addr'
            unlimited:
              name: 0
              value: '""'
    limit_req_module:
      limit_req_zone:
```

```yaml
      global_limit_zone:
        key: ip_limit_action
        size: 10m
        rate: '1r/s'
    limit_req_status: 503
    limit_req:
      global_limit_zone:
        burst: 5
        enabled: true
```

To apply the request limiting to a particular site, define the limit_req on a site level. For example:

```yaml
nginx:
  server:
    site:
      nginx_proxy_openstack_api_keystone:
        limit_req_module:
          limit_req:
            global_limit_zone:
              burst: 5
              nodelay: true
              enabled: true
```

Configuring the geo module

The ngx_http_geo_module module creates variables with values depending on the client IP address.

Syntax	geo [$address] $variable { ... }
Default	—
Context	HTTP

NGINX configuration sample	`geo $my_geo_map {`
	`    default        0;`
	`    127.0.0.1      0;`
	`    10.12.100.1/32 1;`
	`    10.13.0.0/16    2;`
	`    2001:0db8::/32 1;`
	`}`

Example of a Salt pillar for the geo module:

```
nginx:
server:
  geo:
    enabled: true
    items:
      my_geo_map:
        enabled: true
        variable: my_geo_map_variable
        body:
          default:
            value: '0'
          localhost:
            name: 127.0.0.1
            value: '0'
          client:
            name: 10.12.100.1/32
            value: '1'
          network:
            name: 10.13.0.0/16
            value:  '2'
          ipv6_client:
            name: 2001:0db8::/32
            value: '1'
```

All geo variables specified in the pillars, after applying the nginx.server state, will be reflected in the /etc/nginx/conf.d/geo.conf file.

Configuring the mapping

The ngx_http_map_module module creates variables which values depend on values of other source variables specified in the first parameter.

Syntax	map string $variable { ... }
Default	—
Context	HTTP

NGINX configuration sample	`map $my_geo_map_variable $ip_limit_action {`
	`    default "";`
	`    1 $binary_remote_addr;`
	`    0 "";`
	`}`

Example of a Salt pillar for the map module:

```
nginx:
  server:
    map:
      enabled: true
      items:
        global_geo_limiting_map:
          enabled: true
          string: my_geo_map_variable
          variable: ip_limit_action
          body:
            default:
              value: '""'
            limited:
              name: '1'
              value: '$binary_remote_addr'
            unlimited:
              name: '0'
              value: '""'
```

All map variables specified in the pillars, after applying the nginx.server state, will be reflected in the /etc/nginx/conf.d/map.conf file.

Configuring the request limiting

The ngx_http_limit_req_module module limits the request processing rate per a defined key. The module directives include the mandatory limit_req_zone and limit_req directives and an optional limit_req_status directive.

The limit_req_zone directive defines the parameters for the rate limiting.

Syntax	limit_req_zone key zone=name:size rate=rate [sync];
Default	—
Context	HTTP
NGINX configuration sample	`limit_req_zone $binary_remote_addr zone=global_limit_zone1:10m rate=1r/s ;` `limit_req_zone $ip_limit_action zone=global_limit_zone2:10m rate=2r/s ;`

The limit_req directive enables rate limiting within the context where it appears.

Syntax	limit_req zone=name [burst=number] [nodelay | delay=number];
Default	—
Context	HTTP, server, location
NGINX configuration sample	`limit_req zone=global_limit_zone1 burst=2 ;` `limit_req zone=global_limit_zone2 burst=4 nodelay ;`

The limit_req_status directive sets the status code to return in response to rejected requests.

Syntax	limit_req_status code;
Default	limit_req_status 503;
Context	http, server, location that corresponds to the nginx:server and nginx:server:site definitions of a pillar.
NGINX configuration sample	`limit_req_status 429;`

Example of a Salt pillar for limit_req_zone and limit_req:

```yaml
nginx:
server:
  limit_req_module:
    limit_req_zone:
      global_limit_zone1:
        key: binary_remote_addr
        size: 10m
        rate: '1r/s'
      global_limit_zone2:
        key: ip_limit_action
        size: 10m
```

```
      rate: '2r/s'
    limit_req_status: 429
    limit_req:
      global_limit_zone1:
        burst: 2
        enabled: true
      global_limit_zone2:
        burst: 4
        enabled: true
        nodelay: true
```

In the configuration example above, the states are kept in a 10 megabyte global_limit_zone1 and global_limit_zone2 zones. An average request processing rate cannot exceed 1 request per second for global_limit_zone1 and 2 requests per second for global_limit_zone2.

The $binary_remote_addr, a client's IP address, serves as a key for the global_limit_zone1 zone. And the mapped $ip_limit_action variable is a key for the global_limit_zone2 zone.

To apply the request limiting to a particular site, define the limit_req on a site level. For example:

```
nginx:
  server:
    site:
      nginx_proxy_openstack_api_keystone:
        limit_req_module:
          limit_req:
            global_limit_zone:
              burst: 5
              nodelay: true
              enabled: true
```

Configuring the connection limiting

The ngx_http_limit_conn_module module limits the number of connections per defined key. The main directives include limit_conn_zone and limit_conn.

The limit_conn_zone directive sets parameters for a shared memory zone that keeps states for various keys. A state is the current number of connections. The key value can contain text, variables, and their combination. The requests with an empty key value are not accounted.

Syntax	limit_conn_zone key zone=name:size;
Default	—
Context	HTTP
NGINX configuration sample	`limit_conn_zone $binary_remote_addr zone=global_limit_conn_zone:20m;` `limit_conn_zone $binary_remote_addr zone=openstack_web_conn_zone:10m;`

The limit_conn directive sets the shared memory zone and the maximum allowed number of connections for a given key value. When this limit is exceeded, the server returns the error in reply to a request.

Syntax	limit_conn zone number;
Default	—
Context	HTTP, server, location
NGINX configuration sample	`limit_conn global_limit_conn_zone 100;` `limit_conn_status 429;`

Example of a Salt pillar with limit_conn_zone and limit_conn:

```
nginx:
  server:
    limit_conn_module:
      limit_conn_zone:
        global_limit_conn_zone:
          key: 'binary_remote_addr'
          size: 20m
          enabled: true
        api_keystone_conn_zone:
          key: 'binary_remote_addr'
          size: 10m
          enabled: true
      limit_conn:
        global_limit_conn_zone:
          connections: 100
          enabled: true
      limit_conn_status: 429
```

©2025, Mirantis Inc.

To apply the connection limiting to a particular site, define limit_conn on a site level. For example:

```yaml
nginx:
  server:
    site:
      nginx_proxy_openstack_api_keystone:
        limit_conn_module:
          limit_conn_status: 429
          limit_conn:
            api_keystone_conn_zone:
              connections: 50
              enabled: true
```

Configure request limiting for custom locations

> Note
>
> This feature is available starting from the MCP 2019.2.20 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

If required, you can configure request limiting for custom locations. To do so, add the limit pillar in the location:some_location section. The limits configured for a location predetermine the limits configured for a site. The following methods are supported:

- By IP
- By HTTP (get, post, and so on)

Configuration sample:

```
nginx:
  server:
    site:
      nginx_proxy_openstack_api_keystone:
        location:
          /some_location/:
            limit:
              enabled: true
              methods:
                ip:
                  enabled: True
                get:
                  enabled: True
                  rate: 120r/s
                  burst: 600
                  size: 20m
                  nodelay: True
                post:
                  enabled: True
                  rate: 50r/m
                  burst: 80
```

Configure load balancing for Horizon

Starting from the Q4‘18 MCP version, Horizon works in the load balancing mode by default. All requests to Horizon are terminated and forwarded to the Horizon back end by HAProxy bound on a virtual IP address. HAProxy serves as a balancer and manages requests according to the defined policy, which is round-robin by default, among proxy nodes. This approach allows for load reduction on one proxy node and spreading the load among all proxy nodes.

> Note
>
> If the node, which the user is connected to, has failed and the user is reconnected to another node, the user will be logged out from the dashboard. As a result, the The user is not authorized page opens, which is the expected behavior in this use case. To continue working with the dashboard, the user has to sign in to Horizon again from the Log In page.

This section provides the instruction on how to manually configure Horizon load balancing for the existing OpenStack deployments that are based on earlier MCP release versions.

To enable active-active mode for Horizon:

1. Log in to the Salt Master node.
2. Update to the 2019.2.0 Build ID MCP version or higher.
3. Verify that the system.apache.server.site.horizon class has been added to your Reclass model. By default, the class is defined in the ./system/apache/server/site/horizon.yml file on the Reclass system level as follows:

```
parameters:
  _param:
    apache_ssl:
      enabled: false
    apache_horizon_ssl: ${_param:apache_ssl}
    apache_horizon_api_address: ${_param:horizon_server_bind_address}
    apache_horizon_api_host: ${linux:network:fqdn}
  apache:
    server:
      bind:
        listen_default_ports: false
      enabled: true
      default_mpm: event
      modules:
        - wsgi
      site:
        horizon:
          enabled: false
          available: true
```

```
      type: wsgi
      name: openstack_web
      ssl: ${_param:apache_horizon_ssl}
      wsgi:
        daemon_process: horizon
        processes: 3
        threads: 10
        user: horizon
        group: horizon
        display_name: '%{GROUP}'
        script_alias: '/ /usr/share/openstack-dashboard/openstack_dashboard/wsgi/django.wsgi'
        application_group: '%{GLOBAL}'
        authorization: 'On'
      limits:
        request_body: 0
      host:
        address: ${_param:apache_horizon_api_address}
        name: ${_param:apache_horizon_api_host}
        port: 8078
      locations:
        - uri: /static
          path: /usr/share/openstack-dashboard/static
      directories:
        dashboard_static:
          path: /usr/share/openstack-dashboard/static
          order: 'allow,deny'
          allow: 'from all'
          modules:
            mod_expires.c:
              ExpiresActive: 'On'
              ExpiresDefault: '"access 6 month"'
            mod_deflate.c:
              SetOutputFilter: 'DEFLATE'
        dashboard_wsgi:
          path: /usr/share/openstack-dashboard/openstack_dashboard/wsgi
          order: 'allow,deny'
          allow: 'from all'
      log:
        custom:
          format: >-
            %v:%p %{X-Forwarded-For}i %h %l %u %t \"%r\" %>s %D %O \"%{Referer}i\" \"%{User-Agent}i\"
        error:
          enabled: true
          level: debug
          format: '%M'
          file: '/var/log/apache2/openstack_dashboard_error.log'
```

4. Verify that the system.apache.server.site.horizon has been added to the Reclass system level in the ./system/horizon/server/single.yml file as follows:

```
classes:
  - service.horizon.server.single
  - system.horizon.upgrade
  - system.horizon.server.iptables
  - system.apache.server.single
```

```
  - system.memcached.server.single
  - system.apache.server.site.horizon
```

5. Verify that the definition for the system.haproxy.proxy.listen.openstack.openstack_web class has been added to the Reclass cluster level in the in the proxy nodes configuration file:

```
parameters:
  _param:
    haproxy_openstack_web_check_params: check
  haproxy:
    proxy:
      listen:
        openstack_web:
          type: custom
          check: false
          sticks: ${_param:haproxy_openstack_web_sticks_params}
          binds:
          - address: ${_param:cluster_vip_address}
            port: ${_param:haproxy_openstack_web_bind_port}
          servers:
          - name: ${_param:cluster_node01_hostname}
            host: ${_param:cluster_node01_address}
            port: 8078
            params: ${_param:haproxy_openstack_web_check_params}
          - name: ${_param:cluster_node02_hostname}
            host: ${_param:cluster_node02_address}
            port: 8078
            params: ${_param:haproxy_openstack_web_check_params}
```

6. Add the system.haproxy.proxy.listen.openstack.openstack_web class to the Horizon node configuration file, for example, cluster/<cluster_name>/openstack/dashboard.yml:

```
classes:
  - system.haproxy.proxy.listen.openstack.openstack_web
```

7. In the Horizon node configuration file (edited in the previous step), define the host names and IP addresses for all proxy nodes used in the deployment for the dashboard node and verify that the HAProxy checks the availability of Horizon.

   Congifuration example for two proxy nodes:

```
parameters:
  _param:
    cluster_node01_hostname: ${_param:openstack_proxy_node01_hostname}
    cluster_node01_address: ${_param:openstack_proxy_node01_address}
    cluster_node02_hostname: ${_param:openstack_proxy_node02_hostname}
```

```
    cluster_node02_address: ${_param:openstack_proxy_node02_address}
    haproxy_openstack_web_bind_port: ${_param:horizon_public_port}
    haproxy_openstack_web_check_params: check inter 10s fastinter 2s downinter 3s rise 3 fall 3 check-ssl verify none
  horizon:
    server:
      cache:
        ~members:
        - host: ${_param:openstack_proxy_node01_address}
          port: 11211
        - host: ${_param:openstack_proxy_node02_address}
          port: 11211
```

8. If the HTTP to HTTPS redirection will be used, add the following configuration to the Horizon node configuration file:

```
parameters:
  haproxy:
    proxy:
      listen:
        openstack_web_proxy:
          mode: http
          format: end
          force_ssl: true
          binds:
          - address: ${_param:cluster_vip_address}
            port: 80
```

9. Disable the NGINX servers requests for Horizon by replacing the NGINX class with the HAProxy class in the proxy node configuration file.

   Replace:

```
- system.nginx.server.proxy.openstack_web
```

   with

```
- system.haproxy.proxy.single
```

10. Remove the nginx_redirect_openstack_web_redirect.conf and nginx_proxy_openstack_web.conf Horizon sites from /etc/nginx/sites-enabled/.

11. Restart the NGINX service on the proxy nodes:

```
salt 'prx*' cmd.run 'systemctl restart nginx'
```

12. Verify that Keepalived keeps track on HAProxy by adding the haproxy variable for the keepalived_vrrp_script_check_multiple_processes parameter:

```
parameters:
  _param:
    keepalived_vrrp_script_check_multiple_processes: 'nginx haproxy'
```

13. Enable SSL for Horizon:

```
parameters:
  _param:
    apache_ssl:
      enabled: true
      authority: ${_param:salt_minion_ca_authority}
      engine: salt
      key_file:  /srv/salt/pki/${_param:cluster_name}/${salt:minion:cert:proxy:common_name}.key
      cert_file: /srv/salt/pki/${_param:cluster_name}/${salt:minion:cert:proxy:common_name}.crt
      chain_file: /srv/salt/pki/${_param:cluster_name}/${salt:minion:cert:proxy:common_name}-with-chain.crt
```

14. Define the address to be bound by Memcached in the cluster/<cluster_name>/openstack/proxy.yml file:

```
parameters:
  _param:
    openstack_memcached_server_bind_address: ${_param:single_address}
```

15. Verify that the Horizon Salt formula is updated to the the version higher than 2016.12.1+201812072002.e40b950 and the Apache Salt formula is updated to the version higher than 0.2+201811301717.acb3391.

16. Delete the NGINX sites from the proxy nodes that proxy Horizon requests and possible redirection from HTTP to HTTPS.

17. Apply the haproxy and horizon states on the proxy nodes:

```
salt -C 'I@horizon:server' state.sls horizon
salt -C 'I@horizon:server' state.sls haproxy
```

Expose a hardware RNG device to Nova instances

> Warning
>
> This feature is available starting from the MCP 2019.2.3 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

MCP enables you to define the path to an Random Number Generator (RNG) device that will be used as the source of entropy on the host. The default source of entropy is /dev/urandom. Other available options include /dev/random and /dev/hwrng.

The example structure of the RNG definition in the Nova pillar:

```
nova:
  controller:
    libvirt:
      rng_dev_path: /dev/random

  compute:
    libvirt:
      rng_dev_path: /dev/random
```

The procedure included in this section can be used for both existing and new MCP deployments.

To define the path to an RNG device:

1. Log in to the Salt Master node.
2. In classes/cluster/<cluster_name>/openstack/control.yml, define the rng_dev_path parameter for nova:contoroller:

   ```
   nova:
     controller:
       libvirt:
         rng_dev_path: /dev/random
   ```

3. In classes/cluster/<cluster_name>/openstack/compute/init.yml, define the rng_dev_path parameter for nova:compute:

   ```
   nova:
     compute:
       libvirt:
         rng_dev_path: /dev/random
   ```

4. Apply the changes:

```
salt -C 'I@nova:controller' state.sls nova.controller
salt -C 'I@nova:compute' state.sls nova.compute
```

Set the directory for lock files

Note

This feature is available starting from the MCP 2019.2.7 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

You can set the directory for lock files for the Ceilometer, Cinder, Designate, Glance, Ironic, Neutron, and Nova OpenStack services by specifying the lock_path parameter in the Reclass model. This section provides the example of the lock path configuration for Nova.

To set the lock path for Nova:

1. Open your Git project repository with the Reclass model on the cluster level.
2. Define the lock_path parameter:
   1. In openstack/control.yml, specify:

      ```
      parameters:
        nova:
          controller:
            concurrency:
              lock_path: '/var/lib/nova/tmp'
      ```

   2. In openstack/compute.yml, specify:

      ```
      parameters:
        nova:
          compute:
            concurrency:
              lock_path: '/var/lib/nova/tmp'
      ```

3. Apply the changes from the Salt Master node:

   ```
   salt -C 'I@nova:controller or I@nova:compute' saltutil.refresh_pillar
   salt -C 'I@nova:controller' state.apply nova.controller
   salt -C 'I@nova:compute' state.apply nova.compute
   ```

Add the Nova CpuFlagsFilter custom filter

> Note
>
> This feature is available starting from the MCP 2019.2.10 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

CpuFlagsFilter is a custom Nova scheduler filter for live migrations. The filter ensures that the CPU features of a live migration source host match the target host. Use the CpuFlagsFilter filter only if your deployment meets the following criteria:

- The CPU mode is set to host-passthrough or host-model. For details, see MCP Deployment Guide: Configure a CPU model.
- The OpenStack compute nodes have heterogeneous CPUs.
- The OpenStack compute nodes are not organized in aggregates with the same CPU in each aggregate.

To add the Nova CpuFlagsFilter custom filter:

1. Open your project Git repository with the Reclass model on the cluster level.
2. Open the classes/cluster/<cluster_name>/openstack/control.yml file for editing.
3. Verify that the cpu_mode parameter is set to host-passthrough or host-model.
4. Add CpuFlagsFilter to the scheduler_default_filters parameter for nova:contoroller:

   ```
   nova:
     controller:
       scheduler_default_filters: “DifferentHostFilter,SameHostFilter,RetryFilter,AvailabilityZoneFilter,RamFilter,CoreFilter,DiskFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter,PciPassthroughFilter,NUMATopologyFilter,AggregateInstanceExtraSpecsFilter,CpuFlagsFilter”
   ```

5. Log in to the Salt Master node.
6. Apply the changes:

   ```
   salt -C 'I@nova:controller' state.sls nova.controller
   ```

Disable Nova cell mapping

> Note
>
> This feature is available starting from the MCP 2019.2.16 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

You may need to disable cell mapping and database migrations for the OpenStack Compute service (Nova), for example, to faster redeploy the nova state.

> Note
>
> Enable Nova cell mapping before performing any update or upgrade.

To disable Nova cell mapping:

1. In defaults/openstack/init.yml on the system Reclass level, set the nova_control_update_cells parameter to False:

   ```
   _param:
     nova_control_update_cells: False
   ```

2. Log in to the Salt Master node.
3. Refresh cache, synchronize Salt pillars and resources:

   ```
   salt '*' saltutil.clear_cache && \
   salt '*' saltutil.refresh_pillar && \
   salt '*' saltutil.sync_all
   ```

Clean up an OpenStack database

> Note
>
> This feature is available starting from the MCP 2019.2.12 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

Using the Deploy - Openstack Database Cleanup Jenkins pipeline job, you can automatically clean up stale records from the Nova, Cinder, Heat, or Glance database to make it smaller. This is helpful before any update or upgrade activity. You can execute the Deploy - Openstack Database Cleanup Jenkins pipeline job without a maintenance window, just as for an online dbsync.

To clean up an OpenStack database:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In <cluster_name>/openstack/control.yml, specify the pillars below with the following parameters:

    - Set db_purge to True.
    - Set days as required. If you skip setting the days parameter:
        - For Nova and Heat, all stale records will be archived or purged.
        - For Cinder on OpenStack Pike, records older than 1 day will be deleted. For Cinder on OpenStack Queens, all entries will be deleted.
        - For Glance, all entries will be deleted.
    - For Nova, set max_rows to limit the rows for deletion. It is safe to specify 500-1000 rows. Unlimited cleanup may cause the database being inaccessible and, as a result, OpenStack being inoperable.

    For example:

    - For Nova:

    ```
    nova:
      controller:
        db_purge:
          enabled: True
          max_rows: 1000
          days: 45
    ```

    - For Cinder:

    ```
    cinder:
      controller:
    ```

```
    db_purge:
      enabled: True
      days: 45
```

- For Heat:

```
heat:
  server:
    db_purge:
      enabled: True
      days: 45
```

- For Glance Available since 2019.2.17 maintenance update:

```
glance:
  server:
    db_purge:
      enabled: True
      days: 45
```

3. Log in to the Jenkins web UI.
4. Open the Deploy - Openstack Database Cleanup Jenkins pipeline job.
5. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.

6. Click Deploy.

The Jenkins pipeline job workflow:

1. For Nova, move the deleted rows from the production tables to shadow tables. Data from shadow tables is purged to save disk space.
2. For Cinder, purge the database entries that are marked as deleted.
3. For Heat, purge the database entries that are marked as deleted.
4. For Glance, purge the database entries that are marked as deleted.

Galera operations

This section describes the Galera service operations you may need to perform after the deployment of an MCP cluster.

Configure arbitrary Galera parameters

> Note
>
> This feature is available starting from the MCP 2019.2.13 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes how to configure arbitrary Galera parameters using the wsrep_provider_options Galera variable. For details, see Galera parameters.

To configure arbitrary Galera parameters:

1. Open your project Git repository with the Reclass model on the cluster level.
2. In openstack/database/init.yml, specify wsrep_provider_options as required. For example:

```
parameters:
    ...
  galera:
    master:
      wsrep_provider_options:
        gcomm.thread_prio: "rr:2"
    slave:
      wsrep_provider_options:
        gcomm.thread_prio: "rr:2"
     ...
```

3. Log in to the Salt Master node.
4. Apply the changes to Galera:

```
salt -C 'I@galera:master or I@galera:slave' saltutil.refresh_pillar
salt -C 'I@galera:master or I@galera:slave' state.apply galera
```

Enable Cinder coordination

Note

This feature is available starting from the MCP 2019.2.16 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

To avoid race conditions in the OpenStack Block Storage service (Cinder) with an active/active configuration, you can use a coordination manager system with MySQL as a back end. For example, to prevent deletion of a volume that is being used to create another volume or prevent from attaching a volume that is already being attached.

To enable Cinder coordination:

1. From the Salt Master node, verify that the salt-formula-cinder package version is 2016.12.1+202108101137.19b6edd~xenial1_all or later.
2. From an OpenStack controller node, verify that the python-tooz package version is 1.60.2-1.0~u16.04+mcp4 or later.
3. Open the cluster level of your deployment model.
4. In <cluster_name>/openstack/control.yml, specify the following configuration:

   ```
   cinder:
     controller:
       coordination:
         enabled: true
         backend: mysql
   ```

5. Log in to the Salt Master node.
6. Apply the changes:

   ```
   salt -C 'I@cinder:controller' state.sls cinder
   ```

Kubernetes operations

Caution!

Kubernetes support termination notice

Starting with the MCP 2019.2.5 update, the Kubernetes component is no longer supported as a part of the MCP product. This implies that Kubernetes is not tested and not shipped as an MCP component. Although the Kubernetes Salt formula is available in the community driven SaltStack formulas ecosystem, Mirantis takes no responsibility for its maintenance.

Customers looking for a Kubernetes distribution and Kubernetes lifecycle management tools are encouraged to evaluate the Mirantis Kubernetes-as-a-Service (KaaS) and Docker Enterprise products.

This section includes topics that describe operations with your Kubernetes environment.

©2025, Mirantis Inc.

Monitor connectivity between the Kubernetes nodes using Netchecker

Caution!

Kubernetes support termination notice

Starting with the MCP 2019.2.5 update, the Kubernetes component is no longer supported as a part of the MCP product. This implies that Kubernetes is not tested and not shipped as an MCP component. Although the Kubernetes Salt formula is available in the community driven SaltStack formulas ecosystem, Mirantis takes no responsibility for its maintenance.

Customers looking for a Kubernetes distribution and Kubernetes lifecycle management tools are encouraged to evaluate the Mirantis Kubernetes-as-a-Service (KaaS) and Docker Enterprise products.

The Mirantis Cloud Platform automatically deploys Netchecker as part of an MCP Kubernetes Calico-based deployment. Netchecker enables network connectivity and network latency monitoring for the Kubernetes nodes.

This section includes topics that describe how to configure and use Netchecker.

View Netchecker metrics

MCP automatically configures Netchecker during the deployment of the Kubernetes cluster. Therefore, Netchecker starts gathering metrics as soon as the Kubernetes cluster is up and running. You can view Netchecker metrics to troubleshoot connectivity between the Kubernetes nodes.

To view Netchecker metrics:

1. Log in to the Kubernetes Master node.
2. Obtain the IP address of the Netchecker server pod:

   ```
   kubectl get pod -o json --selector='app==netchecker-server' -n \
   netchecker | grep podIP
   ```

3. Obtain the Netchecker container port number:

   ```
   kubectl get pod -o json --selector='app==netchecker-server' -n \
   netchecker | grep containerPort
   ```

4. View all metrics provided by Netchecker:

   ```
   curl <netchecker-pod-ip>:<port>/metrics
   ```

5. View the list of Netchecker agents metrics:

   ```
   curl <netchecker-pod-ip>:<port>/metrics | grep ncagent
   ```

   Example of system response:

   ```
   # HELP ncagent_error_count_total Total number of errors (keepalive miss
   # count) for the agent.
   # TYPE ncagent_error_count_total counter
   ncagent_error_count_total{agent="cmp01-private_network"} 0
   ncagent_error_count_total{agent="cmp02-private_network"} 0
   ncagent_error_count_total{agent="ctl01-private_network"} 0
   ncagent_error_count_total{agent="ctl02-private_network"} 0
   ncagent_error_count_total{agent="ctl03-private_network"} 0

   ...
   ```

   For the list of Netchecker metrics, see: Netchecker metrics description.

Netchecker metrics description

The following table lists Netchecker metrics. The metrics with the ncagent_ prefix are used to monitor the Kubernetes environment.

Netchecker metrics

Metric	Description
go_*	A set of default golang metrics provided by the Prometheus library.
process_*	A set of default process metrics provided by the Prometheus library.
ncagent_report_count_total (label agent)	A counter that calculates the number of total reports from every Netchecker agent separated by label.
ncagent_error_count_total (label agent)	A counter that calculates the number of total errors from every agent separated by label. Netchecker increases the value of the counter each time the Netchecker agent fails to send a report within the reporting_interval * 2 timeframe.
ncagent_http_probe_connection_result	A gauge that represents the connection result between an HTTP server and a Netchecker agent. Possible values: 0 - error, 1 - success.
ncagent_http_probe_code	A gauge that represents the HTTP status code. Returns 0 if there is no HTTP response.
ncagent_http_probe_total_time_ms	A gauge that represents the total duration of an HTTP transaction.
ncagent_http_probe_content_transfer_time_ms	A gauge that represents the duration of content transfer from the first response byte till the end (in ms).
ncagent_http_probe_tcp_connection_time_ms	A gauge that represents the TCP connection establishing time in ms.
ncagent_http_probe_dns_lookup_time_ms	A gauge that represents the DNS lookup time in ms.
ncagent_http_probe_connect_time_ms	A gauge that represents connection time in ms.
ncagent_http_probe_server_processing_time_ms	A gauge that represents the server processing time in ms.

©2025, Mirantis Inc.

Transition to containers

> Caution!
>
> Kubernetes support termination notice
>
> Starting with the MCP 2019.2.5 update, the Kubernetes component is no longer supported as a part of the MCP product. This implies that Kubernetes is not tested and not shipped as an MCP component. Although the Kubernetes Salt formula is available in the community driven SaltStack formulas ecosystem, Mirantis takes no responsibility for its maintenance.
>
> Customers looking for a Kubernetes distribution and Kubernetes lifecycle management tools are encouraged to evaluate the Mirantis Kubernetes-as-a-Service (KaaS) and Docker Enterprise products.

Transitioning from virtual machines to containers is a lengthy and complex process that in some environments may take years. If you want to leverage Kubernetes features while you continue using the existing applications that run in virtual machines, Mirantis provides an interim solution that enables running virtual machines orchestrated by Kubernetes.

To enable Kubernetes to run virtual machines, you need to deploy and configure a virtual machine runtime for Kubernetes called Virtlet. Virtlet is a Kubernetes Container Runtime Interface (CRI) implementation that is packaged as a Docker image and contains such components as libvirt daemon, QEMU/KVM wrapper, and so on.

Virtlet enables you to run unmodified QEMU/KVM virtual machines that do not include an additional containerd layer as in similar solutions in Kubernetes. Virtlet supports all standard Kubernetes objects, such as ReplicaSets, deployments, DaemonSets, and so on, as well as their operations. For information on operations with Kubernetes objects, see: Kubernetes documentation.

Unmodified QEMU/KVM virtual machines enable you to run:

- Unikernels
- Applications that are hard to containerize
- NFV workloads
- Legacy applications

Compared to regular Kubernetes pods, Virtlet pods have the following limitations:

- Only one virtual machine per pod is allowed.
- Virtual machine volumes (pod volumes) must be specified using the FlexVolume driver. Standard Kubernetes directory-based volumes are not supported except for the use case of Kubernetes Secrets and ConfigMaps. If a Secret or a ConfigMap is mounted to a VM pod, its content is copied into an appropriate location inside the VM using the cloud-init mechanism.

- No support for kubectl exec.

For details on Virtlet operations, see: Virtlet documentation.

This section describes how to create and configure Virtlet pods as well as provides examples of pods for different services.

For an instruction on how to update Virtlet, see Update Virtlet.

Prerequisites

To have a possibility to run virtual machines as Kubernetes pods, your environment must meet the following prerequisites:

- An operational Kubernetes environment with enabled Virtlet functionality.
- SELinux and AppArmor must be disabled on the Kubernetes nodes.
- The Kubernetes node names must be resolvable by the DNS server configured on the Kubernetes nodes.

Example of a pod configuration

You need to define a pod for each virtual machine that you want to place under Kubernetes orchestration.

Pods are defined as .yaml files. The following text is an example of a pod configuration for a VM with Virtlet:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: cirros-vm
  annotations:
    kubernetes.io/target-runtime: virtlet
    VirtletVCPUCount: "1"
    VirtletSSHKeys: |
      ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCaJEcFDXEK2Zb
      X0ZLS1EIYFZRbDAcRfuVjpstSc0De8+sV1aiu+dePxdkuDRwqFt
      Cyk6dEZkssjOkBXtri00MECLkir6FcH3kKOJtbJ6vy3uaJc9w1E
      Ro+wyl6SkAh/+JTJkp7QRXj8oylW5E20LsbnA/dIwWzAF51PPwF
      7A7FtNg9DnwPqMkxFo1Th/buOMKbP5ZA1mmNNtmzbMpMfJATvVy
      iv3ccsSJKOiyQr6UG+j7sc/7jMVz5Xk34Vd0l8GwcB0334MchHc
      kmqDB142h/NCWTr8oLakDNvkfC1YneAfAO41hDkUbxPtVBG5M/o
      7P4fxoqiHEX+ZLfRxDtHB53 me@localhost
    VirtletCloudInitUserDataScript: |
      #!/bin/sh
      echo "Hi there"
spec:
 affinity:
   nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
       nodeSelectorTerms:
       - matchExpressions:
         - key: extraRuntime
           operator: In
           values:
           - virtlet
containers:
- name: cirros-vm
  image: virtlet/download.cirros-cloud.net/0.3.5/
  cirros-0.3.5-x86_64-disk.img
  resources:
    limits:
      memory: 128Mi
```

The following table describes the pod configuration parameters:

Pod definition paramaters

Parameter	Description
apiVersion	Version of the Kubernetes API.
kind	Type of the file. For all pod configurations, the kind parameter value is Pod.

metadata	Specifies a number of parameters required for the pod configuration, including:
	• name - the name of the pod.
	• annotations - a subset of metadata parameters in the form of strings. Numeric values must be quoted:
	  • kubernetes.io/target-runtime - defines that this pod belongs to the Virtlet runtime.
	  • VirtletVCPUCount - (optional) specifies the number of virtual CPUs. The default value is 1.
	  • VirtletSSHKeys - one or many SSH keys, one key per line.
	  • VirtletCloudInitUserDataScript - user data for the cloud-init script.

spec	Pod specification, including:
	• nodeAffinity - the specification in the example above ensures that Kubernetes runs this pod only on the nodes that have the extraRuntime=virtlet label. This label is used by the Virtlet DaemonSet to select nodes that must have the Virtlet runtime.
	• containers - a container configuration that includes:
	  • name - name of the container.
	  • image - specifies the path to a network location where the Docker image is stored. The path must start with the virtlet prefix followed by the URL to the required location.
	  • resources - defines the resources, such as memory limitation for the libvirt domain.

Example of a pod definition for an ephemeral device

Virtlet stores all ephemeral volumes in the local libvirt pool storage in var/lib/virtlet/volumes. The volume configuration is defined under the volume section.

The following text is an example of a pod (virtual machine) definition with an ephemeral volume of 2048 MB capacity.

```
apiVersion: v1
kind: Pod
metadata:
  name: test-vm-pod
  annotations:
    kubernetes.io/target-runtime: virtlet
spec:
  affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: extraRuntime
          operator: In
          values:
          - virtlet
  containers:
  - name: test-vm
    image: download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
    volumeMounts:
 - name: containerd
   mountPath: /var/lib/containerd
volumes:
- name: containerd
  flexVolume:
    driver: "virtlet/flexvolume_driver"
    options:
      type: qcow2
      capacity: 2048MB
```

©2025, Mirantis Inc.

Example of a pod definition for a Ceph RBD

If your virtual machines store data in Ceph, you can attach Ceph RBDs to the virtual machines under Kubernetes control by specifying the required RBDs in the virtual machine pod definition. You do not need to mount the devices in the container.

Virtlet supports the following options for Ceph RBD devices:

Option	Paramater
FlexVolume driver	kubernetes.io/flexvolume_driver
Type	ceph
Monitor	ip:port
User	user-name
Secret	user-secret-key
Volume	rbd-image-name
Pool	pool-name

The following text is an example of a virtual machine pod definition with one Ceph RBD volume:

```
apiVersion: v1
kind: Pod
metadata:
  name: cirros-vm-rbd
  annotations:
    kubernetes.io/target-runtime: virtlet
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: extraRuntime
            operator: In
            values:
          - virtlet
containers:
  - name: cirros-vm-rbd
    image: virtlet/image-service.kube-system/cirros
    volumeMounts:
    - name: test
      mountPath: /testvol
volumes:
  - name: test
    flexVolume:
      driver: kubernetes.io/flexvolume_driver
      options:
```

Type: ceph
Monitor: 10.192.0.1:6789
User: libvirt
Secret: AQDTwuVY8rA8HxAAthwOKaQPr0hRc7kCmR/9Qg==
Volume: rbd-test-image
Pool: libvirt-pool

Example of a pod definition for a block device

If you want to mount a block device that is available in the /dev/ directory on the Kubernetes node, you can specify the raw device in the pod definition.

The following text is an example of pod definition with one block device. In this example, the path to the raw device is /dev/loop0 which means that a disk is associated with the path on the Virtlet node.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-vm-pod
  annotations:
    kubernetes.io/target-runtime: virtlet
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
        - key: extraRuntime
          operator: In
          values:
          - virtlet
containers:
  - name: test-vm
    image: download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
    volumeMounts:
  - name: raw
    mountPath: /rawvol
volumes:
- name: raw
  flexVolume:
    driver: "virtlet/flexvolume_driver"
    options:
      type: raw
      path: /dev/loop0
```

Reprovision the Kubernetes Master node

> Caution!
>
> Kubernetes support termination notice
>
> Starting with the MCP 2019.2.5 update, the Kubernetes component is no longer supported as a part of the MCP product. This implies that Kubernetes is not tested and not shipped as an MCP component. Although the Kubernetes Salt formula is available in the community driven SaltStack formulas ecosystem, Mirantis takes no responsibility for its maintenance.
>
> Customers looking for a Kubernetes distribution and Kubernetes lifecycle management tools are encouraged to evaluate the Mirantis Kubernetes-as-a-Service (KaaS) and Docker Enterprise products.

If the Kubernetes Master node became non-operational and recovery is not possible, you can reprovision the node from scratch.

When reprovisioning a node, you can not update some of the configuration data:

- Hostname and FQDN - because it breaks Calico.
- Node role - for example, from Kubernetes Master to Node role. However, you can use the kubectl label node command to reset a node labels later.
- Network plugin - for example, from Calico to Weave.

You can change the following information:

- Host IP(s)
- MAC addresses
- Operating system
- Application certificates

> Caution!
>
> All Master nodes must serve the same apiserver certificate. Otherwise, service tokens will become invalidated.

To reprovision the Kubernetes Master node:

1. Verify that MAAS works properly and provides the DHCP service to assign an IP address and bootstrap an instance.

2. Verify that the target nodes have connectivity with the Salt Master node:

```
salt 'ctl[<NUM>]*' test.ping
```

3. Update modules and states on the new Minion of the Salt Master node:

```
salt 'ctl[<NUM>]*' saltutil.sync_all
```

> Note
>
> The ctl[<NUM>] parameter is the ID of a failed Kubernetes Master node.

4. Create and distribute SSL certificates for services using the salt state:

```
salt 'ctl[<NUM>]*' state.sls salt
```

5. Install Keepalived:

```
salt 'ctl[<NUM>]*' state.sls keepalived -b 1
```

6. Install HAProxy and verify its status:

```
salt 'ctl[<NUM>]*' state.sls haproxy
salt 'ctl[<NUM>]*' service.status haproxy
```

7. Install etcd and verify the cluster health:

```
salt 'ctl[<NUM>]*' state.sls etcd.server.service
salt 'ctl[<NUM>]*' cmd.run "etcdctl cluster-health"
```

   Install etcd with the SSL support:

```
salt 'ctl[<NUM>]*' state.sls salt.minion.cert,etcd.server.service
salt 'ctl[<NUM>]*' cmd.run '. /var/lib/etcd/configenv && etcdctl cluster-health'
```

8. Install Kubernetes:

```
salt 'ctl[<NUM>]*' state.sls kubernetes.master.kube-addons
salt 'ctl[<NUM>]*' state.sls kubernetes.pool
```

9. Set up NAT for Calico:

```
salt 'ctl[<NUM>]*' state.sls etcd.server.setup
```

10. Run master to check consistency:

```
salt 'ctl[<NUM>]*' state.sls kubernetes exclude=kubernetes.master.setup
```

11. Register add-ons:

```
salt 'ctl[<NUM>]*' --subset 1 state.sls kubernetes.master.setup
```

Kubernetes Nodes operations

> Caution!
>
> Kubernetes support termination notice
>
> Starting with the MCP 2019.2.5 update, the Kubernetes component is no longer supported as a part of the MCP product. This implies that Kubernetes is not tested and not shipped as an MCP component. Although the Kubernetes Salt formula is available in the community driven SaltStack formulas ecosystem, Mirantis takes no responsibility for its maintenance.
>
> Customers looking for a Kubernetes distribution and Kubernetes lifecycle management tools are encouraged to evaluate the Mirantis Kubernetes-as-a-Service (KaaS) and Docker Enterprise products.

This section contains the Kubernetes Nodes-related operations.

Add a Kubernetes Node automatically

MCP DriveTrain enables you to automatically scale up the number of Nodes in your MCP Kubernetes cluster if required.

> Note
>
> Currently, only scaling up is supported using the Jenkins pipeline job. Though, you can scale down the number of Kubernetes Nodes manually as described in Remove a Kubernetes Node.

To scale up a Kubernetes cluster:

1. Log in to the Jenkins web UI as an administrator.

   > Note
   >
   > To obtain the password for the admin user, run the salt "cid*" pillar.data _param:jenkins_admin_password command from the Salt Master node.

2. Find the deployment pipeline job that you used to successfully deploy your Kubernetes cluster. For deployment details, refer to the Deploy a Kubernetes cluster procedure. You will reuse the existing deployment pipeline job to scale up your existing Kubernetes cluster.
3. Select the Build with Parameters option from the drop-down menu of the pipeline job.
4. Reconfigure the following parameters:

   Kubernetes parameters to scale up the existing cluster

Parameter	Description
STACK_COMPUTE_COUNT	The number of Kubernetes Nodes to be deployed by the pipeline job. Configure as required for your use case.
STACK_NAME	The Heat stack name to reuse.
STACK_REUSE	Select to reuse the existing Kubernetes deployment that requires scaling up.

5. Click Build to launch the pipeline job.

As a result of the deployment pipeline job execution, your existing Kubernetes cluster will be scaled up during the Scale Kubernetes Nodes stage as configured. The preceding stages of the workflow will be executed as well to ensure proper configuration. However, it will take a significantly shorter period of time to execute these stages, as most of the operations have been already performed during the initial cluster deployment.

Add a Kubernetes Node manually

This section describes how to manually add a Kubernetes Node to your MCP cluster to increase the cluster capacity, for example.

To add a Kubernetes Node manually:

1. Add a physical node using MAAS as described in the MCP Deployment Guide: Provision physical nodes using MAAS.
2. Log in to the Salt Master node.
3. Verify that salt-minion is running on the target node and this node appears in the list of the Salt keys:

   ```
   salt-key
   ```

   Example of system response:

   ```
   cmp0.bud.mirantis.net
   cmp1.bud.mirantis.net
   cmp2.bud.mirantis.net
   ```

4. Apply the Salt states to the target node. For example, to cmp2:

   ```
   salt 'cmp2*' saltutil.refresh_pillar
   salt 'cmp2*' saltutil.sync_all
   salt 'cmp2*' state.apply salt
   salt 'cmp2*' state.apply linux,ntp,openssh,git
   salt 'cmp2*' state.sls kubernetes.pool
   salt 'cmp2*' service.restart 'kubelet'
   salt 'cmp2*' state.apply salt
   salt '*' state.apply linux.network.host
   ```

5. If Virtlet will run on the target node, add the node label:

   ```
   salt -C 'I@kubernetes:master and 01' \
   cmd.run 'kubectl label --overwrite node cmp2 extraRuntime=virtlet'
   ```

6. Log in to any Kubernetes Master node.
7. Verify that the target node appears in the list of the cluster nodes and is in the Ready state:

   ```
   kubectl get nodes
   ```

   Example of system response:

   ```
   NAME STATUS ROLES AGE  VERSION
   cmp0 Ready  node  54m  v1.10.3-3+93532daa6d674c
   ```

```
cmp1 Ready  node  54m  v1.10.3-3+93532daa6d674c
cmp2 Ready  node  2m   v1.10.3-3+93532daa6d674c
```

Remove a Kubernetes Node

This section describes how to remove a Kubernetes Node from your MCP cluster.

To remove a Kubernetes Node:

1. Log in to the Kubernetes Node that you want to remove.
2. Stop and disable the salt-minion service on this node:

   ```
   systemctl stop salt-minion
   systemctl disable salt-minion
   ```

3. Log in to the Salt Master node.
4. Verify that the node name is not registered in salt-key. If the node is present, remove it:

   ```
   salt-key | grep <node_name><NUM>
   salt-key -d <node_name><NUM>.domain_name
   ```

5. Log in to any Kubernetes Master node.
6. Mark the Node as unschedulable to prevent new pods from being assigned to it:

   ```
   kubectl cordon <node_ID>
   kubectl drain <node_ID>
   ```

7. Remove the Kubernetes Node:

   ```
   kubectl delete node cmp<node_ID>
   ```

   Wait until the workloads are gracefully deleted and the Kubernetes Node is removed.
8. Verify that the node is absent in the Kubernetes Nodes list:

   ```
   kubectl get nodes
   ```

9. Open your Git project repository with Reclass model on the cluster level.
10. In infra/config.yml, remove the definition of the Kubernetes Node in question under the reclass:storage:node pillar.
11. Log in to the Kubernetes Node in question.
12. Run the following commands:

    ```
    systemctl stop kubelet
    systemctl disable kubelet
    ```

Reprovision a Kubernetes Node

You may need to reprovision a failed Kubernetes Node. When reprovisioning a Kubernetes Node, you can not update some of the configuration data:

- Hostname and FQDN - because it breaks Calico.
- Node role - for example, from Kubernetes Master to Node role. However, you can use the kubectl label node command to reset a node labels later.

You can change the following information:

- Host IP(s)
- MAC addresses
- Operating system
- Application certificates

> **Caution!**
>
> All Master nodes must serve the same apiserver certificate. Otherwise, service tokens will become invalidated.

To reprovision a Kubernetes Node:

1. In the MAAS web UI, make the required changes to the target Kubernetes Node.
2. Verify that MAAS works properly and provides the DHCP service to assign IP addresses and bootstrap an instance.
3. Proceed with the Add a Kubernetes Node manually procedure starting the step 2.

Use the role-based access control (RBAC)

Caution!

Kubernetes support termination notice

Starting with the MCP 2019.2.5 update, the Kubernetes component is no longer supported as a part of the MCP product. This implies that Kubernetes is not tested and not shipped as an MCP component. Although the Kubernetes Salt formula is available in the community driven SaltStack formulas ecosystem, Mirantis takes no responsibility for its maintenance.

Customers looking for a Kubernetes distribution and Kubernetes lifecycle management tools are encouraged to evaluate the Mirantis Kubernetes-as-a-Service (KaaS) and Docker Enterprise products.

After you enable the role-based access control (RBAC) on your Kubernetes cluster as described in Deployment Guide: Enable RBAC, you can start controlling system access to authorized users by creating, changing, or restricting user or services roles as required. Use the kubernetes.control.role state to orchestrate the role and role binding.

The following example illustrates a configuration of a brand-new role and role binding for a service account:

```
control:
  role:
    etcd-operator:
      kind: ClusterRole
      rules:
        - apiGroups:
            - etcd.coreos.com
          resources:
            - clusters
          verbs:
            - "*"
        - apiGroups:
            - extensions
          resources:
            - thirdpartyresources
          verbs:
            - create
        - apiGroups:
            - storage.k8s.io
          resources:
            - storageclasses
          verbs:
```

```
        - create
      - apiGroups:
        - ""
        resources:
        - replicasets
        verbs:
        - "*"
    binding:
      etcd-operator:
        kind: ClusterRoleBinding
        namespace: test # <-- if no namespace, then it is ClusterRoleBinding
        subject:
          etcd-operator:
            kind: ServiceAccount
```

The following example illustrates a configuration of the test edit permissions for a User in the test namespace:

```
kubernetes:
  control:
    role:
      edit:
        kind: ClusterRole
        # No rules defined, so only binding will be created assuming role
        # already exists.
        binding:
          test:
            namespace: test
            subject:
              test:
                kind: User
```

OpenContrail operations

This section describes how to configure and use the OpeContrail-related features. To troubleshoot the OpenContrail services, refer to Troubleshoot OpenContrail.

Verify the OpenContrail status

To ensure that OpenContrail is up and running, verify the status of all the OpenContrail-related services. If any service fails, restart it as described in Restart the OpenContrail services.

For OpenContrail 4.x

1. Log in to the Salt Master node.
2. Apply one of the following states depending on the Build ID of your MCP cluster:

    * For MCP Build ID 2018.11.0 or later:

      ```
      salt -C 'ntw* or nal*' state.sls opencontrail.upgrade.verify
      ```

      If the state is applied successfully, it means that all OpenContrail services are up and running.
    * For MCP Build ID 2018.8.0 or 2018.8.0-milestone1:

      ```
      salt -C 'ntw* or nal*' cmd.run 'doctrail all contrail-status'
      ```

      In the output, all services must be either in active or backup (for example, for contrail-schema, contrail-svc-monitor, contrail-device-manager services) state.

For OpenContrail 3.2

Eventually, all services should be active except for contrail-device-manager, contrail-schema, and contrail-svc-monitor. These services are in the active state at only one OpenContrail controller ntw node in the cluster switching dynamically between the nodes in case of a failure. On two other OpenContrail controller nodes, these services are in the backup state.

To verify the OpenContrail services status, apply the following state for the OpenContrail ntw and nal nodes from the Salt Master node:

```
salt -C 'ntw* or nal*' cmd.run 'contrail-status'
```

Example of system response:

```
== Contrail Control ==
supervisor-control:           active
contrail-control              active
contrail-control-nodemgr      active
contrail-dns                  active
contrail-named                active

== Contrail Analytics ==
supervisor-analytics:         active
contrail-analytics-api        active
contrail-analytics-nodemgr    active
```

```
contrail-collector           active
contrail-query-engine        active
contrail-snmp-collector      active
contrail-topology            active

== Contrail Config ==
supervisor-config:           active
contrail-api:0               active
contrail-config-nodemgr      active
contrail-device-manager      initializing
contrail-discovery:0         active
contrail-schema              initializing
contrail-svc-monitor         initializing
ifmap                        active

== Contrail Web UI ==
supervisor-webui:            active
contrail-webui               active
contrail-webui-middleware    active

== Contrail Database ==
supervisor-database:         active
contrail-database            active
contrail-database-nodemgr    active

== Contrail Support Services ==
supervisor-support-service:  active
rabbitmq-server              active
```

Restart the OpenContrail services

You may need to restart an OpenContrail service, for example, during an MCP cluster update or upgrade when a service failure is caused by the asynchronous restart order of the OpenContrail services after the kvm nodes update or reboot.

All OpenContrail 4.x services run as the systemd services in a Docker container.

All OpenContrail 3.2 services are managed by the process supervisord. The supervisord daemon is automatically installed with the OpenContrail packages including the following OpenContrail Supervisor groups of services:

- supervisor-database
- supervisor-config
- supervisor-analytics
- supervisor-control
- supervisor-webui

To restart the OpenContrail 4.x services:

1. Log in to the Salt Master node.
2. Restart the required service on the corresponding OpenContrail node using the following example:

   ```
   salt 'ntw03' cmd.run 'doctrail controller service contrail-api restart'
   ```

   > Note
   >
   > For a list of OpenContrail containers names to be used by the doctrail utility, see: The doctrail utility for the OpenContrail containers in OpenStack.

3. If restarting of a service in question does not change its failed status, proceed to further troubleshooting as described in Troubleshoot OpenContrail. For example, to troubleshoot Cassandra not starting, refer to Troubleshoot Cassandra for OpenContrail 4.x.

To restart the OpenContrail 3.2 services:

1. Log in to the required OpenContrail node.
2. Select from the following options:

   - To restart the OpenContrail services group as a whole Supervisor, use the service <supervisor_group_name> restart command. For example:

     ```
     service supervisor-control restart
     ```

- To restart individual services inside the Supervisor group, use the service <supervisor_group_service_name> restart command. For example:

  ```
  service contrail-config-nodemgr restart
  ```

To identify the services names inside a specific OpenContrail Supervisor group, use the supervisorctl -s unix:///tmp/supervisord_<group_name>.sock status command. For example:

```
supervisorctl -s unix:///tmp/supervisord_database.sock status
```

Example of system response:

```
contrail-database                RUNNING    pid 1349, uptime 2 days, 21:12:33
contrail-database-nodemgr        RUNNING    pid 1347, uptime 2 days, 21:12:33

supervisorctl -s unix:///tmp/supervisord_config.sock status

contrail-api:0                   RUNNING    pid 49848, uptime 2 days, 20:11:54
contrail-config-nodemgr          RUNNING    pid 49845, uptime 2 days, 20:11:54
contrail-device-manager          RUNNING    pid 49849, uptime 2 days, 20:11:54
contrail-discovery:0             RUNNING    pid 49847, uptime 2 days, 20:11:54
contrail-schema                  RUNNING    pid 49850, uptime 2 days, 20:11:54
contrail-svc-monitor             RUNNING    pid 49851, uptime 2 days, 20:11:54
ifmap                            RUNNING    pid 49846, uptime 2 days, 20:11:54

supervisorctl -s unix:///tmp/supervisord_analytics.sock status

contrail-analytics-api           RUNNING    pid 1346, uptime 2 days, 21:13:17
contrail-analytics-nodemgr       RUNNING    pid 1340, uptime 2 days, 21:13:17
contrail-collector               RUNNING    pid 1344, uptime 2 days, 21:13:17
contrail-query-engine            RUNNING    pid 1345, uptime 2 days, 21:13:17
contrail-snmp-collector          RUNNING    pid 1341, uptime 2 days, 21:13:17
contrail-topology                RUNNING    pid 1343, uptime 2 days, 21:13:17

supervisorctl -s unix:///tmp/supervisord_control.sock status

contrail-control                 RUNNING    pid 1330, uptime 2 days, 21:13:29
contrail-control-nodemgr         RUNNING    pid 1328, uptime 2 days, 21:13:29
contrail-dns                     RUNNING    pid 1331, uptime 2 days, 21:13:29
contrail-named                   RUNNING    pid 1333, uptime 2 days, 21:13:29

supervisorctl -s unix:///tmp/supervisord_webui.sock status

contrail-webui                   RUNNING    pid 1339, uptime 2 days, 21:13:44
contrail-webui-middleware        RUNNING    pid 1342, uptime 2 days, 21:13:44
```

Access the OpenContrail web UI

Your OpenContrail cluster may not use SSH overall because of not having a certificate authority available. By default, OpenContrail uses SSL and requires certificate authentication. If you attempt to access the OpenContrail UI through the proxy with such configuration, the UI will accept your credentials but will end up in logging you out immediately. As a workaround, you can use HTTP directly to the OpenContrail web UI management VIP bypassing the proxy.

To access the OpenContrail web UI:

1. Obtain the Administrator credentials. Select from the following options depending on your cluster type:

   - For OpenStack:

     1. Log in to the Salt Master node.
     2. Apply the following state:

        ```
        salt 'ctl01*' cmd.run 'cat /root/keystonerc'
        ```

     3. From the output of the command above, record the values of OS_USERNAME and OS_PASSWORD.

   - For Kubernetes:

     1. Log in to any OpenContrail controller node.
     2. Run the following command:

        ```
        cat /etc/contrail/contrail-webui-userauth.js | grep "auth.admin"'
        ```

     3. From the output of the command above, record the values of auth.admin_user and auth.admin_password.

2. In a browser, type either the OpenStack controller node VIP or the Kubernetes controller node VIP on port 8143. For example, https://172.31.110.30:8143.

3. On the page that opens, configure your web browser to trust the certificate if you have not done so yet:

   - In Google Chrome or Chromium, click Advanced > Proceed to <URL> (unsafe).
   - In Mozilla Firefox, navigate to Advanced > Add Exception, enter the URL in the Location field, and click Confirm Security Exception.

   > Note
   >
   > For other web browsers, the steps may vary slightly.

4. Enter the Administrator credentials obtained in the step 1. Leave the Domain field empty unless the default configuration was customized.

5. Click Sign in.

Configure route targets for external access

Configuring the OpenContrail route targets for your Juniper MX routers allows extending the private network outside the MCP cloud.

To configure route targets for external access:

1. Log in to the OpenContrail web UI as described in Access the OpenContrail web UI.
2. Navigate to Configure > Networking > Networks.
3. Click the gear icon of the network that you choose to be external and select Edit.
4. In the Edit window:
    1. Expand Advanced Options.
    2. Select the Shared and External check boxes.
    3. Expand Route Target(s).
    4. Click the + symbol to add ASN and Target.
    5. Enter the corresponding numbers set during provisioning of the Juniper MX router that is used in your MCP cluster.
    6. Click Save.
5. Verify the route targets configuration:
    1. Navigate to Configure > Infrastructure > BGP Routers.
    2. Expand one of the BGP Router or Control Node nodes menu.
    3. Verify that Autonomous System fits ASN set in the previous steps.

Enable Long Lived Graceful Restart in OpenContrail

> Warning
>
> Enabling LLGR causes restart of the Border Gateway Protocol (BGP) peerings.

Enabling of Long Lived Graceful Restart (LLGR) must be performed on both sides of peering - edge gateways and the OpenContrail control plane.

To enable LLGR:

1. Log in to the MX Series router CLI.
2. Add the following lines to the router configuration file:

```
set protocols bgp group <name> family inet-vpn unicast graceful-restart long-lived restarter stale-time 20
set protocols bgp group <name> graceful-restart restart-time 1800
set protocols bgp group <name> graceful-restart stale-routes-time 1800
```

3. Commit the configuration changes to the router.
4. Open your GitHub MCP project repository.
5. Add the following lines to cluster/<name>/opencontrail/control.yml:

```
classes:
...
- system.opencontrail.client.resource.llgr
...
```

6. Commit and push the changes to the project Git repository.
7. Log in to the Salt Master node.
8. Pull the latest changes of the cluster model and the system model that has the system.opencontrail.client.resource.llgr class defined.
9. Update the salt-formula-opencontrail package.
10. Apply the opencontrail state:

```
salt -C 'I@opencontrail:config and *01*' state.sls opencontrail.client
```

©2025, Mirantis Inc.

Use the OpenContrail API client

The contrail-api-cli command-line utility interacts with the OpenContrail API server that allows searching for or modifying API resources as well as supports the unix-style commands. For more information, see the Official contrail-api-cli documentation.

This section contains the following topics:

- Install the OpenContrail API client
- Access the OpenContrail API client
- The contrail-api-cli-extra package

Install the OpenContrail API client

To install contrail-api-cli:

1. Log in to any OpenContrail controller node. For example, ntw01.
2. Install the Python virtual environment for contrail-api-cli:

```
apt-get install python-pip python-dev -y &&\
pip install virtualenv && \
virtualenv contrail-api-cli-venv && \
source contrail-api-cli-venv/bin/activate && \
git clone https://github.com/eonpatapon/contrail-api-cli/ && \
cd contrail-api-cli;sudo python setup.py install
```

Access the OpenContrail API client

To access the OpenContrail API:

1. Use the keystonerc file with credentials and endpoints:

```
source /root/keystonerc
source /root/keystonercv3
```

2. Connect to the OpenContrail API using the following command:

```
contrail-api-cli --host 10.167.4.20 --port 9100 shell
```

   Or you can use your OpenStack credentials. For example:

```
contrail-api-cli --os-user-name admin --os-password workshop  \
--os-auth-plugin v2password --host 10.10.10.254 --port 8082 --protocol http \
--insecure --os-auth-url http://10.10.10.254:5000/v2.0 --os-tenant-name admin shell
```

Note

- MCP uses the 9100 port by default, whereas the OpenContrail API standard port is 8082.
- For the ln command, define a schema in the --schema-version 3.1 parameter. The known versions are the following: 2.21, 3.2, 3.0, 1.10, 3.1.

The contrail-api-cli-extra package

The contrail-api-cli-extra package contains the contrail-api-cli commands to make the OpenContrail installation and operation process easier.

The commands are grouped in different sub-packages and have different purposes:

- clean: detect and remove bad resources
- fix: detect and fix bad resources
- migration: handle data migration when upgrading OpenContrail to a major version
- misc: general-purpose commands
- provision: provision and configure an OpenContrail installation

To install the contrail-api-cli-extra package:

Run the following command:

```
pip install contrail-api-cli-extra
```

The most used contrail-api-cli-extra sub-packages are the following:

- contrail-api-cli_extra.clean - since the sub-package allows removing resources, you must explicitly load the contrail_api_cli.clean namespace to run the commands of this sub-package.

  Example of usage:

  ```
  contrail-api-cli --ns contrail_api_cli.clean <command>
  # usage
  contrail-api-cli --host 10.167.4.21 --port 9100 --ns contrail_api_cli.clean shell
  ```

  This package includes the clean-<type> command. Replace type with the required type of cleaning process. For example:

  - clean-route-target
  - clean-orphaned-acl
  - clean-si-scheduling

    - clean-stale-si

- contrail_api_cli_extra.fix - allows you to verify and fix misconfigured resources. For example, fix multiple security groups or association of a subnet with a virtual network (VN) in a key-value store.

  If this sub-package is installed, it launches with contrail-api-cli automatically.

  Example of usage:

  ```
  fix-vn-id virtual-network/600ad108-fdce-4056-af27-f07f9faa5cae --zk-server 10.167.4.21

  fix-zk-ip --dry-run --zk-server 10.167.4.21:2181 vitrual-network/xxxxxx-xxxxxxx-xxxxxx
  ```

Seealso

Official contrail-api-cli-extra documentation

Define aging time for flow records

> Note
>
> This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

To prevent high memory consumption by vRouter on highly loaded clusters, you can define the required aging time for flow records. Flows are aged depending on the inactivity for a specific period of time. By default, the timeout value is 180 seconds. You can configure the timeout depending on your cluster load needs using the flow_cache_timeout parameter for the contrail-vrouter-agent service.

To configure flow_cache_timeout:

1. Log in to the Salt Master node.
2. In classes/cluster/<cluster_name>/opencontrail/compute.yml of your Reclass model, define the required value in seconds for the flow_cache_timeout parameter:

   ```
   parameters:

     opencontrail:
       ...
       compute
         ...
         flow_cache_timeout: 180
         ...
   ```

3. Apply the changes:

   ```
   salt -C 'I@opencontrail:compute' state.apply opencontrail.compute
   ```

OpenContrail 4.x-specific operations

This section contains the OpenContrail operations applicable to version 4.x.

Modify the OpenContrail configuration

The OpenContrail v4.x configuration files are generated by SaltStack and then mounted into containers as volumes. SaltStack also provides a mechanism to apply configuration changes by restarting the systemd services inside a container.

To modify the OpenContrail v4.x configurations:

1. Log in to the Salt Master node.
2. Make necessary configuration changes in the classes/cluster/<cluster_name>/opencontrail folder.
3. Apply the opencontrail state specific to the OpenContrail configuration file where the changes were made. For example, if you made changes in the database.yaml file:

   ```
   salt -C 'I@opencontrail:database' state.apply opencontrail
   ```

   After the state is applied, the systemd services are automatically restarted inside the OpenContrail container in question to apply the configuration changes.

The doctrail utility for the OpenContrail containers in OpenStack

In an OpenStack-based MCP environment, the OpenContrail installation includes the doctrail utility that provides an easy access to the OpenContrail containers. Its default installation folder is /usr/bin/doctrail.

The doctrail usage is as follows:

```
doctrail {analytics|analyticsdb|controller|all} {<command_to_send>|console}
```

The acceptable destinations used by doctrail are as follows:

- controller for the OpenConrail controller container (console, commands)
- analytics for the OpenConrail analytics container (console, commands)
- analyticsdb for the OpenConrail database of the analytics container (console, commands)
- all for all containers on the host (commands only)

The doctrail commands examples:

```
# Show contrail-status on all containers on this host
doctrail all contrail-status

# Show contrail-status on controller container
doctrail controller contrail-status

# Restart contrail-database on controller container
doctrail controller service contrail-database restart

# Connect to the console of controller container
doctrail controller console

# Connect to the console of analytics container
doctrail analytics console
```

Set multiple contrail-api workers

In the MCP Build ID 2019.2.0, by default, one worker of the contrail-api service is used. Starting from the MCP 2019.2.3 maintenance update, six workers are used by default.

If needed, you can change the default configuration using the instruction below. This section also describes how to stop, start, or restart multiple workers.

To set multiple contrail-api workers for the MCP version 2019.2.3 or later:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In cluster/<cluster name>/opencontrail/control.yml, set the required amount of workers:

   ```
   parameters:
     _param:
       opencontrail_api_workers_count: <required_amount_of_workers>
   ```

3. Log in to the Salt Master node.
4. Refresh pillars:

   ```
   salt '*' saltutil.refresh_pillar
   salt-call state.sls reclass.storage
   ```

5. Apply the Reclass model changes:

   ```
   salt -C 'I@opencontrail:control' state.apply opencontrail
   ```

To set multiple contrail-api workers for the Build ID 2019.2.0:

> **Caution!**
>
> Using the configuration below, you can start setting network entities in a newly created OpenStack project only one minute after this project is created.

1. Open your Git project repository with the Reclass model on the cluster level.
2. In cluster/<cluster name>/opencontrail/control.yml, set the required amount of workers:

   ```
   parameters:
     _param:
       opencontrail_api_workers_count: <required_amount_of_workers>
   ```

3. Log in to the Salt Master node.
4. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
salt-call state.sls reclass.storage
```

5. Apply the Reclass model changes:

   ```
   salt -C 'I@opencontrail:control' state.apply opencontrail
   ```

6. Log in to any ntw node.
7. In /etc/contrail/contrail-api.conf, change the following parameter:

   ```
   [KEYSTONE]
   keystone_sync_on_demand=true
   ```

   to

   ```
   [KEYSTONE]
   keystone_sync_on_demand=false
   ```

8. Restart the OpenContrail controller Docker container:

   ```
   cd /etc/docker/compose/opencontrail/; docker-compose down; docker-compose up -d
   ```

9. Wait until all OpenContrail controller services are up and running. To verify the OpenContrail services status, refer to Verify the OpenContrail status.
10. Repeat the steps 7-9 on the remaining ntw nodes.

To stop, start, or restart multiple workers:

> **Caution!**
>
> We recommend that you do not stop, start, or restart the contrail-api workers by executing the service command as it may cause an unstable workers behavior such as an incorrect number of running workers and race conditions.

* To stop all contrail-api workers on the target node:

  ```
  systemctl stop contrail-api@*
  ```

* To start all contrail-api workers on the target node:

```
systemctl start contrail-api@*
```

- To restart all contrail-api workers on the target node:

```
systemctl restart contrail-api@*
```

Note

You may stop, start, or restart a certain worker by using the worker ID instead of the * character.

Seealso

MCP Reference Architecture: OpenContrail limitations

Enable SSL for an OpenContrail API internal endpoint

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

This section describes how to manually enable SSL for an internal endpoint of the OpenContrail API.

To enable SSL for an OpenContrail API internal endpoint:

1. Open your Git project repository with Reclass model on the cluster level.
2. In cluster/<cluster_name>/opencontrail/init.yml, add the following parameters:

   ```
   parameters:
     _param:
   ...
       opencontrail_api_ssl_enabled: True
       opencontrail_api_protocol: https
   ...
   ```

3. In cluster/<cluster_name>/opencontrail/analytics.yml, cluster/<cluster_name>/opencontrail/control.yml, and cluster/<cluster_name>/opencontrail/compute.yml, add the following class:

   ```
   classes:
   ...
     - system.salt.minion.cert.opencontrail.api
   ...
   ```

4. Log in to the Salt Master node.
5. Create certificates on the OpenContrail nodes:

   ```
   salt -C "I@opencontrail:database or I@opencontrail:compute" state.sls salt.minion.cert
   ```

6. Configure the HAProxy services:

   ```
   salt -C "I@opencontrail:control" state.apply haproxy
   ```

7. Configure the OpenContrail services:

   ```
   salt -C "I@opencontrail:database" state.apply opencontrail exclude=opencontrail.compute
   salt -C "I@opencontrail:compute" state.apply opencontrail.client
   ```

8. Configure the nodes that run neutron-server:

```
salt -C "I@neutron:server" state.apply neutron.server
```

9. Restart the neutron-server service to use SSL to connect to contrail-api VIP:

```
salt -C "I@neutron:server" service.restart neutron-server
```

Enable or disable VNC API statistics

> Note
>
> This feature is available starting from the MCP 2019.2.12 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

On every HTTP resource request, such as create, update, list, and so on, the OpenContrail contrail-api server sends statistics messages to the collector service, which can potentially cause high load on analytics nodes. To disable such statistics messages, use the disable_vnc_api_stats option of the contrail-api service. By default disable_vnc_api_stats is set to True.

To enable or disable VNC API statistics:

1. Log in to the Salt Master node.
2. In classes/cluster/<cluster_name>/opencontrail/control.yml of your Reclass model, specify the required value for the disable_vnc_api_stats parameter. To enable sending statistics messages to the OpenContrail analytics nodes, set disable_vnc_api_stats to False. To suppress the statistics messages, set disable_vnc_api_stats to True. For example:

```
parameters:

  opencontrail:
    ...
    config
      ...
      disable_vnc_api_stats: True
      ...
```

3. Apply the changes:

```
salt -C 'I@opencontrail:control' state.sls_id /etc/contrail/contrail-api.conf opencontrail
```

4. Restart the contrail-api service:

> Warning
>
> Restarting the contrail-api service on all nodes at once causes interruption for CRUD operations with networking objects. Plan the restart in advance or restart contrail-api on each node one by one using a modified target in the following Salt command.

   - If one worker is configured for the OpenContrail API:

```
salt -C 'I@opencontrail:control' cmd.run "doctrail controller service contrail-api* restart"
```

- If multiple workers are configured for the OpenContrail API:

```
salt -C 'I@opencontrail:control' cmd.run "doctrail controller systemctl restart contrail-api@*"
```

Seealso

Enable or disable VMI and VN statistics collection

Enable or disable VMI and VN statistics collection

> Note
>
> This feature is available starting from the MCP 2019.2.12 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

You can enable or disable gathering of statistics, related to virtual machine interfaces (VMIs) and virtual networks (VNs), by the OpenContrail vRouter agent service and sending it to OpenContrail analytics nodes. Such statistics messages can cause heavy load of OpenContrail analytics nodes on large deployments with huge amount of workloads. By default, disable_stats_collection is set to True.

To enable or disable VMI and VN statistics collection:

1. Log in to the Salt Master node.
2. In classes/cluster/<cluster_name>/opencontrail/compute.yml of your Reclass model, specify the required value for the disable_stats_collection parameter. For example:

   ```
   parameters:

     opencontrail:
       ...
       compute
         ...
         disable_stats_collection: True
         ...
   ```

3. Apply the changes:

   ```
   salt -C 'I@opencontrail:compute' state.sls_id /etc/contrail/contrail-vrouter-agent.conf opencontrail
   ```

4. Restart the contrail-vrouter-agent service:

   > Warning
   >
   > Restarting the contrail-vrouter-agent service causes networking service interruption for workloads on the affected node(s).

   ```
   salt -C 'I@opencontrail:compute' service.restart contrail-vrouter-agent
   ```

©2025, Mirantis Inc.

Seealso

Enable or disable VNC API statistics

DevOps Portal

Warning

The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

MCP’s Operations Support System (OSS), known as StackLight, now includes DevOps Portal connected to OSS services. DevOps Portal significantly reduces the complexity of Day 2 cloud operations through services and dashboards with a high degree of automation, availability statistics, resource utilization, capacity utilization, continuous testing, logs, metrics, and notifications. DevOps Portal enables cloud operators to manage larger clouds with greater uptime without requiring large teams of experienced engineers and developers.

This solution builds on MCP operations-centric vision of delivering a cloud environment through a CI/CD pipeline with continuous monitoring and visibility into the platform.

The portal collects a comprehensive set of data about the cloud, offers visualization dashboards, and enables the cloud operator to interact with a variety of tools. More specifically, the DevOps Portal includes the following dashboards:

Push Notification

Warning

The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Push Notification service enables users to send notifications, execute API calls, and open tickets on helpdesk. The service can also connect several systems through a notification queue that transforms and routes API calls from source systems into specific protocols that other systems use to communicate. With the notification system, you can send an API call and transform it into another API call, an email, an AMQP message, or another protocol.

Note

The Push Notification service depends on the following services:

- DevOps Portal web UI
- Elasticsearch cluster (version 5.x.x or higher)
- PostgreSQL database

To view and search for generated notifications, navigate to the Notifications dashboard available in the DevOps Portal UI.

Cloud Health

Warning

The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Cloud Health dashboard is the UI for the Cloud Health Service.

The Cloud Health service collects availability results for all cloud services and failed customer (tenant) interactions (FCI) for a subset of those services. These metrics are displayed so that operators can see both point-in-time health status and trends over time.

Note

The Cloud Health service depends on the following services:

- DevOps Portal web UI
- Grafana service of StackLight LMA

To view the metrics:

1. Log in to the DevOps Portal.
2. Navigate to the Cloud health dashboard.
3. View the metrics on tabs depending on your needs:
    - The Availability tab for the availability results for all cloud services
    - The FCI tab for FCI for a subset of cloud services

Cloud Intelligence

Warning

The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Cloud Intelligence service collects and stores data from MCP services, including OpenStack, Kubernetes, bare metal, and others. The data can be queried to enable use cases such as cost visibility, business insights, cost comparison, chargeback/showback, cloud efficiency optimization, and IT benchmarking. Operators can interact with the resource data using a wide range of queries, for example, searching for the last VM rebooted, total memory consumed by the cloud, number of containers that are operational, and so on.

Note

The Cloud Intelligence service depends on the following services:

- DevOps Portal web UI
- Elasticsearch cluster (version 5.x.x or higher)
- Runbook Automation

To start creating queries that will be submitted to the search engine and display the list of resources:

1. Log in to the DevOps Portal.
2. Navigate to the Cloud Intelligence dashboard.
3. Create your query using the cloud intelligence search query syntax:
    - To search by groups, use:
        - type=vm for instances
        - type=image for images
        - type=flavor for flavors
        - type=host for hosts
        - type=availabilityZone for availability zones
        - type=network for networks
        - type=volume for volumes
        - type=stack for Heat stacks

    - type=tenant for tenants
    - type=user for users
  - To search by field names, specify the field name with the value it contains. For example:

    Note

    The search by query string is case-insensitive.

    - status=active displays all resources with Active in the Status field, meaning they are in active status
    - status=(active saving) displays all resources in Active or Saving statuses
    - name="test_name" displays all resources which Name fields contain the exact phrase test_name
  - To group a number of queries in a single one, use the following boolean search operators:

Type search

Operator	Meaning	Usage example
|	The pipe symbol stands for OR operation	minDisk=0 | minRam=0
+	The plus symbol stands for AND operation	minDisk=0 + minRam=0
-	The minus symbol negates a single token	minRam=0 + -minDisk=40 searches for resource with minRam equal to 0 and minDisk not equal to 40 at the same time
( )	Parentheses signify grouping and precedence	(minDisk=0 minRam=0) + minDisk=40

  - To search for the reserved characters, escape them with \. The whole list of these characters includes + - = & | > < ! ( ) { } [ ] ^ " ~ * ? : \ /.

4. View search results:

  - An item name and most important properties are visible by default.
  - To view the full item properties list, click on the item block.

> Note
>
> If you have the Cleanup service enabled, the Create Janitor rule button is available for the groups that Janitor supports like VMs, Images, and Tenants. The button provides the same functionality as the Create new rule button on the Janitor dashboard with the conditions list prefilled with item-specific properties.

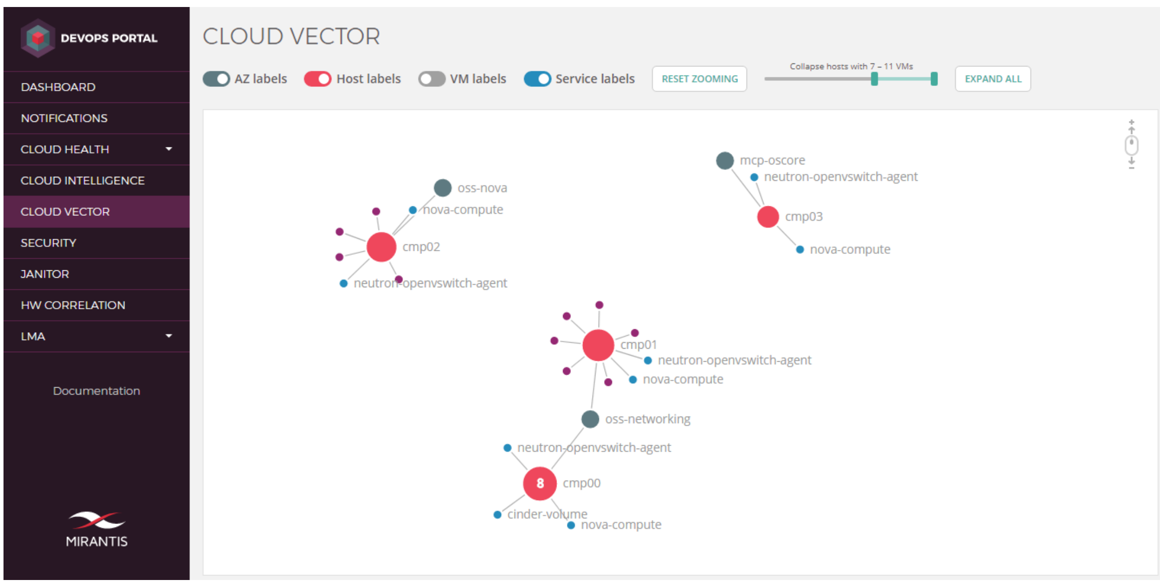
5. Export search results into JSON, YAML, and CSV formats using the corresponding buttons on the Cloud Intelligence dashboard. The exported data contains the original query and the resulting groups with their items.

Cloud Vector

> Warning
>
> The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Cloud Vector dashboard uses a node graph to represent a cloud environment in a form of a cloud map. The entities that build the map include availability zones (AZs), hosts, VMs, and services. Each host represents a compute node in a particular AZ with all VMs and services running on it. Thereby, a cloud map enables you to easily identify the number of nodes running in your cloud environment.

The screen capture below is an example of a cloud map created by Cloud Vector.



> Note
>
> The Cloud Vector dashboard depends on the following services:
>
> - DevOps Portal web UI
> - Cloud Intelligence service

To use the Cloud Vector dashboard:

1. Log in to the DevOps Portal.

2. Navigate to the Cloud Vector dashboard.
3. Proceed with the following available actions as required:

    - Collapse child elements:

        > Note
        >
        > Hosts with more than 50 child VMs are collapsed by default.

        > Note
        >
        > The size of a host circle depends on the number of its child elements. The more VMs a host owns, the bigger it is.

        - Double-click on an AZ or a host to collapse its child elements. If a host is collapsed, the number of its VMs is displayed. Services are not collapsed when you collapse a host.
        - Use the slider to collapse the nodes which VMs count matches the specified conditions.
    - Expand child elements:
        - Double-click on a collapsed element to expand its child elements.
        - Click Expand all to expand all collapsed elements.
    - Drag elements on the canvas:
        - Drag a particular element to move it and all connected elements.
        - Drag the canvas background to change the position of all elements.
        - Click Reset zomming to reset canvas shifts.
    - Scale elements on the canvas:

        > Note
        >
        > Red borders appear if elements are extended beyond the canvas boundaries.

        - Click on the canvas and scroll up or down to zoom in or out.
        - Click Reset zomming to reset scaling.
    - Show and hide node labels:

- Use toggles to show or hide labels of particular entities.
- Hover over a particular element to view its label.

Runbooks

Warning

The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Runbooks Automation service enables operators to create a workflow of jobs that get executed at specific time intervals or in response to specific events. For example, operators can automate periodic backups, weekly report creations, specific actions in response to failed Cinder volumes, and so on.

Note

The Runbooks Automation service is not a lifecycle management tool, appropriate for reconfiguring, scaling, or updating MCP itself as these operations are exclusively performed with DriveTrain.

Note

The Runbooks Automation service depends on the following services:

- DevOps Portal web UI
- PostgreSQL database

Using the Runbooks dashboard, operators can call preconfigured jobs or jobs workflows and track the execution status through the web UI.

Before you proceed with the dashboard, you may need to configure your own jobs or reconfigure the already existing ones to adjust them to special needs of your installation.

Configure Rundeck jobs

Rundeck enables you to easily add jobs to the Runbook Automation service as Rundeck jobs and chain them into workflows. Once the jobs are created and added to your Reclass model, you execute them through the DevOps Portal UI.

Create users

To configure Users in the Rundeck service:

1. Use the following structure to configure users through pillar parameters:

```
parameters:
  rundeck:
    server:
      users:
        user1:
          name: USER_NAME_1
          password: USER_PWD_1
          roles:
            - user
            - admin
            - architect
            - deploy
            - build
        user2:
          name: USER_NAME_2
          password:  USER_PWD_2
          roles:
            - user
            - deploy
            - build
        ...
```

> Note
>
> Currently, the default access control list (ACL) properly supports only admin users. Therefore, the user2 user in the configuration structure above will not be able to run jobs and view projects.

2. Create API tokens for non-interactive communications with Rundeck API. To configrue tokens, specify the required parameters in metadata. For example:

```
parameters:
  rundeck:
    server:
      tokens:
        admin: token0
        User2: token4
```

3. Apply the rundeck.server state:

```
salt-call state.sls rundeck.server
```

4. Restart the Rundeck service:

```
docker service update --force rundeck_rundeck
```

Create projects

To create Projects in the Rundeck service:

1. Use the following structure to configure projects through pillar parameters:

```
parameters:
  rundeck:
    client:
      project:
        test_project:
          description: PROJECT_DECRIPTION
          node:
            node01:
              nodename: NODE_NAME
              hostname: HOST_NAME
              username: USER_NAME
              tags: TAGS
```

   For example:

```
parameters:
  rundeck:
    client:
      project:
        test_project:
          description: "Test project"
          node:
            node01:
              nodename:node-1
              hostname: 10.20.0.1
              username: runbook
              tags: [cicd, docker]
            node02:
              nodename: node-2
              hostname: 10.20.0.2
              username: runbook
              tags: [cicd, docker]
            node03:
              Nodename: node-3
              hostname: 10.20.0.3
              username: runbook
              tags: [cicd, docker]
```

   All configured nodes in a particular project are available to run jobs and commands within this project. Also, nodes can be tagged that allows for filtering when executing commands and jobs on nodes.

2. The Rundeck metadata has a preconfigured user to access other nodes. The user is called runbook. You need to configure it on nodes before you can use jobs or commands in projects. To configure the runbook user, inherit classes of your nodes from the following class specifying the rundeck_runbook_public_key and rundeck_runbook_private_key parameters:

   ```
   classes:
     - system.rundeck.client.runbook
   ```

3. Apply the linux and openssh states:

   ```
   salt '*' state.sls linux.system.user
   salt '*' state.sls openssh.server
   ```

Configure jobs importing

You can configure a Git repository for each project and store Rundeck jobs in the repository. The following extract is an example of a Rundeck job that you can define in the Git repository for importing:

```
- description: Shows uptime of the system.
  executionEnabled: true
  group: systools
  loglevel: INFO
  name: uptime
  nodeFilterEditable: false
  nodefilters:
    dispatch:
      excludePrecedence: true
      keepgoing: true
      rankOrder: ascending
      threadcount: 1
    filter: tags:cicd
  nodesSelectedByDefault: true
  options:
  - enforced: true
    name: pretty
    values:
    - -p
  scheduleEnabled: true
  sequence:
    commands:
    - exec: uptime ${option.pretty}
    keepgoing: false
    pluginConfig:
      WorkflowStrategy:
        node-first: null
    strategy: node-first
```

This approach has the following limitations:

- Changes introduced using the git commit --amend command are not supported.
- The name parameter in job definition files is required. The value of the name parameter may differ from the file name and determines the resulting name of a job.
- You can configure not more than one remote repository per project
- An incorrect or non-existent branch definition may not result in a Salt configuration error leading to an empty job list.
- The Salt state may not recover jobs if you have specified the branch incorrectly. For example, if the jobs are lost due to the incorrect branch definition, the synchronization of jobs may be lost even if the correct branch is defined later and the Salt state is restarted.

To configure job importing for a project:

1. To use a remote repository as a source of jobs, extend the project’s metadata as required. A minimal configuration includes the address parameter for the import plugin:

```
parameters:
  rundeck:
    client:
      project:
        test_project:
          plugin:
            import:
              address: https://github.com/path/to/repo.git
```

A complete list of all available parameters for the import plugin includes:

Import plugin parameters

Parameter and default value	Values	Description
address: https://github.com/path/to/repo.git	String	A valid Git URL (Required)
branch: master	String	The name of a repository branch (Optional)
import_uuid_behavior: remove	String preserve, remove, or archive	The UUID importing mode in job descriptions (Optional)
format: yaml	String yaml or xml	The extension of files containing job definitions (Optional)
path_template: ${job.group}${job.name}.${config.format}	String	The pattern to recognize job definition files (Optional)
file_pattern: '.*.yaml'	Regex	The regex that filters jobs for importing (Optional)

Example of the import plugin configuration with all the available parameters:

```
parameters:
  rundeck:
    client:
      project:
        test_project:
          plugin:
            import:
              address: https://github.com/akscram/rundeck-jobs.git
```

```
            branch: master
            import_uuid_behavior: remove
            format: yaml
            path_template: ${job.group}${job.name}.${config.format}
            file_pattern: '.*\.yaml'
```

2. Apply the Rundeck client state:

```
salt-call state.sls rundeck.client
```

Configure iFrame forwarding

By default, the Rundeck service configuration does not enable you to get access through an external proxy address and exposed rundeck port, which is 14440 by default. Although, you can easily forward the Runbooks dashboard through a proxy endpoint in case of using Devops Portal through external proxy networks.

To configure iFrame forwarding:

1. Configure iFrame forwarding on the cluster level by specifying the following parameters in the oss/client.yml:

   ```
   rundeck_forward_iframe: True
   rundeck_iframe_host: <external-proxy-endpoint>
   rundeck_iframe_port: <external-proxy-port>
   rundeck_iframe_ssl: False
   ```

2. Apply the updated rundeck.server formula:

   ```
   salt -C 'I@rundeck:server' state.sls rundeck.server
   ```

3. Verify that there are no cached modules, grains, and so on; and minion configuration is updated:

   ```
   salt '*' saltutil.clear_cache
   salt -C 'I@docker:swarm:role:master' state.sls salt
   ```

4. Refresh and update your deployment:

   ```
   salt '*' saltutil.refresh_beacons
   salt '*' saltutil.refresh_grains
   salt '*' saltutil.refresh_modules
   salt '*' saltutil.refresh_pillar
   salt '*' saltutil.sync_all
   ```

5. Recreate the Rundeck stack:

   ```
   docker stack rm rundeck
   salt -C 'I@docker:swarm:role:master' state.sls docker.client
   salt -C 'I@rundeck:client' state.sls rundeck.client
   ```

6. Specify a custom endpoint for the DevOps portal on the cluster level of the Reclass model in the oss/client.yml file:

   ```
   devops_portal:
     config:
       service:
         rundeck:
   ```

```
    endpoint:
      address: ${_param:rundeck_iframe_host}
      port: ${_param:rundeck_iframe_port}
      https: ${_param:rundeck_iframe_ssl}
```

7. Recreate the DevOps portal stack:

```
docker stack rm devops-portal
salt -C 'I@devops_portal:config' state.sls devops_portal.config
salt -C 'I@docker:swarm:role:master' state.sls docker.client
```

Now, you can add an additional configuration for proxying the defined address and apply it on the proxy nodes.

Configure an external datasource

You can enable the Runbooks automation service to use an external datasource through the Salt metadata. This section explains how to configure the service to use the PostgreSQL database as an external source for the datastore.

To enable the PostgreSQL database support:

1. Define the following parameters on the cluster level of your Reclass model in the oss/client.yml file:

   ```
   parameters:
    _param:
      rundeck_postgresql_username: rundeck
      rundeck_postgresql_password: password
      rundeck_postgresql_database: rundeck
      rundeck_postgresql_host: ${_param:control_vip_address}
      rundeck_postgresql_port: 5432
    rundeck:
      server:
        datasource:
          engine: postgresql
          host: ${_param:rundeck_postgresql_host}
          port: ${_param:rundeck_postgresql_port}
          username: ${_param:rundeck_postgresql_username}
          password: ${_param:rundeck_postgresql_password}
          database: ${_param:rundeck_postgresql_database}
   ```

2. Recreate Rundeck and PostgreSQL stacks:

   ```
   docker stack rm postgresql rundeck
   salt -C 'I@rundeck:server' state.sls rundeck.server
   salt -C 'I@docker:swarm:role:master' state.sls docker.client
   salt -C 'I@postgresql:client' state.sls postgresql.client
   salt -C 'I@rundeck:client' state.sls rundeck.client
   ```

3. Verify that the Rundeck tables exist in PostgreSQL by logging as a Rundeck user in to PostgreSQL from the monitoring node where the OSS services are running and checking the log output for the base_report table. For example:

   ```
   psql -h <postgresql_ip> -U rundeck -W password
   rundeck=> \d
                  List of relations
    Schema |            Name            |   Type   |  Owner
   --------+----------------------------+----------+---------
    public | auth_token                 | table    | rundeck
    public | base_report                | table    | rundeck
    public | execution                  | table    | rundeck
    public | hibernate_sequence         | sequence | rundeck
   ```

```
 public | log_file_storage_request   | table    | rundeck
 public | node_filter                | table    | rundeck
 public | notification               | table    | rundeck
 public | orchestrator               | table    | rundeck
 public | plugin_meta                | table    | rundeck
 public | project                    | table    | rundeck
 public | rdoption                   | table    | rundeck
 public | rdoption_values            | table    | rundeck
 public | rduser                     | table    | rundeck
 public | report_filter              | table    | rundeck
 public | scheduled_execution        | table    | rundeck
 public | scheduled_execution_filter | table    | rundeck
 public | storage                    | table    | rundeck
 public | workflow                   | table    | rundeck
 public | workflow_step              | table    | rundeck
 public | workflow_workflow_step     | table    | rundeck
(20 rows)

rundeck=> select * from base_report;
```

Run preconfigured jobs from web UI

The Rundeck jobs and workflows are run automatically depending on configuration. Though, the DevOps Portal enables you to run the preconfigured Rundeck jobs and workflows using the web UI and track the progress of their execution.

To run a Rundeck job:

1. Log in to the DevOps Portal.
2. Navigate to the Runbooks dashboard.
3. Select the project you are interested in.
4. Navigate to the Jobs tab in the top navigation bar. The jobs page will display all jobs you are authorized to view.
5. If the jobs were defined inside groups, they will appear as a listing grouped into a folder. To reveal a folder content, press the folder icon.
6. Navigate to a required job, and clck on it. The job details page opens. This page contains the configuration parameters for this job as well as statistics, activity, and definition details for it.
7. To run the job, click Run job now.
8. Once you have started the job execution, follow the job's output in the Execution follow page.

Seealso

DriveTrain

Security

> Warning
>
> The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Security dashboard is the UI for the Security Audit Service, aka Security Monkey. The service runs tests to track and evaluate security-related tenant changes and configurations. Using the Security dashboard, you can search through these audit findings and review them.

> Note
>
> The Security Audit service depends on the following services:
>
> - DevOps Portal web UI
> - Push Notification service
> - PostgreSQL database

To review security item details:

1. Log in to the DevOps Portal.
2. To quickly access current security items with unjustified issues click the Security issues widget on the Dashboard tab. The Security items page opens.
3. Click the name of the required security item in the Items section to view its details. The item details page opens.
4. To revise the configuration change that caused the security item raise, use the Revisions section. The affected parts of configuration are color coded:
   - Green stands for additions
   - Red stands for deletions
5. To justify the unjustified issues, use the Issues section:
   1. Check the unjustified issue or issues.
   2. Edit the Justification field specifying the reason of justification. For example, This has been approved by the Security team.
   3. Click Justify.
6. To attach a comment containing any required content to an item:
   1. In the Item comments section, paste the comment to the comment field.

   2. Click Add comment.
7. To search for specific issues, use the Issues section. Each issue has a link to a page of the corresponding item containing the details of the issue.

Janitor

Warning

The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Janitor dashboard is the UI for the Cleanup service, also known as Janitor Monkey.

The Cleanup service is a tool that enables you to automatically detect and clean up unused resources in your MCP deployment that may include:

- OpenStack resources: virtual machines
- AWS resources: instances, EBS volumes, EBS snapshots, auto scaling groups, launch configurations, S3 bucket, security groups, and Amazon Machine Images (AMI)

The architecture of the service allows for easy configuration of the scanning schedule as well as a number of other operations. This section explains how to configure the Cleanup service to fit your business requirements as well as how to use the Janitor dashboard in the DevOps Portal.

Note

The Cleanup service depends on the following services:

- DevOps Portal web UI
- Push Notification service
- Cloud Intelligence service

Overview of the resource termination workflow

The resource termination workflow includes the following stages:

1. Determining and marking the clean-up candidate.

   The Cleanup service applies a set of pre-configured rules to the resources available in your cluster on a regular basis. If any of the rules is hold, the resource becomes a clean-up candidate and is marked accordingly.

2. Deleting the resource.

   The Cleanup service deletes the resource if the resource is marked as a clean-up candidate and the scheduled termination time passes. The resource owner can manually delete the resource to release it earlier.

Configure the scanning schedule

By default, the Janitor service scans your MCP cluster for unused resources once every hour from 11:00 a.m. to 11:59 p.m. in the Pacific Time Zone (PT) on week days. Though, you can easily override the default schedule by defining the simianarmy.properties environment variables depending on the needs of your environment.

The Janitor service schedule parameters

Parameter	Default	Description
simianarmy.scheduler.frequency	1	The parameter is connected to the frequencyUnit parameter determining the scanning cycle. The 1 frequency together with the HOURS frequencyUnit means that the scanning will be performed once every hour.
simianarmy.scheduler.frequencyUnit	HOURS	The available values include the java.util.concurrent.TimeUnit enum values.
simianarmy.calendar.openHour	11	Sets the time when the service starts performing any action (scheduling, deleting) on week days.
simianarmy.calendar.closeHour	11	Sets the time when the service stops performing any action (scheduling, deleting) on week days.
simianarmy.calendar.timezone	America/Los_Angeles	The time zone which the Janitor service operates in.

To configure the scanning schedule:

1. Log in to the Salt Master node.
2. In the classes/cluster/${_param:cluster_name}/oss/client.yml file of the Reclass model, define the Cleanup service schedule parameters as required. For example:

   ```
   docker:
     client:
       stack:
         janitor_monkey:
           environment:
             simianarmy.scheduler.frequency: 3
             simianarmy.scheduler.frequencyUnit: MINUTES
   ```

3. To apply the changes, recreate the stack:

```
salt -C 'I@docker:swarm:role:master' cmd.run 'docker stack rm janitor_monkey'
salt '*' saltutil.refresh_pillar
salt -C 'I@docker:swarm:role:master' state.sls docker.client
```

Clean up resources using web UI

The unused resources are cleaned up automatically according to the termination schedule. If you need to release an unused item earlier, you can terminate it manually using te DevOps Portal web UI.

To clean up resources manually:

1. Log in to the DevOps Portal web UI.
2. Navigate to the Janitor > Items tab.
3. Check the items you need to clean up immediately from the list of resources scheduled for termination.
4. Click Terminate.

Hardware Correlation

> Warning
>
> The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Hardware (HW) correlation dashboard provides the capability to generate an on-demand dashboard that graphically illustrates the consumption of the physical host resources by VMs running on this host. For example, you can search for a VM and get a dashboard with the CPU, memory, and disk consumption for the compute node where this specific VM is running.

> Note
>
> The HW correlation dashboard depends on the following services:
>
> - DevOps Portal web UI
> - Cloud Intelligence service
> - Prometheus service of StackLight LMA

To use the HW correlation dashboard:

1. Log in to the DevOps Portal.
2. Navigate to the HW correlation dashboard.
3. To generate a dashboard for a compute host(s) on which a specific VM(s) is running:
    1. If required, filter the VMs list by tenants where they are running using the VM tenants filter.
    2. Select a VM(s) from the VMs drop-down list.

        > Note
        >
        > By default, if the VM tenants filter is not used, all available VMs are present in the VMs drop-down list.

    3. If required, select resources from the Resources drop-down list.
    4. Click Search.
4. Read the generated dashboard:

- View the name of a compute host to which specified VMs belong and the list of selected VMs which are running on this host.
- Examine the line graphs illustrating the resources consumption. To view the values with their measures, hover over a required line.

> Note
>
> The y-axis may contain suffixes, such as K, M, G, and others. These suffixes correspond to prefixes of the units of measurement, such as Kilo, Mega, Giga, and so on depending on the measure.

5. To scale graphs by y-axis:
   - Click Zoom In to set y-axis start to the lowest point of selected lines and the y-axis end to the highest point of selected lines.
   - Click Zoom Out to switch to the default view where y-axis starts at 0 and ends at the highest point of all the lines on a chart.
6. To scale graphs by x-axis:
   - Click Expand to expand a graph to the full width.
   - Click Collapse to switch to the default view.
7. To select and hide lines on graphs:
   - Click on an item under a graph or on a line itself to view only the selected line. Combine this action with Zoom In for the detailed view.
   - Hover over an item under a graph to highlight the related line and mute the others.

DriveTrain

Warning

The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The DriveTrain dashboard provides access to a custom Jenkins interface. Operators can perform the following operations through the DevOps Portal UI:

- View the list of Jenkins jobs by views with job names and last build statuses and descriptions.
- View specific Jenkins job information including a list of builds with their statuses and descriptions as well as the stages for the last five builds.
- Analyze a specific build information including stages, console output, and artifact list on a build information page.
- View a job console output in real time and manage the build flow.
- Execute Jenkins jobs with custom parameters and re-run builds.

To perform all the above operations, use the DriveTrain dashboard available in the DevOps Portal UI.

Cloud Capacity Management

> Warning
>
> The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Cloud Capacity Management service provides point in time resource consumption data for OpenStack by displaying parameters such as total CPU utilization, memory utilization, disk utilization, and number of hypervisors. The related dashboard is based on data collected by the Cloud Intelligence service and can be used for cloud capacity management and other business optimization aspects.

> Note
>
> The Cloud Capacity Management service depends on the following services:
>
> - DevOps Portal web UI
> - Kibana service of StackLight LMA

Heatmaps

> Warning
>
> The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The Heatmaps dashboard provides the information about resource consumption by the cloud environment identifying the load of each node and number of alerts triggered for each node. The dashboard includes heatmaps for the following data:

- Memory utilization
- CPU utilization
- Disk utilization
- Alerts triggered

> Note
>
> The Heatmaps dashboard depends on the following services:
>
> - DevOps Portal web UI
> - Prometheus service of StackLight LMA

To use the Heatmaps dashboard:

1. Log in to the DevOps Portal.
2. Navigate to the Heatmaps dashboard.
3. Switch between the tabs to select a required heatmap.

   Each box on a heatmap represents a hypervisor. The box widget is color-coded:

   - Green represents a normal load or no alerts triggered
   - Orange represents a high load or low number of alerts
   - Red represents an overloaded node or a high number of alerts
4. Specify the parameters for the data to be displayed:

   - Use Now, Last 5m, Last 15m, and Last 30m buttons to view data for a specific time period.
   - Use the Custom button to set custom time period. The time value format includes the number from 1 to 99 and a metric prefix that is m for minutes, h for hours, d for days, and w for weeks. For example, 12h, 3d, and so on.

©2025, Mirantis Inc.

- Use the Max button to display a maximum value of resources consumption or number of alerts during the selected period of time.
- Use the Avg button on the Memory, CPU, and Disk tabs to display an average value of resources consumption during the selected period of time.
- Use the Diff button on the Alerts tab to display the count of alerts triggered since the selected period of time.

> Note
>
> On the Alerts tab, the 0 count of alerts means that either 0 alerts are triggered or Prometheus failed to receive the requested data for a specific node.

LMA

> Warning
>
> The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

The LMA tab provides access to the LMA (Logging, Monitoring, and Alerting) toolchain of the Mirantis Cloud Platform. More specifically, LMA includes:

- The LMA > Logging tab to access Kibana
- The LMA > Monitoring tab to access Grafana
- The LMA > Alerting tab to access the Prometheus web UI

> Note
>
> The LMA tab is only available in the DevOps Portal with LMA deployments and depends on the following services:
>
> - DevOps Portal web UI
> - Prometheus, Grafana, and Kibana services of StackLight LMA

LMA Logging dashboard

The LMA Logging tab provides access to Kibana. Kibana is used for log and time series analytics and provides real-time visualization of the data stored in Elasticsearch.

To access the Kibana dashboards from the DevOps Portal:

1. Log in to the DevOps Portal.
2. Navigate to the LMA > Logging tab.
3. Use Kibana as described in Manage Kibana dashboards and Use Kibana filters and queries.

LMA Monitoring dashboard

The LMA Monitoting tab provides access to the Grafana web service that builds and visually represents metric graphs based on time series databases. A collection of predefined Grafana dashboards contains graphs on particular monitoring endpoints.

To access the Grafana dashboards from the DevOps Portal:

1. Log in to the DevOps Portal.
2. Navigate to the LMA > Monitoring tab.
3. Log in to Grafana.
4. Select the required dashboard from the Home drop-down menu.

For information about the available Grafana dashboards, see View Grafana dashboards. To hide nodes from dashboards, see Hide nodes from dashboards.

LMA Alerting dashboard

The LMA Alerting tab provides access to the Prometheus web UI that enables you to view simple graphs, Prometheus configuration and rules, and states of the monitoring endpoints of your deployment.

To access the Prometheus web UI from the DevOps Portal:

1. Log in to the DevOps Portal.
2. Navigate to the LMA > Alerting tab.
3. Use the upper navigation menu to view alerts, graphs, or statuses. See View graphs and alerts and View Prometheus settings for details.

StackLight LMA operations

Using StackLight LMA, the Logging, Monitoring, and Alerting toolchain of the Mirantis Cloud Platform, cloud operators can monitor OpenStack environments, Kubernetes clusters, and OpenContrail services deployed on the platform and be quickly notified of critical conditions that may occur in the system so that they can prevent service downtimes.

This section describes how to configure and use StackLight LMA.

Configure StackLight LMA components

Once you deploy StackLight LMA, you may need to modify its components. For example, you may need to configure the Prometheus database, define alerting rules, and so on. The configuration of StackLight LMA is stored in Reclass. Therefore, you must modify the Reclass model and re-execute the Salt formulas.

Configure Telegraf

The configuration of the Telegraf agent is stored in the telegraf section of the Reclass model.

To configure Telegraf:

1. Log in to the Salt Master node.
2. Configure the telegraf section in the classes/cluster/cluster_name/init.yml file of the Reclass model as required.
3. Apply the Salt formula:

   ```
   salt -C 'I@linux:system' state.sls telegraf
   ```

Example configuration:

```yaml
telegraf:
  agent:
    enabled: true
    interval: 15
    round_interval: false
    metric_batch_size: 1000
    metric_buffer_limit: 10000
    collection_jitter: 2
    output:
      prometheus_client:
        bind:
          address: 0.0.0.0
          port: 9126
        engine: prometheus
```

In the example above, the Reclass model is converted to a configuration file recognized by Telegraf. For details about options, see the Telegraf documentation and the */meta/telegraf.yml file in every Salt formula.

The input and output YAML dictionaries contain a list of defined inputs and outputs for Telegraf. To add input or output parameters to Telegraf, use the same format as used in */meta/telegraf.yml of the required Salt formula. However, this should be performed only by deployment engineers or developers.

Configure Prometheus

You may need to configure Prometheus, for example, to modify an existing alert. Prometheus configuration is stored in the prometheus:server section of the Reclass model.

To configure Prometheus:

1. Log in to the Salt Master node.
2. Configure the prometheus:server section in the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model as required.
3. Update the Salt mine:

```
salt -C 'I@salt:minion' state.sls salt.minion.grains
salt -C 'I@salt:minion' saltutil.refresh_modules
salt -C 'I@salt:minion' mine.update
```

4. Apply the Salt formula:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
```

Example configuration:

```
prometheus:
  server:
    enabled: true
    bind:
      port: 9090
      address: 0.0.0.0
    storage:
      local:
        engine: "persisted"
        retention: "360h"
        memory_chunks: 1048576
        max_chunks_to_persist: 524288
        num_fingerprint_mutexes: 4096
    alertmanager:
      notification_queue_capacity: 10000
    config:
      global:
        scrape_interval: "15s"
        scrape_timeout: "15s"
        evaluation_interval: "1m"
        external_labels:
          region: 'region1'
    alert:
      PrometheusTargetDownKubernetesNodes:
        if: 'up{job="kubernetes-nodes"} != 1'
        labels:
```

©2025, Mirantis Inc.

```
      severity: down
      service: prometheus
    annotations:
      summary: 'Prometheus target down'
```

The following table describes the available settings.

Settings description

Setting	Description
storage	The storage YAML dictionary stores the configuration options for the Prometheus storage database. These options are passed to the Prometheus server through the command-line arguments.
config	The config YAML dictionary contains the options that will be placed in the Prometheus configuration file. For more information, see Prometheus configuration documentation.
alert	The alert YAML dictionary is used to generate Prometheus alerting rules. For more information, see Alerting rules. Alternatively, you can import alerts from the */meta/prometheus.yml file of any Salt formula. However, this should be performed only by deployment engineers or developers.

Caution!

The Prometheus data directory is mounted from the Docker host. If you restart a container, it can be spawned on a different host. This can cause Prometheus to start with an empty storage. In such case, the data will still be available on the previous host.

Seealso

Manage alerts

Configure Prometheus long-term storage

You may need to configure Prometheus long-term storage to change the external labels, scrape intervals and timeouts, and so on. Since Prometheus long-term storage and Prometheus Relay are connected, you can use the same configuration file to modify Prometheus Relay, for example, to change the bind port. The configuration of Prometheus long-term storage and Prometheus Relay is stored in the prometheus:server and prometheus:relay sections of the Reclass model.

To configure Prometheus long-term storage and Prometheus Relay:

1. Log in to the Salt Master node.
2. Configure the prometheus:server and prometheus:relay sections in the classes/cluster/<cluster_name>/stacklight/telemetry.yml file of the Reclass model as required.
3. Apply the Salt formula:

   ```
   salt -C 'I@prometheus:relay' state.sls prometheus
   ```

Example configuration of Prometheus long-term storage:

```
prometheus:
 server:
   dir:
     config: /etc/prometheus
     data: /var/lib/prometheus/data
   bind:
     port: 9090
     address: 0.0.0.0
   storage:
     local:
       retention: 4320h
   config:
     global:
       scrape_interval: 30s
       scrape_timeout: 30s
       evaluation_interval: 15s
       external_labels:
         region: region1
```

Example configuration of Prometheus Relay:

```
prometheus
 relay:
   enabled: true
   bind:
     port: 8080
```

©2025, Mirantis Inc.

```yaml
client:
  timeout: 12
```

Note

Configuring the timeout for Prometheus Relay is supported starting from the MCP 2019.2.4 maintenance update. To obtain the feature, follow the steps described in Apply maintenance updates.

Configure Alertmanager

The configuration of Alertmanager is stored in the prometheus:alertmanager section of the Reclass model. For available configuration settings, see the Alertmanager documentation.

To configure Alertmanager:

1. Log in to the Salt Master node.
2. Configure the prometheus:alertmanager section in the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model as required.
3. Apply the Salt formula:

```
salt -C 'I@docker:swarm:role:master and I@prometheus:server' state.sls prometheus.alertmanager
```

Example configuration:

```
prometheus:
  alertmanager:
    enabled: true
    bind:
      address: 0.0.0.0
      port: 9093
    config:
      global:
        resolve_timeout: 5m
      route:
        group_by: ['alertname', 'region', 'service']
        group_wait: 60s
        group_interval: 5m
        repeat_interval: 3h
        receiver: HTTP-notification
      inhibit_rules:
        - source_match:
            severity: 'down'
          target_match:
            severity: 'critical'
          equal: ['region', 'service']
        - source_match:
            severity: 'down'
          target_match:
            severity: 'warning'
          equal: ['region', 'service']
        - source_match:
            severity: 'critical'
         target_match:
            severity: 'warning'
          equal: ['alertname', 'region', 'service']
      receivers:
```

```
    - name: 'HTTP-notification'
      webhook_configs:
        - url: http://127.0.0.1
          send_resolved: true
```

Configure the logging system components

The logging system components include Fluentd (log collector), Elasticsearch, Elasticsearch Curator, and Kibana. You can modify the Reclass model to configure the logging system components. For example, your can configure Fluentd to gather logs from a custom entity.

Configure Fluentd

Fluentd gathers system and service logs and pushes them to the default output destination such as Elasticsearch, file, and so on. You can configure Fluentd to gather logs from custom entities, remove the default entities from the existing Fluentd configuration, as well as to filter and route logs. Additionally, you can configure Fluentd to expose metrics generated from logs to Prometheus.

Configure logs gathering

You can configure Fluentd to gather logs from custom entities, remove the default entities from the existing Fluentd configuration, as well as to filter and route logs. During configuration, you can define the following parameters:

- input, to gather logs from external sources such as a log file, TCP socket, and so on. For details, see Input plugin overview.
- filter, to filter the log entries gathered by the Input plugin. For example, to add, change, or remove fields. For details, see Filter plugin overview.
- match, to push final log entries to a given destination such as Elasticsearch, file, and so on. For details, see Output plugin overview.
- label, to connect the inputs. Logs gathered by Fluentd are processed from top-to-bottom of a configuration file. The label parameter connects the inputs, filters, and matches into a single flow. Using the label parameter ensures that filters for a given label are defined after input and before match.

> Note
>
> Perform all changes in the Reclass model. Add the custom log parsing rules used by a single environment to the cluster model. Place the log parsing rules for all deployments to the /meta/ directory in the Reclass model for a particular node. For details, see the */meta/fluentd.yml file of the required Salt formula.

To configure logs gathering:

1. Log in to the Salt Master node.
2. On the cluster level, specify the following snippets in the Reclass model for a particular node as required:

    - To add a new input:

    ```
    fluentd:
      agent:
        config:
          input:
            file_name:
              input_name:
                parameterA: 10
                parameterB: C
              input_nameB:
                parameterC: ABC
    ```

    - To add a new filter:

```yaml
fluentd:
  agent:
    config:
      filter:
        file_name:
          filter_name:
            parameterA: 10
            parameterB: C
          filter_nameB:
            parameterC: ABC
```

- To add a new match:

```yaml
fluentd:
  agent:
    config:
      match:
        file_name:
          match_name:
            parameterA: 10
            parameterB: C
          match_nameB:
            parameterC: ABC
```

- If the service requires a more advanced processing than gathering logs from an external source (input), add a label. For example, if you want to add filtering, use the label parameter that defines the whole flow. All entries in label are optional. So you can define filter and match but skip input.

```yaml
fluentd:
  agent:
    config:
      label:
        label_name:
          input:
            input1:
              parameter1: abc
            input2:
              parameter1: abc
          filter:
            filter1:
              parameter1: abc
              parameter2: abc
          match:
            match1:
              parameter1: abc
```

Example:

```
fluentd:
  agent:
    config:
      label:
        docker:
          input:
            container:
              type: tail
              tag: temp.docker.container.*
              path: /var/lib/docker/containers/*/*-json.log
              path_key: log_path
              pos_file: {{ positiondb }}/docker.container.pos
              parser:
                type: json
                time_format: '%Y-%m-%dT%H:%M:%S.%NZ'
                keep_time_key: false
          filter:
            enrich:
              tag: 'temp.docker.container.**'
              type: record_transformer
              enable_ruby: true
              record:
                - name: severity_label
                  value: INFO
                - name: Severity
                  value: 6
                - name: programname
                  value: docker
          match:
            cast_service_tag:
              tag: 'temp.docker.container.**'
              type: rewrite_tag_filter
              rule:
                - name: log_path
                  regexp: '^.*\/(.*)-json\.log$'
                  result: docker.container.$1
            push_to_default:
              tag: 'docker.container.*'
              type: relabel
              label: default_output
```

- To forward the logs gathered from a custom service to the default output, change the final match statement to default_output.

  Example:

```yaml
fluentd:
  agent:
    config:
      label:
        custom_daemon:
          input:
            ...
          match:
            push_to_default:
              tag: 'some.tag'
              type: relabel
              label: default_output
```

> Note
>
> The default output is defined in the system Reclass model. For details, see Default output. All Fluentd labels defined in /meta/ must use this mechanism to ensure log forwarding to the default output destination.

- To disable input, filter, match, or label, specify enabled: false for the required Fluentd entity.

  Example:

  ```yaml
  fluentd:
    agent:
      config:
        label:
          docker:
            enabled: false
  ```

3. Apply the following state:

```
salt -C 'node_name' state.sls fluentd
```

Add an additional output for Fluentd

If you have a syslog server and want StackLight LMA to send logs to this server, configure an additional output for Fluentd. In this case, Fluentd will push logs both to your syslog server and to Elasticsearch, which is the default target.

To add an additional output for Fluentd:

1. Download and install the td-agent-additional-plugins package on every host that runs Fluentd:

   ```
   apt-get install --only-upgrade td-agent-additional-plugins
   ```

2. Open your Git project repository with the Reclass model on the cluster level.
3. In the classes/cluster/<cluster_name>/infra/init.yml file, perform the following changes:

   1. Comment the system.fluentd.label.default_output.elasticsearch class.
   2. Copy the default_output parameters and rename to elasticsearch_output.
   3. To apply the existing filters for all outputs, copy the default output filter section to the new default output.
   4. Add syslog_output and specify the parameters as required.

   Example:

   ```
   classes:
   - system.fluentd
   - system.fluentd.label.default_metric
   - system.fluentd.label.default_metric.prometheus
   ## commented out
   #- system.fluentd.label.default_output.elasticsearch
   - service.fluentd.agent.output.syslog
   parameters:
     fluentd:
       agent:
         plugin:
           fluent-plugin-remote_syslog:
             deb: ['td-agent-additional-plugins']
         config:
           label:
             ## renamed previous default_output -> elasticsearch_output
             elasticsearch_output:
               match:
                 elasticsearch_output:
                   tag: "**"
                   type: elasticsearch
                   host: ${_param:fluentd_elasticsearch_host}
                   port: ${_param:elasticsearch_port}
             syslog_output:
   ```

```
        match:
          syslog_output:
            tag: "**"
            type: syslog
            host: 127.0.0.1
            port: 514
            ## optional params:
            # format: xxx
            # severity: xxx
            # facility: xxx
            # protocol: xxx
            # tls: xxx
            # ca_file: xxx
            # verify_mode: xxx
            # packet_size: xxx
            # timeout: xxx
            # timeout_exception: xxx
            # keep_alive: xxx
            # keep_alive_idle: xxx
            # keep_alive_cnt: xxx
            # keep_alive_intvl: xxx
      default_output:
       ## copy of previous default_output filter section
        # filter: {}
        match:
          send_to_default:
            tag: "**"
            type: copy
            store:
              - type: relabel
                label: syslog_output
              - type: relabel
                label: elasticsearch_output
```

4. Log in to the Salt Master node.
5. Synchronize Salt modules and refresh Salt pillars:

```
salt '*' saltutil.sync_all
salt '*' saltutil.refresh_pillar
```

6. Apply the following state:

```
salt '*' state.sls fluentd
```

Enable sending CADF events to external SIEM systems

> Note
>
> This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

You can configure Fluentd running on the RabbitMQ nodes to forward the Cloud Auditing Data Federation (CADF) events to specific external security information and event management (SIEM) systems, such as Splunk, ArcSight, or QRadar. The procedure below provides a configuration example for Splunk.

To enable sending CADF events to Splunk:

1. Open your project Git repository with Reclass model on the cluster level.
2. In classes/cluster/cluster_name/stacklight, create a custom notification channel, for example, fluentd_splunk.yml with the following pillar specifying the hosts and ports in the splunk_output and syslog_output parameters:

```
parameters:
 fluentd:
   agent:
     config:
       label:
         audit_messages:
           filter:
             get_payload_values:
               tag: audit
               type: record_transformer
               enable_ruby: true
               record:
                 - name: Logger
                   value: ${fluentd:dollar}{ record.dig("publisher_id") }
                 - name: Severity
                   value: ${fluentd:dollar}{ {'TRACE'=>7,'DEBUG'=>7,'INFO'=>6,\
                   'AUDIT'=>6,'WARNING'=>4,'ERROR'=>3,'CRITICAL'=>2}\
                   [record['priority']].to_i }
                 - name: Timestamp
                   value: ${fluentd:dollar}{ DateTime.strptime(record.dig\
                   ("payload", "eventTime"), "%Y-%m-%dT%H:%M:%S.%N%z").strftime\
                   ("%Y-%m-%dT%H:%M:%S.%3NZ") }
                 - name: notification_type
                  value: ${fluentd:dollar}{ record.dig("event_type") }
                 - name: severity_label
                   value: ${fluentd:dollar}{ record.dig("priority") }
```

```
        - name: environment_label
          value: ${_param:cluster_domain}
        - name: action
          value: ${fluentd:dollar}{ record.dig("payload", "action") }
        - name: event_type
          value: ${fluentd:dollar}{ record.dig("payload", "eventType") }
        - name: outcome
          value: ${fluentd:dollar}{ record.dig("payload", "outcome") }
    pack_payload_to_json:
      tag: audit
      require:
        - get_payload_values
      type: record_transformer
      enable_ruby: true
      remove_keys: '["payload", "timestamp", "publisher_id", "priority"]'
      record:
        - name: Payload
          value: ${fluentd:dollar}{ record["payload"].to_json }
  match:
    send_to_default:
      tag: "**"
      type: copy
      store:
        - type: relabel
          label: splunk_output
        - type: relabel
          label: syslog_output
splunk_output:
  match:
    splunk_output:
      tag: "**"
      type: splunk_hec
      host: <splunk_host>
      port: <splunk_port>
      token: <splunk_token>
syslog_output:
  match:
    syslog_output:
      tag: "**"
      type: syslog
      host: <syslog_host>
      port: <syslog_port>
```

3. In openstack/message_queue.yml:

    1. Replace the system.fluentd.notifications class with the following ones:

```
classes:
- system.fluentd.label.notifications.input_rabbitmq
- system.fluentd.label.notifications.notifications
```

2. Add the custom Fluentd channel as required. For example:

```
cluster.<cluster_name>.stacklight.fluentd_splunk
```

4. Log in to the Salt Master node.
5. Apply the fluentd state on the msg nodes:

```
salt -C 'I@rabbitmq:server' state.sls fluentd
```

Enable Fluentd to expose metrics generated from logs

You can enable exposing metrics that are based on the log events. This allows monitoring of various activities such as disk failures (metric hdd_errors_total). By default, Fluentd generates metrics from the logs it gathers. However, you must configure Fluentd to expose such metrics to Prometheus. Prometheus gathers Fluentd metrics as a static Prometheus endpoint. For details, see Add a custom monitoring endpoint. To generate metrics from logs, StackLight LMA uses the fluent-plugin-prometheus plugin.

To configure Fluentd to expose metrics generated from logs:

1. Log in to the Salt Master node.
2. Add the following class to the cluster/<cluster_name>/init.yml file of the Reclass model:

   ```
   system.fluentd.label.default_metric.prometheus
   ```

   This class creates a new label default_metric that is used as a generic interface to expose new metrics to Prometheus.
3. (Optional) Create a filter for metric.metric_name to generate the metric.

   Example:

   ```
   reclass:
   fluentd:
     agent:
       label:
         default_metric:
           filter:
             metric_out_of_memory:
               tag: metric.out_of_memory
               type: prometheus
               metric:
                 - name: out_of_memory_total
                   type: counter
                   desc: The total number of OOM.
               label:
                 - name: host
                   value: ${Hostname}
             metric_hdd_errors_parse:
               tag: metric.hdd_errors
               type: parser
               key_name: Payload
               parser:
                 type: regexp
                 format: '/(?<device>[sv]d[a-z]+\d*)/'
             metric_hdd_errors:
               tag: metric.hdd_errors
               require:
   ```

```
          - metric_hdd_errors_parse
        type: prometheus
        metric:
          - name: hdd_errors_total
            type: counter
            desc: The total number of hdd errors.
        label:
          - name: host
            value: ${Hostname}
          - name: device
            value: ${device}
  systemd:
    output:
      push_to_default:
        tag: '*.systemd'
        type: copy
        store:
          - type: relabel
            label: default_output
          - type: rewrite_tag_filter
            rule:
              - name: Payload
                regexp: '^Out of memory'
                result: metric.out_of_memory
              - name: Payload
                regexp: >-
                  'error.+[sv]d[a-z]+\d*'
                result: metric.hdd_errors
              - name: Payload
                regexp: >-
                  '[sv]d[a-z]+\d*.+error'
                result: metric.hdd_errors
      push_to_metric:
        tag: 'metric.**'
        type: relabel
        label: default_metric
```

Configure log rotation

Fluentd uses two options to modify the log files rotation, the logrotate parameter that controls log rotation on a daily basis and the internal td_agent_log_rotate_size parameter, which sets the internal log rotation by file size and is set to 10 MB by default. If a log file exceeds this limit, the internal log rotation service of Fluentd applies the log rotation. You can modify td_agent_log_rotate_size if required.

To configure log rotation:

1. Log in to the Salt Master node.
2. Specify the following parameter in the cluster/<cluster_name>/init.yml file of the Reclass model:

   ```
   parameters:
     fluentd:
       agent:
         td_agent_log_rotate_size: <custom_value_in_bytes>
   ```

3. Apply the following state:

   ```
   salt -C 'I@fluentd:agent' state.sls fluentd
   ```

Configure Elasticsearch

The configuration parameters of Elasticsearch are defined in the corresponding sections of the Reclass model.

To configure Elasticsearch:

1. Log in to the Salt Master node.
2. Configure the parameters:elasticsearch section in the classes/cluster/<cluster_name>/stacklight/log.yml file of the Reclass model as required. For example, to limit the heap size, specify the following snippet:

```
parameters:
  elasticsearch:
    server:
      heap:
        size: 31
```

3. Apply the Salt state:

```
salt -C 'I@elasticsearch:server' state.sls elasticsearch
```

For configuration examples, see the README.rst at Elasticsearch Salt formula.

Configure Elasticsearch Curator

The Elasticearch Curator tool manages the data (indices) and the data retention policy in Elasticsearch clusters. You can modify the indices and the retention policy.

To configure Elasticsearch Curator:

1. Open your Reclass model Git repository on the cluster level.
2. Modify the classes/cluster/<cluster_name>/stacklight/log_curator.yml file as required:

   - To configure indices, set the required prefixes using the elasticsearch_curator_indices_pattern parameter. The default value is "^(log|audit)-.*$", meaning that Curator manages the indices with log- and audit- prefixes.
   - To configure the retention policy for logs and audit indices, specify the elasticsearch_curator_retention_period parameter. The retention period is set to 31 days by default.
   - To configure the retention policy for notification indices, specify the elasticsearch_curator_notifications_retention_period parameter. The retention period is set to 90 days by default.
3. Log in to the Salt Master node.
4. Apply the following state:

```
salt -C 'I@elasticsearch:server' state.sls_id elasticsearch_curator_action_config elasticsearch
```

Configure Kibana

The configuration parameters of Kibana are defined in the corresponding sections of the Reclass model.

To configure Kibana:

1. Log in to the Salt Master node.
2. Configure the parameters:kibana section in the classes/cluster/<cluster_name>/stacklight/server.yml of the Reclass model as required.
3. Apply the Salt state:

   ```
   salt -C 'I@kibana:server' state.sls kibana
   ```

For configuration examples, see the README.rst at Kibana Salt formula.

Configure Grafana

The configuration of Grafana is stored in the grafana section of the Reclass model.

To configure Grafana:

1. Log in to the Salt Master node.
2. Configure the grafana section in the classes/cluster/<cluster_name>/stacklight/server.yml file of the Reclass model as required.
3. Apply the Salt formulas:

```
salt -C 'I@grafana:server' state.sls grafana.server
salt -C 'I@grafana:client' state.sls grafana.client
```

Example configuration:

```
grafana:
  server:
    enabled: true
    bind:
      address: 127.0.0.1
      port: 3000
    database:
      engine: mysql
      host: 127.0.0.1
      port: 3306
      name: grafana
      user: grafana
      password: db_pass
    auth:
      basic:
        enabled: true
    admin:
      user: admin
      password: admin_pass
    dashboards:
      enabled: false
      path: /var/lib/grafana/dashboards
```

Configure InfluxDB

Warning

InfluxDB, including InfluxDB Relay and remote storage adapter, is deprecated in the Q4`18 MCP release and will be removed in the next release.

The configuration of InfluxDB is stored in the parameters:influxdb section of the Reclass model.

To configure InfluxDB:

1. Log in to the Salt Master node.
2. Configure the parameters:influxdb section in the classes/cluster/<cluster_name>/stacklight/server.yml file of the Reclass model as required.
3. Apply the Salt state:

```
salt -C 'I@influxdb:server' state.sls influxdb
```

For configuration examples, see the README.rst at InfluxDB Salt formula.

Configure authentication for Prometheus and Alermanager

> Note
>
> This feature is available starting from the MCP 2019.2.7 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

You can configure basic authentication to access Prometheus and Alertmanager web UI through the proxy nodes that are available if external access to cloud resources is enabled in your OpenStack deployment.

This section describes how to configure authentication for Prometheus and Alertmanager by defining new passwords instead of the default ones on the existing MCP deployments updated to 2019.2.7. For new clusters starting from the MCP 2019.2.7 maintenance update, you define custom credentials for Prometheus and Alertmanager during the cluster creation.

To configure authentication for Prometheus and Alertmanager:

1. Log in to the Salt Master node.
2. Obtain user names and passwords:

   1. For Alertmanager:

      ```
      salt -C 'I@horizon:server' pillar.get _param:nginx_proxy_prometheus_alertmanager_password
      salt -C 'I@horizon:server' pillar.get _param:nginx_proxy_prometheus_alertmanager_user
      ```

   2. For Prometheus:

      ```
      salt -C 'I@horizon:server' pillar.get _param:nginx_proxy_prometheus_server_user
      salt -C 'I@horizon:server' pillar.get _param:nginx_proxy_prometheus_server_password
      ```

3. Change the default credentials for Prometheus and Alertmanager:

   1. Open the classes/cluster/<cluster_name>/stacklight/proxy.yml file for editing.
   2. Specify new passwords using the following parameters:

      ```
      parameters:
        _param:
          nginx_proxy_prometheus_alertmanager_password: <password>
          nginx_proxy_prometheus_server_password: <password>
      ```

   3. Optional. Specify new user names using the following parameters:

      ```
      parameters:
        _param:
      ```

```
nginx_proxy_prometheus_alertmanager_user: <user_name>
nginx_proxy_prometheus_server_user: <user_name>
```

4. On all proxy nodes, synchronize Salt modules and apply the nginx state. For example:

```
salt 'prxNode01*' saltutil.sync_all
salt 'prxNode01*' state.sls nginx
```

5. Verify authentication through the proxy nodes. For example:

```
salt 'prxNode01*' pillar.get _param:cluster_vip_address
salt 'prxNode01*' pillar.get nginx:server:site:nginx_proxy_prometheus_server:proxy:port
salt 'prxNode01*' pillar.get nginx:server:site:nginx_proxy_prometheus_alertmanager:proxy:port
curl https://<cluster_vip_address>:<prometheus_server_port>
curl -u <username>:<password> https://<cluster_vip_address>:<prometheus_server_port>
```

Enable Docker garbage collection

To avoid unused Docker images and volumes consuming the entire disk space, you can enable a clean-up cron job for old StackLight LMA containers and volumes. By default, the cron job runs daily at 6:00 a.m. and cleans stopped StackLight LMA images and containers that are older than one week.

To enable Docker garbage collection:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In classes/cluster/<cluster_name>/stacklight/server.yml, specify the following parameter:

   ```
   _param:
     docker_garbage_collection_enabled: true
   ```

3. Optional. To change the default parameters, use:

   ```
   linux:
     system:
       cron:
         user:
           root:
             enabled: true
       job:
         docker_garbage_collection:
           command: docker system prune -f --filter until=$(date +%s -d "1 week ago")
           enabled: ${_param:docker_garbage_collection_enabled}
           user: root
           hour: 6
           minute: 0
   ```

4. Log in to the Salt Master node.
5. Apply the following state:

   ```
   salt -C 'I@docker:swarm' state.sls linux.system.cron
   ```

Disable HTTP probes for OpenStack public endpoints

> Note
>
> This feature is available starting from the MCP 2019.2.12 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

By default, Telegraf checks all endpoints from the OpenStack service catalog, including the public, admin, and internal endpoints. In some cases, public endpoints may be unreachable from the Telegraf container. For such cases, you can configure StackLight to skip HTTP probes for OpenStack public endpoints.

To disable or enable HTTP probes for OpenStack public endpoints:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In <cluster_model>/stacklight/server.yml, specify the skip_public_endpoints parameter as required:

```
parameters:
  telegraf:
    remote_agent:
      input:
        openstack_api:
          skip_public_endpoints: true
```

3. Log in to the Salt Master node.
4. Apply the telegraf state to the mon nodes:

```
salt 'mon*' state.sls telegraf
```

Restart StackLight LMA components

You may need to restart one of the StackLight LMA components. For example, if its service hangs.

Restart services running in Docker Swarm

The Prometheus, Alertmanager, Alerta, Pushgateway, and Grafana services are running in the Docker Swarm mode. This section describes how to restart these services.

To restart services running in Docker Swarm:

1. Log in to the Salt Master node.
2. Issue one of the following commands depending on the service you want to restart:
    - To restart Prometheus:

      ```
      salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run \
      "docker service update monitoring_server --force"
      ```
    - To restart Alertmanager:

      ```
      salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run \
      "docker service update monitoring_alertmanager --force"
      ```
    - To restart Alerta:

      ```
      salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run \
      "docker service update monitoring_alerta --force"
      ```
    - To restart Pushgateway:

      ```
      salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run \
      "docker service update monitoring_pushgateway --force"
      ```
    - To restart Grafana:

      ```
      salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run \
      "docker service update dashboard_grafana --force"
      ```
    - To restart Prometheus Relay:

      ```
      salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run \
      "docker service update monitoring_relay --force"
      ```
    - To restart Prometheus Remote Agent:

      ```
      salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run \
      "docker service update monitoring_remote_agent --force"
      ```

Restart the logging system components

The logging system components include Fluentd, Elasticsearch, and Kibana. If required, you can restart these components.

To restart Fluentd

The Fluentd process that is responsible for collecting logs is td-agent (Treasure Data Agent). The td-agent process starts automatically when the fluentd Salt state is applied. To manually start and stop it, use the Salt commands.

The following example shows how to restart the td-agent process from the Salt Master node on all Salt Minion nodes with names that start with ctl:

```
# salt 'ctl*' service.restart td-agent
```

Alternatively, SSH to the node and use the service manager (systemd or upstart) to restart the service. For example:

```
# ssh ctl01.mcp-lab-advanced.local
# service td-agent restart
```

See the salt.modules.service documentation for more information on how to use the Salt service execution module.

To restart Elasticsearch

Run the following command from the Salt Master node:

```
# salt 'log*' service.restart elasticsearch
```

To restart Kibana

Run the following command from the Salt Master node:

```
# salt 'log*' service.restart kibana
```

©2025, Mirantis Inc.

Restart Telegraf

The Telegraf service is called telegraf.

To restart Telegraf on all nodes:

1. Log in to the Salt Master node.
2. Run the following command:

   ```
   salt -C 'I@telegraf:agent' service.restart telegraf
   ```

Restart InfluxDB

> Warning
>
> InfluxDB, including InfluxDB Relay and remote storage adapter, is deprecated in the Q4`18 MCP release and will be removed in the next release.

The InfluxDB service is called influxdb.

To restart InfluxDB on all nodes:

1. Log in to the Salt Master node.
2. Run the following command:

   ```
   salt -C 'I@influxdb:server' service.restart influxdb -b 1
   ```

Restart InfluxDB Relay

> Warning
>
> InfluxDB, including InfluxDB Relay and remote storage adapter, is deprecated in the Q4`18 MCP release and will be removed in the next release.

The InfluxDB Relay service is called influxdb-relay.

To restart InfluxDB Relay:

1. Log in to the Salt Master node.
2. Run the following command:

   ```
   salt -C 'I@influxdb:server' service.restart influxdb-relay -b 1
   ```

Restart Prometheus Relay and Prometheus long-term storage

You can restart Prometheus Relay and Prometheus long-term storage, for example, if one of the services hangs. Since these services are connected, you must restart both.

To restart Prometheus Relay and Prometheus long-term storage:

1. Log in to the Salt Master node.
2. Run the following commands:

   ```
   salt -C 'I@prometheus:relay' service.restart prometheus
   salt -C 'I@prometheus:relay' service.restart prometheus-relay
   ```

Manage endpoints, metrics, and alerts

You can easily configure Stacklight LMA to support new monitoring endpoints, add custom metrics and alerts, and modify or disable the existing alerts.

Add a custom monitoring endpoint

If required, you can add a custom monitoring endpoint to Prometheus, such as Calico, etcd, or Telegraf.

To add a custom monitoring endpoint:

1. Log in to the Salt Master node.
2. Configure the prometheus:server section in the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model as required. Add the monitoring endpoint IP and port.

   Example:

   ```
   prometheus:
     server:
       target:
         static:
           endpoint_name:
             endpoint:
               - address: 1.1.1.1
                 port: 10
               - address: 2.2.2.2
                 port: 10
   ```

3. Apply the Salt formula:

   ```
   salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
   ```

Add a custom metric

If required, you can add a custom metric, for example, to monitor a third-party software. This section describes how to add a custom metric to Telegraf.

> Note
>
> To add a custom metric to a new endpoint, you must first add the new endpoint to Prometheus as described in Add a custom monitoring endpoint.

To add a custom metric to Telegraf:

1. Log in to the Salt Master node.
2. Edit the telegraf section in the classes/cluster/cluster_name/init.yml file of the Reclass model as required.

   Example:

   ```
   telegraf:
     agent:
       input:
         procstat:
           process:
             memcached:
               exe: memcached
         memcached:
           servers:
             - address: {{ server.bind.address | replace("0.0.0.0", "127.0.0.1") }}
               port: {{ server.bind.port }}
   ```

3. Apply the Telegraf Salt formula:

   ```
   salt -C 'I@linux:system' state.sls telegraf
   ```

Manage alerts

You can easily extend StackLight LMA to support a new service check by adding a custom alert. You may also need to modify or disable the default alerts as required.

To create a custom alert:

1. Log in to the Salt Master node.
2. Add the new alert to the prometheus:server:alert section in the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model. Enter the alert name, alerting conditions, severity level, and annotations that will be shown in the alert message.

   Example:

   ```
   prometheus:
     server:
       alert:
         EtcdFailedTotalIn5m:
           if: >-
             sum by(method) (rate(etcd_http_failed_total{code!~"4[0-9]{2}"}[5m]))
             / sum by(method) (rate(etcd_http_received_total[5m])) > {{
             prometheus_server.get('alert', {}).get('EtcdFailedTotalin5m', \
             {}).get('var', {}).get('threshold', 0.01) }}
           labels:
             severity: warning
             service: etcd
           annotations:
             summary: 'High number of HTTP requests are failing on etcd'
             description: '{{ $value }}% of requests for {{ $labels.method }} \
             failed on etcd instance {{ $labels.instance }}'
   ```

3. Apply the Salt formula:

   ```
   salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
   ```

4. To view the new alert, see the Prometheus logs:

   ```
   docker service logs monitoring_server
   ```

   Alternatively, see the Alerts tab of the Prometheus web UI.

To modify a default alert:

1. Log in to the Salt Master node.
2. Modify the required alert in the prometheus:server:alert section in the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model.
3. Apply the Salt formula:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
```

4. To view the changes, see the Prometheus logs:

```
docker service logs monitoring_server
```

   Alternatively, see the alert details in the Alerts tab of the Prometheus web UI.

To disable an alert:

1. Log in to the Salt Master node.
2. Create the required alert definition in the prometheus:server:alert section in the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model and set the enabled parameter to false.

   Example:

```
prometheus:
  server:
    alert:
      EtcdClusterSmall:
        enabled: false
```

3. Apply the Salt formula:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
```

4. Verify the changes in the Alerts tab of the Prometheus web UI.

Configure StackLight LMA to send notifications

To enable StackLight LMA to send notifications, you must modify the Reclass model. By default, email notifications will be sent. However, you can configure StackLight LMA to send notifications to Salesforce, Slack, or other receivers. Additionally, you can specify a notification receiver for a particular alert or alert types or disable notifications.

Enable Alertmanager notifications

To enable StackLight LMA to send notifications, you must modify the Reclass model. By default, email notifications will be sent. However, you can configure StackLight LMA to send notifications to Salesforce, Slack, or other notifications receivers. You can also specify a notification channel for a particular alert or disable notifications.

Enable email or Slack notifications

This section describes how to enable StackLight LMA to send notifications to email, Slack, or to both notification channels using the Alertmanager service on an existing MCP cluster. By default, StackLight LMA uses Alertmanager and the SMTP protocol or the webhook receiver to send email or Slack notifications respectively.

> **Note**
>
> Skip this section if you require only email notifications and have already defined the variables for Alertmanager email notifications during the deployment model creation as described in MCP Deployment Guide: Infrastructure related parameters: Alertmanager email notifications.

To enable StackLight LMA to send notifications through Alertmanager:

1. Log in to the Salt Master node.
2. Open the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model for editing.
3. Add the following classes:

   - For email notifications:

     ```
     classes:
     [...]
     - system.prometheus.alertmanager.notification.email
     - system.prometheus.server.alert.labels_add.route
     ```

   - For Slack notifications:

     ```
     classes:
     [...]
     - system.prometheus.alertmanager.notification.slack
     - system.prometheus.server.alert.labels_add.route
     ```

4. Define the following variables:

   - For email notifications:

     ```
     parameters:
       _param:
         alertmanager_notification_email_from: <email_from>
         alertmanager_notification_email_host: <smtp_server:port>
         alertmanager_notification_email_password: <email_password>
         alertmanager_notification_email_require_tls: <email_require_tls>
     ```

```
    alertmanager_notification_email_to: <email_to>
    alertmanager_notification_email_username: <email_username>
```

> Note
>
> Using the alertmanager_notification_email_host parameter, specify both the host and the port number of the SMTP server. For example, host.com:25.

* For Slack notifications:

```
parameters:
  _param:
    alertmanager_notification_slack_api_url: https://hooks.slack.com/services/<webhook/integration/token>
```

5. Set one or multiple notification channels by using the _param:prometheus_server_alert_label_route parameter. The default value is email, which means that email notifications will be sent.

   Example:

```
parameters:
  _param:
    prometheus_server_alert_label_route: email;slack;
```

6. Apply the Salt formulas:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.alertmanager -b 1
```

Enable Salesforce notifications

This section describes how to enable StackLight LMA to create Salesforce cases from Prometheus alerts on an existing cluster. StackLight LMA uses Alertmanager and the Salesforce notifier service to create the Salesforce cases.

> **Note**
>
> Skip this section if you have already defined the variables for Alertmanager Salesforce notifications during the deployment model creation as described in MCP Deployment Guide: General deployment parameters and MCP Deployment Guide: Infrastructure related parameters: Alertmanager Salesforce notifications.
>
> If you configured Salesforce notifications through the Push Notification service, first proceed to Switch to Alertmanager-based notifications.

To enable StackLight LMA to send Salesforce notifications through Alertmanager:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In classes/cluster/<cluster_name>/stacklight/client.yml, specify:

```
classes:
- system.docker.swarm.stack.monitoring.sf_notifier

[...]

parameters:
  _params:
    docker_image_sf_notifier: "${_param:mcp_docker_registry}/openstack-docker/sf_notifier:${_param:mcp_version}"
```

3. In classes/cluster/<cluster_name>/stacklight/server.yml, specify:

```
classes:
- system.prometheus.alertmanager.notification.salesforce
- system.prometheus.sf_notifier.container

[...]

parameters:
  _params:
    sf_notifier_sfdc_auth_url: "<salesforce_instance_http_endpoint>"
    sf_notifier_sfdc_username: "<customer_account_email>"
    sf_notifier_sfdc_password: "<customer_account_password>"
    sf_notifier_sfdc_organization_id: "<organization_id>"
    sf_notifier_sfdc_environment_id: "<cloud_id>"
    sf_notifier_sfdc_sandbox_enabled: "True/False"
```

> Warning
>
> If you have previously configured email notifications through Alertmanager, verify that the prometheus_server_alert_label_route parameter in server.yml includes not only the email but also salesforce values.

4. Log in to the Salt Master node.
5. Refresh Salt pillars:

```
salt '*' saltutil.refresh_pillar
```

6. Create the directory structure for the Salesforce notifier service:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.sf_notifier
```

7. Start the sf-notifier service in Docker container:

```
salt -C 'I@docker:swarm:role:master' state.sls docker.client
```

8. Update the Prometheus configuration to create metrics target and alerts:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b 1
```

9. Update the Alertmanager configuration to create the webhook receiver:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.alertmanager -b 1
```

©2025, Mirantis Inc.

Configure Alertmanager integrations

> Note
>
> This feature is available starting from the MCP 2019.2.9 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes how to enable StackLight LMA to send notifications to specific receivers such as PagerDuty, OpsGenie, and so on using the Alertmanager service on an existing MCP cluster. For a list of supported receivers, see Prometheus Alertmanager documentation: Receiver.

The changes performed within the following procedure are backward compatible with previous MCP releases. Therefore, you do not need to change any of the already configured Alertmanager receivers and routes.

To enable Alertmanager integrations:

1. Open your project Git repository with the Reclass model on the cluster level.
2. Open the <classes/cluster/<cluster>/stacklight/server.yml> file for editing.
3. Configure the Alertmanager receiver route as required:

```
parameters:
  prometheus:
    alertmanager:
      enabled: true
      config:
        route:
          routes:
            <route_name>:
              receiver: <receiver_name>
              match_re:
                - label: '<name_of_the_alert_label>'
                  value: '<regex_to_identify_the_route>'
              continue: true
        receiver:
          <receiver_name>:
            enabled: true
            generic_configs:  # <- here is the difference
              <chosen_Alertmanager_receiver_type>:
                <receiver_endpoint_name>:
                  <receiver_config>
```

Parameters description

Parameter name	Description
routes	Specify a unique route name.
receiver	Specify a unique receiver name.
match_re	• In label, specify the name of the label to filter the alerts. • In value, specify a regular expression that Alertmanager will use to route the alert.
receiver	Specify a unique receiver name. Set to the same value as routes.
generic_configs	• Specify one of the available Alertmanager receiver types. • Specify a unique receiver endpoint name. • Specify the endpoint configuration in the YAML format.

Example configuration:

```
parameters:
  prometheus:
    alertmanager:
      enabled: true
      config:
        route:
          routes:
            opsgenie:
              receiver: HTTP-opsgenie
              match_re:
                - label: route
                  value: '(.*opsgenie.*)'
              continue: true
        receiver:
          HTTP-opsgenie:
            enabled: true
            generic_configs:
              opsgenie_configs:
                opsgenie-endpoint:
                  api_url: "https://example.app.eu.opsgenie.com/"
                  api_key: "opsgeniesecretkey"
                  send_resolved: true
```

4. If you have previously configured email notifications through Alertmanager, verify that the prometheus_server_alert_label_route parameter also includes the receiver that you are configuring. For example:

```
parameters:
  _param:
    prometheus_server_alert_label_route: email;opsgenie;
```

5. Log in to the Salt Master node.
6. Verify the Reclass model:

```
reclass -i
```

   The output should not include any errors.
7. Verify that the changes render properly:

```
salt '*mon*' state.sls prometheus.alertmanager test=True
```

   Example of system response for the configuration provided above:

```
routes:
  - receiver: default
[...]
  # opsgenie
  - receiver: HTTP-opsgenie
    continue: True
    match_re:
      route: '(.*opsgenie.*)'
[...]
receivers:
  - name: 'default'
[...]
  - name: 'HTTP-opsgenie'
    opsgenie_configs:
    # opsgenie-endpoint
    -
      api_key: opsgeniesecretkey
      api_url: https://example.app.eu.opsgenie.com/
      send_resolved: true
[...]
```

8. Update the Prometheus and Alertmanager configuration:

```
salt '*mon*' state.sls prometheus.server -b 1
salt '*mon*' state.sls prometheus.alertmanager
```

Switch to Alertmanager-based notifications

This section describes how to switch from the Push Notification service to the Alertmanager-based notifications. For more information on the Alertmanager-based notifications, see MCP Reference Architecture: StackLight LMA components.

> **Caution!**
>
> Before you start, perform the following prerequisite steps:
>
> 1. Upgrade StackLight LMA to the newest version as described in Upgrade StackLight LMA to Build ID 2019.2.0.
> 2. For the Salesforce notifications, verify that you have access to the Salesforce instance and have a customer account in Salesforce.

> **Warning**
>
> The Push Notification service uses the md5 hashing algorithm for creating alert IDs for the Salesforce notifications, whereas the Salesforce notifier service uses sha256 by default. Switching to sha256 without migration of Salesforce cases may lead to cases loss from the Salesforce notifier scope. If these services have different hashing set up, case duplication with the same subject and status but different IDs may occur. Therefore, Mirantis recommends explicitly setting the md5 hashing algorithm in your model configuration as described below. Migration of old cases is not supported.

To Switch to the Alertmanager-based notifications:

1. Open your Git project repository with Reclass model on the cluster level.
2. Set up the Alertmanager-based notifications:
   - For email notifications, follow the procedure described in Enable email or Slack notifications.
   - For Salesforce notifications:
     1. Set the md5 hashing algorithm in the <cluster>/stacklight/server.yml file:

        ```
        _params:
          sf_notifier_alert_id_hash_func: md5
        ```

     2. Set up the Salesforce notifier service as described in Enable Salesforce notifications.
3. Disable the Push Notification service:

1. Open the classes/cluster/<cluster_name>/stacklight/server.yml file for editing.
2. Remove the following class:

```
- system.prometheus.alertmanager.notification.pushkin
```

3. Remove the following parameters:

```
alertmanager_notification_pushkin_host: <host>
alertmanager_notification_pushkin_port: <port>
```

4. Log in to the Salt Master node.
5. Refresh Salt pillars:

```
salt '*' saltutil.refresh_pillar
```

6. Apply the following state:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.alertmanager -b 1
```

7. (Optional) Remove the Push Notification service:

   1. Verify that only the Push Notification service uses the database in the Docker container.
   2. In the Git project repository with Reclass model on the cluster level, open the classes/cluster/<cluster_name>/stacklight/server.yml file for editing.
   3. Remove the following classes:

   ```
   - system.haproxy.proxy.listen.oss.pushkin
   - system.haproxy.proxy.listen.oss.postgresql
   - system.docker.swarm.stack.pushkin
   - system.docker.swarm.stack.postgresql
   - system.docker.swarm.network.oss_backend
   - system.postgresql.client.pushkin
   - system.postgresql.client.pushkin.sfdc
   - system.glusterfs.server.volume.postgresql
   - system.glusterfs.server.volume.pushkin
   - system.glusterfs.client.volume.postgresql
   - system.glusterfs.client.volume.pushkin
   ```

   4. Remove the following parameters:

   ```
   postgresql_client_user: 'postgres'
   postgresql_client_password: 'postgrespassword'
   ```

   5. Log in to the Salt Master node.

6. Apply the changes:

```
salt -C 'I@haproxy:proxy' state.sls haproxy.proxy
salt -C 'I@docker:swarm and I@prometheus:server' state.sls docker.client
```

7. Remove the Push Notification service from Docker Swarm:

```
salt -C 'I@docker:swarm and I@prometheus:server' cmd.run 'docker stack rm pushkin'
```

8. Remove the PostgreSQL service from Docker Swarm:

```
salt -C 'I@docker:swarm and I@prometheus:server' cmd.run 'docker stack rm  postgresql'
```

9. Remove the Docker images for the Push Notification service and PostgreSQL:

```
salt -C 'I@docker:swarm and I@prometheus:server' cmd.run 'docker rmi <postgres_image> <pushkin_image>'
```

10. Stop the Push Notification service and PostgreSQL:

```
salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run 'echo "y" | gluster volume stop pushkin'
salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run 'echo "y" | gluster volume stop postgresql'
```

11. Delete the volumes:

```
salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run 'echo "y" | gluster volume delete pushkin'
salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run 'echo "y" | gluster volume delete postgresql'
```

12. Remove the directories for GlusterFS bricks:

```
salt -C 'I@docker:swarm and I@prometheus:server' cmd.run 'rm -rf /srv/glusterfs/pushkin'
salt -C 'I@docker:swarm and I@prometheus:server' cmd.run 'rm -rf /srv/glusterfs/postgres'
```

13. (Optional) Uninstall GlusterFS:

```
sudo apt remove -y glusterfs-server
```

Customize alerts for notifications

Once configured, StackLight LMA sends notifications about all alerts. However, you can disable a particular alert or change its default notification route. For example, you can configure StackLight LMA to send email notifications about one alert and Salesforce notifications about the other one.

To customize alerts for notifications:

1. Log in to the Salt Master node.
2. Open the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model for editing.
3. Edit the required alert:

    1. Type the alert name.
    2. Set the route parameter to salesforce or email. Alternatively, leave the route parameter empty to disable notifications for the specified alert.

        Example:

        ```
        prometheus:
          server:
            alert:
              AlertName:
                labels:
                  route: salesforce
        ```

4. Apply the Salt formula:

    ```
    salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
    ```

Enable notifications filtering

You can configure StackLight LMA to filter notifications and send the particular ones to specified notification channels. Starting from the maintenance update 2019.2.9, you can also configure notifications subroutes.

Enable notifications filtering

You can enable StackLight LMA to filter notifications and send the particular ones to specified notification channels. For example, you can configure StackLight LMA to send the CRITICAL notifications to email and WARNING notifications to Slack.

To enable notifications filtering:

1. Log in to the Salt Master node.
2. Open the classes/cluster/<cluster_name>/stacklight/server.yml file of the Reclass model for editing.
3. Specify the label and value parameters in the match_re section for a particular receiver.

   Examples:

   To send the CRITICAL notifications to email:

```
parameters:
  prometheus:
    alertmanager:
      enabled: true
      config:
        route:
          routes:
            email:
              receiver: SMTP
              match_re:
                - label: route
                  value: '(.*email.*)'
                - label: severity
                  value: critical
              continue: true
```

   To send the CRITICAL and WARNING notifications to Slack:

```
parameters:
  prometheus:
    alertmanager:
      enabled: true
      config:
        route:
          routes:
            slack:
              receiver: HTTP-slack
              match_re:
                - label: route
                  value: '(.*slack.*)'
                - label: severity
```

```
        value: critical|warning
      continue: true
```

4. Apply the changes:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
```

Configure notifications subroutes

> Note
>
> This feature is available starting from the MCP 2019.2.9 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

You can configure subroutes for the required notifications. For example, you can configure StackLight to send some notifications to email while also selecting the ones from these that have the CRITICAL severity and sending them to PagerDuty as well.

The changes performed within the following procedure are backward compatible with previous MCP releases and will not affect any of the already configured Alertmanager receivers and routes.

To configure notifications subroutes:

1. Open the classes/cluster/<cluster_name>/stacklight/server.yml file of the Reclass model for editing.
2. Configure the the subroute as required:

```
parameters:
  prometheus:
    alertmanager:
      enabled: true
      config:
        route:
          routes:
            <route name>:
              receiver: <receiver name>
                [...]
              routes:  # indicates the subroutes to configure
                <subroute_name>:
                  receiver: '<subroute receiver name>'
                    [...]
               [...]
```

   For example, to send the Apache service alerts to email but filter the ones with CRITICAL severity and send them to PagerDuty:

```
parameters:
  prometheus:
    alertmanager:
      enabled: true
      config:
        route:
```

```
      routes:
        email_alerts:
          receiver: team-X-email
          match_re:
            - label: service
              value: apache
          routes:
            pager_alerts:
              match:
                severity: critical
              receiver: team-X-pager
          continue: true
```

3. Log in to the Salt Master node.
4. Verify the Reclass model:

```
reclass -i
```

   The output should not include any errors.

5. Verify that the changes render properly:

```
salt '*mon*' state.sls prometheus.alertmanager test=True
```

   Example of system response for the configuration provided above:

```
[...]
route:
  receiver: default
[...]
  routes:
  # email_alers
  - receiver: team-X-email
    continue: True
    match:
      service: apache
      routes:
      # pager_alerts
      - receiver: team-X-pager
        match:
          severity: critical
[...]
```

6. Apply the changes:

```
salt '*mon*' state.sls prometheus.alertmanager
```

Configure multiple emails for Alertmanager notifications

By default, you can set only one email for notifications through Alertmanager during the deployment model creation. However, you can configure Alertmanager to send notifications to multiple emails as required.

To configure multiple emails for Alertmanager notifications:

1. Open your Git project repository with Reclass model on the cluster level.
2. In the classes/cluster/cluster_name/stacklight/server.yml file, specify the emails as required, for example, by splitting the alerts by severity as shown below.

   Example:

```
parameters:
  prometheus:
    alertmanager:
      enabled: true
      config:
        route:
          routes:
            email-common:
              receiver: SMTP-common
              continue: true
            email-critical:
              receiver: SMTP-critical
              match_re:
                - label: severity
                  value: critical
              continue: true
        receiver:
          SMTP-common:
            enabled: true
            email_configs:
              smtp_server:
                to: common@email.com
                from: ${_param:alertmanager_notification_email_from}
                auth_username: ${_param:alertmanager_notification_email_username}
                auth_password: ${_param:alertmanager_notification_email_password}
                smarthost: ${_param:alertmanager_notification_email_host}
                require_tls: ${_param:alertmanager_notification_email_require_tls}
                send_resolved: true
          SMTP-critical:
            enabled: true
            email_configs:
              smtp_server:
                to: critical@email.com
                from: ${_param:alertmanager_notification_email_from}
                auth_username: ${_param:alertmanager_notification_email_username}
```

```
            auth_password: ${_param:alertmanager_notification_email_password}
            smarthost: ${_param:alertmanager_notification_email_host}
            require_tls: ${_param:alertmanager_notification_email_require_tls}
            send_resolved: true
```

3. Log in to the Salt Master node.
4. Apply the following state:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.alertmanager -b 1
```

Seealso

Alertmanager documentation

Enable notifications through the Push Notification service

Warning

The DevOps Portal has been deprecated in the Q4`18 MCP release tagged with the 2019.2.0 Build ID.

This section describes how to enable StackLight LMA to send notifications to email, Salesforce or to both notification channels using the Push Notification service of the DevOps Portal.

To enable StackLight LMA to send notifications through the Push Notification Service:

1. Log in to the Salt Master node.
2. Open the classes/cluster/cluster_name/stacklight/server.yml file of the Reclass model for editing.
3. Add the following classes:

```
classes:
 [...]
- system.prometheus.alertmanager.notification.pushkin
- system.prometheus.server.alert.labels_add.route
```

4. Define the following variables:

```
parameters:
  _param:
    alertmanager_notification_pushkin_host: ${_param:haproxy_pushkin_bind_host}
    alertmanager_notification_pushkin_port: ${_param:haproxy_pushkin_bind_port}
    alertmanager_notification_pushkin_host: 172.16.10.101
    alertmanager_notification_pushkin_port: 16666
```

5. Set one or multiple notification channels by using the _param:prometheus_server_alert_label_route parameter. The default value is email, which means that email notifications will be sent.

   Example:

```
parameters:
  _param:
    prometheus_server_alert_label_route: email;salesforce;
```

6. Configure email or Salesforce integration:

©2025, Mirantis Inc.

- For deployments with the DevOps Portal, follow the procedure described in MCP Deployment Guide: Configure Salesforce integration for OSS manually or MCP Deployment Guide: Configure email integration for OSS manually.
- For deployments without the DevOps Portal, modify the classes/cluster/cluster_name/stacklight/server.yml file to configure the Push Notification service, including:
  - Docker stack for the Push Notification service
  - PostgreSQL database for the Push Notification service
  - GlusterFS volumes to synchronize the data between monitoring nodes

  Example:

```
classes:
  [...]
#Glusterfs configuration for Push Notification Service
- system.linux.system.repo.glusterfs
- system.glusterfs.client.cluster
- system.glusterfs.server.cluster
- system.glusterfs.server.volume.postgresql
- system.glusterfs.server.volume.pushkin
- system.glusterfs.client.volume.postgresql
- system.glusterfs.client.volume.pushkin

#Docker stack and network configurations
- system.docker.swarm.stack.pushkin
- system.docker.swarm.stack.postgresql
- system.docker.swarm.network.oss_backend

# Haproxy for Push Notification service
- system.haproxy.proxy.listen.oss.pushkin
- system.haproxy.proxy.listen.oss.postgresql

# Postgresql configuration for Push Notification Service
- system.postgresql.client.pushkin
- system.postgresql.client.sfdc

parameters:
  _param:
    [...]
#Glusterfs configuration for Push Notification Service
    glusterfs_service_host: ${_param:stacklight_monitor_address}
    glusterfs_node01_address: ${_param:stacklight_monitor_node01_address}
    glusterfs_node02_address: ${_param:stacklight_monitor_node02_address}
    glusterfs_node03_address: ${_param:stacklight_monitor_node03_address}

# Postgresql configuration for Push Notification Service
```

```
    postgresql_client_user: 'postgres'
    postgresql_client_password: 'postgrespassword'

# Email configuration for Push Notification Service
    pushkin_smtp_host: smtp.gmail.com
    pushkin_smtp_port: 587
    webhook_from: your_sender@mail.com
    pushkin_email_sender_password: your_sender_password
    webhook_recipients: "recepient1@mail.com,recepient2@mail.com"
    webhook_login_id: 14
    webhook_application_id: 4

# Salesforce configuration for Push Notification Service
    sfdc_auth_url: ''
    sfdc_username: ''
    sfdc_password: ''
    sfdc_consumer_key: ''
    sfdc_consumer_secret: ''
    environment: ''
    environment_id: ''
    sfdc_environment_id: ''
    sfdc_organization_id: ''
    sfdc_sandbox_enabled: False

# Alertmanager configuration for Push Notification Service
    alertmanager_notification_pushkin_host: ${_param:stacklight_monitor_address}
    alertmanager_notification_pushkin_port: 8887
```

> Note
>
> For Salesforce parameters definition, see MCP Deployment Guide: OSS parameters.

7. Apply the Salt formulas:

```
salt -C 'I@glusterfs:server' state.sls glusterfs.server.service
salt -C 'I@glusterfs:server' state.sls glusterfs.server.setup
salt -C 'I@glusterfs:client' state.sls glusterfs.client
salt -C 'I@docker:swarm:role:master and I@prometheus:server' state.sls docker.client
salt -C 'I@postgresql:client' state.sls postgresql.client
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b1
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.alertmanager -b 1
```

Use the Prometheus web UI

The Prometheus web UI enables you to view simple graphs, Prometheus configuration and rules, and the state of the monitoring endpoints. This section describes how to use the Prometheus web UI.

©2025, Mirantis Inc.

Connect to the Prometheus web UI

This section describes how to access the Prometheus web UI.

To connect to the Prometheus web UI:

1. Log in to the Salt Master node.
2. Obtain the Prometheus hostname (IP address), port, and protocol to use as a URL:

   ```
   salt -C 'I@horizon:server' pillar.item nginx:server:site:nginx_proxy_prometheus_server:host
   ```

   Example of system response:

   ```
   prx01.stacklight-dev-2019-2-9.local:
       ----------
       nginx:server:site:nginx_proxy_prometheus_server:host:
           ----------
           name:
               10.13.0.80
           port:
               15010
           protocol:
               https
   ```

3. Enter the URL obtained in the step 2 to a web browser.

> Note
>
> Starting from MCP 2019.2.7, to access the Prometheus web UI from an external network, obtain the credentials by running the following commands from the Salt Master node:
>
> ```
> salt -C 'I@horizon:server' pillar.get _param:nginx_proxy_prometheus_server_user
> salt -C 'I@horizon:server' pillar.get _param:nginx_proxy_prometheus_server_password
> ```
>
> For more details, see Configure authentication for Prometheus and Alermanager.

View graphs and alerts

Using the Prometheus web UI, you can view simple graphs for such metrics as cluster CPU allocation or memory used. Additionally, you can view alerts.

To view graphs:

1. Connect to the Prometheus web UI as described in Connect to the Prometheus web UI.
2. Select the required metric from the drop-down list. Alternatively, enter the metric in the Expression field.
3. Click Execute.
4. Navigate to the Graph tab to view the selected graph. If required, specify the time range.
5. Navigate to the Console tab to view the elements and their values.

> Note
>
> To view multiple graphs, click Add Graph and follow steps 2-4.

> Note
>
> The graphs disappear once you reload the page.

To view alerts:

1. Connect to the Prometheus web UI as described in Connect to the Prometheus web UI.
2. For a list of alerts, navigate to Alerts. Red alerts are the enabled ones.
3. Click on a red alert to view the details on the metric that raised the alert.

> Note
>
> The Active Since column displays the time since an alert has fired but does not display the time prior to the start of the Prometheus service itself. Therefore, if you restart the Prometheus service while having some alerts in the firing state, the Active Since column will display the Prometheus service restart time for these alerts instead of the original value.

View Prometheus settings

Using the Prometheus web UI, you can view the Prometheus settings, rules, monitoring endpoints and their state, and so on.

To view the settings:

1. Connect to the Prometheus web UI as described in Connect to the Prometheus web UI.
2. Select the required item from the Status drop-down list.

Use the Alertmanager web UI

The Alertmanager web UI enables you to view the most recent fired alerts and silence them, as well as view the Alertmanager configuration.

The Alertmanager web UI provides the following functionality:

- The Silences tab displays the existing silences and allows you to define new ones for particular alerts.
- The Alerts tab displays the alerts in the fired state.
- The Status tab displays the Alertmanager status and settings.

To connect to the Alertmanager web UI:

1. Log in to the Salt Master node.
2. Obtain the Alertmanager hostname (IP address), port, and protocol to use as a URL:

   ```
   salt -C 'I@horizon:server' pillar.item nginx:server:site:nginx_proxy_prometheus_alertmanager:host
   ```

   Example of system response:

   ```
   prx01.stacklight-dev-2019-2-9.local:
       ----------
       nginx:server:site:nginx_proxy_prometheus_alertmanager:host:
           ----------
           name:
               10.13.0.80
           port:
               15011
           protocol:
               https
   ```

3. Enter the URL obtained in the step 2 to a web browser.

Note

Starting from MCP 2019.2.7, to access the Alertmanager web UI from an external network, obtain the credentials by running the following commands from the Salt Master node:

```
salt -C 'I@horizon:server' pillar.get _param:nginx_proxy_prometheus_alertmanager_password
salt -C 'I@horizon:server' pillar.get _param:nginx_proxy_prometheus_alertmanager_user
```

For more details, see Configure authentication for Prometheus and Alermanager.

Use the Alerta web UI

The Alerta web UI enables you to view the most recent or watched alerts, as well as group and filter alerts according to your needs.

To connect to the Alerta web UI:

1. Log in to the Salt Master node.
2. Obtain the Alerta hostname (IP address), port, and protocol to use as a URL:

   ```
   salt -C 'I@horizon:server' pillar.item nginx:server:site:nginx_proxy_alerta:host
   ```

   Example of system response:

   ```
   prx01.stacklight-dev-2019-2-9.local:
       ----------
       nginx:server:site:nginx_proxy_alerta:host:
           ----------
           name:
               10.13.0.80
           port:
               15017
           protocol:
               https
   ```

3. Obtain the Alerta administrator password:

   * Starting from the MCP 2019.2.11 maintenance update:

     ```
     salt -C 'I@prometheus:alerta and I@docker:swarm:role:master' pillar.item _param:alerta_admin_password
     ```

     Example of system response:

     ```
     mon01.queens-ovs-stacklight-2k19-2-om.local:
         ----------
         _param:alerta_admin_password:
             zbfNxydJnTwzgA61CRmrTiZ9w4Gmt5Qf
     ```

   * Prior to the MCP 2019.2.11 maintenance update:

     ```
     salt -C 'I@prometheus:alerta and I@docker:swarm:role:master' pillar.item docker:client:stack:monitoring:service:alerta:environment:ADMIN_PASSWORD
     ```

     Example of system response:

     ```
     mon01.stacklight-dev-2019-2-9.local:
         ----------
     ```

```
docker:client:stack:monitoring:service:alerta:environment:ADMIN_PASSWORD:
    miUsex85Idx9YZLw3JBRegcYialYCFMA
```

4. Enter the URL obtained in the step 2 to a web browser.
5. Log in to the Alerta web UI using the default admin@alerta.io user name and the password obtained in the step 3.

Use Grafana

Grafana is a web service that builds and visually represents metric graphs based on time series databases. A collection of predefined Grafana dashboards contains graphs on particular endpoints.

In Prometheus-based StackLight LMA, Grafana has the following special aspects:

- Grafana dashboards do not include annotations based on alerts. To view the alerts, use the Prometheus or Alertmanager web UI. Use Grafana only to visualize the data for a selected period instead of cluster monitoring.
- Grafana dashboards display only the existing information about nodes, disks, interfaces, and so on, according to a particular time frame, which is set to one hour by default. For example, if a node is offline for 30 minutes and the time frame is set to five minutes, the dashboards will not display this node in the drop-down menus. In this case, see the list of alerts in the Prometheus or Alertmanager web UI.

> Warning
>
> Most OpenStack dashboards include the API Availability panel that displays only the OK or DOWN states and does not display the warning states. For warning states, use the Prometheus or Alertmanager web UI.

This section describes how to connect to Grafana, view the available dashboards, and so on.

Connect to Grafana

To access Grafana, use the public IP exposed by the proxy node, the default 8084 port, and the HTTPS protocol.

To connect to Grafana:

1. Log in to the Salt Master node.
2. Obtain the Grafana hostname (IP address), port, and protocol to use as a URL:

   ```
   salt -C 'I@horizon:server' pillar.item nginx:server:site:nginx_proxy_grafana:host
   ```

   Example of system response:

   ```
   prx01.stacklight-dev-2019-2-9.local:
       ----------
       nginx:server:site:nginx_proxy_grafana:host:
           ----------
           name:
               10.13.0.80
           port:
               8084
           protocol:
               https
   ```

3. Obtain the Grafana administrator password:

   * Starting from the MCP 2019.2.11 maintenance update:

     ```
     salt -C 'I@grafana:client and I@docker:swarm:role:master' pillar.item _param:grafana_password
     ```

     Example of system response:

     ```
     mon01.queens-ovs-stacklight-2k19-2-om.local:
         ----------
         _param:grafana_password:
             jrRk4Fa6pxrjq7hbVeqm2drz6Ezgpr4m
     ```

   * Prior to the MCP 2019.2.11 maintenance update:

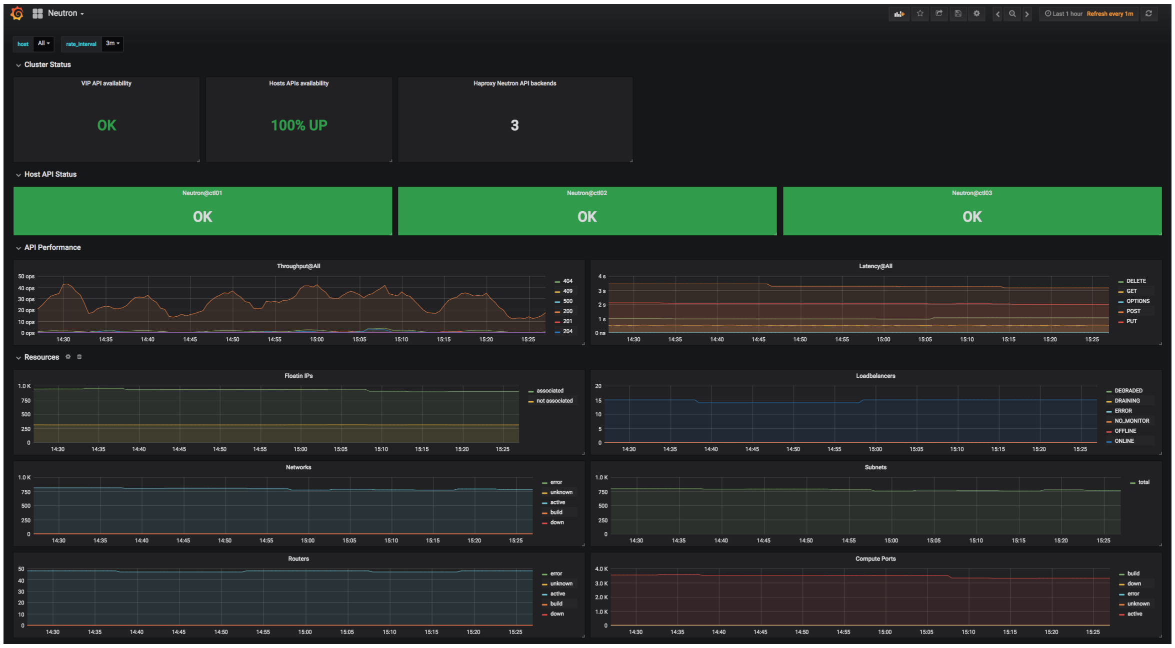
     ```
     salt -C 'I@grafana:client and I@docker:swarm:role:master' pillar.item docker:client:stack:dashboard:service:grafana:environment:GF_SECURITY_ADMIN_PASSWORD
     ```

     Example of system response:

     ```
     mon01.stacklight-dev-2019-2-9.local:
         ----------
         docker:client:stack:dashboard:service:grafana:environment:GF_SECURITY_ADMIN_PASSWORD:
             Z06IsBRBY4h67PCczp24vERqh7fGY3dm
     ```

4. Enter the URL obtained in the step 2 to a web browser.
5. Log in to the Grafana web UI using the default admin user name and the password obtained in the step 3.

Once done, select the required dashboard from the Home drop-down list. For details about the available Grafana dashboards, see View Grafana dashboards. The following is an example of a Grafana dashboard:



View Grafana dashboards

This section describes the available Grafana dashboards.

Main dashboard

The Main dashboard displays the statuses of the services deployed on the cluster. For example, the statuses of OpenStack-related services such as Cinder, Glance, Nova, and so on. Using the Main dashboard, you can quickly determine if any of the services are down or require attention. The Main dashboard consists of the following sections:

* The Middleware section provides the statuses of the services which are not part of OpenStack.
* The OpenStack Control Plane section provides the statuses of OpenStack-related services.

The services may have one of the following statuses:

* UP if the service is up and running
* WARN if the service is down on less than half of the nodes
* CRIT if the service is down on more than half of the nodes
* DOWN if the service is down on all nodes
* No value if the metric is not available
* UNKW if the response value is higher than 1, meaning that the status of the service is unknown

Click a required service to open its dashboard with all metrics. Alternatively, click the arrow sign on a particular service to open its dashboard in a separate tab.

Kubernetes dashboards

This section describes the Kubernetes-related dashboards available in Grafana.

Dashboard	Description
Calico cluster monitoring	Displays the entire Calico cluster usage, including the cluster status, host status, Bird and Felix resources.
Etcd cluster	Provides the cluster status and an overview of the cluster behavior (raft) and the usage of the Etcd instances.
Kubernetes cluster monitoring	Provides metrics related to the entire Kubernetes cluster, including the cluster status, host status, and the resources consumption. **Note** For the deployments with OpenContrail, the Kubernetes cluster monitoring dashboard does not include the Proxy status panel.

OpenStack dashboards

This section describes Grafana dashboards for OpenStack that you can use to explore different time-series facets of your OpenStack environment.

Dashboard	Description
Cinder	Provides a detailed view of the OpenStack Block Storage service metrics, such as the cluster status, host API status, API performance, Cinder services, and resources usage.
Glance	Provides a detailed view of the OpenStack Image service metrics, including the overall health status of the Glance service cluster, host API status, API performance, and various indicators of the virtual resources usage, such as the number of images and snapshots, their status, size, and visibility.
Heat	Provides a detailed view of the OpenStack Orchestration service, including the cluster status, host API status, and API performance metrics.
Ironic Available since 2019.2.6	Note This feature is available starting from the MCP 2019.2.6 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates. Provides a detailed overview of the OpenStack Bare Metal Provisioning service, including the cluster status, host API status, Ironic services, nodes, and drivers metrics.
Keystone	Provides information about the OpenStack Identity service, including the cluster status, host API status, API performance, and the number of active and disabled Keystone users and tenants.
Neutron	Provides a detailed view of the OpenStack networking service, including the cluster status, host API status, API performance, and resources consumption.

Nova dashboards	• Nova - availability zones is available starting from the MCP 2019.2.8 maintenance update and provides the OpenStack availability zones statistics such as the availability zone and hypervisor usage, including the CPU, RAM, and disk usage.
	• Nova - overview provides an overview of the cluster status, host API status, API performance, and the status of Nova services.
	• Nova - hypervisor overview displays the CPU, RAM, and disk usage by particular hosts.
	• Nova - utilization displays the general information on CPU, RAM, and disk usage, as well as the aggregate and hypervisors usage.
	• Nova - instances:
	  • Prior to the MCP 2019.2.7 maintenance update, provides the host usage information, such as the number of running instances and tasks, and the instance usage, such as the CPU, memory usage, and so on.
	  • Starting from the MCP 2019.2.7 maintenance update, provides a more comprehensive information about instances, such as the CPU, RAM, disk throughput usage and allocation and allows sorting the metrics by top instances.
	• Nova - users and Nova - tenants dashboards are available starting from the MCP 2019.2.7 maintenance update and provide information about CPU, RAM, disk throughput, IOPS, and space usage and allocation and allow sorting the metrics by top users or tenants.

Octavia	Provides a detailed view of the OpenStack Octavia service that is coupled with the Neutron LBaaS to display the load balancing state of your OpenStack environment. The dashboard displays the Octavia API availability and the number of resources for Octavia/LBaaS, including the load balancers, listeners, pools, members, and monitors.
OpenStack overview	Provides a detailed view of your MCP OpenStack environment, including the cloud usage metrics, API errors, and allocations per aggregate.
OpenStack Tenants Removed since 2019.2.7	Note Starting from the MCP 2019.2.7 update, the dashboard has been removed in favor of the Nova - users and Nova - tenants dashboards that are more informative. Provides a detailed view of Nova instances usage per tenants and users, including the CPU and memory usage, disks I/O, and network RX/TX.

OpenContrail dashboards

This section describes the dashboards that provide metrics for the OpenContrail services deployed on the platform.

<table>
<tr><th>Dashboard</th><th>Description</th></tr>
<tr><td>Cassandra</td><td>Provides metrics about the Cassandra service, including the number of Cassandra endpoints, the number of native and thrift clients, information about the clients requests, the compaction engine rates, the storage metrics, as well as the heap memory and memory pool usage of the Cassandra JVM.</td></tr>
<tr><td>OpenContrail Controller</td><td>Displays the overall status of the OpenContrail APIs, the number of OpenContrail API sessions, the host status, and the number of BGP and vRouters Extensible Messaging and Presence Protocol (XMPP) sessions that are in the up and down state.

Note
For OpenContrail v3.2, the dashboard additionally includes the number of OpenContrail Discovery API servers.</td></tr>
<tr><td>OpenContrail vRouter</td><td>Displays the vRouter statistics, such as the state and total number of vRouters, service status, the number of vRouters Extensible Messaging and Presence Protocol (XMPP) and vRouters Link-Local Services (LLS) sessions, as well as the number of active and aged vRouter flows and vRouter errors.

Note
For OpenContrail v4.x, the dashboard does not include the DNS-XMPP metrics.</td></tr>
<tr><td>Zookeeper</td><td>Provides a detailed view of the ZooKeeper service, including the overall status of the cluster and the statistics metrics, such as the latency, packets, ephemerals, approximate data size, alive connections, and so on.</td></tr>
</table>

KPI dashboards

This section describes the Key Performance Indicator (KPI) dashboards that provide an overview of the infrastructure stability.

Dashboard	Description
KPI Downtime	Provides an overview of the instances availability, such as average uptime of VMs, the number of available VMs, the VMs uptime percentage, instances in the ERROR state, and instances in the ACTIVE state but not responding.
KPI Provisioning	Provides the percentage of instance provisioning failures based on the compute.instance.create.start, compute.instance.create.end, and compute.instance.create.error Nova notifications.

StackLight LMA components dashboards

This section describes the dashboards that provide metrics for the StackLight LMA components.

Dashboard	Description
Alertmanager	Provides performance metrics for the Prometheus Alertmanager service, including the overall health status of the service, the number of firing and resolved alerts received for various time periods, the rate of successful and failed notifications, and the resources consumption.
Elasticsearch	Provides information about the overall health status of the Elasticsearch cluster, including the state of the shards and various metrics of resources consumption.
Grafana	Provides performance metrics for the Grafana service, including the total number of Grafana entities, CPU and memory usage.
InfluxDB Deprecated	Warning InfluxDB, including InfluxDB Relay and remote storage adapter, is deprecated in the Q4`18 MCP release and will be removed in the next release. Provides statistics about the InfluxDB processes running in the InfluxDB cluster including various metrics of resources consumption.
InfluxDB Relay Deprecated	Warning InfluxDB, including InfluxDB Relay and remote storage adapter, is deprecated in the Q4`18 MCP release and will be removed in the next release. Provides the overall health status of the InfluxDB Relay cluster and metrics for the InfluxDB Relay instances and backends.
Kibana	Provides performance metrics for the Kibana service, including the service status and resources consumption, such as the number of threads, CPU, and memory usage.
Prometheus Performances	Provides metrics for the Prometheus component itself, such as the sample ingestion rate and system usage statistics per server. This enables you to check the availability and performance behavior of the Prometheus servers.
Prometheus Relay	Provides metrics for the Prometheus Relay component itself, including the service status and resources consumption.

Prometheus Stats	Provides statistics about the Prometheus server and includes the overall status of the Prometheus service, the uptime of the Prometheus service, the chunks number of the local storage memory, target scrapes, queries duration, and so on.
Pushgateway	Provides performance metrics for the Prometheus Pushgateway service, including the overall health status of the service, the rate of samples received for various time periods, and the resources consumption.
Remote Storage Adapter Deprecated	Warning InfluxDB, including InfluxDB Relay and remote storage adapter, is deprecated in the Q4`18 MCP release and will be removed in the next release. Provides the overall status of the remote storage adapter service, including the number of sent, received, and ignored samples and the resources consumption.

Ceph dashboards

The Ceph dashboards provide a detailed view of the Ceph cluster, hosts, OSDs, RADOS Gateway instances, and pools.

Dashboard	Description
Ceph Cluster	Provides metrics related to the Ceph cluster, including the overall health status of the Ceph service cluster, capacity, latency, and recovery metrics.
Ceph Hosts Overview	Provides an overview of the host-related metrics, such as the number of monitors, OSD hosts, average usage of resources across the cluster, network and hosts load.
Ceph OSD device details	Provides metrics related to the Ceph OSD and physical device performance.
Ceph OSD Overview	Provides metrics related to Ceph OSDs, including the OSD read and write latencies and the distribution of PGs per OSD. Note Starting from the MCP 2019.2.5 maintenance update, the Distribution of PGs per OSD panel displays the data in bars instead of lines.
Ceph Pools Overview	Provides metrics for Ceph pools, including the client IOPS and throughput by pool and pools capacity usage.
Ceph RGW Instance Details	Provides detailed graphs on the RADOS Gateway host.
Ceph RGW Overview	Provides metrics related to the RADOS Gateway instances, including the latencies, requests, and bandwidth.
Ceph RBD Overview Available since 2019.2.10. TechPreview.	Provides metrics related to RADOS Block Device images, including the throughput, latencies, and IOPS. Note Optional, disabled by default. Available as technical preview starting from the MCP 2019.2.10 maintenance update and requires Ceph Nautilus. For details, see Enable RBD monitoring.

Operating system dashboards

This section describes the dashboards that provide a detailed view of the operating system.

Dashboard	Description
Bond	Provides the status of Linux bond interfaces, such as the count of bond slave failures, bond status, and bond slave status.
NTP	Provides metrics about the Network Time Protocol(NTP) services on hosts.
System disk I/O	Provides a detailed overview of the file system metrics, including the used and free space and inodes, and series metrics, including the latency, I/O wait, and so on.
System networking	Displays the network-related metrics, such as the throughput, packets, errors, dropped packets on a selected interface, and information about network data processing.
System overview	Provides a detailed overview of the operating system metrics, including the overall status of the system, memory available, swap and CPU usage, and so on.

Support services dashboards

This section describes the dashboards for support services, such as RabbitMQ, HAProxy, MySQL, and so on.

Dashboard	Description
Apache	Displays the overall status of the Apache cluster and provides performance metrics for the Apache service, such as the number of requests, bytes transmitted, the number of connections, workers states, and current workers in the idle state.
Docker	Provides the Docker cluster status, the status of Docker containers, metrics about Docker images, as well as the performance metrics of the Docker host, such as the number of threads, CPU, and disk I/O.
GlusterFS	Provides the operational status of the GlusterFS cluster and service, as well as the usage reporting of the shared volumes.
HAProxy	Provides the overall status of the HAProxy cluster and various metrics related to the front-end and back-end servers.
Jenkins	Provides general information about the Jenkins service, including the number of online and total Jenkins nodes, the queue size, JVM free memory and uptime, as well as various metrics on executors, jobs, and resources usage.
Keepalived	Provides metrics about the Keepalived service, such as the current status of the Keepalived instances, status for a specific period of time, process responsiveness, and the state of the Virtual Router Redundancy Protocol (VRRP) of Keepalived.
Memcached	Provides the overall status of the Memcached service, including the metrics on the Memcached servers, memory usage, operations and network metrics.
MySQL	Provides detailed information about the MySQL cluster status and service usage, including the cluster size, the average size of the receive and send queries, network I/O, locks, threads, queries, and so on.
Nginx	Provides metrics about the NGINX service, such as the overall status of the NGINX cluster and information about NGINX requests and connections.

RabbitMQ	Provides general information about the RabbitMQ cluster, including the host status, cluster statistics, and resources consumption. Note Starting from the MCP 2019.2.3 maintenance update, the Cluster stats section displays the Queued messages and Message rates panels instead of Messages and Cluster stats.

Hide nodes from dashboards

When you remove a node from the environment, it is still displayed in the host drop-down lists of the Grafana dashboards.

To hide a node from the list:

1. Connect to Grafana.
2. Navigate to the required dashboard.
3. Click the gear icon at the top left corner and select Templating.
4. In the Variables tab, click Edit.
5. Edit the Regex text box in the Query Options section. For example:
    - To hide cfg01, add the following text:

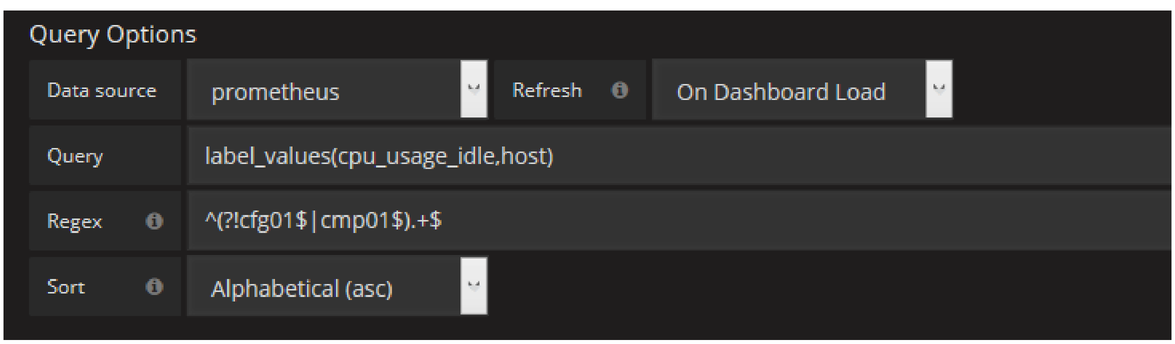
      ```
      ^(?!cfg01$).+$
      ```

    - To hide more than one node, add more conditions:

      ```
      ^(?!cfg01$|cmp01$).+$
      ```

6. Click Update to apply the changes.

Example:



Add the Gnocchi data source to Grafana

By default, Grafana uses Prometheus as a data source to provide graphs and charts. If your OpenStack version is Pike or newer and you have deployed Tenant Telemetry as described in MCP Deployment Guide: Deploy Tenant Telemetry, you can also add Gnocchi as the data source for Grafana. This allows StackLight LMA gather the data both from Prometheus and from Gnocchi, the Tenant Telemetry database, for further processing and displaying through Grafana dashboards.

Prerequisites

Before you add Gnocchi as a data source to Grafana, verify that your environment meets the following requirements.

- Due to a limitation in Grafana, Gnocchi and Keystone must be accessible through the same HTTP vhost.
- Keystone must return public endpoints for Gnocchi and for its own services.
- The public endpoint must be resolvable from Grafana and your browser.
- Cross-Origin Resource Sharing (CORS) must be enabled in Gnocchi and Keystone to allow requests from Grafana. Additionally, CORS must be enabled in your browser.

To satisfy the requirements, complete the following prerequisite steps:

1. Enable CORS for the Gnocchi and Keystone servers:
    1. Open your project Git repository with the Reclass model on the cluster level.
    2. In the openstack/telemetry.yml file, add the following configuration for Gnocchi, specifying the vhost FQDN as allowed_origin:

       ```
       gnocchi:
         server:
           cors:
             allowed_origin: "http://example.com"
       ```

    3. In the openstack/control.yml file, add the following configuration for Keystone, specifying the vhost FQDN as allowed_origin:

       ```
       keystone:
         server:
           cors:
             allowed_origin: "http://example.com"
       ```

2. Set the endpoints for Keystone and Gnocchi:
    1. In the openstack/init.yml file, specify the following parameters:

       ```
       _params:
           gnocchi_public_host: example.com
           gnocchi_public_port: 80
           gnocchi_public_path: '/metric'
           keystone_public_path: '/identity'
           keystone_public_address: example.com
           keystone_public_port: 80
       ```

3. Log in to the Salt Master node.
4. Apply the following states:

```
salt 'ctl*' state.sls keystone
salt -C 'I@keystone:client' state.sls keystone.client
salt 'mdb*' state.sls gnocchi
```

5. Configure the same HTTP host for Gnocchi and Keystone:

    1. Create a new file /etc/nginx/sites-enabled/nginx_proxy_gnocchi.conf for the vhost configuration for Gnocchi and specify the following parameters:

    ```
    server {
      listen 80 default_server;
      server_name _;

      location ~ ^/metric/?(.*) {
        proxy_pass     http://172.16.10.250:8041/$1;

        proxy_pass_request_body on;
        proxy_set_header       Host          $host;
        proxy_set_header       X-Real-IP     $remote_addr;
        proxy_set_header       X-Forwarded-For $proxy_add_x_forwarded_for;
      }

      location ~ ^/identity/?(.*) {
        proxy_pass http://172.16.10.254:5000/$1;

        proxy_pass_request_body on;
        proxy_set_header       Host          $host;
        proxy_set_header       X-Real-IP     $remote_addr;
        proxy_set_header       X-Forwarded-For $proxy_add_x_forwarded_for;
      }
    }
    ```

    2. Verify the new configuration and restart NGINX:

    ```
    salt -C 'I@nginx:server' cmd.run 'nginx -t && nginx -s reload'
    ```

6. Make the HTTP host resolvable from the browser through an internal DNS or a static configuration on hosts, for example, /etc/hosts in Linux.

    For a static configuration, run the following command in your console to add a record in /etc/hosts, where example.com is the vhost FQDN:

    ```
    echo "<grafana_public_ip_address> example.com" >> /etc/hosts
    ```

7. Enable CORS requests in your browser or install a corresponding browser extension. For example, use the Chrome extension or Firefox extension.

Once done, proceed to Add the Gnocchi data source.

Add the Gnocchi data source

Once you perform the steps described in Prerequisites, perform the steps below to add the Gnocchi data source to Grafana.

To add the Gnocchi data source:

1. Open your Git project repository with Reclass model on the cluster level.
2. In the stacklight/client.yml file, add the following class:

```
- system.grafana.client.datasource.gnocchi
```

3. In the stacklight/server.yml file, specify the following parameters, where example.com is the vhost FQDN:

```
docker_image_grafana: docker-prod-local.artifactory.mirantis.com/openstack-docker/grafana:stable
grafana_gnocchi_address: example.com
```

4. Log in to the Salt Master node.
5. Apply the following states:

```
salt 'mon*' state.sls docker.client
salt 'mon*' state.sls grafana.client
```

6. If the HTTP vhost name is not resolvable from the Docker container, run the following command from the host that runs the container to add a record in /etc/hosts of Grafana containers:

```
GRAFANA_CONTAINER_ID=`docker ps | grep grafana | cut -d" " -f1`
HOST_IP=`hostname -I | cut -d " " -f1`
docker exec -u root -it $GRAFANA_CONTAINER_ID bash -c "echo \"$HOST_IP example.com\" >> /etc/hosts"
```

7. In the Grafana web UI, navigate to Configuration > Data Sources.
8. Open the Gnocchi data source and click Save & Test.

Use Kibana

Kibana is used for log and time series analytics. Kibana provides real time visualization of the data stored in Elasticsearch and allows you to diagnose issues. This section describes how to connect to Kibana and use its dashboards.

Connect to Kibana

To access Kibana, use the public VIP exposed by the proxy node, the default 5601 port, and the HTTPS protocol.

To connect to Kibana:

1. Log in to the Salt Master node.
2. Obtain the Kibana hostname (IP address), port, and protocol to use as a URL:

   ```
   salt -C 'I@horizon:server' pillar.item nginx:server:site:nginx_proxy_kibana:host
   ```

   Example of system response:

   ```
   prx01.stacklight-dev-2019-2-9.local:
   ----------
   nginx:server:site:nginx_proxy_kibana:host:
       ----------
       name:
           10.13.0.80
       port:
           5601
       protocol:
           https
   ```

3. Enter the URL obtained in the step 2 to a web browser.

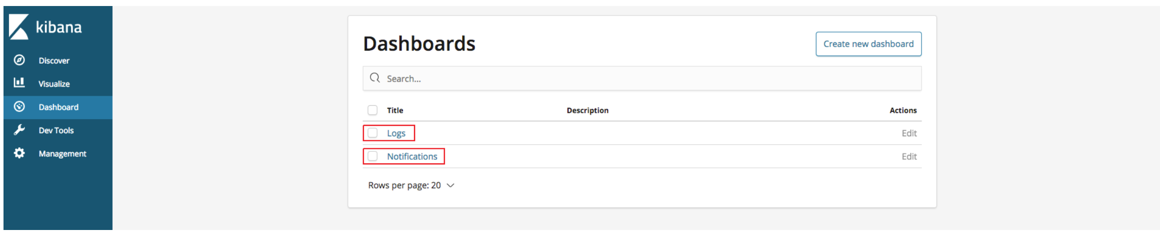
No credentials are required to connect to Kibana.

Manage Kibana dashboards

Kibana contains the following built-in dashboards:

- The Logs analytics dashboard that is used to visualize and search the logs.
- The Notifications analytics dashboard that is used to visualize and search the notifications. This dashboard is available if you enable the feature in the Collector settings.
- The Audit analytics dashboard that is used to visualize and search for the OpenStack CADF notifications.

To switch from one dashboard to another, click Dashboard and select the required one as shown in the screen capture below:



Each dashboard provides a single pane of glass for visualizing and searching for the logs and notifications of your deployment.

The Kibana dashboard for logs is divided into several sections:

1. A time-picker control to select the required time period and refresh frequency.
2. A text box to enter search queries.
3. The logs analytics with six different panels showing the following stack graphs:
    1. All logs per source
    2. All logs per severity
    3. All logs for top 10 sources
    4. All logs for top 10 programs
    5. All logs for top 10 hosts
    6. The number of logs per severity

   The table of log messages is sorted in the reverse chronological order.

Use Kibana filters and queries

Filters and queries have similar syntax but are used for different purposes:

- Filters are used to restrict what is displayed in the Kibana dashboard.
- Queries are used for free-text search.

You can combine multiple queries and compare the results. You can also further filter the log messages. For example, to select the Hostname filter:

1. Expand a log entry.
2. Select the Hostname field by clicking on the magnifying glass icon as follows:

   This will apply a new filter in the Kibana dashboard:

Filtering works for any field that has been indexed for the log entries that are present in the Kibana dashboard.

Filters and queries can also use wildcards that can be combined with the field names like in Logger.keyword: <name>. For example, to display only the Nova logs, enter Logger.keyword:openstack.nova in the query text box as follows:


Dashboard / Logs
Full screen
Share
Clone
Edit
Auto-refresh
Last 1 hour
Logger.keyword:openstack.nova
Options
Refresh
Hostname.keyword: "ctl02"
Add a filter
Actions
LOG MESSAGES OVER TIME PER SOURCE
openstack.nova
Count
Timestamp per minute
LOG MESSAGES OVER TIME PER SEVERITY
INFO
Count
Timestamp per minute
TOP 10 SOURCES
INFO
Count
openstack.nova
Number of log messages per Logger
TOP 10 PROGRAMS
INFO
Count
nova-api
nova
nova-placement-wsgi
nova-conductor
nova-scheduler
nova-consoleauth
Number of log messages per programname
NUMBER OF LOG MESSAGES PER SEVERITY
TOP 10 HOSTS

StackLight LMA alerts

This section provides a detailed overview of the available StackLight LMA alerts including their customization capabilities and troubleshooting recommendations, as well as describes the alerts that require post-deployment configuration. This section also provides an instruction on how to generate the list of alerts for a particular MCP deployment.

Available StackLight LMA alerts

This section describes the available StackLight LMA alerts grouped by services.

Core services

This section describes the alerts available for the core services.

Apache

This section describes the alerts for the Apache service.

- ApacheServiceDown
- ApacheServiceOutage
- ApacheWorkersAbsent

ApacheServiceDown

Severity	Minor
Summary	The Apache service on the {{ $labels.host }} node is down.
Raise condition	apache_up != 1
Description	Raises when the Apache service on a particular host does not respond to the Telegraf service, typically meaning that the Apache service is down or misconfigured on that host. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the Apache service status by running systemctl status apache2 on the affected node. • Inspect the Telegraf logs by running journalctl -u telegraf on the affected node.
Tuning	Not required

ApacheServiceOutage

Severity	Critical
Summary	All Apache services within the {{ $labels.cluster }} cluster are down.
Raise condition	count(label_replace(apache_up, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster) == count(label_replace(apache_up == 0, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster)
Description	Raises when all Apache services across the cluster do not respond to the Telegraf plugin, typically indicating some global deployment or configuration issues. The cluster label in the raised alert is a set of nodes with the same host name prefix, for example, mon, cid.

Troubleshooting	• Verify the Apache service status by running systemctl status apache2 on the affected node. • Inspect the Telegraf logs by running journalctl -u telegraf on the affected node.
Tuning	Not required

ApacheWorkersAbsent

Severity	Minor
Summary	The Apache service on the {{ $labels.host }} node has no available workers for 2 minutes.
Raise condition	apache_IdleWorkers == 0
Description	Raises when the Apache service on a particular host has no free idle workers.
Troubleshooting	Increase the MaxClients Apache parameter value using the apache.server.mpm.servers.max_requests pillar.
Tuning	Not required

Bond interfaces

This section describes the alerts for bond interfaces.

- BondInterfaceDown
- BondInterfaceSlaveDown
- BondInterfaceSlaveDownMajor
- BondInterfaceSingleSlave

BondInterfaceDown

Severity	Critical
Summary	The {{ $labels.bond }} bond interface on the {{ $labels.host }} node has all ifaces down.
Raise condition	bond_status < 1
Description	Raises when the bond network interface and all slave interfaces are down, typically indicating network interface misconfiguration or issues with slave interfaces. The host and bond labels in the raised alert contain the host name of the affected node and the affected interface. For details on the bond interface, see the /proc/net/bonding/{bond_name} file on the affected node.
Tuning	Not required

BondInterfaceSlaveDown

Severity	Warning
Summary	The {{ $labels.bond }} bond interface slave {{ $labels.interface }} on the {{ $labels.host }} node is down.
Raise condition	bond_slave_status < 1
Description	Raises when the slave network interface of a bond interface is down, typically indicating a network misconfiguration. The host, bond, and interface labels in the raised alert contain the host name of the affected node, bond interface name, and the name of the interface slave. For details on the bond interface, see the /proc/net/bonding/{bond_name} file on the affected node.

Tuning	Not required

BondInterfaceSlaveDownMajor

Severity	Major
Summary	More than 50% of {{ $labels.bond }} bond interface slaves on the {{ $labels.host }} node are down.
Raise condition	sum(bond_slave_status) by (bond,host) <= 0.5 * count(bond_slave_status) by (bond,host)
Description	Raises when more than 50% of slave network interfaces of a bond interface are down, typically indicating a network misconfiguration. The host, bond, and interface labels in the raised alert contain the host name of the affected node, bond interface name, and the name of the interface slave. For details on the bond interface, see the /proc/net/bonding/{bond_name} file on the affected node.
Tuning	Not required

BondInterfaceSingleSlave

Available starting from the 2019.2.4 maintenance update

Severity	Major
Summary	The {{ $labels.bond }} bond interface on the {{ $labels.host }} node has only one slave.
Raise condition	count(bond_slave_status) by (bond,host) == 1
Description	Raises when the bond interface has only one slave, typically indicating a network misconfiguration since the bond interface must have at least two slave interfaces. The host, bond, and interface labels in the raised alert contain the host name of the affected node, bond interface name, and the name of the interface slave. For details on the bond interface, see the /proc/net/bonding/{bond_name} file on the affected node.
Tuning	Not required

©2025, Mirantis Inc.

Docker

This section describes the alerts for the Docker service.

- DockerdProcessDown
- DockerServiceOutage
- DockerService {{ camel_case_name }} ReplicasDownMinor
- DockerService {{ camel_case_name }} ReplicasDownMajor
- DockerService {{ camel_case_name }} Outage
- DockerdServiceReplicaFlapping

DockerdProcessDown

Severity	Minor
Summary	The dockerd process on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="dockerd"} == 0
Description	Raises when Telegraf cannot find running dockerd processes on a host. The host label in the raised alert contains the name of the affected node.
Troubleshooting	• Verify Docker status by running systemctl status docker on the affected node. • Inspect Docker logs using journalctl -u docker.
Tuning	Not required

DockerServiceOutage

Severity	Critical
Summary	All dockerd processes within the {{ $labels.cluster }} cluster are down.
Raise condition	count(label_replace(procstat_running{process_name="dockerd"}, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster) == count(label_replace(procstat_running{process_name="dockerd"} == 0, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster)

Description	Raises when Telegraf cannot find running dockerd processes on all hosts of a cluster. The cluster label in the raised alert is a set of nodes with the same host name prefix, for example, mon or cid.
Troubleshooting	• Inspect the DockerdProcessDown alerts for the host names of the affected nodes. • Verify the Docker service status using service docker status. • Inspect Docker logs using journalctl -u docker.
Tuning	Not required

DockerService {{ camel_case_name }} ReplicasDownMinor

Severity	Minor
Summary	More than 30% of the Docker Swarm {{ full_service_name }} service replicas are down for 2 minutes.
Raise condition	• In 2019.2.8 and prior: {{ service.deploy.replicas }} - min(docker_swarm_tasks_running{{ '{' + label_selector + '}' }}) >= {{ service.deploy.replicas }} * {{ monitoring.replicas_failed_warning_threshold_percent }} • In 2019.2.9 and newer: min(docker_swarm_tasks_running{{ '{' + label_selector + '}' }}) / min(docker_swarm_tasks_desired{{ '{' + label_selector + '}' }}) <= {{ 1 - monitoring.replicas_failed_warning_threshold_percent }}
Description	A generated set of alerts for each Docker Swarm service. Applicable only for the replicated Docker Swarm services. Raises when the cluster has more than 30% of unavailable replicas. The service_name label in the raised alert contains the Docker service name.
Troubleshooting	Run docker service ps <service_name> on any node of the affected cluster to verify the Docker service.
Tuning	Not required

DockerService {{ camel_case_name }} ReplicasDownMajor

Severity	Major
Summary	More than 60% of the Docker Swarm {{ full_service_name }} service replicas are down for 2 minutes.

Raise condition	• In 2019.2.8 and prior: {{ service.deploy.replicas }} - min(docker_swarm_tasks_running{{ '{' + label_selector + '}' }}) >= {{ service.deploy.replicas }} * {{ monitoring.replicas_failed_critical_threshold_percent }} • In 2019.2.9 and newer: min(docker_swarm_tasks_running{{ '{' + label_selector + '}' }}) / min(docker_swarm_tasks_desired{{ '{' + label_selector + '}' }}) <= {{ 1 - monitoring.replicas_failed_critical_threshold_percent }}
Description	A generated set of alerts for each Docker Swarm service. Applicable only for the replicated Docker Swarm services. Raises when the cluster has more than 60% of unavailable service replicas. The service_name label in the raised alert contains the Docker service name.
Troubleshooting	Run docker service ps <service_name> on any node of the affected cluster to verify the Docker service.
Tuning	Not required

DockerService {{ camel_case_name }} Outage

Severity	Critical
Summary	All Docker Swarm {{ full_service_name }} replicas are down for 2 minutes.
Raise condition	• In 2019.2.5 and prior: docker_swarm_tasks_running{{ '{' + label_selector + '}' }} == 0 or absent (docker_swarm_tasks_running{{ '{' + label_selector + '}' }}) == 1 • In 2019.2.6-2019.2.8: docker_swarm_tasks_desired{{ '{' + label_selector + '}' }} > 0 and (docker_swarm_tasks_running{{ '{' + label_selector + '}' }} == 0 or absent (docker_swarm_tasks_running{{ '{' + label_selector + '}' }}) == 1) • In 2019.2.9 and newer: min(docker_swarm_tasks_desired{{ '{' + label_selector + '}' }}) > 0 and (min(docker_swarm_tasks_running{{ '{' + label_selector + '}' }}) == 0 or absent(docker_swarm_tasks_running{{ '{' + label_selector + '}' }}) == 1)
Description	A generated set of alerts for each Docker Swarm service. Applicable only for the replicated Docker Swarm services. Raises when the cluster has no available service replicas. The service_name label in the raised alert contains the Docker service name.

©2025, Mirantis Inc.

Troubleshooting	Run docker service ps <service_name> on any node of the affected cluster to verify the Docker service.
Tuning	Not required

DockerdServiceReplicaFlapping

Available starting from the 2019.2.6 maintenance update

Severity	Critical
Summary	The Docker Swarm {{ $labels.service_name }} service replica is flapping for 15 minutes.
Raise condition	sum(changes(docker_swarm_tasks_running[10m])) by (service_name) > 0
Description	Raises when the container with the service cannot start properly in the Docker Swarm cluster and stops after the start, meaning that the service may be unavailable. However, the service unavailability alert may not fire because the container is being constantly restarted.
Troubleshooting	Inspect the failed container logs using docker logs <container_id>.
Tuning	Not required

Galera

This section describes the alerts for the Galera cluster.

- GaleraServiceDown
- GaleraServiceOutage
- GaleraNodeNotReady
- GaleraNodeNotConnected

GaleraServiceDown

Severity	Minor
Summary	The Galera service on the {{ $labels.host }} node is down.
Raise condition	mysql_up != 1
Description	Raises when MySQL on a host does not respond to Telegraf, typically indicating that MySQL is not running on that node. The host label in the raised alert contains the name of the affected node.
Troubleshooting	• Verify the MySQL status on the affected node using service mysql status. • If MySQL is up and running, inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

GaleraServiceOutage

Severity	Critical
Summary	All Galera services within the {{ $labels.cluster }} cluster are down.
Raise condition	count(label_replace(mysql_up, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster) == count(label_replace(mysql_up == 0, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster)
Description	Raises when all MySQL services across the cluster do not respond to Telegraf, typically indicating deployment or configuration issues.

Troubleshooting	• Verify the MySQL status on any Galera node using service mysql status. • If MySQL is up and running, inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

GaleraNodeNotReady

Severity	Major
Summary	The Galera service on the {{ $labels.host }} node is not ready to serve queries for 1 minute.
Raise condition	mysql_wsrep_ready != 1
Description	Raises when the Write Set Replication (WSREP) in the MySQL service is not ready, typically indicating that the MySQL process is running but the WSREP is not in the ready state, meaning that the node is not a part of the Galera cluster.
Troubleshooting	Inspect the MySQL logs on the affected node using journalctl -u mysql.
Tuning	Not required

GaleraNodeNotConnected

Severity	Major
Summary	The Galera service on the {{ $labels.host }} node is not connected to the cluster for 1 minute.
Raise condition	mysql_wsrep_connected != 1
Description	Raises when the Write Set Replication (WSREP) in the MySQL service is not in the connected state, typically indicating that the MySQL process is running but the WSREP did not establish the required connections with other nodes within the Galera cluster due to the WSREP misconfiguration in MySQL or a network issue.
Troubleshooting	• Inspect the MySQL logs on the affected node using journalctl -u mysql. • Verify if proper hosts are used in Galera.

Tunin g	Not required

GlusterFS

This section describes the alerts for the GlusterFS service.

- GlusterfsServiceMinor
- GlusterfsServiceOutage
- GlusterfsInodesUsedMinor
- GlusterfsInodesUsedMajor
- GlusterfsSpaceUsedMinor
- GlusterfsSpaceUsedMajor
- GlusterfsMountMissing

GlusterfsServiceMinor

Severity	Minor
Summary	The GlusterFS service on the {{ $labels.host }} host is down.
Raise condition	procstat_running{process_name="glusterd"} < 1
Description	Raises when Telegraf cannot find running glusterd processes on the kvm hosts.
Troubleshooting	• Verify the GlusterFS status using systemctl status glusterfs-server. • Inspect GlusterFS logs in the /var/log/glusterfs/ directory.
Tuning	Not required

GlusterfsServiceOutage

Severity	Critical
Summary	All GlusterFS services are down.
Raise condition	glusterfs_up != 1

Description	Raises when the Telegraf service cannot connect or gather metrics from the GlusterFS service, typically meaning the GlusterFS, Telegraf monitoring_remote_agent service, or network issues.
Troubleshooting	• Inspect the Telegraf monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on any mon node. • Verify the GlusterFS status using systemctl status glusterfs-server. • Inspect GlusterFS logs in the /var/log/glusterfs/ directory.
Tuning	Not required

GlusterfsInodesUsedMinor

Severity	Minor
Summary	More than 80% of GlusterFS {{ $labels.volume }} volume inodes are used for 2 minutes.
Raise condition	glusterfs_inodes_percent_used >= {{ monitoring.inodes_percent_used_minor_threshold_percent*100 }} and glusterfs_inodes_percent_used < {{ monitoring.inodes_percent_used_major_threshold_percent*100 }}
Description	Raises when GlusterFS uses more than 80% and less than 90% of available inodes. The volume label in the raised alert contains the affected GlusterFS volume.
Troubleshooting	• Verify the number of objects stored in GlusterFS. • If possible, increase the number of inodes.

Tuning	Typically, you should not change the default value. If the alert is constantly firing, verify the available inodes on the GlusterFS nodes and adjust the threshold according to the number of available inodes. In the Prometheus Web UI, use the raise condition query for a longer period of time to define the best threshold.
	For example, to change the threshold to the 90 - 95% interval:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        GlusterfsInodesUsedMinor:`
	`          if: >-`
	`            glusterfs_inodes_percent_used >= 90 and \`
	`            glusterfs_inodes_percent_used < 95`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

GlusterfsInodesUsedMajor

Severity	Major
Summary	{{ $value }}% of GlusterFS {{ $labels.volume }} volume inodes are used for 2 minutes.
Raise condition	glusterfs_inodes_percent_used >= {{ monitoring.inodes_percent_used_major_threshold_percent*100 }}

©2025, Mirantis Inc.

<table>
<tr><td>Description</td><td>Raises when GlusterFS uses more than 90% of available inodes. The volume label in the raised alert contains the affected GlusterFS volume.</td></tr>
<tr><td>Troubleshooting</td><td>

- Verify the number of objects stored in GlusterFS.
- If possible, increase the number of inodes.

</td></tr>
<tr><td>Tuning</td><td>

Typically, you should not change the default value. If the alert is constantly firing, verify the available inodes on the GlusterFS nodes and adjust the threshold according to the number of available inodes. In the Prometheus Web UI, use the raise condition query for a longer period of time to define the best threshold.
For example, to change the threshold to 95%:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           GlusterfsInodesUsedMajor:
             if: >-
               glusterfs_inodes_percent_used >= 95
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

GlusterfsSpaceUsedMinor

Severity	Minor

Summary	{{ $value }}% of GlusterFS {{ $labels.volume }} volume disk space is used for 2 minutes.
Raise condition	glusterfs_space_percent_used >= {{ monitoring.space_percent_used_minor_threshold_percent*100 }} and glusterfs_space_percent_used < {{ monitoring.space_percent_used_major_threshold_percent*100 }}
Description	Raises when GlusterFS uses more than 80% and less than 90% of available space. The volume label in the raised alert contains the affected GlusterFS volume.
Troubleshooting	• Inspect the data stored in GlusterFS. • Increase the GlusterFS capacity.

Tuning	Typically, you should not change the default value. If the alert is constantly firing, verify the available space on the GlusterFS nodes and adjust the threshold accordingly. In the Prometheus Web UI, use the raise condition query for a longer period of time to define the best threshold.
	For example, to change the threshold to the 90-95% interval:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        GlusterfsSpaceUsedMinor:`
	`          if: >-`
	`            glusterfs_space_percent_used >= 90 and \`
	`            glusterfs_space_percent_used < 95`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

GlusterfsSpaceUsedMajor

Severity	Minor
Summary	{{ $value }}% of GlusterFS {{ $labels.volume }} volume disk space is used for 2 minutes.
Raise condition	glusterfs_space_percent_used >= {{ monitoring.space_percent_used_major_threshold_percent*100 }}

Description	Raises when GlusterFS uses more than 90% of available space. The volume label in the raised alert contains the affected GlusterFS volume.
Troubleshooting	• Inspect the data stored in GlusterFS. • Increase the GlusterFS capacity.

Tuning	Typically, you should not change the default value. If the alert is constantly firing, verify the available space on the GlusterFS nodes and adjust the threshold accordingly. In the Prometheus Web UI, use the raise condition query for a longer period of time to define the best threshold.
	For example, to change the threshold to 95%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	    `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	    `classes:`
	    `- cluster.<cluster_name>.stacklight.custom.alerts`
	    `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	  `parameters:`
	  `  prometheus:`
	  `    server:`
	  `      alert:`
	  `        GlusterfsSpaceUsedMajor:`
	  `          if: >-`
	  `            glusterfs_space_percent_used >= 95`
	3. From the Salt Master node, apply the changes:
	  `salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

GlusterfsMountMissing

Available since the 2019.2.11 maintenance update

Severity	Major
Summary	GlusterFS mount point is not mounted.
Raise condition	delta(glusterfs_mount_scrapes:rate5m{fstype=~"(fuse.)?glusterfs"}[5m]) < 0 or glusterfs_mount_scrapes:rate5m{fstype=~"(fuse.)?glusterfs"} == 0
Description	Raises when a GlusterFS mount point is not mounted. The path, host, and device labels in the raised alert contain the path, node name, and device of the affected mount point.
Troubleshooting	To perform a manual remount, apply the salt '*' state.sls glusterfs.client Salt state from the Salt Master node.
Tuning	Not required

HAProxy

This section describes the alerts for the HAProxy service.

- HaproxyServiceDown
- HaproxyServiceDownMajor
- HaproxyServiceOutage
- HaproxyHTTPResponse5xxTooHigh
- HaproxyBackendDown
- HaproxyBackendDownMajor
- HaproxyBackendOutage

HaproxyServiceDown

Severity	Minor
Summary	The HAProxy service on the {{ $labels.host }} node is down.
Raise condition	haproxy_up != 1
Description	Raises when the HAProxy service on a node does not respond to Telegraf, typically meaning that the HAproxy process is in the DOWN state on that node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the HAProxy status by running systemctl status haproxy on the affected node. • If HAProxy is up and running, inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

HaproxyServiceDownMajor

Severity	Major
Summary	More than 50% of HAProxy services within the {{ $labels.cluster }} cluster are down.

Raise condition	count(label_replace(haproxy_up, "cluster", "$1", "host", "([^0-9]+).+") != 1) by (cluster) >= 0.5 * count(label_replace(haproxy_up, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster)
Description	Raises when the HAProxy service does not respond to Telegraf on more than 50% of cluster nodes. The cluster label in the raised alert contains the cluster prefix, for example, ctl, dbs, or mon.
Troubleshooting	• Inspect the HaproxyServiceDown alerts for the host names of the affected nodes. • Inspect dmesg and /var/log/kern.log. • Inspect the logs in /var/log/haproxy.log. • Inspect the Telegraf logs using journalctl -u telegraf.
Tuning	Not required

HaproxyServiceOutage

Severity	Critical
Summary	All HAProxy services within the {{ $labels.cluster }} cluster are down.
Raise condition	count(label_replace(haproxy_up, "cluster", "$1", "host", "([^0-9]+).+") != 1) by (cluster) == count(label_replace(haproxy_up, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster)
Description	Raises when the HAProxy service does not respond to Telegraf on all nodes of a cluster, typically indicating deployment or configuration issues. The cluster label in the raised alert contains the cluster prefix, for example, ctl, dbs, or mon.
Troubleshooting	• Inspect the HaproxyServiceDown alerts for the host names of the affected nodes. • Inspect dmesg and /var/log/kern.log. • Inspect the logs in /var/log/haproxy.log. • Inspect the Telegraf logs using journalctl -u telegraf.
Tuning	Not required

HaproxyHTTPResponse5xxTooHigh

Severity	Warning

Summary	The average per-second rate of 5xx HTTP errors on the {{ $labels.host }} node for the {{ $labels.proxy }} back end is {{ $value }} (as measured over the last 2 minutes).
Raise condition	rate(haproxy_http_response_5xx{sv="FRONTEND"}[2m]) > 1
Description	Raises when the HTTP 5xx responses sent by HAProxy increased for the last 2 minutes, indicating a configuration issue with the HAProxy service or back-end servers within the cluster. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Inspect the HAproxy logs by running journalctl -u haproxy on the affected node and verify the state of the back-end servers.
Tuning	Not required

HaproxyBackendDown

Severity	Minor
Summary	The {{ $labels.proxy }} back end on the {{ $labels.host }} node is down.
Raise condition	increase(haproxy_chkdown{sv="BACKEND"}[1m]) > 0
Description	Raises when an internal HAProxy check for the back-end availability reported the back-end outage. The host and proxy labels in the raised alert contain the host name of the affected node and the service proxy name.

Troubleshooting	• Inspect the HAProxy logs by running journalctl -u haproxy on the affected node.
	• Verify the state of the affected back-end server:
	  • Verify that the server is responding and the back-end service is active and responsive.
	  • Verify the state of the back-end service using an HTTP GET request, for example, curl -XGET http://ctl01:8888/. Typically, the 200 response code indicates the healthy state.

Tuning	Not required

HaproxyBackendDownMajor

Severity	Major
Summary	More than 50% of {{ $labels.proxy }} backends are down.
Raise condition	• In 2019.2.10 and prior: 0.5 * avg(sum(haproxy_active_servers{type=""server""}) by (host, proxy) + sum(haproxy_backup_servers{type=""server""}) by (host, proxy)) by (proxy) >= avg(sum(haproxy_active_servers{type=""backend""}) by (host, proxy) + sum(haproxy_backup_servers{type=""backend""}) by (host, proxy)) by (proxy) • In 2019.2.11 and newer: avg(sum(haproxy_active_servers{type="server"}) by (host, proxy) + sum(haproxy_backup_servers{type="server"}) by (host, proxy)) by (proxy) - avg(sum(haproxy_active_servers{type="backend"}) by (host, proxy) + sum(haproxy_backup_servers{type="backend"}) by (host, proxy)) by (proxy) >= 0.5 * avg(sum(haproxy_active_servers{type="server"}) by (host, proxy) + sum(haproxy_backup_servers{type="server"}) by (host, proxy)) by (proxy)
Description	Raises when at least half of the back-end servers (>=50%) used by the HAProxy service are in the DOWN state. The host and proxy labels in the raised alert contain the host name of the affected node and the service proxy name.

Troubleshooting	• Inspect the HAProxy logs by running journalctl -u haproxy on the affected node.
	• Verify the state of the affected back-end server:
	  • Verify that the server is responding and the back-end service is active and responsive.
	  • Verify the state of the back-end service using an HTTP GET request, for example, curl -XGET http://ctl01:8888/. Typically, the 200 response code indicates the healthy state.

Tuning	Not required

HaproxyBackendOutage

Severity	Critical
Summary	All {{ $labels.proxy }} backends are down.
Raise condition	max(haproxy_active_servers{sv=""BACKEND""}) by (proxy) + max(haproxy_backup_servers{sv=""BACKEND""}) by (proxy) == 0

Description	Raises when all back-end servers used by the HAProxy service across the cluster are not available to process the requests proxied by HAProxy, typically indicating deployment or configuration issues. The proxy label in the raised alert contains the service proxy name.
Troubleshooting	• Verify the affected backends. • Inspect the HAProxy logs by running journalctl -u haproxy on the affected node. • Inspect Telegraf logs by running journalctl -u telegraf on the affected node.
Tuning	Not required

Keepalived

- KeepalivedProcessDown
- KeepalivedProcessNotResponsive
- KeepalivedFailedState
- KeepalivedUnknownState
- KeepalivedMultipleIPAddr
- KeepalivedServiceOutage

KeepalivedProcessDown

Severity	Major
Summary	The Keepalived process on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="keepalived"} == 0
Description	Raised when Keepalived on a particular host does not respond Telegraf, typically indicating that Keepalived is down. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the Keepalived status on the affected node using systemctl status keepalived. • Inspect the Keepalived logs on the affected node using journalctl -u keepalived. • Inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

KeepalivedProcessNotResponsive

Severity	Major
Summary	The Keepalived process on the {{ $labels.host }} node is not responding.
Raise condition	keepalived_up == 0

Description	Raises when Keepalived on a particular host does not respond to Telegraf, typically indicating that Keepalived is running but is not responsive on that node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the Keepalived status on the affected node using service keepalived status. • Inspect the Keepalived logs on the affected node using journalctl -u keepalived. • Inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

KeepalivedFailedState

Severity	Minor
Summary	The Keepalived VRRP {{ $labels.name }} is in the FAILED state on the {{ $labels.host }} node.
Raise condition	keepalived_state == 0
Description	Raises when the Keepalived Virtual Router Redundancy Protocol (VRRP) is in the FAILED state on a node, typically indicating network issues. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Inspect the Keepalived logs on the affected node using journalctl -u keepalived. • Inspect the Telegraf logs on the affected node using journalctl -u telegraf. • Inspect the affected node for any network issues.
Tuning	Not required

KeepalivedUnknownState

Severity	Minor
Summary	The Keepalived VRRP {{ $labels.name }} is in the UNKNOWN state on the {{ $labels.host }} node.
Raise condition	keepalived_state == -1

Description	Raises when the Keepalived Virtual Router Redundancy Protocol (VRRP) is in the UNKNOWN state on a node, typically indicating that Keepalived has improperly reported its state or Telegraf cannot gather the state. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Inspect the Keepalived logs on the affected node using journalctl -u keepalived. • Inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

KeepalivedMultipleIPAddr

Severity	Major
Summary	The Keepalived {{ $labels.ip }} virtual IP is assigned more than once.
Raise condition	count(ipcheck_assigned) by (ip) > 1
Description	Raises when the virtual IP address (VIP) of Keepalived is assigned more than once (on more than one node within a cluster).
Troubleshooting	On each node of the Keepalived cluster, ctl nodes by default, verify if the VIP is assigned on two or more nodes or interfaces using the ip a | grep VIP_address command.
Tuning	Not required

KeepalivedServiceOutage

Severity	Critical
Summary	All Keepalived processes within the {{ $labels.cluster}} cluster are down.
Raise condition	count(label_replace(procstat_running{process_name="keepalived"}, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster) == count(label_replace(procstat_running{process_name="keepalived"} == 0, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster)
Description	Raises when all Keepalived services across the cluster do not respond to Telegraf, typically indicating configuration or deployment issues.

Troubleshooting	• Inspect the KeepalivedProcessDown alerts for the host names of the affected nodes. • Inspect the Keepalived logs on the affected nodes using journalctl -u keepalived. • Inspect the Telegraf logs on the affected nodes using journalctl -u telegraf.
Tuning	Not required

libvirt

This section describes the alerts for the libvirt service.

LibvirtDown

Severity	Critical
Summary	The libvirt metric exporter fails to gather metrics on the {{ $labels.host }} node for 2 minutes.
Raise condition	libvirt_up == 0
Description	Raises when libvirt_exporter fails to gather metrics for 2 minutes. The host label in the raised alert contains the hostname of the affected node.
Troubleshooting	• Verify the libvirt-exporter service status using systemctl status libvirt-exporter. • Inspect the libvirt-exporter service logs using journalctl -u libvirt-exporter or in /var/log/libvirt-exporter. • Inspect the libvirt logs using journalctl -u libvirt or in /var/log/libvirt.
Tuning	Not required

Memcached

This section describes the alerts for the Memcached service.

- MemcachedServiceDown
- MemcachedServiceRespawn
- MemcachedConnectionThrottled
- MemcachedConnectionsNoneMinor
- MemcachedConnectionsNoneMajor
- MemcachedItemsNoneMinor
- MemcachedEvictionsLimit

MemcachedServiceDown

Severity	Minor
Summary	The Memcached service on the {{ $labels.host }} node is down.
Raise condition	memcached_up == 0
Description	Raised when Telegraf cannot gather metrics from the Memcached service, typically indicating that Memcached is down on one node and caching does not work on that node. The host label in the raised alert contains the host name of the affected node
Troubleshooting	• Verify the Memcached service status using systemctl status memcached. • Inspect the Memcached service logs using journalctl -xfu memcached.
Tuning	Not required

MemcachedServiceRespawn

Removed since the 2019.2.4 maintenance update.

Severity	Warning
Summary	The Memcached service on the {{ $labels.host }} node was respawned.
Raise condition	memcached_uptime < 180

Description	Raises when the Memcached service uptime is below 180 seconds, indicating that it was recently respawned (restarted). If Memcached respawning happened during maintenance, the alert is expected. Otherwise, this alert indicates an issue with the service. The host label in the raised alert contains the host name of the affected node. Warning The alert is a partial duplicate of MemcachedServiceDown and has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, verify and disable this alert.
Troubleshooting	• Verify the Memcached service status using systemctl status memcached. • Inspect the Memcached service logs using journalctl -xfu memcached.
Tuning	Disable the alert as described in Manage alerts.

MemcachedConnectionThrottled

Severity	Warning
Summary	More than 5 client connections to the Memcached database on the {{ $labels.host }} node throttle for 2 minutes.
Raise condition	increase(memcached_conn_yields[1m]) > 5
Description	Raises when the number of times the Memcached connection was throttled reaches 5 over the last minute. This warning appears with the Too many open connections error message in Memcached. Too many connections may cause an error in writing because of the process starvation (blocking). To avoid this, Memcached throttles the connection. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Use telnet to connect to Memcached by running telnet localhost 11211 on the affected node. Then run stats to obtain the server information. • Inspect the Memcached service logs using journalctl -xfu memcached. • Adjust the threshold if required.

<table>
<tr><td>Tuning</td><td>
To change the throttling threshold to 10:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:

```
touch cluster/<cluster_name>/stacklight/custom/alerts.yml
```

Define the new file in cluster/<cluster_name>/stacklight/server.yml:

```
classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...
```


In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

```
parameters:
  prometheus:
    server:
      alert:
        MemcachedConnectionThrottled:
          if: >-
            increase(memcached_conn_yields[1m]) > 10
```

From the Salt Master node, apply the changes:

```
salt 'I@prometheus:server' state.sls prometheus.server
```

Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

MemcachedConnectionsNoneMinor

Severity	Minor
Summary	The Memcached database on the {{ $labels.host }} node has no open connections.
Raise condition	memcached_curr_connections == 0
Description	Raises when no connections to Memcached exist on one node, typically indicating that the connections were dropped. The state may affect performance. The host label in the raised alert contains the host name of the affected node.

Troubleshooting	• Use telnet to connect to Memcached by running telnet localhost 11211 on the affected node. Then run stats to obtain the server information. • Inspect the Memcached service logs using journalctl -xfu memcached.
Tuning	Not required

MemcachedConnectionsNoneMajor

Severity	Major
Summary	The Memcached database has no open connections on all nodes.
Raise condition	count(memcached_curr_connections == 0) == count(memcached_up)
Description	Raises when no connections to Memcached exist on all nodes, indicating that Memcached has no client connected to it and does not receive data.
Troubleshooting	• Use telnet to connect to Memcached by running telnet localhost 11211 on the affected node. Then run stats to obtain the server information. • Inspect the Memcached service logs using journalctl -xfu memcached.
Tuning	Not required

MemcachedItemsNoneMinor

Removed since the 2019.2.4 maintenance update.

Severity	Minor
Summary	The Memcached database on the {{ $labels.host }} node is empty.
Raise condition	memcached_curr_items == 0

Description	Raises when a Memcached database has no items on one node. As Memcached is an in-memory database, this may be the result of Memcached respawn. Otherwise, investigate the reason. The host label in the raised alert contains the host name of the affected node. Warning The alert has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, disable this alert.
Troubleshooting	1. To confirm the issue, use telnet to connect to Memcached by running telnet localhost 11211 on the affected node. 2. Run stats and search for curr_items and evictions to verify that the items were not removed before their TTL. 3. Run stats items for further details on the status of the items.
Tuning	Disable the alert as described in Manage alerts.

MemcachedEvictionsLimit

Severity	Warning
Summary	More than 10 evictions in the Memcached database occurred on the {{ $labels.host }} node during the last minute.
Raise condition	increase(memcached_evictions[1m]) > 10
Description	Raises when the number of Memcached items that were removed before the ending of TTL has increased by 10 (default threshold) over the last minute. Memcached is used on the OpenStack controller nodes to cache the service authentication tokens. A high number of evictions indicates a heavy token rotation since old items must be removed to free the space for the new ones, based on pseudo-LRU. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	1. Use telnet to connect to Memcached by running telnet localhost 11211 on the affected node. 2. Run stats slabs and search for total_pages, chunk_size, and chunks_per_page to verify if the slabs consume too much space.

<table>
<tr>
<td>Tuning</td>
<td>
To change the evictions limit to 60:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           MemcachedEvictionsLimit:
             if: >-
               increase(memcached_evictions[1m]) > 60
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.
</td>
</tr>
</table>

NGINX

This section describes the alerts for the NGINX service.

- NginxServiceDown
- NginxServiceOutage
- NginxDroppedIncomingConnections

NginxServiceDown

Severity	Minor
Summary	The NGINX service on the {{ $labels.host }} node is down.
Raise condition	nginx_up != 1
Description	Raises when the NGINX service on a host node does not respond to Telegraf, typically indicating that the NGINX service is not running on that node for 1 minute. The host label in the raised alert contains the name of the affected node.
Troubleshooting	• Verify the NGINX status on the affected node using service nginx status. • If NGINX is up and running, inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

NginxServiceOutage

Severity	Critical
Summary	All NGINX processes within the {{ $labels.cluster }} cluster are down.
Raise condition	count(label_replace(nginx_up, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster) == count(label_replace(nginx_up == 0, "cluster", "$1", "host", "([^0-9]+).+")) by (cluster)
Description	Raises when all NGINX services across a cluster do not respond to Telegraf, typically indicating deployment or configuration issues. The cluster label in the raised alert contains the prefix of a cluster, for example, ctl, dbs, or mon.

Troubleshooting	Inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

NginxDroppedIncomingConnections

Severity	Minor
Summary	NGINX drops {{ $value }} accepted connections per second for 5 minutes.
Raise condition	irate(nginx_accepts[5m]) - irate(nginx_handled[5m]) > 0
Description	Raises when NGINX has dropped the accepted connections for the last 5 minutes, indicating that NGINX does not handle every incoming connection, which may be caused by a resource or configuration limit. The host label contains the name of the affected node.
Troubleshooting	Inspect the NGINX logs using journalctl -u nginx.
Tuning	Not required

NTP

This section describes the alerts for the NTP service.

NtpOffsetTooHigh

Severity	Warning
Summary	The NTP offset on the {{ $labels.host }} node is more than 200 milliseconds for 2 minutes.
Raise condition	ntpq_offset >= 200
Description	Raises when the NTP offset on a node reaches the threshold of 200 milliseconds for 2 minutes, typically indicating that the host fails to synchronize the time with the NTP server or the NTP server is malfunctioning. A too high offset affects the metrics collection and querying the time series database. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Synchronize the time with a properly operating NTP server: 1. Enter the NTP CLI by running ntpq on the affected node. 2. List the NTP peers by running peers and exit the NTP CLI. 3. Set the date and time using ntpdate -q <peer_from_list>. If the issue persists: 1. Enter the NTP CLI by running ntpq on the affected node. 2. List the associations by running as. 3. Investigate the reason for the server rejection by running rv <association_id> with a chosen association ID. 4. Inspect the output for the occurrence of flash code, rootdispersion, dispersion, and jitter. Avoid syncing with servers that have a large dispersion.

<table>
<tr>
<td>Tuning</td>
<td>

For example, to change the threshold of the NTP offset to 500:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
    prometheus:
      server:
        alert:
          NtpOffsetTooHigh:
            if: >-
              ntpq_offset >= 500
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

Open vSwitch

This section describes the alerts for the Open vSwitch (OVS) processes.

> Warning
>
> - Monitoring of the OVS processes is available starting from the MCP 2019.2.3 update.
> - The OVSInstanceArpingCheckDown alert is available starting from the MCP 2019.2.4 update.
> - The OVSTooManyPortRunningOnAgent, OVSErrorOnPort, OVSNonInternalPortDown and OVSGatherFailed alerts are available starting from the MCP 2019.2.6 update.

- ProcessOVSVswitchdMemoryWarning
- ProcessOVSVswitchdMemoryCritical
- OVSInstanceArpingCheckDown
- OVSTooManyPortRunningOnAgent
- OVSErrorOnPort
- OVSNonInternalPortDown
- OVSGatherFailed

ProcessOVSVswitchdMemoryWarning

Available starting from the 2019.2.3 maintenance update

Severity	Warning
Summary	The ovs-vswitchd process consumes more than 20% of system memory.
Raise condition	procstat_memory_vms{process_name="ovs-vswitchd"} / on(host) mem_total > 0.2
Description	Raises when the virtual memory of the ovs-switchd process exceeds 20% of the host memory.
Tuning	Not required

ProcessOVSVswitchdMemoryCritical

Available starting from the 2019.2.3 maintenance update

Severity	Critical
Summary	The ovs-vswitchd process consumes more than 30% of system memory.
Raise condition	procstat_memory_vms{process_name="ovs-vswitchd"} / on(host) mem_total > 0.3
Description	Raises when the virtual memory of the ovs-switchd process exceeds 30% of the host memory.
Tuning	Not required

OVSInstanceArpingCheckDown

Available starting from the 2019.2.4 maintenance update

Severity	Major
Summary	The OVS instance arping check is down.
Raise condition	instance_arping_check_up == 0
Description	Raises when the OVS instance arping check on the {{ $labels.host }} node is down for 2 minutes. The host label in the raised alert contains the affected node name.
Tuning	Not required

OVSTooManyPortRunningOnAgent

Available starting from the 2019.2.6 maintenance update

Severity	Major
Summary	The number of OVS port is {{ $value }} (ovs-vsctl list port) on the {{ $labels.host }} host, which is more than the expected limit.
Raise condition	sum by (host) (ovs_bridge_status) > 1500

Description	Raises when too many networks are created or OVS does not properly clean up the OVS ports. OVS may malfunction if too many ports are assigned to a single agent. Warning For production environments, configure the alert after deployment.
Troubleshooting	• Run ovs-vsctl show from the affected node and openstack port list from the OpenStack controller nodes and inspect the existing ports. • Remove the unneeded ports or redistribute the OVS ports.

Tuning	For example, to change the threshold to 1600:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        OVSTooManyPortRunningOnAgent:`
	`          if: >-`
	`            sum by (host) (ovs_bridge_status)  > 1600`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

OVSErrorOnPort

Available starting from the 2019.2.6 maintenance update

Severity	Critical
Summary	The {{ $labels.port }} OVS port on the {{ $labels.bridge }} bridge running on the {{ $labels.host }} host is reporting errors.
Raise condition	ovs_bridge_status == 2
Description	Raises when an OVS port reports errors, indicating that the port is not working properly.
Troubleshooting	1. From the affected node, run ovs-vsctl show. 2. Inspect the output for error entries.
Tuning	Not required

OVSNonInternalPortDown

Available starting from the 2019.2.6 maintenance update

Severity	Critical
Summary	The {{ $labels.port }} OVS port on the {{ $labels.bridge }} bridge running on the {{ $labels.host }} host is down.
Raise condition	ovs_bridge_status{type!="internal"} == 0
Description	Raises when the port on the OVS bridge is in the DOWN state, which may lead to an unexpected network disturbance.
Troubleshooting	1. From the affected node, run ip a to verify if the port is in the DOWN state. 2. If required, bring the port up using ifconfig <interface> up.
Tuning	Note required

OVSGatherFailed

Available starting from the 2019.2.6 maintenance update

Severity	Critical
Summary	Failure to gather the OVS information on the {{ $labels.host }} host.

Raise condition	ovs_bridge_check == 0
Description	Raises when the check script for the OVS bridge fails to gather data. OVS is not monitored.
Troubleshooting	Run /usr/local/bin/ovs_parse_bridge.py from the affected host and inspect the output.
Tuning	Not required

RabbitMQ

This section describes the alerts for the RabbitMQ service.

- RabbitmqServiceDown
- RabbitmqServiceOutage
- RabbitMQUnequalQueueCritical
- RabbitmqDiskFullWarning
- RabbitmqDiskFullCritical
- RabbitmqMemoryLowWarning
- RabbitmqMemoryLowCritical
- RabbitmqMessagesTooHigh
- RabbitmqErrorLogsTooHigh
- RabbitmqErrorLogsMajor
- RabbitmqFdUsageWarning
- RabbitmqFdUsageCritical

RabbitmqServiceDown

Severity	Critical
Summary	The RabbitMQ service on the {{$labels.host}} node is down.
Raise condition	rabbitmq_up == 0
Description	Raises when the RabbitMQ service is down on one node, which affects the RabbitMQ availability. The alert raises 1 minute after the issue occurrence. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the RabbitMQ status using systemctl status rabbitmq-server. • Inspect the RabbitMQ logs in /var/log/rabbitmq. • Verify that the node has enough resources, such as disk space or RAM.
Tuning	Not required

RabbitmqServiceOutage

Severity	Critical
Summary	All RabbitMQ services are down.
Raise condition	count(rabbitmq_up == 0) == count(rabbitmq_up)
Description	Raises when RabbitMQ is down on all nodes, indicating that the service is unavailable. The alert raises 1 minute after the issue occurrence.
Troubleshooting	• Verify the RabbitMQ status on the msg nodes using systemctl status rabbitmq-server. • Inspect the RabbitMQ logs in the /var/log/rabbitmq directory on the msg nodes. • Verify that the node has enough resources such as disk space or RAM.
Tuning	Not required

RabbitMQUnequalQueueCritical

Available starting from the 2019.2.5 maintenance update

Severity	Critical
Summary	The RabbitMQ service has unequal number of queues across the cluster instances.
Raise condition	max(rabbitmq_overview_queues) != min(rabbitmq_overview_queues)
Description	Raises when the RabbitMQ cluster nodes have inconsistent number of queues for 10 minutes. This issue can occur after service restart and cause the inaccessibility of RabbitMQ.
Troubleshooting	Contact Mirantis support.
Tuning	Not required

RabbitmqDiskFullWarning

Severity	Warning

Summary	The RabbitMQ service on the {{$labels.host}} node has less than 500 MB of free disk space.
Raise condition	rabbitmq_node_disk_free <= rabbitmq_node_disk_free_limit * 10
Description	Rasies when the consumption of the available disk space by RabbitMQ reaches 500 MB (by default, 10 multiplied by 50 MB). RabbitMQ checks the available disk space more frequently as it shrinks, which can affect system load.
Troubleshooting	Free or add more disk space on the affected node.

Tuning	To change the threshold to 15:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	  `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	  `classes:`
	  `- cluster.<cluster_name>.stacklight.custom.alerts`
	  `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        RabbitmqDiskFullWarning:`
	`          if: >-`
	`            rabbitmq_node_disk_free <= rabbitmq_node_disk_free_limit * 15`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

RabbitmqDiskFullCritical

Severity	Critical
Summary	The RabbitMQ disk space on the {{$labels.host}} node is full.
Raise condition	rabbitmq_node_disk_free <= rabbitmq_node_disk_free_limit
Description	Raises when RabbitMQ uses all the available disk space (less than 50 MB by default). The alert is cluster-wide. RabbitMQ blocks producers and in-memory messages from paging to the disk. Frequent disk checks contribute to load growth. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Add more disk space on the affected node.
Tuning	Not required

RabbitmqMemoryLowWarning

Severity	Warning
Summary	The RabbitMQ service uses more than 80% of memory on the {{$labels.host}} node for 2 minutes.
Raise condition	100 * rabbitmq_node_mem_used / rabbitmq_node_mem_limit >= 100 * 0.8
Description	Raises when the RabbitMQ memory consumption reaches the warning threshold (80% of allocated memory by default). The host label in the raised alert contains the host name of the affected node.

Troubleshooting	• Edit the high memory watermark in the service configuration.
	• Increase paging for RabbitMQ through CLI. For example:
	`rabbitmqctl -n rabbit@msg01 set_global_parameter \`
	`vm_memory_high_watermark_paging_ratio 0.75`
	The service restart will reset this change.
	• Add more memory on the affected node.

Tuning	To change the watermark:
	1. On the cluster level of the Reclass model, specify the vm_high_watermark parameter in openstack/message_queue.yml. For example:
	`rabbitmq:`
	`  server:`
	`    memory:`
	`      vm_high_watermark: 0.8`
	2. From the Salt Master node, apply the following states one by one:
	`salt '*' saltutil.refresh_pillar`
	`salt -C 'I@rabbitmq:server' state.sls rabbitmq.server`

RabbitmqMemoryLowCritical

Severity	Critical
Summary	The RabbitMQ service on the {{$labels.host}} node is out of memory.
Raise condition	rabbitmq_node_mem_used >= rabbitmq_node_mem_limit
Description	RabbitMQ uses all the allocated memory and blocks all connections that are publishing messages to prevent further usage growth. The host label in the raised alert contains the host name of the affected node. If other system services consume more RAM, the system may start to swap, which can cause the Erlang VM crash and RabbitMQ can go down on the node.

Troubleshooting	• Increase paging for RabbitMQ through CLI. For example:
	`rabbitmqctl -n rabbit@msg01 set_global_parameter \`
	`vm_memory_high_watermark_paging_ratio 0.75`
	The service restart will reset this change.
	• Add more memory on the affected node.

Tuning	Not required

RabbitmqMessagesTooHigh

<table>
<tr><td>Severity</td><td>Warning</td></tr>
<tr><td>Summary</td><td>The RabbitMQ service on the {{$labels.host}} node has received more than 2^20 messages.</td></tr>
<tr><td>Raise condition</td><td>rabbitmq_overview_messages > 2^20</td></tr>
<tr><td>Description</td><td>Raises when the quantity of messages received by RabbitMQ exceeds the warning limit, (by default, 1024 multiplied by 1024), typically indicating that some consumer may not peek messages from the queues.</td></tr>
<tr><td>Troubleshooting</td><td>Verify if huge queues are present using rabbitmqctl list_queues.</td></tr>
<tr><td>Tuning</td><td>

For example, to change the disk warning threshold to 2^21:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           RabbitmqMessagesTooHigh:
             if: >-
               rabbitmq_overview_messages > 2^21
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

RabbitmqErrorLogsTooHigh

Severity	Critical Major in 2019.2.4
Summary	The rate of errors in RabbitMQ logs is more than 0.2 error messages per second on the {{$labels.host}} node as measured over 5 minutes.
Raise condition	• In 2019.2.4: sum(rate(log_messages{service="rabbitmq",level=~"(?i:(error|emergency|fatal))"}[5m])) without (level) > 0.2 • In 2019.2.5: sum(rate(log_messages{service="rabbitmq",level=~"(?i:(error|critical))"}[5m])) without (level) > 0.05
Description	Raises when the average per-second rate of the error, fatal, or emergency messages in the RabbitMQ logs on the node is more than 0.2 per second. Fluentd forwards all logs from RabbitMQ to Elasticsearch and counts the number of log messages per severity. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Inspect the log files in the /var/log/rabbitmq directory on the affected node.

<table>
<tr>
<td>Tuning</td>
<td>

Typically, you should not change the default value. If the alert is constantly firing, inspect the RabbitMQ logs in the Kibana web UI. However, you can adjust the threshold to an acceptable error rate for a particular environment. In the Prometheus Web UI, use the raise condition query to view the appearance rate of a particular message type in logs for a longer period of time and define the best threshold.
For example, to change the threshold to 0.4:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           RabbitmqErrorLogsTooHigh:
             if: >-
               sum(rate(log_messages{service=""rabbitmq"", level=~""(?i:\
               (error|emergency|fatal))""}[5m])) without (level) > 0.4
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

RabbitmqErrorLogsMajor

Available starting from the 2019.2.5 maintenance update

Severity	Major
Summary	The RabbitMQ logs on the {{ $labels.host }} node contain errors (as measured over the last 30 minutes).

Raise condition	sum(increase(log_messages{service="rabbitmq",level=~"(?i:(error|critical))"}[30m])) without (level) > 0
Description	Raises when the error or critical messages appear in the RabbitMQ logs on a node. The host label in the raised alert contains the name of the affected node. Fluentd forwards all logs from RabbitMQ to Elasticsearch and counts the number of log messages per severity.
Troubleshooting	Inspect the log files in the /var/log/rabbitmq directory on the affected node.
Tuning	Not required

RabbitmqFdUsageWarning

Available starting from the 2019.2.6 maintenance update

Severity	Warning
Summary	The RabbitMQ service uses {{ $value }}% of all available file descriptors on the {{ $labels.host }} node.
Raise condition	rabbitmq_node_fd_used / rabbitmq_node_fd_total * 100 >= 70
Description	Raises when the RabbitMQ instance uses 70% of available file descriptors. Warning For production environments, configure the alert after deployment.
Troubleshooting	• Inspect openstack/control.yml in the cluster model to verify if the default value of the OpenStack service rpc_workers was overwritten. • Decrease the rpc_workers value and apply the state for the corresponding service.

Tuning	For example, to change the threshold to 60%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        RabbitmqFdUsageWarning:`
	`          if: >-`
	`            rabbitmq_node_fd_used / rabbitmq_node_fd_total * 100 >= 60`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

RabbitmqFdUsageCritical

Available starting from the 2019.2.6 maintenance update

Severity	Critical
Summary	The RabbitMQ service uses {{ $value }}% of all available file descriptors on the {{ $labels.host }} node.
Raise condition	rabbitmq_node_fd_used / rabbitmq_node_fd_total * 100 >= 95

<table>
<tr><td>Description</td><td>Raises when the RabbitMQ instance uses 95% of available file descriptors, indicating that RabbitMQ is about to crash.

Warning
For production environments, configure the alert after deployment.</td></tr>
<tr><td>Troubleshooting</td><td>• Inspect openstack/control.yml in the cluster model to verify if the default value of the OpenStack service rpc_workers was overwritten.
• Decrease the rpc_workers value and apply the state for the corresponding service.</td></tr>
<tr><td>Tuning</td><td>For example, to change the threshold to 87%:
1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
  1. Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre> 2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        RabbitmqFdUsageCritical:
          if: >-
            rabbitmq_node_fd_used / rabbitmq_node_fd_total * 100 >= 87</pre>3. From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>4. Verify the updated alert definition in the Prometheus web UI.</td></tr>
</table>

Reclass

> Note
>
> This feature is available starting from the MCP 2019.2.9 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes the Reclass modifications alerts.

- ReclassUnstagedChanges
- ReclassStagedChanges
- ReclassRemoteDesync

ReclassUnstagedChanges

Severity	Warning
Summary	{{ $labels.value }} files under the {{ $labels.directory }} directory have been modified without being committed.
Raise condition	reclass_files_unstaged_changes > 0
Description	Raises when the Reclass model has been modified but the local changes have not been added to a potential Git commit.
Troubleshooting	• If the alert is unexpected, reset the model to the remote HEAD. • If the alert is expected, add the changes using git add.
Tuning	Not required

ReclassStagedChanges

Severity	Warning
Summary	{{ $labels.value }} files under the {{ $labels.directory }} directory have diverged from the expected local state.
Raise condition	reclass_files_staged_changes > 0

Description	Raises when the local changes in the Reclass model have been added to a Git commit but not committed yet.
Troubleshooting	• If the alert is unexpected, reset the model to the remote HEAD. • If the alert is expected, commit the changes using git commit.
Tuning	Not required

ReclassRemoteDesync

Severity	Warning
Summary	{{ $labels.value }} local files under the {{ $labels.directory }} directory have diverged from the expected remote state.
Raise condition	reclass_remote_desync > 0
Description	Raises when the local and remote configurations in the Reclass model have diverged, meaning that the local history has changed.
Troubleshooting	• If the alert is unexpected, reset the model to the remote HEAD. • If the alert is expected, push the changes using git push.
Tuning	Not required

Salt

This section describes the alerts for the Salt Master and Salt Minion services.

- SaltMasterServiceDown
- SaltMinionServiceDown

SaltMasterServiceDown

Severity	Critical
Summary	The salt-master service on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="salt-master"} == 0
Description	Raises when Telegraf cannot find running salt-master processes on a node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the salt-master service status using systemctl status salt-master. • Inspect the salt-master service logs in /var/log/salt/master.
Tuning	Not required

SaltMinionServiceDown

Severity	Critical
Summary	The salt-minion service on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="salt-minion"} == 0
Description	Raises when Telegraf cannot find running salt-minion processes on a node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the salt-minion service status using systemctl status salt-minion. • Inspect the salt-minion service logs in /var/log/salt/minion.
Tuning	Not required

SMART disks

This section describes the alerts for SMART disks.

> Warning
>
> SMART disks monitoring is available starting from the MCP 2019.2.3 update. For the existing MCP deployments, manually enable SMART disks monitoring as described in Enable SMART disk monitoring.

> Warning
>
> SMART disks alerts have been removed starting from the MCP 2019.2.9 update.

- SystemSMARTDiskUDMACrcErrorsTooHigh
- SystemSMARTDiskHealthStatus
- SystemSMARTDiskReadErrorRate
- SystemSMARTDiskSeekErrorRate
- SystemSMARTDiskTemperatureHigh
- SystemSMARTDiskReallocatedSectorsCount
- SystemSMARTDiskCurrentPendingSectors
- SystemSMARTDiskReportedUncorrectableErrors
- SystemSMARTDiskOfflineUncorrectableSectors
- SystemSMARTDiskEndToEndError

SystemSMARTDiskUDMACrcErrorsTooHigh

Available starting from the 2019.2.3 maintenance update

Severity	Warning
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node is reporting UDMA CRC errors for 5 minutes.
Raise condition	increase(smart_device_udma_crc_errors[1m]) > 0

Description	Raises when the SMART Telegraf input plugin (using smartmontools) detects the SMART UDMA CRC error messages on a host every minute for 5 minutes. The host and device labels in the raised alert contain the name of the affected node and the affected device.
Troubleshooting	Inspect the disk SMART data using the smartctl command.

Tuning	For example, to change the threshold to 5 errors during 5 minutes:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	   1. Create a file for alert customizations:
	   `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	   `classes:`
	   `- cluster.<cluster_name>.stacklight.custom.alerts`
	   `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemSMARTDiskUDMACrcErrorsTooHigh:`
	`          if: >-`
	`            increase(smart_device_udma_crc_errors[5m]) > 5`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemSMARTDiskHealthStatus

Available starting from the 2019.2.3 maintenance update

Severity	Warning
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node is reporting bad health status for 1 minute.

Raise condition	smart_device_health_ok == 0
Description	Raises when the SMART Telegraf input plugin (using smartmontools) detects bad health status of a SMART disk on a host for 1 minute. The host and device labels in the raised alert contain the name of the affected node and the affected device.
Troubleshooting	Inspect the disk SMART data using the smartctl command.
Tuning	Not required

SystemSMARTDiskReadErrorRate

Available starting from the 2019.2.3 maintenance update

Severity	Warning
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node is reporting an increased read error rate for 5 minutes.
Raise condition	increase(smart_device_read_error_rate[1m]) > 0
Description	Raises when the SMART Telegraf input plugin (using smartmontools) detects the ReadErrorRate messages on a host every minute for the last 5 minutes. The host and device labels in the raised alert contain the name of the affected node and the affected device.
Troubleshooting	Inspect the disk SMART data using the smartctl command.

<table>
<tr><td>Tuning</td><td>
For example, to change the threshold to 5 errors during 5 minutes:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
    1. Create a file for alert customizations:

```
touch cluster/<cluster_name>/stacklight/custom/alerts.yml
```

    2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

```
classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...
```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

```
parameters:
  prometheus:
    server:
      alert:
        SystemSMARTDiskReadErrorRate:
          if: >-
            increase(smart_device_read_error_rate[5m]) > 5
```

3. From the Salt Master node, apply the changes:

```
salt 'I@prometheus:server' state.sls prometheus.server
```

4. Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

SystemSMARTDiskSeekErrorRate

Available starting from the 2019.2.3 maintenance update

Severity	Warning
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node is reporting an increased seek error rate for 5 minutes.
Raise condition	increase(smart_device_seek_error_rate[1m]) > 0
Description	Raises when the SMART Telegraf input plugin (using smartmontools) detects the SeekErrorRate messages on a host every minute for the last 5 minutes. The host and device labels in the raised alert contain the name of the affected node and the affected device.

<table>
<tr><td>Troubleshooting</td><td>Inspect the disk SMART data using the smartctl command.</td></tr>
<tr><td>Tuning</td><td>To change the threshold to 5 errors during 5 minutes:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>
Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        SystemSMARTDiskSeekErrorRate:
          if: >-
            increase(smart_device_seek_error_rate[5m]) > 5</pre>
From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>
Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

SystemSMARTDiskTemperatureHigh

Available starting from the 2019.2.3 maintenance update

Severity	Warning
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node has a temperature of {{ $value }}C for 5 minutes.
Raise condition	smart_device_temp_c >= 60

Description	Raises when the SMART Telegraf input plugin detects that the SMART disk temperature on a host is above the default threshold of 60°C for the last 5 minutes. The host and device labels in the raised alert contain the name of the affected node and the affected device.
Troubleshooting	Inspect the disk SMART data using the smartctl command.

Tuning	For example, to change the threshold to >=40C degrees:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemSMARTDiskTemperatureHigh:`
	`          if: >-`
	`            smart_device_temp_c >= 40`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemSMARTDiskReallocatedSectorsCount

Available starting from the 2019.2.3 maintenance update

Severity	Major Minor in 2019.2.4
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node has reallocated {{ $value }} sectors.

Raise condition	increase(smart_attribute_raw_value{name="Reallocated_Sector_Ct"}[10m]) > 0
Description	Raises when the SMART Telegraf input plugin (using smartmontools) detects the ReallocatedSectorsCount messages every 10 minutes. The host and device labels in the raised alert contain the name of the affected node and the affected device.
Troubleshooting	Inspect the disk SMART data using the smartctl command.

Tuning	For example, to change the threshold 5 errors during 5 minutes:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemSMARTDiskReallocatedSectorsCount:`
	`          if: >-`
	`           increase(smart_attribute_raw_value\`
	`           {name="Reallocated_Sector_Ct"}[5m]) > 5`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemSMARTDiskCurrentPendingSectors

Available starting from the 2019.2.3 maintenance update

Severity	Major [Minor in 2019.2.4]
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node has {{ $value }} current pending sectors.
Raise condition	`increase(smart_attribute_raw_value{name="Current_Pending_Sector"} [10m]) > 0`
Description	Raises when the SMART Telegraf input plugin (using smartmontools) detects the CurrentPendingSectors messages every 10 minutes. The host and device labels in the raised alert contain the name of the affected node and the affected device.
Troubleshooting	Inspect the disk SMART data using the smartctl command.

Tuning	For example, to change the threshold to 5 errors during 5 minutes:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	  `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in `cluster/<cluster_name>/stacklight/server.yml`:
	  `classes:`
	  `- cluster.<cluster_name>.stacklight.custom.alerts`
	  `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemSMARTDiskCurrentPendingSectors:`
	`          if: >-`
	`            increase(smart_attribute_raw_value\`
	`            {name="Current_Pending_Sector"}[5m]) > 5`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemSMARTDiskReportedUncorrectableErrors

Available starting from the 2019.2.3 maintenance update

Severity	Major Minor in 2019.2.4
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node has {{ $value }} reported uncorrectable errors.
Raise condition	increase(smart_attribute_raw_value{name="Reported_Uncorrect"}[10m]) > 0
Description	Raises when the SMART Telegraf input plugin (using smartmontools) detects the ReportedUncorrectableErrors messages every 10 minutes. The host and device labels in the raised alert contain the name of the affected node and the affected device.
Troubleshooting	Inspect the disk SMART data using the smartctl command.

<table>
<tr>
<td>Tuning</td>
<td>
For example, to change the threshold to 5 errors during 5 minutes:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre><code>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</code></pre>

Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre><code>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</code></pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre><code>parameters:
  prometheus:
    server:
      alert:
        SystemSMARTDiskReportedUncorrectableErrors:
          if: >-
            increase(smart_attribute_raw_value\
            {name="Reported_Uncorrect"}[5m]) > 5</code></pre>

From the Salt Master node, apply the changes:
<pre><code>salt 'I@prometheus:server' state.sls prometheus.server</code></pre>

Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

SystemSMARTDiskOfflineUncorrectableSectors

Available starting from the 2019.2.5 maintenance update

Severity	Major
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node has {{ $value }} offline uncorrectable sectors.
Raise condition	smart_attribute_raw_value{name="Offline_Uncorrectable"} > 0
Description	Raises when the SMART input plugin detects that the Offline_Uncorrectable parameter is not 0. The host and device labels in the raised alert contain the name of the affected node and the affected device.

Troubleshooting	Inspect the disk SMART data using the smartctl command.
Tuning	Not required

SystemSMARTDiskEndToEndError

Available starting from the 2019.2.3 maintenance update

Severity	Major Minor in 2019.2.4
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node has {{ $value }} end-to-end errors.
Raise condition	increase(smart_attribute_raw_value{name="End-to-End_Error"}[10m]) > 0
Description	Raises when the SMART Telegraf input plugin (using smartmontools) detects the EndToEndError messages every 10 minutes. The host and device labels in the raised alert contain the name of the affected node and the affected device.
Troubleshooting	Inspect the disk SMART data using the smartctl command.

<table>
<tr>
<td>Tuning</td>
<td>

For example, to change the threshold to 5 errors during 5 minutes:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           SystemSMARTDiskEndToEndError:
             if: >-
               increase(smart_attribute_raw_value\
               {name="End-to-End_Error"}[5m]) > 5
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

SSL certificates

This section describes the alerts for SSL certificates.

> Warning
>
> Monitoring of the functionality is available starting from the MCP 2019.2.3 update.

- CertificateExpirationWarning
- CertificateExpirationCritical

CertificateExpirationWarning

Available starting from the 2019.2.3 maintenance update

Severity	Warning
Summary	The {{ $labels.source }} certificate on the {{ $labels.host }} node expires in less than 60 days.
Raise condition	x509_cert_expiry / (24 * 60 * 60) < 60
Description	Raises when a certificate on a node expires in less than 60 days. The host and source labels in the raised alert contain the host name of the affected node and the path to the certificate.
Troubleshooting	Update the certificates.
Tuning	Not required

CertificateExpirationCritical

Available starting from the 2019.2.3 maintenance update

Severity	Critical
Summary	The {{ $labels.source }} certificate on the {{ $labels.host }} node expires in less than 30 days.
Raise condition	x509_cert_expiry / (24 * 60 * 60) < 30

Description	Raises when a certificate on a node expires in less than 30 days. The host and source labels in the raised alert contain the host name of the affected node and the path to the certificate.
Troubleshooting	Update the certificates.
Tuning	Not required

System

This section describes the system alerts.

- SystemCpuFullWarning
- SystemLoadTooHighWarning
- SystemLoadTooHighCritical
- SystemDiskFullWarning
- SystemDiskFullMajor
- SystemDiskInodesFullWarning
- SystemDiskInodesFullMajor
- SystemDiskErrorsTooHigh
- SystemDiskBacklogWarning
- SystemDiskBacklogCritical
- SystemDiskRequestQueuedWarning
- SystemDiskRequestQueuedCritical
- SystemMemoryFullWarning
- SystemMemoryFullMajor
- SystemSwapFullWarning
- SystemSwapFullMinor
- SystemRxPacketsDroppedTooHigh
- SystemTxPacketsDroppedTooHigh
- SystemTxPacketsErrorTooHigh
- SystemRxPacketsErrorTooHigh
- CronProcessDown
- SshdProcessDown
- SshFailedLoginsTooHigh
- PacketsDroppedByCpuWarning
- PacketsDroppedByCpuMinor
- NetdevBudgetRanOutsWarning
- SystemCpuIoWaitWarning
- SystemCpuIoWaitCritical
- SystemCpuStealTimeWarning
- SystemCpuStealTimeCritical

SystemCpuFullWarning

Severity	Warning
Summary	The average CPU usage on the {{ $labels.host }} node is more than 90% for 2 minutes.
Raise condition	100 - avg_over_time(cpu_usage_idle{cpu="cpu-total"}[5m]) > 90
Description	Raises when the average CPU idle time on a node is less than 10% for the last 5 minutes, indicating that the node is under load. The host label in the raised alert contains the name of the affected node.
Troubleshooting	Inspect the output of the top command on the affected node.

<table>
<tr>
<td>Tuning</td>
<td>

For example, to change the threshold to 20%:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           SystemCpuFullWarning:
             if: >-
               100 - avg_over_time(cpu_usage_idle{cpu=""cpu-total""}[5m]) > 80
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

SystemLoadTooHighWarning

Severity	Warning
Summary	The system load per CPU on the {{ $labels.host }} node is more than 1 for 5 minutes.
Raise condition	• In 2019.2.7 and prior: system_load5 / system_n_cpus > 1 • In 2019.2.8: system_load15 / system_n_cpus > 1 • In 2019.2.9 and newer: system_load15{host!~".*cmp[0-9]+"} / system_n_cpus > {{ load_threshold }}
Description	Raises when the average load on the node is higher than 1 per CPU core over the last 5 minutes, indicating that the system is overloaded, many processes are waiting for CPU time. The host label in the raised alert contains the name of the affected node.

Troubleshooting	Inspect the output of the uptime and top commands on the affected node.

Tuning	For example, to change the threshold to 1.5 per core:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	    1. Create a file for alert customizations:
	    `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	    2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	    `classes:`
	    `- cluster.<cluster_name>.stacklight.custom.alerts`
	    `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter depending on the raise conditions provided above. For example:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemLoadTooHighWarning:`
	`          if: >-`
	`            system_load15{host!~".*cmp[0-9]+"} / system_n_cpus > 1.5`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemLoadTooHighCritical

Severity	Critical Prior to 2019.2.8, warning
Summary	The system load per CPU on the {{ $labels.host }} node is more than 2 for 5 minutes.
Raise condition	• In 2019.2.7 and prior: system_load5 / system_n_cpus > 2 • In 2019.2.8: system_load15 / system_n_cpus > 2 • In 2019.2.9 and newer: system_load15{host!~".*cmp[0-9]+"} / system_n_cpus > {{ load_threshold }}

Description	Raises when the average load on the node is higher than 2 per CPU over the last 5 minutes, indicating that the system is overloaded, many processes are waiting for CPU time. The host label in the raised alert contains the name of the affected node.
Troubleshooting	Inspect the output of the uptime and top commands on the affected node.

Tuning	For example, to change the threshold to 3 per core:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter depending on the raise conditions provided above. For example:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemLoadTooHighCritical:`
	`          if: >-`
	`            system_load15{host!~".*cmp[0-9]+"} / system_n_cpus > 3`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemDiskFullWarning

Severity	Warning
Summary	The disk partition ({{ $labels.path }}) on the {{ $labels.host }} node is more than 85% full for 2 minutes.

Raise condition	disk_used_percent >= 85
Description	Raises when the disk partition on a node is 85% full. The host, device, and path labels in the raised alert contain the name of the affected node, device, and the path to the mount point.
Troubleshooting	• Verify the used and free disk space on the node using the df command. • Increase the disk space on the affected node or remove unused data.

Tuning	For example, to change the threshold to 90%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	    touch cluster/<cluster_name>/stacklight/custom/alerts.yml
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	    classes:
	    - cluster.<cluster_name>.stacklight.custom.alerts
	    ...
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	  parameters:
	   prometheus:
	    server:
	     alert:
	      SystemDiskFullWarning:
	       if: >-
	        disk_used_percent >= 90
	3. From the Salt Master node, apply the changes:
	  salt 'I@prometheus:server' state.sls prometheus.server
	4. Verify the updated alert definition in the Prometheus web UI.

SystemDiskFullMajor

Severity	Major

<table>
<tr><td>Summary</td><td>The disk partition ({{ $labels.path }}) on the {{ $labels.host }} node is 95% full for 2 minutes.</td></tr>
<tr><td>Raise condition</td><td>disk_used_percent >= 95</td></tr>
<tr><td>Description</td><td>Raises when the disk partition on a node is 95% full. The host, device, and path labels in the raised alert contain the name of the affected node, device, and the path to the mount point.</td></tr>
<tr><td>Troubleshooting</td><td>

Verify the used and free disk space on the node using the df command.
Increase the disk space on the affected node or remove unused data.

</td></tr>
<tr><td>Tuning</td><td>
For example, to change the threshold to 99%:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>

Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        SystemDiskFullMajor:
          if: >-
            disk_used_percent >= 99</pre>

From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>

Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

SystemDiskInodesFullWarning

Severity	Warning
Summary	The {{ $labels.host }} node uses more than 85% of disk inodes in the {{ $labels.path }} volume for 2 minutes.
Raise condition	100 * disk_inodes_used / disk_inodes_total >= 85.0
Description	Raises when the usage of inodes of a disk partition on the node is 85%. The host, device, and path labels in the raised alert contain the name of the affected node, device, and the path to the mount point.
Troubleshooting	• Verify the used and free inodes on the affected node using the df -i command. • If the disk is not full on the affected node, identify the reason for the inodes leak or remove unused files.

Tuning	For example, to change the threshold to 90%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemDiskInodesFullWarning:`
	`          if: >-`
	`            100 * disk_inodes_used / disk_inodes_total >= 90`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemDiskInodesFullMajor

Severity	Major
Summary	The {{ $labels.host }} node uses more than 95% of disk inodes in the {{ $labels.path }} volume for 2 minutes.
Raise condition	100 * disk_inodes_used / disk_inodes_total >= 95.0
Description	Raises when the usage of inodes of a disk partition on the node is 95%. The host, device, and path labels in the raised alert contain the name of the affected node, device, and the path to the mount point.
Troubleshooting	• Verify the used and free inodes on the affected node using the df -i command. • If the disk is not full on the affected node, identify the reason for the inodes leak or remove unused files.

<table>
<tr>
<td>Tuning</td>
<td>

For example, to change the threshold to 99%:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           SystemDiskInodesFullMajor:
             if: >-
               100 * disk_inodes_used / disk_inodes_total >= 99
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

SystemDiskErrorsTooHigh

Severity	Warning
Summary	The {{ $labels.device }} disk on the {{ $labels.host }} node is reporting errors for 5 minutes.
Raise condition	increase(hdd_errors_total[1m]) > 0
Description	Raises when disk error messages were detected every minute for the last 5 minutes in the syslog (/var/log/syslog) on a host. Fluentd parses the syslog for the error.*\b[sv]d[a-z]{1,2}\d{0,3}\b.* and \b[sv]d[a-z]{1,2}\d{0,3}\b.*error regular expressions and increases the count in case of success. The host and device labels in the raised alert contain the name of the affected node and the affected device.

Troubleshooting	Inspect the syslog journal for error words using grep.

Tuning	For example, to change the threshold to 5 errors during 5 minutes:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemDiskErrorsTooHigh:`
	`          if: >-`
	`            increase(hdd_errors_total[5m]) > 5`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemDiskBacklogWarning

Available starting from the 2019.2.14 maintenance update

Severity	Warning
Summary	Disk {{ $labels.name }} backlog warning.
Raise condition	rate(diskio_weighted_io_time{name=~"(hd[a-z]?|sd[a-z]?|nvme[0-9]?[a-z]?[0-9]?)"}[1m]) / 1000 > 10

<table>
<tr><td>Description</td><td>I/O requests for the {{ $labels.name }} disk on the {{ $labels.host }} node exceeded the concurrency level of 10 during the last 10 minutes.</td></tr>
</table>

SystemDiskBacklogCritical

Available starting from the 2019.2.14 maintenance update

<table>
<tr><td>Severity</td><td>Critical</td></tr>
<tr><td>Summary</td><td>Disk {{ $labels.name }} backlog critical.</td></tr>
<tr><td>Raise condition</td><td>rate(diskio_weighted_io_time{name=~"(hd[a-z]?|sd[a-z]?|nvme[0-9]?[a-z ?[0-9]?)"}[1m]) / 1000 > 20</td></tr>
<tr><td>Description</td><td>I/O requests for the {{ $labels.name }} disk on the {{ $labels.host }} node exceeded the concurrency level of 20 during the last 10 minutes.</td></tr>
</table>

SystemDiskRequestQueuedWarning

Available starting from the 2019.2.14 maintenance update

<table>
<tr><td>Severity</td><td>Warning</td></tr>
<tr><td>Summary</td><td>Disk {{ $labels.name }} requests were queued for 90% of time.</td></tr>
<tr><td>Raise condition</td><td>rate(diskio_io_time{name=~"(hd[a-z]?|sd[a-z]?|nvme[0-9]?[a-z]?[0-9]?) "}[1m]) / 1000 > 0.9</td></tr>
<tr><td>Description</td><td>I/O requests for the {{ $labels.name }} disk on the {{ $labels.host }} node spent 90% of the device time in queue during the last 10 minutes.</td></tr>
</table>

SystemDiskRequestQueuedCritical

Available starting from the 2019.2.14 maintenance update

<table>
<tr><td>Severity</td><td>Critical</td></tr>
<tr><td>Summary</td><td>Disk {{ $labels.name }} requests were queued for 98% of time.</td></tr>
<tr><td>Raise condition</td><td>rate(diskio_io_time{name=~"(hd[a-z]?|sd[a-z]?|nvme[0-9]?[a-z]?[0-9]?) "}[1m]) / 1000 > 0.98</td></tr>
<tr><td>Description</td><td>I/O requests for the {{ $labels.name }} disk on the {{ $labels.host }} node spent 98% of the device time in queue during the last 10 minutes.</td></tr>
</table>

SystemMemoryFullWarning

Severity	Warning
Summary	The {{ $labels.host }} node uses {{ $value }}% of memory for 2 minutes.
Raise condition	mem_used_percent > 90 and mem_available < 8 * 2^30
Description	Raises when a node uses more than 90% of RAM and less than 8 GB of memory is available, indicating that the node is under a high load. The host label in the raised alert contains the name of the affected node. Warning For production environments, configure the alert after deployment.
Troubleshooting	• Verify the free and used RAM on the affected node using free -h. • Identify the service that consumes RAM.

Tuning	For example, to change the threshold to 80%:
	
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	   1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemMemoryFullWarning:`
	`          if: >-`
	`            mem_used_percent > 80 and mem_available < 8 * 2^30`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemMemoryFullMajor

Severity	Major
Summary	The {{ $labels.host }} node uses {{ $value }}% of memory for 2 minutes.
Raise condition	mem_used_percent > 95 and mem_available < 4 * 2^30

Description	Raises when a node uses more than 95% of RAM and less than 4 GB of memory is available, indicating that the node is under a high load. The host label in the raised alert contains the name of the affected node. Warning For production environments, configure the alert after deployment.
Troubleshooting	• Verify the free and used RAM on the affected node using free -h. • Identify the service that consumes RAM.

Tuning	For example, to change the threshold to 90%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemMemoryFullMajor:`
	`          if: >-`
	`            mem_used_percent > 90 and mem_available < 4 * 2^30`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemSwapFullWarning

Removed since the 2019.2.4 maintenance update

Severity	Warning
Summary	The swap on the {{ $labels.host }} node is more than 50% used for 2 minutes.
Raise condition	swap_used_percent >= 50.0
Description	Raises when the swap on a node is 50% used, indicating that the node is under a high load or out of RAM. The host label in the raised alert contains the name of the affected node. Warning The alert has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, disable this alert.
Troubleshooting	• Verify the free and used RAM and swap on the affected node using free -h. • Identify the service that consumes RAM and swap.
Tuning	Disable the alert as described in Manage alerts.

SystemSwapFullMinor

Removed since the 2019.2.4 maintenance update

Severity	Minor
Summary	The swap on the {{ $labels.host }} node is more than 90% used for 2 minutes.
Raise condition	swap_used_percent >= 90.0
Description	Raises when the swap on a node is 90% used, indicating that the node is under a high load or out of RAM. The host label contains the name of the affected node. Warning The alert has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, disable this alert.

Troubleshooting	• Verify the free and used RAM and swap on the affected node using free -h. • Identify the service that consumes RAM and swap.
Tuning	Disable the alert as described in Manage alerts.

SystemRxPacketsDroppedTooHigh

Severity	Warning
Summary	More than 60 packets received by the {{ $labels.interface }} interface on the {{ $labels.host }} node were dropped during the last minute.
Raise condition	increase(net_drop_in[1m]) > 60 unless on (host,interface) bond_slave_active == 0
Description	Raises when the number of dropped RX packets on an interface (except the bond slave interfaces that are in the BACKUP state) is higher than 60 for the last minute, according to the data from /proc/net/dev of the affected node. The host and interface labels in the raised alert contain the name of the affected node and the affected interface on that node. The reasons can be as follows: • Full softnet backlog • Wrong VLAN tags, packets received with unknown or unregistered protocols • IPv6 frames in case if the server is configured only for IPv4 Warning For production environments, configure the alert after deployment.
Troubleshooting	Inspect the output of the ip -s a command.

Tuning	For example, to change the threshold to 600 packets per 1 minute:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	   1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemRxPacketsDroppedTooHigh:`
	`          if: >-`
	`            increase(net_drop_in[1m]) > 600 unless on (host,interface) \`
	`            bond_slave_active == 0`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemTxPacketsDroppedTooHigh

Severity	Warning
Summary	More than 100 packets transmitted by the {{ $labels.interface }} interface on the {{ $labels.host }} node were dropped during the last minute.
Raise condition	increase(net_drop_out[1m]) > 100

<table>
<tr><td>Description</td><td>Raises when the number of dropped TX packets on the interface is higher than 100 for the last 1 minute, according to the data from /proc/net/dev of the affected node. The host and interface labels in the raised alert contain the name of the affected node and the affected interface on that node.

Warning
For production environments, configure the alert after deployment.</td></tr>
<tr><td>Troubleshooting</td><td>Inspect the output of the ip -s a command.</td></tr>
<tr><td>Tuning</td><td>For example, to change the threshold to 1000 packets per 1 minute:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre><code>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</code></pre>
Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre><code>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</code></pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre><code>parameters:
  prometheus:
    server:
      alert:
        SystemTxPacketsDroppedTooHigh:
          if: >-
            increase(net_drop_out[1m]) > 1000</code></pre>
From the Salt Master node, apply the changes:
<pre><code>salt 'I@prometheus:server' state.sls prometheus.server</code></pre>
Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

SystemRxPacketsErrorTooHigh

Available starting from the 2019.2.15 maintenance update

Severity	Warning
Summary	{{ net_rx_error_threshold }}{%- raw %} received packets had errors.
Raise condition	increase(net_err_in[1m]) > {{ net_rx_error_threshold }}
Description	Raises when the number of packets received by an interface on a node had errors during the last minute. Warning For production environments, configure the alert after deployment.
Troubleshooting	Inspect the output of the ip -s a command.

<table>
<tr><td>Tuning</td><td>
For example, to change the threshold to 600 packets per 1 minute:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           SystemRxPacketsErrorTooHigh:
             if: >-
               increase(net_err_in[1m]) > 600
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

SystemTxPacketsErrorTooHigh

Available starting from the 2019.2.15 maintenance update

Severity	Warning
Summary	{{ net_tx_error_threshold }}{%- raw %} transmitted packets had errors.
Raise condition	increase(net_err_out[1m]) > {{ net_tx_error_threshold }}

Description	Raises when a number of packets transmitted by an interface on a node had errors during the last minute. Warning For production environments, configure the alert after deployment.
Troubleshooting	Inspect the output of the ip -s a command.

Tuning	For example, to change the threshold to 1000 packets per 1 minute:
	
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemTxPacketsErrorTooHigh:`
	`          if: >-`
	`            increase(net_err_out[1m]) > 1000`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

CronProcessDown

Severity	Critical
Summary	The cron process on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="cron"} == 0
Description	Raises when Telegraf cannot find running cron processes on a node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the cron service status using systemctl status cron. • Inspect the Telegraf logs using journalctl -u telegraf.
Tuning	Not required

SshdProcessDown

Severity	Critical
Summary	The SSH process on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="sshd"} == 0
Description	Raises when Telegraf cannot find running sshd processes on a node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the sshd service status using systemctl status sshd. • Inspect the Telegraf logs using journalctl -u telegraf.
Tuning	Not required

SshFailedLoginsTooHigh

Severity	Warning
Summary	More than 5 failed SSH login attempts on the {{ $labels.host }} node during the last 5 minutes.
Raise condition	increase(failed_logins_total[5m]) > 5

Description	Raises when more than 5 failed logins log messages were detected in the syslog (/var/log/syslog) for the last 5 minutes. Fluentd parses the syslog for the ^Invalid user regular expressions and increases the count in case of success. The host label in the raised alert contains the name of the affected node.
Troubleshooting	Inspect the syslog journal for Invalid user words using grep.

Tuning	For example, to change the threshold to 50 packets per 1 hour:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SshFailedLoginsTooHigh:`
	`          if: >-`
	`            increase(failed_logins_total[1h]) > 50`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

PacketsDroppedByCpuWarning

Available starting from the 2019.2.3 maintenance update

Severity	Warning
Summary	The {{ $labels.cpu }} CPU on the {{ $labels.host }} node dropped {{ $value }} packets during the last 10 minutes.

Raise condition	floor(increase(nstat_packet_drop[10m])) > 0
Description	Raises when the number of packets dropped by CPU due to the lack of space in the processing queue is more than 0 for the last 10 minutes, according to the data in column 2 of the /proc/net/softnet_stat file. CPU starts to drop associated packets when its queue (backlog) is full because the interface receives packets faster than the kernel can process them. The host and cpu labels in the raised alert contain the name of the affected node and the CPU. **Warning** For production environments, configure the alert after deployment.
Troubleshooting	Increase the backlog size by modifying the net.core.netdev_max_backlog kernel parameter. For example: `sudo sysctl -w net.core.netdev_max_backlog=3000` For details, see kernel documentation.

<table>
<tr><td>Tuning</td><td>
For example, to change the threshold to 100 packets per 1 hour:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

```
touch cluster/<cluster_name>/stacklight/custom/alerts.yml
```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

```
classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...
```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

```
parameters:
  prometheus:
    server:
      alert:
        PacketsDroppedByCpuWarning:
          if: >-
            floor(increase(nstat_packet_drop[1h])) > 100
```

3. From the Salt Master node, apply the changes:

```
salt 'I@prometheus:server' state.sls prometheus.server
```

4. Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

PacketsDroppedByCpuMinor

Severity	Minor
Summary	The {{ $labels.cpu }} CPU on the {{ $labels.host }} node dropped {{ $value }} packets during the last 10 minutes.
Raise condition	floor(increase(nstat_packet_drop[10m])) > 100

Description	Raises when the number of packets dropped by CPU due to the lack of space in the processing queue is more than 100 for the last 10 minutes, according to the data in column 2 of the /proc/net/softnet_stat file. CPU starts to drop associated packets when its queue (backlog) is full because the interface receives packets faster than the kernel can process them. The host and cpu labels in the raised alert contain the name of the affected node and the CPU. Warning For production environments, configure the alert after deployment.
Troubleshooting	Increase the backlog size by modifying the net.core.netdev_max_backlog kernel parameter. For example: `sudo sysctl -w net.core.netdev_max_backlog=3000` For details, see kernel documentation.

Tuning	For example, to change the threshold to 500 packets per 1 hour:
	
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        PacketsDroppedByCpuMinor:`
	`          if: >-`
	`            floor(increase(nstat_packet_drop[1h])) > 500`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

NetdevBudgetRanOutsWarning

Severity	Warning
Summary	The rate of net_rx_action loops terminations on the {{ $labels.host }} node is {{ $value }} per second during the last 5 minutes.
Raise condition	max(rate(nstat_time_squeeze[5m])) without (cpu) > 0.1

Description	Raises when the average rate of the net_rx_action loop terminations is greater than 0.1 per second for the last 5 minutes, according to the data in column 3 of the /proc/net/softnet_stat file. The alert typically indicates budget consumption or reaching of the time limit. The net_rx_action loop starts processing the packets from the memory to which the device transferred the packets through Direct Memory Access (DMA). Running out of budget or time can cause loop termination. Terminations of net_rx_action indicate that the CPU does not have enough quotas (budget or time) to proceed with all associated packets or some drivers encounter system issues. The host label in the raised alert contains the host name of the affected node. Warning For production environments, configure the alert after deployment.

Troubleshooting	• Verify the number of dropped packets by CPU.
	• If no dropped packets exist, increase the budget or time interval to resolve the issue:
	  • Increase the budget by modifying the net.core.netdev_budget kernel parameter. For example:
	`sysctl -w net.core.netdev_budget=600`
	For details, see kernel documentation.
	  • Increase the time interval by modifying the net.core.netdev_budget_usecs kernel parameter. For example:
	`sudo sysctl -w net.core.netdev_budget_usecs=5000`
	For details, see kernel documentation.

<table>
<tr><td>Tuning</td><td>
For example, to change the threshold to 0.2:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre><code>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</code></pre>

Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre><code>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</code></pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre><code>parameters:
  prometheus:
    server:
      alert:
        NetdevBudgetRanOutsWarning:
          if: >-
            max(rate(nstat_time_squeeze[5m])) without (cpu) > 0.2</code></pre>

From the Salt Master node, apply the changes:
<pre><code>salt 'I@prometheus:server' state.sls prometheus.server</code></pre>

Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

SystemCpuIoWaitWarning

Available starting from the 2019.2.14 maintenance update

Severity	Warning
Summary	CPU waited for I/O 40% of time.
Raise condition	cpu_usage_iowait > 40
Description	The CPU on the {{ $labels.host }} node spent 40% of time waiting for I/O.

SystemCpuIoWaitCritical

Available starting from the 2019.2.14 maintenance update

Severity	Critical
Summary	CPU waited for I/O 50% of time.
Raise condition	cpu_usage_iowait > 50
Description	The CPU on the {{ $labels.host }} node spent 50% of time waiting for I/O.

SystemCpuStealTimeWarning

Available starting from the 2019.2.6 maintenance update

Severity	Warning
Summary	The CPU steal time was above 5.0% on the {{ $labels.host }} node for 5 minutes.
Raise condition	cpu_usage_steal > 5.0
Description	Raises when a VM vCPU spends 5% of time waiting for real CPU for the last 5 minutes, typically occurring during high load in case of CPU shortage. Waiting for resources slows down the processes in the VM. **Warning** For production environments, configure the alert after deployment.

Tuning	For example, to change the threshold to 2%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	   1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemCpuStealTimeWarning:`
	`          if: >-`
	`            cpu_usage_steal > 2.0`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

SystemCpuStealTimeCritical

Available starting from the 2019.2.6 maintenance update

Severity	Critical
Summary	The CPU steal time was above 10.0% on the {{ $labels.host }} node for 5 minutes.
Raise condition	cpu_usage_steal > 10.0

Description	Raises when a VM vCPU spends 10% of time waiting for real CPU for the last 5 minutes, typically occurring during high load in case of CPU shortage. Waiting for resources slows down the processes in the VM. Warning For production environments, configure the alert after deployment.

Tuning	For example, to change the threshold to 5%:
	
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	   1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        SystemCpuStealTimeCritical:`
	`          if: >-`
	`            cpu_usage_steal > 5.0`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

OpenStack

This section describes the alerts available for the OpenStack services.

OpenStack service endpoints

> Note
>
> This feature is available starting from the MCP 2019.2.11 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes the alert for the OpenStack service endpoints outage.

- OpenstackServiceEndpointDown

OpenstackServiceEndpointDown

Severity	Critical
Summary	OpenStack API endpoint is down.
Raise condition	openstack_api_check_status == 0
Description	Raises when an API endpoint from the OpenStack service catalog did not pass the HTTP-probe check. The service_name and interface labels in the raised alert contain the affected service name and interface.
Troubleshooting	Verify the availability of the affected endpoint from the OpenStack service catalog. For a list of all available endpoints, run openstack endpoint list.
Tuning	Not required

Cinder

This section describes the alerts for Cinder.

- CinderApiOutage
- CinderApiDown
- CinderApiEndpointDown
- CinderApiEndpointDownMajor
- CinderApiEndpointsOutage
- CinderServiceDown
- CinderServicesDownMinor
- CinderServicesDownMajor
- CinderServiceOutage
- CinderVolumeProcessDown
- CinderVolumeProcessesDownMinor
- CinderVolumeProcessesDownMajor
- CinderVolumeServiceOutage
- CinderErrorLogsTooHigh

CinderApiOutage

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Cinder API is not accessible for all available Cinder endpoints in the OpenStack service catalog.
Raise condition	max(openstack_api_check_status{name=~"cinder.*"}) == 0
Description	Raises when the checks against all available internal Cinder endpoints in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response codes for Cinder, Cinderv2, and Cinderv3 are 200 and 300. For a list of all available endpoints, run openstack endpoint list.
Troubleshooting	Verify the availability of internal Cinder endpoints (URLs) from the output of openstack endpoint list.

Tuning	Not required

CinderApiDown

Removed since the 2019.2.11 maintenance update

Severity	Major
Summary	Cinder API is not accessible for the {{ $labels.name }} endpoint.
Raise condition	openstack_api_check_status{name=~"cinder.*"} == 0
Description	Raises when the check against one available internal Cinder endpoints in the OpenStack service catalog does not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response codes for Cinder, Cinderv2, and Cinderv3 are 200 and 300. For a list of all available endpoints, run openstack endpoint list.
Troubleshooting	Verify the availability of internal Cinder endpoints (URLs) from the output of the openstack endpoint list command.
Tuning	Not required

CinderApiEndpointDown

Severity	Minor
Summary	The cinder-api endpoint on the {{ $labels.host }} node is not accessible for 2 minutes.
Raise condition	http_response_status{name=~"cinder-api"} == 0
Description	Raises when the check against a Cinder API endpoint does not pass, typically meaning that the service endpoint is down or unreachable due to connectivity issues. The host label in the raised alert contains the host name of the affected node. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.

Tuning	Not required

CinderApiEndpointDownMajor

Severity	Major
Summary	More than 50% of cinder-api endpoints are not accessible for 2 minutes.
Raise condition	count(http_response_status{name=~"cinder-api"} == 0) >= count(http_response_status{name=~"cinder-api"}) * 0.5
Description	Raises when the check against a Cinder API endpoint does not pass on more than 50% of OpenStack controller nodes. For details on the affected nodes, see the host label in the CinderApiEndpointDown alerts. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the CinderApiEndpointDown alerts for the host names of the affected nodes. • Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

CinderApiEndpointsOutage

Severity	Critical
Summary	All available cinder-api endpoints are not accessible for 2 minutes.
Raise condition	count(http_response_status{name=~"cinder-api"} == 0) == count(http_response_status{name=~"cinder-api"})
Description	Raises when the check against a Cinder API endpoint does not pass on all OpenStack controller nodes. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.

Troubleshooting	• Inspect the CinderApiEndpointDown alerts for the host names of the affected nodes. • Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

CinderServiceDown

Severity	Minor
Summary	The {{ $labels.binary }} service on the {{ $labels.hostname }} node is down.
Raise condition	openstack_cinder_service_state == 0
Description	Raises when a Cinder service on the OpenStack controller or compute node is in the DOWN state. For the list of Cinder services, see Cinder Block Storage service overview. The binary and hostname labels contain the name of the service that is in the DOWN state and the node that hosts the service.

Troubleshooting	• Verify the list of Cinder services and their states using openstack volume service list.
	• Verify the status of the corresponding Cinder service on the affected node using systemctl service <binary>.
	• Inspect the logs of the corresponding Cinder service on the affected node in the /var/log/cinder/ directory.
	• Verify the Telegraf monitoring_remote_agent service:
	  • Verify the status of the monitoring_remote_agent service using docker service ls.
	  • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on one of the mon nodes.

Tuning	Not required

CinderServicesDownMinor

Severity	Minor

Summary	More than 30% of {{ $labels.binary }} services are down.
Raise condition	count by(binary) (openstack_cinder_service_state == 0) >= on(binary) count by(binary) (openstack_cinder_service_state) * 0.3
Description	Raises when a Cinder service is in the DOWN state on more than 30% of the ctl or cmp hosts. For the list of services, see Cinder Block Storage service overview. Inspect the hostname label in the CinderServiceDown alerts for details on the affected services and nodes.

Troubleshooting	• Verify the list of Cinder services and their states using openstack volume service list.
	• Verify the status of the corresponding Cinder service on the affected node using systemctl service <binary>.
	• Inspect the logs of the corresponding Cinder service on the affected node in the /var/log/cinder/ directory.
	• Verify the Telegraf monitoring_remote_agent service:
	  • Verify the status of the monitoring_remote_agent service using docker service ls.
	  • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on one of the mon nodes.

Tuning	Not required

CinderServicesDownMajor

Severity	Major
Summary	More than 60% of {{ $labels.binary }} services are down.
Raise condition	count by(binary) (openstack_cinder_service_state == 0) >= on(binary) count by(binary) (openstack_cinder_service_state) * 0.6
Description	Raises when a Cinder service is in the DOWN state on more than 60% of the ctl or cmp hosts. For the list of services, see Cinder Block Storage service overview. Inspect the hostname label in the CinderServiceDown alerts for details on the affected services and nodes.

Troubleshooting	• Verify the list of Cinder services and their states using openstack volume service list.
	• Verify the status of the corresponding Cinder service on the affected node using systemctl service <binary>.
	• Inspect the logs of the corresponding Cinder service on the affected node in the /var/log/cinder/ directory.
	• Verify the Telegraf monitoring_remote_agent service:
	  • Verify the status of the monitoring_remote_agent service using docker service ls.
	  • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on one of the mon nodes.

Tuning	Not required

CinderServiceOutage

Severity	Critical
Summary	All {{ $labels.binary }} services are down.
Raise condition	count by(binary) (openstack_cinder_service_state == 0) == on(binary) count by(binary) (openstack_cinder_service_state)
Description	Raises when a Cinder service is in the DOWN state on all ctl or cmp hosts. For the list of services, see Cinder Block Storage service overview. Inspect the hostname label in the CinderServiceDown alerts for details on the affected services and nodes.

Troubleshooting	• Verify the list of Cinder services and their states using openstack volume service list.
	• Verify the status of the corresponding Cinder service on the affected node using systemctl service <binary>.
	• Inspect the logs of the corresponding Cinder service on the affected node in the /var/log/cinder/ directory.
	• Verify the Telegraf monitoring_remote_agent service:
	  • Verify the status of the monitoring_remote_agent service using docker service ls.
	  • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on one of the mon nodes.

Tuning	Not required

CinderVolumeProcessDown

Available starting from the 2019.2.8 maintenance update

Severity	Minor
Summary	A cinder-volume process is down.
Raise condition	procstat_running{process_name="cinder-volume"} == 0
Description	Raises when a cinder-volume process on a node is down. The host label in the raised alert contains the affected node.
Troubleshooting	• Log in to the corresponding node and verify the process status using systemctl status cinder-volume. • Inspect the cinder-volume log files in the /var/log/cinder/ directory.
Tuning	Not required

CinderVolumeProcessesDownMinor

Available starting from the 2019.2.8 maintenance update

Severity	Minor
Summary	30% of cinder-volume processes are down.
Raise condition	count(procstat_running{process_name="cinder-volume"} == 0) >= count (procstat_running{process_name="cinder-volume"}) * {{ minor_threshold }}
Description	Raises when at least one cinder-volume process is down. Includes the number of cinder-volume processes in the DOWN state (>= {%- endraw %}{{minor_threshold*100}}%{%- raw %}).
Troubleshooting	• Log in to the corresponding node and verify the process status using systemctl status cinder-volume. • Inspect the cinder-volume log files in the /var/log/cinder/ directory.
Tuning	Not required

CinderVolumeProcessesDownMajor

Available starting from the 2019.2.8 maintenance update

Severity	Major
Summary	60% of cinder-volume processes are down.
Raise condition	count(procstat_running{process_name="cinder-volume"} == 0) >= count (procstat_running{process_name="cinder-volume"}) * {{ major_threshold }}
Description	Raises when at least two cinder-volume processes are down. Includes the number of cinder-volume processes in the DOWN state (>= {%- endraw %}{{major_threshold*100}}%{%- raw %}).
Troubleshooting	• Log in to the corresponding node and verify the process status using systemctl status cinder-volume. • Inspect the cinder-volume log files in the /var/log/cinder/ directory.
Tuning	Not required

CinderVolumeServiceOutage

Available starting from the 2019.2.8 maintenance update

Severity	Critical
Summary	The cinder-volume service is down.
Raise condition	count(procstat_running{process_name="cinder-volume"} == 0) == count (procstat_running{process_name="cinder-volume"})
Description	Raises when all cinder-volume processes are down.
Troubleshooting	• Log in to the corresponding node and verify the process status using systemctl status cinder-volume. • Inspect the cinder-volume log files in the /var/log/cinder/ directory.
Tuning	Not required

CinderErrorLogsTooHigh

Severity	Warning
Summary	The average per-second rate of errors in Cinder logs on the {{ $labels.host }} node is larger than 0.2 messages.

<table>
<tr><td>Raise condition</td><td>sum without(level) (rate(log_messages{level=~"(?i:(error|emergency|fatal))", service="cinder"}[5m])) > 0.2</td></tr>
<tr><td>Description</td><td>Raises when the average per-second rate of error, fatal, or emergency messages in Cinder logs on the node is more than 0.2 per second. The host label in the raised alert contains the affected node. Fluentd forwards all logs from Cinder to Elasticsearch and counts the number of log messages per severity.</td></tr>
<tr><td>Troubleshooting</td><td>

Inspect the log files in the /var/log/cinder/ directory on the corresponding node.
Inspect Cinder logs in the Kibana web UI.

</td></tr>
<tr><td>Tuning description</td><td>Typically, you should not change the default value. However, you can adjust the threshold to an acceptable error rate for a particular environment. In the Prometheus Web UI, use the raise condition query to view the appearance rate of a particular message type in logs for a longer period of time and define the best threshold.
To change the threshold to 0.4:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>

Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        CinderErrorLogsTooHigh:
          if: >-
            sum(rate(log_messages{service="cinder", \
            level=~"(?i:(error|emergency|fatal))"}[5m])) without (level) > 0.4</pre>

From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>

Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

Glance

This section describes the alerts for Glance.

- GlanceApiOutage
- GlareApiOutage
- GlanceApiEndpointDown
- GlanceApiEndpointsDownMajor
- GlanceApiEndpointsOutage
- GlanceErrorLogsTooHigh

GlanceApiOutage

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Glance API is not accessible for the Glance endpoint in the OpenStack service catalog.
Raise condition	openstack_api_check_status{name="glance"} == 0
Description	Raises when the checks against all available internal Glance endpoints in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response codes for Glance are 200 and 300.
Troubleshooting	Obtain the list of available endpoints using openstack endpoint list and verify the availability of internal Glance endpoints (URLs) from the list.
Tuning	Not required

GlareApiOutage

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Glare API is not accessible for the Glare endpoint in the OpenStack service catalog.

Raise condition	openstack_api_check_status{name="glare"} == 0
Description	Raises when the checks against all available internal Glare endpoints in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response codes for Glare are 200 and 300.
Troubleshooting	Obtain the list of available endpoints using openstack endpoint list and verify the availability of internal Glance endpoints (URLs) from the list.
Tuning	Not required

GlanceApiEndpointDown

Severity	Minor
Summary	The {{ $labels.name }} endpoint on the {{ $labels.host }} node is not accessible for 2 minutes.
Raise condition	http_response_status{name=~"glance.*"} == 0
Description	Raises when the check against the Glance API endpoint does not pass, typically meaning that the service endpoint is down or unreachable due to connectivity issues. The host label in the raised alert contains the host name of the affected node. Telegraf sends an HTTP request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

GlanceApiEndpointsDownMajor

Severity	Major
Summary	More than 50% of {{ $labels.name }} endpoints are not accessible for 2 minutes.

Raise condition	count by(name) (http_response_status{name=~"glance.*"} == 0) >= count by(name) (http_response_status{name=~"glance.*"}) * 0.5
Description	Raises when the check against the Glance API endpoint does not pass on more than 50% of the ctl nodes, typically meaning that the service endpoint is down or unreachable due to connectivity issues. For details on the affected nodes, see the host label in the GlanceApiEndpointDown alerts. Telegraf sends an HTTP request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

GlanceApiEndpointsOutage

Severity	Critical
Summary	All available {{ $labels.name }} endpoints are not accessible for 2 minutes.
Raise condition	count by(name) (http_response_status{name=~"glance.*"} == 0) == count by(name) (http_response_status{name=~"glance.*"})
Description	Raises when the check against the Glance API endpoint does not pass on all controller nodes, typically meaning that the service endpoint is down or unreachable due to connectivity issues. For details on the affected nodes, see the host label in the GlanceApiEndpointDown alerts. Telegraf sends an HTTP request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

GlanceErrorLogsTooHigh

Severity	Warning

<table>
<tr><td>Summary</td><td>The average per-second rate of errors in Glance logs on the {{ $labels.host }} node is {{ $value }} (as measured over the last 5 minutes).</td></tr>
<tr><td>Raise condition</td><td>sum without(level) (rate(log_messages{level=~"(?i:(error|emergency|fatal))",service="glance"}[5m])) > 0.2</td></tr>
<tr><td>Description</td><td>Raises when the average per-second rate of error, fatal or emergency messages in Glance logs on the node is more than 0.2 messages per second. The host label in the raised alert contains the affected node. Fluentd forwards all logs from Glance to Elasticsearch and counts the number of log messages by severity.</td></tr>
<tr><td>Troubleshooting</td><td>Inspect the log files in /var/log/glance/ on the corresponding node.</td></tr>
</table>

<table>
<tr>
<td>Tuning</td>
<td>

Typically, you should not change the default value. If the alert is constantly firing, inspect the Glance error logs in Kibana and adjust the threshold to an acceptable error rate for a particular environment. In the Prometheus Web UI, use the raise condition query to view the appearance rate of a particular message type in logs for a longer period of time and define the best threshold.
For example, to change the threshold to 0.4:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

    1. Create a file for alert customizations:

        ```
        touch cluster/<cluster_name>/stacklight/custom/alerts.yml
        ```

    2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

        ```
        classes:
        - cluster.<cluster_name>.stacklight.custom.alerts
        ...
        ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

    ```
    parameters:
      prometheus:
        server:
          alert:
            GlanceErrorLogsTooHigh:
              if: >-
                sum(rate(log_messages{service="glance", \
                level=~"(?i:(error|emergency|fatal))"}[5m])) without (level) > 0.4
    ```

3. From the Salt Master node, apply the changes:

    ```
    salt 'I@prometheus:server' state.sls prometheus.server
    ```

4. Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

Heat

This section describes the alerts for Heat.

- HeatApiOutage
- HeatApiDown
- HeatApiEndpointDown
- HeatApiEndpointsDownMajor
- HeatApiEndpointsOutage
- HeatErrorLogsTooHigh

HeatApiOutage

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Heat API is not accessible for all available Heat endpoints in the OpenStack service catalog.
Raise condition	max(openstack_api_check_status{name=~"heat.*"}) == 0
Description	Raises when the checks against all available internal Heat endpoints in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response codes are 200 and 300 for Heat and 200, 300, and 400 for Heat CFN. For a list of all available endpoints run openstack endpoint list.
Troubleshooting	Verify the availability of internal Heat endpoints (URLs) from the output of openstack endpoint list.
Tuning	Not required

HeatApiDown

Removed since the 2019.2.11 maintenance update

Severity	Major
Summary	Heat API is not accessible for the {{ $labels.name }} endpoint.

Raise condition	openstack_api_check_status{name=~"heat.*"} == 0
Description	Raises when the checks against one of the available internal Heat endpoints in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the Openstack service catalog and compares the expected and actual HTTP response codes. The expected response codes are 200 and 300 for Heat and 200, 300, and 400 for Heat CFN. For a list of all available endpoints run openstack endpoint list.
Troubleshooting	Verify the availability of internal Heat endpoints (URLs) from the output of openstack endpoint list.
Tuning	Not required

HeatApiEndpointDown

Severity	Minor
Summary	The {{ $labels.name }} endpoint on the {{ $labels.host }} node is not accessible for 2 minutes.
Raise condition	http_response_status{name=~"heat.*-api"} == 0
Description	Raises when the check against a Heat API endpoint does not pass, typically meaning that the service endpoint is down or unreachable due to connectivity issues. The host label in the raised alert contains the hostname of the affected node. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

HeatApiEndpointsDownMajor

Severity	Major
Summary	{{ $value }} {{ $labels.name }} endpoints (>= 50%) are not accessible for 2 minutes.

Raise condition	count by(name) (http_response_status{name=~"heat.*-api"} == 0) >= count by(name) (http_response_status{name=~"heat.*-api"}) * 0.5
Description	Raises when the check against a Heat API endpoint does not pass on more than 50% of the ctl nodes, typically meaning that the service endpoint is down or unreachable due to connectivity issues. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the HeatApiEndpointDown alerts for the host names of the affected nodes. • Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

HeatApiEndpointsOutage

Severity	Critical
Summary	All available {{ $labels.name }} endpoints are not accessible for 2 minutes.
Raise condition	count by(name) (http_response_status{name=~"heat.*-api"} == 0) == count by(name) (http_response_status{name=~"heat.*-api"})
Description	Raises when the check against a Heat API endpoint does not pass on all OpenStack controller nodes, typically indicating that the service endpoint is down or unreachable due to connectivity issues. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the HeatApiEndpointDown alerts for the host names of the affected nodes. • Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

HeatErrorLogsTooHigh

Severity	Warning

Summary	The average per-second rate of errors in Heat logs on the {{ $labels.host }} node is {{ $value }} as measured over the last 5 minutes.
Raise condition	sum without(level) (rate(log_messages{level=~"(?i:(error|emergency|fatal))",service="heat"}[5m])) > 0.2
Description	Raises when the average per-second rate of error, fatal or emergency messages in Heat logs on the node is more than 0.2 per second. Fluentd forwards all logs from Cinder to Elasticsearch and counts the number of log messages per severity. The host label in the raised alert contains the affected node.
Troubleshooting	Inspect the log files in the /var/log/heat/ directory on the corresponding node.

<table>
<tr>
<td>Tuning</td>
<td>
Typically, you should not change the default value. If the alert is constantly firing, inspect the Heat error logs in the Kibana web UI. However, you can adjust the threshold to an acceptable error rate for a particular environment. In the Prometheus Web UI, use the raise condition query to view the appearance rate of a particular message type in logs for a longer period of time and define the best threshold.
For example, to change the threshold to 0.4:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           HeatErrorLogsTooHigh:
             if: >-
               sum(rate(log_messages{service=""heat"", level=~""(?i:\
               (error|emergency|fatal))""}[5m])) without (level) > 0.4
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.
</td>
</tr>
</table>

Ironic

> **Note**
>
> This feature is available starting from the MCP 2019.2.6 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes the alerts for Ironic.

- IronicErrorLogsTooHigh
- IronicProcessDown
- IronicProcessDownMinor
- IronicProcessDownMajor
- IronicProcessOutage
- IronicDriversMissing
- IronicApiEndpointDown
- IronicApiEndpointsDownMajor
- IronicApiEndpointsOutage
- IronicApiOutage

IronicErrorLogsTooHigh

Severity	Warning
Summary	The average per-second rate of errors in Ironic logs on the {{ $labels.host }} node is {{ $value }} (as measured over the last 5 minutes).
Raise condition	sum(rate(log_messages{service="ironic",level=~"(?i:(error|emergency|fatal))"}[5m])) without (level) > 0.2

<table>
<tr><td>Description</td><td>
Raises when the average per-second rate of error, fatal, or emergency messages in Ironic logs on the node is more than 0.2 per second, which is approximately 1 message per 5 seconds for all nodes in the cluster. Fluentd forwards all logs from Ironic to Elasticsearch and counts the number of log messages per severity. The host label in the raised alert contains the host name of the affected node.

Warning
For production environments, configure the alert after deployment.
</td></tr>
<tr><td>Troubleshooting</td><td>Inspect the log files in the /var/log/ironic/ directory on the affected node.</td></tr>
<tr><td>Tuning</td><td>
For example, to change the threshold to 0.1 (one error per every 10 seconds for the entire cluster):

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
  1. Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>
  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>
2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        IronicErrorLogsTooHigh:
          if: >-
            sum(rate(log_messages{service=""ironic"",level=~""(?i:\
            (error|emergency|fatal))""}[5m])) without (level) > 0.1</pre>
3. From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>
4. Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

IronicProcessDown

Severity	Minor
Summary	The {{ $labels.process_name }} process on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name=~"ironic-.*"} == 0
Description	Raises when an Ironic process (API or conductor) on a host is down. The process_name and host labels contain the name of the affected process and the affected node.
Troubleshooting	• Log in to the corresponding node and verify the process status using systemctl status <process_name>. • Inspect the log files in the /var/log/ironic/<process_name> directory.
Tuning	Not required

IronicProcessDownMinor

Severity	Minor
Summary	The {{ $labels.process_name }} process is down on 33% of nodes.
Raise condition	count(procstat_running{process_name=~"ironic-.*"} == 0) by (process_name) >= count(procstat_running{process_name=~"ironic-.*"}) by (process_name) * 0.33
Description	Raises when 33% of Ironic processes (API or conductor) are down. The process_name label contains the name of the affected processes.
Troubleshooting	• Log in to the corresponding node and verify the process status using systemctl status <process_name>. • Inspect the log files in the /var/log/ironic/<process_name> directory.
Tuning	Not required

IronicProcessDownMajor

Severity	Major
Summary	The {{ $labels.process_name }} process is down on 66% of nodes.
Raise condition	count(procstat_running{process_name=~"ironic-.*"} == 0) by (process_name) >= count(procstat_running{process_name=~"ironic-.*"}) by (process_name) * 0.66
Description	Raises when 66% of Ironic processes (API or conductor) are down. The process_name label contains the name of the affected processes.
Troubleshooting	• Log in to the corresponding node and verify the process status using systemctl status <process_name>. • Inspect the log files in the /var/log/ironic/<process_name> directory.
Tuning	Not required

IronicProcessOutage

Severity	Critical
Summary	The {{ $labels.process_name }} process is down on all nodes.
Raise condition	count(procstat_running{process_name=~"ironic-.*"} == 0) by (process_name) == count(procstat_running{process_name=~"ironic-.*"}) by (process_name)
Description	All specified Ironic processes (API or conductor) are down. The process_name label contains the name of the affected processes.
Troubleshooting	• Log in to the corresponding node and verify the process status using systemctl status <process_name>. • Inspect the log files in the /var/log/ironic/<process_name> directory.
Tuning	Not required

IronicDriversMissing

Severity	Major
Summary	The ironic-conductor {{ $labels.driver }} back-end driver is missing on {{ $value }} node(s).
Raise condition	scalar(count(procstat_running{process_name=~"ironic-conductor"} == 1)) - count(openstack_ironic_driver) by (driver) > 0
Description	Raises when Ironic conductors have a different number of back-end drivers enabled. The cluster performance is not affected. However, the cluster may loose HA.
Troubleshooting	Inspect the Drivers panel of the Ironic Grafana dashboard for the nodes that have the disabled driver.
Tuning	Not required

IronicApiEndpointDown

Severity	Minor
Summary	The {{ $labels.name }} endpoint on the {{ $labels.host }} node is not accessible for 2 minutes.
Raise condition	http_response_status{name=~"ironic-api.*"} == 0
Description	Raises when an Ironic API endpoint (deploy or public API) was not responding to HTTP health checks for 2 minutes. The name and host labels contain the name of the affected endpoint and the affected node.
Troubleshooting	• Inspect the IronicProcessDown alert for the ironic-api process. • Log in to the corresponding node and verify the process status using systemctl status <process_name>. • Inspect the log files in the /var/log/ironic/<process_name> directory.
Tuning	Not required

IronicApiEndpointsDownMajor

Severity	Major
Summary	{{ $value }} of {{ $labels.name }} endpoints (>= 50%) are not accessible for 2 minutes.
Raise condition	count(http_response_status{name=~"ironic-api.*"} == 0) by (name) >= count(http_response_status{name=~"ironic-api.*"}) by (name) * 0.5
Description	Raises when at least 50% of Ironic API endpoints (deploy or public API) were not responding to HTTP health checks for 2 minutes. The name label contains the name of the affected endpoint.
Troubleshooting	• Inspect the IronicProcessDown alert for the ironic-api process. • Log in to the corresponding node and verify the process status using systemctl status <process_name>. • Inspect the log files in the /var/log/ironic/<process_name> directory.
Tuning	Not required

IronicApiEndpointsOutage

Severity	Critical
Summary	All available {{ $labels.name }} endpoints are not accessible for 2 minutes.
Raise condition	count(http_response_status{name=~"ironic-api.*"} == 0) by (name) == count(http_response_status{name=~"ironic-api.*"}) by (name)
Description	Raises when all Ironic API endpoints (deploy or public API) were not responding to HTTP health checks for 2 minutes. The name label contains the name of the affected endpoint.
Troubleshooting	• Inspect the IronicProcessDown alert for the ironic-api process. • Log in to the corresponding node and verify the process status using systemctl status <process_name>. • Inspect the log files in the /var/log/ironic/<process_name> directory.
Tuning	Not required

IronicApiOutage

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Ironic API is not accessible for all available Ironic endpoints in the OpenStack service catalog for 2 minutes.
Raise condition	max(openstack_api_check_status{service="ironic"}) == 0
Description	Raises when the Ironic API or conductor service is in the DOWN state on all ctl or bmt hosts. For the exact nodes and services, inspect the host and process_name labels of the IronicProcessDown alerts.

Troubleshooting	• Inspect the IronicProcessDown alert for the ironic-api process.
	• Log in to the corresponding node and verify the process status using systemctl status <process_name>.
	• Inspect the log files in the /var/log/ironic/<process_name> directory.
	• Verify the Telegraf monitoring_remote_agent service:
	  • Verify the status of the monitoring_remote_agent service using docker service ls.
	  • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on one of the mon nodes.

Tuning	Not required

Keystone

This section describes the alerts for Keystone.

- KeystoneApiOutage
- KeystoneApiEndpointDown
- KeystoneApiEndpointssDownMajor
- KeystoneApiEndpointsOutage
- KeystoneErrorLogsTooHigh
- KeystoneApiResponseTimeTooHigh
- KeystoneKeysRotationFailure

KeystoneApiOutage

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Keystone API is not accessible for the Keystone endpoint in the OpenStack service catalog.
Raise condition	openstack_api_check_status{name=~"keystone.*"} == 0
Description	Raises when the checks against all available internal Keystone endpoints in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response codes for Keystone are 200 and 300. For a list of all available endpoints, run openstack endpoint list.
Troubleshooting	Verify the availability of internal Keystone endpoints (URLs) from the output of openstack endpoint list.
Tuning	Not required

KeystoneApiEndpointDown

Severity	Minor
Summary	The {{ $labels.name }} endpoint on the {{ $labels.host }} node is not accessible for 2 minutes.

Raise condition	http_response_status{name=~"keystone.*"} == 0
Description	Raises when the check against the Keystone API endpoint does not pass, typically meaning that the service endpoint is down or unreachable due to connectivity issues. Telegraf sends an HTTP request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

KeystoneApiEndpointssDownMajor

Severity	Major
Summary	{{ $value }} {{ $labels.name }} endpoints (>= 50%) are not accessible for 2 minutes.
Raise condition	count by(name) (http_response_status{name=~"keystone.*"} == 0) >= count by(name) (http_response_status{name=~"keystone.*"}) * 0.5
Description	Raises when the check against a Keystone API endpoint does not pass on more than 50% of the ctl nodes, typically indicating that the service endpoint is down or unreachable due to connectivity issues. Telegraf sends an HTTP request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the KeystoneApiEndpointDown for the affected nodes. • Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

KeystoneApiEndpointsOutage

Severity	Critical

Summary	All available {{ $labels.name }} endpoints are not accessible for 2 minutes.
Raise condition	count by(name) (http_response_status{name=~"keystone.*"} == 0) == count by(name) (http_response_status{name=~"keystone.*"})
Description	Raises when the check against a Keystone API endpoint does not pass on all OpenStack controller nodes, typically indicating that the service endpoint is down or unreachable due to connectivity issues. Telegraf sends an HTTP request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the KeystoneApiEndpointDown for the affected nodes. • Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

KeystoneErrorLogsTooHigh

Severity	Warning
Summary	The average per-second rate of errors in the Keystone logs on the {{ $labels.host }} node is {{ $value }} (as measured over the last 5 minutes).
Raise condition	• In 2019.2.10 and prior: sum without(level)(rate(log_messages{level=~"(?i:(error|emergency|fatal))",service="keystone"}[5m])) > 0.2 • In 2019.2.11 and newer: sum(rate(log_messages{service=~"keystone|keystone-wsgi",level=~"(?i:(error|emergency|fatal))"}[5m])) without (level) > {{ log_threshold }}
Description	Raises when the average per-second rate of the error, fatal, or emergency messages in Keystone logs on the node is more than 0.2 per second. Fluentd forwards all logs from Cinder to Elasticsearch and counts the number of log messages per severity. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Inspect the log files in the /var/log/keystone/ directory on the corresponding node.

Tuning	Typically, you should not change the default value. If the alert is constantly firing, inspect the Keystone error logs in Kibana. You can adjust the threshold to an acceptable error rate for a particular environment. In the Prometheus web UI, use the raise condition query to view the appearance rate of a particular message type in logs for a longer period of time and define the best threshold.
	For example, to change the threshold to 0.4:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	   1. Create a file for alert customizations:
	   `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	   `classes:`
	   `- cluster.<cluster_name>.stacklight.custom.alerts`
	   `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        KeystoneErrorLogsTooHigh:`
	`          if: >-`
	`            sum(rate(log_messages{service="keystone", level=~"(?i:\`
	`            (error|emergency|fatal))"}[5m])) without (level) > 0.4`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

KeystoneApiResponseTimeTooHigh

Severity	Warning
Summary	The Keystone API response time for GET and POST requests on the {{ $labels.host }} node is higher than 3 seconds for 2 minutes.
Raise condition	max by(host) (openstack_http_response_times{http_method=~"^(GET|POST)$", http_status=~"^2..$", quantile="0.9", service="keystone"}) >= 3

<table>
<tr><td>Description</td><td>Raises when the GET and POST requests to the Keystone API take more than 3 seconds.</td></tr>
<tr><td>Troubleshooting</td><td>Verify the performance of the OpenStack controller node.</td></tr>
<tr><td>Tuning</td><td>For example, to change the threshold to 5 seconds:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>
Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        KeystoneApiResponseTimeTooHigh:
          if: >-
            max by(host) (openstack_http_response_times{http_method=~"^\
            (GET|POST)$", http_status=~"^2..$", quantile="0.9", \
            service="keystone"}) >= 5</pre>
From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>
Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

KeystoneKeysRotationFailure

Available since the 2019.2.11 maintenance update

Severity	Major

Summary	Keystone keys rotation failure.
Raise condition	increase(log_messages{service="keystone-keys-rotation",level="ERROR"} [2h]) > 0
Description	Raises when a Keystone user failed to rotate fernet or credential keys across the OpenStack control nodes. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Inspect the /var/log/keystone/keystone-rotate.log log on the affected node.
Tuning	Not required

Neutron

This section describes the alerts for Neutron.

- NeutronApiOutage
- NeutronApiEndpointDown
- NeutronApiEndpointsDownMajor
- NeutronApiEndpointsOutage
- NeutronAgentDown
- NeutronAgentsDownMinor
- NeutronAgentsDownMajor
- NeutronAgentsOutage
- NeutronErrorLogsTooHigh

NeutronApiOutage

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Neutron API is not accessible for the Neutron endpoint in the OpenStack service catalog.
Raise condition	openstack_api_check_status{name="neutron"} == 0
Description	Raises when the checks against all available internal Neutron endpoints in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response code for Neutron is 200. For a list of all available endpoints, run openstack endpoint list.
Troubleshooting	Verify the availability of internal Neutron endpoints (URLs) from the output of openstack endpoint list.
Tuning	Not required

NeutronApiEndpointDown

Severity	Minor

Summary	The neutron-api endpoint on the {{ $labels.host }} node is not accessible for 2 minutes.
Raise condition	http_response_status{name="neutron-api"} == 0
Description	Raises when the check against a Neutron API endpoint does not pass, typically indicating that the service endpoint is down or unreachable due to connectivity issues. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

NeutronApiEndpointsDownMajor

Severity	Major
Summary	More than 50% of neutron-api endpoints are not accessible for 2 minutes.
Raise condition	count(http_response_status{name="neutron-api"} == 0) >= count (http_response_status{name="neutron-api"}) * 0.5
Description	Raises when the check against a Neutron API endpoint does not pass on more than 50% of OpenStack controller nodes, typically indicating that the service endpoint is down or unreachable due to connectivity issues. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file. To identify the affected node, see the host label in the NeutronApiEndpointDown alert.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

NeutronApiEndpointsOutage

Severity	Critical

Summary	All available neutron-api endpoints are not accessible for 2 minutes.
Raise condition	count(http_response_status{name="neutron-api"} == 0) == count (http_response_status{name="neutron-api"})
Description	Raises when the check against a Neutron API endpoint does not pass on all OpenStack controller nodes, typically indicating that the service endpoint is down or unreachable due to connectivity issues. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file. To identify the affected node, see the host label in the NeutronApiEndpointDown alert.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf. • Verify the configured URL availability using curl.
Tuning	Not required

NeutronAgentDown

Severity	Minor
Summary	The {{ $labels.binary }} agent on the {{ $labels.hostname }} node is down.
Raise condition	openstack_neutron_agent_state == 0
Description	Raises when a Neutron agent is in the DOWN state, according to the information from the Neutron API. For the list of Neutron services, see Networking service overview. This alert can also indicate issues with the Telegraf monitoring_remote_agent service. The binary and hostname labels contain the name of the agent that is in the DOWN state and the node that hosts the agent.
Troubleshooting	• Verify the statuses of Neutron agents using openstack network agent list. • Verify the status of the monitoring_remote_agent by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NeutronAgentsDownMinor

Severity	Minor
Summary	More than 30% of {{ $labels.binary }} agents are down.
Raise condition	count by(binary) (openstack_neutron_agent_state == 0) >= on(binary) count by(binary) (openstack_neutron_agent_state) * 0.3
Description	Raises when more than 30% of Neutron agents of the same type are in the DOWN state, according to the information from the Neutron API. For the list of Neutron services, see Networking service overview. This alert can also indicate issues with the Telegraf monitoring_remote_agent service. The binary label contains the name of the agent that is in the DOWN state.
Troubleshooting	• Verify the statuses of Neutron agents using openstack network agent list. • Inspect the NeutronAgentDown alert for the nodes and services that are in the DOWN state. • Verify the status of the monitoring_remote_agent by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on one of the mon nodes.
Tuning	Not required

NeutronAgentsDownMajor

Severity	Major
Summary	More than 60% of {{ $labels.binary }} agents are down.
Raise condition	count by(binary) (openstack_neutron_agent_state == 0) >= on(binary) count by(binary) (openstack_neutron_agent_state) * 0.6
Description	Raises when more than 60% of Neutron agents of the same type are in the DOWN state, according to the information from the Neutron API. For the list of Neutron services, see Networking service overview. This alert can also indicate issues with the Telegraf monitoring_remote_agent service. The binary label contains the name of the agent that is in the DOWN state.

Troubleshooting	• Verify the statuses of Neutron agents using openstack network agent list. • Inspect the NeutronAgentDown alert for the nodes and services that are in the DOWN state. • Verify the status of the monitoring_remote_agent by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on one of the mon nodes.
Tuning	Not required

NeutronAgentsOutage

Severity	Critical
Summary	All {{ $labels.binary }} agents are down.
Raise condition	count by(binary) (openstack_neutron_agent_state == 0) == on(binary) count by(binary) (openstack_neutron_agent_state)
Description	Raises when all Neutron agents of the same type are in the DOWN state and unavailable, according to the information from the Neutron API. For the list of Neutron services, see Networking service overview. This alert can also indicate issues with the Telegraf monitoring_remote_agent service. The binary label contains the name of the agent that is in the DOWN state.
Troubleshooting	• Verify the statuses of Neutron agents using openstack network agent list. • Inspect the NeutronAgentDown alert for the nodes and services that are in the DOWN state. • Verify the status of the monitoring_remote_agent by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on one of the mon nodes.
Tuning	Not required

NeutronErrorLogsTooHigh

Severity	Warning
Summary	The average per-second rate of errors in Neutron logs on the {{ $labels.host }} node is {{ $value }} (as measured over the last 5 minutes).

Raise condition	sum without(level) (rate(log_messages{level=~"(?i:(error|emergency|fatal))",service="neutron"}[5m])) > 0.2
Description	Raises when the average per-second rate of the error, fatal, or emergency messages in Neutron logs on the node is more than 0.2 per second. Fluentd forwards all logs from Neutron to Elasticsearch and counts the number of log messages per severity. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Inspect the Neutron logs in the /var/log/neutron/ directory on the affected node.

Tuning	Typically, you should not change the default value. If the alert is constantly firing, inspect the Neutron error logs in the Kibana web UI. However, you can adjust the threshold to an acceptable error rate for a particular environment. In the Prometheus web UI, use the raise condition query to view the appearance rate of a particular message type in logs for a longer period of time and define the best threshold.
	For example, to change the threshold to 0.4:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	  `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	  `classes:`
	  `- cluster.<cluster_name>.stacklight.custom.alerts`
	  `...`
	2. In the defined alert customizations file, modify the alert by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        NeutronErrorLogsTooHigh:`
	`          if: >-`
	`            sum(rate(log_messages{service="neutron", level=~"(?i:\`
	`            (error|emergency|fatal))"}[5m])) without (level) > 0.4`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

Nova

This section describes the alerts for Nova.

Nova service

This section describes the Nova API and services alerts.

- NovaApiOutage
- NovaApiDown
- NovaApiEndpointDown
- NovaApiEndpointsDownMajor
- NovaApiEndpointsOutage
- NovaServiceDown
- NovaServicesDownMinor
- NovaComputeServicesDownMinor
- NovaServicesDownMajor
- NovaComputeServicesDownMajor
- NovaServiceOutage
- NovaErrorLogsTooHigh
- NovaComputeSystemLoadTooHighWarning
- NovaComputeSystemLoadTooHighCritical

NovaApiOutage

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Nova API is not accessible for all available Nova endpoints in the OpenStack service catalog.
Raise condition	max(openstack_api_check_status{name=~"nova.*|placement"}) == 0
Description	Raises when the checks against all available internal Nova or placement endpoints in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response codes are 200 for nova and nova20, 200 and 401 for placement. For a list of all available endpoints, run openstack endpoint list.
Troubleshooting	• Verify the states of Nova endpoints from the output of openstack endpoint list. • Inspect the NovaApiDown alert for the nodes and services that are in the DOWN state. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NovaApiDown

Removed since the 2019.2.11 maintenance update

Severity	Major
Summary	Nova API is not accessible for the {{ $labels.name }} endpoint.
Raise condition	openstack_api_check_status{name=~"nova.*|placement"} == 0
Description	Raises when the checks against one of the available internal Nova or placement endpoint in the OpenStack service catalog do not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response codes are 200 for nova and nova20, 200 and 401 for placement. For a list of all available endpoints, run openstack endpoint list.
Troubleshooting	• Verify the states of Nova endpoints from the output of openstack endpoint list. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NovaApiEndpointDown

Severity	Minor
Summary	The nova-api endpoint on the {{ $labels.host }} node is not accessible for 2 minutes.
Raise condition	http_response_status{name=~"nova-api"} == 0
Description	Raises when the check against a Nova API endpoint does not pass on a node. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.
Troubleshooting	• Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf/. • Verify the configured URL availability using curl.

Tuning	Not required

NovaApiEndpointsDownMajor

Severity	Major
Summary	{{ $value }} nova-api endpoints (>= 0.5 * 100) are not accessible for 2 minutes.
Raise condition	count(http_response_status{name=~"nova-api"} == 0) >= count(http_response_status{name=~"nova-api"}) * 0.5
Description	Raises when the check against a Nova API endpoint does not pass on more than 50% of OpenStack controller nodes. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected 200 response code and actual HTTP response codes. For details, see HTTP response input plugin.
Troubleshooting	• Inspect the NovaApiEndpointDown alert for the nodes and services that are in the DOWN state. • Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf/. • Verify the configured URL availability using curl.
Tuning	Not required

NovaApiEndpointsOutage

Severity	Critical
Summary	All available nova-api endpoints are not accessible for 2 minutes.
Raise condition	count(http_response_status{name=~"nova-api"} == 0) == count (http_response_status{name=~"nova-api"})
Description	Raises when the check against a Nova API endpoint does not pass on all OpenStack controller nodes. Telegraf sends a request to the URL configured in /etc/telegraf/telegraf.d/input-http_response.conf on the corresponding node and compares the expected and actual HTTP response codes from the configuration file.

Troubleshooting	• Inspect the NovaApiEndpointDown alert for the nodes and services that are in the DOWN state. • Inspect the Telegraf logs using journalctl -u telegraf or in /var/log/telegraf/. • Verify the configured URL availability using curl.
Tuning	Not required

NovaServiceDown

Severity	Minor
Summary	The {{ $labels.binary }} service on the {{ $labels.hostname }} node is down.
Raise condition	openstack_nova_service_state == 0
Description	Raises when the Nova controller or compute service (os-service) is in the DOWN state, according to the data from Nova API. For details, see Compute services and Compute service overview. The binary and hostname labels in the raised alert contain the service name that is in the DOWN state and the affected node name.
Troubleshooting	• Verify the states of Nova services from the output of the openstack compute service list command. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NovaServicesDownMinor

Severity	Minor
Summary	{{ $value }} {{ $labels.binary }} services (>=0.3 * 100%) are down.
Raise condition	count(openstack_nova_service_state{binary!~"nova-compute"} == 0) by (binary) >= on (binary) count (openstack_nova_service_state{binary!~"nova-compute"}) by (binary) * 0.3

Description	Raises when more than 30% of Nova controller services of the same type are in the DOWN state, according to the data from Nova API. For details, see Compute services and Compute service overview.
Troubleshooting	• Inspect the NovaServiceDown alert for the nodes and services that are in the DOWN state. • Verify the states of Nova services from the output of the openstack compute service list command. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NovaComputeServicesDownMinor

Severity	Minor
Summary	{{ $value }} nova-compute services (>= 0.25 * 100%) are down.
Raise condition	count(openstack_nova_service_state{binary="nova-compute"} == 0) >= count(openstack_nova_service_state{binary="nova-compute"}) * 0.25
Description	Raises when more than 25% of Nova compute services are in the DOWN state, according to the data from Nova API. For details, see Compute services and Compute service overview.
Troubleshooting	• Inspect the NovaServiceDown alert for the nodes and services that are in the DOWN state. • Verify the states of Nova services from the output of the openstack compute service list command. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NovaServicesDownMajor

Severity	Major
Summary	{{ $value }} {{ $labels.binary }} services (>= 0.25 * 100%) are down.
Raise condition	count(openstack_nova_service_state{binary!~"nova-compute"} == 0) by (binary) >= on (binary)count(openstack_nova_service_state {binary!~"nova-compute"}) by (binary) * 0.6
Description	Raises when more than 60% of Nova controller services of the same type are in the DOWN state, according to the data from Nova API. For details, see Compute services and Compute service overview.
Troubleshooting	• Inspect the NovaServiceDown alert for the nodes and services that are in the DOWN state. • Verify the states of Nova services from the output of the openstack compute service list command. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NovaComputeServicesDownMajor

Severity	Major
Summary	{{ $value }} nova-compute services (>= 0.5 * 100%) are down.
Raise condition	count(openstack_nova_service_state{binary="nova-compute"} == 0) >= count(openstack_nova_service_state{binary="nova-compute"}) * 0.5
Description	Raises when more than 50% of Nova compute services are in the DOWN state, according to the data from Nova API. For details, see Compute services and Compute service overview.

Troubleshooting	• Inspect the NovaServiceDown alert for the nodes and services that are in the DOWN state. • Verify the states of Nova services from the output of the openstack compute service list command. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NovaServiceOutage

Severity	Critical
Summary	All {{ $labels.binary }} services are down.
Raise condition	count(openstack_nova_service_state == 0) by (binary) == on (binary) count(openstack_nova_service_state) by (binary)
Description	Raises when Nova controller or compute services of the same type are in the DOWN state, according to the data from Nova API. For details, see Compute services and Compute service overview. The binary and hostname labels in the raised alert contain the service name that is in the DOWN state and the affected node name.
Troubleshooting	• Verify the states of Nova services from the output of the openstack compute service list command. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Not required

NovaErrorLogsTooHigh

Severity	Warning
Summary	The average per-second rate of errors in the Nova logs on the {{ $labels.host }} node is more than 0.2 messages per second (as measured over the last 5 minutes).

<table>
<tr><td>Raise condition</td><td>sum(rate(log_messages{service="nova",level=~"(?i:(error|emergency|fatal))"}[5m])) without (level) > 0.2</td></tr>
<tr><td>Description</td><td>Raises when the average per-second rate of the error, fatal, or emergency messages in Nova logs on the node is more than 0.2 per second. Fluentd forwards all logs from Nova to Elasticsearch and counts the number of log messages per severity. The host label in the raised alert contains the name of the affected node.</td></tr>
<tr><td>Troubleshooting</td><td>Inspect the log files in the /var/log/nova/ directory of the affected node.</td></tr>
<tr><td>Tuning</td><td>Typically, you should not change the default value. If the alert is constantly firing, inspect the Nova error logs in the Kibana web UI. However, you can adjust the threshold to an acceptable error rate for a particular environment. In the Prometheus Web UI, use the raise condition query to view the appearance rate of a particular message type in logs for a longer period of time and define the best threshold.
For example, to change the threshold to 0.4:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

  1. Create a file for alert customizations:

    touch cluster/<cluster_name>/stacklight/custom/alerts.yml

  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

    classes:
    - cluster.<cluster_name>.stacklight.custom.alerts
    ...

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

  parameters:
    prometheus:
      server:
        alert:
          NovaErrorLogsTooHigh:
            if: >-
              sum(rate(log_messages{service="nova", level=~"(?i:\
              (error|emergency|fatal))"}[5m])) without (level) > 0.4

3. From the Salt Master node, apply the changes:

  salt 'I@prometheus:server' state.sls prometheus.server

4. Verify the updated alert definition in the Prometheus web UI.</td></tr>
</table>

NovaComputeSystemLoadTooHighWarning

Available starting from the 2019.2.9 maintenance update

Severity	Warning
Summary	The system load per CPU on the {{ $labels.host }} node is more than 1 for 5 minutes.
Raise condition	system_load15{host=~".*cmp[0-9]+"} / system_n_cpus > 1.0
Description	Raises when the average load on an OpenStack compute node is higher than 1 per CPU core over the last 5 minutes, indicating that the system is overloaded, many processes are waiting for CPU time. The host label in the raised alert contains the name of the affected node.
Troubleshooting	Inspect the output of the uptime and top commands on the affected node.

<table>
<tr>
<td>Tuning</td>
<td>
For example, to change the threshold to 1.5 per core:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre><code>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</code></pre>

Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre><code>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</code></pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre><code>parameters:
  prometheus:
    server:
      alert:
        NovaComputeSystemLoadTooHighWarning:
          if: >-
            system_load15{host=~".*cmp[0-9]+"} / system_n_cpus > 1.5</code></pre>

From the Salt Master node, apply the changes:
<pre><code>salt 'I@prometheus:server' state.sls prometheus.server</code></pre>

Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

NovaComputeSystemLoadTooHighCritical

Available starting from the 2019.2.9 maintenance update

Severity	Critical
Summary	The system load per CPU on the {{ $labels.host }} node is more than 2 for 5 minutes.
Raise condition	system_load15{host=~".*cmp[0-9]+"} / system_n_cpus > 2.0
Description	Raises when the average load on an OpenStack compute node is higher than 2 per CPU over the last 5 minutes, indicating that the system is overloaded, many processes are waiting for CPU time. The host label in the raised alert contains the name of the affected node.

<table>
<tr><td>Troubleshooting</td><td>Inspect the output of the uptime and top commands on the affected node.</td></tr>
<tr><td>Tuning</td><td>

For example, to change the threshold to 3 per core:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

    1. Create a file for alert customizations:

```
touch cluster/<cluster_name>/stacklight/custom/alerts.yml
```

    2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

```
classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...
```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

```
parameters:
  prometheus:
    server:
      alert:
        NovaComputeSystemLoadTooHighCritical:
          if: >-
            system_load15{host=~".*cmp[0-9]+"} / system_n_cpus > 3
```

3. From the Salt Master node, apply the changes:

```
salt 'I@prometheus:server' state.sls prometheus.server
```

4. Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

Nova resources

This section describes the alerts for Nova resources consumption.

> **Warning**
>
> The following set of alerts has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, disable these alerts as described in Manage alerts.

- NovaHypervisorVCPUsFullMinor
- NovaHypervisorVCPUsFullMajor
- NovaHypervisorMemoryFullMajor
- NovaHypervisorMemoryFullCritical
- NovaHypervisorDiskFullMajor
- NovaHypervisorDiskFullCritical
- NovaAggregateMemoryFullMajor
- NovaAggregateMemoryFullCritical
- NovaAggregateDiskFullMajor
- NovaAggregateDiskFullCritical
- NovaTotalVCPUsFullMinor
- NovaTotalVCPUsFullMajor
- NovaTotalMemoryFullMajor
- NovaTotalMemoryFullCritical
- NovaTotalDiskFullMajor
- NovaTotalDiskFullCritical

NovaHypervisorVCPUsFullMinor

Removed since the 2019.2.4 maintenance update

Severity	Minor
Summary	{{ $value }} VCPUs on the {{ $labels.hostname }} node (>= {{ cpu_minor_threshold * 100 }}%) are used.
Raise condition	label_replace(system_load15, "hostname", "$1", "host", "(.*)") > on (hostname) openstack_nova_vcpus * 0.85
Description	Raises when the hypervisor consumes more than 85% of the available VCPU (the average load for 15 minutes), according to the data from Nova API and the load average from /proc/loadavg on the appropriate node. For details, see Hypervisors. The hostname label in the raised alert contains the affected node name.

Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaHypervisorVCPUsFullMajor

Removed since the 2019.2.4 maintenance update

Severity	Major
Summary	{{ $value }} VCPUs on the {{ $labels.hostname }} node (>= {{ cpu_major_threshold * 100 }}%) are used.
Raise condition	label_replace(system_load15, "hostname", "$1", "host", "(.*)") > on (hostname) openstack_nova_vcpus * 0.95
Description	Raises when the hypervisor consumes more than 95% of the available VCPU (the average load for 15 minutes), according to the data from Nova API and the load average from /proc/loadavg on the appropriate node. For details, see Hypervisors. The hostname label in the raised alert contains the name of the affected node.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaHypervisorMemoryFullMajor

Removed since the 2019.2.4 maintenance update

Severity	Major
Summary	{{ $value }}MB of RAM on the {{ $labels.hostname }} node (>= {{ ram_major_threshold * 100 }}%) is used.

Raise condition	openstack_nova_used_ram > openstack_nova_ram * 0.85
Description	Raises when the hypervisor allocates more than 85% of the available RAM, according to the data from Nova API. For details, see Hypervisors. The hostname label in the raised alert contains the name of the affected node.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaHypervisorMemoryFullCritical

Removed since the 2019.2.4 maintenance update

Severity	Critical
Summary	{{ $value }}MB of RAM on the {{ $labels.hostname }} node (>= {{ ram_critical_threshold * 100 }}%) is used.
Raise condition	openstack_nova_used_ram > openstack_nova_ram * 0.95
Description	Raises when the hypervisor allocates more than 95% of the available RAM, according to the data from Nova API. For details, see Hypervisors. The hostname label in the raised alert contains the name of the affected node.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaHypervisorDiskFullMajor

Removed since the 2019.2.4 maintenance update

Severity	Major
Summary	{{ $value }}GB of disk space on the {{ $labels.hostname }} node (>= {{ disk_major_threshold * 100 }}%) is used.
Raise condition	openstack_nova_used_disk > openstack_nova_disk * 0.85
Description	Raises when the hypervisor allocates more than 85% of the available disk space, according to the data from Nova API. For details, see Hypervisors. The hostname label in the raised alert contains the name of the affected node.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaHypervisorDiskFullCritical

Removed since the 2019.2.4 maintenance update

Severity	Critical
Summary	{{ $value }}GB of disk space on the {{ $labels.hostname }} node (>= {{ disk_critical_threshold *100 }}%) is used.
Raise condition	openstack_nova_used_disk > openstack_nova_disk * 0.95
Description	Raises when the hypervisor allocates more than 95% of the available disk space, according to the data from Nova API. For details, see Hypervisors. The hostname label in the raised alert contains the name of the affected node.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaAggregateMemoryFullMajor

Removed since the 2019.2.4 maintenance update

Severity	Major
Summary	{{ $value }}MB of RAM on the {{ $labels.aggregate }} aggregate is used (at least {{ ram_major_threshold * 100}}%).
Raise condition	openstack_nova_aggregate_used_ram > openstack_nova_aggregate_ram * 0.85
Description	Raises when the RAM allocation over all hypervisors within a host aggregate is more than 85% of the total available RAM, according to the data from Nova API. For details, see Hypervisors and Host aggregates. The aggregate label in the raised alert contains the name of the affected host aggregate.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the list of host aggregate members using the openstack aggregate list and openstack aggregate show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaAggregateMemoryFullCritical

Removed since the 2019.2.4 maintenance update

Severity	Critical
Summary	{{ $value }}MB of RAM on the {{ $labels.aggregate }} aggregate (>= {{ ram_critical_threshold * 100 }}%) is used.
Raise condition	openstack_nova_aggregate_used_ram > openstack_nova_aggregate_ram * 0.95
Description	Raises when the RAM allocation over all hypervisors within a host aggregate is more than 95% of the total available RAM, according to the data fromNova API. For details, see Hypervisors and Host aggregates. The aggregate label in the raised alert contains the name of the affected host aggregate.

Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the list of host aggregate members using the openstack aggregate list and openstack aggregate show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaAggregateDiskFullMajor

Removed since the 2019.2.4 maintenance update

Severity	Major
Summary	{{ $value }}GB of disk space on the {{ $labels.aggregate }} aggregate (>= {{ disk_major_threshold *100 }}%) is used.
Raise condition	openstack_nova_aggregate_used_disk > openstack_nova_aggregate_disk * 0.85
Description	Raises when the disk space allocation over all hypervisors within a host aggregate is more than 95% of the total available disk space, according to the data from Nova API. For details, see Hypervisors and Host aggregates. The aggregate label in the raised alert contains the name of the affected host aggregate.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the list of host aggregate members using the openstack aggregate list and openstack aggregate show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaAggregateDiskFullCritical

Removed since the 2019.2.4 maintenance update

Severity	Critical
Summary	{{ $value }}GB of disk space on the {{ $labels.aggregate }} aggregate (>= {{ disk_critical_threshold *100 }}%) is used.
Raise condition	openstack_nova_aggregate_used_disk > openstack_nova_aggregate_disk * 0.95
Description	Raises when the disk space allocation over all hypervisors within a host aggregate is more than 95% of the total available disk space over all hypervisors within the host aggregate, according to the data from Nova API. For details, see Hypervisors and Host aggregates. The aggregate label in the raised alert contains the name of the affected host aggregate.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the list of host aggregate members using the openstack aggregate list and openstack aggregate show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaTotalVCPUsFullMinor

Removed since the 2019.2.4 maintenance update

Severity	Minor
Summary	{{ $value }} VCPUs in the cloud (>= {{ cpu_minor_threshold * 100 }}%) are used.
Raise condition	sum(label_replace(system_load15, "hostname", "$1", "host", "(.*)") and on (hostname) openstack_nova_vcpus) > max(sum(openstack_nova_vcpus) by (instance)) * 0.85
Description	Raises when the VCPU consumption over all hypervisors (the average load for 15 minutes) is more than 85% of the total available VCPU, according to the data from Nova API and /proc/loadavg on the appropriate node. For details, see Hypervisors.

Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaTotalVCPUsFullMajor

Removed since the 2019.2.4 maintenance update

Severity	Major
Summary	{{ $value }} VCPUs in the cloud (>= {{cpu_major_threshold * 100 }}%) are used.
Raise condition	sum(label_replace(system_load15, "hostname", "$1", "host", "(.*)") and on (hostname) openstack_nova_vcpus) > max(sum(openstack_nova_vcpus) by (instance)) * 0.95
Description	Raises when the VCPU consumption over all hypervisors (the average load for 15 minutes) is more than 95% of the total available VCPU, according to the data from Nova API and /proc/loadavg on the appropriate node. For details, see Hypervisors.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaTotalMemoryFullMajor

Removed since the 2019.2.4 maintenance update

Severity	Major
Summary	{{ $value }}MB of RAM in the cloud (>= {{ram_major_threshold * 100}}%) is used.

Raise condition	openstack_nova_total_used_ram > openstack_nova_total_ram * 0.85
Description	Raises when the RAM allocation over all hypervisors is more than 85% of the total available RAM, according to the data from Nova API. For details, see Hypervisors.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaTotalMemoryFullCritical

Removed since the 2019.2.4 maintenance update

Severity	Critical
Summary	{{ $value }}MB of RAM in the cloud (>= {{ram_critical_threshold * 100 }}%) is used.
Raise condition	openstack_nova_total_used_ram > openstack_nova_total_ram * 0.95
Description	Raises when the RAM allocation over all hypervisors is more than 95% of the total available RAM, according to the data from Nova API. For details, see Hypervisors.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaTotalDiskFullMajor

Removed since the 2019.2.4 maintenance update

Severity	Major

Summary	{{ $value }}GB of disk space in the cloud (>= {{disk_major_threshold * 100 }}%) is used.
Raise condition	openstack_nova_total_used_disk > openstack_nova_total_disk * 0.85
Description	Raises when the disk space allocation over all hypervisors is more than 85% of the total disk space, according to the data from Nova API. For details, see Hypervisors.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

NovaTotalDiskFullCritical

Removed since the 2019.2.4 maintenance update

Severity	Critical
Summary	{{ $value }}GB of disk space in the cloud (>= {{disk_critical_threshold * 100 }}%) is used.
Raise condition	openstack_nova_total_used_disk > openstack_nova_total_disk * 0.95
Description	Raises when the disk space allocation over all hypervisors is more than 95% of the total disk space, according to the data from Nova API. For details, see Hypervisors.
Troubleshooting	• Verify the hypervisor capacity using the openstack hypervisor list or openstack hypervisor show commands. • Verify the status of the monitoring_remote_agent service by running docker service ls on a mon node. • Inspect the monitoring_remote_agent service logs by running docker service logs monitoring_remote_agent on a mon node.
Tuning	Disable the alert as described in Manage alerts.

Octavia

This section describes the alerts for Octavia.

- OctaviaApiDown
- OctaviaErrorLogsTooHigh

OctaviaApiDown

Removed since the 2019.2.11 maintenance update

Severity	Critical
Summary	Octavia API is not accessible for all available Octavia endpoints in the OpenStack service catalog for 2 minutes.
Raise condition	max(openstack_api_check_status{service="octavia-api"}) == 0
Description	Raises when the checks against one available internal Octavia endpoint in the OpenStack service catalog does not pass. Telegraf sends HTTP requests to the URLs from the OpenStack service catalog and compares the expected and actual HTTP response codes. The expected response code for Octavia is 200. For a list of all available endpoints, run openstack endpoint list.
Troubleshooting	Verify the availability of internal Octavia endpoints (URLs) from the output of the openstack endpoint list command.
Tuning	Not required

OctaviaErrorLogsTooHigh

Severity	Warning
Summary	The average per-second rate of errors in Octavia logs on the {{ $labels.host }} node is more than 0.2 error messages per second (as measured over the last 5 minutes).
Raise condition	sum(rate(log_messages{service="octavia",level=~"error|emergency|fatal"}[5m])) without (level) > 0.2
Description	Raises when the average per-second rate of the error, fatal, or emergency messages in the Octavia logs on the node is more than 0.2 per second. Fluentd forwards all logs from Octavia to Elasticsearch and counts the number of log messages per severity. The host label in the raised alert contains the host name of the affected node.

Troubleshooting	Inspect the log files in the /var/log/octavia/ directory on the affected node.

Tuning	Typically, you should not change the default value. If the alert is constantly firing, inspect the Octavia error logs in the Kibana web UI. However, you can adjust the threshold to an acceptable error rate for a particular environment. In the Prometheus Web UI, use the raise condition query to view the appearance rate of a particular message type in logs for a longer period of time and define the best threshold.
	For example, to change the threshold to 0.4:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        OctaviaErrorLogsTooHigh:`
	`          if: >-`
	`            sum(rate(log_messages{service=""octavia"", level=~""(?i:\`
	`            (error|emergency|fatal))""}[5m])) without (level) > 0.4`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

Kubernetes

This section describes the alerts available for a Kubernetes cluster.

Calico

This section describes the alerts for Calico.

- CalicoProcessDown
- CalicoProcessDownMinor
- CalicoProcessDownMajor
- CalicoProcessOutage

CalicoProcessDown

Severity	Minor
Summary	The Calico {{ $labels.process_name }} process on the {{ $labels.host }} node is down for 2 minutes.
Raise condition	procstat_running{process_name=~"calico-felix|bird|bird6|confd"} == 0
Description	Raises when Telegraf cannot find running processes with names calico-felix, bird, bird6, confd on any ctl host. The process_name and host labels in the raised alert contain the name of a particular process and the host name of the affected node respectively.
Troubleshooting	• Inspect dmesg and /var/log/kern.log • Inspect the logs in /var/log/calico • Inspect the output of the systemctl status containerd and journalctl -u containerd commands
Tuning	Not required

CalicoProcessDownMinor

Severity	Minor
Summary	More than 30% of Calico {{ $labels.process_name }} processes are down for 2 minutes.
Raise condition	count(procstat_running{process_name=~"calico-felix|bird|bird6|confd"} == 0) by (process_name) > count(procstat_running{process_name=~"calico-felix|bird|bird6|confd"}) by (process_name) * {{ instance_minor_threshold_percent }}
Description	Raises when Telegraf cannot find running processes with names calico-felix, bird, bird6, confd on more than 30% of the ctl hosts. The process_name label in the raised alert contains the name of a particular process.

Troubleshooting	• Inspect the CalicoProcessDown alerts for the host names of the affected nodes • Inspect dmesg and /var/log/kern.log • Inspect the logs in /var/log/calico • Inspect the output of the systemctl status containerd and journalctl -u containerd commands
Tuning	Not required

CalicoProcessDownMajor

Severity	Major
Summary	More than 60% of Calico {{ $labels.process_name }} processes are down for 2 minutes.
Raise condition	count(procstat_running{process_name=~"calico-felix|bird|bird6|confd"} == 0) by (process_name) > count(procstat_running{process_name=~"calico-felix|bird|bird6|confd"}) by (process_name) * {{ instance_major_threshold_percent }}
Description	Raises when Telegraf cannot find running processes with names calico-felix, bird, bird6, confd on more than 60% of ctl hosts. The process_name label in the raised alert contains the name of a particular process.
Troubleshooting	• Inspect the CalicoProcessDown alerts for host names of the affected nodes • Inspect dmesg and /var/log/kern.log • Inspect the logs in /var/log/calico • Inspect the output of the systemctl status containerd and journalctl -u containerd commands
Tuning	Not required

CalicoProcessOutage

Severity	Critical
Summary	All Calico {{ $labels.process_name }} processes are down for 2 minutes.
Raise condition	count(procstat_running{process_name=~"calico-felix|bird|bird6|confd"}) by (process_name) == count(procstat_running{process_name=~"calico-felix|bird|bird6|confd"} == 0) by (process_name)

Description	Raises when Telegraf cannot find running processes with names calico-felix, bird, bird6, confd on all ctl hosts. The process_name label in the raised alert contains the name of a particular process.
Troubleshooting	• Verify the CalicoProcessDown alerts for host names of the affected nodes • Inspect dmesg and /var/log/kern.log • Inspect the logs in /var/log/calico • Inspect the output of the systemctl status containerd and journalctl -u containerd commands
Tuning	Not required

etcd

This section describes the alerts for the etcd service.

- EtcdRequestFailureTooHigh
- EtcdInstanceNoLeader
- EtcdServiceDownMinor
- EtcdServiceDownMajor
- EtcdServiceOutage

EtcdRequestFailureTooHigh

Severity	Minor
Summary	More than 1% of HTTP {{ $labels.method }} requests to the etcd API failed on the {{ $labels.instance }} instance.
Raise condition	sum by(method) (rate(etcd_http_failed_total[5m])) / sum by(method) (rate(etcd_http_received_total[5m])) > 0.01
Description	Raises when the total percentage rate of failed HTTP requests from the client to etcd is higer than 1%. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the etcd service status on the affected node using systemctl status etcd. • Inspect the etcd service logs using journalctl -u etcd.

©2025, Mirantis Inc.

Tuning	For example, to change the threshold to 2%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        EtcdRequestFailureTooHigh:`
	`          if: >-`
	`            sum by(method) (rate(etcd_http_failed_total[5m])) / sum\`
	`            by(method) (rate(etcd_http_received_total[5m])) > 0.02`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

EtcdInstanceNoLeader

Severity	Major
Summary	The etcd {{ $labels.instance }} instance has no leader.
Raise condition	etcd_server_has_leader != 1
Description	Raises when the etcd server reports that it has no leader.

Troubleshooting	Verify all etcd services on the ctl nodes: • Verify the etcd service status using systemctl status etcd. • Inspect the etcd service logs using journalctl -u etcd.
Tuning	Not required

EtcdServiceDownMinor

Severity	Minor
Summary	The etcd {{ $labels.instance }} instance is down for 2 minutes.
Raise condition	up{job='etcd'} == 0
Description	Raises when Prometheus fails to scrape the etcd target. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the availability of the etcd target on the mon nodes. To obtain the address of a target, navigate to Status -> Targets -> etcd in the Prometheus web UI. • Verify the etcd service status on the affected nodes using systemctl status etcd and journalctl -u etcd.
Tuning	Not required

EtcdServiceDownMajor

Severity	Major
Summary	More than 30% of etcd instances are down for 2 minutes.
Raise condition	count(up{job='etcd'} == 0) > count(up{job='etcd'}) * 0.3
Description	Raises when Prometheus fails to scrape more than 30% of etcd targets. Inspect the EtcdServiceDownMinor alerts for the host names of the affected nodes.
Troubleshooting	• Verify the availability of the etcd target on the mon nodes. To obtain the address of a target, navigate to Status -> Targets -> etcd in the Prometheus web UI. • Verify the etcd service status on the affected nodes using systemctl status etcd and journalctl -u etcd.

Tuning	Not required

EtcdServiceOutage

Severity	Critical
Summary	All etcd services within the cluster are down.
Raise condition	count(up{job='etcd'} == 0) == count(up{job='etcd'})
Description	Raises when Prometheus fails to scrape all etcd targets. Inspect the EtcdServiceDownMinor alerts for the host names of the affected nodes.
Troubleshooting	• Verify the availability of the etcd target on the mon nodes. To obtain the address of a target, navigate to Status -> Targets -> etcd in the Prometheus web UI. • Verify the etcd service status on the affected nodes using systemctl status etcd and journalctl -u etcd.
Tuning	Not required

Kubernetes

This section describes the alerts for Kubernetes.

- ContainerScrapeError
- KubernetesProcessDown
- KubernetesProcessDownMinor
- KubernetesProcessDownMajor
- KubernetesProcessOutage

ContainerScrapeError

Severity	Warning
Summary	Prometheus was not able to scrape metrics from the container on the {{ $labels.instance }} Kubernetes instance.
Raise condition	container_scrape_error != 0
Description	Raises when cadvisor fails to scrape metrics from a container.
Tuning	Not required

KubernetesProcessDown

Severity	Minor
Summary	The Kubernetes {{ $labels.process_name }} process on the {{ $labels.host }} node is down for 2 minutes.
Raise condition	procstat_running{process_name=~"hyperkube-.*"} == 0
Description	Raises when Telegraf cannot find running hyperkube-kubelet, hyperkube-proxy, hyperkube-apiserver, hyperkube-controller-manager, and hyperkube-scheduler processes on any ctl host and hyperkube-kubelet or hyperkube-proxy processes on any cmp host. The process_name label in the raised alert contains the process name.

Troubleshooting	• Verify the containerd status on the affected node using systemctl containerd status. • Verify the Docker status on the affected node using systemctl docker status. • For issues with cmp, verify criproxy using systemctl criproxy status. • Inspect the logs in /var/log/kubernetes.log.
Tuning	Not required

KubernetesProcessDownMinor

Severity	Minor
Summary	{{ $value }} Kubernetes {{ $labels.process_name }} processes (>= {{ instance_minor_threshold_percent * 100}}%) are down for 2 minutes.
Raise condition	count(procstat_running{process_name=~"hyperkube-.*"} == 0) by (process_name) > count(procstat_running{process_name=~"hyperkube-.*"}) by (process_name) * {{ instance_minor_threshold_percent }}
Description	Raises when Telegraf cannot find running hyperkube-kubelet, hyperkube-proxy, hyperkube-apiserver, hyperkube-controller-manager, and hyperkube-scheduler processes on more than 30% of the ctl or cmp hosts. The process_name label in the raised alert contains the process name. For the affected nodes, see the host label in the KubernetesProcessDown alerts.
Troubleshooting	• Verify the containerd status on the affected node using systemctl containerd status. • Verify the Docker status on the affected node using systemctl docker status. • For issues with cmp, verify criproxy using systemctl criproxy status. • Inspect the logs in /var/log/kubernetes.log.
Tuning	Not required

KubernetesProcessDownMajor

Severity	Major
Summary	{{ $value }} Kubernetes {{ $labels.process_name }} processes (>= {{ instance_major_threshold_percent * 100}}%) are down for 2 minutes.
Raise condition	count(procstat_running{process_name=~"hyperkube-.*"} == 0) by (process_name) > count(procstat_running{process_name=~"hyperkube-.*"}) by (process_name) * {{ instance_major_threshold_percent }}

Description	Raises when Telegraf cannot find running hyperkube-kubelet, hyperkube-proxy, hyperkube-apiserver, hyperkube-controller-manager, and hyperkube-scheduler processes on more than 60% of the ctl or cmp hosts. The process_name label in the raised alert contains the process name. For the affected nodes, see the host label in the KubernetesProcessDown alerts.
Troubleshooting	• Verify the containerd status on the affected node using systemctl containerd status. • Verify the Docker status on the affected node using systemctl docker status. • For issues with cmp, verify criproxy using systemctl criproxy status. • Inspect the logs in /var/log/kubernetes.log.
Tuning	Not required

KubernetesProcessOutage

Severity	Critical
Summary	All Kubernetes {{ $labels.process_name }} processes are down for 2 minutes.
Raise condition	count(procstat_running{process_name=~"hyperkube-.*"}) by (process_name) == count(procstat_running{process_name=~"hyperkube-.*"} == 0) by (process_name)
Description	Raises when Telegraf cannot find running hyperkube-kubelet, hyperkube-proxy, hyperkube-apiserver, hyperkube-controller-manager, and hyperkube-scheduler processes on all ctl and cmp hosts. The process_name label in the raised alert contains the process name.
Troubleshooting	• Verify the containerd status on the affected node using systemctl containerd status. • Verify the Docker status on the affected node using systemctl docker status. • For issues with cmp, verify criproxy using systemctl criproxy status. • Inspect the logs in /var/log/kubernetes.log.
Tuning	Not required

OpenContrail

This section describes the alerts available for OpenContrail.

Cassandra

This section describes the alerts for the Cassandra service.

- CassandraServiceDown
- CassandraServiceDownMinor
- CassandraServiceDownMajor
- CassandraServiceOutage

CassandraServiceDown

Severity	Minor
Summary	The Cassandra service on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="cassandra-server"} == 0
Description	Raises when Telegraf cannot find running cassandra-server processes on the ntw or nal hosts. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Inspect the Cassandra logs in the /var/log/cassandra/ directory on the affected node.
Tuning	Not required

CassandraServiceDownMinor

Severity	Minor
Summary	More than 30% of Cassandra services are down.
Raise condition	count(procstat_running{process_name="cassandra-server"} == 0) >= count(procstat_running{process_name="cassandra-server"}) *{{ monitoring.services_failed_warning_threshold_percent }}
Description	Raises when Telegraf cannot find running cassandra-server processes on more than 30% of ntw and nal hosts.

Troubleshooting	• Inspect the CassandraServiceDown alert for the host names of the affected nodes. • Inspect the Cassandra logs in /var/log/cassandra/.
Tuning	Not required

CassandraServiceDownMajor

Severity	Major
Summary	More than 60% of Cassandra services are down.
Raise condition	count(procstat_running{process_name="cassandra-server"} == 0) >= count(procstat_running{process_name="cassandra-server"}) *{{ monitoring.services_failed_critical_threshold_percent }}
Description	Raises when Telegraf cannot find running cassandra-server processes on more than 60% of ntw and nal hosts.
Troubleshooting	• Inspect the CassandraServiceDown alert for the host names of the affected nodes. • Inspect the Cassandra logs in /var/log/cassandra/.
Tuning	Not required

CassandraServiceOutage

Severity	Critical
Summary	All Cassandra services are down.
Raise condition	count(procstat_running{process_name="cassandra-server"} == 0) == count(procstat_running{process_name="cassandra-server"})
Description	Raises when Telegraf cannot find running cassandra-server processes on all ntw and nal hosts.
Troubleshooting	• Inspect the CassandraServiceDown alert for the host names of the affected nodes. • Inspect the Cassandra logs in /var/log/cassandra/.
Tuning	Not required

Kafka

This section describes the alerts for the Kafka service.

- KafkaServiceDown
- KafkaServiceDownMinor
- KafkaServiceDownMajor
- KafkaServiceOutage

KafkaServiceDown

Severity	Minor
Summary	The Kafka service on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="kafka-server"} == 0
Description	Raises when Kafka on a particular host does not respond to Telegraf, typically indicating that Kafka is unavailable on that node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the Kafka status by running systemctl status kafka on the affected node. • Inspect the Kafka logs on the affected node using journalctl -u kafka. • Inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

KafkaServiceDownMinor

Severity	Minor
Summary	{{ $value }} Kafka services are down (at least {{monitoring.services_failed_warning_threshold_percent*100}}%).
Raise condition	count(procstat_running{process_name="kafka-server"} == 0) >= count(procstat_running{process_name="kafka-server"}) * 0.3
Description	Raises when the Kafka cluster has more than 30% of unavailable services.

Troubleshooting	• Inspect the KafkaServiceDown alerts for the host names of the affected nodes. • Inspect the Kafka logs on the affected node using journalctl -u kafka.
Tuning	Not required

KafkaServiceDownMajor

Severity	Major
Summary	{{ $value }} Kafka services are down (at least {{monitoring.services_failed_critical_threshold_percent*100}}%).
Raise condition	count(procstat_running{process_name="kafka-server"} == 0) >= count(procstat_running{process_name="kafka-server"}) * 0.6
Description	Raises when the Kafka cluster has more than 60% of unavailable services.
Troubleshooting	• Inspect the KafkaServiceDown alerts for the host names of the affected nodes. • Inspect the Kafka logs on the affected node using journalctl -u kafka.
Tuning	Not required

KafkaServiceOutage

Severity	Critical
Summary	All Kafka services are down.
Raise condition	count(procstat_running{process_name="kafka-server"} == 0) == count(procstat_running{process_name="kafka-server"})
Description	Raises when all Kafka services across the cluster do not respond to Telegraf, typically indicating deployment or configuration issues.
Troubleshooting	If Kafka is up and running, inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

OpenContrail

This section describes the general alerts for OpenContrail, such as the API, processes, instance, and health check alerts.

- ContrailApiDown
- ContrailApiDownMinor
- ContrailApiDownMajor
- ContrailApiOutage
- ContrailProcessDown
- ContrailProcessDownMinor
- ContrailProcessDownMajor
- ContrailProcessOutage
- ContrailHealthCheckDisabled
- ContrailHealthCheckFailed
- OpencontrailInstancePingCheckDown

ContrailApiDown

Severity	Minor
Summary	The {{ $labels.name }} API endpoint on the {{$labels.host }} node is not accessible for 2 minutes.
Raise condition	http_response_status{name=~"contrail.*"} == 0
Description	Raises when the HTTP check for the OpenContrail API endpoint is failing. The host and name labels in the raised alert contain the host name of the affected node and the service name.

Troubleshooting	• Before debugging OpenContrail, inspect the Neutron API, Keystone, and Neutron server alerts, if any.
	• Verify the service status using systemctl status <service_name> and the service logs in /var/log/contrail/.
	• If the process is still running, obtain the details about the service state:
	  • Trace the system calls using strace -p <pid> -e trace=network.
	  • List the open files, including the network sockets and devices using lsof -p <pid>.
	  • Analyze the packets sent to the port used by the service using tcpdump -nei any port <portnum> -A -s 1500.

Tuning	Not required

ContrailApiDownMinor

Severity	Minor
Summary	More than 30% of {{ $labels.name }} API endpoints are not accessible for 2 minutes.
Raise condition	count(http_response_status{name=~"contrail.*"} == 0) by (name) >= count(http_response_status{name=~"contrail.*"}) by (name) * 0.3
Description	Raises when 30% of the OpenContrail API HTTP checks fail. The name label in the raised alert contains the affected service name.

Troubleshooting	• Inspect the ContrailApiDown alerts for the host names of the affected nodes.
	• Verify the service status using systemctl status <service_name> and the service logs in /var/log/contrail/.
	• If the process is still running, obtain the details about the service state:
	  • Trace the system calls using strace -p <pid> -e trace=network.
	  • List the open files, including the network sockets and devices using lsof -p <pid>.
	  • Analyze the packets sent to the port used by the service using tcpdump -nei any port <portnum> -A -s 1500.

Tuning	Not required

ContrailApiDownMajor

Severity	Major
Summary	More than 60% of {{ $labels.name }} API endpoints are not accessible for 2 minutes.
Raise condition	count(http_response_status{name=~"contrail.*"} == 0) by (name) >= count(http_response_status{name=~"contrail.*"}) by (name) * 0.6
Description	Raises when 60% of the OpenContrail API HTTP checks fail. The name label in the raised alert contains the affected service name.

Troubleshooting	• Inspect the ContrailApiDown alerts for the host names of the affected nodes.
	• Verify the service status using systemctl status <service_name> and the service logs in /var/log/contrail/.
	• If the process is still running, obtain the details about the service state:
	  ◦ Trace the system calls using strace -p <pid> -e trace=network.
	  ◦ List the open files, including the network sockets and devices using lsof -p <pid>.
	  ◦ Analyze the packets sent to the port used by the service using tcpdump -nei any port <portnum> -A -s 1500.

Tuning	Not required

ContrailApiOutage

Severity	Critical
Summary	The {{ $labels.name }} API is not accessible for all available endpoints for 2 minutes.
Raise condition	count(http_response_status{name=~"contrail.*"} == 0) by (name) == count(http_response_status{name=~"contrail.*"}) by (name)
Description	Raises when the HTTP checks fail for all OpenContrail API endpoints. The name label in the raised alert contains the affected service name.

Troubleshooting	• Inspect the ContrailApiDown alerts for the host names of the affected nodes.
	• Verify the service status using systemctl status <service_name> and the service logs in /var/log/contrail/.
	• If the process is still running, obtain the details about the service state:
	  • Trace the system calls using strace -p <pid> -e trace=network.
	  • List the open files, including the network sockets and devices using lsof -p <pid>.
	  • Analyze the packets sent to the port used by the service using tcpdump -nei any port <portnum> -A -s 1500.

Tuning	Not required

ContrailProcessDown

Severity	Minor
Summary	The {{ $labels.process_name }} process on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name=~"contrail.*"} == 0
Description	Raises when the OpenContrail process is down on one node. The host and process_name labels in the raised alert contain the host name of the affected node and the process name.

Troubleshooting	• Verify the service status using systemctl status <service_name> and the service logs in /var/log/contrail/.
	• If the process is still running, obtain the details about the service state:
	  • Trace the system calls using strace -p <pid> -e trace=network.
	  • List the open files, including the network sockets and devices using lsof -p <pid>.
	  • Analyze the packets sent to the port used by the service using tcpdump -nei any port <portnum> -A -s 1500.

Tuning	Not required

ContrailProcessDownMinor

Severity	Minor

Summary	More than 30% of {{ $labels.process_name }} processes are down.
Raise condition	count(procstat_running{process_name=~"contrail.*"} == 0) by (process_name) >= 0.3 * count (procstat_running{process_name=~"contrail.*"}) by (process_name)
Description	Raises when 30% of the OpenContrail processes (by name) are in the DOWN state. The process_name in the raised alert contains the affected process name.

Troubleshooting	• Inspect the ContrailProcessDown alerts for the host names of the affected nodes.
	• Verify the service status using systemctl status <service_name> and the service logs in /var/log/contrail/.
	• If the process is still running, obtain the details about the service state:
	  • Trace the system calls using strace -p <pid> -e trace=network.
	  • List the open files, including the network sockets and devices using lsof -p <pid>.
	  • Analyze the packets sent to the port used by the service using tcpdump -nei any port <portnum> -A -s 1500.

Tuning	Not required

ContrailProcessDownMajor

Severity	Major
Summary	More than 60% {{ $labels.process_name }} processes are down.
Raise condition	count(procstat_running{process_name=~"contrail.*"} == 0) by (process_name) >= 0.6 * count (procstat_running{process_name=~"contrail.*"}) by (process_name)
Description	Raises when 60% of the OpenContrail processes (by name) are in the DOWN state. The process_name in the raised alert contains the affected process name.

Troubleshooting	• Inspect the ContrailProcessDown alerts for the host names of the affected nodes.
	• Verify the service status using systemctl status <service_name> and the service logs in /var/log/contrail/.
	• If the process is still running, obtain the details about the service state:
	  • Trace the system calls using strace -p <pid> -e trace=network.
	  • List the open files, including the network sockets and devices using lsof -p <pid>.
	  • Analyze the packets sent to the port used by the service using tcpdump -nei any port <portnum> -A -s 1500.

Tuning	Not required

ContrailProcessOutage

Severity	Critical
Summary	All {{ $labels.process_name }} processes are down.
Raise condition	count(procstat_running{process_name=~"contrail.*"} == 0) by (process_name) == count(procstat_running{process_name=~"contrail.*"}) by (process_name)
Description	Raises when an OpenContrail process is in the DOWN state on all nodes, indicating that the process is unavailable. The process_name in the raised alert contains the affected process name.

Troubleshooting	• Inspect the ContrailProcessDown alerts for the host names of the affected nodes.
	• Verify the service status using systemctl status <service_name> and the service logs in /var/log/contrail/.
	• If the process is still running, obtain the details about the service state:
	  • Trace the system calls using strace -p <pid> -e trace=network.
	  • List the open files, including the network sockets and devices using lsof -p <pid>.
	  • Analyze the packets sent to the port used by the service using tcpdump -nei any port <portnum> -A -s 1500.

Tuning	Not required

ContrailMetadataCheck

Available starting from the 2019.2.3 maintenance update

Severity	Critical
Summary	The OpenContrail metadata on the {{ $labels.host }} node is unavailable for 15 minutes.
Raise condition	min(exec_contrail_instance_metadata_present) by (host) == 0
Description	Raises when the curl 127.0.0.1:8085/Snh_LinkLocalServiceInfo | grep -c 169.254.169.254 HTTP query returns 0, indicating that the OpenContrail metadata service is unavailable on the affected node.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Configure > Infrastructure > Link local services and verify that metadata is configured. 3. Inspect the Telegraf logs using journalctl -u telegraf.
Tuning	Not required

ContrailHealthCheckDisabled

Available starting from the 2019.2.4 maintenance update

Severity	Critical
Summary	The OpenContrail health check is disabled.
Raise condition	absent(contrail_health_exit_code) == 1
Description	Raises when the metric from the contrail-status check script is absent.
Troubleshooting	Inspect the Telegraf logs on the ntw nodes.
Tuning	Not required

ContrailHealthCheckFailed

Available starting from the 2019.2.4 maintenance update

Severity	Critical
Summary	The OpenContrail health check failed for the {{ $labels.contrail_service }} service on the {{ $labels.host }} node.
Raise condition	contrail_health_exit_code != 0
Description	Raises when any contrail service from the output of contrail-status is inactive. The contrail_service label in the raised alert contains the affected service name.
Troubleshooting	Inspect the affected service.
Tuning	Not required

OpencontrailInstancePingCheckDown

Available starting from the 2019.2.4 maintenance update

Severity	Major
Summary	The OpenContrail instance ping check on the {{ $labels.host }} node is down for 2 minutes.
Raise condition	instance_ping_check_up == 0
Description	Raises when the OpenContrail instance ping check on a node is down for 2 minutes. The host label in the raised alert contains the affected node name.
Tuning	Not required

OpenContrail flows

This section describes the alerts for OpenContrail flows.

> **Warning**
>
> The following set of alerts has been removed starting from the 2019.2.3 maintenance update. For the existing MCP deployments, disable these alerts as described in Manage alerts.

- ContrailFlowsActiveTooHigh
- ContrailFlowsDiscardedTooHigh
- ContrailFlowsDroppedTooHigh
- ContrailFlowsFragErrTooHigh
- ContrailFlowsNextHopInvalidTooHigh
- ContrailFlowsInterfaceInvalidTooHigh
- ContrailFlowsLabelInvalidTooHigh
- ContrailFlowsQueueSizeExceededTooHigh
- ContrailFlowsTableFullTooHigh

ContrailFlowsActiveTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	More than 100 OpenContrail vRouter flows per second on the {{ $labels.host }} node are active for 2 minutes.
Raise condition	deriv(contrail_vrouter_flows_active[5m]) >= 100
Description	Raises when the number of active OpenContrail vRouter flows counted over 5 minutes reaches the growth speed of 100 per second.
Tuning	Disable the alert as described in Manage alerts.

ContrailFlowsDiscardedTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	The average per-second rate of discarded OpenContrail vRouter flows on the {{ $labels.host }} node is more than 0.1 for 2 minutes.
Raise condition	rate(contrail_vrouter_flows_discard[5m]) >= 0.1
Description	Raises when the number of discarded OpenContrail vRouter flows counted over 5 minutes reaches the growth speed of 0.1 per second.
Tuning	Disable the alert as described in Manage alerts.

ContrailFlowsDroppedTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	The average per-second rate of dropped OpenContrail vRouter flows on the {{ $labels.host }} node is more than 0.2 for 2 minutes.
Raise condition	rate(contrail_vrouter_flows_flow_action_drop[5m]) >= 0.2
Description	Raises when the number of dropped OpenContrail vRouter flows counted over 5 minutes reaches the growth speed of 0.2 per second.
Tuning	Disable the alert as described in Manage alerts.

ContrailFlowsFragErrTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	More than 100 OpenContrail vRouter flows had fragment errors on the {{$labels.host }} node for 2 minutes.
Raise condition	min(contrail_vrouter_flows_frag_err) by (host) >= 100
Description	Raises when the number of OpenContrail vRouter flows with fragment errors counted over 5 minutes reaches 100.
Tuning	Disable the alert as described in Manage alerts.

ContrailFlowsNextHopInvalidTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	The average per-second rate of OpenContrail vRouter flows with an invalid next hop on the {{ $labels.host }} node is more than 0.1 for 2 minutes.
Raise condition	rate(contrail_vrouter_flows_invalid_nh[5m]) >= 0.1
Description	Raises when the number of OpenContrail vRouter flows with an invalid next hop counted over 5 minutes reaches the growth speed of 0.1 per second.
Tuning	Disable the alert as described in Manage alerts.

ContrailFlowsInterfaceInvalidTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	The average per-second rate of OpenContrail vRouter flows with an invalid composite interface on the {{ $labels.host}} node is more than 0.05 for 2 minutes.
Raise condition	rate(contrail_vrouter_flows_composite_invalid_interface[5m]) >= 0.05
Description	Raises when the number of OpenContrail vRouter flows with invalid composite interfaces counted over 5 minutes reaches the growth speed of 0.05 per second.
Tuning	Disable the alert as described in Manage alerts.

ContrailFlowsLabelInvalidTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	More than 100 OpenContrail vRouter flows had an invalid composite interface on the {{$labels.host }} node for 2 minutes.
Raise condition	min(contrail_vrouter_flows_invalid_label) by (host) >= 100
Description	Raises when the number of OpenContrail vRouter flows with an invalid label (specifying next hop) reaches 100 by default.

Tuning	Disable the alert as described in Manage alerts.

ContrailFlowsQueueSizeExceededTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	Warning. The average per-second rate of OpenContrail vRouter flows exceeding the queue size on the {{ $labels.host }} node is more than 0.1 for 2 minutes.
Raise condition	rate(contrail_vrouter_flows_flow_queue_limit_exceeded[5m]) >= 0.1
Description	Raises when the number of queue exceeded errors counted over 5 minutes reaches the growth speed of 0.1 per second.
Tuning	Disable the alert as described in Manage alerts.

ContrailFlowsTableFullTooHigh

Removed since the 2019.2.3 maintenance update

Severity	Warning
Summary	More than 100 OpenContrail vRouter flows had a full table on the {{$labels.host }} node for 2 minutes.
Raise condition	min(contrail_vrouter_flows_flow_table_full) by (host) >= 100
Description	Raises when the OpenContrail vRouter flow table size reaches 100 on one node.
Tuning	Disable the alert as described in Manage alerts.

OpenContrail vRouter

This section describes the alerts for the OpenContrail vRouter alerts.

- ContrailBGPSessionsNoEstablished
- ContrailBGPSessionsNoActive
- ContrailBGPSessionsDown
- ContrailXMPPSessionsMissingEstablished
- ContrailXMPPSessionsMissing
- ContrailXMPPSessionsDown
- ContrailXMPPSessionsTooHigh
- ContrailXMPPSessionsChangesTooHigh
- ContrailVrouterXMPPSessionsZero
- ContrailVrouterXMPPSessionsTooHigh
- ContrailVrouterXMPPSessionsChangesTooHigh
- ContrailVrouterDNSXMPPSessionsZero
- ContrailVrouterDNSXMPPSessionsTooHigh
- ContrailVrouterDNSXMPPSessionsChangesTooHigh
- ContrailVrouterLLSSessionsTooHigh
- ContrailVrouterLLSSessionsChangesTooHigh
- ContrailGlobalVrouterConfigCheckDisabled
- ContrailGlobalVrouterConfigCheckFailed

ContrailBGPSessionsNoEstablished

Severity	Warning
Summary	There are no established OpenContrail BGP sessions on the {{ $labels.host }} node for 2 minutes.
Raise condition	max(contrail_bgp_session_count) by (host) == 0
Description	Raises when no BGP sessions in the established state (FSM) exist on a node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the bgp_peer module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.
Tuning	Not required

ContrailBGPSessionsNoActive

Severity	Warning
Summary	There are no active OpenContrail BGP sessions on the {{ $labels.host }} node for 2 minutes.
Raise condition	max(contrail_bgp_session_up_count) by (host) == 0
Description	Raises when no BGP sessions in the active state (FSM) exist on a node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the bgp_peer module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.
Tuning	Not required

ContrailBGPSessionsDown

Severity	Warning
Summary	The OpenContrail BGP sessions on the {{ $labels.host }} node are down for 2 minutes.
Raise condition	min(contrail_bgp_session_down_count) by (host) > 0
Description	Raises when a node has BGP sessions in the down state. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the bgp_peer module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.

Tuning	Not required

ContrailXMPPSessionsMissingEstablished

Severity	Warning
Summary	The OpenContrail XMPP sessions in the established state are missing on the compute cluster for 2 minutes.
Raise condition	count(contrail_vrouter_xmpp) * 2 - sum(contrail_xmpp_session_up_count) > 0
Description	Raises when the compute cluster has no OpenContrail XMPP sessions in the established state (FSM). No assumption is made for equal sessions distribution across the cluster. The vRouter can have 0 sessions in the working state. However, a properly operating compute cluster must have at least 2 connections per vRouter.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.
Tuning	Not required

ContrailXMPPSessionsMissing

Severity	Warning
Summary	The OpenContrail XMPP sessions are missing on the compute cluster for 2 minutes.
Raise condition	count(contrail_vrouter_xmpp) * 2 - sum(contrail_xmpp_session_count) > 0
Description	Raises when the compute cluster has no OpenContrail XMPP sessions in any state. The conditions are the same as for the ContrailXMPPSessionsMissingEstablished alert.

Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.
Tuning	Not required

ContrailXMPPSessionsDown

Severity	Warning
Summary	The {{ $labels.host }} node contains the OpenContrail XMPP sessions in the down state for 2 minutes.
Raise condition	min(contrail_xmpp_session_down_count) by (host) > 0
Description	Raises when a node has OpenConrail XMPP sessions in the DOWN state. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.
Tuning	Not required

ContrailXMPPSessionsTooHigh

Severity	Warning
Summary	There are more than 500 open OpenContrail XMPP sessions on the {{ $labels.host }} node for 2 minutes.

Raise condition	min(contrail_xmpp_session_count) by (host) >= 500
Description	Raises when the number of OpenContrail XMPP sessions reaches 500 on one node. The host label in the raised alert contains the host name of the affected node. Warning For production environments, configure the alert after deployment.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.

Tuning	For example, to change the threshold to 1000 sessions:
	
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	  `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	  `classes:`
	  `- cluster.<cluster_name>.stacklight.custom.alerts`
	  `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        ContrailXMPPSessionsTooHigh:`
	`          if: >-`
	`            min(contrail_xmpp_session_count) by (host) >= 1000`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

ContrailXMPPSessionsChangesTooHigh

Severity	Warning
Summary	The OpenContrail XMPP sessions on the {{ $labels.host }} node have changed more than 100 times.
Raise condition	abs(delta(contrail_xmpp_session_count[2m])) >= 100

Description	Raises when the number of OpenContrail XMPP session changes reaches 100 on one node, calculated as an absolute difference of the first and last point in a two-minute time frame. The host label in the raised alert contains the host name of the affected node. Warning For production environments, configure the alert after deployment.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.

<table>
<tr><td>Tuning</td><td>
For example, to change the threshold to >= 250 sessions:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
    1. Create a file for alert customizations:

```
touch cluster/<cluster_name>/stacklight/custom/alerts.yml
```

    2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

```
classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...
```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

```
parameters:
  prometheus:
    server:
      alert:
        ContrailXMPPSessionsChangesTooHigh:
          if: >-
            abs(delta(contrail_xmpp_session_count[2m])) >= 250
```

3. From the Salt Master node, apply the changes:

```
salt 'I@prometheus:server' state.sls prometheus.server
```

4. Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

ContrailVrouterXMPPSessionsZero

Severity	Warning
Summary	There are no OpenContrail vRouter XMPP sessions on the {{ $labels.host }} node for 2 minutes.
Raise condition	min(contrail_vrouter_xmpp) by (host) == 0
Description	Raises when a node has no OpenContrail vRouter XMPP sessions. The host label in the raised alert contains the host name of the affected node.

Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.
Tuning	Not required

ContrailVrouterXMPPSessionsTooHigh

Severity	Warning
Summary	There are more than 10 open OpenContrail vRouter XMPP sessions on the {{ $labels.host }} node for 2 minutes.
Raise condition	min(contrail_vrouter_xmpp) by (host) >= 10
Description	Raises when the number of OpenContrail vrouter XMPP sessions reaches 10 on one node. The host label in the raised alert contains the host name of the affected node. Warning For production environments, configure the alert after deployment.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > Infrastructure > Control Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.

©2025, Mirantis Inc.

Tuning	For example, to change the threshold to 20 sessions:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        ContrailVrouterXMPPSessionsTooHigh:`
	`          if: >-`
	`            min(contrail_vrouter_xmpp) by (host) >= 20`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

ContrailVrouterXMPPSessionsChangesTooHigh

Severity	Warning
Summary	The OpenContrail vRouter XMPP sessions on the {{$labels.host }} node have changed more than 5 times.
Raise condition	abs(delta(contrail_vrouter_xmpp[2m])) >= 5

Description	Raises when the number of OpenContrail vRouter XMPP session changes reaches 5 on one node, calculated as an absolute difference of the first and last points in a two-minute time frame. The host label in the raised alert contains the host name of the affected node. Warning For production environments, configure the alert after deployment.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > DNS Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.

Tuning	For example, to change the threshold to 10 sessions:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        ContrailVrouterXMPPSessionsChangesTooHigh:`
	`          if: >-`
	`            abs(delta(contrail_vrouter_xmpp[2m])) >= 10`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

ContrailVrouterDNSXMPPSessionsZero

Severity	Warning
Summary	There are no OpenContrail vRouter DNS-XMPP sessions on the {{ $labels.host }} node for 2 minutes.
Raise condition	min(contrail_vrouter_dns_xmpp) by (host) == 0
Description	Raises when one node has no OpenContrail DNS-XMPP sessions. The host label in the raised alert contains the host name of the affected node.

Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > DNS Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.
Tuning	Not required

ContrailVrouterDNSXMPPSessionsTooHigh

Severity	Warning
Summary	More than 10 OpenContrail vRouter DNS-XMPP sessions are open on the {{ $labels.host }} node for 2 minutes.
Raise condition	min(contrail_vrouter_dns_xmpp) by (host) >= 10
Description	Raises when the number of OpenContrail DNS-XMPP sessions reaches 10 on one node. The host label in the raised alert contains the host name of the affected node. Warning For production environments, configure the alert after deployment.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > DNS Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.

Tuning	For example, to change the threshold to 20 sessions:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        ContrailVrouterDNSXMPPSessionsTooHigh:`
	`          if: >-`
	`            min(contrail_vrouter_dns_xmpp) by (host) >= 20`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

ContrailVrouterDNSXMPPSessionsChangesTooHigh

Severity	Warning
Summary	The OpenContrail vRouter DNS-XMPP sessions on the {{ $labels.host }} node have changed more than 5 times.
Raise condition	abs(delta(contrail_vrouter_dns_xmpp[2m])) >= 5

Description	Raises when the number of OpenContrail DNS-XMPP session changes reaches 5 on one node, calculated as an absolute difference of the first and last points in a two-minute time frame. Warning For production environments, configure the alert after deployment.
Troubleshooting	1. Log in to the OpenContrail web UI using the credentials from /etc/contrail/contrail-webui-userauth.js on the network nodes. 2. Navigate to Monitor > DNS Nodes and select the affected node to inspect the analytics data of the OpenContrail controller nodes. 3. In Introspect, inspect the introspection data filtered by request type. Select the xmpp_server module. 4. Verify the BGP routers configuration in Configure > Infrastructure > BGP Routers.

Tuning	For example, to change the threshold to 10 sessions:
	
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	   1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        ContrailVrouterDNSXMPPSessionsChangesTooHigh:`
	`          if: >-`
	`            abs(delta(contrail_vrouter_dns_xmpp[2m])) >= 10`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

ContrailVrouterLLSSessionsTooHigh

Severity	Warning
Summary	There are more than 10 open OpenContrail vRouter LLS sessions on the {{ $labels.host }} node for 2 minutes.
Raise condition	min(contrail_vrouter_lls) by (host) >= 10

<table>
<tr><td>Description</td><td>Raises when the number of OpenContrail vRouter LocalLinkService sessions reaches 10 on one node.

Warning
For production environments, configure the alert after deployment.</td></tr>
<tr><td>Tuning</td><td>For example, to change the threshold to 20 sessions:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>
Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        ContrailVrouterLLSSessionsTooHigh:
          if: >-
            min(contrail_vrouter_lls) by (host) >= 20</pre>
From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>
Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

ContrailVrouterLLSSessionsChangesTooHigh

Severity	Warning
Summary	The OpenContrail vRouter LLS sessions on the {{$labels.host }} node have changed more than 5 times.

<table>
<tr><td>Raise condition</td><td>abs(delta(contrail_vrouter_lls[2m])) >= 5</td></tr>
<tr><td>Description</td><td>Raises when the number of OpenContrail vRouter LLS session changes reaches 5 on one node, calculated as an absolute difference of the first and last points in a two-minute time frame.

Warning
For production environments, configure the alert after deployment.</td></tr>
<tr><td>Tuning</td><td>For example, to change the threshold to 10 sessions:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>
Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        ContrailVrouterLLSSessionsChangesTooHigh:
          if: >-
            abs(delta(contrail_vrouter_lls[2m])) >= 10</pre>
From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>
Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

ContrailGlobalVrouterConfigCheckDisabled

Available since 2019.2.4

Severity	Critical
Summary	The OpenContrail global vRouter configuration check is disabled.
Raise condition	absent(contrail_global_vrouter_config_exit_code) == 1
Description	Raises when Prometheus has no metric with the contrail_global_vrouter_config_exit_code name.
Troubleshooting	Inspect the Telegraf logs on the ntw nodes.
Tuning	Not required

ContrailGlobalVrouterConfigCheckFailed

Available since 2019.2.4

Severity	Critical
Summary	The OpenContrail global vRouter configuration check failed on the {{ $labels.host }} node.
Raise condition	contrail_global_vrouter_config_exit_code != 0
Description	Raises when the OpenContrail Virtual Network Controller (VNC) API returns 0 or more than 1 global-vrouter-configs.
Troubleshooting	Inspect the output of the contrail-status command on any ntw node.
Tuning	Not required

Redis

This section describes the alerts for the Redis service.

- RedisServiceDown
- RedisServiceDownMinor
- RedisServiceDownMajor
- RedisServiceOutage

RedisServiceDown

Severity	Minor
Summary	The Redis service on the {{$labels.host}} node is down.
Raise condition	procstat_running{process_name="redis-server"} == 0
Description	Raises when Telegraf cannot find running redis-server processes on a node, typically indicating MEM consumption on the node, Redis port usage by another process, or wrong permissions set for Redis configuration or log files. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the redis-server service status using systemctl status redis-server. • Inspect the redis-server service logs in /var/log/redis/redis-server.log.
Tuning	Not required

RedisServiceDownMinor

Severity	Minor
Summary	More than 30% of Redis services are down.
Raise condition	count(procstat_running{process_name="redis-server"} == 0) >= count (procstat_running{process_name="redis-server"}) * 0.3
Description	Raises when Telegraf cannot find running redis-server processes by default on more than 30% of the ntw and nal hosts.

Troubleshooting	• Inspect the RedisServiceDown alerts for the host names of the affected nodes. • Verify the redis-server service status using systemctl status redis-server. • Inspect the redis-server service logs in /var/log/redis/redis-server.log.
Tuning	Not required

RedisServiceDownMajor

Severity	Major
Summary	More than 60% of Redis services are down.
Raise condition	count(procstat_running{process_name="redis-server"} == 0) >= count (procstat_running{process_name="redis-server"}) * 0.6
Description	Raises when Telegraf cannot find running redis-server processes by default on more than 60% of the mtr hosts.
Troubleshooting	• Inspect the RedisServiceDown alerts for the host names of the affected nodes. • Verify the redis-server service status using systemctl status redis-server. • Inspect the redis-server service logs in /var/log/redis/redis-server.log.
Tuning	Not required

RedisServiceOutage

Severity	Critical
Summary	All Redis services are down.
Raise condition	count(procstat_running{process_name="redis-server"} == 0) == count (procstat_running{process_name="redis-server"})
Description	Raises when Telegraf cannot find running redis-server processes on all ntw and nal hosts.
Troubleshooting	• Inspect the RedisServiceDown alerts for the host names of the affected nodes. • Verify the redis-server service status using systemctl status redis-server. • Inspect the redis-server service logs in /var/log/redis/redis-server.log.

Tunin g	Not required

ZooKeeper

This section describes the alerts for the ZooKeeper service.

- ZookeeperServiceDown
- ZookeeperServiceErrorWarning
- ZookeeperServicesDownMinor
- ZookeeperServicesDownMajor
- ZookeeperServiceOutage

ZookeeperServiceDown

Severity	Minor
Summary	The ZooKeeper service on the {{ $labels.host }} node is down for 2 minutes.
Raise condition	zookeeper_up == 0
Description	Raises when the ZooKeeper service on a host node does not respond to Telegraf, typically indicating that ZooKeeper is down on that node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the ZooKeeper status on the affected node using service zookeeper status. • If ZooKeeper is up and running, inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

ZookeeperServiceErrorWarning

Severity	Warning
Summary	The ZooKeeper service on the {{ $labels.host }} node is not responding for 2 minutes.
Raise condition	zookeeper_service_health == 0

Description	Raises when the ZooKeeper service on a node is not healthy (in operational mode), typically indicating that the service is unresponsive due to a high load or an operating system or hardware issue on the node.
Troubleshooting	• Inspect dmesg and /var/log/kern.log. • Inspect the logs in /var/log/zookeeper.
Tuning	Not required

ZookeeperServicesDownMinor

Severity	Minor
Summary	More than 30% of ZooKeeper services are down for 2 minutes.
Raise condition	count(zookeeper_up == 0) >= count(zookeeper_up) * 0.3
Description	Raises when a ZooKeeper cluster has more than 30% of unavailable services.
Troubleshooting	Inspect the ZooKeeper logs on any node of the affected cluster using journalctl -u zookeeper.
Tuning	Not required

ZookeeperServicesDownMajor

Severity	Major
Summary	More than 60% of ZooKeeper services are down for 2 minutes.
Raise condition	count(zookeeper_up == 0) >= count(zookeeper_up) * 0.6
Description	Raises when a ZooKeeper cluster has more than 60% of unavailable services.
Troubleshooting	Inspect the ZooKeeper logs on any node of the affected cluster using journalctl -u zookeeper.

Tuning	Not required

ZookeeperServiceOutage

Severity	Critical
Summary	All ZooKeeper services are down for 2 minutes.
Raise condition	count(zookeeper_up == 0) == count(zookeeper_up)
Description	Raises when all ZooKeeper services across a cluster do not respond to Telegraf, typically indicating deployment or configuration issues.
Troubleshooting	• Inspect the ZooKeeper logs on any node of the affected cluster using journalctl -u zookeeper. • If ZooKeeper is up and running, inspect the Telegraf logs on the affected node using journalctl -u telegraf.
Tuning	Not required

Ceph

This section describes the alerts for the Ceph cluster.

- CephClusterHealthMinor
- CephClusterHealthCritical
- CephMonitorDownMinor
- CephOsdDownMinor
- CephOsdSpaceUsageWarning
- CephOsdSpaceUsageMajor
- CephPool{pool_name}SpaceUsageWarning
- CephPool{pool_name}SpaceUsageCritical
- CephOsdPgNumTooHighWarning
- CephOsdPgNumTooHighCritical
- CephPredictOsdIOPSthreshold
- CephPredictOsdIOPSauto
- CephPredictUsageRAM
- CephPredictOsdWriteLatency
- CephPredictOsdReadLatency
- CephPredictPoolSpace
- CephPredictPoolIOPSthreshold
- CephPredictPoolIOPSauto

CephClusterHealthMinor

Severity	Minor
Summary	The Ceph cluster is in the WARNING state. For details, run ceph -s.
Raise condition	ceph_health_status == 1
Description	Raises according to the status reported by the Ceph cluster.

Troubleshooting	Run the ceph -s command on any Ceph node to identify the reason and resolve the issue depending on the output.
Tuning	Not required

CephClusterHealthCritical

Severity	Critical
Summary	The Ceph cluster is in the CRITICAL state. For details, run ceph -s.
Raise condition	ceph_health_status == 2
Description	Raises according to the status reported by the Ceph cluster.
Troubleshooting	Run the ceph -s command on any Ceph node to identify the reason and resolve the issue depending on the output.
Tuning	Not required

CephMonitorDownMinor

Severity	Minor
Summary	{{ $value }}% of Ceph Monitors are down. For details, run ceph -s.
Raise condition	count(ceph_mon_quorum_status) - sum(ceph_mon_quorum_status) > 0
Description	Raises if any of the Ceph Monitors in the Ceph cluster is down.
Troubleshooting	Inspect the /var/log/ceph/ceph-mon.<hostname>.log logs on the affected cmn node.
Tuning	Not required

CephOsdDownMinor

Severity	Minor
Summary	{{ $value }}% of Ceph OSDs are down. For details, run ceph osd tree.
Raise condition	count(ceph_osd_up) - sum(ceph_osd_up) > 0
Description	Raises if any of the Ceph OSD nodes in the Ceph cluster is down.
Troubleshooting	Inspect the /var/log/ceph/ceph-osd.<hostname>.log logs on the affected osd node.
Tuning	Not required

CephOsdSpaceUsageWarning

Severity	Warning
Summary	{{ $value }} bytes of the Ceph OSD space (>=75%) is used for 3 minutes. For details, run cephdf.
Raise condition	ceph_cluster_total_used_bytes > ceph_cluster_total_bytes * {{threshold}}
Description	Raises when a Ceph OSD used space capacity exceeds the threshold of 75%.
Troubleshooting	• Remove unused data from the Ceph cluster. • Add more Ceph OSDs to the Ceph cluster. • Adjust the warning threshold (use with caution).

<table>
<tr><td>Tuning</td><td>
For example, to change the threshold to 80%:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
   1. Create a file for alert customizations:

```
touch cluster/<cluster_name>/stacklight/custom/alerts.yml
```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

```
classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...
```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

```
parameters:
  prometheus:
    server:
      alert:
        CephOsdSpaceUsageWarning:
          if: >-
            ceph_cluster_total_used_bytes > ceph_cluster_total_bytes * 0.8
```

3. From the Salt Master node, apply the changes:

```
salt 'I@prometheus:server' state.sls prometheus.server
```

4. Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

CephOsdSpaceUsageMajor

Severity	Major
Summary	{{ $value }} bytes of the Ceph OSD space (>=85%) is used for 3 minutes. For details, run cephdf.
Raise condition	ceph_cluster_total_used_bytes > ceph_cluster_total_bytes * {{threshold}}
Description	Raises when a Ceph OSD used space capacity exceeds the threshold of 85%.

<table>
<tr><td>Troubleshooting</td><td>

- Remove unused data from the Ceph cluster.
- Add more Ceph OSDs to the Ceph cluster.
- Adjust the warning threshold (use with caution).

</td></tr>
<tr><td>Tuning</td><td>

For example, to change the threshold to 95%:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           CephOsdSpaceUsageMajor:
             if: >-
               ceph_cluster_total_used_bytes > ceph_cluster_total_bytes * 0.95
   ```

3. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```

4. Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

CephPool{pool_name}SpaceUsageWarning

Severity	Warning
Summary	The Ceph {{pool_name}} pool uses 75% of available space for 3 minutes. For details, run ceph df.
Raise condition	ceph_pool_bytes_used / (ceph_pool_bytes_used + ceph_pool_max_avail) * on(pool_id) group_left(name) ceph_pool_metadata{name="{{pool_name}}"} > {{threshold}}

Description	Raises when a Ceph pool used space capacity exceeds the threshold of 75%.
Troubleshooting	• Add more Ceph OSDs to the Ceph cluster. • Temporarily move the affected pool to the less occupied disks of the cluster.

Tuning	Should be tuned per pool. For example, to change the threshold to 80% for pool volumes:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        CephPoolvolumesSpaceUsageWarning:`
	`          if: >-`
	`            ceph_pool_bytes_used / (ceph_pool_bytes_used + ceph_pool_max_avail) * \`
	`            on(pool_id) group_left(name) ceph_pool_metadata{name="volumes"} > 0.8`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

CephPool{pool_name}SpaceUsageCritical

Severity	Critical
Summary	The Ceph {{pool_name}} pool uses 85% of available space for 3 minutes. For details, run ceph df.

Raise condition	ceph_pool_bytes_used / (ceph_pool_bytes_used + ceph_pool_max_avail) * on(pool_id) group_left(name) ceph_pool_metadata{name="{{pool_name}}"} > {{threshold}}
Description	Raises when a Ceph pool used space capacity exceeds the threshold of 85%.
Troubleshooting	• Add more Ceph OSDs to the Ceph cluster. • Temporarily move the affected pool to the less occupied disks of the cluster.

Tuning	Should be tuned per pool. For example, to change the threshold to 90% for pool volumes:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	  `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	  `classes:`
	  `- cluster.<cluster_name>.stacklight.custom.alerts`
	  `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        CephPoolvolumesSpaceUsageCritical:`
	`          if: >-`
	`            ceph_pool_bytes_used / (ceph_pool_bytes_used + ceph_pool_max_avail) * \`
	`            on(pool_id) group_left(name) ceph_pool_metadata{name="volumes"} > 0.9`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

CephOsdPgNumTooHighWarning

Severity	Warning

<table>
<tr><td>Summary</td><td>Some Ceph OSDs contain more than 200 PGs. This may have a negative impact on the cluster performance. For details, run ceph pg dump.</td></tr>
<tr><td>Raise condition</td><td>max(ceph_osd_numpg) > 200</td></tr>
<tr><td>Description</td><td>Raises when the number of PGs on Ceph OSDs is higher than the default threshold of 200.</td></tr>
<tr><td>Troubleshooting</td><td>When designing a Ceph cluster, keep 100-300 PGs per Ceph OSD and up to 400 PGs if SSD disks are used. For a majority of deployments that use modern hardware, it is safe to keep approximately 300 PGs.</td></tr>
<tr><td>Tuning</td><td>For example, to change the threshold to 400 PGs:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre><code>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</code></pre>
Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre><code>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</code></pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre><code>parameters:
  prometheus:
    server:
      alert:
        CephOsdPgNumTooHighWarning:
          if: >-
            max(ceph_osd_numpg) > 400</code></pre>
From the Salt Master node, apply the changes:
<pre><code>salt 'I@prometheus:server' state.sls prometheus.server</code></pre>
Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

CephOsdPgNumTooHighCritical

Severity	Critical

Summary	Some Ceph OSDs contain more than 300 PGs. This may have a negative impact on the cluster performance. For details, run ceph pg dump.
Raise condition	max(ceph_osd_numpg) > 300
Description	Raises when the number of PGs on Ceph OSDs is bigger than the default threshold of 300.
Troubleshooting	When designing a Ceph cluster, keep 100-300 PGs per Ceph OSD and up to 400 PGs if SSD disks are used. For a majority of deployments that use modern hardware, it is safe to keep approximately 300 PGs.

Tuning	For example, to change the threshold to 500 PGs:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	  `touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	  `classes:`
	  `- cluster.<cluster_name>.stacklight.custom.alerts`
	  `...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        CephOsdPgNumTooHighCritical:`
	`          if: >-`
	`            max(ceph_osd_numpg) > 500`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

> Note
>
> Ceph prediction alerts have been added starting from the MCP 2019.2.3 update and should be enabled manually. For details, see Enable the Ceph Prometheus plugin.

CephPredictOsdIOPSthreshold

Available starting from the 2019.2.3 maintenance update

Severity	Minor
Summary	The IOPS on the {{ $labels.ceph_daemon }} Ceph OSD are increasing rapidly.
Raise condition	predict_linear(ceph_osd_op:rate5m[{{threshold}}d], {{threshold}} * 86400) > {{osd_iops_limit}}
Description	Predicts the IOPS consumption per Ceph OSD in a specified time range, 1 week by default. The threshold parameter defines the time range. Warning For production environments, configure osd_iops_limit after deployment depending on the used hardware. For exemplary estimates for different hardware types, see IOPS.

Tuning	For example, to change osd_iops_limit to 200:
	
	1. On the cluster level of the Reclass model in the cluster/<cluster_name>/ceph/common.yml file, add:
	`parameters:`
	` _param:`
	`   osd_iops_limit: 200`
	2. From the Salt Master node, apply the changes:
	`salt "I@prometheus:server" state.sls prometheus.server`
	3. Verify the updated alert definition in the Prometheus web UI.

CephPredictOsdIOPSauto

Available starting from the 2019.2.3 maintenance update

Severity	Minor
Summary	The IOPS on the {{ $labels.ceph_daemon }} Ceph OSD are increasing rapidly.
Raise condition	predict_linear(ceph_osd_op:rate5m[{{threshold}}d], {{threshold}} * 86400) > avg_over_time(ceph_osd_op:rate5m[1d]) * {{ iops_threshold }}
Description	Predicts the IOPS consumption per OSD in a specified time range, 1 week by default. The threshold parameter defines the time range. Warning For production environments, configure osd_iops_threshold after deployment depending on the current cluster load and estimated limits from CephPredictOsdIOPSthreshold.

Tuning	For example, to change osd_iops_threshold to 2:
	
	1. On the cluster level of the Reclass model in the cluster/<cluster_name>/ceph/common.yml file, add:
	
	`parameters:`
	`  _param:`
	`    osd_iops_threshold: 2`
	
	2. From the Salt Master node, apply the changes:
	
	`salt "I@prometheus:server" state.sls prometheus.server`
	
	3. Verify the updated alert definition in the Prometheus web UI.

CephPredictUsageRAM

Available starting from the 2019.2.3 maintenance update

Severity	Minor
Summary	The {{$labels.host}} host may run out of available RAM next week.
Raise condition	predict_linear(mem_free{host=~"cmn.*|rgw.*|osd.*"}[{{threshold}}d], {{threshold}} * 86400) < 0
Description	Predicts the exhaustion of the available RAM on Ceph nodes in a defined time range.

Tuning	Not required

CephPredictOsdWriteLatency

Available starting from the 2019.2.3 maintenance update

Severity	Minor
Summary	The {{$labels.name}} on the {{$labels.host}} host may become unresponsive shortly. Verify the OSDs top load on the Ceph OSD Overview Grafana dashboard.
Raise condition	predict_linear(diskio_write_time:rate5m {host=~"osd.*",name=~"sd[b-z]*"}[{{threshold}}d], {{threshold}} * 86400) > avg_over_time(diskio_write_time:rate5m[1d]) * {{write_latency_threshold}}
Description	Predicts the OSD disks responsiveness in a specified time range based on the write latency. The threshold parameter defines the time range. The write_latency_threshold parameter defines the differences to detect in the write latency. Warning For production environments, configure write_latency_threshold after deployment.

Tuning	For example, to change write_latency_threshold to 2:
	
	1. On the cluster level of the Reclass model in the cluster/<cluster_name>/ceph/common.yml file, add:
	
	`parameters:`
	`  _param:`
	`    write_latency_threshold: 2`
	
	2. From the Salt Master node, apply the changes:
	
	`salt "I@prometheus:server" state.sls prometheus.server`
	
	3. Verify the updated alert definition in the Prometheus web UI.

CephPredictOsdReadLatency

Available starting from the 2019.2.3 maintenance update

Severity	Minor

Summary	The {{$labels.name}} on the {{$labels.host}} host may become unresponsive shortly. Verify the OSDs top load on the Ceph OSD Overview Grafana dashboard.
Raise condition	predict_linear(diskio_read_time:rate5m{host=~"osd.*",name=~"sd[b-z]*"}[{{threshold}}d], {{threshold}} * 86400) > avg_over_time(diskio_read_time:rate5m[1d]) * {{read_latency_threshold}}
Description	Predicts the OSD disks responsiveness in a specified time range based on the read latency. The threshold parameter defines the time range. The read_latency_threshold parameter defines the differences to detect in the read latency. Warning For production environments, configure read_latency_threshold after deployment.

Tuning	For example, to change read_latency_threshold to 2:
	1. On the cluster level of the Reclass model in the cluster/<cluster_name>/ceph/common.yml file, add:
	`parameters:`
	`  _param:`
	`    read_latency_threshold: 2`
	2. From the Salt Master node, apply the changes:
	`salt "I@prometheus:server" state.sls prometheus.server`
	3. Verify the updated alert definition in the Prometheus web UI.

CephPredictPoolSpace

Available starting from the 2019.2.3 maintenance update

Severity	Minor
Summary	The {{pool_name}} pool may consume more than {{100*space_threshold}}% of the available capacity in 1 week. For details, run ceph df and plan proper actions.
Raise condition	predict_linear(ceph_pool_bytes_used[{{threshold}}d], {{threshold}} * 86400) * on(pool_id) group_left(name) ceph_pool_metadata{name="{{pool_name}}"} > (ceph_pool_bytes_used + ceph_pool_max_avail) * {{space_threshold}} * on(pool_id) group_left(name) ceph_pool_metadata{name="{{pool_name}}"}

Description	Predicts the exhaustion of all available capacity of a pool in a defined time range. The threshold parameter specifies the time range to use. The space_threshold parameter defines the capacity threshold, similar to the one set in CephPool{pool_name}SpaceUsageCritical. **Warning** For production environments, configure space_threshold after deployment.

Tuning	For example, to change space_threshold to 85:
	
	1. On the cluster level of the Reclass model in the cluster/<cluster_name>/ceph/common.yml file, add:
	
	`parameters:`
	`  _param:`
	`    space_threshold: 85`
	
	2. From the Salt Master node, apply the changes:
	
	`salt "I@prometheus:server" state.sls prometheus.server`
	
	3. Verify the updated alert definition in the Prometheus web UI.

CephPredictPoolIOPSthreshold

Available starting from the 2019.2.3 maintenance update

Severity	Minor
Summary	The IOPS in the {{pool_name}} are increasing rapidly.
Raise condition	predict_linear(ceph_pool_ops:rate5m[{{threshold}}d], {{threshold}} * 86400) * on(pool_id) group_left(name) ceph_pool_metadata{name="{{pool_name}}"} > {{ iops_limit }}
Description	Predicts the IOPS consumption per pool in a specified time range, 1 week by default. The threshold parameter specifies the time range to use. **Warning** For production environments, after deployment, set pool_iops_limit to osd_iops_limit from CephPredictOsdIOPSthreshold multiplied by the number OSDs for this pool.

Tuning	For example, to change pool_iops_limit to 2000:
	1. On the cluster level of the Reclass model in the cluster/<cluster_name>/ceph/common.yml file, add:
	`parameters:`
	`  _param:`
	`    pool_iops_limit: 2000`
	2. From the Salt Master node, apply the changes:
	`salt "I@prometheus:server" state.sls prometheus.server`
	3. Verify the updated alert definition in the Prometheus web UI.

CephPredictPoolIOPSauto

Available starting from the 2019.2.3 maintenance update

Severity	Minor
Summary	The IOPS in the {{pool_name}} are increasing rapidly.
Raise condition	predict_linear(ceph_pool_ops:rate5m[{{threshold}}d], {{threshold}} * 86400) * on(pool_id) group_left(name) ceph_pool_metadata{name="{{pool_name}}"} > avg_over_time(ceph_pool_ops:rate5m[1d]) * {{ iops_threshold }}
Description	Predicts the IOPS utilisation per pool in a specified time range, 1 week by default. The threshold parameter specifies the time range to use. Warning For production environments, after deployment, set pool_iops_threshold to iops_limit from CephPredictOsdIOPSAuto muliplied by the number of OSDs connected to each pool.

Tuning	For example, to change pool_iops_threshold to 3:
	1. On the cluster level of the Reclass model in the cluster/<cluster_name>/ceph/common.yml file, add:
	`parameters:`
	`  _param:`
	`    pool_iops_threshold: 3`
	2. From the Salt Master node, apply the changes:
	`salt "I@prometheus:server" state.sls prometheus.server`
	3. Verify the updated alert definition in the Prometheus web UI.

RadosGWOutage

Available only in the 2019.2.10 maintenance update

Severity	Critical
Summary	RADOS Gateway outage.
Raise condition	max(openstack_api_check_status{name=~"radosgw.*"}) == 0
Description	Raises if RADOS Gateway is not accessible for all available RADOS Gateway endpoints in the OpenStack service catalog.
Tuning	Not required

RadosGWDown

Available only in the 2019.2.10 maintenance update

Severity	Major
Summary	The {{ $labels.name }} endpoint is not accessible.
Raise condition	openstack_api_check_status{name=~"radosgw.*"} == 0
Description	Raises if RADOS Gateway is not accessible for the {{ $labels.name }} endpoint.

Tuning	Not required

StackLight LMA

This section describes the alerts available for the StackLight LMA services.

Alertmanager

This section describes the alerts for the Alertmanager service.

- AlertmanagerNotificationFailureWarning
- AlertmanagerAlertsInvalidWarning

AlertmanagerNotificationFailureWarning

Severity	Warning
Summary	Alertmanager has {{ $value }} failed notifications for {{$labels.integration }} on the {{ $labels.instance}} instance for 2 minutes.
Raise condition	increase(alertmanager_notifications_failed_total[2m]) > 0
Description	Raises when Alertmanager fails to send a notification to a specific receiver or channel for the following exemplary reasons: • Slack: wrong API key or channel name • Email: authentication issues or the SMTP server is not available • Webhook: the server is not available
Troubleshooting	Run docker service logs monitoring_alertmanager on any mon node and inspect the Alertmanager logs.
Tuning	Not required

AlertmanagerAlertsInvalidWarning

Severity	Warning
Summary	An average of {{ $value }} Alertmanager {{$labels.integration }} alerts on the {{ $labels.instance }} instance are invalid for 2 minutes.
Raise condition	increase(alertmanager_alerts_invalid_total[2m]) > 0

Description	Raises when Alertmanager receives an alert with errors, for example: • Missing start and end time • Empty labels • Wrong labels or annotations. The key or value must not be empty, must start with a char and must be alphanumeric.
Troubleshooting	Run docker service logs monitoring_alertmanager on any mon node and inspect the Alertmanager logs.
Tuning	Not required

Elasticsearch

This section describes the alerts for the Elasticsearch service.

- ElasticsearchClusterHealthStatusMajor
- ElasticsearchClusterHealthStatusCritical
- ElasticsearchServiceDown
- ElasticsearchServiceDownMinor
- ElasticsearchServiceDownMajor
- ElasticsearchServiceOutage
- ElasticsearchDiskWaterMarkMinor
- ElasticsearchDiskWaterMarkMajor
- ElasticsearchExporterNoDailyLogs

ElasticsearchClusterHealthStatusMajor

<table>
<tr><td>Severity</td><td>Major</td></tr>
<tr><td>Summary</td><td>The Elasticsearch cluster status is YELLOW for 2 minutes.</td></tr>
<tr><td>Raise condition</td><td>elasticsearch_cluster_health_status == 2</td></tr>
<tr><td>Description</td><td>Raises when the Elasticsearch cluster status is YELLOW for 2 minutes, meaning that Elasticsearch has allocated all of the primary shards but some or all of the replicas have not been allocated. For the exact reason, inspect the Elasticsearch logs on the log nodes in /var/log/elasticsearch/elasticsearch.log. To verify the current status of the cluster, run curl -XGET '<host>:<port>/_cat/health?pretty', where host is elasticsearch:client:server:host and port is elasticsearch:client:server:port defined in your model.</td></tr>
<tr><td>Troubleshooting</td><td>Verify the status of the shards using curl -XGET '<host>:<port>/_cat/shards'. For details, see Cluster Allocation Explain API.
If UNASSIGNED shards are present, reallocate the shards by running the following command on the log nodes:<pre>curl -XPUT '<host>:<port>/_cluster/settings?pretty' -H 'Content-Type: \
application/json' -d' { "persistent": \
{"cluster.routing.allocation.enable": "all" }}'</pre>
Manually reallocate the unassigned shards as described in Cluster Reroute.</td></tr>
</table>

Tuning	Not required

ElasticsearchClusterHealthStatusCritical

Severity	Critical
Summary	The Elasticsearch cluster status is RED for 2 minutes.
Raise condition	elasticsearch_cluster_health_status == 3
Description	Raises when the Elasticsearch cluster status is RED for 2 minutes, meaning that some or all of primary shards are not ready. For the exact reason, inspect the Elasticsearch logs on the log nodes in /var/log/elasticsearch/elasticsearch.log. To verify the current status of the cluster, run curl -XGET '<host>:<port>/_cat/health?pretty', where host is elasticsearch:client:server:host and port is elasticsearch:client:server:port defined in your model.

Troubleshooting	• Verify that the Elasticsearch service is running on all log nodes using service elasticsearch status.
	• Verify the status of the shards using curl -XGET '<host>:<port>/_cat/shards'. For details, see Cluster Allocation Explain API.
	• Enable shard allocation by running the following command on the log nodes:
	`curl -XPUT '<host>:<port>/_cluster/settings?pretty' -H 'Content-Type: \`
	`application/json' -d' { "persistent": \`
	`{"cluster.routing.allocation.enable": "all" }}'`
	• Manually reallocate the unassigned shards as described in Cluster Reroute.
	For more troubleshooting details, see Official Elasticsearch documentation.

Tuning	Not required

ElasticsearchServiceDown

Severity	Minor
Summary	The Elasticsearch service on the {{ $labels.host }} node is down.
Raise condition	elasticsearch_up{host=~".*"} == 0

Description	Raises when the Elasticsearch service is down on a log node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the status of the service by running systemctl status elasticsearch on the affected node. • Inspect the Elasticsearch logs in /var/log/elasticsearch/elasticsearch.log for the exact reason.
Tuning	Not required

ElasticsearchServiceDownMinor

Severity	Minor
Summary	30% of Elasticsearch services are down for 2 minutes.
Raise condition	count(elasticsearch_up{host=~".*"} == 0) >= count(elasticsearch_up{host=~".*"}) * 0.3
Description	Raises when the Elasticsearch service is down on more than 30% of the log nodes. By default, 3 log nodes are present, meaning that the service is down on one node.
Troubleshooting	• Inspect the ElasticsearchServiceDown alerts for the host names of the affected nodes. • Verify the Elasticsearch status by running the systemctl status elasticsearch command on the affected node. • Inspect the Elasticsearch logs in /var/log/elasticsearch/elasticsearch.log for the exact reason.
Tuning	Not required

ElasticsearchServiceDownMajor

Severity	Major
Summary	60% of Elasticsearch services are down for 2 minutes.
Raise condition	count(elasticsearch_up{host=~".*"} == 0) >= count(elasticsearch_up{host=~".*"}) * 0.6
Description	Raises when the Elasticsearch service is down on the more than 60% of log nodes. By default, 3 log nodes are present, meaning that the service is down on two nodes.

Troubleshooting	• Inspect the ElasticsearchServiceDown alerts for the host names of the affected nodes. • Verify the Elasticsearch status by running the systemctl status elasticsearch command on the affected node. • Inspect the Elasticsearch logs in /var/log/elasticsearch/elasticsearch.log for the exact reason.
Tuning	Not required

ElasticsearchServiceOutage

Severity	Critical
Summary	All Elasticsearch services within the cluster are down.
Raise condition	count(elasticsearch_up{host=~".*"} == 0) == count(elasticsearch_up{host=~".*"})
Description	Raises when the Elasticsearch service is down on all log nodes.
Troubleshooting	• Inspect the ElasticsearchServiceDown alerts for the host names of the affected nodes. • Verify the Elasticsearch status by running the systemctl status elasticsearch command on the affected node. • Inspect the Elasticsearch logs in /var/log/elasticsearch/elasticsearch.log for the exact reason.
Tuning	Not required

ElasticsearchDiskWaterMarkMinor

Severity	Minor
Summary	The Elasticsearch {{ $labels.instance }} instance uses 60% of disk space on the {{ $labels.host }} node for 5 minutes.
Raise condition	(max by(host, instance) (elasticsearch_fs_total_total_in_bytes) - max by(host, instance) (elasticsearch_fs_total_available_in_bytes)) / max by(host, instance) (elasticsearch_fs_total_total_in_bytes) >= 0.6
Description	Raises when the Elasticsearch instance uses 60% of disk space on the log node for 5 minutes. To verify the available and used disk space, run df -h.

<table>
<tr><td>Troubleshooting</td><td>

Free or extend the disk space on the Elasticsearch partition.
Decrease the default retention period for Elasticsearch as described in Configure Elasticsearch Curator.

</td></tr>
<tr><td>Tuning</td><td>
Typically, you should not change the default value. If the alert is constantly firing, verify the available disk space on the log nodes and adjust the threshold according to the available space. Additionally, in the Prometheus Web UI, use the raise condition query to view the graph for a longer period of time and define the best threshold.
For example, change the threshold to 80%:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre><code>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</code></pre>

Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre><code>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</code></pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre><code>parameters:
  prometheus:
    server:
      alert:
        ElasticsearchDiskWaterMarkMinor:
          if: >-
            (max(elasticsearch_fs_total_total_in_bytes) by (host, instance)\
             - max(elasticsearch_fs_total_available_in_bytes) by \
             (host, instance)) / max(elasticsearch_fs_total_total_in_bytes)\
              by (host, instance) >= 0.8</code></pre>

From the Salt Master node, apply the changes:
<pre><code>salt 'I@prometheus:server' state.sls prometheus.server</code></pre>

Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

ElasticsearchDiskWaterMarkMajor

Severity	Major

Summary	The Elasticsearch {{ $labels.instance }} instance uses 75% of disk space on the {{ $labels.host }} node for 5 minutes.
Raise condition	(max by(host, instance) (elasticsearch_fs_total_total_in_bytes) - max by(host, instance) (elasticsearch_fs_total_available_in_bytes)) / max by(host, instance) (elasticsearch_fs_total_total_in_bytes) >= 0.75
Description	Raises when the Elasticsearch instance uses 75% of disk space on the log node for 5 minutes. To verify the available and used disk space, run df -h.
Troubleshooting	• Free or extend the disk space on the Elasticsearch partition. • Decrease the default retention period for Elasticsearch as described in Configure Elasticsearch Curator.

Tuning	Typically, you should not change the default value. If the alert is constantly firing, verify the available disk space on the log nodes and adjust the threshold according to the available space. Additionally, in the Prometheus Web UI, use the raise condition query to view the graph for a longer period of time and define the best threshold. For example, change the threshold to 90%:
	
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	   1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        ElasticsearchDiskWaterMarkMajor:`
	`          if: >-`
	`            (max(elasticsearch_fs_total_total_in_bytes) by (host, instance)\`
	`             - max(elasticsearch_fs_total_available_in_bytes) by \`
	`             (host, instance)) / max(elasticsearch_fs_total_total_in_bytes)\`
	`              by (host, instance) >= 0.9`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

ElasticsearchExporterNoDailyLogs

Available since 2019.2.10

Severity	Warning
Summary	No new logs sent from a node within the last 3 hours.

Raise condition	(sum by (host) (changes(logs_program_host_doc_count[3h])) or sum by (host) (up{host!=""})*0) == 0
Description	Raises when no new logs were shipped from the {{ $labels.host }} node within the last 3 hours.
Troubleshooting	Verify that Fluentd is operating properly on the affected node.
Tuning	No tuning

Heka

This section describes the alerts for the Heka service.

HekaOutputQueueStalled

Severity	Warning
Summary	The {{ $labels.queue }} queue is stalled on node {{$labels.host }} for more than 1 hour. The corresponding Heka service is either down or stuck.
Raise condition	heka_output_queue_size > 134217728
Description	Raises when Heka freezes and the output queue is larger than 134217728 (128 MB). The host label in the raised alert contains the name of the affected node.
Troubleshooting	Restart the corresponding Heka log collector using service log_collector restart.
Tuning	Not required

InfluxDB

This section describes the alerts for InfluxDB, InfluxDB Relay, and remote storage adapter.

> Warning
>
> InfluxDB, including InfluxDB Relay and remote storage adapter, is deprecated in the Q4`18 MCP release and will be removed in the next release.

- InfluxdbServiceDown
- InfluxdbServicesDownMinor
- InfluxdbServicesDownMajor
- InfluxdbServiceOutage
- InfluxdbSeriesMaxNumberWarning
- InfluxdbSeriesMaxNumberCritical
- InfluxdbHTTPClientErrorsWarning
- InfluxdbHTTPPointsWritesFailWarning
- InfluxdbHTTPPointsWritesDropWarning
- InfluxdbRelayBufferFullWarning
- InfluxdbRelayRequestsFailWarning
- RemoteStorageAdapterMetricsSendingWarning
- RemoteStorageAdapterMetricsIgnoredWarning

InfluxdbServiceDown

Severity	Minor
Summary	The InfluxDB service on the {{ $labels.host }} node is down.
Raise condition	influxdb_up == 0
Description	Raises when the InfluxDB service on one of the mtr nodes is down. The host label in the raised alert contains the host name of the affected node.

Troubleshooting	• Verify the InfluxDB service status on the affected node using systemctl status influxdb. • Inspect the InfluxDB service logs on the affected node using journalctl -xfu influxdb. • Verify the available disk space using df -h.
Tuning	Not required

InfluxdbServicesDownMinor

Severity	Minor
Summary	More than 30% of InfluxDB services are down.
Raise condition	count(influxdb_up == 0) >= count(influxdb_up) * 0.3
Description	Raises when InfluxDB services are down on more than 30% of mtr nodes.
Troubleshooting	• Inspect the InfluxdbServiceDown alerts for the host names of the affected nodes. • Verify the InfluxDB service status on the affected node using systemctl status influxdb. • Inspect the InfluxDB service logs on the affected node using journalctl -xfu influxdb. • Verify the available disk space using df -h.
Tuning	Not required

InfluxdbServicesDownMajor

Severity	Major
Summary	More than 60% of InfluxDB services are down.
Raise condition	count(influxdb_up == 0) >= count(influxdb_up) * 0.6
Description	Raises when InfluxDB services are down on more than 60% of the mtr nodes.

Troubleshooting	• Inspect the InfluxdbServiceDown alerts for the host names of the affected nodes. • Verify the InfluxDB service status on the affected node using systemctl status influxdb. • Inspect the InfluxDB service logs on the affected node using journalctl -xfu influxdb. • Verify the available disk space using df -h.
Tuning	Not required

InfluxdbServiceOutage

Severity	Critical
Summary	All InfluxDB services are down.
Raise condition	count(influxdb_up == 0) == count(influxdb_up)
Description	Raises when InfluxDB services are down on all mtr nodes.
Troubleshooting	• Inspect the InfluxdbServiceDown alerts for the host names of the affected nodes. • Verify the InfluxDB service status on the affected node using systemctl status influxdb. • Inspect the InfluxDB service logs on the affected node using journalctl -xfu influxdb. • Verify the available disk space using df -h.
Tuning	Not required

InfluxdbSeriesMaxNumberWarning

Severity	Warning
Summary	The InfluxDB database contains 950000 time series.
Raise condition	influxdb_database_numSeries >= 950000

Description	Raises when the number of series collected by InfluxDB reaches the threshold of 95%. InfluxDB continues collecting the data. However, reaching the maximum series threshold is critical.
Troubleshooting	• Decrease the retention policy for the affected database. • Remove unused data. • Increase the maximum number of series to keep in the database.

Tuning

Typically, you should not change the default value. If the alert is constantly firing, increase the max_series_per_database parameter to a ten times bigger value.
For example, to change the threshold to 9 500 000 and the number of series to 10 000 000:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

    1. Create a file for alert customizations:

        ```
        touch cluster/<cluster_name>/stacklight/custom/alerts.yml
        ```

    2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

        ```
        classes:
        - cluster.<cluster_name>.stacklight.custom.alerts
        ...
        ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

    ```
    parameters:
      prometheus:
        server:
          alert:
            InfluxdbSeriesMaxNumberWarning:
              if: >-
                influxdb_database_numSeries >= 9500000
    ```

3. In cluster/<cluster_name>/stacklight/telemetry.yml, add:

    ```
    parameters:
      influxdb:
        server:
          data:
            max_series_per_database: 10000000
    ```

4. From the Salt Master node, apply the changes:

    ```
    salt 'I@prometheus:server' state.sls prometheus.server
    salt -C 'I@influxdb:server' influxdb.server
    ```

5. Verify the updated alert definition in the Prometheus web UI.

InfluxdbSeriesMaxNumberCritical

Severity	Critical
Summary	The InfluxDB database contains 1000000 time series. No more series can be saved.
Raise condition	influxdb_database_numSeries >= 1000000
Description	Raises when the number of series collected by InfluxDB reaches the critical threshold of 1M series. InfluxDB is available but cannot collect more data. Any write request to the database ends with the HTTP 500 status code and the max series per database exceeded error message. It is not possible to define the data that has not been recorded. Warning For production environments, after deployment set both the threshold and the max_series_per_database parameter value to 10 000 000.
Troubleshooting	• Decrease the retention policy for the affected database. • Remove unused data. • Increase the maximum number of series to keep in the database.

<table>
<tr><td>Tunin g</td><td>

For example, to change the number of series and the threshold to 10 000 000:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

   ```
   parameters:
     prometheus:
       server:
         alert:
           InfluxdbSeriesMaxNumberCritical:
             if: >-
               influxdb_database_numSeries >= 10000000
   ```

3. In cluster/<cluster_name>/stacklight/telemetry.yml, add:

   ```
   parameters:
     influxdb:
       server:
         data:
           max_series_per_database: 10000000
   ```

4. From the Salt Master node, apply the changes:

   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   salt -C 'I@influxdb:server' influxdb.server
   ```

5. Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

InfluxdbHTTPClientErrorsWarning

Severity	Warning
Summary	An average of 5% of HTTP client requests on the {{ $labels.host }} node fail.

<table>
<tr><td>Raise condition</td><td>rate(influxdb_httpd_clientError[1m]) / rate(influxdb_httpd_req[1m]) * 100 > 5</td></tr>
<tr><td>Description</td><td>Raises when the percentage of client error HTTP requests rate reaches the threshold of 5%, indicating issues with the request format, service performance, or the maximum number of series being reached. The host label in the raised alert contains the host name of the affected node.</td></tr>
<tr><td>Troubleshooting</td><td>

- Inspect InfluxDB logs on the affected node using journalctl -xfu influxdb.
- Verify if the InfluxdbSeriesMaxNumberWarning is firing.

</td></tr>
<tr><td>Tuning</td><td>

For example, to change the threshold to 10%:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
   1. Create a file for alert customizations:
      ```
      touch cluster/<cluster_name>/stacklight/custom/alerts.yml
      ```
   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
      ```
      classes:
      - cluster.<cluster_name>.stacklight.custom.alerts
      ...
      ```
2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
   ```
   parameters:
     prometheus:
       server:
         alert:
           InfluxdbHTTPClientErrorsWarning:
             if: >-
               rate(influxdb_httpd_clientError[1m]) / \
               rate(influxdb_httpd_req[1m]) * 100 > 10
   ```
3. From the Salt Master node, apply the changes:
   ```
   salt 'I@prometheus:server' state.sls prometheus.server
   ```
4. Verify the updated alert definition in the Prometheus web UI.

</td></tr>
</table>

InfluxdbHTTPPointsWritesFailWarning

Severity	Warning
Summary	More than 5% of HTTP points writes on the {{ $labels.host }} node fail.
Raise condition	rate(influxdb_httpd_pointsWrittenFail[1m]) / (rate(influxdb_httpd_pointsWrittenOK[1m]) + rate(influxdb_httpd_pointsWrittenFail[1m]) + rate(influxdb_httpd_pointsWrittenDropped[1m])) * 100 > 5
Description	Raises when the percentage of client failed HTTP write requests reached the threshold of 5%, indicating a non-existing database or reaching of the maximum series threshold. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Inspect the InfluxDB logs on the affected node using journalctl -xfu influxdb. • Verify if the InfluxdbSeriesMaxNumberWarning is firing.

<table>
<tr>
<td>Tuning</td>
<td>
For example, to change the threshold to 10%:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>

Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        InfluxdbHTTPPointsWritesFailWarning:
          if: >-
            rate(influxdb_httpd_pointsWrittenFail[1m]) / \
            (rate(influxdb_httpd_pointsWrittenOK[1m]) + \
            rate(influxdb_httpd_pointsWrittenFail[1m]) + \
            rate(influxdb_httpd_pointsWrittenDropped[1m])) * 100 > 10</pre>

From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>

Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

InfluxdbHTTPPointsWritesDropWarning

Severity	Warning
Summary	More than 5% of HTTP points writes on the {{ $labels.host }} node were dropped.
Raise condition	rate(influxdb_httpd_pointsWrittenDropped[1m]) / (rate(influxdb_httpd_pointsWrittenOK[1m]) + rate(influxdb_httpd_pointsWrittenFail[1m]) + rate(influxdb_httpd_pointsWrittenDropped[1m])) * 100 > 5

<table>
<tr><td>Description</td><td>Raises when the percentage of client HTTP drop measurements requests reaches the threshold of 5%. Dropping of measurements must be a controlled operation, determined by the retention policy or manual actions. This alert is expected during maintenance. Otherwise, investigate the reasons. The host label in the raised alert contains the host name of the affected node.</td></tr>
<tr><td>Troubleshooting</td><td>Inspect the InfluxDB logs on the affected node using journalctl -xfu influxdb.</td></tr>
<tr><td>Tuning</td><td>For example, to change the threshold to 10%:

On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

Create a file for alert customizations:
<pre>touch cluster/<cluster_name>/stacklight/custom/alerts.yml</pre>
Define the new file in cluster/<cluster_name>/stacklight/server.yml:
<pre>classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...</pre>

In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
<pre>parameters:
  prometheus:
    server:
      alert:
        InfluxdbHTTPPointsWritesDropWarning:
          if: >-
            rate(influxdb_httpd_pointsWrittenDropped[1m]) /
            (rate(influxdb_httpd_pointsWrittenOK[1m]) + \
            rate(influxdb_httpd_pointsWrittenFail[1m]) + \
            rate(influxdb_httpd_pointsWrittenDropped[1m])) * 100 > 10</pre>
From the Salt Master node, apply the changes:
<pre>salt 'I@prometheus:server' state.sls prometheus.server</pre>
Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

InfluxdbRelayBufferFullWarning

Severity	Warning

Summary	The InfluxDB Relay {{ $labels.host }} back-end buffer is 80% full.
Raise condition	influxdb_relay_backend_buffer_bytes / 5.36870912e+08 * 100 > 80
Description	Raises when the percentage of InfluxDB Relay summarized buffers usage reaches 80% of the threshold set to 512 MB and may be connected with InfluxDB issues. When the buffer is full, the requests cannot be cached.
Troubleshooting	Increase the buffer size as required.

<table>
<tr><td>Tuning</td><td>
For example, to change the threshold to 90% and the buffer size to 1024mb:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

   1. Create a file for alert customizations:

```
touch cluster/<cluster_name>/stacklight/custom/alerts.yml
```

   2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

```
classes:
- cluster.<cluster_name>.stacklight.custom.alerts
...
```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

```
parameters:
  prometheus:
    server:
      alert:
        InfluxdbRelayBufferFullWarning:
          if: >-
            influxdb_relay_backend_buffer_bytes / 2^20  > 1024 * 0.9
```

3. In the _params section in cluster/<cluster_name>/stacklight/telemetry.yml, specify:

```
influxdb_relay_buffer_size_mb: 1024
```

4. From the Salt Master node, apply the changes:

```
salt 'I@prometheus:server' state.sls prometheus.server
salt -C 'I@influxdb:server' state.sls influxdb.relay
```

5. Verify the updated alert definition in the Prometheus web UI.
</td></tr>
</table>

InfluxdbRelayRequestsFailWarning

Severity	Warning
Summary	An average of 5% of InfluxDB Relay requests on the {{ $labels.host }} node fail.

Raise condition	rate(influxdb_relay_failed_requests_total[1m]) / rate(influxdb_relay_requests_total[1m]) * 100 > 5
Description	Raises when the percentage of InfluxDB Relay failed requests reaches the threshold of 5%, indicating issues with the InfluxDB Relay back end availability.
Troubleshooting	• Inspect the InfluxDB logs on the affected node using journalctl -xfu influxdb. • Inspect the InfluxdbRelayBufferFullWarning alert. • Inspect the InfluxdbSeriesMaxNumberWarning or InfluxdbSeriesMaxNumberCritical alerts.

Tuning	For example, to change threshold to 10%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        InfluxdbRelayRequestsFailWarning:`
	`          if: >-`
	`            rate(influxdb_relay_failed_requests_total[1m]) / \`
	`            rate(influxdb_relay_requests_total[1m]) * 100 > 10`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

RemoteStorageAdapterMetricsSendingWarning

Severity	Warning
Summary	The remote storage adapter metrics on sent to received ratio on the {{ $labels.instance }} instance is less than 0.9.
Raise condition	increase(sent_samples_total{job="remote_storage_adapter"}[1m]) / on (job, instance) increase(received_samples_total[1m]) < 0.9
Description	Raises when the sent to received metrics ratio of the remote storage adapter reaches 90%. If this ratio decreases, the adapter stops sending new metrics to a remote storage.
Troubleshooting	• Verify that the remote storage adapter container is operating by running docker ps on the mon nodes. • Inspect the remote storage service logs by running docker service logs montoring_remote_storage_adapter on any mon node.

<table>
<tr>
<td>Tuning</td>
<td>

For example, change the threshold to 2 per 10 minutes:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.

    1. Create a file for alert customizations:

        ```
        touch cluster/<cluster_name>/stacklight/custom/alerts.yml
        ```

    2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

        ```
        classes:
        - cluster.<cluster_name>.stacklight.custom.alerts
        ...
        ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

    ```
    parameters:
      prometheus:
        server:
          alert:
            RemoteStorageAdapterMetricsSendingWarning:
              if: >-
                increase(sent_samples_total{job="remote_storage_adapter"}[10m])\
                 / on (job, instance) increase(received_samples_total[10m]) < 1
    ```

3. From the Salt Master node, apply the changes:

    ```
    salt 'I@prometheus:server' state.sls prometheus.server
    ```

4. Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

RemoteStorageAdapterMetricsIgnoredWarning

Severity	Warning
Summary	More than 5% of remote storage adapter metrics on the {{ $labels.instance }} instance are invalid.
Raise condition	increase(prometheus_influxdb_ignored_samples_total {job="remote_storage_adapter"}[1m]) / on (job, instance) increase(sent_samples_total[1m]) >= 0.05
Description	Rasies when the ignored to sent metrics ratio of the remote storage adapter reaches the default 5%, indicating that at least 5% of the metrics sent from the remote storage adapter were ignored by InfluxDB.

<table>
<tr>
<td>Troubleshooting</td>
<td>

- Inspect the InfluxDB alerts.
- Inspect the remote storage service logs by running docker service logs montoring_remote_storage_adapter on any mon node.

</td>
</tr>
<tr>
<td>Tuning</td>
<td>

For example, to change the threshold to 10%:

1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
    1. Create a file for alert customizations:

        ```
        touch cluster/<cluster_name>/stacklight/custom/alerts.yml
        ```

    2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:

        ```
        classes:
        - cluster.<cluster_name>.stacklight.custom.alerts
        ...
        ```

2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:

    ```
    parameters:
      prometheus:
        server:
          alert:
            RemoteStorageAdapterMetricsIgnoredWarning:
              if: >-
                increase(prometheus_influxdb_ignored_samples_total\
                {job="remote_storage_adapter"}[1m]) / on (job, instance) \
                increase(sent_samples_total[1m]) >= 0.1
    ```

3. From the Salt Master node, apply the changes:

    ```
    salt 'I@prometheus:server' state.sls prometheus.server
    ```

4. Verify the updated alert definition in the Prometheus web UI.

</td>
</tr>
</table>

Kibana

This section describes the alerts for the Kibana service.

- KibanaProcessDown
- KibanaProcessesDownMinor
- KibanaProcessesDownMajor
- KibanaServiceOutage

KibanaProcessDown

Severity	Minor
Summary	The Kibana process on the {{ $labels.host }} node is down.
Raise condition	procstat_running{process_name="kibana"} == 0
Description	Raises when Telegraf cannot find a running kibana process, typically indicating that the Kibana process is down on one node. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	• Verify the Kibana service status on the affected node using systemctl status kibana. • Inspect the Kibana service logs using journalctl -xfu kibana.
Tuning	Not required

KibanaProcessesDownMinor

Severity	Minor
Summary	More than 30% of Kibana processes are down.
Raise condition	count(procstat_running{process_name="kibana"} == 0) >= count(procstat_running{process_name="kibana"}) * 0.3
Description	Raises when Telegraf cannot find running kibana processes on more than 30% of the log hosts.

Troubleshooting	• Inspect the KibanaProcessDown alerts for the host names of the affected nodes. • Verify the Kibana service status on the affected node using systemctl status kibana. • Inspect the Kibana service logs using journalctl -xfu kibana.
Tuning	Not required

KibanaProcessesDownMajor

Severity	Major
Summary	More than 60% of Kibana processes are down.
Raise condition	count(procstat_running{process_name="kibana"} == 0) >= count(procstat_running{process_name="kibana"}) * 0.6
Description	Raises when Telegraf cannot find running kibana processes with on more than 60% of the log hosts.
Troubleshooting	• Inspect the KibanaProcessDown alerts for the host names of the affected nodes. • Verify the Kibana service status on the affected node using systemctl status kibana. • Inspect the Kibana service logs using journalctl -xfu kibana.
Tuning	Not required

KibanaServiceOutage

Severity	Critical
Summary	All Kibana processes are down.
Raise condition	count(procstat_running{process_name="kibana"} == 0) == count(procstat_running{process_name="kibana"})
Description	Raises when Telegraf cannot find running kibana processes on all the log hosts.

Troubleshooting	• Inspect the KibanaProcessDown alerts for the host names of the affected nodes. • Verify the Kibana service status on the affected node using systemctl status kibana. • Inspect the Kibana service logs using journalctl -xfu kibana.
Tuning	Not required

MongoDB

This section describes the MongoDB alerts.

- MongoDBServiceDown
- MongoDBServiceOutage
- MongoDBNoPrimaryMember

MongoDBServiceDown

Severity	Minor
Summary	The MongoDB service on the {{ $labels.host }} node is down for 1 minute.
Raise condition	mongodb_up == 0
Description	Raises when the MongoDB process is in the DOWN state on a host. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Inspect the MongoDB logs using journalctl -u mongodb on the affected node or in /var/log.
Tuning	Not required

MongoDBServiceOutage

Severity	Critical
Summary	All MongoDB services are down for 1 minute.
Raise condition	count(mongodb_up == 0) == count(mongodb_up)
Description	Raises when the MongoDB processes are in the DOWN state on all nodes.
Troubleshooting	• Inspect the MongoDBServiceDown alert for the affected node. • Inspect the MongoDB logs using journalctl -u mongodb on the affected node or in /var/log.

Tuning	Not required

MongoDBNoPrimaryMember

Severity	Critical
Summary	MongoDB cluster has no primary member for 1 minute.
Raise condition	absent(mongodb_state == 1)
Description	Raises when the MongoDB cluster has no primary member.
Troubleshooting	Inspect the MongoDB logs using journalctl -u mongodb on a node with MongoDB or in /var/log.
Tuning	Not required

Prometheus

This section describes the alerts for the Prometheus service.

- PrometheusTargetDown
- PrometheusTargetSamplesOrderWarning
- PrometheusTargetSamplesBoundsWarning
- PrometheusTargetSamplesDuplicateWarning
- PrometheusDataIngestionWarning
- PrometheusRemoteStorageQueueFullWarning
- PrometheusRelayServiceDown
- PrometheusRelayServiceDownMajor
- PrometheusRelayServiceOutage
- PrometheusLTSServiceDown
- PrometheusLTSServiceDownMajor
- PrometheusLTSServiceOutage
- PrometheusRuleEvaluationsFailed

PrometheusTargetDown

Severity	Critical
Summary	The Prometheus target for the {{ $labels.job }} job on the {{ $labels.host or $labels.instance }} node is down for 2 minutes.
Raise condition	up != 1
Description	Raises when Prometheus fails to scrape a target for 2 minutes. The reasons depend on the target type. For example, Telegraf-related or connectivity issues, Fluentd misconfiguration, issues with libvirt or JMX Exporter.
Troubleshooting	Depending on the target type: • Inspect the Telegraf logs using journalctl -u telegraf. • Inspect the Fluentd logs using journalctl -u td-agent. • Inspect the libvirt-exporter logs using journalctl -u libvirt-exporter. • Inspect the jmx-exporter logs using journalctl -u jmx-exporter.

Tuning	Not required

PrometheusTargetSamplesOrderWarning

Removed since the 2019.2.4 maintenance update.

Severity	Warning
Summary	{{ $value }} Prometheus samples on the {{$labels.instance }} instance are out of order (as measured over the last minute).
Raise condition	increase(prometheus_target_scrapes_sample_out_of_order_total[1m]) > 0
Description	Raises when Prometheus observes time series samples with a wrong order. Warning The alert has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, disable this alert.
Tuning	Disable the alert as described in Manage alerts.

PrometheusTargetSamplesBoundsWarning

Removed since the 2019.2.4 maintenance update.

Severity	Warning
Summary	{{ $value }} Prometheus samples on the {{$labels.instance }} instance have time stamps out of bounds (as measured over the last minute).
Raise condition	increase(prometheus_target_scrapes_sample_out_of_bounds_total[1m]) > 0
Description	Raises when Prometheus observes samples with time stamps greater than the current time. Warning The alert has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, disable this alert.

Tuning	Disable the alert as described in Manage alerts.

PrometheusTargetSamplesDuplicateWarning

Removed since the 2019.2.4 maintenance update.

Severity	Warning
Summary	{{ $value }} Prometheus samples on the {{$labels.instance }} instance have duplicate time stamps (as measured over the last minute).
Raise condition	increase(prometheus_target_scrapes_sample_duplicate_timestamp_total [1m]) > 0
Description	Raises when Prometheus observes samples with duplicated time stamps. Warning The alert has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, disable this alert.
Tuning	Disable the alert as described in Manage alerts.

PrometheusDataIngestionWarning

Removed since the 2019.2.4 maintenance update.

Severity	Warning
Summary	The Prometheus service writes on the {{$labels.instance }} instance do not keep up with the data ingestion speed for 10 minutes.
Raise condition	prometheus_local_storage_rushed_mode != 0

Description	Raises when the Prometheus service writes do not keep up with the data ingestion speed for 10 minutes. Warning The alert is deprecated for Prometheus versions newer than 1.7 and has been removed starting from the 2019.2.4 maintenance update. For the existing MCP deployments, disable this alert.
Tuning	Disable the alert as described in Manage alerts.

PrometheusRemoteStorageQueueFullWarning

Severity	Warning
Summary	The Prometheus remote storage queue on the {{$labels.instance }} instance is 75% full for 2 minutes.
Raise condition	prometheus_remote_storage_queue_length / prometheus_remote_storage_queue_capacity * 100 > 75
Description	Raises when Prometheus remote write queue is 75% full.
Troubleshooting	Inspect the remote write service configuration in the remote_write section in /srv/volumes/local/prometheus/config/prometheus.yml.

Tuning	For example, to change the warning threshold to 90%:
	1. On the cluster level of the Reclass model, create a common file for all alert customizations. Skip this step to use an existing defined file.
	  1. Create a file for alert customizations:
	`touch cluster/<cluster_name>/stacklight/custom/alerts.yml`
	  2. Define the new file in cluster/<cluster_name>/stacklight/server.yml:
	`classes:`
	`- cluster.<cluster_name>.stacklight.custom.alerts`
	`...`
	2. In the defined alert customizations file, modify the alert threshold by overriding the if parameter:
	`parameters:`
	`  prometheus:`
	`    server:`
	`      alert:`
	`        PrometheusRemoteStorageQueueFullWarning:`
	`          if: >-`
	`            prometheus_remote_storage_queue_length / \`
	`            prometheus_remote_storage_queue_capacity * 100 > 90`
	3. From the Salt Master node, apply the changes:
	`salt 'I@prometheus:server' state.sls prometheus.server`
	4. Verify the updated alert definition in the Prometheus web UI.

PrometheusRelayServiceDown

Severity	Minor
Summary	The Prometheus Relay service on the {{$labels.host}} node is down for 2 minutes.
Raise condition	procstat_running{process_name="prometheus-relay"} == 0
Description	Raises when Telegraf cannot find running prometheus-relay processes on any mtr host.

Troubleshooting	• Verify the status of the Prometheus Relay service on the affected node using service prometheus-relay status. • Inspect the Prometheus Relay logs on the affected node using journalctl -u prometheus-relay.
Tuning	Not required

PrometheusRelayServiceDownMajor

Severity	Major
Summary	More than 50% of Prometheus Relay services are down for 2 minutes.
Raise condition	count(procstat_running{process_name="prometheus-relay"} == 0) >= count (procstat_running{process_name="prometheus-relay"}) * 0.5
Description	Raises when Telegraf cannot find running prometheus-relay processes on more than 50% of the mtr hosts.
Troubleshooting	• Inspect the PrometheusRelayServiceDown alerts for the host names of the affected nodes. • Verify the status of the Prometheus Relay service on the affected node using service prometheus-relay status. • Inspect the Prometheus Relay logs on the affected node using journalctl -u prometheus-relay.
Tuning	Not required

PrometheusRelayServiceOutage

Severity	Critical
Summary	All Prometheus Relay services are down for 2 minutes.
Raise condition	count(procstat_running{process_name="prometheus-relay"} == 0) == count (procstat_running{process_name="prometheus-relay"})
Description	Raises when Telegraf cannot find running prometheus-relay processes on all mtr hosts.

Troubleshooting	• Inspect the PrometheusRelayServiceDown alerts for the host names of the affected nodes. • Verify the status of the Prometheus Relay service on the affected node using service prometheus-relay status. • Inspect the Prometheus Relay logs on the affected node using journalctl -u prometheus-relay.
Tuning	Not required

PrometheusLTSServiceDown

Severity	Minor
Summary	The Prometheus long-term storage service on the {{$labels.host}} node is down for 2 minutes.
Raise condition	procstat_running{process_name="prometheus"} == 0
Description	Raises when Telegraf cannot find running prometheus processes on any mtr host.
Troubleshooting	• Verify the status of the Prometheus service on the affected node using service prometheus status. • Inspect the Prometheus logs on the affected node using journalctl -u prometheus.
Tuning	Not required

PrometheusLTSServiceDownMajor

Severity	Major
Summary	More than 50% of the Prometheus long-term storage services are down for 2 minutes.
Raise condition	count(procstat_running{process_name="prometheus"} == 0) >= count (procstat_running{process_name="prometheus"}) * 0.5
Description	Raises when Telegraf cannot find running prometheus processes on more than 50% of the mtr hosts.

Troubleshooting	• Inspect the PrometheusLTSServiceDown alerts for the host names of the affected nodes. • Verify the status of the Prometheus service on the affected node using service prometheus status. • Inspect the Prometheus logs on the affected node using journalctl -u prometheus.
Tuning	Not required

PrometheusLTSServiceOutage

Severity	Critical
Summary	All Prometheus long-term storage services are down for 2 minutes.
Raise condition	count(procstat_running{process_name="prometheus"} == 0) == count (procstat_running{process_name="prometheus"})
Description	Raises when Telegraf cannot find running prometheus processes on all mtr hosts.
Troubleshooting	• Inspect the PrometheusLTSServiceDown alerts for the host names of the affected nodes. • Verify the status of the Prometheus service on the affected node using service prometheus status. • Inspect the Prometheus logs on the affected node using journalctl -u prometheus.
Tuning	Not required

PrometheusRuleEvaluationsFailed

Available starting from the 2019.2.7 maintenance update

Severity	Warning
Summary	The Prometheus server for the {{ $labels.job }} job on the {{ or $labels.host $labels.instance }} node has failed evaluations for recording rules. Verify the rules state in the Status/Rules section of the Prometheus web UI.
Raise condition	rate(prometheus_rule_evaluation_failures_total[5m]) > 0

Description	Raises when the number of evaluation failures of Promethues recording rules continuously increases for 10 minutes. The issue typically occurs once you reload the Prometheus service after configuration changes.
Troubleshooting	Verify the syntax and metrics in the recently added custom Prometheus recording rules in the cluster model.
Tuning	Not required

Salesforce notifier

This section describes the alerts for the Salesforce notifier service.

- SfNotifierDown
- SfNotifierAuthFailure
- SfNotifierErrorsWarning

SfNotifierDown

Severity	Critical
Summary	The sf-notifier service is down for 2 minutes.
Raise condition	absent(sf_auth_ok) == 1
Description	Rasies when the Docker container with the sf-notifier service is down for 2 minutes. In this case, notifications to Salesforce cannot be sent.
Troubleshooting	• Inspect the Salesforce notifier service logs using docker service logs monitoring_sf_notifier. • Since the issue may be connected to Docker or system, inspect the Docker and system alerts.
Tuning	Not required

SfNotifierAuthFailure

Severity	Critical
Summary	The sf-notifier service fails to authenticate to Salesforce for 2 minutes.
Raise condition	sf_auth_ok == 0
Description	Raises when the sf-notifier service fails to authenticate to Salesforce. In this case, notifications to Salesforce cannot be sent. The alert fires after 2 minutes.

Troubleshooting	• Verify the authentication credentials.
	• Verify that the instance type is sandbox.
	• Update the cluster model parameters and redeploy the sf-container:
	1. On the cluster level of the Reclass model, update the cluster model parameters as required, paying attention to sandbox and password.
	2. Refresh Salt:
	`salt '*' saltutil.refresh_pillar`
	3. Redeploy the sf-notifier container:
	`salt -C 'I@promethes:server and I@docker:server' state.sls docker`

Tuning	Not required

SfNotifierErrorsWarning

Removed starting from the MCP 2019.2.2 update

Severity	Warning
Summary	An average of {{ $value }} sf-notifier error requests appear for 2 minutes.
Raise condition	increase(sf_error_count_total[2m]) > 0
Description	Raises when the number of Salesforce notifier errors started increasing over the last 2 minutes, indicating an issue with sending of the alert notifications to Salesforce. The issue is typically connected with the limits set in the Salesforce instance. The alert fires after 2 minutes. Warning The alert has been removed starting from the 2019.2.2 maintenance update. For the existing MCP deployments, disable this alert.
Troubleshooting	Inspect the sf-notifier service logs in /srv/volumes/local/sf_notifier/logs/sfnotifier.log on the mon node that runs the service container. If the logs contain many entries with Salesforce exceptions, the issue can be connected with Salesforce instance limits or the account used by sf-notifier.

Tuning	Disable the alert as described in Manage alerts.

Telegraf

This section describes the alerts for the Telegraf service.

- TelegrafGatherErrors
- TelegrafRemoteGatherErrors

TelegrafGatherErrors

Available starting from the 2019.2.5 maintenance update

Severity	Major
Summary	Telegraf failed to gather metrics.
Raise condition	• In 2019.2.9 and prior: rate(internal_agent_gather_errors[10m]) > 0 • In 2019.2.10 and newer: rate(internal_agent_gather_errors{job!="remote_agent"}[10m]) > 0
Description	Raises when Telegraf has gathering errors on a node for the last 10 minutes. The host label in the raised alert contains the host name of the affected node.
Troubleshooting	Inspect the Telegraf logs by running journalctl -u telegraf on the affected node.
Tuning	Not required

TelegrafRemoteGatherErrors

Available starting from the 2019.2.10 maintenance update

Severity	Major
Summary	Remote Telegraf failed to gather metrics.
Raise condition	rate(internal_agent_gather_errors{job="remote_agent"}[10m]) > 0
Description	Raises when remote Telegraf has gathering errors for the last 10 minutes.
Troubleshooting	Inspect the Telegraf monitoring_remote_agent service logs.

Tunin g	Not required

Alerts that require tuning

After deploying StackLight LMA, you may need to customize some of the predefined alerts depending on the needs of your MCP deployment. This section provides the list of alerts that require customization after deployment. Some other alerts can also be configured as required. For an entire list of alerts and their tuning capabilities, see Available StackLight LMA alerts.

Component	Alert
System	1. NetdevBudgetRanOutsWarning 2. PacketsDroppedByCpuMinor 3. PacketsDroppedByCpuWarning 4. SystemMemoryFullMajor 5. SystemMemoryFullWarning 6. SystemRxPacketsDroppedTooHigh 7. SystemTxPacketsDroppedTooHigh 8. SystemTxPacketsErrorTooHigh Available since 2019.2.15 9. SystemRxPacketsErrorTooHigh Available since 2019.2.15 10. SystemCpuStealTimeWarning Available since 2019.2.6 11. SystemCpuStealTimeCritical Available since 2019.2.6 12. OVSTooManyPortRunningOnAgent Available since 2019.2.6
OpenStack	1. IronicErrorLogsTooHigh Available since 2019.2.6 2. RabbitmqFdUsageWarning Available since 2019.2.6 3. RabbitmqFdUsageCritical Available since 2019.2.6
OpenContrail	1. ContrailVrouterLLSSessionsChangesTooHigh 2. ContrailVrouterLLSSessionsTooHigh 3. ContrailVrouterDNSXMPPSessionsChangesTooHigh 4. ContrailVrouterDNSXMPPSessionsTooHigh 5. ContrailVrouterXMPPSessionsChangesTooHigh 6. ContrailVrouterXMPPSessionsTooHigh 7. ContrailXMPPSessionsChangesTooHigh 8. ContrailXMPPSessionsTooHigh

Ceph	**Note** Ceph prediction alerts are available starting from the 2019.2.3 maintenance update and must be enabled manually. 1. CephPredictOsdIOPSthreshold 2. CephPredictOsdIOPSauto 3. CephPredictOsdWriteLatency 4. CephPredictOsdReadLatency 5. CephPredictPoolSpace 6. CephPredictPoolIOPSthreshold 7. CephPredictPoolIOPSauto
InfluxDB Deprecated in Q4`18	InfluxdbSeriesMaxNumberCritical

Generate the list of alerts for a particular deployment

To view the list of available alerts, you can automatically generate the documentation for your particular MCP cluster deployment. Alternatively, you can see the list of alerts using the Prometheus web UI.

To generate the documentation:

1. Log in to the Salt Master node.
2. Run the Sphinx Salt state:

   ```
   salt -C 'I@sphinx:server' state.sls sphinx
   ```

3. Refer to the deployment plan to obtain the public VIP associated with the proxy nodes. For details, see openstack_proxy_address in the Product related parameters subsection of the Create a deployment metadata model using the Model Designer UI section in MCP Deployment Guide.
4. Paste the obtained VIP with the port 8090 to a web browser to access the generated documentation.
5. Navigate to the Functional Alarms Definitions section to see the list of alerts for your particular MCP cluster deployment.

Add new features to an existing StackLight LMA deployment

This section describes how to install new functionality on an existing StackLight LMA deployment or integrate StackLight LMA with additional services.

For example, the Alerta service installs automatically when you deploy StackLight LMA but you must install Alerta manually if you already have a running StackLight LMA deployment without Alerta.

Install Alerta

Alerta is a tool that receives, consolidates, and deduplicates the alerts sent by Alertmanager and visually represents them through a web UI. Alerta provides an overview of the most recent and watched alerts, and enables you to group or filter the alerts according to your needs.

To install Alerta on an existing StackLight LMA deployment:

1. Log in to the Salt Master node.
2. Update the system level of your Reclass model.
3. In the stacklight/server.yml file, add the following classes:

   ```
   - system.mongodb.server.cluster
   - system.prometheus.alerta
   - system.prometheus.alertmanager.notification.alerta
   - system.prometheus.server.alert.alerta_relabel
   ```

4. In the stacklight/client.yml file:

   1. Add the system.docker.swarm.stack.monitoring.alerta class.
   2. Add the following parameters:

      ```
      _param:
        docker_image_alerta: docker-prod-local.artifactory.mirantis.com/mirantis/external/alerta-web:stable
      ```

5. In the stacklight/init.yml file, specify the following parameters:

   ```
   alerta_admin_username: admin@alerta.io
   alerta_admin_password: password
   ```

6. In the stacklight/proxy.yml file, add the following class:

   ```
   - system.nginx.server.proxy.monitoring.alerta
   ```

7. Refresh the pillar:

   ```
   salt '*' saltutil.refresh_pillar
   ```

8. Install MongoDB and the Alerta container:

   ```
   salt -C 'I@mongodb:server' state.sls mongodb.server
   sleep 30
   salt -C 'I@mongodb:server' state.sls mongodb.cluster
   salt -C 'I@prometheus:server' state.sls prometheus -b 1
   salt -C 'I@docker:swarm:role:master and I@prometheus:server' state.sls docker
   salt -C `I@nginx:server` state.sls nginx
   ```

Enable Gainsight integration

Caution!

Starting from the MCP 2019.2.9 update, the Gainsight integration service is considered as deprecated.

Gainsight is a customer relationship management (CRM) tool and extension for Salesforce. Gainsight integration service queries Prometheus for the metrics data and sends the data to Gainsight. Mirantis uses the collected data for further analysis and reports to improve the quality of customer support. For more information, see MCP Reference Architecture: StackLight LMA components.

Note

Gainsight formats the data using Single Quote for Quote Char and commas as separators.

To enable Gainsight integration service on an existing StackLight LMA deployment:

1. Log in to the Salt Master node.
2. Update the system level of your Reclass model.
3. Add the classes and parameters to stacklight/client.yml as required:
    1. For OpenStack environments, add the default Openstack-related metrics:

       ```
       - system.prometheus.gainsight.query.openstack
       ```

    2. Add the main Gainsight class:

       ```
       - system.docker.swarm.stack.monitoring.gainsight
       ```

    3. Specify the following parameters:

       ```
       parameters:
         _param:
           docker_image_prometheus_gainsight: docker-prod-local.artifactory.mirantis.com/openstack-docker/gainsight:${_param:mcp_version}
           gainsight_csv_upload_url: <URL_to_Gainsight_API>
           gainsight_account_id: <customer_account_ID_in_Salesforce>
           gainsight_environment_id: <customer_environment_ID_in_Salesforce>
           gainsight_app_org_id: <Mirantis_organization_ID_in_Salesforce>
           gainsight_access_key: <Gainsight_access_key>
           gainsight_job_id: <Gainsight_job_ID>
           gainsight_login: <Gainsight_login>
           gainsight_csv_retention: <retention_in_days>
       ```

> Note
>
> To obtain the values for the above parameters, contact Mirantis Customer Success Team through cs-team@mirantis.com.
>
> The retention period for CSV files is set to 180 days by default.

4. Optional. Customize the frequency of CSV uploads to Gainsight by specifying the duration parameter in the prometheus block.

   Example:

   ```
   _param:
   prometheus:
     gainsight:
       crontab:
         duration: '0 0 * * *'
   ```

4. Refresh Salt grains and pillars:

   ```
   salt '*' state.sls salt.minion.grains && salt '*' mine.update && salt '*' saltutil.refresh_pillar
   ```

5. Deploy the configuration files for the Gainsight Docker service:

   ```
   salt -C 'I@docker:swarm' state.sls prometheus.gainsight
   ```

6. Deploy the new monitoring_gainsight Docker service:

   ```
   salt -C 'I@docker:swarm:role:master' state.sls docker.client
   ```

Enable monitoring of the Open vSwitch processes

> Warning
>
> This feature is available starting from the MCP 2019.2.3 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

If you have deployed Neutron Open vSwitch (OVS) as a networking solution for your OpenStack environment, you can enable StackLight LMA to monitor the OVS processes and to issue an alert if the memory consumption of an OVS process exceeds 20% and 30% set by default. The procedure below implies updating of the monitoring configuration for the nodes that run OVS, typically cmp and gtw.

To enable monitoring of the OVS processes:

1. Log in to the Salt Master node.
2. Verify that OVS is enabled:

   ```
   salt -C "I@linux:network:bridge:openvswitch" test.ping
   ```

   The command output displays the nodes that run openvswitch, for example, cmp and gtw.
3. Open your Git project repository with the Reclass model on the cluster level.
4. In openstack/compute/init.yml and openstack/gateway.yml, specify the following parameters:

   ```
   parameters:
     telegraf:
       agent:
         input:
           procstat:
             process:
               ovs-vswitchd:
                 exe: ovs-vswitchd
     prometheus:
       server:
         alert:
           ProcessOVSmemoryWarning:
             if: procstat_memory_vms{process_name="ovs-vswitchd"} / on(host) mem_total > 0.2
             for: 5m
             labels:
               severity: warning
               service: ovs
             annotations:
               summary: "ovs-vswitchd takes more than 20% of system memory"
   ```

```
    description: "ovs-vswitchd takes more than 20% of system memory"
  ProcessOVSmemoryCritical:
    if: procstat_memory_vms{process_name="ovs-vswitchd"} / on(host) mem_total > 0.3
    for: 5m
    labels:
      severity: critical
      service: ovs
    annotations:
      summary: "ovs-vswitchd takes more than 30% of system memory"
      description: "ovs-vswitchd takes more than 30% of system memory"
```

5. Apply the changes:

   1. Refresh Salt pillars:

      ```
      salt '*' saltutil.refresh_pillar
      ```

   2. Add the Telegraf configuration:

      ```
      salt -C "I@linux:network:bridge:openvswitch" state.sls telegraf.agent
      ```

   3. Add the Prometheus alerts:

      ```
      salt 'mon*' state.sls prometheus.server
      ```

Enable SSL certificates monitoring

Warning

This feature is available starting from the MCP 2019.2.3 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

If you use SSL certificates in your MCP deployment, you can configure StackLight LMA to monitor such certificates and issue an alert when a certificate is due to expire. By default, the alerts raise if a certificate expires less than in 60 and 30 days. This allows for generating a new certificate and replacing the existing one on time to prevent from a cluster outage caused by an expired certificate.

To enable SSL certificates monitoring:

1. Log in to the Salt Master node.
2. Verify that you have updated the salt-formulas-salt package.
3. Update the Salt mine:

```
salt -C 'I@salt:minion' state.sls salt.minion.grains
salt -C 'I@salt:minion' mine.update
```

4. Update the Telegraf configuration:

```
salt -C 'I@telegraf:agent' state.sls telegraf
```

5. Update the Prometheus configuration:

```
salt -C 'I@prometheus:server and I@docker:swarm' state.sls prometheus.server
```

Enable SMART disk monitoring

> Warning
>
> This feature is available starting from the MCP 2019.2.3 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

If your MCP cluster includes physical disks that support Self-Monitoring, Analysis and Reporting Technology (SMART), you can configure StackLight LMA to monitor such disks by parsing their SMART data and to raise alerts if disk errors occur. By default, all disks on the bare metal servers will be scanned.

To enable SMART disk monitoring:

1. Log in to the Salt Master node.
2. Verify that you have updated the salt-formulas-linux package.
3. Install the smartmontools package as a required dependency:

   ```
   salt -C 'I@salt:minion' cmd.run 'apt update; apt install -y smartmontools'
   ```

4. (Optional) Modify the default parameters per node or server as required, for example, to exclude a specific device from the checks.

   ```
   (...)
   parameters:
     _param:
     (...)
     telegraf:
       agent:
         input:
           smart:
             excludes:
               - /dev/sdd
   ```

5. Refresh Salt pillar:

   ```
   salt -C 'I@salt:minion' saltutil.refresh_pillar
   ```

6. Update the Salt mine:

   ```
   salt -C 'I@salt:minion' state.sls salt.minion.grains
   salt -C 'I@salt:minion' mine.update
   ```

7. Update the Telegraf configuration:

©2025, Mirantis Inc.

```
salt -C 'I@telegraf:agent' state.sls telegraf
```

8. Update the Prometheus configuration:

```
salt -C 'I@prometheus:server and I@docker:swarm' state.sls prometheus.server
```

Enable Prometheus Elasticsearch exporter

> Note
>
> This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

Prometheus Elasticsearch exporter queries the Elasticsearch data and exposes it as Prometheus metrics that you can view in the Prometheus web UI. This section describes how to install Prometheus Elasticsearch exporter on an existing MCP cluster. For new clusters starting from the MCP 2019.2.4 maintenance update, Prometheus Elasticsearch exporter is enabled by default.

To enable Prometheus Elasticsearch exporter on an existing MCP cluster:

1. Open your Git project repository with Reclass model on the cluster level.
2. In cluster/<cluster_name>/stacklight/client.yml, add the following class:

```
classes:
- system.docker.swarm.stack.monitoring.elasticsearch_exporter
```

3. In cluster/<cluster_name>/stacklight/server.yml, add the following classes:

```
classes:
- system.prometheus.server.target.dns.elasticsearch_exporter
- system.prometheus.elasticsearch_exporter
- system.prometheus.elasticsearch_exporter.queries.compute
```

4. Log in to the Salt Master node.
5. Refresh the Salt pillar:

```
salt '*' saltutil.refresh_pillar
```

6. Add the prometheus-es-exporter directory structure and configuration:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.elasticsearch_exporter
```

7. Deploy Prometheus Elasticsearch exporter:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls docker.client
```

8. Verify that the container has been deployed:

```
salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run 'docker service ps monitoring_elasticsearch_exporter'
```

9. Verify that querying of the Elasticsearch cluster does not cause errors:

```
salt -C 'I@docker:swarm:role:master and I@prometheus:server' cmd.run 'docker service logs monitoring_elasticsearch_exporter'
```

10. Add Prometheus Elasticsearch exporter to Prometheus targets:

```
salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server
```

Enable TLS for StackLight LMA

Note

This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

To assure the confidentiality and integrity of the communication between Prometheus and Telegraf, Fluentd and Elasticsearch inside your MCP deployment, you can use cryptographic protective measures, such as the Transport Layer Security (TLS) protocol. In this case, Prometheus scrapes the data from Telegraf and Fluentd sends data to Elasticsearch or Elasticsearch VIP endpoint through encrypted channels. This section describes how to enable TLS v1.2 for an existing StackLight LMA deployment to provide message integrity with SHA384 MAC and RSA TLS certificate signature verification.

Warning

The functionality does not cover encryption of the traffic between HAProxy and Elasticsearch.

To enable TLS for StackLight LMA:

1. Open your project Git repository with Reclass model on the cluster level.
2. In infra/config/nodes.yml, add the following classes to the stacklight_log_node01 section:

```
stacklight_log_node01:
  classes:
  - system.elasticsearch.client.single
  - system.kibana.client.single
  - system.vnf_onboarding.common.kibana
  - system.elasticsearch.client.ssl
  - system.kibana.client.ssl
```

3. In infra/init.yml, add the following classes:

```
- system.salt.minion.cert.fluentd_prometheus
- system.fluentd.label.default_metric.prometheus_ssl
- system.salt.minion.cert.telegraf_agent
- system.telegraf.agent.output.prometheus_client_ssl
```

4. In stacklight/init.yml, specify the following parameter:

```
fluentd_elasticsearch_scheme: https
```

5. In stacklight/log.yml:

   1. Replace the system.haproxy.proxy.listen.stacklight.elasticsearch class with the following one:

      ```
      system.haproxy.proxy.listen.stacklight.elasticsearch_ssl
      ```

   2. Add the following classes:

      ```
      - system.kibana.server.ssl
      - system.salt.minion.cert.elasticsearch
      ```

6. Log in to the Salt Master node.
7. Apply the following states one by one:

   ```
   salt -C "I@salt:master" state.sls reclass
   salt -C "I@salt:minion" state.sls salt.minion.cert
   salt -C "I@salt:minion" state.sls salt.minion.grains
   salt -C "I@salt:minion" mine.update
   salt -C "I@fluentd:agent" state.sls fluentd
   salt -C "I@docker:swarm:role:master and I@prometheus:server" state.sls prometheus.server
   salt -C "I@elasticsearch:server" state.sls haproxy
   salt -C "I@elasticsearch:server"  state.sls elasticsearch.server
   salt -C "I@kibana:server" state.sls kibana.server
   ```

Enable Ironic monitoring

> Note
>
> This feature is available starting from the MCP 2019.2.6 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

If you have deployed Ironic in your OpenStack environment, you can enable StackLight LMA to monitor Ironic processes and health.

To enable Ironic monitoring:

1. Log in to the Salt Master node.
2. Verify that Ironic nodes are up and running:

   ```
   salt -C 'I@ironic:api' test.ping
   ```

3. Update the Telegraf configuration:

   ```
   salt -C 'I@ironic:api' state.sls telegraf
   ```

4. Update the td-agent configuration:

   ```
   salt -C 'I@ironic:api' state.sls fluentd
   ```

5. Customize the IronicErrorLogsTooHigh alert as required.
6. Update the Prometheus alerts configuration:

   ```
   salt -C 'I@docker:swarm and I@prometheus:server' state.sls prometheus.server -b 1
   ```

7. Update the Grafana dashboards:

   ```
   salt -C 'I@grafana:client' state.sls grafana.client
   ```

Back up and restore

This section describes how to back up and restore MCP Control Plane.

Back up and restore the Salt Master node

This section describes how to create backups of the Salt Master node using Backupninja. During the Salt Master node backup, Backupninja creates backup configuration scripts containing the data about PKI and CA certificates located in the /etc/salt/pki and /etc/pki/ca directories as well as the cluster model metadata located in the /srv/salt/reclass directory. Using these backup scripts, you can easily restore your Salt Master node in case of hardware and software failures.

Back up and restore the Salt Master node prior to 2019.2.5

This section describes how to back up and restore the Salt Master node prior to the MCP 2019.2.5 maintenance update.

Enable a backup schedule for the Salt Master node using Backupninja

This section describes how to create a backup schedule for the Salt Master node using the Backupninja utility. By default, Backupninja runs daily at 1:00 AM.

To create a backup schedule for the Salt Master node using Backupninja:

1. Log in to the Salt Master node.
2. In cluster/infra/config/init.yml, verify that the following classes and parameters are present:

   ```
   classes:
   - system.backupninja.client.single
   - system.openssh.client.root
   parameters:
     _param:
       backupninja_backup_host: <IP>
     salt:
       master:
         backup: true
       minion:
         backup: true
   ```

   > **Note**
   >
   > The backupninja_backup_host parameter is the IP address of the server where Backupninja runs. For example, the kvm03 node. To obtain this IP address or the host name, run salt -C 'I@backupninja:server' grains.item fqdn_ip4 or salt -C 'I@backupninja:server' grains.item fqdn respectively.

3. In cluster/infra/backup/server.yml, verify that the following class is present:

   ```
   classes:
   - system.backupninja.server.single
   ```

4. Optionally, override the default configuration of the backup schedule as described in Configure a backup schedule for the Salt Master node.
5. Apply the salt.minion state:

   ```
   salt -C 'I@backupninja:server or I@backupninja:client' state.sls salt.minion
   ```

6. Refresh grains and mine for the backupninja client node:

   ```
   salt -C 'I@backupninja:client' state.sls salt.minion.grains
   salt -C 'I@backupninja:client' mine.flush
   salt -C 'I@backupninja:client' mine.update
   ```

©2025, Mirantis Inc.

7. Apply the backupninja state to the backupninja client node:

```
salt -C 'I@backupninja:client' state.sls backupninja
```

8. Refresh grains for the backupninja server node:

```
salt -C 'I@backupninja:server' state.sls salt.minion.grains
```

9. Apply the backupninja state to the backupninja server node:

```
salt -C 'I@backupninja:server' state.sls backupninja
```

Once you perform the procedure, two backup configuration scripts should be created. By default, the scripts are stored in the /etc/backup.d/ directory and will run daily at 1:00 AM.

Seealso

- Configure a backup schedule for the Salt Master node
- Enable a backup schedule for the Salt Master node using Backupninja
- Restore the Salt Master node

Configure a backup schedule for the Salt Master node

By default, Backupninja runs daily at 1.00 AM. This section describes how to override the default configuration of the backup schedule for the Salt Master node. To enable the backup schedule for the Salt Master node in your deployment, refer to Enable a backup schedule for the Salt Master node using Backupninja.

To configure a backup schedule for the Salt Master node:

1. Log in to the Salt Master node.
2. Edit the cluster/infra/config/init.yml file as required. Select from the following options:

    - Set the exact time of the backup server role to override the default backup time. The backup_times parameters include:

        - day_of_week

            The day of a week to perform backups. Specify 0 for Sunday, 1 for Monday, and so on. If not set, defaults to *.

        - day_of_month

            The day of a month to perform backups. For example, 20, 25, and so on. If not set, defaults to *.

            > Note
            >
            > Only day_of_week or day_of_month can be active at the same time. If both are defined, day_of_week is prioritized.

        - hour

            The hour to perform backups. Uses the 24-hour format. If not defined, defaults to 1.

        - minute

            The minute to perform backups. For example, 5, 10, 59, and so on. If not defined, defaults to 00.

      Configuration example:

      ```
      parameters:
        backupninja:
          enabled: true
          client:
            backup_times:
              day_of_week: 1
              hour: 15
              minute: 45
      ```

> Note
>
> These settings will change the global Backupninja schedule. If not set differently for individual steps, it will run all steps in the right order. This is recommended way of defining the exact backup order.

- Disable automatic backups:

```
parameters:
  backupninja:
    enabled: true
    client:
      auto_backup_disabled: true
```

- Re-enable automatic backups by setting the auto_backup_disabled parameter to false or delete the related line in cluster/infra/config/init.yml:

```
parameters:
  backupninja:
    enabled: true
    client:
      auto_backup_disabled: false
```

3. Apply changes by performing the steps 5-10 of the Enable a backup schedule for the Salt Master node using Backupninja procedure.

Create an instant backup of the Salt Master node

After you create a backup schedule as described in Enable a backup schedule for the Salt Master node using Backupninja, you may also need to create an instant backup of your Salt Master node.

To create an instant backup of the Salt Master node using Backupninja:

1. Verify that you have completed the steps described in Enable a backup schedule for the Salt Master node using Backupninja.
2. Log in to the Salt Master node.
3. Move local files to the backupninja server using rsync. For example:

   ```
   backupninja -n --run /etc/backup.d/200.backup.rsync
   ```

> Seealso
>
> - Restore the Salt Master node

Restore the Salt Master node

You may need to restore the Salt Master node after a hardware or software failure.

To restore the Salt Master node from a Backupninja rsync backup:

1. Redeploy the Salt Master node using the day01 image with the configuration ISO drive for the Salt Master VM as described in Deploy the Salt Master node.

   > **Caution!**
   >
   > Make sure to securely back up the configuration ISO drive image. This image contains critical information required to re-install your cfg01 node in case of storage failure, including master key for all encrypted secrets in the cluster metadata model.
   >
   > Failure to back up the configuration ISO image may result in loss of ability to manage MCP in certain hardware failure scenarios.

2. Log in to the Salt Master node.
3. Configure your deployment model by including the following pillar in cluster/infra/config/init.yml:

   ```
   parameters:
     salt:
       master:
         initial_data:
           engine: backupninja
           source: kvm03       # the backupninja server that stores Salt Master backups
           host: cfg01.<domain_name>  # for example: cfg01.deploy-name.local
       minion:
         initial_data:
           engine: backupninja
           source: kvm03       # the backupninja server that stores Salt Master backups
           host: cfg01.<domain_name>  # for example: cfg01.deploy-name.local
   ```

4. Verify that the pillar for Backupninja is present:

   ```
   salt-call pillar.data backupninja
   ```

   If the pillar is not present, configure it as described in Enable a backup schedule for the Salt Master node using Backupninja.

5. Verify that the pillar for master and minion is present:

   ```
   salt-call pillar.data salt:minion:initial_data
   salt-call pillar.data salt:master:initial_data
   ```

If the pillar is not present, verify the pillar configuration in cluster/infra/config/init.yml described above.

6. Apply the salt.master.restore and salt.minion.restore states.

   Mirantis recommends running the following command using Linux GNU Screen or alternatives.

   ```
   salt-call state.sls salt.master.restore,salt.minion.restore
   ```

   Running the states above restores the Salt Master node`s PKI and CA certificates and creates files as a flag in the /srv/salt/ directory that indicates the Salt Master node restore is completed.

   > Caution!
   >
   > If you rerun the state, it will not restore the Salt Master node again. To repeat the restore procedure, first delete the master-restored and minion-restored files from the /srv/salt directory and rerun the above states.

7. Verify that the Salt Master node is restored:

   ```
   salt-key
   salt -t2 '*' saltutil.refresh_pillar
   ls -la /etc/pki/ca/salt_master_ca/
   ```

Back up and restore the Salt Master node starting from 2019.2.5

This section describes how to back up and restore the Salt Master node starting from the MCP 2019.2.5 maintenance update.

Enable a backup schedule for the Salt Master node using Backupninja

This section describes how to create a backup schedule for the Salt Master node using the Backupninja utility. By default, Backupninja runs daily at 1:00 AM.

To create a backup schedule for the Salt Master node using Backupninja:

1. Log in to the Salt Master node.
2. In cluster/infra/config/init.yml, verify that the following classes and parameters are present:

```
classes:
- system.backupninja.client.single
- system.openssh.client.root
parameters:
  _param:
    backupninja_backup_host: <IP>
  salt:
    master:
      backup: true
    minion:
      backup: true
```

> Note
>
> The backupninja_backup_host parameter is the IP address or the host name of a node running the backupninja server. To obtain this IP address or host name, run salt -C 'I@backupninja:server' grains.item fqdn_ip4 or salt -C 'I@backupninja:server' grains.item fqdn respectively.

3. In cluster/infra/backup/server.yml, verify that the following class is present:

```
classes:
- system.backupninja.server.single
```

4. Optionally, override the default configuration of the backup schedule as described in Configure a backup schedule for the Salt Master node.
5. Apply the salt.minion state:

```
salt -C 'I@backupninja:server or I@backupninja:client' state.sls salt.minion
```

6. Refresh grains and mine for the backupninja client node:

```
salt -C 'I@backupninja:client' state.sls salt.minion.grains
salt -C 'I@backupninja:client' mine.update
```

7. Apply the backupninja state to the backupninja client node:

```
salt -C 'I@backupninja:client' state.sls backupninja
```

8. Refresh grains on the backupninja server node:

```
salt -C 'I@backupninja:server' state.sls salt.minion.grains
```

9. Apply the backupninja state to the backupninja server node:

```
salt -C 'I@backupninja:server' state.sls backupninja
```

Once you perform the procedure, two backup configuration scripts should be created.

Configure a backup schedule for the Salt Master node

> **Warning**
>
> This configuration presupposes manual backups or backups performed by a cron job. If you use the Backupninja backup pipeline job, see Configure the Backupninja backup pipeline.

By default, Backupninja runs daily at 1.00 a.m. This section describes how to override the default configuration of the backup schedule for the Salt Master node. To enable the backup schedule for the Salt Master node in your deployment, refer to Enable a backup schedule for the Salt Master node using Backupninja.

To configure a backup schedule for the Salt Master node:

1. Log in to the Salt Master node.
2. Edit the cluster/infra/config/init.yml file as required. Select from the following options:

   - Set the exact time of the backup server role to override the default backup time. The backup_times parameters include:

     - day_of_week

       The day of a week to perform backups. Specify 0 for Sunday, 1 for Monday, and so on. If not set, defaults to *.
     - day_of_month

       The day of a month to perform backups. For example, 20, 25, and so on. If not set, defaults to *.

       > **Note**
       >
       > Only day_of_week or day_of_month can be active at the same time. If both are defined, day_of_week is prioritized.
     - hour

       The hour to perform backups. Uses the 24-hour format. If not defined, defaults to 1.
     - minute

       The minute to perform backups. For example, 5, 10, 59, and so on. If not defined, defaults to 00.

     Configuration example:

```
parameters:
  backupninja:
    enabled: true
    client:
      backup_times:
        day_of_week: 1
        hour: 15
        minute: 45
```

> Note
>
> These settings will change the global Backupninja schedule. If not set differently for individual steps, it will run all steps in the right order. This is recommended way of defining the exact backup order.

- Disable automatic backups:

```
parameters:
  backupninja:
    enabled: true
    client:
      auto_backup_disabled: true
```

- Re-enable automatic backups by setting the auto_backup_disabled parameter to false or delete the related line in cluster/infra/config/init.yml:

```
parameters:
  backupninja:
    enabled: true
    client:
      auto_backup_disabled: false
```

3. Apply changes by performing the steps 5-10 of the Enable a backup schedule for the Salt Master node using Backupninja procedure.

Create an instant backup of the Salt Master node

This section instructs you on how to create an instant backup of the Salt Master node using Backupninja.

To create an instant backup of the Salt Master node using Backupninja:

1. Verify that you have completed the steps described in Enable a backup schedule for the Salt Master node using Backupninja.
2. Select from the following options:
   - Create an instant backup of the Salt Master node automatically as described in Create an instant backup using Backupninja pipeline.
   - Create an instant backup of the Salt Master node manually:
     1. Log in to the Salt Master node.
     2. Create a backup by moving local files to the backupninja server using the backupninja script which uses rsync. For example:

        ```
        backupninja -n --run /etc/backup.d/200.backup.rsync
        ```

Restore the Salt Master node

You may need to restore the Salt Master node after a hardware or software failure. This section instructs you on how to restore the Salt Master node using Backupninja.

To restore the Salt Master node using Backupninja:

Select from the following options:

- Restore the Salt Master node automatically as described in Restore the services using Backupninja pipeline.
- Restore the Salt Master node manually:

  1. Redeploy the Salt Master node using the day01 image with the configuration ISO drive for the Salt Master VM as described in Deploy the Salt Master node.

     > **Caution!**
     >
     > Make sure to securely back up the configuration ISO drive image. This image contains critical information required to re-install your cfg01 node in case of storage failure, including master key for all encrypted secrets in the cluster metadata model.
     >
     > Failure to back up the configuration ISO image may result in loss of ability to manage MCP in certain hardware failure scenarios.

  2. Log in to the Salt Master node.
  3. On the cluster level of the Reclass model, add the following pillar in the cluster/infra/config/init.yml file:

     ```
     parameters:
       salt:
         master:
           initial_data:
             engine: backupninja
             source: ${_param:backupninja_backup_host}  # the backupninja server that stores Salt Master backups, for example: kvm03
             host: ${_param:infra_config_hostname}.${_param:cluster_domain}  # for example: cfg01.deploy-name.local
             home_dir: '/path/to/backups/' # for example: '/srv/volumes/backup/backupninja'
         minion:
           initial_data:
             engine: backupninja
             source: ${_param:backupninja_backup_host}  # the backupninja server that stores Salt Master backups, for example: kvm03
             host: ${_param:infra_config_hostname}.${_param:cluster_domain}  # for example: cfg01.deploy-name.local
             home_dir: '/path/to/backups/' # for example: '/srv/volumes/backup/backupninja'
     ```

  4. Verify that the pillar for Backupninja is present:

     ```
     salt-call pillar.data backupninja
     ```

     If the pillar is not present, configure it as described in Enable a backup schedule for the Salt Master node using Backupninja.

5. Verify that the pillar for master and minion is present:

   ```
   salt-call pillar.data salt:minion:initial_data
   salt-call pillar.data salt:master:initial_data
   ```

   If the pillar is not present, verify the pillar configuration in cluster/infra/config/init.yml described above.

6. Apply the salt.master.restore and salt.minion.restore states.

   Mirantis recommends running the following command using Linux GNU Screen or alternatives.

   ```
   salt-call state.sls salt.master.restore,salt.minion.restore
   ```

   Running the states above restores the Salt Master node PKI and CA certificates and creates files as a flag in the /srv/salt/ directory that indicates the Salt Master node restore is completed.

   Caution!

   If you rerun the state, it will not restore the Salt Master node again. To repeat the restore procedure, first delete the master-restored and minion-restored files from the /srv/salt directory and rerun the above states.

7. Verify that the Salt Master node is restored:

   ```
   salt-key
   salt -t2 '*' saltutil.refresh_pillar
   ls -la /etc/pki/ca/salt_master_ca/
   ```

Back up and restore an OpenStack Control Plane

This section describes how to back up and restore an OpenStack Control Plane and Cinder volumes. For a high avalability (HA) OpenStack environment, you need to manually back up only Glance images, Cinder volumes, and databases.

Back up and restore a MySQL database

The Mirantis Cloud Platform (MCP) uses MySQL databases to store the data generated by different components of MCP. Mirantis recommends backing up your MySQL databases daily to ensure the integrity of your data. You should create an instant backup before upgrading your MySQL database. You may also need to create a MySQL database backup for testing purposes.

MCP uses the Xtrabackup utility to back up MySQL databases. Xtrabackup installs automatically with your cloud environment using a SaltStack formula and includes the following components:

- Xtrabackup server stores backups from rsynced Xtrabackup client nodes and runs on any node, for example, the Salt Master node.
- Xtrabackup client sends database backups to the Xtrabackup server and runs on the MySQL Galera Database Master node.

This section describes how to create and restore your MySQL databases using Xtrabackup.

Enable a backup schedule for a MySQL database

To ensure the consistent and timely backing up of your data, create a backup schedule using Xtrabackup.

To create a backup schedule for a MySQL database:

1. Log in to the Salt Master node.
2. Verify the configuration of the backup server nodes:

   ```
   salt -C 'I@xtrabackup:server' test.ping
   ```

   If the output of the command above is not empty, move to the next step. Otherwise, configure the xtrabackup server role by adding the following lines in cluster/infra/config/init.yml:

   > Note
   >
   > By default, Xtrabackup keeps three complete backups and their incrementals on the xtrabackup client node.

   ```
   classes:
   - system.xtrabackup.server.single
   parameters:
     _param:
        xtrabackup_public_key: <generate_your_keypair>
   ```

3. Sync the pillar data:

   ```
   salt '*' saltutil.sync_all
   ```

4. Verify the configuration of the backup client nodes:

   ```
   salt -C 'I@xtrabackup:client' test.ping
   ```

   If the output is not empty, move to the next step. Otherwise, configure the xtrabackup client role by adding the following lines in cluster/openstack/database/init.yml:

   ```
   classes:
   - system.xtrabackup.client.single
   parameters:
     _param:
       xtrabackup_remote_server: cfg01
       root_private_key: |
         <generate_your_keypair>
   ```

5. Verify that the xtrabackup_remote_server parameter is defined correctly:

   ```
   salt -C 'I@xtrabackup:client' pillar.get _param:xtrabackup_remote_server
   ```

   If the system response does not contain an IP address, add the following parameter to cluster/openstack/database/init.yml:

   ```
   parameters:
     _param:
       xtrabackup_remote_server: <host_name>
   ```

   Substitute <host_name> with the resolvable host name of the host on which the Xtrabackup server is running. For example, kvm03, which is the default value in MCP.

6. Optionally, override the default Xtrabackup configuration as described in Configure a backup schedule for a MySQL database.
7. Run the following command on the Salt Master node:

   ```
   salt '*' saltutil.refresh_pillar
   ```

8. Apply the salt.minion state:

   ```
   salt -C 'I@xtrabackup:client or I@xtrabackup:server' state.sls salt.minion
   ```

9. Refresh grains for the xtrabackup client node:

   ```
   salt -C 'I@xtrabackup:client' saltutil.sync_grains
   ```

10. Update the mine for the xtrabackup client node:

    ```
    salt -C 'I@xtrabackup:client' mine.flush
    salt -C 'I@xtrabackup:client' mine.update
    ```

11. Apply the xtrabackup client state:

    ```
    salt -C 'I@xtrabackup:client' state.sls openssh.client,xtrabackup
    ```

12. Apply the linux.system.cron state:

    ```
    salt -C 'I@xtrabackup:server' state.sls linux.system.cron
    ```

13. Apply the xtrabackup server state:

    ```
    salt -C 'I@xtrabackup:server' state.sls xtrabackup
    ```

Seealso

- Configure a backup schedule for a MySQL database
- Create an instant backup of a MySQL database
- Restore a MySQL database
- Restore a Galera cluster

Configure a backup schedule for a MySQL database

This section instructs you on how to configure a backup schedule for a MySQL database using the Xtrabackup service.

MCP provides the following options for the MySQL backup schedule configuration:

- Recommended. Back up the MySQL database using the Jenkins pipeline job to run the Xtrabackup script on a predefined schedule.
- Back up the MySQL database by running the Xtrabackup script using the predefined crontab record.

Depending on your preferences, proceed with one of the following sections:

Configure a backup schedule in the Jenkins pipeline

This section describes how to configure the MySQL backup schedule in the Galera database backup Jenkins pipeline.

To configure the backup schedule for a MySQL database:

1. Verify that you have enabled the schedule as described in Enable a backup schedule for a MySQL database.
2. Verify that cron for the innobackupex-runner script is disabled:
    1. Log in to the Salt Master node.
    2. Verify that the value of the cron parameter is defined in the pillar data. The parameter should be set to false.

        ```
        salt -C 'I@xtrabackup:client' pillar.data xtrabackup:client:cron
        salt -C 'I@xtrabackup:server' pillar.data xtrabackup:server:cron
        ```

        If the parameter is not defined or set to true, manually change the value to false in the infra/backup/client_mysql.yml on the cluster Reclass level.
    3. If the schedule was configured before, remove the old schedule configuration by removing the backup_times pillar block from the cluster/infra/config/init.yml and cluster/openstack/database/init.yml files if it is present in parameters:xtrabackup:client.
    4. Apply the xtrabackup state:

        ```
        salt -C 'I@xtrabackup:client or I@xtrabackup:server' state.sls xtrabackup
        ```

    5. On the dbs01 node, verify that crontab is working and the Xtrabackup job is disabled:

        ```
        crontab -l
        # Lines below here are managed by Salt, do not edit
        # SALT_CRON_IDENTIFIER:/usr/local/bin/innobackupex-runner.sh
        # DISABLED 0 */12 * * * /usr/local/bin/innobackupex-runner.sh
        ```

    6. Verify the system response of the following command:

        ```
        salt -C I@xtrabackup:client pillar.get _param
        ```

        The system response should include the following parameters for the Jenkins trigger but the values may vary:

        ```
        parameters:
          _param:
            backup_min: "0"
            backup_hour: "*/12"
        ```

```
    backup_day_of_month: "*"
    backup_month: "*"
    backup_day_of_week: "*"
```

If the above parameters are not defined, set them up in infra/init.yml on the cluster Reclass level.

3. Configure the Galera database backup pipeline as required:

    1. Open the cluster Reclass level of your deployment.
    2. Move the include of the infra.backup.client_mysql level from openstack/database/master.yml to openstack/database/init.yml.
    3. Refresh the grains to be able to get correct IP addresses for /.ssh/authorized_keys.

    ```
    salt -C I@xtrabackup:client state.sls salt
    salt -C I@xtrabackup:client mine.update
    salt -C I@xtrabackup:client saltutil.sync_all
    ```

    4. Apply the xtrabackup state to configure access for new nodes:

    ```
    salt -C I@xtrabackup:server state.sls xtrabackup
    ```

    5. In cicd/control/leader.yml, add the following class:

    ```
    classes:
    - system.jenkins.client.job.deploy.galera_database_backup
    ```

    6. Re-apply the jenkins state on the cid01 node:

    ```
    salt -C I@jenkins:client state.sls jenkins
    ```

    7. Optional. To define a custom backup time, override the backup paramateters in the infra/init.yml file:

    ```
    parameters:
      _param:
        backup_min: "0"
        backup_hour: "*/12"
        backup_day_of_month: "*"
        backup_month: "*"
        backup_day_of_week: "*"
    ```

Backup parameters details

Parameter name	Description	Possible values
backup_min	Value in minutes when the backup should run	0-59, *
backup_hour	Value in hours when the backup should run	0-23, *
backup_day_of_month	Value in days of month when the backup should run	1-31, *
backup_month	Value in months when the backup should run	0-23, *
backup_day_of_week	Value in days of week when the backup should run	0-6, *

All parameters can also use / to mark the iteration. For example, the */12 value for the backup_hour parameter means that the backup will start every twelfth hour. See Jenkins cron syntax in the official Jenkins documentation for the details.

Configure a backup schedule using the Xtrabackup script directly

This section describes how to configure the MySQL backup schedule using the Xtrabackup script based on the crontab record.

> Note
>
> We recommend that you Configure a backup schedule in the Jenkins pipeline instead of using the Xtrabackup script directly.

By default, Xtrabackup stores three complete backups with their incremental backups. This section describes how to override the default configuration of the backup schedule for a MySQL database. To enable the backup schedule in your deployment, refer to Enable a backup schedule for a MySQL database.

To configure a backup schedule for a MySQL database:

1. Log in to the Salt Master node.
2. Select from the following options:

   - Override the default Xtrabackup configuration by setting a custom time interval in cluster/infra/config/init.yml.

     1. Set the following parameters as required:

        hours_before_full
        : Sets the full backup frequency. If set to 48, the full backup is performed every two days. Within 48 hours, only incremental backups are performed.

        full_backups_to_keep
        : Sets the number of full backups to keep.

        incr_before_full
        : Sets the number of incremental backups to be performed between full backups. If set to 3, three incremental backups will be performed between full backups.

        cron
        : If set to false, disables the automatic backup by removing the cron job that triggers an automatic backup. Set cron: true to enable the automatic backup.

        Configuration example:

        ```yaml
        parameters:
          xtrabackup:
            server:
              enabled: true
              hours_before_full: 48
              full_backups_to_keep: 5
        ```

```
incr_before_full: 3
cron: false
```

2. Verify that the hours_before_full parameter of the xtrabackup client in cluster/openstack/database/init.yml matches the same parameter of the xtrabackup server in cluster/infra/config/init.yml.

- Set the exact backup time for the Xtrabackup server role in cluster/infra/config/init.yml and the Xtrabackup client role in cluster/openstack/database/init.yml by configuring the backup_times section.

  The backup_times parameters include:

  day_of_week
  : The day of a week to perform backups. Specify 0 for Sunday, 1 for Monday, and so on. If not set, defaults to *.

  day_of_month
  : The day of a month to perform backups. For example, 20, 25, and so on. If not set, defaults to *.

  > Note
  >
  > Only day_of_week or day_of_month can be active at the same time. If both are defined, day_of_week is prioritized.

  month
  : The month to perform backups. Available values include 1 for January, 2 for February, and so on up to 12 for December.

  hour
  : The hour to perform backups. Uses the 24-hour format. If not defined, defaults to 1.

  minute
  : The minute to perform backups. For example, 5, 10, 59, and so on. If not defined, defaults to 00.

  > Note
  >
  > If any of the individual backup_times parameters is not defined, the default * value will be used. For example, if the minute parameter is *, the backup will run every minute, which is usually not desired.

> **Caution!**
>
> Only backup_times section or hours_before_full(incr) can be active. If both are defined, the backup_times section will be prioritized.

Configuration example for the Xtrabackup server role:

```
parameters:
  xtrabackup:
    server:
      enabled: true
      full_backups_to_keep: 3
      incr_before_full: 3
      backup_dir: /srv/backup
      backup_times:
        day_of_week: 0
        hour: 4
        minute: 52
      key:
        xtrabackup_pub_key:
          enabled: true
          key: key
```

Configuration example for the Xtrabackup client role:

```
parameters:
  xtrabackup:
    client:
      enabled: true
      full_backups_to_keep: 3
      incr_before_full: 3
      backup_times:
        day_of_week: 0
        hour: 4
        minute: 52
      compression: true
      compression_threads: 2
      database:
        user: username
        password: password
      target:
      host: cfg01
      cron: true
```

The cron parameter, if set to false, disables the script that automatically triggers the backup scripts. For the correct backup strategy, set up cron to true or use the backup pipeline as described in Configure a backup schedule in the Jenkins pipeline.

3. Apply changes by performing the steps 5-10 of the Enable a backup schedule for a MySQL database procedure.

Seealso

Restore a Galera cluster

Create an instant backup of a MySQL database

This section instructs you on how to create an instant backup of a MySQL database using the Xtrabackup service.

To create an instant backup for a MySQL database using Xtrabackup:

1. Verify that you have completed the steps described in Enable a backup schedule for a MySQL database and optionally in Configure a backup schedule for a MySQL database.
2. Create an instant backup either using Jenkins or manually.

Create an instant backup of a MySQL database automatically

After you create a backup schedule as described in Enable a backup schedule for a MySQL database, you may also need to create an instant backup of a MySQL database.

To create an instant backup automatically:

1. Log in to the Jenkins web UI.
2. Open the Galera database backup pipeline.

    1. Specify the required parameters:

        Galera database backup parameters

Parameter	Description and values
SALT_MASTER_URL	Define the IP address of your Salt Master node host and the salt-api port. For example, http://172.18.170.27:6969.
SALT_MASTER_CREDENTIALS	Define credentials_id as credentials for the connection.
OVERRIDE_BACKUP_NODE	Fill in the name of the node to back up if you want to override the automatic node selection. The default value is none that triggers the automatic node selection workflow.

    2. Click Deploy.

        The pipeline workflow:

        1. Primary component location and node selection. The pipeline locates the current primary component and selects one of its nodes to use as a source of data for the backup.

            Note

            The pipeline skips this stage if the OVERRIDE_BACKUP_NODE parameter is defined with a preferred node name.

        2. Backup preparation. Several necessary steps are run to verify that the openssh and xtrabackup states are up to date.
        3. Backup.
        4. Cleanup. The cleanup script is triggered to clean the temporary directories and old backups.
3. Verify that the backup has been created and, in case of the remote backup storage, moved correctly.

Create an instant backup of a MySQL database manually

After you create a backup schedule as described in Enable a backup schedule for a MySQL database, you may also need to create an instant backup of a MySQL database.

To create an instant backup manually:

1. Log in to the MySQL database master node. For example, dbs01.
2. Run the following script:

   ```
   /usr/local/bin/innobackupex-runner.sh
   ```

3. Verify that a complete backup has been created on the MySQL database master node:

   ```
   ls /var/backups/mysql/xtrabackup/full
   ```

   If you rerun /usr/local/bin/innobackupex-runner.sh, it creates an incremental backup for the previous complete backup in /var/backups/mysql/xtrabackup/incr.

   You can pass the following flags with the innobackupex-runner.sh script:

   - -s makes the script to skip the cleanup. It can be useful if you want to trigger a manual backup keeping all the previous backups. Be aware that once you run the script without the -s flag or if an automatic backup is triggered, the backups will be cleaned and only the defined number of the latest backups will be kept.
   - -f forces the script to run the full backup instead of an incremental one.
   - -c forces the script to run only the cleanup.
4. Verify that the complete backup has been rsynced to the xtrabackup server node from the Salt Master node:

   ```
   salt -C 'I@xtrabackup:server' cmd.run 'ls /var/backups/mysql/xtrabackup/full'
   ```

   > Note
   >
   > If you run /usr/local/bin/innobackupex-runner.sh more than once, at least one incremental backup is created in /var/backups/mysql/xtrabackup/incr on the node.

> Seealso
>
> Restore a Galera cluster

Restore a MySQL database

You may need to restore a MySQL database after a hardware or software failure or if you want to create a clone of the existing database from a backup.

To restore a MySQL database using Xtrabackup:

1. Log in to the Salt Master node.
2. In the cluster/infra/backup/client_mysql.yml file, add the following configuration for the Xtrabackup client. If the file does not exist, edit cluster/openstack/database/init.yml.

   ```
   parameters:
     xtrabackup:
       client:
         enabled: true
         restore_full_latest: 1
         restore_from: remote
   ```

   where:

   - restore_full_latest can have the following values: 1 or 2. 1 means restoring the database from the last complete backup and its increments. 2 means restoring the second latest complete backup and its increments.
   - restore_from can have the following values: local or remote. The remote value uses scp to get the files from the xtrabackup server.
3. Proceed with either automatic restore steps using the Jenkins web UI pipeline or with manual restore steps:

   - Automatic restore steps:

     1. Add the upgrade pipeline to DriveTrain:

        1. Verify that the following lines are present in cluster/cicd/control/leader.yml:

           ```
           classes:
           - system.jenkins.client.job.deploy.galera_verify_restore
           ```

        2. Run the salt -C 'I@jenkins:client' state.sls jenkins.client state.
     2. Log in to the Jenkins web UI.
     3. Open the Verify and Restore Galera pipeline.
     4. Specify the following parameters:

Parameter	Description and values
RESTORE_TYPE	Set job execution type. Select ONLY_RESTORE.

SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.

5. Click Deploy.
6. Open the Console Output of the build from the navigation menu to control the deployment progress and manually confirm actions when prompted during the job execution.
7. Once the Verify and Restore Galera pipeline finishes successfully, revert the changes made in cluster/openstack/database/init.yml in the step 2.

- Manual restore steps:

1. Stop the mysql service on the MySQL Galera Database dbs02 and dbs03 nodes:

```
salt -C 'I@galera:slave' service.stop mysql
```

2. Remove the MySQL log files from the MySQL Galera Database dbs02 and dbs03 nodes:

```
salt -C 'I@galera:slave' cmd.run 'rm /var/lib/mysql/ib_logfile*'
```

3. Stop the mysql service on the MySQL Galera Database Master node:

```
salt -C 'I@galera:master' service.stop mysql
```

4. Log in to the MySQL Galera Database Master node.
5. Replace the wsrep_cluster_address row in /etc/mysql/my.cnf with the following:

```
wsrep_cluster_address="gcomm://"
```

6. Log in to the Salt Master node.
7. Move the MySQL database files to a new location /root/mysql/mysql.bak/ on the MySQL Galera Database Master node:

```
salt -C 'I@galera:master' cmd.run 'mkdir -p /root/mysql/mysql.bak/'
salt -C 'I@galera:master' cmd.run 'mv /var/lib/mysql/* /root/mysql/mysql.bak'
salt -C 'I@galera:master' cmd.run 'rm /etc/salt/.galera_bootstrap'
```

8. Verify that the MySQL database files are removed from /var/lib/mysql/ on the MySQL Galera Database Master node:

```
salt -C 'I@galera:master' cmd.run 'ls /var/lib/mysql/'
salt -C 'I@galera:master' cmd.run 'ls -ld /var/lib/mysql/.?*'
```

9. Log in to the MySQL Galera Database Master node where the restore operation occurs.
10. Run the following state that restores the databases and creates a file in /var/backups/mysql/xtrabackup/dbrestored.

    Mirantis recommends running the following command using Linux GNU Screen or alternatives.

    ```
    salt-call state.sls xtrabackup.client.restore
    ```

    If you rerun the state, it will not restore the database again. To repeat the restore procedure, first delete the /var/backups/mysql/xtrabackup/dbrestored file and then rerun the above xtrabackup state again.
11. Log in to the Salt Master node.
12. Verify that the MySQL database files are present again on the MySQL Galera Database Master node:

    ```
    salt -C 'I@galera:master' cmd.run 'ls /var/lib/mysql/'
    ```
13. Start the mysql service on the MySQL Galera Database Master node:

    ```
    salt -C 'I@galera:master' service.start mysql
    ```

    > Note
    >
    > This process takes a certain amount of time and does not provide an immediate output.
14. Start the mysql service on the MySQL Galera Database dbs02 and dbs03 nodes from the Salt Master node:

    ```
    salt -C 'I@galera:slave' service.start mysql
    ```

> Note
>
> This process takes a certain amount of time and does not provide an immediate output.

15. Verify that all MySQL Galera Database nodes joined the Galera cluster:

```
salt -C 'I@galera:master' mysql.status | grep -A1 wsrep_cluster_size
```

16. Revert the changes made in cluster/openstack/database/init.yml in the step 2 and in /etc/mysql/my.cnf in the step 5.

> Seealso
>
> Restore a Galera cluster

Back up and restore Glance images

This section instructs you on how to back up and restore Glance images and covers both the local storage and the Ceph back end for Glance configurations. Both procedures presuppose that the Glance API is in the working state.

To back up Glance images:

1. Log in to the Salt Master node.
2. Verify that there is enough space on the OpenStack controller node to which you will copy the Glance images.
3. Copy the images to the backup destination preserving the context. For example, to copy the images to the ctl01 node:

   ```
   salt -C 'I@glance:server and *01*' cmd.run '. /root/keystonercv3; cd <PATH_TO_BACKUP>; for i in `openstack image list -c ID -f value`; \
   do openstack image save --file $i $i; done'
   ```

4. If needed, move the backed up images to the external backup location.

To restore Glance images:

1. Log in to the Salt Master node.
2. If needed, copy the images from the external backup location to an OpenStack controller node.
3. Copy the images from the OpenStack controller node to the Glance folder with the images. In the example commands below, the ctl01 node is used as the OpenStack controller node containing the backed up Glance images:

   > Note
   >
   > GlusterFS will automatically replicate the restored images to all other OpenStack controller nodes.

   - If the local storage is used, run:

     ```
     salt -C 'I@glance:server and *01*' cmd.run "cp -a <PATH_TO_BACKUP>/. /var/lib/glance/images/"
     ```

   - If the Ceph back end is used, run:

     ```
     salt -C 'I@glance:server and *01*' cmd.run ". /root/keystonercv3; cd <PATH_TO_BACKUP>; for i in `openstack image list -c ID -f value`; \
     do rbd import -p images $i; rbd snap create -p images $i@snap; done"
     ```

4. Verify that the restored Glance images are available:

   ```
   salt -C 'I@keystone:client' cmd.run ". /root/keystonercv3; openstack image list"
   ```

Note

If the context of the Glance images files is lost, run:

```
salt -C 'I@glance:server' cmd.run "chown glance:glance <IMAGE_FILE_NAME>"
salt -C 'I@glance:server' cmd.run "chmod 640 <IMAGE_FILE_NAME>"
```

Back up and restore Cinder volumes and snapshots

This section describes how to back up and restore Cinder volumes and snapshots in an OpenStack cluster with Ceph. Starting from the MCP 2019.2.4 maintenance update, the Ceph backup engine parameters are enabled by default. For MCP versions earlier than 2019.2.4, enable these parameters manually.

©2025, Mirantis Inc.

Manually enable the Ceph backup engine for Cinder

If your MCP version is earlier than 2019.2.4, before you back up or restore the Cinder volumes and snapshots, manually enable the Ceph backup engine parameters as described below. If your MCP version is 2019.2.4 or later, these parameters are enabled by default and you can proceed to Create a backup or restore Cinder volumes and snapshots right away.

To manually enable the Ceph backup engine for Cinder:

1. Log in to the Salt Master node.
2. Open your Git project repository with the Reclass model on the classes/cluster/<cluster_name>/ level.
3. In /ceph/setup.yml, add a new pool called backups. For the pg_num and pgp_num parameters, copy the values from any existing Ceph pools, since these values are created from one source for all pools.

   For example:

   ```
   parameters:
     ceph:
       setup:
         pool:
           backups:
             pg_num: 8
             pgp_num: 8
             type: replicated
             application: rbd
   ```

4. In /ceph/common.yml, add the profile rbd pool=backups OSD permission for the cinder user to the existing Ceph keyring OSD permissions for Cinder:

   ```
   parameters:
     ceph:
       common:
         keyring:
           cinder:
             caps:
               osd: "profile rbd pool=volumes, profile rbd-read-only pool=images, profile rbd pool=backups"
   ```

5. In /openstack/init.yml, add the following definitions:

   ```
   parameters:
     _param:
       cinder_ceph_backup_pool: backups
       cinder_ceph_stripe_count: 0
       cinder_ceph_stripe_unit: 0
       cinder_ceph_backup_user: cinder
       cinder_ceph_chunk_size: 134217728
   ```

6. In /openstack/control.yml, add the following classes:

```
classes:
- system.cinder.control.backup.ceph
- system.cinder.volume.backup.ceph
```

7. Update pillars and create a new pool:

```
sudo salt '*' saltutil.refresh_pillar
sudo salt 'cmn01*' state.apply ceph.setup
```

8. Log in to the cmn01 node as a root user.
9. Update the keyring for the cinder user. In the command below, substitute the --cap osd value with the one added to /ceph/common.yml in the step 4. For example:

```
ceph-authtool /etc/ceph/ceph.client.cinder.keyring -n client.cinder \
 --cap osd 'profile rbd pool=volumes, profile rbd-read-only pool=images, profile rbd pool=backups' \
 --cap mon 'allow r, allow command \"osd blacklist\"'
```

10. Apply the changes for Ceph:

```
ceph auth import -i /etc/ceph/ceph.client.cinder.keyring
```

11. Install the cinder-backup package and adjust the Cinder configuration:

```
sudo salt 'ctl*' state.apply cinder
```

Now, you can proceed to Create a backup or restore Cinder volumes and snapshots.

Create a backup or restore Cinder volumes and snapshots

This section describes how to back up and restore Cinder volumes and snapshots in an OpenStack cluster. If your MCP version is earlier than 2019.2.4, before proceeding with the steps below, Manually enable the Ceph backup engine for Cinder.

> **Note**
>
> MCP sets the cinder-backup service to work with Ceph as a backup driver. The configuration parameters of the Cinder backup service are defined in the Salt formula. For details and configuration examples, see the SaltStack Cinder formula.

To back up a Cinder volume:

Create a full backup of a volume by running the following command on the Salt Master node:

```
salt -C 'I@keystone:client' cmd.run ". /root/keystonercv3; \
      openstack volume backup create --force <VOLUME_ID>"
```

> **Note**
>
> After a full backup is created, use the --incremental flag for further backups.

To back up a Cinder snapshot, run:

```
salt -C 'I@keystone:client' cmd.run ". /root/keystonercv3; \
      openstack volume backup create --force --snapshot <SNAPSHOT_ID> \
                                              <VOLUME_ID>"
```

> **Note**
>
> For further backups, use the --incremental flag.

To restore Cinder volumes or snapshots, run:

```
salt -C 'I@keystone:client' cmd.run ". /root/keystonercv3; \
      openstack volume backup restore <BACKUP_ID> <VOLUME_ID>"
```

Seealso

Back up and restore volumes and snapshots

Back up and restore OpenContrail

This section describes how to back up and restore Cassandra and ZooKeeper databases for OpenContrail 3.2 and 4.x.

Back up and restore a Cassandra database

MCP uses a custom script to back up Cassandra databases. Cassandra is part of every OpenContrail deployment and the backup utility includes the following components:

- Cassandra server stores Cassandra backups rsynced from the Cassandra client nodes and runs on any node, for example, the Salt Master node.
- Cassandra client sends database backups to the Cassandra server and runs on one node of the Cassandra cluster.

This section describes how to create and restore Cassandra databases for OpenContrail 3.2 and 4.x.

OpenContrail 3.2: Create a backup schedule for a Cassandra database

This section describes how to create a backup schedule for a Cassandra database for your OpenContrail 3.2 cluster.

To create a backup schedule for a Cassandra database:

1. Log in to the Salt Master node.
2. Configure the cassandra server role:

    1. Add the following class to cluster/infra/config.yml:

        ```
        classes:
        - system.cassandra.backup.server.single
        parameters:
          _param:
             cassandra_backup_public_key: <generate_your_keypair>
        ```

        By default, adding this include statement results in the Cassandra backup server keeping five full backups. To change the default setting, include the following pillar to cluster/infra/config.yml.

        ```
        parameters:
          cassandra:
            backup:
              cron: True
              server:
                enabled: true
                hours_before_full: 24
                full_backups_to_keep: 5
        ```

    2. Add the following lines to cluster/infra/config.yml:

        ```
        reclass:
          storage:
            node:
              opencontrail_control_node01:
                classes:
                - cluster.${_param:cluster_name}.opencontrail.control_init
        ```

3. Configure the cassandra client role by adding the following lines to cluster/opencontrail/control_init.yml and specifying the SSH key pair. Create this file, if not present.

    ```
    classes:
    - system.cassandra.backup.client.single
    parameters:
      _param:
    ```

```
  cassandra_remote_backup_server: cfg01
  root_private_key: |
    <generate_your_keypair>
```

By default, adding this include statement results in Cassandra keeping three complete backups on the cassandra client node. The rsync command moves the backup files to the Salt Master node. To change the default setting, include the following pillar to cluster/opencontrail/control_init.yml:

```
parameters:
  cassandra:
    backup:
      cron: True
      client:
        enabled: true
        cleanup_snaphots: true
        full_backups_to_keep: 3
        hours_before_full: 24
        target:
          host: cfg01
```

> Note
>
> - The target.host parameter must contain the resolvable hostname of the host where the cassandra server is running.
> - Mirantis recommends setting true for the cleanup_snaphots parameter so that the backup script removes all previous database snapshots once the current database backup is done.

4. If you customized the default parameters, verify that the hours_before_full parameter of the cassandra client in cluster/opencontrail/control_init.yml matches the same parameter of the cassandra server in cluster/infra/config.yml.
5. Run the following command on the Salt Master node:

```
salt '*' saltutil.refresh_pillar
```

6. Apply the salt.minion state:

```
salt -C 'I@cassandra:backup:client or I@cassandra:backup:server' state.sls salt.minion
```

7. Refresh grains for the cassandra client node:

```
salt -C 'I@cassandra:backup:client' saltutil.sync_grains
```

8. Update the mine for the cassandra client node:

```
salt -C 'I@cassandra:backup:client' mine.flush
salt -C 'I@cassandra:backup:client' mine.update
```

9. Apply the following state on the cassandra client nodes:

```
salt -C 'I@cassandra:backup:client' state.sls openssh.client,cassandra.backup
```

10. Apply the following state on the cassandra server nodes:

```
salt -C 'I@cassandra:backup:server' state.sls cassandra
```

Seealso

- OpenContrail 3.2: Create an instant backup of a Cassandra database
- OpenContrail 3.2: Restore the Cassandra database

OpenContrail 3.2: Create an instant backup of a Cassandra database

After you create a backup schedule as described in OpenContrail 3.2: Create a backup schedule for a Cassandra database, you may also need to create an instant backup of a Cassandra database on your OpenContrail 3.2 cluster.

To create an instant backup of a Cassandra database:

1. Verify that you have completed the steps described in OpenContrail 3.2: Create a backup schedule for a Cassandra database.
2. Log in to the Salt Master node.
3. Run the following state:

   ```
   salt-call state.sls reclass
   ```

4. Log in to the OpenContrail control node that holds the Cassandra backup client role, for example, ntw01.
5. Run the following script:

   ```
   /usr/local/bin/cassandra-backup-runner-call.sh
   ```

   > Note
   >
   > The output for some keyspaces can return an Error statement if it does not contain any .db file. If it is not for all keyspaces, then such behaviour is normal.

6. Verify that a complete backup has been created on the Cassandra backup client node:

   ```
   ls /var/backups/cassandra/full
   ```

7. Log in to the Cassandra backup server node.
8. Verify that the complete backup was rsynced to this node:

   ```
   ls /var/backups/cassandra/full'
   ```

> Seealso
>
> OpenContrail 3.2: Restore the Cassandra database

OpenContrail 3.2: Restore the Cassandra database

You may need to restore the Cassandra database after a hardware or software failure.

To restore the Cassandra database:

1. Log in to the Salt Master node.
2. Add the following lines to cluster/opencontrail/control_init.yml:

   ```
   cassandra:
     backup:
       client:
         enabled: true
         restore_latest: 1
         restore_from: remote
   ```

   where:

   - restore_latest can have, for example, the following values:
     - 1, which means restoring the database from the last complete backup.
     - 2, which means restoring the database from the second latest complete backup.
   - restore_from can have the local or remote values. The remote value uses scp to get the files from the cassandra server.
3. Restore the Cassandra database using the Jenkins web UI:
   1. Add the upgrade pipeline to DriveTrain:
      1. Add the following lines to cluster/cicd/control/leader.yml:

         ```
         classes:
         - system.jenkins.client.job.deploy.update.restore_cassandra
         ```

      2. Run the salt -C 'I@jenkins:client' state.sls jenkins.client state.
   2. Log in to the Jenkins web UI.
   3. Open the cassandra - restore pipeline.
   4. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.

   5. Click Deploy.

OpenContrail 4.x: Create a backup schedule for a Cassandra database

This section describes how to create a backup schedule for a Cassandra database for your OpenContrail 4.x cluster.

To create a backup schedule for a Cassandra database:

1. Log in to the Salt Master node.
2. Configure the cassandra server role.

   By default, the Cassandra backup server keeps five full backups. You can change the default settings by specifying the following parameters in cluster/<cluster_name>/infra/backup/server.yml:

   ```
   parameters:
     _param:
     ...
     cassandra:
       backup:
         cron: true
         backup_dir: /srv/volumes/backup/cassandra
         server:
           enabled: true
           hours_before_full: 24
           full_backups_to_keep: 5
           key:
             cassandra_pub_key:
               enabled: true
               key: ${_param:cassandra_backup_public_key}
   ```

3. Enable the scheduler backup process and configure the cassandra client role:

   1. Add the following parameters to cluster/<cluster_name>/opencontrail/control_init.yml:

      ```
      parameters:
        _param:
        ...
        cassandra:
          backup:
            cron: true
      ```

      By default, the Cassandra backup procedure keeps three complete backups on the cassandra client node. The rsync command moves the backup files to the Cassandra backup server.

   2. Optional. Change the default settings by specifying the following parameters in cluster/<cluster_name>/opencontrail/control_init.yml:

```
parameters:
  _param:
  ...
  cassandra:
    backup:
      cron: true
      client:
        enabled: true
        cleanup_snaphots: true
        full_backups_to_keep: 5
        hours_before_full: 24
        ...
```

> Note
>
> Mirantis recommends setting true for the cleanup_snaphots parameter so that the backup script removes all previous database snapshots once the current database backup is done.

3. Verify or add the specified host as the Cassandra backup server in cluster/<cluster_name>/infra/backup/client_cassandra.yml:

```
parameters:
  _param:
    cassandra_remote_backup_server: ${_param:infra_kvm_node03_address}
```

   The cassandra_remote_backup_server parameter must contain the resolvable hostname of the host on which the cassandra server is running.

4. Verify or add other options for the Cassandra client node in cluster/<cluster_name>/infra/backup/client_cassandra.yml:

```
parameters:
  ...
  cassandra:
    backup:
      cron: true
      client:
        containers:
        - opencontrail_controller_1
        enabled: true
        full_backups_to_keep: 5
        hours_before_full: 24
```

```
    target:
      host: cfg01
      backup_dir: /srv/volumes/backup/cassandra/${linux:system:name}
    backup_times:
      hour: '2'
      minute: '0'
```

4. If you customized the default parameters, verify that the hours_before_full parameter of the cassandra client in cluster/<cluster_name>/opencontrail/control_init.yml matches the same parameter of the cassandra server in cluster/<cluster_name>/infra/backup/server.yml.
5. Refresh pillars on the target nodes:

```
salt -C 'I@cassandra:backup' saltutil.refresh_pillar
```

6. Apply the salt.minion state:

```
salt -C 'I@cassandra:backup:client or I@cassandra:backup:server' state.sls salt.minion
```

7. Refresh grains for the cassandra client node:

```
salt -C 'I@cassandra:backup:client' saltutil.sync_grains
```

8. Update the mine for the cassandra client node:

```
salt -C 'I@cassandra:backup:client' mine.flush
salt -C 'I@cassandra:backup:client' mine.update
```

9. Apply the following state to add a Cassandra user:

```
salt -C 'I@cassandra:backup:server' state.apply linux.system.user
```

10. Apply the following state on the cassandra client nodes:

```
salt -C 'I@cassandra:backup:client' state.apply openssh.client.private_key,cassandra.backup
```

11. Apply the following state on the cassandra server nodes:

```
salt -C 'I@cassandra:backup:server' state.sls cassandra
```

Seealso

- OpenContrail 4.x: Create an instant backup of a Cassandra database
- OpenContrail 4.x: Restore the Cassandra database

OpenContrail 4.x: Create an instant backup of a Cassandra database

After you create a backup schedule as described in OpenContrail 4.x: Create a backup schedule for a Cassandra database, you may also need to create an instant backup of a Cassandra database on your OpenContrail 4.x cluster.

To create an instant backup of a Cassandra database:

1. Verify that you have completed the steps described in OpenContrail 4.x: Create a backup schedule for a Cassandra database.
2. Log in to the Salt Master node.
3. Run the following state:

   ```
   salt-call state.sls reclass
   ```

4. Log in to the OpenContrail control node that holds the Cassandra backup client role, for example, ntw01.
5. Run the following script:

   ```
   /usr/local/bin/cassandra-backup-runner-call.sh
   ```

   > Note
   >
   > The output for some keyspaces can return an Error statement if it does not contain any .db file. If it is not for all keyspaces, then such behaviour is normal.

6. Verify that a complete backup has been created on the Cassandra backup client node:

   ```
   ls /var/backups/cassandra/full
   ```

7. Log in to the Cassandra backup server node.
8. Verify that the complete backup was rsynced to this node:

   ```
   ls /srv/volumes/backup/cassandra/full
   ```

> Seealso
>
> OpenContrail 4.x: Restore the Cassandra database

OpenContrail 4.x: Restore the Cassandra database

You may need to restore the Cassandra database after a hardware or software failure.

> **Warning**
>
> During the restore procedure, all current Cassandra data is deleted. Therefore, starting from the MCP 2019.2.5 maintenance update, a database backup in Cassandra is not created before the restore procedure. If a backup of current data is required, you can create an instant backup. For details, see: OpenContrail 4.x: Create an instant backup of a Cassandra database.

To restore the Cassandra database:

1. Log in to the Salt Master node.
2. Open your project Git repository with the Reclass model on the cluster level.
3. Add the following snippet to cluster/<cluster_name>/infra/backup/client_cassandra.yml:

   ```
   cassandra:
     backup:
       client:
         enabled: true
         restore_latest: 1
         restore_from: remote
   ```

   where:

   - restore_latest can have, for example, the following values:
     - 1, which means restoring the database from the last complete backup.
     - 2, which means restoring the database from the second latest complete backup.
   - restore_from can have the local or remote values. The remote value uses scp to get the files from the cassandra server.
4. Restore the Cassandra database using the Jenkins web UI:
   1. Verify that the following class is present in cluster/cicd/control/leader.yml:

      ```
      classes:
      - system.jenkins.client.job.deploy.update.restore_cassandra
      ```

      If you manually add this class, apply the changes:

      ```
      salt -C 'I@jenkins:client' state.sls jenkins.client
      ```

2. Log in to the Jenkins web UI.
3. Open the cassandra - restore pipeline.
4. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.

5. Click Deploy.

Back up and restore a ZooKeeper database

MCP uses a custom script to back up a ZooKeeper database. ZooKeeper is a part of every OpenContrail deployment and the backup utility includes the following components:

- ZooKeeper server stores ZooKeeper backups rsynced from the ZooKeeper client nodes and runs on any node, for example, the Salt Master node.
- ZooKeeper client sends database backups to the ZooKeeper server and always runs on all the nodes of the ZooKeeper cluster. However, the back up is performed only on the leader node.

This section describes how to create and restore the ZooKeeper cluster on your OpenContrail 3.2 or 4.x cluster.

OpenContrail 3.2: Create a backup schedule for a ZooKeeper database

This section describes how to create a backup schedule for a ZooKeeper database on an OpenContrail 3.2 cluster.

To create a backup schedule for a ZooKeeper database:

1. Log in to the Salt Master node.
2. Configure the zookeeper server role by adding the following class to cluster/infra/config.yml:

   ```
   classes:
   - system.zookeeper.backup.server.single
   parameters:
     _param:
        zookeeper_backup_public_key: <generate_your_keypair>
   ```

   By default, adding this include statement results in ZooKeeper backup server keeping five full backups. To change the default setting, include the following pillar to cluster/infra/config.yml.

   ```
   parameters:
     zookeeper:
       backup:
         server:
           enabled: true
           hours_before_full: 24
           full_backups_to_keep: 5
   ```

3. Configure the zookeeper client role by adding the following lines to cluster/opencontrail/control.yml:

   ```
   classes:
   - system.zookeeper.backup.client.single
   parameters:
     _param:
       zookeeper_remote_backup_server: cfg01
       root_private_key: |
         <generate_your_keypair>
   ```

   By default, adding this include statement results in ZooKeeper keeping three complete backups on the zookeeper client node. The rsync command moves the backup files to the Salt Master node. To change the default setting, include the following pillar to cluster/opencontrail/control.yml:

   ```
   parameters:
     zookeeper:
       backup:
   ```

```
    client:
      enabled: true
      full_backups_to_keep: 3
      hours_before_full: 24
      target:
        host: cfg01
```

> Note
>
> The target.host parameter must contain the resolvable hostname of the host where the zookeeper server is running.

4. If you customized the default parameters, verify that the hours_before_full parameter of the zookeeper client in cluster/opencontrail/control.yml matches the same parameter of the zookeeper server in cluster/infra/config.yml.
5. Run the following command on the Salt Master node:

```
salt '*' saltutil.refresh_pillar
```

6. Apply the salt.minion state:

```
salt -C 'I@zookeeper:backup:client or I@zookeeper:backup:server' state.sls salt.minion
```

7. Refresh grains for the zookeeper client node:

```
salt -C 'I@zookeeper:backup:client' saltutil.sync_grains
```

8. Update the mine for the zookeeper client node:

```
salt -C 'I@zookeeper:backup:client' mine.flush
salt -C 'I@zookeeper:backup:client' mine.update
```

9. Apply the following state for the zookeeper client node:

```
salt -C 'I@zookeeper:backup:client' state.sls openssh.client,zookeeper.backup
```

10. Apply the following states for the zookeeper server node:

```
salt -C 'I@zookeeper:backup:server' state.apply linux.system
salt -C 'I@zookeeper:backup:server' state.sls zookeeper.backup
```

Seealso

- OpenContrail 3.2: Create an instant backup of ZooKeeper
- OpenContrail 3.2: Restore a ZooKeeper database

OpenContrail 3.2: Create an instant backup of ZooKeeper

After you create a backup schedule as described in OpenContrail 3.2: Create a backup schedule for a ZooKeeper database, you may also need to create an instant backup of a ZooKeeper database on an OpenContrail 3.2 cluster.

To create an instant backup of a ZooKeeper database:

1. Verify that you have completed the steps described in OpenContrail 3.2: Create a backup schedule for a ZooKeeper database.
2. Find out which OpenContrail control node is the ZooKeeper leader.

   ```
   salt -C 'I@opencontrail:control' cmd.run 'echo stat | nc localhost 2181 | grep leader'
   ```

3. Log in to the OpenContrail control leader node, for example, ntw01.
4. Run the following script:

   ```
   /usr/local/bin/zookeeper-backup-runner.sh
   ```

5. Verify that a complete backup has been created on the OpenContrail control leader node:

   ```
   ls /var/backups/zookeeper/full
   ```

6. Log in to the zookeeper server node and verify that the complete backup was rsynced to this node by executing the following command:

   ```
   ls /var/backups/zookeeper/full
   ```

Seealso

OpenContrail 3.2: Restore a ZooKeeper database

OpenContrail 3.2: Restore a ZooKeeper database

You may need to restore a ZooKeeper database after a hardware or software failure.

To restore a ZooKeeper database:

1. Log in to the Salt Master node.
2. Add the following lines to cluster/opencontrail/control.yml:

   ```
   zookeeper:
     backup:
       client:
         enabled: true
         restore_latest: 1
         restore_from: remote
   ```

   where:

   - restore_latest can have, for example, the following values:
     - 1, which means restoring the database from the last complete backup.
     - 2, which means restoring the database from the second latest complete backup.
   - restore_from can have the local or remote values. The remote value uses scp to get the files from the zookeeper server.
3. Proceed either with automatic restore steps using the Jenkins web UI pipeline or with manual restore steps:

   - Automatic restore steps:
     1. Add the upgrade pipeline to DriveTrain:
        1. Add the following lines to cluster/cicd/control/leader.yml:

           ```
           classes:
           - system.jenkins.client.job.deploy.update.restore_zookeeper
           ```
        2. Run the salt -C 'I@jenkins:client' state.sls jenkins.client state.
     2. Log in to the Jenkins web UI.
     3. Open the zookeeper - restore pipeline.
     4. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.

SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.

5. Click Deploy.

- Manual restore steps:

1. Stop the supervisor-config service on the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' service.stop supervisor-config
```

2. Stop the supervisor-control service on the OpenContrail control nodes:

```
salt -C 'I@opencontrail:control' service.stop supervisor-control
```

3. Stop the zookeeper service on the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' service.stop zookeeper
```

4. Remove the Zookeeper files on OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'rm -rf /var/lib/zookeeper/version-2/*'
```

5. Run the zookeeper state.

   Mirantis recommends running the following command using Linux GNU Screen or alternatives.

```
salt -C 'I@opencontrail:control' cmd.run "su root -c 'salt-call state.sls zookeeper'"
```

   This state restores the databases and creates a file in /var/backups/zookeeper/dbrestored.

   Caution!

   If you rerun the state, it will not restore the database again. To repeat the restore procedure, first delete the /var/backups/zookeeper/dbrestored file and then rerun the zookeeper state again.

6. Start the zookeeper service on the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' service.start zookeeper
```

©2025, Mirantis Inc.

7. Start the supervisor-config service on the OpenContrail control nodes:

   ```
   salt -C 'I@opencontrail:control' service.start supervisor-config
   ```

8. Start the supervisor-control service on the OpenContrail control nodes:

   ```
   salt -C 'I@opencontrail:control' service.start supervisor-control
   ```

9. Start the zookeeper service on the OpenContrail controller nodes:

   ```
   salt -C 'I@opencontrail:control' service.start zookeeper
   ```

10. Verify that the Zookeeper files are present again on OpenContrail controller nodes:

    ```
    salt -C 'I@opencontrail:control' cmd.run 'ls /var/lib/zookeeper/version-2'
    ```

11. Wait 60 seconds and verify that the OpenContrail is in correct state on OpenContrail controller nodes:

    ```
    salt -C 'I@opencontrail:control' cmd.run 'contrail-status'
    ```

OpenContrail 4.x: Create a backup schedule for a ZooKeeper database

This section describes how to create a backup schedule for a ZooKeeper database on an OpenContrail 4.x cluster.

To create a backup schedule for a ZooKeeper database:

1. Log in to the Salt Master node.
2. Configure the zookeeper server role using the following parameters in cluster/<cluster_name>/infra/backup/server.yml:

```
parameters:
  zookeeper:
    backup:
      cron: true
      server:
        enabled: true
        hours_before_full: 24
        full_backups_to_keep: 5
        backup_dir: /srv/volumes/backup/zookeeper
```

   By default, ZooKeeper backup server keeps five full backups.

3. Configure the zookeeper client role by adding the following parameters to cluster/<cluster_name>/infra/backup/client_zookeeper.yml:

```
parameters:
  _param:
    zookeeper_remote_backup_server: ${_param:infra_kvm_node03_address}
  zookeeper:
    backup:
      cron: True
      client:
        enabled: true
        full_backups_to_keep: 3
        hours_before_full: 24
        containers:
        - opencontrail_controller_1
```

   Caution!

   The zookeeper_remote_backup_server parameter must contain the resolvable hostname of the host on which the zookeeper server is running.

The ZooKeeper backup process keeps three complete backups on the zookeeper client node. The rsync command moves the backup files to Zookeeper server node.

4. If you customized the default parameters, verify that the hours_before_full parameter of the zookeeper client in cluster/<cluster_name>/infra/backup/client_zookeeper.yml matches the same parameter of the zookeeper server in cluster/<cluster_name>/infra/backup/server.yml.
5. Run the following command on the Salt Master node:

```
salt '*' saltutil.refresh_pillar
```

6. Apply the salt.minion state:

```
salt -C 'I@zookeeper:backup:client or I@zookeeper:backup:server' state.sls salt.minion
```

7. Refresh grains for the zookeeper client node:

```
salt -C 'I@zookeeper:backup:client' saltutil.sync_grains
```

8. Update the mine for the zookeeper client node:

```
salt -C 'I@zookeeper:backup:client' mine.flush
salt -C 'I@zookeeper:backup:client' mine.update
```

9. Apply the following state on the zookeeper client node:

```
salt -C 'I@zookeeper:backup:client' state.sls openssh.client,zookeeper.backup
```

10. Apply the following state on the zookeeper server node:

```
salt -C 'I@zookeeper:backup:server' state.sls zookeeper.backup
```

Seealso

- OpenContrail 4.x: Create an instant backup of ZooKeeper
- OpenContrail 4.x: Restore a ZooKeeper database

OpenContrail 4.x: Create an instant backup of ZooKeeper

After you create a backup schedule as described in OpenContrail 4.x: Create a backup schedule for a ZooKeeper database, you may also need to create an instant backup of a ZooKeeper database on an OpenContrail 4.x cluster.

To create an instant backup of a ZooKeeper database:

1. Verify that you have completed the steps described in OpenContrail 4.x: Create a backup schedule for a ZooKeeper database.
2. Find out which OpenContrail control node is the ZooKeeper leader.

   ```
   salt -C 'I@opencontrail:control' cmd.run 'echo stat | nc localhost 2181 | grep leader'
   ```

3. Log in to the OpenContrail control leader node, for example, ntw01.
4. Run the following script:

   ```
   /usr/local/bin/zookeeper-backup-runner.sh
   ```

5. Verify that a complete backup has been created on the OpenContrail control leader node:

   ```
   ls /var/backups/zookeeper/full
   ```

6. Log in to the zookeeper server node and verify that the complete backup was rsynced to this node by executing the following command:

   ```
   ls /srv/volumes/backup/zookeeper/full
   ```

Seealso

OpenContrail 4.x: Restore a ZooKeeper database

OpenContrail 4.x: Restore a ZooKeeper database

You may need to restore a ZooKeeper database after a hardware or software failure.

To restore a ZooKeeper database:

1. Log in to the Salt Master node.
2. Open your project Git repository with the Reclass model on the cluster level.
3. Add the following snippet to cluster/<cluster_name>/infra/backup/client_zookeeper.yml:

   ```
   zookeeper:
     backup:
       client:
         enabled: true
         restore_latest: 1
         restore_from: remote
   ```

   where:

   - restore_latest can have, for example, the following values:
     - 1, which means restoring the database from the last complete backup.
     - 2, which means restoring the database from the second latest complete backup.
   - restore_from can have the local or remote values. The remote value uses scp to get the files from the zookeeper server.
4. Proceed either with automatic restore steps using the Jenkins web UI pipeline or with manual restore steps:

   - Automatic restore steps:
     1. Verify that the following class is present in cluster/cicd/control/leader.yml:

        ```
        classes:
        - system.jenkins.client.job.deploy.update.restore_zookeeper
        ```

        If you manually add this class, apply the changes:

        ```
        salt -C 'I@jenkins:client' state.sls jenkins.client
        ```
     2. Log in to the Jenkins web UI.
     3. Open the zookeeper - restore pipeline.
     4. Specify the following parameters:

Parameter	Description and values

SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.

5. Click Deploy.

- Manual restore steps:

1. Stop the config services on the OpenContrail control nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-api'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-schema'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-svc-monitor'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-device-manager'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-config-nodemgr'
```

2. Stop the control services on the OpenContrail control nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-control'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-named'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-dns'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl stop contrail-control-nodemgr'
```

3. Stop the zookeeper service on the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller service zookeeper stop'
```

4. Remove the ZooKeeper files from the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'rm -rf /var/lib/config_zookeeper_data/version-2/*'
```

5. Run the zookeeper state.

   Mirantis recommends running the following command using Linux GNU Screen or alternatives.

```
salt -C 'I@opencontrail:control' state.apply zookeeper.backup
```

   This state restores the databases and creates a file in /var/backups/zookeeper/dbrestored.

> **Caution!**
>
> If you rerun the state, it will not restore the database again. To repeat the restore procedure, first delete the /var/backups/zookeeper/dbrestored file and then rerun the zookeeper state again.

6. Verify that the ZooKeeper files are present again on the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller ls /var/lib/zookeeper/version-2'
```

7. Start the zookeeper service on the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller service zookeeper start'
```

8. Start the config services on the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-api'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-schema'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-svc-monitor'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-device-manager'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-config-nodemgr'
```

9. Start the control services on the OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-control'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-named'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-dns'
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller systemctl start contrail-control-nodemgr'
```

10. Wait 60 seconds and verify that the OpenContrail is in correct state on OpenContrail controller nodes:

```
salt -C 'I@opencontrail:control' cmd.run 'doctrail controller contrail-status'
```

Back up and restore a MAAS PostgreSQL database

The Mirantis Cloud Platform (MCP) uses the PostgreSQL database to store all state information for the MAAS server provisioning tool of MCP. Mirantis recommends backing up your MAAS PostgreSQL databases daily to ensure the integrity of your data. You may also need to create a database backup for testing purposes.

MCP uses the Backupninja utility to back up and restore the MAAS PostgreSQL database. You can also use Backupninja to copy the MAAS PostgreSQL databases.

Backupninja installs automatically with your cloud environment using a SaltStack formula and includes the following components:

- Backupninja server collects requests from the Backupninja client nodes and runs on any node, for example, the Salt Master node.
- Backupninja client sends database backups to the Backupninja server and runs on the database cluster nodes.

Backupninja collects and backs up all the MAAS PostgreSQL database data to successfully restore the system from scratch.

This section describes how to create and restore your MAAS PostgreSQL databases using Backupninja.

Back up a MAAS PostgreSQL database prior to 2019.2.5

This section describes how to back up a MAAS PostgreSQL database prior to the MCP 2019.2.5 maintenance update.

Enable a backup schedule for a MAAS PostgreSQL database using Backupninja

This section describes how to create a backup schedule for a MAAS PostgreSQL database using the Backupninja utility. By default, Backupninja runs daily at 1.00 AM.

To create a backup schedule for a MAAS PostgreSQL database:

1. Log in to the Salt Master node.
2. Add the following lines to cluster/infra/maas.yml:

   ```
   classes:
   - system.backupninja.client.single
   - system.openssh.client.root
   parameters:
     _param:
       backupninja_backup_host: <IP>
       root_private_key: |
         <generate_your_keypair>
   ```

   > Note
   >
   > The backupninja_backup_host parameter is the backupninja server that runs on any server, for example, on the Salt Master node.

3. Include the following pillar to the node that runs the backupninja server and will store the database backups remotely:

   ```
   classes:
   - system.backupninja.server.single
   parameters:
     _param:
        backupninja_public_key: <generate_your_keypair>
   ```

4. Optionally, override the default configuration of the backup schedule as described in Configure a backup schedule for a MAAS PostgreSQL database.
5. Apply the salt.minion state:

   ```
   salt -C 'I@backupninja:client or I@backupninja:server' state.sls salt.minion
   ```

6. Refresh grains for the backupninja client node:

   ```
   salt -C 'I@backupninja:client' saltutil.sync_grains
   ```

7. Update the mine for the backupninja client node:

```
salt -C 'I@backupninja:client' mine.flush
salt -C 'I@backupninja:client' mine.update
```

8. Apply the backupninja state on the backupninja client node:

   ```
   salt -C 'I@backupninja:client' state.sls backupninja
   ```

   Applying this state creates two backup configuration scripts. By default, the scripts are stored in the /etc/backup.d/ directory and will run daily at 1:00 AM.

9. Refresh grains for the backupninja server node:

   ```
   salt -C 'I@backupninja:server' state.sls salt.minion.grains
   ```

10. Apply the backupninja state on the backupninja server node:

   ```
   salt -C 'I@backupninja:server' state.sls backupninja
   ```

Configure a backup schedule for a MAAS PostgreSQL database

By default, Backupninja runs daily at 1.00 AM. This section describes how to override the default configuration of the backup schedule. To enable the backup schedule in your deployment, refer to Enable a backup schedule for a MAAS PostgreSQL database using Backupninja.

To configure a backup schedule for a MAAS PostgreSQL database:

1. Log in to the Salt Master node.
2. Edit the cluster/infra/maas.yml file as required. Select from the following options:

    - Set the exact time of the backup server role to override the default backup time. The backup_times parameters include:

        - day_of_week

            The day of a week to perform backups. Specify 0 for Sunday, 1 for Monday, and so on. If not set, defaults to *.

        - day_of_month

            The day of a month to perform backups. For example, 20, 25, and so on. If not set, defaults to *.

            > Note
            >
            > Only day_of_week or day_of_month can be active at the same time. If both are defined, day_of_week is prioritized.

        - hour

            The hour to perform backups. Uses the 24 hour format. If not defined, defaults to 1.

        - minute

            The minute to perform backups. For example, 5, 10, 59, and so on. If not defined, defaults to 00.

      Configuration example:

      ```
      parameters:
        _param:
          backupninja:
            enabled: true
            client:
              backup_times:
                day_of_week: 1
                hour: 15
                minute: 45
      ```

> Note
>
> These settings will change global Backupninja schedule. If not set differently for individual steps, it will run all steps in the right order. This is recommended way of defining exact backup order.

- Disable automatic backups:

```
parameters:
  _param:
    backupninja:
      enabled: true
      client:
        auto_backup_disabled: true
```

- Re-enable automatic backups by setting the auto_backup_disabled parameter to false or delete the related line in cluster/infra/maas.yml:

```
parameters:
  _param:
    backupninja:
      enabled: true
      client:
        auto_backup_disabled: false
```

3. Apply changes by performing the steps 5-10 of the Enable a backup schedule for a MAAS PostgreSQL database using Backupninja procedure.

Back up and restore a MAAS PostgreSQL database starting from 2019.2.5

This section describes how to back up and restore a MAAS PostgreSQL database starting from the MCP 2019.2.5 maintenance update.

Enable a backup schedule for a MAAS PostgreSQL database using Backupninja

This section describes how to create a backup schedule for a MAAS PostgreSQL database using the Backupninja utility. By default, Backupninja runs daily at 1.00 AM.

To create a backup schedule for a MAAS PostgreSQL database:

1. Log in to the Salt Master node.
2. Add the following lines to cluster/infra/maas.yml:

```
classes:
- system.backupninja.client.single
- system.openssh.client.root
parameters:
  _param:
    backupninja_backup_host: <IP>
    root_private_key: |
      <generate_your_keypair>
```

> Note
>
> The backupninja_backup_host parameter is the IP address or the host name of a node running the backupninja server. To obtain this IP address or host name, run salt -C 'I@backupninja:server' grains.item fqdn_ip4 or salt -C 'I@backupninja:server' grains.item fqdn respectively.

3. Verify that root_private_key exists:

```
salt-call pillar.get '_param:root_private_key'
```

4. If the root_private_key parameter is missing, include the following pillar to the node that runs the backupninja server and will store the database backups remotely:

```
classes:
- system.backupninja.server.single
parameters:
  _param:
     backupninja_public_key: <generate_your_keypair>
```

5. Optionally, override the default configuration of the backup schedule as described in Configure a backup schedule for a MAAS PostgreSQL database.
6. Apply the salt.minion state:

```
salt -C 'I@backupninja:client or I@backupninja:server' state.sls salt.minion
```

7. Refresh grains for the backupninja client node:

```
salt -C 'I@backupninja:client' saltutil.sync_grains
```

8. Update the mine for the backupninja client node:

```
salt -C 'I@backupninja:client' mine.update
```

9. Apply the backupninja state on the backupninja client node:

```
salt -C 'I@backupninja:client' state.sls backupninja
```

   Applying this state creates two backup configuration scripts. By default, the scripts are stored in the /etc/backup.d/ directory and will run daily at 1:00 AM.

10. Refresh grains for the backupninja server node:

```
salt -C 'I@backupninja:server' state.sls salt.minion.grains
```

11. Apply the backupninja state on the backupninja server node:

```
salt -C 'I@backupninja:server' state.sls backupninja
```

Configure a backup schedule for a MAAS PostgreSQL database

> Warning
>
> This configuration presupposes manual backups or backups performed by a cronjob. If you use the Backupninja backup pipeline job, see Configure the Backupninja backup pipeline.

By default, Backupninja runs daily at 1.00 AM. This section describes how to override the default configuration of the backup schedule. To enable the backup schedule in your deployment, refer to Enable a backup schedule for a MAAS PostgreSQL database using Backupninja.

To configure a backup schedule for a MAAS PostgreSQL database:

1. Log in to the Salt Master node.
2. Edit the cluster/infra/maas.yml file as required. Select from the following options:

    - Set the exact time of the backup server role to override the default backup time. The backup_times parameters include:

        - day_of_week

            The day of a week to perform backups. Specify 0 for Sunday, 1 for Monday, and so on. If not set, defaults to *.

        - day_of_month

            The day of a month to perform backups. For example, 20, 25, and so on. If not set, defaults to *.

            > Note
            >
            > Only day_of_week or day_of_month can be active at the same time. If both are defined, day_of_week is prioritized.

        - hour

            The hour to perform backups. Uses the 24 hour format. If not defined, defaults to 1.

        - minute

            The minute to perform backups. For example, 5, 10, 59, and so on. If not defined, defaults to 00.

      Configuration example:

      ```
      parameters:
        _param:
          backupninja:
      ```

```
    enabled: true
    client:
      backup_times:
        day_of_week: 1
        hour: 15
        minute: 45
```

Note

These settings will change global Backupninja schedule. If not set differently for individual steps, it will run all steps in the right order. This is recommended way of defining exact backup order.

- Disable automatic backups:

  ```
  parameters:
    _param:
      backupninja:
        enabled: true
        client:
          auto_backup_disabled: true
  ```

- Re-enable automatic backups by setting the auto_backup_disabled parameter to false or delete the related line in cluster/infra/maas.yml:

  ```
  parameters:
    _param:
      backupninja:
        enabled: true
        client:
          auto_backup_disabled: false
  ```

3. Apply changes by performing the steps 5-10 of the Enable a backup schedule for a MAAS PostgreSQL database using Backupninja procedure.

Create an instant backup of a MAAS PostgreSQL database

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

After you create a backup schedule, you may also need to create an instant backup of a MAAS ProgreSQL database.

To create an instant backup of a MAAS PostgreSQL database using the Backupninja service:

1. Verify that you have completed the steps described in Enable a backup schedule for a MAAS PostgreSQL database using Backupninja.
2. Select one of the following options:
   - Create an instant backup of a MAAS PostgreSQL database automatically as described in Create an instant backup using Backupninja pipeline.
   - Create an instant backup of a MAAS PostgreSQL database manually:
     1. Verify that you have the postgresql-client-9.6 package installed on the Salt Master node. Otherwise, install it:

        ```
        salt -C 'I@backupninja:client' pkg.install postgresql-client,postgresql-client-9.6
        ```
     2. Verify that you have completed the steps described in Enable a backup schedule for a MAAS PostgreSQL database using Backupninja.
     3. Log in to the MAAS node, for example, cfg01.
     4. Make a backup of file_permissions.txt for MAAS:

        ```
        which getfacl && getfacl -pR /var/lib/maas/ > /var/lib/maas/file_permissions.txt
        ```
     5. Compress all MAAS PostgreSQL databases and store them in /var/backups/postgresql/:

        ```
        backupninja -n --run /etc/backup.d/102.pgsql
        ```
     6. Move the local backup files to the backupninja server using rsync:

        ```
        backupninja -n --run /etc/backup.d/200.backup.rsync
        ```

Restore the MAAS PostgreSQL database using Backupninja

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

You may need to restore a MAAS PostgreSQL database after a hardware or software failure or if you want to create a clone of the existing database from a backup. This section instructs you on how to run a restore of a MAAS PostgreSQL database using the Backupninja service.

To restore a MAAS PostgreSQL database using the Backupninja service:

Select one of the following options:

- Restore the MAAS PostgreSQL database automatically as described in Restore the services using Backupninja pipeline.
- Restore the MAAS PostgreSQL database manually:
  1. Log in to the Salt Master node.
  2. Include the following pillar to cluster/infra/maas.yml to restore the MAAS PostgreSQL database from any database node:

     ```
     classes:
     - system.maas.region.single
     - system.maas.region.restoredb
     parameters:
       _param:
         backupninja_backup_host: <IP>
     ```

  3. Apply the maas.region state to the corresponding nodes with MAAS.

     Mirantis recommends running the following command using Linux GNU Screen or alternatives.

     ```
     salt -C I@maas:region state.sls maas.region
     ```

     Running this state restores the databases and creates a file for every restored database in /root/maas/flags.

Caution!

If you rerun the state, it will not restore the database again. To repeat the restore procedure for any database, first delete the database file from /root/maas/flags and then rerun the maas.region state again.

Back up and restore the Dogtag server files and database

This section describes how to back up and restore the Dogtag server files and database.

Back up the Dogtag server files and database prior to 2019.2.6

This section describes how to back up the Dogtag server files and database prior to the MCP 2019.2.6 maintenance update.

Prepare for the Dogtag backup

This section describes how to prepare your environment for the backup of the Dogtag server files and database.

To prepare your environment for the Dogtag backup:

1. Log in to Salt Master node.
2. Open the cluster Reclass level of your deployment.
3. In openstack/barbican.yml, add the following configuration:

```
classes:
- system.openssh.client.root
parameters:
  backupninja:
    client:
      enabled: false
```

4. Apply the openssh state on the nodes with Dogtag to create the private key:

```
salt -C 'I@dogtag:server' state.sls openssh
```

5. Verify that the private key has been created with correct permissions:

```
salt -C 'I@dogtag:server' cmd.run 'ls -la /root/.ssh'
```

   The file with the following permissions should be present:

```
-r-------- 1 root root <DATE> <TIME> id_rsa
```

6. Update authenticated hosts on the backup server node:

```
salt -C '*' state.sls salt.minion.grains
salt -C '*' mine.update
salt -C 'I@backupninja:server' state.sls backupninja
```

7. Verify that the authorized_keys file on the backup server node contains the IP addresses for the kmn nodes:

```
salt -C 'I@backupninja:server' cmd.run "su -l backupninja -c 'cat ~/.ssh/authorized_keys'"
```

   Example of system response:

```
no-pty,from="10.11.0.67,10.11.0.66,10.11.0.15,10.11.0.65" ssh-rsa AAAAB3NzaC.....6tS7iqfkZ
```

> Note
>
> Note the IP addresses from the from parameter.

Now, you can proceed with Back up the Dogtag server files and database.

Back up the Dogtag server files and database

After you complete Prepare for the Dogtag backup, you can perform the Dogtag backup.

> **Warning**
>
> The content of a Dogtag database is tightly connected with the content of a Barbican database running on the Galera cluster. Therefore, we recommend running backups of Dogtag and Barbican simultaneously.

To back up the Dogtag server files and database:

1. Log in to Salt Master node.
2. Obtain the host name of the remote server node:

   ```
   salt -C 'I@backupninja:server' pillar.get "linux:network:fqdn"
   ```

3. Create a backup directory:

   ```
   salt -C 'I@dogtag:server and *01*' cmd.run "mkdir -p /var/backups/dogtag"
   ```

4. Export the signing certificate and key:

   1. Export the credentials. For example:

      ```
      salt -C 'I@dogtag:server and *01*' cmd.run "grep internal= /var/lib/pki/pki-tomcat/conf/password.conf | awk -F= '{print $2}' > /etc/dogtag/internal.txt"
      salt -C 'I@dogtag:server and *01*' cmd.run "grep internaldb= /var/lib/pki/pki-tomcat/conf/password.conf | awk -F= '{print $2}' > /etc/dogtag/pass.txt"
      ```

   2. Export the certificate. For example:

      ```
      salt -C 'I@dogtag:server and *01*' cmd.run "PKCS12Export -debug -d /var/lib/pki/$PKINAME/alias -p /etc/dogtag/internal.txt -o /etc/dogtag/ca-certs.p12 -w /etc/dogtag/pass.txt"
      ```

   3. Remove the internal password file:

      ```
      salt -C 'I@dogtag:server and *01*' cmd.run "rm -f /etc/dogtag/internal.txt"
      ```

5. Export CSR:

   ```
   salt -C 'I@dogtag:server and *01*' cmd.run "echo '-----BEGIN NEW CERTIFICATE REQUEST-----' > /etc/dogtag/ca_signing.csr"
   salt -C 'I@dogtag:server and *01*' cmd.run "sed -n '/^ca.signing.certreq=/ s/^[^=]*=// p' < /var/lib/pki/pki-tomcat/ca/conf/CS.cfg >> /etc/dogtag/ca_signing.csr"
   salt -C 'I@dogtag:server and *01*' cmd.run "echo '-----END NEW CERTIFICATE REQUEST-----' >> /etc/dogtag/ca_signing.csr"
   ```

6. Run the database backup:

   ```
   salt -C 'I@dogtag:server and *01*' cmd.run "/usr/sbin/db2bak-online -Z pki-tomcat -j /etc/dogtag/pass.txt -A /var/lib/dirsrv/slapd-pki-tomcat/bak"
   ```

   Note the backup directory from the system response. For example:

   ```
   Back up directory: /var/lib/dirsrv/slapd-pki-tomcat/bak/pki-tomcat-2019_8_14_11_28_25
   ```

©2025, Mirantis Inc.

7. Remove the Dogtag password file:

```
salt -C 'I@dogtag:server and *01*' cmd.run "rm -f /etc/dogtag/pass.txt"
```

8. Create a .tar archive that contains the Dogtag server files. For example:

```
salt -C 'I@dogtag:server and *01*' cmd.run "tar czvf /var/backups/dogtag/dogtag_backup-$(date +%F_%H-%M-%S).tar.gz --ignore-failed-read -C / \
etc/pki/pki-tomcat \
etc/dogtag/ca-certs.p12 \
etc/dogtag/ca_signing.csr \
etc/sysconfig/pki-tomcat \
etc/sysconfig/pki/tomcat/pki-tomcat \
etc/systemd/system/pki-tomcatd.target.wants/pki-tomcatd@pki-tomcat.service \
var/lib/pki/pki-tomcat \
var/log/pki/pki-tomcat \
usr/share/pki/server/conf/database.conf \
usr/share/pki/server/conf/schema.conf
```

9. Create the remote backup directory. For example:

```
salt -C 'I@backupninja:server' cmd.run "mkdir -p /srv/volumes/backup/backupninja/dogtag"
```

10 Transfer the data to the remote node. For example:

```
salt -C 'I@dogtag:server and *01*' cmd.run "/usr/bin/rsync -rhtPpv --rsync-path=rsync --progress /var/backups/dogtag/* -e ssh backupninja@<remote_node>:/srv/volumes/backup/backupninja/dogtag"
```

> Note
>
> The command above transfers all backups from the backup directory, and not just the last one, unless they are already present on the remote node.

11 Verify that the file transfer has been completed:

```
salt -C 'I@backupninja:server' cmd.run "ls -l /srv/volumes/backup/backupninja/dogtag"
```

The backup directory should include the dogtag_backup-<some_timestamp>.tar.gz file.

Back up and restore the Dogtag server files and database starting from 2019.2.6

This section describes how to back up the Dogtag server files and database starting from the MCP 2019.2.6 maintenance update. MCP uses the Backupninja utility to back up the Dogtag server files and database.

> Note
>
> The Dogtag restore procedure is available starting from the 2019.2.11 maintenance update.

Prepare for the Dogtag backup and restore

This section describes how to prepare your environment for the backup and restore of the Dogtag server files and database.

To prepare for the Dogtag backup and restore:

1. Log in to the Salt Master node.
2. Open the cluster Reclass level of your deployment.
3. Verify that the cluster/<cluster_name>/infra/backup/client_dogtag.yml file with the Backupninja configuration for Dogtag exists. Otherwise, create such file with the following content:

```
classes:
- system.backupninja.client.single
- cluster.<cluster_name>.infra.backup.allow_root_ssh
parameters:
  _param:
    backupninja_backup_host: ${_param:infra_kvm_node03_address}
  backupninja:
    client:
      target:
        home_dir: /srv/volumes/backup/backupninja
        engine: rsync
        engine_opts: "-av --delete --recursive --safe-links"
      log:
        color: ${_param:backupninja_color_log}
```

4. In openstack/barbican.yml, add the client_dogtag file if it is not included:

```
- cluster.<cluster_name>.infra.backup.client_dogtag
```

5. Refresh grains:

```
salt -C '*' state.sls salt.minion.grains
salt -C '*' mine.update
```

6. Apply the openssh state on the Dogtag server nodes:

```
salt -C 'I@dogtag:server and I@backupninja:client' state.sls openssh
```

7. Apply the backupninja state on the Dogtag server nodes:

```
salt -C 'I@dogtag:server and I@backupninja:client' state.sls backupninja
```

8. Apply the backupninja state on the Backupninja server node:

```
salt -C 'I@backupninja:server' state.sls backupninja
```

Now, you can proceed with Back up the Dogtag server files and database.

Back up the Dogtag server files and database

After you complete Prepare for the Dogtag backup and restore, you can perform the Dogtag backup.

> **Warning**
>
> The content of a Dogtag database is tightly connected with the content of a Barbican database running on a Galera cluster. Therefore, we recommend running backups of Dogtag and Barbican simultaneously. For this reason, the Backupninja backup pipeline is triggered to back up Dogtag always once the backup of a Galera cluster using the Galera database backup pipeline is finished.

To back up the Dogtag server files and database:

1. Verify that you have completed the steps described in Prepare for the Dogtag backup and restore.
2. Select one of the following options:
   - Create an instant backup of the Dogtag files and database automatically as described in Create an instant backup using Backupninja pipeline.
   - Create an instant backup of the Dogtag files and database manually:
     1. Log in to the Salt Master node.
     2. Run the backup script:

        ```
        salt -C 'I@dogtag:server:role:master' cmd.run "su root -c 'backupninja --now -d'"
        ```

Restore the Dogtag server files and database

After you complete the Back up the Dogtag server files and database procedure, you can perform the Dogtag restore.

> Note
>
> - The Dogtag restore procedure is available starting from the 2019.2.11 maintenance update.
> - If Dogtag restore has been already performed on your cluster, first delete the .dogtag_restored file in /etc/salt.

To restore the Dogtag server files and database:

1. Verify that you have completed the steps described in Prepare for the Dogtag backup and restore.
2. Select from the following options:
   - Restore Dogtag files and database automatically as described in Restore the services using Backupninja pipeline.
   - Restore Dogtag files and database manually:
     1. Log in to the Salt Master node.
     2. In cluster/openstack/barbican.yml, configure the following pillar:

        ```
        parameters:
          dogtag:
            server:
              initial_data:
                engine: rsync
                source: ${_param:backupninja_backup_host}
                host: ${_param:openstack_barbican_node01_hostname}.${_param:cluster_domain}
                home_dir: '/path/to/backups/' # defaults to '/srv/volumes/backup/backupninja' if not defined
        ```

     3. Stop the database service on secondary nodes to temporarily disable replication:

        ```
        salt -C 'I@dogtag:server:role:slave' service.stop 'dirsrv@pki-tomcat.service'
        ```

     4. Apply the restore state.

        Mirantis recommends running the following command using Linux GNU Screen or alternatives.

        ```
        salt -C 'I@dogtag:server:role:master' state.sls dogtag.server.restore
        ```

     5. Start the database service on secondary nodes to enable replication:

```
salt -C 'I@dogtag:server:role:slave' service.start 'dirsrv@pki-tomcat.service'
```

6. Verify that all services are running:

```
salt -C 'I@dogtag:server' service.status "dirsrv@pki-tomcat.service"
```

   Expected system response:

```
kmn02.example.domain.local:
  True
kmn03.example.domain.local:
  True
kmn01.example.domain.local:
  True
```

7. Apply the states below in the following strict order:

```
salt -C 'I@salt:master' state.sls salt,reclass
salt -C 'I@dogtag:server and *01*' state.sls dogtag.server
salt -C 'I@dogtag:server' state.sls dogtag.server
salt -C 'I@haproxy:proxy' state.sls haproxy
salt -C 'I@barbican:server and *01*' state.sls barbican.server
salt -C 'I@barbican:server' state.sls barbican.server
salt -C 'I@barbican:server' cmd.run "rm /etc/barbican/alias/*"
salt -C 'I@barbican:server' service.restart apache2
```

   If the barbican.server state fails with the "Rendering SLS 'base:barbican.server' failed: Jinja variable 'dict object' has no attribute 'key'" error that may occur, for example, due to the mine data deletion after calling the mine.flush function:

   1. Obtain the Dogtag certificate location:

   ```
   salt -C 'I@dogtag:server:role:master' pillar.get dogtag:server:export_pem_file_path
   ```

      Example of system response:

   ```
   /etc/dogtag/kra_admin_cert.pem
   ```

   2. Apply the following state:

> Note
>
> In the state below, substitute the certificate path with the one you obtained in the previous step.

```
salt -C 'I@dogtag:server:role:master' mine.send dogtag_admin_cert \
mine_function=cmd.run 'cat /etc/dogtag/kra_admin_cert.pem'
```

3. Rerun the failed Barbican state.

©2025, Mirantis Inc.

Configure and use the Backupninja pipeline

This section describes the Backupninja pipelines. Before proceeding, verify that you have performed the steps described in the following sections depending on your use case:

- Back up and restore the Salt Master node starting from 2019.2.5
- Back up and restore a MAAS PostgreSQL database starting from 2019.2.5
- Back up and restore the Dogtag server files and database starting from 2019.2.6

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

Configure the Backupninja backup pipeline

The Backupninja backup pipeline supports backup of the following components:

- MAAS, including the PostgreSQL database that contains the data for MAAS and machines that are registered and managed by MAAS
- Salt Master, including the PKI keys, certificates, the Salt Minion on cfg01, and the metadata model
- Dogtag server files and database

To use the Backupninja backup pipeline, you must override the current Backupninja configuration and disable the default backup schedule. This section describes how to configure the Backupninja backup pipeline.

To configure the Backupninja backup pipeline:

1. Log in to the Salt Master node.
2. Verify that Backupninja scheduling is set to manual:

   ```
   salt-call pillar.get backupninja:client
   ```

   Example of system response:

   ```
   parameters:
     backupninja:
       client:
         scheduling:
           when:
             - manual
   ```

3. If the pillars do not include the parameter above or it has a different value:

   1. Manually override the value by adding the following block to infra/backup/client_common.yml.

      ```
      parameters:
        backupninja:
          client:
            scheduling:
              when:
                - manual
      ```

   2. Apply the backupninja state:

      ```
      salt -C 'I@backupninja:client' state.sls backupninja
      ```

4. Verify that the class with the Backupninja backup Jenkins job is present in the pillar:

```
salt -C 'I@jenkins:client and not I@salt:master' pillar.get jenkins:client:job:backupninja_backup
```

5. If the pillars do not include the Backupninja backup pipeline:

    1. Add the following class to cicd/control/leader.yml:

    ```
    - system.jenkins.client.job.deploy.backupninja_backup
    ```

    2. Rerun the jenkins state on the cid01 node:

    ```
    salt -C 'I@jenkins:client and not I@salt:master' state.sls jenkins.client
    ```

Configure the Backupninja restore pipeline

The Backupninja restore pipeline supports restore of the following components:

- MAAS, including the PostgreSQL database that contains the data for MAAS and machines that are registered and managed by MAAS
- Salt Master, including the PKI keys, certificates, the Salt Minion on cfg01, and the metadata model

To configure the Backupninja restore pipeline:

1. Log in to the Salt Master node.
2. Verify that the class with the Backupninja restore Jenkins job is present in the pillar:

   ```
   salt -C 'I@jenkins:client and not I@salt:master' pillar.get jenkins:client:job:backupninja_restore
   ```

3. If the pillars do not include the Backupninja restore pipeline:

   1. Add the following class to cicd/control/leader.yml:

      ```
      - system.jenkins.client.job.deploy.backupninja_restore
      ```

   2. Rerun the jenkins state on the cid01 node:

      ```
      salt -C 'I@jenkins:client and not I@salt:master' state.sls jenkins.client
      ```

4. If you plan to restore the Salt Master node and MAAS:

   1. In cluster/infra/config/init.yml, configure the pillar for salt-master and salt-minion:

      ```
      parameters:
        salt:
        master:
          initial_data:
            engine: backupninja
            source: ${_param:backupninja_backup_host}  # the backupninja server that stores Salt Master backups, for example: kvm03
            host: ${_param:infra_config_hostname}.${_param:cluster_domain}  # for example: cfg01.deploy-name.local
            home_dir: '/path/to/backups/' # for example: '/srv/volumes/backup/backupninja'
        minion:
          initial_data:
            engine: backupninja
            source: ${_param:backupninja_backup_host}  # the backupninja server that stores Salt Master backups
            host: ${_param:infra_config_hostname}.${_param:cluster_domain}  # for example: cfg01.deploy-name.local
            home_dir: '/path/to/backups/' # for example: '/srv/volumes/backup/backupninja'
      ```

      > Note
      >
      > If salt-master restore has been already performed on your cluster, first delete the master-restored and minion-restored files in /srv/salt.

2. In cluster/infa/maas.yml, add the following class for MAAS restore:

```
- system.maas.region.restoredb
```

5. Since 2019.2.12 If you plan to restore the Keystone credential keys and such restore operation has been already performed on your cluster, first delete the .keystone_restored file in /etc/salt.

Create an instant backup using Backupninja pipeline

This section describes how to create an instant backup of the MAAS PostgreSQL database, the Salt Master node, and the Dogtag server files and database using the Backupninja Jenkins pipeline job.

> Note
>
> Backup of the Dogtag server and database using the pipeline is available starting from the 2019.2.6 maintenance update only. For previous versions, use the pipeline job only to back up the Salt Master node and MAAS PostgreSQL database. To back up Dogtag on the MCP version prior to 2019.2.6, see: Back up the Dogtag server files and database prior to 2019.2.6.

> Warning
>
> The pipeline job executes all configured Backupninja jobs present on the related MAAS, Salt Master, and Dogtag hosts.

To create an instant backup using the Jenkins pipeline job:

1. Verify that you have completed the steps described in Enable a backup schedule for the Salt Master node using Backupninja and Enable a backup schedule for a MAAS PostgreSQL database using Backupninja to backup Salt master node and MaaS.
2. Verify that you have completed the steps described in Prepare for the Dogtag backup and restore.
3. Configure the backup pipeline job as described in Configure the Backupninja backup pipeline.
4. Log in to the Jenkins web UI.
5. Select from the following options:
    - For MCP versions prior to 2019.2.6, open the Backupninja salt-master/MaaS backup pipeline job.
    - For MCP versions starting from 2019.2.6, open the Backupninja backup pipeline pipeline job.
6. Specify the required parameters:

Backupninja backup pipeline parameters

Parameter	Description and values

SALT_MASTER_URL	Define the IP address of your Salt Master node host and the salt-api port. For example, http://172.18.170.27:6969.
SALT_MASTER_CREDENTIALS	Define credentials_id as credentials for the connection.
ASK_CONFIRMATION	Select if you want the pipeline job to wait for a manual confirmation before running specific steps.
BACKUP_SALTMASTER_AND_MAAS Added since 2019.2.6	Select to back up Salt Master and MAAS.
BACKUP_DOGTAG Added since 2019.2.6	Select to back up the Dogtag files and database.
BACKUP_KEYSTONE_CREDENTIAL_KEYS Added since 2019.2.12	Select to back up the Keystone credential keys.

7. Click Build.

> Warning
>
> During the first backup using the Jenkins pipeline job, a manual confirmation may be required to install the required packages. After the first successful run, no manual confirmation is required.
>
> To disable the manual confirmation, set the ASK_CONFIRMATION parameter to False.

The pipeline job workflow:

1. Pillar verification. Verify that initial_data in pillars are defined correctly to prevent any issues related to a wrong configuration during the execution of the pipeline job.
2. Backup location check. Verify that the backup node is ready and all required packages are installed.
3. Backup preparation. Verify that the backupninja states are up to date.
4. Backup execution.

8. Verify that the backup has been created and, in case of a remote backup storage, moved correctly.

Restore the services using Backupninja pipeline

This section describes how to restore the MAAS PostgreSQL database, Salt Master node, and Dogtag server files and database using the Backupninja Jenkins pipeline job.

> Note
>
> Restore of the Dogtag server and database using the pipeline is available starting from the 2019.2.11 maintenance update only. For previous versions, use the Backupninja pipeline job only to restore the Salt Master node and MAAS PostgreSQL database.

To restore the services using the Jenkins pipeline job:

1. Verify that you have completed the steps described in Configure the Backupninja restore pipeline.
2. Log in to the Jenkins web UI.
3. Select from the following options:
   - For MCP versions prior to 2019.2.6, open the Backupninja restore salt-master/MaaS backup pipeline job.
   - For MCP versions starting from 2019.2.6, open the Backupninja restore pipeline pipeline job.
4. Specify the required parameters:

Backupninja restore pipeline parameters

Parameter	Description and values
SALT_MASTER_URL	Add the IP address of your Salt Master node host and the salt-api port. For example, http://172.18.170.27:6969.
CREDENTIALS_ID	Add credentials_id as credentials for the connection.
RESTORE_SALTMASTER_AND_MAAS Added since 2019.2.6	Select to restore Salt Master and MAAS.
RESTORE_KEYSTONE_CREDENTIAL_KEYS Added since 2019.2.12	Select to restore the Keystone credential keys.
RESTORE_DOGTAG Added since 2019.2.11	Select to restore the Dogtag files and database.

5. Click Build.

   The Jenkins pipeline job workflow:

1. Pillar verification. Verify that initial_data in pillars are defined correctly to prevent any issues related to a wrong configuration during the execution of the pipeline job.
2. Perform the restore.

   If the pipeline job fails during the Dogtag restore with the "Rendering SLS 'base:barbican.server' failed: Jinja variable 'dict object' has no attribute 'key'" error that may occur, for example, due to the mine data deletion after calling the mine.flush function:

   1. Obtain the Dogtag certificate location:

      ```
      salt -C 'I@dogtag:server:role:master' pillar.get \
      dogtag:server:export_pem_file_path
      ```

      Example of system response:

      ```
      /etc/dogtag/kra_admin_cert.pem
      ```

   2. Apply the following state:

      Note

      In the state below, substitute the certificate path with the one you obtained in the previous step.

      ```
      salt -C 'I@dogtag:server:role:master' mine.send dogtag_admin_cert \
      mine_function=cmd.run 'cat /etc/dogtag/kra_admin_cert.pem'
      ```

   3. Rerun the failed Barbican state.

6. Verify that the restore completed and, in case of a remote backup storage, moved correctly:

   1. Verify the Salt Master node:

      1. Verify the Salt keys in /etc/salt/pki/.
      2. Verify the model in /srv/salt/reclass.
      3. Try to perform a test ping on the available Salt Minions using test.ping.

   2. Verify MAAS:

      1. Verify the MAAS UI.
      2. List the available machines in MAAS through the maasng.list_machines Salt module.

7. Verify Dogtag:

   1. Verify the server data.

2. Verify the saved secrets in the database.

Scheduled maintenance with a planned power outage

This section provides the instruction on how to correctly power off you MCP deployment partially or entirely in case of power outage due to a planned maintenance, upgrade, or power supply testing.

> Caution!
>
> If you plan on powering off a whole MCP cluster, verify that your cloud environment workloads and applications are ready for the compute nodes shutdown before proceeding.

Shut down an MCP OpenStack environment partially

You can power off your MCP OpenStack environment partially. Before you start powering off the required nodes, verify that two Galera dbs nodes are running. If only one Galera node is shut down and at least one another is running, you will avoid any downtime in your cloud environment.

Once the verification is performed, proceed with powering off the required nodes using the following command:

```
salt '<node_name>' system.poweroff
```

Shut down the whole MCP OpenStack environment

Before you proceed with the procedure, verify that all backups are up to date and, if possible, are stored externally.

If you have Ceph in your MCP cluster, power it off as described in Shut down a Ceph cluster.

To shut down an MCP OpenStack environment:

1. Power off the prx nodes to discontinue all external connections:

   ```
   salt 'prx*' system.poweroff
   ```

2. Power off the OpenStack bare metal bmt controller nodes:

   ```
   salt 'bmt*' system.poweroff
   ```

3. Power off the ctl nodes:

   ```
   salt 'ctl*' system.poweroff
   ```

4. If OpenContrail is deployed, power off the ntw and nal nodes:

   ```
   salt 'ntw*' system.poweroff
   salt 'nal*' system.poweroff
   ```

5. Power off the gtw nodes:

   ```
   salt 'gtw*' system.poweroff
   ```

6. Power off the msg nodes:

   ```
   salt 'msg*' system.poweroff
   ```

7. Power off the dbs nodes in the following strict order:

   ```
   salt 'dbs03*' system.poweroff
   salt 'dbs02*' system.poweroff
   salt 'dbs01*' system.poweroff
   ```

8. Power off all StackLight LMA nodes:

   ```
   salt 'log*' system.poweroff
   salt 'mon*' system.poweroff
   salt 'mtr*' system.poweroff
   ```

9. If you have any containers running on the cid nodes, for example, Jenkins, stop them.

10. Power off the cid nodes:

    ```
    salt 'cid*' system.poweroff
    ```

11. If local Aptly is used, power off the Aptly node.

    ```
    salt 'apt01*' system.poweroff
    ```

12. Power off the cmp nodes:

    ```
    salt 'cmp*' system.poweroff
    ```

13. Verify that all control plane instances are down except for the cfg01 node if the latter runs on a KVM node:

    ```
    salt 'kvm*' cmd.run 'virsh list | grep -v cfg01'
    ```

14. Power off the kvm nodes.

    Since the cfg01 node usually runs on a KVM node, postpone the KVM nodes shutdown, for example, for 15 minutes, to safely shut down the cfg01 node:

    > Warning
    >
    > After running the command below, you will lose any access to the environment in 15 minutes.

    ```
    salt 'kvm*' cmd.run 'shutdown --poweroff 15' --async
    ```

15. Power off the cfg01 node before the kvm nodes have powered off:

    ```
    salt 'cfg01*' system.poweroff
    ```

Start an MCP OpenStack environment after a power-off

This section instructs on how to correctly start an MCP OpenStack environment after it was powered off.

If you have Ceph in your MCP cluster, start it as described in Start a Ceph cluster.

To start an MCP OpenStack environment:

1. Start the kvm01 node hosting the dbs01 node.
2. Start the kvm02 node hosting the dbs02 node.
3. Start the kvm03 node hosting the dbs03 node.

   Note

   By default, the autostart option is enabled for the host nodes. Therefore, all VMs they host will start automatically.

   Otherwise, start the database server nodes in the following strict order:

   1. Start the dbs01 node.
   2. Start the dbs02 node.
   3. Start the dbs03 node.

4. Verify that the cluster is up. In case of issues, restore the cluster as described in Restore a Galera cluster.
5. Start the RabbitMQ server msg nodes.
6. Verify that the cluster is up. In case of issues, proceed with Troubleshoot RabbitMQ.
7. Start the gtw nodes.
8. Start the OpenStack controller ctl nodes.
9. Start the bare metal bmt nodes.
10. For the OpenContrail deployments, start the nal and ntw nodes.
11. Verify that the cluster is up. In case of issues, proceed with Troubleshoot OpenContrail.
12. Start the proxy prx nodes.
13. Start the StackLight LMA mon, mtr, and log nodes.
14. Start the StackLight OSS cid nodes.

Seealso

Troubleshoot an MCP OpenStack environment

Shut down a Ceph cluster

This section describes how to shut down an entire Ceph cluster. Make sure that your Ceph cluster is healthy before proceeding with the following steps.

1. Stop all clients that write to Ceph.
2. Shut down the RADOS Gateway nodes:

   ```
   salt 'rgw*' system.poweroff
   ```

3. Set the Ceph OSD flags:

   ```
   ceph osd set noout
   ceph osd set nobackfill
   ceph osd set norecover
   ceph osd set norebalance
   ceph osd set nodown
   ceph osd set pause
   ```

4. Shut down the Ceph OSD nodes one by one:

   ```
   salt 'osd01*' system.poweroff
   salt 'osd02*' system.poweroff
   salt 'osd03*' system.poweroff
   ...
   ```

5. Shut down the Ceph Monitor nodes one by one:

   ```
   salt 'cmn01*' system.poweroff
   salt 'cmn02*' system.poweroff
   salt 'cmn03*' system.poweroff
   ...
   ```

Start a Ceph cluster

This section describes how to correctly start a Ceph cluster after it was powered off.

1. Power on the Ceph Monitor nodes.
2. Power on the Ceph OSD nodes.
3. Wait for all nodes to become available.
4. Unset the Ceph OSD flags:

   ```
   ceph osd unset noout
   ceph osd unset nobackfill
   ceph osd unset norecover
   ceph osd unset norebalance
   ceph osd unset nodown
   ceph osd unset pause
   ```

5. Power on the RADOS Gateway nodes.
6. Start all clients that write to Ceph.

©2025, Mirantis Inc.

Upgrade and update an MCP cluster

A typical MCP cluster includes multiple components, such as DriveTrain, StackLight, OpenStack, OpenContrail, and Ceph. Most of MCP components have their own versioning schema. For the majority of the components, MCP supports multiple versions at once.

The upgrade of an MCP deployment to a new version is a multi-step process that needs to take into account the cross-dependencies between the components of the platform, and compatibility matrix of supported versions of the components.

The MCP components that do not have their own versioning schema within MCP and are versioned by the MCP release include:

- The DriveTrain components: Aptly, Gerrit, Jenkins, Reclass, Salt formulas and their subcomponents
- StackLight LMA

Caution!

Before proceeding with the upgrade procedure, verify that you have updated DriveTrain including Aptly, Gerrit, Jenkins, Reclass, Salt formulas, and their subcomponents to the current MCP release version. Otherwise, the current MCP product documentation is not applicable to your MCP deployment.

Note

Starting from the MCP 2019.2.16 maintenance update, before proceeding with any update or upgrade procedure, first verify that Nova cell mapping is enabled. For details, see Disable Nova cell mapping.

Note

Starting from the MCP 2019.2.17 maintenance update, before proceeding with the next update procedure, you can verify that the model contains information about the necessary fixes and workarounds. For details, see Verify DriveTrain.

For the MCP components with support for multiple versions, such as OpenStack or OpenContrail, you usually can select between two operations:

- **Minor version update (maintenance update)**

New minor versions of the components artifacts are installed. Services are restarted as necessary. This kind of update allows you to obtain the latest bug and security fixes for the components, but it typically does not change the components capabilities.

- **Major version update (upgrade)**

  New major versions of the components artifacts are installed. Additional orchestration tasks are executed to change the components configuration, if necessary. This kind of update typically changes and improves the components capabilities.

The following table outlines a general upgrade and update procedure workflow of an MCP cluster. For the detailed upgrade and update workflow of MCP components, refer to the corresponding sections below.

General upgrade and update procedure workflow

#	Stage	Description

1	Upgrade or update DriveTrain	Perform the basic LCM update or upgrade:
		1. Update the Reclass system.
		2. Fetch the corresponding Git repositories.
		3. Update all binary repository definitions on the Salt Master node.
		4. Update and sync all Salt formulas.
		5. Apply the linux.repo,linux.user and openssh states on all nodes.
		6. Upgrade or update the DriveTrain services.
		7. Optional. Upgrade system packages on the Salt Master node.
		8. Upgrade or update GlusterFS:
		    1. Upgrade or update packages for the GlusterFS server on each target host one by one.
		    2. Upgrade or update packages for the GlusterFS clients and re-mount volumes on each target GlusterFS client host one by one.
		    3. Obtain the cluster.max-op-version option value from GlusterFS and compare it with cluster.op-version to identify whether a version upgrade is required.
		    4. Update cluster.op-version.
		9. Optional. Configure allowed and rejected IP addresses for the GlusterFS volumes.

<table>
<tr><td>2</td><td>Upgrade or update OpenContrail (if applicable)</td><td>
1. Verify the OpenContrail service statuses.
2. Back up the Cassandra and ZooKeeper data.
3. Stop the Neutron server services.
4. Upgrade or update the OpenContrail analytics nodes simultaneously. During upgrade, new Docker containers for the OpenContrail analytics nodes are spawned. During update, the corresponding Docker images are updated.
5. Upgrade or update the OpenContrail controller nodes. During upgrade, new Docker containers for the OpenContrail controller nodes are spawned. During update, the corresponding Docker images are updated. All nodes are upgraded or updated simultaneously except the one that meantime runs the contrail-control service and is upgraded or updated after other nodes.
6. Upgrade or update the OpenContrail packages on the OpenStack controller nodes simultaneously.
7. Start the Neutron server services.
8. Upgrade or update the OpenContrail data plane nodes one by one with the workloads migration if needed since this step implies downtime of the Networking service.
</td></tr>
<tr><td>3</td><td>Upgrade or update OpenStack or Kubernetes</td><td>
For OpenStack:
1. On every OpenStack controller node one by one:
    1. Stop the OpenStack API services.
    2. Upgrade or update the OpenStack packages.
    3. Start the OpenStack services.
    4. Apply the OpenStack states.
    5. Verify that the OpenStack services are up and healthy.
2. Upgrade the OpenStack data plane.

Caution!
We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.
</td></tr>
</table>

4	Upgrade or update Galera	1. Prepare the Galera cluster for the upgrade. 2. Upgrade or update the MySQL and Galera packages on the Galera nodes one by one. 3. Verify the cluster status after upgrade.
5	Upgrade or update RabbitMQ	1. Prepare the Neutron service for the RabbitMQ upgrade or update. 2. Verify that the RabbitMQ upgrade pipeine job is present in Jenkins. 3. Upgrade or update the RabbitMQ component. **Caution!** We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.
		For Kubernetes: 1. Upgrade or update essential Kubernetes binaries, for example, hypercube, etcd, cni. 2. Restart essential Kubernetes services. 3. Upgrade or update the addons definitions with the latest images. 4. Perform the Kubernetes control plane changes, if any, on every Kubernetes Master node one by one. 5. Upgrade or update the Kubernetes Nodes one by one.

6	Upgrade or update StackLight	1. During upgrade, enable the Ceph Prometheus plugin (if applicable).
		2. Upgrade or update system components including Telegraf, Fluentd, Prometheus Relay, libvirt-exporter, and jmx-exporter.
		3. Upgrade or update Elasticsearch and Kibana one by one:
		    1. Stop the corresponding service on all log nodes.
		    2. Upgrade or update the packages to the newest version.
		    3. For Elasticsearch, reload the systemd configuration.
		    4. Start the corresponding service on all log nodes.
		    5. Verify that the Elasticsearch cluster status is green.
		    6. In case of a major version upgrade, transform the indices for the new version of Elasticsearch and migrate Kibana to the new index.
		4. Upgrade or update components running in Docker Swarm:
		    1. Disable and remove the previous versions of monitoring services.
		    2. Rebuild the Prometheus configuration by applying the prometheus state on the mon nodes.
		    3. Disable and remove the previous version of Grafana.
		    4. Start the monitoring services by applying the docker state on the mon nodes.
		    5. Apply the saltutil.sync_all state and the grafana.client state to refresh the Grafana dashboards.

<table>
<tr><td>7</td><td>Upgrade Ceph or Update Ceph</td><td>For upgrade:
Prepare the Ceph cluster for upgrade.
Perform the backup.
Upgrade the Ceph repository on each node one by one.
Upgrade the Ceph packages on each node one by one.
Restart the Ceph services on each node one by one.
Verify the upgrade on each node one by one and wait for user input to proceed.
Perform the post-upgrade procedures.
For update:
Update and install new Ceph packages on the cmn nodes.
Restart Ceph Monitor services on all cmn nodes one by one.
Starting from the 2019.2.8 update, restart Ceph Manager on all mgr nodes one by one.
Update and install new Ceph packages on the osd nodes.
Restart Ceph OSDs services on all osd nodes one by one.
Update and install new Ceph packages on the rgw nodes.
Restart Ceph RADOS Gateway services on all rgw nodes one by one.
After the restart of every service, wait for the system to become healthy.</td></tr>
<tr><td>8</td><td>Update the base operating system</td><td>Install security updates on all nodes.
To reduce the size of new packages to be installed on a cluster during update or upgrade, this is the final step of the procedure. However, you can perform it at any stage to fetch only security patches.</td></tr>
<tr><td>9</td><td>Apply issues resolutions requiring manual application described in the Addressed issues sections of all Maintenance updates.</td><td>Apply fixes that require manual application for all maintenance updates one by one.</td></tr>
</table>

Upgrade an MCP cluster

Caution!

Before proceeding with the upgrade procedure, verify that you have updated DriveTrain including Aptly, Gerrit, Jenkins, Reclass, Salt formulas, and their subcomponents to the current MCP release version. Otherwise, the current MCP product documentation is not applicable to your MCP deployment.

Use the procedures in this section to upgrade your MCP cluster. More specifically, major upgrades enable:

- Upgrading the MCP release version to the latest supported Build ID
- Delivering new features of MCP Control plane
- Updating the LCM platform
- Upgrading host and guest operating systems to new versions including kernels
- Upgrading between major OpenStack releases
- Upgrading a Kubernetes cluster including Calico and etcd
- Upgrading the OpenContrail nodes from version 3.2 to 4.1
- Upgrading StackLight LMA
- Upgrading Ceph

Warning

Before upgrading your MCP cluster, verify the compatibility of different components versions in a specific MCP release. For details, see MCP compatibility matrix.

This section describes the upgrade procedures of the MCP components.

Upgrade DriveTrain to a newer release version

You can upgrade your MCP deployment to a certain MCP release version through DriveTrain using the Deploy - upgrade MCP DriveTrain pipeline. An MCP release version is a stable and product-ready combination of versions of MCP components tagged with a specific Build ID.

This section describes how to upgrade your MCP version from the Build ID 2018.11.0 to 2019.2.0. To upgrade an MCP version with an earlier Build ID, refer to the upgrade paths table in Release Compatibility Matrix: Supported upgrade paths.

Use this procedure to apply minor updates to MCP DriveTrain that will allow for further updates of the MCP components packages to higher minor versions as described in Update an MCP cluster. For a major upgrade of release versions of OpenStack, Kubernetes, and other MCP components, refer to the corresponding sections of Upgrade an MCP cluster.

> **Caution!**
>
> The OpenContrail 4.x packages update is covered in a separate procedure. For details, see: Update the OpenContrail 4.x nodes.
>
> To update the OpenContrail 3.2 packages, refer to the Upgrade the OpenContrail nodes to version 3.2 procedure after you upgrade MCP to a newer release version.

Prerequisites

Before you proceed with the upgrade procedure, complete the following steps:

1. Back up the Salt Master node as described in Back up and restore the Salt Master node.
2. If Jenkins is enabled on the Salt Master node, add the following parameters to ./classes/cluster/<cluster_name>/infra/config/jenkins.yml of your Reclass model:
   - To the _param section:

     ```
     jenkins_master_protocol: http
     ```

   - To the jenkins:client:master section:

     ```
     proto: http
     ```

3. Log in to the Jenkins web UI.
4. Run the following pipeline jobs with BRANCHES set to *:
   - git-mirror-downstream-mk-pipelines
   - git-mirror-downstream-pipeline-library

Caution!

Before you proceed to Upgrade to MCP release version 2019.2.0, verify that these jobs succeed.

Upgrade to MCP release version 2019.2.0

This section describes how to upgrade the MCP release version on your deployment from the Build ID 2018.11.0 to 2019.2.0.

To upgrade to MCP release version 2019.2.0:

1. Verify that you have completed the steps described in Prerequisites.
2. Log in to the Salt Master node.
3. From the /srv/salt/reclass/classes/cluster directory of your Reclass model, verify that the correct Salt formulas and OpenContrail repositories are enabled in your deployment:

   > Note
   >
   > Starting from the 2019.2.0 MCP version, the Salt formulas and OpenContrail repositories are moved from http://apt.mirantis.com to http://mirror.mirantis.com.

   ```
   grep -r --exclude-dir=aptly -l 'system.linux.system.repo.mcp.salt'
   ```

   If the matches are found, replace accordingly on the cluster Reclass level:

   - system.linux.system.repo.mcp.salt replace with system.linux.system.repo.mcp.apt_mirantis.salt-formulas
   - system.linux.system.repo.mcp.updates replace with system.linux.system.repo.mcp.apt_mirantis.update
   - system.linux.system.repo.mcp.contrail replace with system.linux.system.repo.mcp.apt_mirantis.contrail
   - system.linux.system.repo.mcp.extra replace with system.linux.system.repo.mcp.apt_mirantis.extra
4. Depending on your cluster configuration, add the update repositories for the required components. The list of the update repositories include cassandra, ceph, contrail, docker, elastic, extra, kubernetes_extra, openstack, percona, salt-formulas, saltstack, and ubuntu.

   For example, to add the update repository for OpenStack:

   1. Change the directory to /srv/salt/reclass/classes/cluster.
   2. Verify whether the OpenStack component is present in the model:

      ```
      grep -r --exclude-dir=aptly -l 'system.linux.system.repo.mcp.apt_mirantis.openstack'
      ```

   3. If matches are found, include the update repository in your Reclass model by editing the files that include these matches:

```
classes:
- system.linux.system.repo.mcp.apt_mirantis.openstack
- system.linux.system.repo.mcp.apt_mirantis.update.openstack
```

5. Open the project Git repository with your Reclass model on the cluster level.
6. In /infra/backup/client_mysql.yml, verify that the following parameters are defined:

```
parameters:
  xtrabackup:
    client:
      cron: false
```

7. In /infra/backup/server.yml, verify that the following parameters are defined:

```
parameters:
  xtrabackup:
    server:
      cron: false
  # if ceph is enabled
  ceph:
    backup:
      cron: false
```

8. If any physical node in your cluster has LVM physical volumes configured, for example, the root partition, define these volumes in your Reclass model.

   You can verify where LVM is configured using the pvdisplay or lvm pvs command. Apply the following state from the Salt Master node:

```
salt '*' cmd.run 'lvm pvs'
```

   For example, if one of your physical nodes has /dev/sda1 for a physical volume of a volume group vgroot and a logical volume lvroot (/dev/vgroot/lvroot) mounted as /, add the following pillar data for this node:

```
parameters:
  linux:
    storage:
      lvm:
        vgroot:
          enabled: true
          devices: /dev/sda1
```

   For example, if all your compute nodes have LVM physical volume configured, add the above pillar to /openstack/compute/init.yml.

> Warning
>
> You must add the above pillar data for all nodes with all LVM volume groups configured. Otherwise, the LVM configuration will be updated improperly during the upgrade and a node will be unable to boot from the logical volume.

9. If OpenContrail 3.2 is used, verify that the following configurations are present in your Reclass model:

   - In the /infra/backup/client_zookeeper.yml and /infra/backup/server.yml files:

     ```
     parameters:
       zookeeper:
         backup:
           cron: false
     ```

   - In the /infra/backup/client_cassandra.yml and /infra/backup/server.yml files:

     ```
     parameters:
       cassandra:
         backup:
           cron: false
     ```

> Caution!
>
> The OpenContrail 4.x update is covered in a separate procedure. For details, see: Update the OpenContrail 4.x nodes.

10. If OpenStack Telemetry is used, switch Redis to use password authentication:

> Warning
>
> During this procedure, a short Tenant Telemetry downtime occurs.

   1. In /infra/secrets.yml, add a password for Redis:

      ```
      parameters:
        _param:
          openstack_telemetry_redis_password_generated: <very_strong_password>
      ```

- The password can contain uppercase or lowercase letters from the latin alphabet (A-Z) and digits (0-9).
- The recommended password length is 32 characters.

> Warning
>
> Since the key feature of Redis is high performance, an attacker can try many passwords per second. Therefore, create a very strong password to prevent information leak.

2. In /openstack/init.yml, add the following parameter:

```
parameters:
  _param:
    openstack_telemetry_redis_password: ${_param:openstack_telemetry_redis_password_generated}
```

3. In /openstack/telemetry.yml, update the following definitions:

   - Update the openstack_telemetry_redis_url parameter value. For example:

   ```
   parameters
     _param:
       openstack_telemetry_redis_url: redis://openstack:${_param:openstack_telemetry_redis_password}@${_param:redis_sentinel_node01_address}:26379?sentinel=master_1&sentinel_fallback=${_param:redis_sentinel_node02_address}:26379&sentinel_fallback=${_param:redis_sentinel_node03_address}:26379
   ```

   - Add the password parameter to the following section:

   ```
   redis:
     cluster:
       ...
       password: ${_param:openstack_telemetry_redis_password}
       ...
   ```

4. Refresh pillars:

```
salt 'mdb*' saltutil.pillar_refresh
```

5. Apply the changes:

   - For the Redis cluster:

   > Warning
   >
   > After applying the Redis states, the Tenant Telemetry services will not be able to connect to Redis.

```
salt -C 'I@redis:cluster:role:master' state.sls redis
salt -C 'I@redis:server' state.sls redis
```

- For the Tenant Telemetry components:

```
salt -C 'I@gnocchi:server' state.sls gnocchi.server
salt -C 'I@ceilometer:server' state.sls ceilometer.server
salt -C 'I@aodh:server' state.sls aodh.server
```

> Note
>
> Once you apply the Salt states above, Tenant Telemetry will become fully operational again.

11. In /cicd/control/leader.yml, verify that the following class is present:

```
classes:
- system.jenkins.client.job.deploy.update
```

12. Commit the changes to your local repository.
13. Log in to the Jenkins web UI.
14. Verify that you do not have any unapproved scripts in Jenkins:
    1. Navigate to Manage Jenkins > In-process script approval.
    2. Approve pending scripts if any.
15. Run the Deploy - upgrade MCP DriveTrain pipeline in the Jenkins web UI specifying the following parameters as required:

Deploy - upgrade MCP DriveTrain pipeline parameters

Parameter	Description
SALT_MASTER_URL	Salt Master API URL string
SALT_MASTER_CREDENTIALS	Jenkins credentials to access the Salt Master API
BATCH_SIZE Added since 2019.2.6 update	The batch size for Salt commands targeted for a large amount of nodes. Set to an absolute number of nodes (integer) or percentage, for example, 20 or 20%. For details, see MCP Deployment Guide: Configure Salt Master threads and batching.

OS_DIST_UPGRADE For updates starting from 2019.2.2 to 2019.2.8 or later	Applicable only for maintenance updates starting from 2019.2.2 or later to 2019.2.8 and later. Set to true to upgrade the system packages including kernel using apt-get dist-upgrade on the cid* nodes. Note Prior to the maintenance update 2019.2.8, manually specify the parameter in the DRIVE_TRAIN_PARAMS field. For later versions, the parameter is predefined and set to false by default.
OS_UPGRADE For updates starting from 2019.2.2 to 2019.2.8 or later	Applicable only for maintenance updates starting from 2019.2.2 or later to 2019.2.8 and later. Set to true to upgrade all installed applications using apt-get upgrade on the cid* nodes. Note Prior to the maintenance update 2019.2.8, manually specify the parameter in the DRIVE_TRAIN_PARAMS field. For later versions, the parameter is predefined and set to false by default.
APPLY_MODEL_WORKAROUNDS For maintenance updates to 2019.2.8 or later	Applicable only for maintenance updates to 2019.2.8 and later. Recommended. Select to apply the cluster model workarounds automatically unless you manually added some patches to the model before the update.
UPGRADE_SALTSTACK	Upgrade the SaltStack packages (salt-master, salt-api, salt-common) on the Salt Master node and the salt-minion package on all nodes.
UPDATE_CLUSTER_MODEL	Automatically apply the cluster model changes required for the target MCP version: 1. Replace the mcp_version parameter with TARGET_MCP_VERSION on the cluster level of Reclass model 2. Apply other backward incompatible cluster model updates
UPDATE_PIPELINES	Automatically update the pipeline-library and mk-pipelines repositories from upstream/local mirror into Gerrit

UPDATE_LOCAL_REPOS Deprecated since MCP 2019.2.0	Update local repositories on the Aptly node if applicable.
GIT_REFSPEC	Git version of the Reclass system to use (branch or tag). Must match TARGET_MCP_VERSION.
MK_PIPELINES_REFSPEC	Git version of mk/mk-pipelines to use (branch or tag). Must match TARGET_MCP_VERSION.
PIPELINE_TIMEOUT	The time for the Jenkins job to complete, set to 12 hours by default. If the time is exceeded, the Jenkins job will be aborted.

TARGET_MCP_VERSION	Target version of MCP that will correspond to the mcp_version parameter value in your Reclass model. Select from the following options:
	• If you upgrade DriveTrain to the latest major version, set the corresponding latest available Build ID.
	• To apply maintenance updates to a major MCP release version, select from the following options:
	  • Specify your current MCP version. For example, if your current MCP version is 2019.2.0, set 2019.2.0.
	  • Starting from 2019.2.8, alternatively, set the TARGET_MCP_VERSION, MK_PIPELINES_REFSPEC, and GIT_REFSPEC parameters to the target maintenance update version. For example, to update from 2019.2.7 to 2019.2.8, specify 2019.2.8.

The pipeline workflow:

1. All required updates and fixes for upgrade are applied.
2. Packages for Salt formulas and Reclass are updated, cluster model is updated.
3. The following DriveTrain components are updated:
   - Local repositories if needed
   - Salt Master and minions
   - Jenkins
   - Docker

16. Optional. To upgrade system packages on the Salt Master node, select from the following options:

   - If you are upgrading DriveTrain to the latest major version, run the following commands one by one from the cfg01 node:

```
apt-get update
apt-get dist-upgrade
```

- If you are applying maintenance updates to a major MCP version, run the following commands one by one from the cfg01 node:

  ```
  apt-get update
  apt-get upgrade
  ```

17. Proceed to Upgrade GlusterFS.

Upgrade GlusterFS

> Note
>
> This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

This section describes how to upgrade GlusterFS on your deployment from version 3.8 to 5.5 using three dedicated pipeline jobs.

If you do not have any services that run on top of the GlusterFS volumes except the Docker Swarm services such as Jenkins, Gerrit, LDAP, you can use the all-in-one Update GlusterFS Jenkins pipeline job that consequently executes three dedicated pipeline jobs described below with the default parameters.

Otherwise, if you do have any services that run on top of the GlusterFS volumes except the Docker Swarm services, Mirantis recommends updating the GlusterFS components separately using the granular pipeline jobs to better control the upgrade process.

> Note
>
> This procedure does not include the backup of data located on the GlusterFS volumes.

To upgrade GlusterFS to version 5.5:

1. Log in to the Salt Master node.
2. In infra/init.yml of your Reclass model, add the following parameter:

   ```
   parameters:
     _param:
       linux_system_repo_mcp_glusterfs_version_number: 5
   ```

3. Apply the following state:

   ```
   salt '*' state.apply linux.system.repo
   ```

4. Log in to your MCP cluster Jenkins web UI.
5. Verify that you do not have any unapproved scripts in Jenkins:

   1. Navigate to Manage Jenkins > In-process script approval.
   2. Approve pending scripts if any.
6. Upgrade GlusterFS on servers by running the Update glusterfs servers pipeline job with the following parameters as required:

Update glusterfs servers pipeline job parameters

Parameter	Description and values
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
TARGET_SERVERS	Salt compound target to match the nodes to be upgraded. For example, G@osfamily:debian or *. Defaults to I@glusterfs:server.
IGNORE_SERVER_STATUS	Not recommended. Select not to validate the GlusterFS server availability before the upgrade. If some servers are unavailable, then data on a volume may be unavailable during the upgrade.
IGNORE_NON_REPLICATED_VOLUMES	Not recommended. Select to upgrade GlusterFS even with a non-replicated volume. You may lose data on the non-replicated volumes during the upgrade. Therefore, if you have such volumes, Mirantis recommends stopping them before the upgrade.

The pipeline job workflow:

1. Select only the TARGET_SERVERS hosts on which the new GlusterFS packages are available and not installed.
2. Verify that all TARGET_SERVERS are available and connected to the cluster. This step is skipped if the IGNORE_SERVER_STATUS parameter is set to true.
3. Verify that all GlusterFS volumes are replicated. This step is skipped if the IGNORE_NON_REPLICATED_VOLUMES parameter is set to true.
4. Upgrade the GlusterFS packages on each target host one by one.

7. Upgrade GlusterFS on clients by running the Update glusterfs clients pipeline job with the following parameters as required:

> **Caution!**
>
> Except for the Docker Swarm services such as Jenkins, Gerrit, LDAP, the pipeline job ignores any other services that use the GlusterFS volumes. Therefore, Mirantis recommends stopping any not Docker Swarm-related services that use the GlusterFS volumes before running the pipeline job and upgrade clients one by one by targeting the global name of a client using the TARGET_SERVERS parameter.

Note

During the upgrade of the GlusterFS clients, all infrastructure services such as Jenkins, Gerrit, LDAP are unavailable for several minutes.

Update glusterfs clients pipeline job parameters

Parameter	Description and values
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
TARGET_SERVERS	Salt compound target to match the nodes to be upgraded. For example, G@osfamily:debian or *. Defaults to I@glusterfs:client.
IGNORE_SERVER_STATUS	Not recommended. Select not to validate the GlusterFS server availability before the upgrade. If some servers are unavailable, then data on volumes may be unavailable during the upgrade. Mirantis recommends verifying that the cluster is fully functional and healthy before proceeding with the upgrade.
IGNORE_SERVER_VERSION	Not recommended. Select not to validate that all GlusterFS servers are upgraded to the same version. Mirantis highly recommends upgrading all GlusterFS servers before you proceed to upgrading the GlusterFS clients.

The pipeline job workflow:

1. Select only the TARGET_SERVERS hosts on which the new GlusterFS packages are available and not installed.
2. Verify that all GlusterFS servers are available and connected to the cluster. This step is skipped if the IGNORE_SERVER_STATUS parameter is set to true.
3. Verify that all GlusterFS servers have the same package version that will be installed on the GlusterFS clients. This step is skipped if the IGNORE_SERVER_VERSION parameter is set to true.
4. Upgrade the GlusterFS package and re-mount the volumes on each host one by one.

8. Upgrade the GlusterFS cluster.op-version option to verify that all GlusterFS servers and clients use the updated protocol. Run the Update glusterfs cluster.op-version pipeline job with the following parameters as required:

Update glusterfs cluster.op-version pipeline job parameters

Parameter	Description and values
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
CLUSTER_OP_VERSION	Leave empty to use the cluster.max-op-version option value defined in GlusterFS. Otherwise, specify the value for the GlusterFS cluster.op-version option to use.
IGNORE_SERVER_VERSION	Not recommended. Select not to validate that all GlusterFS servers are upgraded.
IGNORE_CLIENT_VERSION	Not recommended. Select not to validate that all GlusterFS servers are upgraded.

The pipeline job workflow:

1. If CLUSTER_OP_VERSION is empty, obtain the cluster.max-op-version option value from GlusterFS.
2. Obtain the current cluster.op-version and compare it the CLUSTER_OP_VERSION parameter value to identify whether a version update is required.
3. Verify that all GlusterFS servers are upgraded to the version equal or above the one defined in CLUSTER_OP_VERSION. This step is skipped if the IGNORE_SERVER_VERSION parameter is set to true.
4. Verify that all GlusterFS clients are upgraded to the version equal or above the one defined in CLUSTER_OP_VERSION. This step is skipped if the IGNORE_CLIENT_VERSION parameter is set to true.
5. Update cluster.op-version to the value defined in CLUSTER_OP_VERSION.

9. Optional. Recommended. Configure allowed and rejected IP addresses for the GlusterFS volumes.

Seealso

Update GlusterFS

Rollback MCP to a previous release version

You can rollback your MCP deployment to a previous MCP release version through DriveTrain using the Deploy - update cloud pipeline.

To rollback to the previous stable MCP release version:

1. Verify that the correct previous Build ID release repositories are available. These include the local repositories present in the mirror image backup as well as the local aptly repositories.
2. In the infra/init.yml file of the Reclass model, specify the previous Build ID in the mcp_version parameter.
3. In the infra/init.yml file of the Reclass model, verify that the following pillar is present.

   ```
   parameters:
     linux:
       system:
         purge_repos: true
   ```

4. Update the classes/system Git submodule of the Reclass model to the commit of the required Build ID by running the following command from the classes/system directory:

   ```
   git pull origin release/BUILD_ID
   ```

5. Commit and push the changes to the Git repository where the cluster Reclass model is located.
6. Select from the following options:

   - The ROLLBACK_BY_REDEPLOY was not selected for the update pipeline:

     1. Roll back the Salt Master node:

        1. On the KVM node hosting the Salt Master node, run the following commands:

           ```
           virsh destroy cfg01.domain
           virsh define /var/lib/libvirt/images/cfg01.domain.xml
           virsh start cfg01.domain; virsh snapshot-delete cfg0X.domain --metadata ${SNAPSHOT_NAME}
           rm /var/lib/libvirt/images/cfg01.domain.${SNAPSHOT_NAME}.qcow2
           rm /var/lib/libvirt/images/cfg01.domain.xml
           ```

        2. On the Salt Master node, apply the linux.system.repo Salt state.
     2. Roll back the CI/CD nodes:

        1. On the KVM nodes hosting the CI/CD nodes, run the following commands:

           ```
           virsh destroy cid0X.domain
           virsh define /var/lib/libvirt/images/cid0X.domain.xml
           virsh start cid0X.domain
           virsh snapshot-delete cid0X.domain --metadata ${SNAPSHOT_NAME}
           ```

```
rm /var/lib/libvirt/images/cid0X.domain.${SNAPSHOT_NAME}.qcow2
rm /var/lib/libvirt/images/cid0X.domain.xml
```

2. On all CI/CD nodes, restart the docker service and apply the linux.system.repo Salt state.

- The ROLLBACK_BY_REDEPLOY parameter was selected for the update pipeline:

    1. Roll back the Salt Master node as described in Back up and restore the Salt Master node.
    2. Redeploy the CI/CD nodes:

        1. Virsh destroy and undefine the cid nodes:

        ```
        virsh destroy cid<NUM>.<DOMAIN_NAME>
        virsh undefine cid<NUM>.<DOMAIN_NAME>
        ```

        2. Remove the cid salt-keys from the Salt Master node.
        3. Redeploy the CI/CD nodes using the Deploy CI/CD procedure.

7. Run the Deploy - update cloud pipeline in the Jenkins web UI specifying the following parameters as required:

Deploy - update cloud pipeline parameters

Parameter	Description
CEPH_OSD_TARGET	The Salt targeted physical Ceph OSD osd nodes.
CID_TARGET	The Salt targeted CI/CD cid nodes.
CMN_TARGET	The Salt targeted Ceph monitor cmn nodes.
CMP_TARGET	The Salt targeted physical compute cmp nodes.
CTL_TARGET	The Salt targeted controller ctl nodes.
DBS_TARGET	The Salt targeted database dbs nodes.
GTW_TARGET	The Salt targeted physical or virtual gateway gtw nodes.
INTERACTIVE	Ask interactive questions during the pipeline run. If not selected, the pipeline will either succeed or fail.
KVM_TARGET	The Salt targeted physical KVM kvm nodes.
LOG_TARGET	The Salt targeted log storage and visualization log nodes.
MON_TARGET	The Salt targeted StackLight LMA monitoring node mon nodes.
MSG_TARGET	The Salt targeted RabbitMQ server msg nodes.
MTR_TARGET	The Salt targeted StackLight LMA metering mtr nodes.
NAL_TARGET	The Salt targeted OpenContrail 3.2 analytics nal nodes.

NTW_TARGET	The Salt targeted OpenContrail 3.2 controller ntw nodes.
PER_NODE	Target nodes will be managed one by one. Recommended.
PRX_TARGET	The Salt targeted proxy prx nodes.
RGW_TARGET	The Salt targeted RADOS gateway rgw nodes.
ROLLBACK_BY_REDEPLOY	Select if live snapshots were taken during update.
SALT_MASTER_URL	URL of Salt Master node API.
SALT_MASTER_CREDENTIALS	ID of the Salt Master node API credentials stored in Jenkins.
SNAPSHOT_NAME	Live snapshot name.
STOP_SERVICES	Stop API services before the rollback.
PURGE_PKGS	The space-separated list of pkgs=versions to be purged on the physical targeted machines. For example, pkg_name1=pkg_version1 pkg_name2=pkg_version2.
REMOVE_PKGS	The space-separated list of pkgs=versions to be removed on the physical targeted machines. For example, pkg_name1=pkg_version1 pkg_name2=pkg_version2.

<table>
<tr>
<td>RESTORE_CONTRAIL_DB</td>
<td>
Restore the Cassandra and ZooKeeper databases for OpenContrail 3.2. OpenContrail 4.x is not supported. Select only if rollback of the OpenContrail 3.2 controller nodes failed or a specific backup defined in the cluster model is required.
If RESTORE_CONTRAIL_DB is selected, add the following configuration to the cluster/opencontrail/control.yml file of your Reclass model:

For ZooKeeper:

<pre><code>parameters:
  zookeeper:
    backup:
      client:
        enabled: true
        restore_latest: 1
        restore_from: remote
</code></pre>

For Cassandra:

<pre><code>parameters:
  cassandra:
    backup:
      client:
        enabled: true
        restore_latest: 1
        restore_from: remote
</code></pre>

</td>
</tr>
<tr>
<td>RESTORE_GALERA</td>
<td>
Restore the Galera database. Select only if the rollback of the database nodes failed or a specific backup defined in the cluster model is required.
If RESTORE_GALERA is selected, add the xtrabackup restore lines to the cluster/openstack/database/init.yml of your Reclass model:

<pre><code>parameters:
  xtrabackup:
    client:
      enabled: true
      restore_full_latest: 1
      restore_from: remote
</code></pre>
</td>
</tr>
</table>

ROLLBACK_PKG_VERSIONS	Copy back the list of package versions installed before the update acquired from the pipeline before pkgs were upgraded. The space-separated list of pkgs=versions to roll back to on the physical targeted machines. For example, pkg_name1=pkg_version1 pkg_name2=pkg_version2. If ROLLBACK_PKG_VERSIONS is empty, apt --allow-downgrades dist-upgrade will be run on the targeted physical machines. If ROLLBACK_PKG_VERSIONS contains the salt-minion package, you will have to rerun the pipeline for every targeted physical machine as it will be disconnected.
TARGET_ROLLBACKS	The comma-separated list of nodes to rollback. The valid values include ctl, prx, msg, dbs, log, mon, mtr, ntw, nal, gtw-virtual, cmn, rgw, cmp, kvm, osd, gtw-physical.
TARGET_REBOOT	The comma-separated list of physical nodes to reboot after a rollback. The valid values include cmp, kvm, osd, gtw-physical. **Caution!** When the kvm node is defined, the pipeline can be interrupted due to the Jenkins slave reboot. If so, remove the already updated nodes from TARGET_UPDATES and rerun the pipeline.
TARGET_HIGHSTATE	The comma-separated list of physical nodes to run Salt highstate on after a rollback. The valid values include cmp, kvm, osd, gtw-physical.

Common rollback workflow for different nodes types:

1. Downtime for VCP occurs if a rollback for VCP VMs is required.
2. If the ROLLBACK_BY_REDEPLOY parameter is selected, the VCP VMs will be destroyed, undefined, and their salt-key will be deleted from the Salt Master node.
3. Verification of the service or API status is done.

The procedure of the pipeline if ROLLBACK_BY_REDEPLOY is not selected:

1. VCP VMs by their target type are destroyed.
2. Live snapshot is deleted if any and its original base file is used to boot the VM.
3. Repositories are updated.
4. Physical machines are rolled back:
    1. Repositories are updated.
    2. Packages defined in PURGE_PKGS are purged.

3. Packages defined in REMOVE_PKGS are removed.
4. Package versions defined in ROLLBACK_PKG_VERSIONS are installed.
5. The Salt highstate is applied.

8. Verify that the following lines are not present in cluster/infra/backup/client_mysql.yml:

```
parameters:
  xtrabackup:
    client:
      cron: false
```

9. Verify that the following lines are not present in cluster/infra/backup/server.yml:

```
parameters:
  xtrabackup:
    server:
      cron: false
```

10. If OpenContrail 3.2 is used:

    1. Verify that the following lines are not present in cluster/infra/backup/client_zookeeper.yml and cluster/infra/backup/server.yml:

    ```
    parameters:
      zookeeper:
        backup:
          cron: false
    ```

    2. Verify that the following lines are not present in cluster/infra/backup/client_cassandra.yml and cluster/infra/backup/server.yml:

    ```
    parameters:
      cassandra:
        backup:
          cron: false
    ```

11. If the ROLLBACK_BY_REDEPLOY parameter was selected and the Deploy - cloud update pipeline succeeds, continue the rollback:

    1. Roll back the Salt Master node as described in Back up and restore the Salt Master node procedure if necessary.
    2. If Ceph is enabled and you want to rollback Ceph monitoring nodes, proceed with Restore a Ceph Monitor node.
    3. Redeploy the nodes based on your model by running the Deploy - OpenStack pipeline. See the pipeline configuration details in Deploy an OpenStack environment, step 10.

> Note
>
> Specify k8s in the Install parameter field in case of an MCP Kubernetes deployment.

4. Rerun the Deploy - cloud update pipeline with RESTORE_GALERA and RESTORE_CONTRAIL_DB selected.

Upgrade the OpenContrail nodes from version 3.2 to 4.1

> **Caution!**
>
> Before proceeding with the upgrade procedure, verify that you have updated DriveTrain including Aptly, Gerrit, Jenkins, Reclass, Salt formulas, and their subcomponents to the current MCP release version. Otherwise, the current MCP product documentation is not applicable to your MCP deployment.

This section describes how to upgrade the OpenContrail nodes of an Ocata- or Pike-based MCP cluster to version 4.1 using the Deploy - upgrade Opencontrail to 4.x pipeline. To update OpenContrail from version 4.0 to 4.1, refer to Update the OpenContrail 4.x nodes.

> Note
>
> Upgrade the OpenContrail cluster before the OpenStack upgrade.

> Warning
>
> The OpenContrail data plane traffic will be affected during the upgrade procedure since restarting of a vRouter agent leads to connectivity interruption. Plan migrations of the critical workloads accordingly before the vRouter agent restart.

The OpenContrail upgrade to version 4.1 is available only from version 3.2. Therefore, if you have an older OpenContrail version, for example, 3.2.3 or 3.1.1, first update the OpenContrail packages on the OpenContrail nodes to the latest supported 3.2.x version using the Update the OpenContrail 3.2 package procedure.

The high-level workflow of the OpenContrail upgrade pipeline job is as follows:

1. Verify the OpenContrail services statuses.
2. Stop the Neutron API that will be unavailable during the whole upgrade procedure.
3. Back up the Cassandra and ZooKeeper data.
4. Stop all running OpenContrail analytics services.
5. Upgrade or update the OpenContrail analytics nodes simultaneously. During upgrade, new Docker containers for the OpenContrail analytics nodes are spawned. During update, the corresponding Docker images are updated.

©2025, Mirantis Inc.

6. Upgrade or update the OpenContrail controller nodes. During upgrade, new Docker containers for the OpenContrail controller nodes are spawned. During update, the corresponding Docker images are updated. All nodes are upgraded or updated simultaneously except the one that meantime runs the contrail-control service and is upgraded or updated after other nodes.
7. Upgrade the OpenContrail packages on the OpenStack controller nodes simultaneously.
8. Start the Neutron server services.
9. Upgrade the OpenContrail data plane nodes one by one with the workloads migration if needed since this step implies downtime of the Networking service.
10. Verify the OpenContrail services statuses.

Prerequisites

Before you start upgrading the OpenContrail nodes of your MCP cluster, complete the following prerequisite steps:

1. Configure the server and client roles for Cassandra and ZooKeeper as described in OpenContrail 3.2: Create a backup schedule for a Cassandra database and OpenContrail 3.2: Create a backup schedule for a ZooKeeper database.
2. If you are going to upgrade the compute nodes after upgrading the OpenContrail controller nodes, prepare the compute nodes for upgrade. Since the discovery service is removed in OpenContrail starting from v4.0, configure the endpoints statically:

   1. In cluster/<name>/opencontrail/compute.yml, add the system.opencontrail.compute.upgrade class under the system.opencontrail.compute.cluster:

      ```
      classes:
      ...
      - system.opencontrail.compute.cluster
      - system.opencontrail.compute.upgrade
      ...
      ```

   2. Apply the opencontrail.compute state on the compute nodes.
   3. In cluster/<name>/opencontrail/compute.yml, remove the system.opencontrail.compute.upgrade class.

Now, proceed to Prepare the cluster model.

Prepare the cluster model

After you complete the prerequisite steps, prepare your cluster model for the upgrade by configuring your Git project repository as described below.

To prepare the cluster model:

1. Log in to the Salt Master node.
2. In cluster/<name>/opencontrail/init.yml, change the OpenContrail repository component and version to 4.1:

```
_param:
  linux_repo_contrail_component: oc41
  opencontrail_version: 4.1
```

3. In cluster/<name>/openstack/dashboard.yml, add or update the OpenContrail version inside the parameters section:

```
_param:
  opencontrail_version: 4.1
```

4. In cluster/<name>/openstack/init.yml, define the following parameters:

```
_param:
  opencontrail_admin_password: <contrail_user_password>
  opencontrail_admin_user: 'contrail'
```

5. In cluster/<name>/openstack/control/init.yml, add the following class if not present:

```
classes:
...
- system.keystone.client.service.contrail
...
```

6. In cluster/<name>/opencontrail/analytics.yml, change the system.opencontrail.control.analytics class to system.opencontrail.control.analytics4_0.
7. In cluster/<name>/opencontrail/analytics.yml, add or change the opencontrail_kafka_config_dir and opencontrail_kafka_log_dir parameters. For example:

```
_param:
  opencontrail_kafka_config_dir: '/etc/kafka'
  opencontrail_kafka_log_dir: '/var/log/kafka'
```

8. In cluster/<name>/opencontrail/control.yml, change the system.opencontrail.control.control class to system.opencontrail.control.control4_0.
9. In cluster/<name>/opencontrail/analytics.yml and cluster/<name>/opencontrail/control.yml, modify the classes and the parameters sections:

```
classes:
...
- system.linux.system.repo.docker_legacy
...
- service.docker.host
- system.docker.client.compose
...
parameters:
  ...
  docker:
    host:
      pkgs:
        - docker-engine
        - python-docker
```

10. In cluster/<name>/opencontrail/control.yml and cluster/<name>/opencontrail/analytics.yml, remove the following classes:

```
classes:
...
- system.linux.system.repo.mcp.apt_mirantis.contrail
...
- system.linux.system.repo.mcp.apt_mirantis.cassandra
...
```

> Note
>
> These classes are related to the package repositories of OpenContrail and Cassandra and must be removed since the new version of the OpenContrail control plane is delivered as Docker containers.

11. In cluster/<name>/opencontrail/compute.yml, change the system.opencontrail.compute.cluster class to system.opencontrail.compute.cluster4_0.

12. Add the upgrade pipeline to DriveTrain:

    1. Add the OpenContrail upgrade class to cluster/cicd/control/leader.yml:

    ```
    classes:
    - system.jenkins.client.job.deploy.update.upgrade_opencontrail4_0
    ```

    2. Apply the changes:

    ```
    salt -C 'I@jenkins:client' state.sls jenkins.client
    ```

Now, proceed to Upgrade the OpenContrail nodes.

Upgrade the OpenContrail nodes

After you prepare the cluster model of your MCP cluster, proceed to upgrading the OpenContrail controller nodes and the OpenContrail vRouter packages on the compute nodes.

> **Warning**
>
> During the upgrade process, the following resources are affected:
>
> - The instance(s) running on the compute nodes can be unavailable for up to 30 seconds.
> - The creation of new instances is not possible during the same time interval.
>
> Therefore, you must plan a maintenance window as well as test the upgrade before applying it to production.

To upgrade the OpenContrail nodes:

1. Log in to the Jenkins web UI.
2. Open the Deploy - upgrade Opencontrail to 4.x pipeline.
3. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
STAGE_CONTROLLERS_UPGRADE	Select this check box to run upgrade on the OpenContrail controller nodes.

4. Click Deploy.
5. If the pipeline fails with a Docker container with contrail-control service not starting or infinite restarting:

   1. Log in to the container of the affected OpenContrail node.
   2. Verify whether the logs contain errors UID already exists or GID already exists. For example:

      ```
      docker logs --tail 10 --follow --timestamps <container_ID>
      ```

      Example of system response extract:

```
2018-09-13T06:52:15.569311246Z usermod: UID '109' already exists
2018-09-13T06:53:15.714037288Z usermod: UID '109' already exists
2018-09-13T06:54:15.863864322Z usermod: UID '109' already exists
2018-09-13T06:55:16.026689446Z usermod: UID '109' already exists
```

3. If the logs contain the above-mentioned errors:

    1. Log in to one of the affected OpenContrail nodes.
    2. Replace the contrail GID and UID with new ones that are not used by existing group and user. For example, replace 109 with 901. Run the following commands:

    ```
    usermod -u <new uid> contrail
    groupmod -g <new gid> contrail
    ```

    3. Rerun the Deploy - upgrade Opencontrail to 4.x pipeline.

6. If the pipeline fails with some OpenContrail services stuck in the initializing state:

    1. Restart ZooKeeper on all analyticsdb containers.
    2. Once ZooKeeper is up and running, restart the services that are stuck.
    3. Rerun the Deploy - upgrade Opencontrail to 4.x pipeline.

7. If the pipeline fails with the Command ‘docker-compose up -d’ failed error message that causes docker-compose fail to initialize the opencontrail_analytics_1 and opencontrail_analyticsdb_1 containers:

    1. Run the following command from the Salt Master node:

    ```
    salt -C 'nal*' cmd.run "rm -rf /etc/cassandra/cassandra_analytics.yaml \
    && rm -rf /etc/cassandra/cassandra-env-analytics.sh/ \
    && rm -rf /etc/zookeeper/conf/zoo_analytics.cfg"
    ```

    2. Rerun the Deploy - upgrade Opencontrail to 4.x pipeline.

To upgrade the OpenContrail vRouter packages on the compute nodes:

1. Log in to the Jenkins web UI.
2. Open the Deploy - upgrade Opencontrail to 4.x pipeline.
3. Specify the following parameters:

Parameter	Description and values
COMPUTE_TARGET_SERVERS	Add I@opencontrail:compute or target the global name of your compute nodes, for example cmp001*.

COMPUTE_TARGET_SUBSET_LIVE	Add 1 to run the upgrade first on only one of the nodes defined in the COMPUTE_TARGET_SERVERS field. After this stage is done, in the upgrade pipeline you will be asked to continue the upgrade all nodes defined in the COMPUTE_TARGET_SERVERS field.
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
STAGE_COMPUTES_UPGRADE	Select this check box to run upgrade on the compute nodes.

4. Click Deploy. For details how to monitor the deployment process, see: MCP Deployment Guide: View the deployment details.

The Deploy - upgrade Opencontrail to 4.x pipeline upgrade stages are as follows:

1. If STAGE_COMPUTES_UPGRADE is selected, the pipeline upgrades the OpenContrail packages on the compute nodes in two iterations:

    1. Upgrade the sample nodes defined by COMPUTE_TARGET_SUBSET_LIVE.
    2. After a manual confirmation, upgrade all compute nodes targeted by COMPUTE_TARGET_SERVERS.

2. If STAGE_CONTROLLERS_UPGRADE is selected, the pipeline stops the OpenContrail services on the OpenContrail nal nodes and starts the analytics and analytics db containers. Once done, the same procedure applies to the OpenContrail ntw nodes.

Seealso

Roll back the OpenContrail nodes

Roll back the OpenContrail nodes

You can roll back the OpenContrail nodes and the OpenContrail packages on the compute nodes if the upgrade fails. You can find a full log of the specific upgrade build using Build History > Console Output > Full Log in Jenkins UI.

To roll back the OpenContrail nodes:

1. Log in to the Salt Master node.
2. Revert the changes made in Prepare the cluster model section.
3. If you are rolling back only the compute nodes while having the OpenContrail controller nodes already upgraded to v4.x, add the system.opencontrail.compute.upgrade class to cluster/<name>/opencontrail/compute.yml.
4. Log in to the Jenkins web UI.
5. Open the Deploy - upgrade Opencontrail to 4.x pipeline.
6. Specify the following parameters as required:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
COMPUTE_TARGET_SERVERS	Add the same string that you used during the upgrade that failed.
COMPUTE_TARGET_SUBSET_LIVE	Add the same string that you used during the upgrade that failed.
STAGE_CONTROLLERS_ROLLBACK	Select this check box to roll back the OpenContrail controller nodes.
STAGE_COMPUTES_ROLLBACK	Select this check box to roll back the compute nodes.

7. Click Deploy. For details on how to monitor the deployment process, see: MCP Deployment Guide: View the deployment details.

The Deploy - upgrade Opencontrail to 4.x pipeline rollback stages are as follows:

1. If STAGE_CONTROLLERS_ROLLBACK is selected, the pipeline stops the OpenContrail containers with the ntw and nal nodes and starts the OpenContrail services. The process requires manual confirmations that are based on the output of the nodetool status and contrail-status commands.
2. If STAGE_COMPUTES_ROLLBACK is selected, the pipeline downgrades the OpenContrail packages on the compute nodes in two iterations:
    1. Downgrade the sample compute nodes defined by COMPUTE_TARGET_SUBSET_LIVE.

2. After a manual confirmation, downgrade all compute nodes defined by COMPUTE_TARGET_SERVERS.

Upgrade an MCP OpenStack environment

This section provides the reference information to consider when creating a detailed maintenance plan for the upgrade of an OpenStack Ocata environment with OVS networking to the OpenStack Pike release as well as an OpenStack Pike environment with OVS networking to the OpenStack Queens release.

> **Caution!**
>
> - If you run an OpenContrail-based OpenStack environment, follow the procedures in this section to upgrade OpenStack only and skip the OVS-related steps. Once done, proceed with upgrading or updating OpenContrail.
> - To upgrade OpenContrail from version 3.2 to 4.1, use a separate procedure described in Upgrade the OpenContrail nodes from version 3.2 to 4.1.
> - To update OpenContrail from version 4.0 to 4.1, refer to Update the OpenContrail 4.x nodes.

Use the descriptive analysis of the techniques and tools, as well as the high-level upgrade flow included in this section to create a cloud-specific detailed upgrade procedure, assess the risks, estimate possible downtimes, plan the rollback, backup, and testing activities.

> **Caution!**
>
> We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.

> **Note**
>
> This section does not cover the OpenStack packages update procedure. To perform the OpenStack packages update, see Update OpenStack packages.

> **Caution!**
>
> Manila deprecation notice

In the MCP 2019.2.7 update, the OpenStack Manila component is being considered for deprecation. The corresponding capabilities are still available, although not further enhanced.

Starting with the 2019.2.11 maintenance update, the OpenStack Manila component will no longer be supported by Mirantis. For those existing customers who have the Manila functionality explicitly included in the scope of their contracts, Mirantis will continue to fulfill the corresponding support obligations.

OpenStack upgrade levels

The depth of the upgrade can differ depending on a use case. The following table describes the possible upgrade levels.

Levels of upgrade

Level	Description
Application upgrade	Only application packages are upgraded. Includes upgrade of application dependencies to the desired level.
System packages upgrade	Known as apt-get upgrade. Used to install the newest versions of all currently installed packages on the system from the sources enumerated in /etc/apt/sources.list. The packages are retrieved and upgraded. Under no circumstances, the packages are removed. The packages that are not yet installed but required are retrieved and installed. The new versions of the currently installed packages that cannot be upgraded without changing the installation status of another package or packages are left at their current version. An update must be performed first, so that apt-get can know that the new versions of the packages are available.
Kernel upgrade	Known as apt-get dist-upgrade. In addition to performing the function of upgrade, handles the changing dependencies with new versions of packages. The apt-get tool has a smart conflict resolution system and attempts to upgrade the most important packages at the expense of less important ones if necessary. The dist-upgrade command can remove some packages. The /etc/apt/sources.list file contains the list of locations from which the desired package files should be retreived. Reboot might be needed after this type of upgrade.
Release upgrade	Known as a do-release-upgrade upgrade of a Linux distribution to a newer major release.

Note

To minimize the control plane downtime, we recommend performing the application level upgrade first. When the OpenStack component is fully upgraded, proceed with the system upgrade. For details, see The OpenStack formulas structure.

OpenStack upgrade tools overview

This section provides the overview of the tools used during the OpenStack upgrade.

MCP Drivetrain provides the following OpenStack-related upgrade pipelines:

- Deploy - upgrade control VMs
- Deploy - upgrade computes
- Deploy - upgrade OVS gateway

Each job consists of different stages that are described in details in the related sections below.

The control plane upgrade pipeline

The Deploy - upgrade control VMs pipeline job is designed to upgrade the control plane component of OpenStack. The workflow of the job includes the following stages:

Deploy - upgrade control VMs pipeline job workflow

Stage	Description
Pre-upgrade	Only non destructive actions are applied during this phase. Basic API verification is performed. The job is launched on all target servers before moving to the next stage. Online dbsyncs is called explicitly to verify whether the actual upgrade can be initiated. **Caution!** Online database synchronization is required before the upgrade to a new release. Since dbsyncs can take a lot of time, the manual online synchronization before the upgrade is recommended.
Stop OpenStack services	Stops all OpenStack Python services on the target servers. This does not affect the data plane services such as OVS or KVM.
Upgrade OpenStack	Upgrades the OpenStack Python code on the target nodes sequentially. Once the stage is finalized on a specific target node, the basic API checks are performed. No workload downtime is expected.
Upgrade OS	Launches only if OS_UPGRADE or OS_DIST_UPGRADE is checked. A reboot can be performed if required. When node is back online, the basic service checks are performed.

The data plane upgrade pipelines

The Deploy - upgrade OVS gateway and Deploy - upgrade computes pipeline jobs are designed to upgrade the data plane nodes of OpenStack. The resources running on the gateway nodes are smoothly migrated to other gateway nodes to minimize workload downtime. The workflow of these jobs include the following stages:

Deploy - upgrade OVS gateway and Deploy - upgrade computes pipeline jobs workflow

Stage	Description
Pre-upgrade	Ensures that Neutron agents and compute services on the target hosts are alive. Only non destructive actions are applied during this stage.
Upgrade pre: migrate resources	Performs the smooth resource migration to minimize the workload downtime. Neutron agents on node are set to the admin_disabled state to ensure their quick migration to new nodes.
Upgrade OpenStack	Upgrades the OpenStack python code. Once completed, the basic API verification is performed. Before proceeding to the next stage, wait for the agents to become alive.
Upgrade OS	Launches only if OS_UPGRADE or OS_DIST_UPGRADE is checked. Reboot can be performed if required. When the node is back online, the basic service checks are performed.
Upgrade post: enable resources	Verifies that agents and services on nodes are up and adds them back to scheduling.
Post upgrade	Performs the post upgrade cleanup of old configurations and temporary files. Only non destructive actions are applied during this phase.

The OpenStack formulas structure

This section describes the structure of the OpenStack formulas related to upgrades. This defines the upgrade API for the formulas that consist of the states described in the table below. By using these states, you can build a flexible upgrade logic for a particular use-case.

The OpenStack formulas structure

State	Description
<app>.upgrade.service_running	Verify that all services for a particular application are enabled for autostart and running.
<app>.upgrade.service_stopped	Verify that all services for a particular application are disabled for autostart and dead.
<app>.upgrade.pkg_latest	Verify that the packages used by a particular application are upgraded to the latest available version. This will not upgrade the data plane packages like QEMU and OVS since usually a minimal required version in OpenStack services is really old. The data plane packages should be upgraded separately by apt-get upgrade or apt-get dist-upgrade. Application of this state will not autostart service.
<app>.upgrade.render_config	Verify that the configuration is rendered to an actual version.
<app>.upgrade.pre	We assume this state is applied on all nodes in the cloud before running the upgrade. Only non-destructive actions will be applied during this phase. Perform the built-in service check such as keystone-manage doctor and nova-status upgrade.
<app>.upgrade.upgrade.pre	Mostly applicable for the data plane nodes. During this phase, resources will be gracefully removed from the current node if it is possible. The services of the upgraded applications will be set to the admin disabled state to make sure that a node will not participate in the resources scheduling. For example on the gtw nodes, this will set all agents to the admin disabled state and move all routers to other agents.
<app>.upgrade.upgrade	Upgrade applications on a particular target node. Stop services, render configuration, install new packages, run offline dbsync for the ctl nodes, start services. The data plane should not be affected, only OpenStack Python services.
<app>.upgrade.upgrade.post	Add services back to scheduling.
<app>.upgrade.post	This phase should be launched only when the cloud upgrade is completed. During this phase, some services may be restarted and minimal downtime may occur.
<app>.upgrade.verify	Perform the basic health checks such as the API CRUD operations, verification of the failed network agents and compute services.

The upgrade pillar has the following structure:

```
<app>:
  upgrade:
    upgrade_enabled: true|false
    old_release: <release of openstack we upgrade from>
    new_release: <release of ipenstack we upgrade to>
```

Also, an application can use additional upgrade parameters to control the upgrade behaviour. For example, to disable the Neutron router migration, configure the pillar as follows:

```
neutron:
  upgrade:
    resource_migration:
      l3:
        enabled: false
```

OpenStack upgrade workflow

Normally, an OpenStack upgrade includes the following stages:

OpenStack upgrade stages

#	Stage	Description
1	Planning	Includes the creation of the maintenance plan.
2	Pre-upgrade	Includes procedures that do not affect workability of the current OpenStack version such as running a QA cycle, verifying infrastructure, configuring monitoring of workloads and services. Also, during this stage, the backups are created and additional services, servers, and systems are installed to facilitate the upgrade process according to the plan.
3	Upgrade	During this stage, the actual upgrade takes place.
3.1	Control plane upgrade	During this stage, the control plane is being upgraded. It should not have an impact on the data plane, but there might be compatibility issues between the data plane and control plane of different versions. To minimize the control plane downtime, we recommend performing the quickest upgrade depth, which is the application level upgrade first. And, once full OpenStack upgrade is completed, perform the dist-upgrade or even release upgrade by removing one controller node from the cloud, upgrading it, and adding it back.
3.2	Data plane upgrade	During this stage, the servers that host the end-user data applications including the compute, storage, and gateway nodes are being upgraded. Depending on the upgrade requirements, any kind of upgrade depths can be applied.
4	Post upgrade	Includes procedures that will not affect workability, post upgrade testing activities, and cleanup.

Warning

Before you perform the upgrade on a production environment, accomplish the procedure on a staging environment. If the staging environment does not exist, adapt the exact cluster model and launch it inside the cloud as a heat stack, which will act as a staging environment.

Plan the OpenStack upgrade

As a result of the planning stage of the OpenStack upgrade, a detailed maintenance plan is created.

The maintenance plan must include the following parts:

- A strict step-by-step upgrade procedure
- A rollback plan
- A maintenance window schedule for each upgrade phase

> **Note**
>
> The upgrade flow is thoroughly selected by engineers in correspondence with the workload and requirements of a particular cloud.

After the maintenance plan is successfully tested on a staging environment, you can proceed with the actual upgrade in production.

> **Caution!**
>
> Manila deprecation notice
>
> In the MCP 2019.2.7 update, the OpenStack Manila component is being considered for deprecation. The corresponding capabilities are still available, although not further enhanced.
>
> Starting with the 2019.2.11 maintenance update, the OpenStack Manila component will no longer be supported by Mirantis. For those existing customers who have the Manila functionality explicitly included in the scope of their contracts, Mirantis will continue to fulfill the corresponding support obligations.

Limitations

The following are the limitations of the OpenStack upgrade pipelines:

- The pipelines upgrade only the OpenStack component. The upgrade of other VCP components such as RabbitMQ and MySQL is out of scope and should be done independently.

  Caution!

  We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.

- The upgrade of StackLight LMA is out of scope. To obtain the latest version, upgrade StackLight LMA as described in Upgrade StackLight LMA to Build ID 2019.2.0.

Seealso

- Prerequisites
- Upgrade RabbitMQ

Prerequisites

Before you proceed with the OpenStack upgrade, verify the following:

- Upgrade of MCP is done to the latest build ID. Verify that you have updated DriveTrain including Aptly, Gerrit, Jenkins, Reclass, Salt formulas, and their subcomponents to the current MCP release version. Otherwise, the current MCP product documentation is not applicable to your MCP deployment.
- All OpenStack formula states like (nova, neutron, etc) can be launched without errors.
- Online dbsyncs for services are performed before the upgrade maintenance window since this task can take significant time based on a cloud size.
- No failed OpenStack services or nodes are present in the cloud.
- Utilization of the disk space is up to 80% on each target node, which include the ctl*, prx*, gtw*, and cmp* nodes.
- There is enough disk space on the node that will store the backups for the MySQL databases and the existing cluster model.
- For the MCP 2019.2.3 OpenStack environments, verify that the python-tornado package is installed from the latest OpenStack version or the OpenStack version to which you are going to upgrade. Restart the Salt minion after the package installation.

Seealso

- Limitations

Upgrade OpenStack from Ocata to Pike

This section includes the detailed procedure to upgrade OpenStack from Ocata to Pike.

> **Caution!**
>
> We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.

Perform the pre-upgrade activities

The pre-upgrade stage includes the activities that do not affect workability of a currently running OpenStack version as well as the backups creation.

To prepare your OpenStack deployment for the upgrade:

1. Perform the steps described in Configure load balancing for Horizon.
2. On one of the controller nodes, perform the online database migrations for the following services:

   Note

   The database migrations can be time-consuming and create high load on CPU and RAM. We recommend that you perform the migrations in batches.

   - Nova:

     ```
     nova-manage db online_data_migrations
     ```

   - Cinder:

     ```
     cinder-manage db online_data_migrations
     ```

   - Ironic:

     ```
     ironic-dbsync online_data_migrations
     ```

3. Prepare the target nodes for the upgrade:

   1. Log in to the Salt Master node.
   2. Verify that the OpenStack cloud configuration file is present on the controller nodes:

      ```
      salt -C 'I@keystone:client:os_client_config' state.apply keystone.client.os_client_config
      ```

   3. Get the list of all upgradable OpenStack components:

      ```
      salt-call config.get orchestration:upgrade:applications --out=json
      ```

      Example of system response:

      ```
      {
         "<model_of_salt_master_name>": {
            "nova": {
               "priority": 1100
      ```

```
    },
    "heat": {
      "priority": 1250
    },
    "keystone": {
      "priority": 1000
    },
    "horizon": {
      "priority": 1800
    },
    "cinder": {
      "priority": 1200
    },
    "glance": {
      "priority": 1050
    },
    "neutron": {
      "priority": 1150
    },
    "designate": {
      "priority": 1300
    }
  }
}
```

4. Range the components from the output by priority. For example:

```
keystone
glance
nova
neutron
cinder
heat
designate
horizon
```

5. Get the list of all target nodes:

```
salt-key | grep $cluster_domain | \
grep -v $salt_master_hostname | tr '\n' ' '
```

The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters.

The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master.

6. For each target node, get the list of the installed applications:

```
salt <node_name> pillar.items __reclass__:applications --out=json
```

7. Verify that the outdated version of the nova-osapi_compute service is not running:

```
salt -C 'I@galera:master' mysql.query nova 'select services.id, services.host, \
services.binary, services.version from services where services.version < 15'
```

   If the system output contains the nova-osapi_compute service, delete it by running the following commands from any OpenStack controller node:

```
source keystonercv3
openstack compute service delete <nova-osapi_compute_service_id>
```

8. Match the lists of upgradable OpenStack components with the lists of installed applications for each target node.
9. Apply the following states to each target node for each installed application in strict order of priority:

   Warning

   During upgrade, the applications running on the target nodes use the KeystoneRC metadata. To guarantee that the KeystoneRC metadata is exported to mine, verify that you apply the keystone.upgrade.pre formula to the keystone:client:enabled node:

```
salt -C 'I@keystone:client:enabled' state.sls keystone.upgrade.pre
```

```
salt <node_name> state.apply <component_name>.upgrade.pre
salt <node_name> state.apply <component_name>.upgrade.verify
```

   For example, for Nova installed on the cmp01 compute node, run:

```
salt cmp01 state.apply nova.upgrade.pre
salt cmp01 state.apply nova.upgrade.verify
```

On the clouds of medium and large sizes, you may want to automate the step 3 to prepare the target nodes for the upgrade. Use the following script as an example of possible automatization.

```
 #!/bin/bash
#List of formulas that implements upgrade API sorted by priority
all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | \
sort_by(.priority) | map(.key | [(.)]) | add' | \
sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
#List of nodes in cloud
list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
for node in $list_nodes; do
  #List of applications on the given node
  node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.pre
      salt $node state.apply $component.upgrade.verify
    fi
  done
done
```

4. Add the testing workloads to each compute host and monitoring and verify the following:
    - The cloud services are monitored as expected.
    - There are free resources (disk, RAM, CPU) on the kvm, ctl, cmp, and other nodes.
5. Back up the OpenStack databases as described in Back up and restore a MySQL database.
6. Adjust the cluster model:
    1. Include the upgrade pipeline job to DriveTrain:
        1. Add the following lines to classes/cluster/<cluster_name>/cicd/control/leader.yml:

> **Caution!**
>
> If your MCP OpenStack deployment includes the OpenContrail component, do not specify the system.jenkins.client.job.deploy.update.upgrade_ovs_gateway class.

```
classes:
 - system.jenkins.client.job.deploy.update.upgrade
 - system.jenkins.client.job.deploy.update.upgrade_ovs_gateway
 - system.jenkins.client.job.deploy.update.upgrade_compute
```

2. Apply the jenkins.client state on the Jenkins nodes:

```
salt -C 'I@jenkins:client' state.sls jenkins.client
```

2. Set the parameters in classes/cluster/<cluster_name>/infra/init.yml as follows:

```
parameters:
  _param:
    openstack_version: pike
    openstack_old_version: ocata
    openstack_upgrade_enabled: true
```

3. (Optional) Upgrade pillars of all supported OpenStack applications are already included in the Reclass system level. In case of a non-standard setup, the list of the OpenStack applications on each node should be checked and upgrade pillars added for the Openstack applications that do not contain them. For example:

```
<app>:
  upgrade:
    enabled: ${_param:openstack_upgrade_enabled}
    old_release: ${_param:openstack_old_version}
    new_release: ${_param:openstack_version}
```

Note

On the clouds of medium and large sizes, you may want to automate this step. To obtain the list of the OpenStack applications running on a node, use the following script.

```
#!/bin/bash
#List of formulas that implements upgrade API sorted by priority
all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
#List of nodes in cloud
list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
for node in $list_nodes; do
  #List of applications on the given node
```

```
  node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  node_openstack_app=" "
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      node_openstack_app="$node_openstack_app $component"
    fi
  done
  echo "$node : $node_openstack_app"
done
```

4. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
```

7. Prepare the target nodes for the upgrade:

1. Get the list of all upgradable OpenStack components:

```
salt-call config.get orchestration:upgrade:applications --out=json
```

2. Range the components from the output by priority.
3. Get the list of all target nodes:

```
salt-key | grep $cluster_domain | \
grep -v $salt_master_hostname | tr '\n' ' '
```

The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters

The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master

4. For each target node, get the list of installed applications:

```
salt <node_name> pillar.items __reclass__:applications --out=json
```

5. Match the lists of upgradable OpenStack components with the lists of installed applications for each target node.
6. Apply the following states to each target node for each installed application in strict order of priority:

```
salt <node_name> state.apply <component_name>.upgrade.pre
```

> Note
>
> On the clouds of medium and large sizes, you may want to automate this step. Use the following script as an example of possible automatization.

```
 #!/bin/bash
#List of formulas that implements upgrade API
all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
#List of nodes in cloud
list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
for node in $list_nodes; do
  #List of applications on the given node
  node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.pre
    fi
  done
done
```

8. Apply the linux.system.repo state on the target nodes.
9. Proceed to Upgrade the OpenStack control plane.

Upgrade the OpenStack control plane

The OpenStack control plane upgrade stage includes upgrading of the OpenStack services APIs. We recommend that you select the quickest upgrade depth that does not include running OS_UPGRADE or OS_DIST_UPGRADE to minimize the API downtime. You can perform both OS_UPGRADE and OS_DIST_UPGARDE during the post-upgrade stage if required.

To upgrade the OpenStack VCP:

1. Log in to the Jenkins web UI.
2. Run the Deploy - upgrade control VMs pipeline on the OpenStack controller nodes in the interactive mode setting the parameters as follows:

   * TARGET_SERVERS=’ctl*’
   * MODE=INTERACTIVE mode to get the detailed description of the pipeline flow through the stages
3. Verify that the control plane is up and the OpenStack services from the data plane are reconnected and working correctly with the newly upgraded control plane.
4. Run the Deploy - upgrade control VMs on the proxy nodes setting TARGET_SERVERS=’prx*’.
5. Verify that the public API is accessible and Horizon is working.
6. Perform the upgrade of other control plane nodes where required depending on your deployment.
7. Verify that the control plane is upgraded to the intended OpenStack release, APIs work correctly and are available, and the services enable the users to manage their resources.

   > Note
   >
   > The new features of the intended OpenStack release are not available till the data plane nodes are upgraded.
8. Proceed to Upgrade the OpenStack data plane.

Upgrade the OpenStack data plane

The OpenStack data plane includes the servers that host end-user data applications. More specifically, these hosts include compute, storage, and gateway nodes. Depending on the upgrade requirements, you can apply any kind of the upgrade depths while upgrading the data plane.

To upgrade the data plane of your OpenStack deployment:

1. To upgrade the gateway nodes, select one of the following options:

   > **Caution!**
   >
   > Skip this step if your MCP OpenStack deployment includes the OpenContrail component because such configuration does not contain gateway nodes.

   - Non-HA routers are present in the cloud:

     1. Migrate the non-HA routers from the target nodes using the Neutron service and the following commands in particular:

        - l3-agent-router-add
        - l3-agent-router-remove
        - router-list-on-l3-agent
     2. Log in to the Jenkins web UI.
     3. Run the Deploy - upgrade OVS gateway pipeline for the gateway nodes which you have migrated the workloads from.

        > **Note**
        >
        > Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

        > **Note**
        >
        > Since all resources have already been migrated from the nodes, we recommend performing the full upgrade including OS_UPGRADE and OS_DIST_UPGRADE.

4. Migrate the non-HA routers back and rerun the Deploy - upgrade OVS gateway pipeline for the rest of the gateway nodes.
5. Verify that the gateway components are reconnected to the control plane.

- Non-HA routers are not present in the cloud:

1. Log in to the Jenkins web UI.
2. Run the Deploy - upgrade OVS gateway pipeline for all gateway nodes specifying TARGET_SERVERS='gtw*'.

> Note
>
> Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

2. Verify that the gateway components are reconnected to the control plane.

> Caution!
>
> Skip this step if your MCP OpenStack deployment includes the OpenContrail component because such configuration does not contain gateway nodes.

3. Upgrade the OpenStack compute nodes.

1. Estimate and minimize the risks and address the limitations of live migration.

The limitations of the live migration technology include:

> Warning
>
> Before proceeding with live migration in a production environment, assess these risks thoroughly.

- The CPU of a source compute node must have a feature set that is a subset of a feature set of the target compute CPU. Therefore, the migration should be performed between the compute nodes with identical CPUs with, preferably, identical microcode versions.
- During the live migration, the entire memory state of a VM must be copied to another server. In the first place, the memory pages that are being changed at a slower rate are copied. After, the system copies the most active memory pages.

If the number of pages that are being written to all the time is big, the migration process will never finish. High-memory, high-load Windows virtual machines are known to have this particular issue.

- During the live migration, a very short downtime (1-2 seconds max) occurs. The reason for the downtime is that when the memory is copied, the execution context (VCPU state) has to be copied as well, and the execution itself must be switched to a new virtual machine. In addition to a short downtime, this causes a short clock lag on the migrated virtual machine. Therefore, if the migrated machine is hosting a part of a clustered service or system, the downtime and resulting time lag may have an adverse impact on the whole system.
- The QEMU version installed on the source and target hosts should be the same and later than 2.5.

2. Perform the live migration of workloads.
3. Log in to the Jenkins web UI.
4. Run the Deploy - upgrade computes pipeline to upgrade the OpenStack compute nodes which you have migrated the workloads from. It is essential that you upgrade the compute nodes by small batches.

   Caution!

   The impact of the upgrade process should be calculated for each compute node during the planning stage as this step may take a significant amount of time.

   Note

   Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

5. Migrate the workloads back and rerun the Deploy - upgrade computes pipeline for the rest of the compute nodes.

4. Verify that the compute nodes are reconnected to the control plane.
5. Proceed to Perform the post-upgrade activities.

Perform the post-upgrade activities

The post-upgrade activities include the post-upgrade testing cycle and cleanup.

To finalize the upgrade:

1. Perform the full verification cycle of your MCP OpenStack deployment.
2. Verify that the following variables are set in the classes/cluster/<cluster_name>/infra/init.yml file:

   ```
   parameters:
     _param:
       openstack_upgrade_enabled: false
       openstack_version: pike
       openstack_old_version: ocata
   ```

3. Refresh pillars:

   ```
   salt '*' saltutil.refresh_pillar
   ```

4. Remove the test workloads/monitoring.
5. Remove the upgrade leftovers that were created by applying the <app>.upgrade.post state:

   1. Log in to the Salt Master node.
   2. Get the list of all upgradable OpenStack components. For example:

      ```
      salt cfg01* config.get orchestration:upgrade:applications --out=json
      ```

      Example of system response:

      ```
      {
         "<model_of_salt_master_name>": {
            "nova": {
               "priority": 1100
            },
            "heat": {
               "priority": 1250
            },
            "keystone": {
               "priority": 1000
            },
            "horizon": {
               "priority": 1800
            },
            "cinder": {
               "priority": 1200
      ```

```
        },
        "glance": {
            "priority": 1050
        },
        "neutron": {
            "priority": 1150
        },
        "designate": {
            "priority": 1300
        }
    }
}
```

3. Range the components from the output by priority. For example:

```
keystone
glance
nova
neutron
cinder
heat
designate
horizon
```

4. Get the list of all target nodes:

```
salt-key | grep $cluster_domain | \
grep -v $salt_master_hostname | tr '\n' ' '
```

> Note
>
> The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters
>
> The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master

5. For each target node, get the list of installed applications:

```
salt <node_name> pillar.items __reclass__:applications --out=json
```

6. Match the lists of upgradable OpenStack components with the lists of installed applications for each target node.
7. Apply the following states to each target node for each installed application in strict order of priority:

   ```
   salt <node_name> state.apply <component_name>.upgrade.post
   ```

   For example, for Nova installed on the cmp01 compute node, run:

   ```
   salt cmp01 state.apply nova.upgrade.post
   ```

Note

On the clouds of medium and large sizes, you may want to automate this step. Use the following script as an example of possible automatization. Before running the script, verify that you define the $cluster_domain and $salt_master_hostname variables.

```
 #!/bin/bash
#List of formulas that implements upgrade API sorted by priority
all_formulas=$(salt cfg01* config.get orchestration:upgrade:applications   --out=json | \
jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
#List of nodes in cloud
list_nodes=`salt-key | grep $cluster_domain | grep -v $salt_master_hostname | tr '\n' ' '`
for node in $list_nodes; do
  #List of applications on the given node
  node_applications=$(salt $node pillar.items __reclass__:applications   --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.post
    fi
  done
done
```

6. Set the following variables in classes/cluster/<cluster_name>/infra/init.yml:

   ```
   parameters:
     _param:
       openstack_upgrade_enabled: false
       openstack_version: pike
       openstack_old_version: pike
       ...
   ```

7. Refresh pillars:

   ```
   salt '*' saltutil.refresh_pillar
   ```

Upgrade OpenStack from Pike to Queens

This section includes the detailed procedure to upgrade OpenStack from Pike to Queens.

> **Caution!**
>
> We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.

©2025, Mirantis Inc.

Perform the pre-upgrade activities

The pre-upgrade stage includes the activities that do not affect workability of a currently running OpenStack version as well as the backups creation.

To prepare your OpenStack deployment for the upgrade:

1. Perform the steps described in Configure load balancing for Horizon.
2. On one of the controller nodes, perform the online database migrations for the following services:

   Note

   The database migrations can be time-consuming and create high load on CPU and RAM. We recommend that you perform the migrations in batches.

   - Nova:

     ```
     nova-manage db online_data_migrations
     ```

   - Cinder:

     ```
     cinder-manage db online_data_migrations
     ```

   - Ironic:

     ```
     ironic-dbsync online_data_migrations
     ```

3. Prepare the target nodes for the upgrade.

   Note

   On the clouds of medium and large sizes, you may want to automate the steps below to prepare the target nodes for the upgrade. Use the following script as an example of possible automatization.

   ```
   #!/bin/bash
   #List of formulas that implements upgrade API sorted by priority
   all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
   jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
   sed -e 's/"//g' -e 's/,//g' -e 's/[//g' -e 's/]//g')
   #List of nodes in cloud
   list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
   for node in $list_nodes; do
   ```

```
  #List of applications on the given node
  node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.pre
      salt $node state.apply $component.upgrade.verify
    fi
  done
done
```

1. Log in to the Salt Master node.
2. Verify that the OpenStack cloud configuration file is present on the controller nodes:

   ```
   salt -C 'I@keystone:client:os_client_config' state.apply keystone.client.os_client_config
   ```

3. Get the list of all upgradable OpenStack components:

   ```
   salt-call config.get orchestration:upgrade:applications --out=json
   ```

   Example of system response:

   ```
   {
      "<model_of_salt_master_name>": {
         "nova": {
            "priority": 1100
         },
         "heat": {
            "priority": 1250
         },
         "keystone": {
            "priority": 1000
         },
         "horizon": {
            "priority": 1800
         },
         "cinder": {
            "priority": 1200
         },
         "glance": {
            "priority": 1050
         },
         "neutron": {
            "priority": 1150
         },
         "designate": {
   ```

```
            "priority": 1300
        }
    }
}
```

4. Range the components from the output by priority. For example:

```
keystone
glance
nova
neutron
cinder
heat
designate
horizon
```

5. Get the list of all target nodes:

```
salt-key | grep $cluster_domain | \
grep -v $salt_master_hostname | tr '\n' ' '
```

   The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters

   The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master

6. For each target node, get the list of the installed applications:

```
salt <node_name> pillar.items __reclass__:applications --out=json
```

7. Match the lists of upgradable OpenStack components with the lists of installed applications for each target node.

8. If the public endpoint for the Nova placement API was not created before:

   1. Add the following class to the Reclass model in the classes/cluster/<cluster_name>/openstack/proxy.yml file:

   ```
   classes:
   ...
   - system.nginx.server.proxy.openstack.placement
   ```

   2. Refresh pillars on the proxy nodes:

```
salt 'prx*' saltutil.refresh_pillar
```

3. Apply the nginx state on the proxy nodes:

```
salt 'prx*' state.sls nginx
```

9. Apply the following states to each target node for each installed application in strict order of priority:

> Warning
>
> During upgrade, the applications running on the target nodes use the KeystoneRC metadata. To guarantee that the KeystoneRC metadata is exported to mine, verify that you apply the keystone.upgrade.pre formula to the keystone:client:enabled node:
>
> ```
> salt -C 'I@keystone:client:enabled' state.sls keystone.upgrade.pre
> ```

```
salt <node_name> state.apply <component_name>.upgrade.pre
salt <node_name> state.apply <component_name>.upgrade.verify
```

For example, for Nova installed on the cmp01 compute node, run:

```
salt cmp01 state.apply nova.upgrade.pre
salt cmp01 state.apply nova.upgrade.verify
```

4. Add the testing workloads to each compute host and monitoring and verify the following:
   - The cloud services are monitored as expected.
   - There are free resources (disk, RAM, CPU) on the kvm, ctl, cmp, and other nodes.
5. Back up the OpenStack databases as described in Back up and restore a MySQL database.
6. If Octavia is enabled, move the Octavia certificates from the gtw01 to the Salt Master node.
7. Adjust the cluster model for the upgrade:
   1. Include the upgrade pipeline job to DriveTrain:
      1. Add the following lines to classes/cluster/<cluster_name>/cicd/control/leader.yml:

Caution!

If your MCP OpenStack deployment includes the OpenContrail component, do not specify the system.jenkins.client.job.deploy.update.upgrade_ovs_gateway class.

```
classes:
 - system.jenkins.client.job.deploy.update.upgrade
 - system.jenkins.client.job.deploy.update.upgrade_ovs_gateway
 - system.jenkins.client.job.deploy.update.upgrade_compute
```

2. Apply the jenkins.client state on the Jenkins nodes:

```
salt -C 'I@jenkins:client' state.sls jenkins.client
```

2. Set the parameters in classes/cluster/<cluster_name>/infra/init.yml as follows:

```
parameters:
  _param:
    openstack_version: queens
    openstack_old_version: pike
    openstack_upgrade_enabled: true
```

3. (Optional) To upgrade Gnocchi to the version supported in Queens, modify the classes/cluster/<cluster_name>/openstack/init.yml file:

1. Define the following parameters:

```
parameters:
  _param:
    gnocchi_version: 4.2
    gnocchi_old_version: 4.0
```

2. Update the Redis server version from 3.0 to 5.0:

```
parameters:
  redis:
    server:
      version: 5.0
```

4. (Optional) The upgrade pillars of all supported OpenStack applications are already included to the system level of Reclass. In case of a non-standard setup, check the list

of OpenStack applications on each node and add the upgrade pillars to the OpenStack applications that do not contain them. For example:

```
<app>:
  upgrade:
    enabled: ${_param:openstack_upgrade_enabled}
    old_release: ${_param:openstack_old_version}
    new_release: ${_param:openstack_version}
```

> Note
>
> To obtain the list of the OpenStack applications running on a node, use the following script.
>
> ```
> #!/bin/bash
> #List of formulas that implements upgrade API sorted by priority
> all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
> jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
> sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
> #List of nodes in cloud
> list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
> for node in $list_nodes; do
>   #List of applications on the given node
>   node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
>   jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
>   node_openstack_app=" "
>   for component in $all_formulas ; do
>     if [[ " ${node_applications[*]} " == *"$component"* ]]; then
>       node_openstack_app="$node_openstack_app $component"
>     fi
>   done
>   echo "$node : $node_openstack_app"
> done
> ```

5. Enable the Keystone v3 client configuration for the Keystone resources creation by editing classes/cluster/<cluster_name>/openstack/control/init.yml:

```
classes:
- system.keystone.client.v3
```

6. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
```

7. Apply the salt.minion state:

```
salt '*' state.apply salt.minion
```

8. Prepare the target nodes for the upgrade:

1. Get the list of all upgradable OpenStack components:

   ```
   salt-call config.get orchestration:upgrade:applications --out=json
   ```

2. Range the components from the output by priority.
3. Get the list of all target nodes:

   ```
   salt-key | grep $cluster_domain | \
   grep -v $salt_master_hostname | tr '\n' ' '
   ```

   The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters.

   The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master.

4. For each target node, get the list of installed applications:

   ```
   salt <node_name> pillar.items __reclass__:applications --out=json
   ```

5. Match the lists of upgradable OpenStack components with the lists of installed applications for each target node.
6. Apply the following states to each target node for each installed application in strict order of priority:

   ```
   salt <node_name> state.apply <component_name>.upgrade.pre
   ```

> Note
>
> On the clouds of medium and large sizes, you may want to automate this step. Use the following script as an example of possible automatization.
>
> ```
>  #!/bin/bash
> #List of formulas that implements upgrade API
> all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
> jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority" :       $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
> sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
> #List of nodes in cloud
> list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
> for node in $list_nodes; do
>   #List of applications on the given node
>   node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
> ```

```
jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.pre
    fi
  done
done
```

9. Apply the linux.system.repo state on the target nodes.
10. Proceed to Upgrade the OpenStack control plane.

Upgrade the OpenStack extra components

The extra OpenStack components include:

- Manila
- Barbican
- Tenant Telemetry (Aodh, Ceilometer, Panko, and Gnocchi)

MCP supports the upgrade of the extra OpenStack components starting from the Pike OpenStack release deployed with the 2018.11.0 MCP version.

By default, these components are running on dedicated VCP nodes and can be upgraded during separate maintenance windows using the Deploy - upgrade control VMs job as described in Upgrade the OpenStack control plane.

Upgrade the OpenStack control plane

The OpenStack control plane upgrade stage includes upgrading of the OpenStack services APIs. To minimize the API downtime, we recommend that you select the quickest upgrade depth that does not include running OS_UPGRADE or OS_DIST_UPGRADE. You can perform both OS_UPGRADE and OS_DIST_UPGRADE during the post-upgrade stage if required.

To upgrade the OpenStack VCP:

1. Log in to the Jenkins web UI.
2. If Ironic will be upgraded, perform the following steps to upgrade Ironic Conductor before the Ironic API. The nova-compute service running on the Ironic Conductor nodes must be upgraded only after the Nova Controller has been upgraded.

   > Caution!
   >
   > The upgrade of Ironic is available starting from the MCP 2019.2.6 update. See MCP Release Notes: Maintenance updates for details.

   1. Verify that the following variables are set on each Ironic Conductor node in Reclass in the classes/nodes/_generated/<node_name>.yml files:

      ```
      parameters:
        _param:
          nova:
            upgrade:
              enabled: false
      ```

   2. Refresh pillars:

      ```
      salt '*' saltutil.refresh_pillar
      ```

   3. Run the Deploy - upgrade control VMs pipeline job on the Ironic Conductor nodes in the interactive mode setting the TARGET_SERVERS parameter to bmt*.
   4. Once the pipeline job execution is finished, verify that the following variables are set for each Ironic Conductor node in Reclass in the classes/nodes/_generated/<node_name>.yml files:

      ```
      parameters:
        _param:
          nova:
            upgrade:
              enabled: true
          ironic:
      ```

```
upgrade:
  enabled: false
```

5. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
```

3. Run the Deploy - upgrade control VMs pipeline job on the OpenStack controller nodes in the interactive mode setting the parameters as follows:

   - TARGET_SERVERS=ctl*

     After you upgrade the ctl nodes, define the following values one by one to upgrade additional OpenStack components from Pike to Queens as required:

     - TARGET_SERVERS=share* to upgrade the Manila control plane
     - TARGET_SERVERS=mdb* to upgrade the Tenant Telemetry including Ceilometer, Gnocchi, Aodh, and Panko
     - TARGET_SERVERS=kmn* to upgrade Barbican
     - TARGET_SERVERS=bmt* to upgrade Ironic

     > Note
     >
     > During the second execution of the pipeline job on the Ironic Conductor nodes, only nova-compute will be upgraded since Ironic has already been upgraded in step 2.

   - MODE=INTERACTIVE mode to get the detailed description of the pipeline job flow through the stages

4. Verify that the control plane is up and the OpenStack services from the data plane are reconnected and working correctly with the newly upgraded control plane.
5. Run the Deploy - upgrade control VMs pipeline job on the proxy nodes with TARGET_SERVERS='prx*' set.
6. Verify that the public API is accessible and Horizon is working.
7. Perform the upgrade of other control plane nodes where required depending on your deployment.
8. Verify that the control plane is upgraded to the intended OpenStack release, APIs work correctly and are available, and the services enable the users to manage their resources.

> Note
>
> The new features of the intended OpenStack release are not available till the data plane nodes are upgraded.

9. Proceed to Upgrade the OpenStack data plane.

> Seealso
>
> - MCP 2019.2.3 Maintenance Update: OpenStack upgrade-related known issues
> - MCP 2019.2.4 Maintenance Update: OpenStack upgrade-related known issues

Upgrade the OpenStack data plane

The OpenStack data plane includes the servers that host end-user data applications. More specifically, these hosts include compute, storage, and gateway nodes. Depending on the upgrade requirements, you can apply any kind of the upgrade depths while upgrading the data plane.

To upgrade the data plane of your OpenStack deployment:

1. To upgrade the gateway nodes, select one of the following options:

   > Caution!
   >
   > Skip this step if your MCP OpenStack deployment includes the OpenContrail component because such configuration does not contain gateway nodes.

   - Non-HA routers are present in the cloud:

     1. Migrate the non-HA routers from the target nodes using the Neutron service and the following commands in particular:

        - l3-agent-router-add
        - l3-agent-router-remove
        - router-list-on-l3-agent

     2. Log in to the Jenkins web UI.
     3. Run the Deploy - upgrade OVS gateway pipeline for the gateway nodes which you have migrated the workloads from.

        > Note
        >
        > Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

        > Note
        >
        > Since all resources have already been migrated from the nodes, we recommend performing the full upgrade including OS_UPGRADE and OS_DIST_UPGRADE.

4. Migrate the non-HA routers back and rerun the Deploy - upgrade OVS gateway pipeline for the rest of the gateway nodes.
5. Verify that the gateway components are reconnected to the control plane.

- Non-HA routers are not present in the cloud:

1. Log in to the Jenkins web UI.
2. Run the Deploy - upgrade OVS gateway pipeline for all gateway nodes specifying TARGET_SERVERS='gtw*'.

> Note
>
> Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

2. Verify that the gateway components are reconnected to the control plane.

> Caution!
>
> Skip this step if your MCP OpenStack deployment includes the OpenContrail component because such configuration does not contain gateway nodes.

3. Upgrade the OpenStack compute nodes.

1. Estimate and minimize the risks and address the limitations of live migration.

The limitations of the live migration technology include:

> Warning
>
> Before proceeding with live migration in a production environment, assess these risks thoroughly.

- The CPU of a source compute node must have a feature set that is a subset of a feature set of the target compute CPU. Therefore, the migration should be performed between the compute nodes with identical CPUs with, preferably, identical microcode versions.
- During the live migration, the entire memory state of a VM must be copied to another server. In the first place, the memory pages that are being changed at a slower rate are copied. After, the system copies the most active memory pages.

If the number of pages that are being written to all the time is big, the migration process will never finish. High-memory, high-load Windows virtual machines are known to have this particular issue.

- During the live migration, a very short downtime (1-2 seconds max) occurs. The reason for the downtime is that when the memory is copied, the execution context (VCPU state) has to be copied as well, and the execution itself must be switched to a new virtual machine. In addition to a short downtime, this causes a short clock lag on the migrated virtual machine. Therefore, if the migrated machine is hosting a part of a clustered service or system, the downtime and resulting time lag may have an adverse impact on the whole system.
- The QEMU version installed on the source and target hosts should be the same and later than 2.5.

2. Perform the live migration of workloads.
3. Log in to the Jenkins web UI.
4. Run the Deploy - upgrade computes pipeline to upgrade the OpenStack compute nodes which you have migrated the workloads from. It is essential that you upgrade the compute nodes by small batches.

   Caution!

   The impact of the upgrade process should be calculated for each compute node during the planning stage as this step may take a significant amount of time.

   Note

   Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

5. Migrate the workloads back and rerun the Deploy - upgrade computes pipeline for the rest of the compute nodes.

4. Verify that the compute nodes are reconnected to the control plane.
5. Proceed to Perform the post-upgrade activities.

Perform the post-upgrade activities

The post-upgrade activities include the post-upgrade testing cycle and cleanup.

To finalize the upgrade:

1. Perform the full verification cycle of your MCP OpenStack deployment.
2. Verify that the following variables are set in the classes/cluster/<cluster_name>/infra/init.yml file:

```
parameters:
  _param:
    openstack_upgrade_enabled: false
    openstack_version: queens
    openstack_old_version: pike
```

3. (Optional) If Gnocchi was upgraded, define the following parameters in classes/cluster/<cluster_name>/openstack/init.yml:

```
parameters:
  _param:
    gnocchi_version: 4.2
    gnocchi_old_version: 4.0
```

4. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
```

5. Remove the test workloads/monitoring.
6. Remove the upgrade leftovers that were created by applying the <app>.upgrade.post state:

    1. Log in to the Salt Master node.
    2. Get the list of all upgradable OpenStack components. For example:

    ```
    salt cfg01* config.get orchestration:upgrade:applications --out=json
    ```

    Example of system response:

    ```
    {
        "<model_of_salt_master_name>": {
            "nova": {
                "priority": 1100
            },
            "heat": {
                "priority": 1250
            },
    ```

```
    "keystone": {
      "priority": 1000
    },
    "horizon": {
      "priority": 1800
    },
    "cinder": {
      "priority": 1200
    },
    "glance": {
      "priority": 1050
    },
    "neutron": {
      "priority": 1150
    },
    "designate": {
      "priority": 1300
    }
  }
}
```

3. Range the components from the output by priority. For example:

```
keystone
glance
nova
neutron
cinder
heat
designate
horizon
```

4. Get the list of all target nodes:

```
salt-key | grep $cluster_domain | \
grep -v $salt_master_hostname | tr '\n' ' '
```

Note

The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters

> The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master

5. For each target node, get the list of installed applications:

```
salt <node_name> pillar.items __reclass__:applications --out=json
```

6. Match the lists of upgradable OpenStack components with the lists of installed applications for each target node.
7. Apply the following states to each target node for each installed application in strict order of priority:

```
salt <node_name> state.apply <component_name>.upgrade.post
```

   For example, for Nova installed on the cmp01 compute node, run:

```
salt cmp01 state.apply nova.upgrade.post
```

> Note
>
> On the clouds of medium and large sizes, you may want to automate this step. Use the following script as an example of possible automatization. Before running the script, verify that you define the $cluster_domain and $salt_master_hostname variables.

```
 #!/bin/bash
#List of formulas that implements upgrade API sorted by priority
all_formulas=$(salt cfg01* config.get orchestration:upgrade:applications   --out=json | \
jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
#List of nodes in cloud
list_nodes=`salt-key | grep $cluster_domain | grep -v $salt_master_hostname | tr '\n' ' '`
for node in $list_nodes; do
  #List of applications on the given node
  node_applications=$(salt $node pillar.items __reclass__:applications   --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.post
    fi
  done
done
```

7. Set the following variables in classes/cluster/<cluster_name>/infra/init.yml:

```
parameters:
  _param:
    openstack_upgrade_enabled: false
    openstack_version: queens
    openstack_old_version: queens
```

If Gnocchi was upgraded, set the following parameters in the same file:

```
parameters:
  _param:
    gnocchi_version: 4.2
    gnocchi_old_version: 4.2
```

8. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
```

Upgrade Galera

This section includes the instruction on how to upgrade the Galera cluster. You can upgrade Galera either automatically using the corresponding Jenkins pipeline or manually.

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

©2025, Mirantis Inc.

Upgrade Galera automatically

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

This section instructs you on how to upgrade the Galera cluster automatically through Jenkins using the Deploy - upgrade Galera cluster pipeline job.

To upgrade Galera:

1. Log in to the Salt Master node.
2. Open the cluster level of your deployment model.
3. Include the Galera upgrade pipeline job to DriveTrain:
    1. In the classes/cluster/<cluster_name>/cicd/control/leader.yml file, add the following class:

        ```
        classes:
        - system.jenkins.client.job.deploy.update.upgrade_galera
        ```

    2. Apply the jenkins.client state on the Jenkins nodes:

        ```
        salt -C 'I@jenkins:client' state.sls jenkins.client
        ```

    3. In the classes/cluster/<cluster_name>/infra/init.yml file, set the openstack_upgrade_enabled parameter to true:

        ```
        parameters:
          _param:
            openstack_upgrade_enabled: true
        ```

    4. Refresh pillars on the dbs* nodes:

        ```
        salt 'dbs*' saltutil.refresh_pillar
        ```

4. Add repositories with new Galera packages:
    1. Apply the linux.system.repo state on the dbs* nodes:

        ```
        salt 'dbs*' state.sls linux.system.repo
        ```

    2. Update the /etc/salt/minion.d/_orchestration.conf file:

```
salt 'cfg*' state.sls salt.master
```

Warning

Verify that you have installed the latest Galera Salt formula and it is present in /etc/salt/minion.d/_orchestration.conf. The file should include the galera line.

5. Log in to the Jenkins web UI.
6. Open the Deploy - upgrade Galera cluster pipeline.
7. Specify the following parameters as required:

Deploy - upgrade Galera cluster pipeline parameters

Parameter	Description
INTERACTIVE	Specifies the mode to get the detailed description of the pipeline job flow through the stages.
SHUTDOWN_CLUSTER	Shuts down all MySQL instances on the target nodes during upgrade.
OS_DIST_UPGRADE	Upgrades system packages including kernel using apt-get dist-upgrade. Optional.
OS_UPGRADE	Upgrades all installed applications using apt-get upgrade. Optional.
SALT_MASTER_CREDENTIALS	Defines the Salt Master node credentials to use for connection, defaults to salt.
SALT_MASTER_URL	Defines the Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
TARGET_SERVERS	Adds the target database server nodes. Defaults to dbs*.

8. Click Deploy.

   To monitor the deployment process, follow the instructions in MCP Deployment Guide: View the deployment details.

The Deploy - upgrade Galera cluster pipeline workflow

Stage	Description
Pre-upgrade	Only non-destructive actions are applied during this phase. Basic service verification is performed. The job is launched on all target servers before moving to the next stage.

Stop the Galera cluster	All Galera clusters are stopped on the TARGET_SERVERS nodes in the reverse order. For example, dbs03, dbs02, and then dbs01. OpenStack APIs are not accessible starting from this point. This does not affect the data plane services such as OVS or KVM nodes.
Upgrade Galera	The Galera code is upgraded. No workload downtime is expected.
Upgrade OS	Optional. Launches only if OS_UPGRADE or OS_DIST_UPGRADE is selected. A reboot can be performed if required. When the node is back online, the basic service checks are performed.
Start the Galera cluster	The Galera cluster is being started on the TARGET_SERVERS nodes starting with the last instance stooped. For example, the nodes are started in the following order: dbs01, dbs02, and dbs03.

9. Revert the changes in the classes/cluster/<cluster_name>/infra/init.yml file made during step 3.3.

Upgrade Galera manually

This section instructs you on how to manually upgrade the Galera cluster. Only the MySQL and Galera packages will be upgraded. The upgrade of an underlying operating system is out of scope.

During the upgrade, the Galera cluster remains alive while you shut down each MySQL service on the node one by one to upgrade its packages and then restart the service. When the node reconnects, it synchronizes with the cluster as in case of any other outage. The upgrade of an underlying operating system is out of scope.

> Warning
>
> Before performing the upgrade on a production environment:
>
> - Accomplish the procedure on a staging environment to determine the required maintenance window duration.
> - Schedule an appropriate maintenance window to reduce the load on the cluster.
> - Do not shut down the VMs or workloads as networking and storage functions are not affected.

To upgrade the Galera cluster:

1. Prepare the Galera cluster for the upgrade:
    1. Verify that you have added the required repositories on the Galera nodes to download the updated MySQL and Galera packages.
    2. Verify that your Galera cluster is up and running as described in Verify a Galera cluster status.
    3. Create an instant backup of the MySQL database as described in Back up and restore a MySQL database.
2. Log in to the Salt Master node.
3. Obtain the new packages on all Galera nodes:

    ```
    salt -C 'I@galera:*' cmd.run 'apt-get clean; apt-get update'
    ```

4. Verify that the MySQL and Galera packages are available on all Galera nodes:

    ```
    salt -C 'I@galera:*' cmd.run "apt-cache policy mysql-wsrep-5.6 |egrep -i 'installed|candidate'"
    salt -C 'I@galera:*' cmd.run "apt-cache policy galera-3 |egrep -i 'installed|candidate'"
    ```

    Example of system response:

    ```
    dbs02.openstack-ovs-core-ssl-pike-8602.local:
        Installed: 5.6.35-0.1~u16.04+mcp2
    ```

```
      Candidate: 5.6.41-1~u16.04+mcp1
dbs01.openstack-ovs-core-ssl-pike-8602.local:
      Installed: 5.6.35-0.1~u16.04+mcp2
      Candidate: 5.6.41-1~u16.04+mcp1
dbs03.openstack-ovs-core-ssl-pike-8602.local:
      Installed: 5.6.35-0.1~u16.04+mcp2
      Candidate: 5.6.41-1~u16.04+mcp1
```

5. Verify the runtime versions of the MySQL nodes of the Galera cluster:

   ```
   salt -C 'I@galera:*' mysql.version
   ```

   Example of system response:

   ```
   dbs02.openstack-ovs-core-ssl-pike-8602.local:
       5.6.35-0.1~u16.04+mcp2
   dbs01.openstack-ovs-core-ssl-pike-8602.local:
       5.6.35-0.1~u16.04+mcp2
   dbs03.openstack-ovs-core-ssl-pike-8602.local:
       5.6.35-0.1~u16.04+mcp2
   ```

6. Perform the following steps on one of the Galera slave nodes, for example, on the third instance:

   1. Stop the MySQL service:

      ```
      salt -C 'I@galera:* and *03*' service.stop mysql
      ```

   2. Upgrade the packages:

      ```
      salt -C 'I@galera:* and *03*' cmd.run "apt-get -y install --reinstall -o \
      DPkg::Options::=--force-confold -o Dpkg::Options::=--force-confdef mysql-wsrep-5.6 \
      mysql-wsrep-common-5.6 mysql-wsrep-libmysqlclient18 galera-3"
      ```

   3. Apply the galera Salt state:

      ```
      salt -C 'I@galera:* and *03*' state.apply galera
      ```

   4. Verify that your Galera cluster is up and running as described in Verify a Galera cluster status.

7. Perform the step 6 on the remaining Galera nodes one by one.

8. Verify the cluster status after upgrade:

   1. Verify the versions of the installed packages:

```
salt -C 'I@galera:*' mysql.version
```

2. Verify that your Galera cluster is up and running as described in Verify a Galera cluster status.

Starting from the MCP 2019.2.18 maintenance update, you can upgrade your Galera cluster to version 5.7:

Upgrade Galera to v5.7 automatically

> Note
>
> This feature is available starting from the MCP 2019.2.18 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section instructs you on how to automatically upgrade the Galera cluster to version 5.7 using the Deploy - upgrade Galera cluster Jenkins pipeline job.

To upgrade Galera to v5.7 automatically:

1. Verify that you have performed a backup of your database as described in Back up and restore a MySQL database.
2. Perform a MySQL dump to a file that can later be used as data source for manual restoration. Verify that the node has enough free space to make a dump file. Run the following command on a database server node (dbs):

   ```
   mysqldump --defaults-file=/etc/mysql/debian.cnf -AR > %dump_file
   ```

3. Log in to the Salt Master node.
4. Open the cluster level of your deployment model.
5. Specify the new version for Galera packages:

   1. In <cluster_name>/openstack/database/mysql_version.yml, create a new YAML file with the following content:

      ```
      parameters:
        _param:
          galera_mysql_version: "5.7"
      ```

   2. In <cluster_name>/openstack/database/master.yml and <cluster_name>/openstack/database/slave.yml, add the created file:

      ```
      classes:
      ...
      - cluster.<cluster_name>.openstack.database.mysql_version
      ```

   3. Refresh pillars on the database nodes:

      ```
      salt -C "I@galera:master or I@galera:slave" saltutil.refresh_pillar
      ```

   4. Verify that the pillars of the database nodes have Galera version 5.7:

```
salt -C "I@galera:master or I@galera:slave" pillar.get galera:version:mysql
```

> Warning
>
> If the Galera version is not 5.7, resolve the issue before proceeding with the upgrade.

6. Add repositories with new Galera packages:
    1. Apply the linux.system.repo state on the database nodes:

       ```
       salt -C "I@galera:master or I@galera:slave" state.sls linux.system.repo
       ```
    2. Verify the availability of the new MySQL packages:

       ```
       salt -C "I@galera:master or I@galera:slave" cmd.run 'apt-cache policy mysql-wsrep-server-5.7 mysql-wsrep-5.7'
       ```
    3. Verify the availability of the new percona-xtrabackup-24 packages:

       ```
       salt -C "I@galera:master or I@galera:slave" cmd.run 'apt-cache policy percona-xtrabackup-24'
       ```
    4. Verify that the salt-formula-galera version is 1.0+202202111257.6945afc or later:

       ```
       dpkg -l |grep salt-formula-galera
       ```
7. Log in to the Jenkins web UI.
8. Open the Deploy - upgrade Galera cluster pipeline.
9. Specify the following parameters as required:

Deploy - upgrade Galera cluster pipeline parameters

Parameter	Description
INTERACTIVE	Specifies the mode to get the detailed description of the pipeline job flow through the stages.
UPDATE_TO_MYSQL57	Set this flag if you are upgrading MySQL from version 5.6 to 5.7.
SHUTDOWN_CLUSTER	Shuts down all MySQL instances on the target nodes during upgrade.
OS_DIST_UPGRADE	Optional. Upgrades system packages including kernel using apt-get dist-upgrade.

OS_UPGRADE	Optional. Upgrades all installed applications using apt-get upgrade.
SALT_MASTER_CREDENTIALS	Defines the Salt Master node credentials to use for connection, defaults to salt.
SALT_MASTER_URL	Defines the Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
TARGET_SERVERS	Adds the target database server nodes. Defaults to dbs*.

Warning

Verify that you have selected the UPDATE_TO_MYSQL57 and SHUTDOWN_CLUSTER check boxes.

10. Click Deploy.

- To monitor the deployment process, follow the instructions in MCP Deployment Guide: View the deployment details.

The Deploy - upgrade Galera cluster pipeline workflow

Stage	Description
Pre-upgrade	Only non-destructive actions are applied during this phase. Basic service verification is performed. The job is launched on all target servers before moving to the next stage.
Stop the Galera cluster	All Galera clusters are stopped on the TARGET_SERVERS nodes in the reverse order. For example, dbs03, dbs02, and then dbs01. OpenStack APIs are not accessible starting from this point. This does not affect the data plane services such as OVS or KVM nodes.
Upgrade Galera	The Galera code is upgraded. No workload downtime is expected.
Upgrade OS	Optional. Launches only if OS_UPGRADE or OS_DIST_UPGRADE is selected. A reboot can be performed if required. When the node is back online, the basic service checks are performed.
Start the Galera cluster	The Galera cluster is being started on the TARGET_SERVERS nodes starting with the last instance stooped. For example, the nodes are started in the following order: dbs01, dbs02, and dbs03.

Seealso

MCP Release Notes: MySQL upgrade failure

Upgrade Galera to v5.7 manually

> Note
>
> This feature is available starting from the MCP 2019.2.18 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section instructs you on how to manually upgrade the Galera cluster to version 5.7. The upgrade of an underlying operating system is out of scope.

During the upgrade, the Galera cluster will be shut down and all nodes will be shut down one by one. Then package update and service start will be performed.

> Warning
>
> Before performing the upgrade on a production environment:
>
> - Accomplish the procedure on a staging environment to determine the required maintenance window duration.
> - Schedule an appropriate maintenance window to reduce the load on the cluster.
> - Do not shut down the VMs or workloads as networking and storage functions are not affected.

To upgrade Galera to v5.7 manually:

1. Prepare the Galera cluster for the upgrade:
    1. Verify that you have added the required repositories on the Galera nodes to download the updated MySQL and Galera packages.
    2. Verify that your Galera cluster is up and running as described in Verify a Galera cluster status.
    3. Create an instant backup of the MySQL database as described in Back up and restore a MySQL database.
    4. Perform a MySQL dump to a file that can later be used as data source for manual restoration. Verify that the node has enough free space to make a dump file. Run the following command on a database server node (dbs):

        ```
        mysqldump --defaults-file=/etc/mysql/debian.cnf -AR > %dump_file
        ```
2. Log in to the Salt Master node.
3. Open the cluster level of your deployment model.
4. Specify the new version for Galera packages:

1. In <cluster_name>/openstack/database/mysql_version.yml, create a new YAML file with the following content:

```
parameters:
  _param:
    galera_mysql_version: "5.7"
```

2. In <cluster_name>/openstack/database/master.yml and <cluster_name>/openstack/database/slave.yml, add the created file:

```
classes:
...
- cluster.<cluster_name>.openstack.database.mysql_version
```

3. Refresh pillars on the database nodes:

```
salt -C "I@galera:master or I@galera:slave" saltutil.refresh_pillar
```

4. Verify that the pillars of the database nodes have Galera version 5.7:

```
salt -C "I@galera:master or I@galera:slave" pillar.get galera:version:mysql
```

> Warning
>
> If the Galera version is not 5.7, resolve the issue before proceeding with the upgrade.

5. Add repositories with new Galera packages:

   1. Apply the linux.system.repo state on the database nodes:

   ```
   salt -C "I@galera:master or I@galera:slave" state.sls linux.system.repo
   ```

   2. Verify the availability of the new MySQL packages:

   ```
   salt -C "I@galera:master or I@galera:slave" cmd.run 'apt-cache policy mysql-wsrep-server-5.7 mysql-wsrep-5.7'
   ```

   3. Verify the availability of the new percona-xtrabackup-24 packages:

   ```
   salt -C "I@galera:master or I@galera:slave" cmd.run 'apt-cache policy percona-xtrabackup-24'
   ```

   4. Verify that the salt-formula-galera version is 1.0+202202111257.6945afc or later:

   ```
   dpkg -l |grep salt-formula-galera
   ```

6. Verify the runtime versions of the MySQL nodes of the Galera cluster:

   ```
   salt -C "I@galera:master or I@galera:slave" mysql.version
   ```

   Example of system response:

   ```
   dbs02.openstack-ovs-core-ssl-pike-8602.local:
      5.6.51-1~u16.04+mcp1
   dbs01.openstack-ovs-core-ssl-pike-8602.local:
      5.6.51-1~u16.04+mcp1
   dbs03.openstack-ovs-core-ssl-pike-8602.local:
      5.6.51-1~u16.04+mcp1
   ```

7. Perform the following steps on the Galera nodes one by one:

   1. Stop the MySQL service on the node 3:

      ```
      salt -C 'I@galera:slave and *03*' service.stop mysql
      ```

   2. Stop the MySQL service on the node 2:

      ```
      salt -C 'I@galera:slave and *02*' service.stop mysql
      ```

   3. Stop the MySQL service on the master MySQL node:

      ```
      salt -C 'I@galera:master and *01*' service.stop mysql
      ```

8. Perform the following steps on the MySQL master node that was stopped last:

   1. Open /etc/mysql/my.cnf for editing.
   2. Comment out the wsrep_cluster_address line:

      ```
      ...
      #wsrep_cluster_address="gcomm://192.168.2.51,192.168.2.52,192.168.2.53"
      ...
      ```

   3. Add the wsrep_cluster_address parameter without any IP address specified:

      ```
      ...
      wsrep_cluster_address="gcomm://"
      ...
      ```

9. On the same node, upgrade the packages:

   1. Obtain the MySQL root password:

```
salt -C "I@galera:master" config.get mysql:client:server:database:admin:password
```

2. Update the percona-xtrabackup package:

```
apt-get -y install percona-xtrabackup-24
```

3. Update the Galera packages:

```
apt-get -y install -o DPkg::Options::=--force-confold -o Dpkg::Options::=--force-confdef mysql-wsrep-5.7 mysql-wsrep-common-5.7 galera-3
```

4. In the window that opens, enter the root password obtained in the step above.
5. Restart the mysql service:

```
systemctl restart mysql
```

6. Verify the cluster status:

```
salt-call mysql.status | grep -A1 wsrep_cluster_size
```

10. Perform the step 9 on the remaining Galera nodes one by one.
11. On the node where you have changed the wsrep_cluster_address parameter, apply the galera state and restart the service:

```
salt -C "I@galera:master" state.apply galera
salt -C "I@galera:master" service.restart mysql
```

12. Verify that your Galera cluster is up and running:

```
salt -C 'I@galera:master' mysql.status | \
grep -EA1 'wsrep_(local_state_c|incoming_a|cluster_size)'
```

Example of system response:

```
wsrep_cluster_size:
  3

wsrep_incoming_addresses:
  192.168.2.52:3306,192.168.2.53:3306,192.168.2.51:3306

wsrep_local_state_comment:
  Synced
```

Upgrade RabbitMQ

This section instructs you on how to upgrade the RabbitMQ component automatically through Jenkins using the Deploy - upgrade RabbitMQ server pipeline job.

> Note
>
> - This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.
> - Starting from the MCP 2019.2.10 maintenance update, RabbitMQ version 3.8.2 is available. Prior to updating RabbitMQ to the new version, perform the steps described in Apply maintenance updates and the following prerequisite steps.
> - For OpenStack Pike environments with RabbitMQ version 3.8.2, the timeouts are set to rabbit_retry_interval = 5, rabbit_retry_backoff = 10, and kombu_reconnect_delay = 5.0 by default in the oslo.messaging package.

> **Caution!**
>
> We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.

Prerequisites to upgrade RabbitMQ to version 3.8.2:

1. Log in to the Salt Master node.
2. Verify that the salt-formula-rabbitmq version is 0.2+202006030849.6079bf0~xenial1 or newer:

   ```
   dpkg -l |grep salt-formula-rabbitmq
   ```

3. Verify that the salt-formula-oslo-templates version is 2018.1+202006191406.b3839c0~xenial1 or newer.

   ```
   dpkg -l |grep salt-formula-oslo-templates
   ```

4. Perform the following steps depending on your OpenStack version:

   - For OpenStack Queens, verify that the following OpenStack configuration variables are not lower than specified:

```
rabbit_retry_interval = 5
rabbit_retry_backoff = 10
kombu_reconnect_delay = 5.0
```

For example, to verify the value of rabbit_retry_backoff across all nodes:

```
salt '*' cmd.run 'grep -r rabbit_retry_backoff /etc'
```

- For OpenStack Pike, verify that the python-oslo.messaging package version is 5.30.8-1~u16.04+mcp19 or newer:

```
salt '*' pkg.version 'python-oslo.messaging'
```

5. Verify that the RabbitMQ version 3.8.2 is available on the msg nodes:

```
salt -C 'I@rabbitmq:server' cmd.run 'apt-cache policy rabbitmq-server'
```

To upgrade RabbitMQ:

1. Prepare the Neutron server for the RabbitMQ upgrade:

> **Caution!**
>
> This step is required since the Neutron service is sensitive to the RabbitMQ stability. The following Neutron configuration prevents massive resource rescheduling that can lead to the unbalanced load on the gateway nodes and other undesired consequences.

   1. On each OpenStack controller node, modify the neutron.conf file as follows:

```
allow_automatic_dhcp_failover = false
allow_automatic_l3agent_failover = false
```

   2. Restart the neutron-server service:

```
service neutron-server restart
```

2. For the large clusters with more than 50 nodes and more than 100 Open vSwitch ports per node, stop the Neutron Open vSwitch agents on each gateway and compute node. This prevents overloading of the Neutron servers with massive agent resyncs.

```
service neutron-openvswitch-agent stop
```

3. Log in to the Salt Master node.
4. Open the cluster level of your deployment model.
5. Include the RabbitMQ upgrade pipeline job to DriveTrain:
    1. Add the following class to classes/cluster/<cluster_name>/cicd/control/leader.yml:

    ```
    classes:
    - system.jenkins.client.job.deploy.update.upgrade_rabbitmq
    ```

    2. Apply the jenkins.client state on the Jenkins nodes:

    ```
    salt -C 'I@jenkins:client' state.sls jenkins.client
    ```

    3. Set the parameters in classes/cluster/<cluster_name>/infra/init.yml as follows:

    ```
    parameters:
      _param:
        openstack_upgrade_enabled: true
    ```

    4. Refresh pillars on the msg* nodes:

    ```
    salt 'msg*' saltutil.refresh_pillar
    ```

6. Apply the linux.system.repo state on the msg* nodes to add repositories with new RabbitMQ packages:

    ```
    salt 'msg*' state.sls linux.system.repo
    ```

7. Log in to the Jenkins web UI.
8. Open the Deploy - upgrade RabbitMQ server pipeline.
9. Specify the following parameters as required:

Deploy - upgrade RabbitMQ server pipeline parameters

Parameter	Description
INTERACTIVE	Mode to get the detailed description of the pipeline job flow through the stages
OS_DIST_UPGRADE	Upgrades system packages including kernel, aka apt-get dist-upgrade. Optional, launches only if OS_DIST_UPGRADE is selected.
OS_UPGRADE	Upgrades all installed applications, aka apt-get upgrade. Optional, launches only if OS_UPGRADE is selected.

SALT_MASTER_CREDENTIALS	Defines the Salt Master node credentials to use for connection, defaults to salt.
SALT_MASTER_URL	Defines the Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
TARGET_SERVERS	Adds the target RabbitMQ nodes. Defaults to msg*.

10. Click Deploy.
    To monitor the deployment process, follow the instructions in MCP Deployment Guide: View the deployment details.

The pipeline Deploy - upgrade RabbitMQ server workflow

Stage	Description
Pre-upgrade	Only non-destructive actions are applied during this phase. Basic service verification is performed. The job is launched on all target servers before moving to the next stage.
Stop RabbitMQ service	All RabbitMQ services are stopped on the TARGET_SERVERS nodes in the reverse order. For instance, msg03, msg02, and then msg01. OpenStack APIs are not accessible starting from this point. This does not affect the data plane services such as OVS or KVM.
Upgrade RabbitMQ	RabbitMQ and Erlang code are upgraded. No workload downtime is expected.
Upgrade OS	Optional. Launches only if OS_UPGRADE or OS_DIST_UPGRADE is selected. A reboot can be performed if required. When the node is back online, the basic service checks are performed.
Start RabbitMQ service	All Rabbitmq services are running on the TARGET_SERVERS nodes starting with the last instance stooped. For example, msg01, msg02, and then msg03.

11. Starting from the 2019.2.10 update, apply the fluentd state to the msg nodes:

```
salt -C 'I@rabbitmq:server' state.apply fluentd
```

12. Revert the changes in the neutron.conf file made during step 1.

13. Gradually start the Neutron agents if you have stopped them during step 2:

```
service neutron-dhcp-agent start
service neutron-l3-agent start
service neutron-metadata-agent start
service neutron-openvswitch-agent start
```

14. Revert the changes in the classes/cluster/<cluster_name>/infra/init.yml file made during step 5.3.

Update or upgrade Kubernetes

> **Caution!**
>
> Before proceeding with the upgrade procedure, verify that you have updated DriveTrain including Aptly, Gerrit, Jenkins, Reclass, Salt formulas, and their subcomponents to the current MCP release version. Otherwise, the current MCP product documentation is not applicable to your MCP deployment.

This section describes how to automatically upgrade Kubernetes to a major version or update to a minor version. During this procedure, the downtime occurs only while the kube-apiserver VIP of a node to be updated is moving to another active node. The MCP cluster workload is not affected.

> **Caution!**
>
> During the execution of the Kubernetes upgrade pipeline job:
>
> - The Kubernetes Docker container back end is replaced by containerd and all Kubernetes workloads are moved to containerd. However, the Docker service is not stopped and removed in the event some third-party Docker workloads that are not related to the MCP Kubernetes cluster can be running in Docker. If this is not the case and you do not need Docker anymore, you can disable it after the upgrade.
>
>   Starting from the MCP 2019.2.3 maintenance update, due to the conflict between the docker-engine and the containerd runc packages versions, Docker is removed during the upgrade to prevent the conflict. Therefore, Mirantis recommends migrating the third-party Docker workloads running on the MCP Kubernetes cluster, if any, before the upgrade.
> - If any DaemonSet was deployed on your cluster, all Kubernetes nodes will be rebooted to reflect the containerd changes to DaemonSets.
> - No third-party Docker workloads will be removed from Docker but any Docker workload will be stopped during a node reboot.
>
>   Starting from the MCP 2019.2.3 maintenance update, any third-party Docker workload running on the MCP Kubernetes cluster is stopped and removed along with Docker during a node reboot.

Automatically update or upgrade Kubernetes

This section describes how to update or upgrade your Kubernetes cluster including Calico and etcd [12] using the Jenkins Deploy - update Kubernetes cluster pipeline.

To update or upgrade Kubernetes using the Jenkins pipeline:

1. Log in to the Jenkins web UI.
2. Open the Deploy - update Kubernetes cluster pipeline.
3. Specify the following parameters:

Parameter	Description and values
ARTIFACTORY_URL	Optional. Required for conformance tests only. The artifactory URL where Docker images are located. Automatically taken from Reclass.
CALICO_UPGRADE_VERSION	Define the version of the calico-upgrade utility to use during the Calico upgrade. This option is only relevant if UPGRADE_CALICO_V2_TO_V3 is selected.
CMP_TARGET	Add the target Kubernetes cmp nodes. For example, 'cmp* and I@kubernetes:pool'.
CONFORMANCE_RUN_AFTER	Optional. Select to run the Kubernetes conformance tests after you upgrade a Kubernetes cluster.
CONFORMANCE_RUN_BEFORE	Optional. Select to run the Kubernetes conformance tests before you upgrade a Kubernetes cluster.
CTL_TARGET	Add the target Kubernetes ctl nodes. For example, I@kubernetes:master.
• KUBERNETES_CALICO_CALICOCTL_SOURCE • KUBERNETES_CALICO_CALICOCTL_SOURCE_HASH • KUBERNETES_CALICO_CNI_SOURCE_HASH • KUBERNETES_CALICO_BIRDCL_SOURCE • KUBERNETES_CALICO_BIRDCL_SOURCE_HASH • KUBERNETES_CALICO_CNI_IPAM_SOURCE • KUBERNETES_CALICO_CNI_IPAM_SOURCE_HASH	Leave these fields empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version.
KUBERNETES_CALICO_CNI_SOURCE	Leave this field empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version. For testing purposes, you can add a versioned calico/cni image to use in your deployments.

KUBERNETES_CALICO_IMAGE	Leave this field empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version. For testing purposes, you can add a versioned calico/node image to use in your deployments.
KUBERNETES_CALICO_KUBE_CONTROLLERS_IMAGE	Leave this field empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version. For testing purposes, you can add a versioned calico/kube-controllers image to use in your deployments.
KUBERNETES_ETCD_SOURCE [12]	Leave this field empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version. For testing purposes, you can add a versioned binary of the etcd server to use in your deployment.
KUBERNETES_ETCD_SOURCE_HASH [12]	Leave this field empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version. For testing purposes, you can add the checksum of the versioned etcd binary set in the KUBERNETES_ETCD_SOURCE field.
KUBERNETES_HYPERKUBE_SOURCE	Leave this field empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version. For testing purposes, you can add a versioned image to update the Kubernetes Master nodes from. Also, verify that the - cluster.overrides class exists your cluster model. Otherwise, the KUBERNETES_HYPERKUBE_SOURCE, KUBERNETES_PAUSE_IMAGE, KUBERNETES_CALICO_IMAGE, KUBERNETES_CALICO_CALICOCTL_SOURCE, KUBERNETES_CALICO_CNI_SOURCE, and KUBERNETES_CALICO_KUBE_CONTROLLERS_IMAGE variables will not take effect.
KUBERNETES_HYPERKUBE_SOURCE_HASH	Leave this field empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version.
KUBERNETES_PAUSE_IMAGE	Leave this field empty since you have already updated your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version. For testing purposes, you can add a versioned image to use in your deployments.
PER_NODE	Select to update or upgrade the target Kubernetes nodes one by one. The option is required for the draining and cordoning functionality. Recommended.

©2025, Mirantis Inc.

SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
SIMPLE_UPGRADE	Select if you do not need to drain and cordon the nodes during the cluster upgrade. Not recommended.
TARGET_UPDATES	Add the comma-separated list of the Kubernetes nodes to update. Valid values are ctl, cmp. To update only the Kubernetes Nodes, for example, define cmp only.
TEST_K8S_API_SERVER	Optional. Required for conformance tests only. The IP address of a local Kubernetes API server for conformance tests.
UPGRADE_CALICO_V2_TO_V3	Select to upgrade Calico from version 2.6.x to the latest supported version (3.x). This option is required only for upgrade of Calico from version 2.6.x. To update the minor Calico version (for example, from 3.1.x to 3.3.x), do not select this option, since the regular Kubernetes update procedure already includes the update of Calico minor version. **Caution!** The Calico upgrade process implies the Kubernetes services downtime for workloads operations, for example, workloads spawning and removing. The downtime is caused by the necessity of the etcd schema migration where the Calico endpoints data and other Calico configuration data is stored.

4. Click Deploy. To monitor the deployment process, follow the instruction in MCP Deployment Guide: View the deployment details.
5. Obsolete since 2019.2.3 If you do not have any third-party Docker workloads that run outside the MCP Kubernetes cluster, stop and disable Docker on reboot. Run the following commands on any Kubernetes Master node:

```
systemctl stop docker
systemctl disable docker
```

> Note
>
> If your MCP cluster version is 2019.2.3 or later, skip this step since Docker is removed during the upgrade.

The Deploy - update Kubernetes cluster pipeline workflow:

> Note
>
> While any Kubernetes node is being cordoned or drained, other cluster nodes run its services. Therefore, the workload is not affected.

1. Add the hyperkube images defined in the KUBERNETES_HYPERKUBE_SOURCE, KUBERNETES_HYPERKUBE_SOURCE_HASH, and KUBERNETES_PAUSE_IMAGE pipeline fields or defined on the reclass-system level of the Reclass model for the target Kubernetes ctl and cmp nodes during the MCP release version upgrade.
2. Add the calico images defined in the KUBERNETES_CALICO_IMAGE, KUBERNETES_CALICO_CALICOCTL_SOURCE, KUBERNETES_CALICO_CALICOCTL_SOURCE_HASH, KUBERNETES_CALICO_CNI_SOURCE, KUBERNETES_CALICO_CNI_SOURCE_HASH, KUBERNETES_CALICO_BIRDCL_SOURCE, KUBERNETES_CALICO_BIRDCL_SOURCE_HASH, KUBERNETES_CALICO_CNI_IPAM_SOURCE, KUBERNETES_CALICO_CNI_IPAM_SOURCE_HASH, KUBERNETES_CALICO_KUBE_CONTROLLERS_IMAGE pipeline job parameters or defined on the reclass-system level of the Reclass model for the target Kubernetes ctl and cmp nodes during the MCP release version upgrade.
3. If the UPGRADE_CALICO_V2_TO_V3 option is selected, perform the Calico upgrade:
   1. Download the calico-upgrade utility according to the CALICO_UPGRADE_VERSION setting.
   2. Verify that the Calico upgrade is possible (verify the current Calico version and perform a dry run of the data upgrade).
   3. Verify the Calico policy setting.
   4. Perform the Calico data upgrade and lock Calico.
   5. Update the Calico configuration and restart corresponding services. Calico is not operating during this step, so a cluster experiences the workloads operations downtime.
   6. Unlock Calico.
4. Upgrade etcd on the Kubernetes target ctl nodes one by one.
5. For the first Kubernetes target ctl node:

    1. Cordon the node.
    2. Drain the node.
    3. Regenerate SSL certificates for the node services.
    4. Start the containerd service.
    5. Upgrade the hyperkube image.
    6. Restart the node services (api-server, api-scheduler, controller-manager, kube-proxy).
    7. Restart the kubelet service.
    8. If any DaemonSet is deployed on a cluster:
        1. Force remove the DaemonSet pod.
        2. Reboot the node.
    9. Uncordon the node.
6. Complete the previous step on the remaining target ctl nodes one by one.
7. Upgrade add-ons if any on the target ctl nodes.
8. For the first Kubernetes target cmp node:
    1. Cordon the node.
    2. Drain the node.
    3. Regenerate SSL certificates for the node services.
    4. Start the containerd service.
    5. Upgrade the hyperkube image.
    6. Restart the kubelet service.
    7. If any DaemonSet is deployed on a cluster:
        1. Force remove the DaemonSet pod.
        2. Reboot the node.
    8. Uncordon the node.
9. Complete the previous step on the remaining target cmp nodes one by one.
10. Verify the Calico cluster version and integrity.

12(1, 2, 3) The etcd upgrade is added since the MCP 2019.2.3 maintenance update.

Manually upgrade Calico from version 2.6 to 3.3

This section describes the manual Calico upgrade procedure from major version 2.6 to 3.3. To simplify the upgrade process, use the automatic Calico upgrade procedure that is included into the Kubernetes upgrade pipeline. For details, see: Automatically update or upgrade Kubernetes.

> **Note**
>
> To update the minor Calico version (for example, from 3.1.x to 3.3.x), use the regular Kubernetes update procedure described in Update or upgrade Kubernetes.

The upgrade process implies the Calico-related services downtime for about 1-2 minutes on a virtual 5-node cluster. The downtime may vary depending on hardware and cluster configuration.

> **Caution!**
>
> This upgrade procedure is applicable when MCP is upgraded from Build ID 2018.8.0 to a newer MCP release version.
>
> MCP does not support the Calico upgrade path for the MCP Build IDs earlier than 2018.8.0.

The Kubernetes services downtime for workloads operations are caused by the necessity of the etcd schema migration where the Calico endpoints data and other configuration data is stored. Also, the calico-node and calico-kube-controllers components should have the same versions to operate properly, so there will be downtime while these components are being restarted.

To upgrade Calico from version 2.6 to 3.3:

1. Upgrade your MCP cluster to a newer Build ID as described in Upgrade DriveTrain to a newer release version. Once done, the version parameters and configuration files of the Calico components are updated automatically to the latest supported version.
2. Log in to any Kubernetes ctl node where etcd is running.
3. Migrate the etcd schema:
   1. Download the Calico upgrade binary file:

      ```
      wget https://github.com/projectcalico/calico-upgrade/releases/download/v1.0.5/calico-upgrade
      ```

   2. Grant execute permissions to the binary file:

      ```
      chmod +x ./calico-upgrade
      ```

3. Obtain the etcd endpoints:

```
salt-call pillar.get etcd:server:members
```

4. Export the etcd environment variables. For example:

```
export APIV1_ETCD_ENDPOINTS=https://10.70.2.101:4001,https://10.70.2.102:4001,https://10.70.2.103:4001
export APIV1_ETCD_CA_CERT_FILE=/var/lib/etcd/ca.pem
export APIV1_ETCD_CERT_FILE=/var/lib/etcd/etcd-client.crt
export APIV1_ETCD_KEY_FILE=/var/lib/etcd/etcd-client.key
export ETCD_ENDPOINTS=https://10.70.2.101:4001,https://10.70.2.102:4001,https://10.70.2.103:4001
export ETCD_CA_CERT_FILE=/var/lib/etcd/ca.pem
export ETCD_CERT_FILE=/var/lib/etcd/etcd-client.crt
export ETCD_KEY_FILE=/var/lib/etcd/etcd-client.key
```

Substitute APIV1_ETCD_ENDPOINTS and ETCD_ENDPOINTS with corresponding values.

5. Start the Calico upgrade:

```
./calico-upgrade start --no-prompts
```

> Note
>
> After executing this command, Calico pauses to avoid running into an incorrect data state.

4. Apply the new Calico configuration:

   1. Log in to the Salt Master node.
   2. Update basic Calico components:

   ```
   salt -C 'I@kubernetes:pool' state.sls kubernetes.pool
   ```

   3. Log in to the ctl node on which you started the Calico upgrade.
   4. Resume Calico after the etcd schema migration. For example:

   ```
   export APIV1_ETCD_ENDPOINTS=https://10.70.2.101:4001,https://10.70.2.102:4001,https://10.70.2.103:4001
   export APIV1_ETCD_CA_CERT_FILE=/var/lib/etcd/ca.pem
   export APIV1_ETCD_CERT_FILE=/var/lib/etcd/etcd-client.crt
   export APIV1_ETCD_KEY_FILE=/var/lib/etcd/etcd-client.key
   ```

```
export ETCD_ENDPOINTS=https://10.70.2.101:4001,https://10.70.2.102:4001,https://10.70.2.103:4001
export ETCD_CA_CERT_FILE=/var/lib/etcd/ca.pem
export ETCD_CERT_FILE=/var/lib/etcd/etcd-client.crt
export ETCD_KEY_FILE=/var/lib/etcd/etcd-client.key
./calico-upgrade complete --no-prompts
```

5. Log in to the Salt Master node.
6. Update the Kubernetes add-ons:

```
salt -C 'I@kubernetes:master' state.sls kubernetes.master.kube-addons
salt -C 'I@kubernetes:master' state.sls kubernetes exclude=kubernetes.master.setup
salt -C 'I@kubernetes:master' --subset 1 state.sls kubernetes.master.setup
```

7. Restart kubelet:

```
salt -C 'I@kubernetes:pool' service.restart kubelet
```

5. Log in to any ctl node.
6. Verify the Kubernetes cluster consistency:

   1. Verify the Calico version and Calico cluster consistency:

   ```
   calicoctl version
   calicoctl node status
   calicoctl get ipPool
   ```

   2. Verify that the Kubernetes objects are healthy and consistent:

   ```
   kubectl get node -o wide
   kubectl get pod -o wide --all-namespaces
   kubectl get ep -o wide --all-namespaces
   kubectl get svc -o wide --all-namespaces
   ```

   3. Verify the connectivity using Netchecker:

   ```
   kubectl get ep netchecker -n netchecker
   curl {{netchecker_endpoint}}/api/v1/connectivity_check
   ```

Seealso

Calico project documentation

Seealso

- Update Virtlet

Upgrade StackLight LMA to Build ID 2019.2.0

This section describes how to upgrade your Prometheus-based StackLight LMA from MCP Build ID 2018.11.0 to 2019.2.0. StackLight LMA does not have its own versioning schema within MCP and is versioned by MCP releases. To upgrade from older versions, see: MCP Release Compatibility Matrix: Supported upgrade paths.

> Warning
>
> During the upgrade, the existing monitoring services may be disrupted. Therefore, you must plan a maintenance window.

Prerequisites

Before you proceed with upgrading StackLight LMA, perform the following prerequisite steps:

1. Upgrade your MCP cluster as described in Upgrade DriveTrain to a newer release version.

   > Note
   >
   > If you are performing a manitenance update, verify that you have updated your MCP cluster as described in Update DriveTrain instead.

2. Log in to the Salt Master node.
3. Verify that there are no uncommitted changes on the cluster and system levels of the Reclass model:

   1. For the cluster level:

      ```
      cd /srv/salt/reclass/classes/cluster/<cluster_name>/
      git status
      ```

   2. For the system level:

      ```
      cd /srv/salt/reclass/classes/system
      git status
      ```

   Example of system response:

   ```
   On branch <branch_name>
   Your branch is up-to-date with 'origin/<branch_name>'.

   nothing to commit, working tree clean
   ```

   In case of any uncommitted changes, solve the issue before proceeding with the upgrade.

4. Verify that all Salt minions are available:

   ```
   salt '*' test.ping
   ```

   Example of system response:

   ```
   cmp0.<cluster_name>:
       True
   mon01.<cluster_name>:
       True
   gtw01.<cluster_name>:
       True
   ```

```
mon02.<cluster_name>:
    True
prx01.<cluster_name>:
    True
cmp1.<cluster_name>:
    True
mon03.<cluster_name>:
    True
cfg01.<cluster_name>:
    True
```

If any VM is not available, do not proceed before resolving the issue.

5. Verify that no alerts are triggered in the Alertmanager web UI as described in Use the Alertmanager web UI.

> **Caution!**
>
> If any alert is triggered, investigate the issue before proceeding with the upgrade. An alert may have a low severity and it is up to the system administrator to decide whether to proceed with the upgrade in this case.

6. Specify the elasticsearch_version: 6 and kibana_version: 6 parameters in the classes/cluster/<cluster_name>/stacklight/log.yml file of your Reclass model to upgrade Elasticsearch and Kibana from v5 to v6. Otherwise, Elasticsearch and Kibana will be upgraded to the latest minor stable versions.

Once done, proceed to Upgrade StackLight LMA using the Jenkins job.

Upgrade StackLight LMA using the Jenkins job

This section describes how to upgrade StackLight LMA using the Deploy - upgrade Stacklight Jenkins job.

> Warning
>
> Verify that you have completed the steps described in Prerequisites.

To upgrade StackLight LMA using the Jenkins job:

1. Log in to the Jenkins web UI.
2. Open the Deploy - upgrade Stacklight pipeline job.
3. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_URL	The URL of Salt API.
SALT_MASTER_CREDENTIALS	Credentials for Salt API stored in Jenkins.
STAGE_UPGRADE_SYSTEM_PART	Select to upgrade the system part including Telegraf, Fluentd, and Prometheus Relay.
STAGE_UPGRADE_ES_KIBANA	Select to upgrade Elasticsearch and Kibana.
STAGE_UPGRADE_DOCKER_COMPONENTS	Select to upgrade the StackLight LMA components running in Docker Swarm.

4. Click Build.
5. Click Full stage view to track the upgrade process.

   The following table contains the details of the upgrade stages:

The upgrade pipeline workflow

#	Stage	Details

<table>
<tr><td>1</td><td>Update grains and mines</td><td>Refreshes grains on all nodes by applying the salt.minion.grains state.
Refreshes modules on all nodes by running the saltutil.refresh_modules command.
Updates mines on all nodes by running the mine.update command.</td></tr>
<tr><td>2</td><td>Enable the Ceph Prometheus plugin</td><td>If Ceph is installed in the cluster, enables the Ceph Prometheus plugin by applying the ceph.mgr state on the Ceph Monitor nodes.</td></tr>
<tr><td>3</td><td>Upgrade system components</td><td>For each service including Telegraf, Fluentd, Prometheus Relay, libvirt-exporter, and jmx-exporter:
Prepares nodes with installed component, refreshes pillars and applies the linux.system.repo state.
Updates the packages to the latest versions.
Applies the corresponding Salt states for the changes to take effect.
Shows the statuses of the services after the upgrade.</td></tr>
<tr><td>4</td><td>Upgrade Elasticsearch and Kibana</td><td>For Elasticsearch:
Stops the service on all log nodes.
Upgrades the package to the newest version.
Reloads the systemd configuration.
Starts the service on all log nodes.
Verifies that the Elasticsearch cluster status is green.
In case of a major upgrade, transforms the indices for the new version.
For Kibana:
Stops the service on all log nodes.
Upgrades the package to the newest version.
Starts the service on all log nodes.
Shows the status of the service after the upgrade.
In case of a major upgrade, migrates Kibana to the new index.</td></tr>
</table>

5	Upgrade components running in Docker Swarm	1. Disables and removes the previous versions of monitoring services. 2. Rebuilds the Prometheus configuration by applying the prometheus state on the mon nodes. 3. Disables and removes the previous version of Grafana. 4. Starts the monitoring services by applying the docker state on the mon nodes. 5. Synchronizes Salt modules by applying the saltutil.sync_all state. 6. Applies the grafana.client state to refresh the Grafana dashboards.

Note

You may also enable additional functionality, such as Alerta or the Gainsight integration as required. For details, see Add new features to an existing StackLight LMA deployment.

Once done, proceed to Verify StackLight LMA after upgrade.

Verify StackLight LMA after upgrade

Once you have upgraded StackLight LMA as described in Upgrade StackLight LMA to Build ID 2019.2.0, verify that the upgrade succeeded and all StackLight LMA components are up and running.

To verify StackLight LMA after upgrade:

1. Inspect the execution logs of the Salt states for any errors.
2. Verify the availability of the Prometheus, Prometheus long-term storage, Grafana, Kibana, and Alertmanager web UIs as described in the corresponding sections of StackLight LMA operations. If you have deployed Alerta, also verify the Alerta web UI.
3. Verify the Prometheus targets in the Prometheus web UI.
4. Verify the Alertmanager alerts in the Alertmanager web UI.
5. Verify the data availability:
    1. Inspect the Main and Prometheus performances Grafana dashboards to verify that the data sources operate and metrics are available.
    2. Inspect the dashboards of the Kibana web UI for the No results found error message occurrence.
    3. In the Kibana web UI, filter and inspect the error messages, if any.

Upgrade Ceph

You can upgrade your existing Ceph cluster from Jewel to Luminous and from Luminous to Nautilus and roll back Ceph VMs and OSDs if the upgrade fails.

> Warning
>
> The upgrade of Ceph Luminous to Nautilus is supported starting from the 2019.2.10 maintenance update. If your Ceph version is Jewel, first upgrade to Ceph Luminous as described below.

Upgrade the Ceph cluster

This section describes how to upgrade an existing Ceph cluster from Jewel to Luminous and from Luminous to Nautilus. If your Ceph version is Jewel, you must first upgrade to Luminous before upgrading to Ceph Nautilus. The Ceph - upgrade pipeline contains several stages. Each node is upgraded separately and requires user input to verify if the status of the Ceph cluster is correct and if the upgrade of a Ceph node was successful. The upgrade procedure is performed on a node-by-node basis. In case of a failure, the user can immediately roll back each node.

> Note
>
> The following setup provides for the Ceph upgrade to a major version. To update the Ceph packages to the latest minor versions, follow Update Ceph.

> Warning
>
> Before you upgrade Ceph:
>
> 1. If Ceph is being upgraded as part of the MCP upgrade, verify that you have upgraded your MCP cluster as described in Upgrade DriveTrain to a newer release version.
> 2. Verify that you have configured the server and client roles for a Ceph backup as described in Create a backup schedule for Ceph nodes.
> 3. The upgrade of Ceph Luminous to Nautilus is supported starting from the 2019.2.10 maintenance update. Verify that you have performed the following steps:
>   1. Apply maintenance updates.
>   2. Enable the ceph-volume tool.
> 4. If you are upgrading Ceph from a version prior 14.2.20, verify that the ceph:common:config:mon:auth_allow_insecure_global_id_reclaim pillar is unset or set to true.

To upgrade the Ceph cluster:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In ceph/init.yml, specify the ceph_version parameter as required:
   - To upgrade from Jewel to Luminous:

     ```
     _param:
       ceph_version: luminous
     ```

   - To upgrade from Luminous to Nautilus:

```
_param:
  ceph_version: nautilus
  linux_system_repo_mcp_ceph_codename: ${_param:ceph_version}
```

3. Reconfigure package repositories:

   - To upgrade from Jewel to Luminous, in infra/init.yml, specify the linux_system_repo_update_mcp_ceph_url parameter:

     ```
     _param:
       linux_system_repo_update_mcp_ceph_url: ${_param:linux_system_repo_update_url}/ceph-luminous/
     ```

   - To upgrade from Luminous to Nautilus, remove the obsolete package repository by deleting all includes of system.linux.system.repo.mcp.apt_mirantis.ceph from the cluster model.

     The default files that include this class are as follows:

     - ceph/common.yml
     - openstack/init.yml
     - infra/kvm.yml

     Also, remove this class from non-default files of your cluster model, if any.

4. In ceph/mon.yml, verify that the following line is present:

   ```
   classes:
   - system.ceph.mgr.cluster
   ```

5. Commit the changes to your local repository:

   ```
   git add infra/init.yml
   git add ceph/init.yml
   git add ceph/mon.yml
   git commit -m "updated repositories for Ceph upgrade"
   ```

6. Refresh Salt pillars:

   ```
   salt '*' saltutil.refresh_pillar
   ```

7. Unset all flags and verify that the cluster is healthy.

   > Note
   >
   > Proceeding with some flags set on the cluster may cause unexpected errors.

8. Log in to the Jenkins web UI.

©2025, Mirantis Inc.

9. Open the Ceph - upgrade pipeline.
10. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
ADMIN_HOST	Add cmn01* as the Ceph cluster node with the admin keyring.
CLUSTER_FLAGS	Add a comma-separated list of flags to apply before and after the pipeline: • The sortbitwise,noout flags are mandatory for the upgrade of Ceph Jewel to Luminous. • The noout flag is mandatory for the upgrade of Ceph Luminous to Nautilus. • Specify the flags unset in the step 7. The flags will be automatically unset at the end of the pipeline execution.
WAIT_FOR_HEALTHY	Verify that this parameter is selected as it enables the Ceph health check within the pipeline.
ORIGIN_RELEASE	Add the current Ceph release version.
TARGET_RELEASE	Add the required Ceph release version.
STAGE_UPGRADE_MON	Select to upgrade Ceph mon nodes.
STAGE_UPGRADE_MGR	Select to deploy new mgr services or upgrade the existing ones.
STAGE_UPGRADE_OSD	Select to upgrade Ceph osd nodes.
STAGE_UPGRADE_RGW	Select to upgrade Ceph rgw nodes.
STAGE_UPGRADE_CLIENT	Select to upgrade Ceph client nodes, such as ctl, cmp, and others.
STAGE_FINALIZE	Select to set the configurations recommended for TARGET_RELEASE as a final step of the upgrade.

BACKUP_ENABLED	Select to copy the disks of Ceph VMs before upgrade and to back up Ceph directories on OSD nodes. Note During the backup, virtual machines are consequently backed up one after another - each VM is destroyed, the disk is copied and then the VM is started again. After a VM launches, the backup procedure is paused until the VM joins the Ceph cluster again and only then it continues to back up the other node. On OSD nodes, only the /etc/ceph and /var/lib/ceph/ directories are backed up. Mirantis recommends verifying that each OSD has been successfully upgraded before proceeding to the next one.
BACKUP_DIR Added since 2019.2.4 update	Optional. If BACKUP_ENABLED is selected, specify the target directory for the backup.

11. Click Deploy.

> Warning
>
> If the upgrade on the first node fails, stop the upgrade procedure and roll back the failed node as described in Roll back Ceph services.

12. In case of Ceph Luminous to Nautilus upgrade, run the Deploy - Upgrade Stacklight Jenkins pipeline job as described in Upgrade StackLight LMA using the Jenkins job to reconfigure StackLight for the new Ceph metrics.

> ## Caution!
>
> The Jenkins pipeline job changes repositories, runs upgrade packages, and restarts the service on each node of selected groups. As the pipeline installs packages from configured repositories and does not verify the version to install, for some environments the version of Ceph packages can differ between nodes after upgrade. It can affect one node due to manual configuration without a model or an entire group, like Ceph OSD nodes, due to a misconfiguration in Reclass. It is possible to run a cluster with mismatching versions. However, such configuration is not supported and, with a specific version, may cause cluster outage.

The Jenkins pipeline job provides information about packages versions change in the console output for each node. Consider checking them before proceeding to the next node, especially on the first node of each component.

The Ceph - upgrade pipeline workflow:

1. Perform the backup.
2. Set upgrade flags.
3. Perform the following steps for each selected stage for each node separately:
    1. Update Ceph repository.
    2. Upgrade Ceph packages.
    3. Restart Ceph services.
    4. Execute the verification command.
    5. Wait for user input to proceed.
4. Unset the upgrade flags.
5. Set ceph osd require-osd-release as TARGET_RELEASE.
6. Set ceph osd set-require-min-compat-client as ORIGIN_RELEASE.
7. Set CRUSH tunables to optimal.
8. If you enabled scrubbing and deep scrubbing before starting the upgrade, disable them specifying the ceph osd set noscrub and ceph osd set nodeep-scrub flags. Also, remove scrubbing settings if any.

Roll back Ceph services

You may need to roll back Ceph components to a previous version if the upgrade procedure fails. To successfully return Ceph to the previous version, you must manually roll back all Ceph components, including Ceph Monitor nodes, Ceph RADOS Gateway nodes, and Ceph OSD nodes. If the Ceph storage upgrade fails, you can roll back the Ceph VMs (the Monitor and RADOS Gateway nodes) and Ceph OSDs.

During the upgrade, the backed up Ceph VMs are copied to the root directory of the KVM nodes on which the Ceph nodes reside.

To roll back the Ceph VMs:

1. Log in to the KVM node.
2. Copy the backed up disks to the directory with libvirt. For example, to /var/lib/libvirt/images/cmn04.local/system.qcow2.
3. Start the instance:

   ```
   virsh start <NODE_NAME>.<NODE_DOMAIN>
   ```

To roll back the Ceph OSDs:

1. Manually install Ceph packages of the previous version:

   ```
   apt install <CEPH_PACKAGE_1>=<VERSION> <CEPH_PACKAGE_2>=<VERSION>
   ```

2. Restart services for all OSDs:

   ```
   systemctl restart ceph-osd@<osd_num>
   ```

If the step 1 fails, perform the following steps:

1. Purge Ceph packages:

   ```
   apt purge <CEPH_PACKAGE_1> <CEPH_PACKAGE_2>
   ```

2. Manually install Ceph packages of the previous version:

   ```
   apt install <CEPH_PACKAGE_1>=<VERSION> <CEPH_PACKAGE_2>=<VERSION>
   ```

3. Follow the steps described in Restore the metadata of a Ceph OSD node. Alternatively, if restoring from a backup is not possible:

   1. Purge Ceph packages:

      ```
      apt purge <CEPH_PACKAGE_1> <CEPH_PACKAGE_2>
      ```

   2. Verify that no Ceph devices are mounted:

```
umount <device_partition_path>
```

3. Run the following Salt states to redeploy the Ceph OSD node:

```
salt -C 'I@ceph:osd' state.sls ceph.osd
salt -C 'I@ceph:osd' saltutil.sync_grains
salt -C 'I@ceph:osd' state.sls ceph.osd.custom
salt -C 'I@ceph:osd' saltutil.sync_grains
salt -C 'I@ceph:osd' mine.update
salt -C 'I@ceph:setup' state.sls ceph.setup
```

Seealso

Update Ceph

Update an MCP cluster

Use the procedures in this section to apply minor updates to specific components of your MCP deployment with granular control requiring manual interventions.

Minor updates enable:

- Delivering hot fixes to source code of OpenStack, Kubernetes, or MCP Control Plane
- Updating packages for an OpenStack service
- Updating the OpenContrail 4.x nodes
- Applying security patches to operating system components and packages
- Updating Virtlet to minor versions within the scope of a major release version
- Updating Ceph packages to latest minor versions
- Updating StackLight LMA

> Note
>
> To update a Kubernetes cluster, refer to Update or upgrade Kubernetes.

The sequence of steps to be completed for the MCP components during a maintenance update is described in the Apply maintenance updates section of the corresponding MCP maintenance update version in MCP Release Notes: Maintenance updates.

Update local mirrors

If you use local mirrors in the MCP cluster, you must update the local mirror VM before updating DriveTrain to a minor release version. Otherwise, skip this section and proceed with Update DriveTrain.

You can update local mirrors either manually or by replacing the existing local mirror VM with the latest version.

To replace the existing local mirror VM:

> Warning
>
> This procedure implies recreation of the apt01 VM. Therefore, all existing customizations applied to the local mirror VM will be lost. To keep the existing customizations, use the manual procedure below instead.

1. Log in to the Salt Master node.

2. Download the latest version of the prebuilt http://images.mirantis.com/mcp-offline-image-<BUILD-ID>.qcow2 image for the apt01 node from http://images.mirantis.com.
3. Copy the new image to /var/lib/libvirt/images/apt01/.
4. Power off the existing apt01 VM instance:

   ```
   virsh shutdown apt01.<CLUSTER_DOMAIN>
   ```

5. If the local mirror VM is connected to the Salt Master node, temporarily remove it from the available minions:

   ```
   salt-key
   salt-key -d <local-mirror-node-name>
   ```

   > Note
   >
   > The local mirror VM will be automatically connected back to the Salt Master node once the updated VM is deployed.

6. Deploy the apt01 VM instance with the new image as described in MCP Deployment Guide: Deploy the APT node.
7. Once the local mirror VM with the new image is up and running, verify that your current Reclass system model has the correct origin set:

   ```
   cd /srv/salt/reclass/classes/system
   git remote -v
   ```

   Optional. You may set origin to your local mirror VM as it now has the latest update version of the Reclass system model:

   ```
   git remote remove origin
   git remote add origin http://<local_mirror_vm_ip>:8088/reclass-system.git
   ```

8. For an OpenContrail-based MCP cluster with the Build ID 2019.2.5 or earlier:
   1. Verify whether the MCP cluster uses internal_proxy:

      ```
      salt -C 'I@docker:host' pillar.get docker:host:proxy:enabled
      ```

      Depending on the system output, add the corresponding pillar in the next step.

2. Add the following pillar to /opencontrail/control.yml and opencontrail/analytics.yml:

   - For MCP clusters with internal_proxy enabled:

     ```
     parameters:
       ...
       docker:
         host:
           proxy:
             enabled: true
             http: ${_param:http_proxy}
             https: ${_param:http_proxy}
             no_proxy: ${linux:system:proxy:noproxy}
           insecure_registries:
             - ${_param:aptly_server_hostname}:5000
     ```

   - For MCP clusters with internal_proxy disabled:

     ```
     parameters:
       ...
       docker:
         host:
           insecure_registries:
             - ${_param:aptly_server_hostname}:5000
     ```

3. Commit the changes to your local repository.
4. Apply the changes:

   ```
   salt -C 'I@opencontrail:database' state.sls docker.host
   ```

Now, you can proceed with the step 2 of the Update DriveTrain procedure.

To update the local mirror VM manually:

> Note
>
> The procedure below requires access to the Internet.

1. Log in to the Salt Master node.
2. Verify the availability of the local mirror VM. For example:

   ```
   salt 'apt01.local-deployment.local' test.ping
   ```

If the VM does not respond, enable the management of the offline mirror VM through the Salt Master node as described in MCP Deployment Guide: Enable the APT node management in the Reclass model.

3. Update the system level of the Reclass model. For example, using the Git submodule:

> Note
>
> If the Reclass system model has git origin set to the internal mirror repository, update it first.

```
cd /srv/salt/reclass/ && git submodule foreach git fetch
cd /srv/salt/reclass/classes/system && git checkout origin/release/2019.2.0
```

4. Open the cluster level of your Reclass model.
5. In /infra/mirror/init.yml:

    - Add the Git server parameters:

    ```
    git:
      server:
        directory: /srv/git/
        repos: ${_param:default_local_mirrror_content:git_server_repos}
    ```

    - Include the debmirror content class:

    ```
    - system.debmirror.mirror_mirantis_com
    ```

    - Include the Docker registry class:

    ```
    docker:
      client:
        registry:
          target_registry: ${_param:default_local_mirrror_content:docker_client_registry_target_registry}
          image: ${_param:default_local_mirrror_content:docker_client_registry_image}
    ```

    - Include the static files class:

    ```
    linux:
      system:
        file: ${_param:default_local_mirrror_content:linux_system_file}
    ```

    - Include the MAAS mirror class:

```
maas:
  mirror:
    enabled: true
    image:
      sections: ${_param:default_local_mirrror_content:maas_mirror_image_sections}
```

6. For an OpenContrail-based MCP cluster with the Build ID 2019.2.5 or earlier:

    1. Verify whether the MCP cluster uses internal_proxy:

        ```
        salt -C 'I@docker:host' pillar.get docker:host:proxy:enabled
        ```

        Depending on the system output, add the corresponding pillar in the next step.

    2. Add the following pillar to /opencontrail/control.yml and opencontrail/analytics.yml:

        - For the MCP clusters with internal_proxy enabled:

            ```
            parameters:
              ...
              docker:
                host:
                  proxy:
                    enabled: true
                    http: ${_param:http_proxy}
                    https: ${_param:http_proxy}
                    no_proxy: ${linux:system:proxy:noproxy}
                  insecure_registries:
                    - ${_param:aptly_server_hostname}:5000
            ```

        - For the MCP clusters with internal_proxy disabled:

            ```
            parameters:
              ...
              docker:
                host:
                  insecure_registries:
                    - ${_param:aptly_server_hostname}:5000
            ```

7. Commit the changes to your local repository.
8. Synchronize the Salt modules:

    ```
    salt '<offline_node_name>' saltutil.sync_all
    ```

9. Apply the following states:

```
salt '<offline_node_name>' state.sls git.server
salt '<offline_node_name>' state.sls debmirror
salt '<offline_node_name>' state.sls docker.client.registry
salt '<offline_node_name>' state.sls linux.system.file
salt '<offline_node_name>' state.sls maas.mirror
```

10. For an OpenContrail-based MCP cluster, apply the changes made in the step 6:

```
salt -C 'I@opencontrail:database' state.sls docker.host
```

Now, you can proceed with the step 2 of the Update DriveTrain procedure.

Verify DriveTrain

> Note
>
> This feature is available starting from the MCP 2019.2.17 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

Using the Deploy - pre upgrade verify MCP Drivetrain Jenkins pipeline job, you can verify that your deployment model contains information about the necessary fixes and workarounds. Use the Jenkins pipeline job starting from the 2019.2.18 maintenance update.

To verify the deployment model before update:

1. Synchronize the upgrade pipeline jobs with the default parameters:
   1. Log in to the Jenkins web UI.
   2. Run the following Jenkins pipeline jobs with default parameters:
      - git-mirror-downstream-mk-pipelines
      - git-mirror-downstream-pipeline-library
2. Switch mcp-ci/pipeline-library to the version you are planning to update to. In branch value refs/tags/2019.2.X, substitute X with the desired maintenance update version. For example, if you want to apply maintenance update 2019.2.18, use the following command:

   ```
   salt -C "I@jenkins:client:lib" state.sls jenkins.client.lib pillar='{"jenkins": {"client": {"lib" : {"pipeline-library" : {"branch": "refs/tags/2019.2.18"}}}}}'
   ```

3. In the global view, find the Deploy - pre upgrade verify MCP Drivetrain pipeline.
4. Select the Build with Parameters option from the drop-down menu of the pipeline.
5. Specify the following parameters:

   Deploy - pre upgrade verify MCP Drivetrain parameters

Parameter	Description and values
MK_PIPELINES_REFSPEC	Set to the desired maintenance update version. For example, if your version is 2019.2.17 and you want to verify 2019.2.18, set MK_PIPELINES_REFSPEC to 2019.2.18.
PIPELINE_TIMEOUT	The time for the Jenkins job to complete, set to 1 hour by default. If the time exceeds, the Jenkins job will be aborted.
SALT_MASTER_CREDENTIALS	Jenkins credentials to access the Salt Master API.
SALT_MASTER_URL	The URL of the Salt Master node API.

6. Click Build.

7. Once the job finishes, open report.html in the job artifacts. Review the report and fix issues, if any.

Update DriveTrain

Within the scope of a major MCP release version, maintenance updates containing enhancements to the existing features, proactive security and critical bug fixes are being released with minor release versions. For details, see: MCP Release Notes: Maintenance updates. Before applying maintenance updates to an MCP cluster, the DriveTrain update is required.

To update DriveTrain to a minor release version:

1. For MCP clusters with local mirrors enabled, Update local mirrors.
2. If you are applying maintenance update 2019.2.15 or later, synchronize the upgrade pipeline jobs with the default parameters:
    1. Log in to the Jenkins web UI.
    2. Run the following Jenkins pipeline jobs with default parameters:
        - git-mirror-downstream-mk-pipelines
        - git-mirror-downstream-pipeline-library
3. Switch mcp-ci/pipeline-library to the version you are planning to update to. In the branch value refs/tags/2019.2.X, substitute X with the desired maintenance update version. For example, if you are applying maintenance update 2019.2.17, use the following command:

    ```
    salt -C "I@jenkins:client:lib" state.sls jenkins.client.lib pillar='{"jenkins": {"client": {"lib" : {"pipeline-library" : {"branch": "refs/tags/2019.2.17"}}}}}'
    ```

4. Follow the Upgrade DriveTrain to a newer release version procedure and the same Deploy - upgrade MCP DriveTrain Jenkins pipeline job:
    - If you are applying maintenance updates starting from version 2019.2.2 to 2019.2.8, consider specifying the following parameters in the DRIVE_TRAIN_PARAMS field. For later versions, the parameters are predefined and set to false by default.
        - OS_DIST_UPGRADE - do not set to true unless you want the system packages including kernel to be upgraded using apt-get dist-upgrade.
        - OS_UPGRADE - do not set to true unless you want all installed applications to be upgraded using apt-get upgrade.
    - If you are applying maintenance updates from any version to 2019.2.8 or later, select APPLY_MODEL_WORKAROUNDS to apply the cluster model workarounds automatically unless you manually added some patches to the model before the update.
    - Set the TARGET_MCP_VERSION parameter specifying the current MCP release version. For example, if your current MCP version is 2019.2.0, specify 2019.2.0.
    - Starting from 2019.2.8, alternatively, set the TARGET_MCP_VERSION, MK_PIPELINES_REFSPEC, and GIT_REFSPEC parameters to the desired maintenance update version. For example, to update from 2019.2.7 to 2019.2.8, specify 2019.2.8.

> Warning
>
> Starting from the maintenance update 2019.2.15, Jenkins update is run asynchronously. After you update DriveTrain, wait until Jenkins is updated and restarted several times. To verify that Jenkins update is finished, run the salt-run jobs.active command from the Salt Master node. Consider the update finished if the output does not include an active task named jenkins.client.

5. Optional. Upgrade system packages on the Salt Master node as described in the step 16 of the Upgrade to MCP release version 2019.2.0 procedure.
6. Update GlusterFS.

Once you update DriveTrain, proceed with applying maintenance updates to your MCP cluster components as required. For details, see: Update an MCP cluster.

Update GlusterFS

If you do not have any services that run on top of the GlusterFS volumes except the Docker Swarm services such as Jenkins, Gerrit, LDAP, you can use the Jenkins Update GlusterFS pipeline job to automatically update GlusterFS to the latest version. This pipeline job consequently executes the following GlusetFS update dedicated pipeline jobs with the default parameters:

- Update glusterfs servers
- Update glusterfs clients
- Update glusterfs cluster.op-version

For the procedure details and description of the pipeline jobs workflow, see Upgrade GlusterFS.

> **Caution!**
>
> Before you execute the Update GlusterFS pipeline job, complete steps 1-3 of the Upgrade GlusterFS procedure.

If you have any services that run on top of the GlusterFS volumes except the Docker Swarm services, Mirantis recommends updating the GlusterFS components separately using the dedicated pipeline jobs as described in Upgrade GlusterFS.

Update OpenStack packages

This section provides the reference information to consider when updating the OpenStack packages. Use the descriptive analysis of the techniques and tools, as well as the high-level upgrade flow included in this section to create a cloud-specific detailed update procedure, assess the risks, plan the rollback, backup, and testing activities.

> **Caution!**
>
> We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.

OpenStack update vs OpenStack upgrade

This section explains the differences between the OpenStack to a newer major release upgrade and OpenStack packages update.

The main purpose of update is to provide minor updates for the OpenStack packages without changing the major versions of packages.

Upgrade vs update

	Upgrade	Update
Package versions	Between the package versions supported by the sequent major OpenStack releases. For example, upgrade of Nova v15.0.1 (included in Ocata) to v16.0.1 (included in Pike).	Between the package versions within a single major OpenStack release. For example, update of Nova v15.0.1 (included in Ocata) to v15.2.0 (included in Ocata).
Database syncs	Performed and required	Can be performed but are not required
Control plane downtime	Required and expected	Not expected as the control plane nodes will be updated one by one

Limitations

The following are the limitations of the OpenStack upgrade pipelines that are used for the OpenStack packages update as well:

- The pipelines update only the OpenStack component. The update of other VCP components such as msg and dbs nodes is out of scope and should be done separately.
- The update of StackLight LMA is out of scope.

Prerequisites

Before you proceed with the OpenStack packages update, verify the following:

- Upgrade of MCP is done to the latest Build ID. Verify that you have updated DriveTrain including Aptly, Gerrit, Jenkins, Reclass, Salt formulas, and their subcomponents to the current MCP release version. Otherwise, the current MCP product documentation is not applicable to your MCP deployment.
- All OpenStack formulas states such as nova, neutron, and so on can be launched without errors.
- (Optional) Online dbsyncs for services are performed before the update maintenance window since this task can take a significant amount of time.
- No failed OpenStack services or nodes are present in the cloud.
- Utilization of the disk space is up to 80% on each target node, which include the ctl*, prx*, gtw*, and cmp* nodes.
- There is enough disk space on the node that will store the backups for the MySQL databases and the existing cluster model.

Update the OpenStack packages

Generally, the OpenStack upgrade and update procedures have common stages that include pre-upgrade/update, the control plane and data plane upgrade/update, and post-upgrade/update. Moreover, during both upgrading and updating, the same tooling is used. See OpenStack upgrade tools overview for details.

OpenStack packages update stages

#	Stage	Description
1	Planning	Includes the creation of the maintenance plan.
2	Pre-update	Includes procedures that do not affect workability of the current OpenStack cluster state such as running the QA cycle, verifying infrastructure, configuring monitoring of workloads and services. Also, during this stage, the backups are created and additional services, servers, and systems are installed to facilitate the update process according to the plan.
3	Update	The actual update.
3.1	Control plane update	The control plane nodes are being updated. It should not have an impact on the data plane, and compatibility issues between the data plane and control plane of different minor versions are not expected. No downtime of the control plane is expected, as update is performed one by one for the control plane nodes which are in HA.
3.2	Data plane update	The data plane nodes that host the end-user data applications including the compute, storage, and gateway nodes are being updated.
4	Post update	Includes procedures that will not affect workability, such as post-update testing activities, and cleanup.

Warning

Before you perform the update on a production environment, accomplish the procedure on a staging environment. If the staging environment does not exist, adapt the exact cluster model and launch it inside the cloud as a heat stack, which will act as a staging environment.

©2025, Mirantis Inc.

Plan the OpenStack update

As a result of the planning stage of the OpenStack update, a detailed maintenance plan is created.

The maintenance plan must include the following parts:

- A strict step-by-step update procedure
- A rollback plan
- A maintenance window schedule for each update phase

> Note
>
> The update flow is thoroughly selected by engineers in correspondence with the workload and requirements of a particular cloud.

After the maintenance plan is successfully tested on a staging environment, you can proceed with the actual update in production.

Perform the pre-update activities

The pre-update stage includes the activities that do not affect workability of a currently running OpenStack version as well as the backups creation.

To prepare your OpenStack deployment for the update:

1. (Optional) On one of the controller nodes, perform the online database migrations for the following services:

   > Note
   >
   > The database migrations can be time-consuming and create high load on CPU and RAM. We recommend that you perform the migrations in batches.

   - Nova:

     ```
     nova-manage db online_data_migrations
     ```

   - Cinder:

     ```
     cinder-manage db online_data_migrations
     ```

   - Ironic:

     ```
     ironic-dbsync online_data_migrations
     ```

2. Prepare the target nodes for the update:

   1. Log in to the Salt Master node.
   2. Get the list of all updatable OpenStack components:

      ```
      salt-call config.get orchestration:upgrade:applications --out=json
      ```

      Example of system response:

      ```
      {
         "<model_of_salt_master_name>": {
            "nova": {
               "priority": 1100
            },
            "heat": {
               "priority": 1250
            },
            "keystone": {
               "priority": 1000
      ```

```
    },
    "horizon": {
      "priority": 1800
    },
    "cinder": {
      "priority": 1200
    },
    "glance": {
      "priority": 1050
    },
    "neutron": {
      "priority": 1150
    },
    "designate": {
      "priority": 1300
    }
  }
}
```

3. Range the components from the output by priority. For example:

```
keystone
glance
nova
neutron
cinder
heat
designate
horizon
```

4. Get the list of all target nodes:

```
salt-key | grep $cluster_domain | \
grep -v $salt_master_hostname | tr '\n' ' '
```

The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters.

The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master.

5. For each target node, get the list of installed applications:

```
salt <node_name> pillar.items __reclass__:applications --out=json
```

6. Match the lists of updatable OpenStack components with the lists of installed applications for each target node.
7. During update, the applications running on the target nodes use the KeystoneRC metadata. To guarantee that the KeystoneRC metadata is exported to mine, verify that you apply the keystone.upgrade.pre formula to the keystone:client:enabled node:

```
salt -C 'I@keystone:client:enabled' state.sls keystone.upgrade.pre
```

8. Apply the following states to each target node for each installed application in the strict order of priority:

```
salt <node_name> state.apply <component_name>.upgrade.pre
salt <node_name> state.apply <component_name>.upgrade.verify
```

For example, for Nova installed on the cmp01 compute node, run:

```
salt cmp01 state.apply nova.upgrade.pre
salt cmp01 state.apply nova.upgrade.verify
```

Note

On the clouds of medium and large sizes, you may want to automate this step. Use the following script as an example of possible automatization.

```
 #!/bin/bash
#List of formulas that implements upgrade API sorted by priority
all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
#List of nodes in cloud
list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
for node in $list_nodes; do
  #List of applications on the given node
  node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.pre
      salt $node state.apply $component.upgrade.verify
    fi
  done
done
```

3. Add the testing workloads to each compute host and monitoring and verify the following:
   - The cloud services are monitored as expected.
   - There are free resources (disk, RAM, CPU) on the kvm, ctl, cmp, and other nodes.

4. (Optional) Back up the OpenStack databases as described in Back up and restore a MySQL database.
5. Adjust the cluster model:
    1. Include the upgrade pipeline job to DriveTrain:
        1. Add the following lines to cluster/cicd/control/leader.yml:

            Caution!

            If your MCP OpenStack deployment includes the OpenContrail component, do not specify the system.jenkins.client.job.deploy.update.upgrade_ovs_gateway class.

            ```
            classes:
             - system.jenkins.client.job.deploy.update.upgrade
             - system.jenkins.client.job.deploy.update.upgrade_ovs_gateway
             - system.jenkins.client.job.deploy.update.upgrade_compute
            ```
        2. Apply the jenkins.client state on the Jenkins nodes:

            ```
            salt -C 'I@jenkins:client' state.sls jenkins.client
            ```
    2. Set the parameters in classes/cluster/<cluster_name>/infra/init.yml as follows:

        ```
        parameters:
          _param:
            openstack_upgrade_enabled: true
        ```
    3. (Optional) Upgrade pillars of all supported OpenStack applications are already included in the Reclass system level. In case of a non-standard setup, the list of the OpenStack applications on each node should be checked and upgrade pillars added for the Openstack applications that do not contain them. For example:

        ```
        <app>:
          upgrade:
            enabled: ${_param:openstack_upgrade_enabled}
        ```

> Note
>
> On the clouds of medium and large sizes, you may want to automate this step. To obtain the list of the OpenStack applications running on a node, use the following script.
>
> ```
> #!/bin/bash
> #List of formulas that implements upgrade API sorted by priority
> all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
> jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
> sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
> #List of nodes in cloud
> list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
> for node in $list_nodes; do
>   #List of applications on the given node
>   node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
>   jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
>   node_openstack_app=" "
>   for component in $all_formulas ; do
>     if [[ " ${node_applications[*]} " == *"$component"* ]]; then
>       node_openstack_app="$node_openstack_app $component"
>     fi
>   done
>   echo "$node : $node_openstack_app"
> done
> ```

4. Refresh pillars:

```
salt '*' saltutil.refresh_pillar
```

6. Prepare the target nodes for the update:

   1. Get the list of all updatable OpenStack components:

   ```
   salt-call config.get orchestration:upgrade:applications --out=json
   ```

   2. Range the components from the output by priority.
   3. Get the list of all target nodes:

   ```
   salt-key | grep $cluster_domain | \
   grep -v $salt_master_hostname | tr '\n' ' '
   ```

   The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters

   The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master

   4. For each target node, get the list of installed applications:

```
salt <node_name> pillar.items __reclass__:applications --out=json
```

5. Match the lists of updatable OpenStack components with the lists of installed applications for each target node.
6. Apply the following states to each target node for each installed application in strict order of priority:

```
salt <node_name> state.apply <component_name>.upgrade.pre
```

Note

On the clouds of medium and large sizes, you may want to automate this step. Use the following script as an example of possible automatization.

```
 #!/bin/bash
#List of formulas that implements upgrade API
all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
for node in $list_nodes; do
  #List of applications on the given node
  node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.pre
    fi
  done
done
```

7. Apply the linux.system.repo state on the target nodes:

```
salt <node_name> state.apply linux.system.repo
```

8. Proceed to Update the OpenStack control plane.

Update the OpenStack control plane

The OpenStack control plane update stage includes updating of the OpenStack services APIs.

> **Caution!**
>
> Perform the update of the VCP nodes one by one to avoid downtime of the control plane.

To update the OpenStack VCP:

1. Log in to the Jenkins web UI.
2. Perform the update of the OpenStack controller nodes one by one. For each OpenStack controller node, run the Deploy - upgrade control VMs pipeline in the interactive mode setting the parameters as follows:
   - TARGET_SERVERS='<full_ctl_node_name>' where full_ctl_node_name can be, for example, ctl01.domain.local
   - MODE=INTERACTIVE mode to get the detailed description of the pipeline flow through the stages
   - OS_DIST_UPGRADE - do not select unless you want the system packages including kernel to be upgraded using apt-get dist-upgrade. Deselected by default.
   - OS_UPGRADE - do not select unless you want all installed applications to be upgraded using apt-get upgrade. Deselected by default.
3. Verify that the VCP is up and the OpenStack services from the data plane are reconnected and working correctly with the newly updated OpenStack controller nodes. For example, to verify the OpenStack services after updating the ctl01 node, apply the following states:

   ```
   salt 'ctl01*' state.apply heat.upgrade.verify
   salt 'ctl01*' state.apply nova.upgrade.verify
   salt 'ctl01*' state.apply cinder.upgrade.verify
   salt 'ctl01*' state.apply glance.upgrade.verify
   salt 'ctl01*' state.apply neutron.upgrade.verify
   ```

   For details on how to get the list of updatable OpenStack components and verify their status on other nodes, see Perform the pre-update activities.
4. Run the Deploy - upgrade control VMs on each of the proxy nodes one by one setting TARGET_SERVERS='<full_prx_node_name>', where full_prx_node_name> can be, for example, prx01.domain.local.
5. Verify that the public API is accessible and Horizon is working.
6. Perform the update of other VCP nodes where required depending on your deployment by running the Deploy - upgrade control VMs pipeline in the interactive mode setting the parameters as follows:

- TARGET_SERVERS:
    - TARGET_SERVERS=share* to upgrade the Manila control plane
    - TARGET_SERVERS=mdb* to upgrade the Tenant Telemetry including Ceilometer, Gnocchi, Aodh, and Panko
    - TARGET_SERVERS=kmn* to upgrade Barbican
    - TARGET_SERVERS=dns* to upgrade Designate
- MODE=INTERACTIVE mode to get the detailed description of the pipeline flow through the stages
- OS_DIST_UPGRADE - do not select unless you want the system packages including kernel to be upgraded using apt-get dist-upgrade. Deselected by default.
- OS_UPGRADE - do not select unless you want all installed applications to be upgraded using apt-get upgrade. Deselected by default.

7. Verify that the control plane is up and the OpenStack services from the data plane are reconnected and working correctly with the newly updated control plane.
8. From an OpenStack controller node, clean up the old heat-engine records:
    1. Verify that all Heat engines have started and the number of active Heat engines matches the number of OpenStack controller nodes multiplied by the number of Heat engine workers per node:

       ```
       heat-manage service list
       ```

    2. Remove all old, stopped engines:

       ```
       heat-manage service clean
       ```

9. Verify that the OpenStack packages have been updated as expected. For example, to verify the packages after updating the ctl01 node with OS_UPGRADE selected, run the following command:

   ```
   salt 'ctl01*' cmd.run 'apt-get -s -V -u upgrade'
   ```

   To verify the packages after updating the ctl01 node with OS_DIST_UPGRADE selected:

   ```
   salt 'ctl01*' cmd.run 'apt-get -s -V -u dist-upgrade'
   ```

10. Proceed to Update the OpenStack data plane.

Seealso

MCP 2019.2.3 Maintenance Update: Known issues

©2025, Mirantis Inc.

Update the OpenStack data plane

The OpenStack data plane includes the servers that host end-user data applications. More specifically, these hosts include compute, storage, and gateway nodes. Depending on the update requirements, you can apply any kind of the update depths while updating the data plane.

To update the data plane of your OpenStack deployment:

1. Update the gateway nodes depending on your use case:

   > **Caution!**
   >
   > Skip this step if your MCP OpenStack deployment includes the OpenContrail component because such configuration does not contain gateway nodes.

   * If non-HA routers are present in the cloud:

     1. Migrate the non-HA routers from the target nodes using the Neutron service and the following commands in particular:
        * l3-agent-router-add
        * l3-agent-router-remove
        * router-list-on-l3-agent
     2. Log in to the Jenkins web UI.
     3. Run the Deploy - upgrade OVS gateway pipeline for the gateway nodes which you have migrated the workloads from.

        > **Note**
        >
        > Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

        > **Note**
        >
        > Since all resources have already been migrated from the nodes, we recommend performing the full upgrade including OS_UPGRADE and OS_DIST_UPGRADE.

     4. Migrate the non-HA routers back and rerun the Deploy - upgrade OVS gateway pipeline for the rest of the gateway nodes.

- If non-HA routers are not present in the cloud:
    1. Log in to the Jenkins web UI.
    2. Run the Deploy - upgrade OVS gateway pipeline for all gateway nodes specifying TARGET_SERVERS='gtw*'.

        Note

        Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

2. Verify that the gateway components are reconnected to the control plane.

    Caution!

    Skip this step if your MCP OpenStack deployment includes the OpenContrail component because such configuration does not contain gateway nodes.

3. Update the OpenStack compute nodes.
    1. Estimate and minimize the risks and address the limitations of the live migration.

        The limitations of the live migration technology include:

        Warning

        Before proceeding with the live migration in a production environment, assess these risks thoroughly.

        - The CPU of a source compute node must have a feature set that is a subset of a feature set of the target compute CPU. Therefore, the migration should be performed between the compute nodes with identical CPUs with, preferably, identical microcode versions.
        - During the live migration, the entire memory state of a VM must be copied to another server. In the first place, the memory pages that are being changed at a slower rate are copied. After, the system copies the most active memory pages. If the number of pages that are being written to is big, the migration process will never finish. High-memory, high-load Windows virtual machines are known to have this particular issue.

  - During the live migration, a very short downtime (1-2 seconds max) occurs. The reason for the downtime is that when the memory is copied, the execution context (vCPU state) has to be copied as well, and the execution itself must be switched to a new virtual machine. In addition to a short downtime, this causes a short clock lag on the migrated virtual machine. Therefore, if the migrated machine is hosting a part of a clustered service or system, the downtime and resulting time lag may have an adverse impact on the whole system.
  - The QEMU version installed on the source and target hosts should be the same and later than 2.5.

   2. Perform the live migration of workloads.
   3. Log in to the Jenkins web UI.
   4. Run the Deploy - upgrade computes pipeline to update the OpenStack compute nodes which you have migrated the workloads from. It is essential that you update the compute nodes by small batches.

      > Caution!
      >
      > The impact of the update process should be calculated for each compute node during the planning stage as this step may take a significant amount of time.

      > Note
      >
      > Run the pipeline in the interactive mode to get the detailed description of the pipeline flow through the stages.

   5. Migrate the workloads back and rerun the Deploy - upgrade computes pipeline for the rest of the compute nodes.

4. Verify that the compute nodes are reconnected to the control plane.
5. Proceed to Perform the post-update activities.

©2025, Mirantis Inc.

Perform the post-update activities

The post-update activities include the post-update testing cycle and cleanup.

To finalize the update:

1. Perform the full verification cycle of your MCP OpenStack deployment.
2. Verify that the following variables are set in the classes/cluster/<cluster_name>/infra/init.yml file:

   ```
   parameters:
     _param:
       openstack_upgrade_enabled: false
   ```

3. Refresh pillars:

   ```
   salt '*' saltutil.refresh_pillar
   ```

4. Remove the test workloads/monitoring.
5. Remove the update leftovers that were created by applying the <app>.upgrade.post state:

   1. Log in to the Salt Master node.
   2. Get the list of all updatable OpenStack components. For example:

      ```
      salt-call config.get orchestration:upgrade:applications --out=json
      ```

      Example of system response:

      ```
      {
         "<model_of_salt_master_name>": {
            "nova": {
               "priority": 1100
            },
            "heat": {
               "priority": 1250
            },
            "keystone": {
               "priority": 1000
            },
            "horizon": {
               "priority": 1800
            },
            "cinder": {
               "priority": 1200
            },
            "glance": {
               "priority": 1050
      ```

```
    },
    "neutron": {
      "priority": 1150
    },
    "designate": {
      "priority": 1300
    }
  }
}
```

3. Range the components from the output by priority. For example:

```
keystone
glance
nova
neutron
cinder
heat
designate
horizon
```

4. Get the list of all target nodes:

```
salt-key | grep $cluster_domain | \
grep -v $salt_master_hostname | tr '\n' ' '
```

   The cluster_domain variable stands for the name of the domain used as part of the cluster FQDN. For details, see MCP Deployment guide: General deployment parameters: Basic deployment parameters.

   The salt_master_hostname variable stands for the hostname of the Salt Master node and is cfg01 by default. For details, see MCP Deployment guide: Infrastructure related parameters: Salt Master.

5. For each target node, get the list of installed applications:

```
salt <node_name> pillar.items __reclass__:applications --out=json
```

6. Match the lists of the updatable OpenStack components with the lists of installed applications for each target node.
7. Apply the following states to each target node for each installed application in the strict order of priority:

```
salt <node_name> state.apply <component_name>.upgrade.post
```

   For example, for Nova installed on the cmp01 compute node, run:

```
salt cmp01 state.apply nova.upgrade.post
```

Note

On the clouds of medium and large sizes, you may want to automate this step. Use the following script as an example of possible automatization.

```
 #!/bin/bash
#List of formulas that implements upgrade API sorted by priority
all_formulas=$(salt-call config.get orchestration:upgrade:applications --out=json | \
jq '.[] | . as $in | keys_unsorted | map ({"key": ., "priority": $in[.].priority}) | sort_by(.priority) | map(.key | [(.)]) | add' | \
sed -e 's/"//g' -e 's/,//g' -e 's/\[//g' -e 's/\]//g')
#List of nodes in cloud
list_nodes=`salt -C 'I@__reclass__:applications' test.ping --out=text | cut -d: -f1 | tr '\n' ' '`
for node in $list_nodes; do
  #List of applications on the given node
  node_applications=$(salt $node pillar.items __reclass__:applications --out=json | \
  jq 'values |.[] | values |.[] | .[]' | tr -d '"' | tr '\n' ' ')
  for component in $all_formulas ; do
    if [[ " ${node_applications[*]} " == *"$component"* ]]; then
      salt $node state.apply $component.upgrade.post
    fi
  done
done
```

©2025, Mirantis Inc.

Update Galera

To update the Galera cluster, use the Upgrade Galera procedure. Only the MySQL and Galera packages will be updated. The update of an underlying operating system is out of scope.

Update RabbitMQ

To update the RabbitMQ component, use the Upgrade RabbitMQ procedure and the same Deploy - upgrade RabbitMQ server pipeline job.

> **Caution!**
>
> We recommend that you do not upgrade or update OpenStack and RabbitMQ simultaneously. Upgrade or update the RabbitMQ component only once OpenStack is running on the new version.

©2025, Mirantis Inc.

Update the OpenContrail 4.x nodes

> **Caution!**
>
> Before proceeding with the upgrade procedure, verify that you have updated DriveTrain including Aptly, Gerrit, Jenkins, Reclass, Salt formulas, and their subcomponents to the current MCP release version. Otherwise, the current MCP product documentation is not applicable to your MCP deployment.

This section describes how to apply minor updates to the OpenContrail 4.x nodes, for example, from version 4.0 to 4.1.

To update the OpenContrail 3.2 packages, refer to the Upgrade the OpenContrail nodes to version 3.2 procedure.

> **Caution!**
>
> OpenContrail 4.x for Kubernetes 1.12 or later is not supported.

Prerequisites

Before you start updating the OpenContrail nodes of your MCP cluster, complete the following prerequisite steps:

1. Configure the server and client backup roles for Cassandra and ZooKeeper as described in OpenContrail 4.x: Create a backup schedule for a Cassandra database and OpenContrail 4.x: Create a backup schedule for a ZooKeeper database.
2. Verify that all OpenContrail services are up and running on all OpenContrail nodes. See Verify the OpenContrail status.

   > Note
   >
   > If you update MCP from version 2019.2.3, the contrail-webui service may be in the inactive state due to the missing quotation mark in /etc/contrail/config.global.js. To fix the issue, see: MCP 2019.2.3 maintenance updates.

3. Verify that you have enough free disk space on the OpenContrail ntw* and nal* nodes since the update pipeline job will download a new version of Docker images. The required disk utilization must not exceed 80% on each target node.

Once done, proceed to Prepare the cluster model.

Prepare the cluster model

After you complete the prerequisite steps, prepare your cluster model for the update by configuring your Git project repository as described below.

To prepare the cluster model:

1. Log in to the Salt Master node.
2. In cluster/<name>/opencontrail/init.yml file, update the OpenContrail version. For example:

   ```
   _param:
     linux_repo_contrail_component: oc41
     opencontrail_version: 4.1
   ```

3. In cluster/<name>/openstack/dashboard.yml, update the OpenContrail version. For example:

   ```
   _param:
     opencontrail_version: 4.1
   ```

4. In cluster/<name>/openstack/init.yml, verify that the following parameters contain correct values:

   ```
   _param:
     opencontrail_admin_password: <contrail_user_password>
     opencontrail_admin_user: 'contrail'
   ```

5. If you update OpenContrail to version 4.1, in cluster/<name>/opencontrail/analytics.yml, add or change the following parameters:

   ```
   _param:
     opencontrail_kafka_config_dir: '/etc/kafka'
     opencontrail_kafka_log_dir: '/var/log/kafka'
   ```

6. Add the OpenContrail update pipeline job:

   1. Add the following class to cluster/cicd/control/leader.yml:

      ```
      classes:
      - system.jenkins.client.job.deploy.update.update_opencontrail4
      ```

   2. Apply the jenkins.client state:

      ```
      salt -C 'I@jenkins:client' state.sls jenkins.client
      ```

Once done, proceed to Update the OpenContrail nodes.

Update the OpenContrail nodes

After you prepare the cluster model of your MCP cluster, proceed to updating the OpenContrail controller nodes and the OpenContrail vRouter packages on the compute nodes.

> Warning
>
> During the update process, the following resources are affected:
>
> - The instance(s) running on the compute nodes can be unavailable for up to 30 seconds.
> - The creation of new instances is not possible during the same time interval.
>
> Therefore, you must plan a maintenance window as well as test the update on a staging environment before applying it to production.

To update the OpenContrail nodes:

1. Log in to the Jenkins web UI.
2. Verify that you do not have any unapproved scripts in Jenkins:
    1. Navigate to Manage Jenkins > In-process script approval.
    2. Approve pending scripts if any.
3. Open the Deploy - update Opencontrail 4X pipeline.
4. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
STAGE_CONTROLLERS_UPDATE	Select to update the OpenContrail controller nodes.

5. Click Deploy.

To update the OpenContrail vRouter packages on the compute nodes:

1. Log in to the Jenkins web UI.
2. Open the Deploy - update Opencontrail 4X pipeline.
3. Specify the following parameters:

Parameter	Description and values

COMPUTE_TARGET_SERVERS	Add I@opencontrail:compute or target the global name of your compute nodes, for example cmp001*.
COMPUTE_TARGET_SUBSET_LIVE	Add 1 to run the update first on only one of the nodes defined in the COMPUTE_TARGET_SERVERS field. After this stage is done, in the pipeline job console, you will be asked to continue the update the remaining nodes defined in the COMPUTE_TARGET_SERVERS field.
SALT_MASTER_CREDENTIALS	The Salt Master credentials to use for connection, defaults to salt.
SALT_MASTER_URL	The Salt Master node host URL with the salt-api port, defaults to the jenkins_salt_api_url parameter. For example, http://172.18.170.27:6969.
STAGE_COMPUTES_UPDATE	Select to update the compute nodes.

4. Click Deploy. For details how to monitor the deployment process, see: MCP Deployment Guide: View the deployment details.
5. Log in to the Salt Master node.
6. Restart the vrouter-agent and vrouter-nodemgr services on the OpenContrail compute nodes:

```
salt -C 'I@opencontrail:compute' service.restart contrail-vrouter-agent
salt -C 'I@opencontrail:compute' service.restart contrail-vrouter-nodemgr
```

The Deploy - update Opencontrail 4X pipeline workflow:

1. If STAGE_COMPUTES_UPDATE is selected, update the OpenContrail packages on the compute nodes in two iterations:
    1. Update the sample nodes defined by COMPUTE_TARGET_SUBSET_LIVE.
    2. After a manual confirmation, update all compute nodes targeted in COMPUTE_TARGET_SERVERS.
2. If STAGE_CONTROLLERS_UPDATE is selected, download a new version of controller, analytics, and analytics db containers.
3. Stop the running analytics and analytics db containers on nal nodes and start the updated containers.
4. Stop the running controller containers on ntw nodes and start the updated containers.

Seealso

MCP 2019.2.3 Maintenance Update: Known issues

Update Virtlet

This section describes how to install maintenance updates, for example, hot fixes, to Virtlet by updating its version to the latest supported one.

During the update, the Virtlet VM can not be accessible using the kubectl attach command. But it still has the network connectivity. The MCP cluster workload is not affected.

To update Virtlet:

1. Open your project Git repository with Reclass model on the cluster level.
2. Change the Virtlet version in /kubernetes/compute.yml:

   ```
   parameters:
     kubernetes:
       common:
         addons:
           virtlet:
             enabled: true
             namespace: kube-system
             image: mirantis/virtlet:<virtlet_version>
   ```

3. Log in to the Salt Master node.
4. Apply the following states:

   ```
   salt -C 'I@kubernetes:master' state.sls kubernetes.master.kube-addons
   salt -C 'I@kubernetes:master' state.sls kubernetes.master.setup
   ```

5. Verify that Virtlet is updated successfully:
   1. Log in to any Kubernetes Master node.
   2. Run the following command:

      ```
      kubectl rollout status -n kube-system ds/virtlet
      ```

      > Note
      >
      > Substitute virtlet with a custom name if the default virtlet name was changed.

> Seealso
>
> Update or upgrade Kubernetes

Obtain Ubuntu security updates

You can obtain security updates for the host operating system packages of your OpenStack-based MCP cluster starting from the Build ID 2018.8.0 using the update repositories. The update repositories provide for update of the specific Ubuntu packages for the CI/CD, OpenStack, and StackLight LMA nodes.

To obtain Ubuntu security updates, proceed with one of the following:

- For existing deployments:
    1. Add the update repositories.
    2. Apply the security updates as described below either manually or using the dedicated Jenkins pipeline.
- For new deployments, starting from the Q3`18 Release Version, the update repositories are already included. Therefore, proceed with either Apply security updates manually or Apply Ubuntu security updates using Jenkins right away.

> **Caution!**
>
> Due to a limitation, do not upgrade the cfg01 node.

> **Caution!**
>
> Mirantis recommends that you plan a maintenance window due to rebooting of the nodes. The maintenance window time depends on the number of nodes, vRouters, and the complexity of workloads migration and is approximately two hours without regard to migration. During this period, MCP control plane may not be accessible. Network interrupts may occur during the namespaces rebuild because the gateway nodes will be rebooted. Workloads may be interrupted if you reboot the compute nodes.

Add an update repository

This section describes how to add an update repository for the existing MCP deployments starting from the Build ID 2018.8.0 to allow for obtaining the Ubuntu security updates.

To add an update repository:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In the classes/cluster/<cluster_name>/infra/init.yml file, specify the following parameters:

```
parameters:
  _param:
    linux_system_repo_update_url: http://mirror.mirantis.com/update/${_param:mcp_version}/
    linux_system_repo_update_ubuntu_url: ${_param:linux_system_repo_update_url}/ubuntu/
  linux:
    system:
      repo:
        ubuntu_security_update:
          refresh_db: ${_param:linux_repo_refresh_db}
          source: "deb [arch=amd64] ${_param:linux_system_repo_update_ubuntu_url} ${_param:linux_system_codename}-security main restricted universe"
          architectures: amd64
          default: true
        mcp_saltstack:
          pin:
          - enabled: true
            pin: 'release o=SaltStack'
            priority: 50
            package: 'libsodium18'
```

3. Log in to the Salt Master node.
4. Apply the following state:

```
salt "*" state.sls linux.system.repo
```

Once done, proceed to Apply security updates manually.

Apply security updates manually

Once you have created an update repository as described in Add an update repository, or if you already have an update repository, you can apply the security updates.

This section instructs you on how to update Ubuntu packages manually. More specifically, the procedure includes update of the KVM nodes one by one, the VMs running on top of each KVM node, and the OpenStack compute nodes.

If you prefer the automated update, use the Apply Ubuntu security updates using Jenkins procedure.

> **Note**
>
> This documentation does not cover the upgrade of Ceph nodes.

To apply the Ubuntu security updates manually:

1. Log in to the Salt Master node.
2. For OpenStack Ocata with OpenContrail v3.2, update the python-concurrent.futures package. Otherwise, skip this step.

   ```
   salt -C "ntw* or nal*" cmd.run "apt-get -y --force-yes upgrade python-concurrent.futures"
   ```

3. Update the Virtual Control Plane and KVM nodes:

   1. Identify the KVM nodes:

      ```
      salt "kvm*" test.ping
      ```

      Example of system response:

      ```
      kvm01.bud.mirantis.net:
          True
      kvm03.bud.mirantis.net:
          True
      kvm02.bud.mirantis.net:
          True
      ```

      In the example above, three KVM nodes are identified: kvm01, kvm02, and kvm03.

   2. Identify the VMs running on top of each KVM node. For example, for kvm01:

      ```
      salt "kvm01*" cmd.run "virsh list --name"
      ```

Example of system response:

```
kvm01.bud.mirantis.net:
    cfg01
    prx01.<domain_name>
    ntw01.<domain_name>
    msg01.<domain_name>
    dbs01.<domain_name>
    ctl01.<domain_name>
    cid01.<domain_name>
    nal01.<domain_name>
```

3. Using the output of the previous command, upgrade the cid, ctl, gtw/ntw/nal, log, mon, msg, mtr, dbs, and prx VMs of the particular KVM node. Do not upgrade cfg01, cmp, and kvm.

   Note

   Ceph nodes are out of the scope of this procedure.

   Example:

```
for NODE in prx01.<domain_name> \
            ntw01.<domain_name> \
            msg01.<domain_name> \
            dbs01.<domain_name> \
            ctl01.<domain_name> \
            cid01.<domain_name> \
            nal01.<domain_name>
do
  salt "${NODE}*" cmd.run "export DEBIAN_FRONTEND=noninteractive && \
                apt-get update && \
                apt-get -y upgrade && \
                apt-get -y -o Dpkg::Options::="--force-confdef" \
                        -o Dpkg::Options::="--force-confnew" dist-upgrade"
done
```

4. Wait for all services of the cluster to be up and running.

   1. If the KVM node hosts GlusterFS, verify the GlusterFS server and volumes statuses as described in Troubleshoot GlusterFS. Proceed with further steps only if the GlusterFS status is healthy.
   2. If the KVM node hosts a dbs node, verify that the Galera cluster status is Synced and contains at least three nodes as described in Verify a Galera cluster status.

   3. If the KVM node hosts a msg node, verify the RabbitMQ cluster status and that it contains at least three nodes as described in Troubleshoot RabbitMQ.
   4. If the KVM node hosts OpenContrail 4.x, verify that its services are up and running as described in Verify the OpenContrail status. If any service fails, troubleshoot it as required. For details, see: Troubleshoot OpenContrail.
5. Once you upgrade the VMs running on top of a KVM node, upgrade and restart this KVM node itself. For example, kvm01:

```
salt "kvm01*" cmd.run "export DEBIAN_FRONTEND=noninteractive && \
                 apt-get update && \
                 apt-get -y upgrade && \
                 apt-get -y -o Dpkg::Options::="--force-confdef" \
                           -o Dpkg::Options::="--force-confnew" dist-upgrade && \
                 shutdown -r 0"
```

6. Wait for all services of the cluster to be up and running. For details, see substeps of the step 3.4.
7. Repeat the steps 3.1-3.7 for the remaining kvm nodes one by one.

4. Upgrade the OpenStack compute nodes using the example below, where cmp10. is a set of nodes.

Caution!

Before upgrading a particular set of cmp nodes, migrate the critical cloud environment workloads, which should not be powered off, from these nodes to the cmp nodes that are not under maintenance.

Example:

```
# Check that the command will be applied to only needed nodes
salt -E "cmp10." test.ping
# Perform the upgrade
salt -E "cmp10." cmd.run "export DEBIAN_FRONTEND=noninteractive && apt-get \
update && apt-get -y upgrade && apt-get -y -o \
Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confnew" \
dist-upgrade && apt-get -y install linux-headers-generic && shutdown -r 0"
```

Apply Ubuntu security updates using Jenkins

Once you have created an update repository as described in Add an update repository, or if you already have an update repository, you can apply the security updates.

This section instructs you on how to update Ubuntu packages automatically. If you prefer the manual update, use the Apply security updates manually procedure.

To apply the Ubuntu security updates automatically:

1. Log in to the Jenkins web UI.
2. Run the Deploy - update system package(s) pipeline specifying the following parameters as required:

Deploy - update system package(s) pipeline parameters

Parameter	Description
SALT_MASTER_URL	Full Salt API address, for example, https://10.10.10.1:8000
SALT_MASTER_CREDENTIALS	Credentials to the Salt API
TARGET_SERVERS	Salt compound target to match nodes to be updated. For example, [*, G@osfamily:debian].
TARGET_PACKAGES	RSpace delimited list of packages to be updated. For example, [package1=version package2=version]. The empty string means updating all packages to the latest version.
BATCH_SIZE Added since 2019.2.6 update	The batch size for Salt commands targeted for a large amount of nodes. Set to an absolute number of nodes (integer) or percentage, for example, 20 or 20%. For details, see MCP Deployment Guide: Configure Salt Master threads and batching.

Update Ceph

Starting from the Build ID 2019.2.0, you can update Ceph packages to the latest minor versions on the Ceph OSD, Monitor, and RADOS Gateway nodes using the Update Ceph packages pipeline job. During the update, all Ceph services restart one by one and the pipeline does not proceed until the cluster is healthy.

> Warning
>
> If you are updating Ceph from a version prior 14.2.20, verify that the ceph:common:config:mon:auth_allow_insecure_global_id_reclaim pillar is unset or set to true.

> Note
>
> This procedure does not include the backup of Ceph VMs. To back up the Ceph VMs, perform the steps described in Back up and restore Ceph.

To update Ceph packages to the latest minor versions:

1. Verify that you have upgraded your MCP cluster to the latest Build ID as described in Upgrade DriveTrain to a newer release version.
2. Log in to the Jenkins web UI.
3. Open the Update Ceph packages pipeline job.
4. Specify the following parameters:

Parameter	Description and values
SALT_MASTER_CREDENTIALS	Credentials for Salt API stored in Jenkins.
SALT_MASTER_URL	The URL of Salt API. For example, https://10.10.10.1:8000.
TARGET_SERVERS Removed since 2019.2.5 update	Salt compound target to match the nodes to be updated. For example, G@osfamily:debian or * to update the packages on all Ceph nodes.
CLUSTER_FLAGS Added since 2019.2.7 update	Add a comma-separated list of flags to apply before and after the pipeline execution.

5. Click Build.
6. Click Full stage view to track the update process.

> **Caution!**
>
> The Jenkins pipeline job changes repositories, runs upgrade packages, and restarts the service on each node of selected groups. As the pipeline installs packages from configured repositories and does not verify the version to install, for some environments the version of Ceph packages can differ between nodes after upgrade. It can affect one node due to manual configuration without a model or an entire group, like Ceph OSD nodes, due to a misconfiguration in Reclass. It is possible to run a cluster with mismatching versions. However, such configuration is not supported and, with a specific version, may cause cluster outage.
>
> The Jenkins pipeline job provides information about packages versions change in the console output for each node. Consider checking them before proceeding to the next node, especially on the first node of each component.

The following table contains the details of the update stages:

The Update Ceph packages pipeline job workflow

#	Stage	Details
1	List the target servers	1. Selects the Ceph nodes from the available targets. 2. Obtains the minions from TARGET_SERVERS to list all available targets and then selects the ones that include rgw, cmn, or osd in the name.
2	Apply package upgrades on all nodes	Updates and installs new Ceph packages on the selected nodes.
3	Restart the services	1. Restarts Ceph Monitor on all cmn nodes one by one. 2. Starting from the 2019.2.8 update, restarts Ceph Manager on all mgr nodes one by one. 3. Restarts Ceph OSDs on all osd nodes one by one. After a service restart on each node, the Jenkins pipeline job waits for the cluster to become healthy.

Update StackLight LMA

You can apply maintenance updates to your StackLight LMA deployment within the scope of a major MCP release version.

To update StackLight LMA within the scope of a major MCP release version, use the Upgrade StackLight LMA to Build ID 2019.2.0 procedure and the same Deploy - upgrade Stacklight pipeline job. Before updating StackLight LMA, you may need to perform manual steps or select specific pipeline job parameters. For details on a particular maintenance update, see: MCP Release Notes: Maintenance updates.

Cloud verification

MCP enables the deployment, QA, support engineers, and cloud operators to perform functional as well as non-functional types of testing of cloud environments through DriveTrain using the Cloud Verification Pipeline (CVP) tooling.

CVP is a set of tools, procedures, and components allowing for automatic MCP deployments verification. CVP can be applied to the newly built deployments to verify the functionality, performance, and fault tolerance of cloud environments as well as to the exisiting deployments to verify their functionality and performance before and after any changes.

The main characteristics of CVP inlude:

- Non-destructive testing
- Resources cleanup after test run
- Fully automated test runs
- Granular configurable testing
- Extensible tests enabling the operator to add custom tests in the PyTest format and extend the Tempest test framework for OpenStack functional tests

CVP requires the Internet connection to the following URLs:

- https://github.com (for repositories)
- https://files.pythonhosted.org and https://pypi.org (for pip packages)
- http://download.cirros-cloud.net (preferable but not required)

If HTTP or HTTPS proxies are in use, they must be present in the Docker daemon configuration.

CVP pipelines work in the offline mode but some adjustments may be needed for MCP pipelines.

CVP pipelines

Jenkins pipeline	Description	Source code (GitHub)
CVP - Sanity checks	Sanity testing to verify the cloud platform infrastructure components	Mirantis/cvp-sanity-checks
CVP - Functional tests	OpenStack functional integration testing (Tempest)	openstack/tempest Mirantis/cvp-configuration
CVP - Performance tests	OpenStack Rally-based baseline performance testing	Mirantis/cvp-configuration
CVP - HA tests	OpenStack high availability testing	openstack/tempest Mirantis/cvp-configuration
CVP - StackLight tests	StackLight LMA basic verification	Mirantis/stacklight-pytest

©2025, Mirantis Inc.

CVP - Shaker network tests	Data plane networking test	Mirantis/cvp-shaker

CVP pipelines workflow

#	Details
1	Enable CVP in the Reclass model if required.
2	Adjust a test set to the deployment-specific requirements if needed.
3	Configure a related Jenkins pipeline.
4	Build the pipeline.
5	Review test results.

This section explains how to perform different types of cloud verification using corresponding Jenkins pipelines.

Enable CVP pipelines in the deployment model

Cloud verification pipelines are included in the MCP DriveTrain by default. If your deployment does not contain CVP pipelines, you can enable the pipelines in your Reclass model as described in this section. You can also schedule a recurring build job for CVP pipelines as required.

To enable CVP pipelines:

1. Log in to the Salt Master node.
2. In the classes/cluster/<CLUSTER_NAME>/cicd/control/leader.yml file, add the following class:

   ```
   - system.jenkins.client.job.validate
   ```

3. Apply the jenkins.client.job state:

   ```
   salt 'cid01*' state.apply jenkins.client.job
   ```

To schedule a recurring build for a job:

1. Log in to the Salt Master node.
2. Modify the classes/cluster/<CLUSTER_NAME>/cicd/control/leader.yml file as required. For example:

   ```
   jenkins:
     client:
       job:
         cvp-sanity:
           trigger:
             timer:
               spec: 'H 8 * * *'
   ```

   > Note
   >
   > The spec field follows the cron syntax. For more details, see: TimerTrigger.

3. Apply the jenkins.client.job state for the cid01* node:

   ```
   salt 'cid01*' state.apply jenkins.client.job
   ```

Applying the change in the example above schedules the cvp-sanity job execution to 8 am UTC every day.

Perform sanity testing

Sanity tests can be used for basic verification of your MCP deployment helping to troubleshoot infrastructure health and verify whether the enviroment is ready for further testing. You can perform the sanity testing of your environment using the CVP - Sanity checks Jenkins pipeline.

Execute the CVP - Sanity checks pipeline

This section instructs you on how to perform the sanity testing of your deployment using the CVP - Sanity checks Jenkins pipeline.

To perform the sanity testing of your deployment:

1. In a web browser, open http://<ip_address>:8081 to access the Jenkins web UI.

   Note

   The IP address is defined in the classes/cluster/<cluster_name>/cicd/init.yml file of the Reclass model under the cicd_control_address parameter variable.

2. Log in to the Jenkins web UI as admin.

   Note

   To obtain the password for the admin user, run the salt "cid*" pillar.data _param:jenkins_admin_password command from the Salt Master node.

3. In the global view, find the CVP - Sanity checks pipeline.
4. Select the Build with Parameters option from the drop-down menu of the pipeline.
5. Configure the following parameters as required:

CVP - Sanity checks parameters

Parameter	Description
DEBUG_MODE Removed since 2019.2.4 update	If checked, keeps the container (if the IMAGE parameter is defined) after the test is performed for the debugging purposes. This option is deprecated and ignored starting the Q4`18 MCP release.
IMAGE	Specifies the cvp-sanity Docker image (with all dependencies) that will be used during the test run. For offline mode, use the URL from the local artifactory or offline image.
PROXY Removed since 2019.2.4 update	If an environment uses HTTP or HTTPS proxy, verify that you specify it in this field as this proxy address will be used to clone the required repositories and install the Python requirements. This option is deprecated and ignored starting the Q4`18 MCP release.

SALT_MASTER_CREDENTIALS	Specifies the credentials to Salt API stored in Jenkins, included by default. See View credentials details used in Jenkins pipelines.
SALT_MASTER_URL	Specifies the reachable IP address of the Salt Master node and port on which Salt API listens. For example, http://172.18.170.28:6969. To determine on which port Salt API listens: 1. Log in to the Salt Master node. 2. Search for the port in the /etc/salt/master.d/_api.conf file. 3. Verify that the Salt Master node is listening on that port: `netstat -tunelp | grep <PORT>`
TESTS_REPO Removed since 2019.2.4 update	Specifies the repository with the sanity tests that can be either a Github URL or an internal Gerrit repository with custom tests. By default, the value for this parameter is empty. Since this option is deprecated and ignored starting the Q4`18 release, Mirantis recommends leaving this field empty.
TESTS_SET Removed since 2019.2.4 update	Specifies the name of the test to perform or directory to discover tests. By default, it is cvp-sanity/cvp_checks/tests. Leave the field as is for a full test run or specify the test filename. For example, the <default_path>/test_repo_list.py will only run the test for repositories. If a custom image is used and the TESTS_REPO field is empty, provide the full path to the folder with tests.
TESTS_SETTINGS Removed since 2019.2.4 update	Specifies additional environment variables that can be passed to the framework. You can override configuration values with these variables. For example, export skipped_nodes=mtr01.local,log02.local will force the job to skip the specified nodes in all tests. See global_config.yaml for details.

EXTRA_PARAMS Added since 2019.2.4 update	Specifies additional environment variables in the YAML format using the envs key. For example:
	<pre>envs:
	  - tests_set=tests/test_drivetrain.py
	  - skipped_nodes=mtr01.local,log02.local</pre>
	You can override configuration values with these variables. For example, skipped_nodes=mtr01.local,log02.local will force the job to skip the specified nodes in all tests. See global_config.yaml for details. Do not use quotes and spaces in the values of parameters.
	Added since 2019.2.5 Use the force_pull parameter to enable or disable the pull operation for an image before running the container. Set force_pull=false if image pulling is impossible or not required. In this case, the image may be manually uploaded to the target node.
	Added since 2019.2.7 You can override the configuration values using the override_config variable. For example:
	<pre>override_config:
	  skipped_nodes:
	    - log02
	    - apt03</pre>

Note

To perform the DriveTrain sanity testing:

- For MCP versions starting from the 2019.2.4 maintenance update, specify the EXTRA_PARAMS parameters as described in Perform DriveTrain sanity testing.
- For MCP versions before the 2019.2.4 maintencance update, specify the TESTS_SET and TESTS_SETTINGS parameters as described in Perform DriveTrain sanity testing.

6. Click Build.
7. Verify the job status:

    - GREEN, SUCCESS

        Testing has beed performed successfully, no errors found.

    - YELLOW, UNSTABLE

        Some errors occurred during the test run. Proceed to Review the CVP sanity test results.

- RED, FAILURE

  Testing has failed due to issues with the framework or/and pipeline configuration. Review the console output.

Perform DriveTrain sanity testing

This section instructs you on how to perform the DriveTrain sanity testing of your deployment using the CPV - Sanity checks Jenkins pipeline. The sanity checks for DriveTrain include testing of the Gerrit, Jenkins, and OpenLDAP functionality, as well as the DriveTrain services replicas, components and versions, and the Jenkins job branch.

To perform the DriveTrain sanity testing:

1. Perform the steps 1-4 as described in Execute the CVP - Sanity checks pipeline.
2. Using the step 5 of the same procedure, specify the IMAGE, SALT_MASTER_CREDENTIALS, SALT_MASTER_URL parameters.
3. Configure the following parameters as required:

Parameter	Description
TESTS_SET Removed since 2019.2.4 update	Specify cvp-sanity/cvp_checks/tests/test_drivetrain.py to perform only the DriveTrain sanity testing.
TESTS_SETTINGS Removed since 2019.2.4 update	drivetrain_version=<your_mcp_version>

EXTRA_PARAMS Added since 2019.2.4	Specify tests/test_drivetrain.py to perform only the DriveTrain sanity testing. For example:
	`envs:`
	`  - tests_set=tests/test_drivetrain.py`
	Added since 2019.2.7 You can override the configuration values using the override_config variable. For example:
	`override_config:`
	`  skipped_nodes:`
	`    - log02`
	`    - apt03`

4. Perform the remaining steps of the Execute the CVP - Sanity checks pipeline procedure.

The CPV - Sanity checks pipeline workflow for DriveTrain:

1. Run the test_drivetrain_openldap job:
   1. Obtain the LDAP server and port from the _param:haproxy_openldap_bind_port and _param:haproxy_openldap_bind_host pillars.
   2. Obtain the LDAP admin password from the _param:openldap_admin_password pillar.
   3. Using the admin user name from the openldap:client:server:auth:user pillar and password from the previous step, connect to the LDAP server.
   4. Create a test user DT_test_user.

   5. Add the created test user to the admins group.
   6. Using the test user credentials, try obtaining the information about the cvp-sanity job.
   7. Using the test user credentials, try connecting to the Gerrit server and check the list of patches created by this user.
   8. Delete the user from the admins group and from the LDAP server.
2. Run the test_drivetrain_jenkins_job job:
   1. Obtain the Jenkins server and port from the _param:haproxy_jenkins_bind_port and _param:haproxy_jenkins_bind_host pillars.
   2. Obtain the Jenkins admin password from the _param:openldap_admin_password pillar.
   3. Connect to Jenkins using the credentials from the previous steps.
   4. Execute <jenkins_test_job> on the cid node.

      > Note
      >
      > The pipeline checks whether a DT-test-job exists and if it does not exist - creates a new empty DT-test-job job. You can specify another job using the jenkins_test_job option in the TESTS_SETTINGS parameter.

   5. Wait 180 seconds for the SUCCESS status of the job.
3. Run the test_drivetrain_gerrit job:
   1. Obtain the Gerrit server and port from the _param:haproxy_gerrit_bind_port and _param:haproxy_gerrit_bind_host pillars.
   2. Obtain the Gerrit admin password from the _param:openldap_admin_password pillar.
   3. Connect to Gerrit using the credentials from the previous steps.
   4. Add a new test-dt-<current_date> project.
   5. Create a new file and send it on review.
   6. As an admin user, add Code-Review +2 to the patch and merge it.
   7. Delete the test project.
4. Run the test_drivetrain_services_replicas job:
   1. Obtain the list of Docker services from the cid node.
   2. Compare the number of expected and actual number of replicas.
5. Run the test_drivetrain_components_and_versions job:
   1. Obtain the list of Docker services from the cid node.
   2. Compare the list of expected and actual list of the services.

3. Compare the version of the service with the drivetrain_version version.

> Note
>
> Prior to the 2019.2.4 update The test will be skipped if drivetrain_version was not defined in the TESTS_SETTINGS parameter.

6. Run the test_jenkins_jobs_branch job:

   1. Obtain the list of all jobs from Jenkins.
   2. Compare the versions of the jobs with the drivetrain_version version.

> Note
>
> Prior to the 2019.2.4 update The test will be skipped if drivetrain_version was not defined in the TESTS_SETTINGS parameter.
>
> In MCP Build ID 2019.2.0, this test will fail due to an issue with all Jenkins jobs having the release/2019.2.0 version instead of 2019.2.0. This issue has been fixed since the 2019.2.2 maintenance update.
>
> The master version for the deploy-update-salt job is an expected behavior.

Review the CVP sanity test results

This section explains how to review the CVP Sanity checks trace logs from the Jenkins web UI.

To review the CVP Sanity checks results:

1. Log in to the Jenkins web UI.
2. Navigate to the build that you want to review.
3. Find Test Results at the bottom of the Build page.
4. Scroll down and click Show all failed tests to view a complete list of the failed tests.
5. Click on the test of concern for details.

   > Note
   >
   > A test name corresponds to the test file path in the test source repository. For example:
   >
   > - For MCP versions starting from the 2019.2.4 maintenance update, tests.test_mtu.test_mtu[mtr] corresponds to cvp-sanity/tests/test_mtu.py.
   > - For MCP versions before the 2019.2.4 maintenance update, cvp_checks.tests.test_mtu.test_mtu[mtr] corresponds to cvp-sanity-checks/cvp_checks/tests/test_mtu.py.

6. Review the lines beginning with the E letter in the trace.

   The following example illustrates the error for the mtr01 node, the mtu value for ens2 interface for which differs from other MTUs in the mtr group (1500 versus 1450).

```
        for node in total:
            nodes.append(node)
            my_set.update(total[node].keys())
        for interf in my_set:
            diff = []
            row = []
            for node in nodes:
                if interf in total[node].keys():
                    diff.append(total[node][interf])
                    row.append("{}: {}".format(node, total[node][interf]))
                else:
                    row.append("{}: No interface".format(node))
            if diff.count(diff[0]) < len(nodes):
                row.sort()
                row.insert(0, interf)
                mtu_data.append(row)
>       assert len(mtu_data) == 0, \
            "Several problems found for {0} group: {1}".format(
            group, json.dumps(mtu_data, indent=4))
E       AssertionError: Several problems found for mtr group: [
E             [
E                 "ens2",
E                 "mtr01.kkb.ezweb.ne.jp: 1450",
E                 "mtr02.kkb.ezweb.ne.jp: 1500",
E                 "mtr03.kkb.ezweb.ne.jp: 1500"
E             ]
E         ]
E       assert 1 == 0
```

Note

If you do not see the full log trace, use the Console output -> View as plain text Jenkins menu instead.

Perform OpenStack functional integration testing

The CVP - Functional tests pipeline performs functional integration testing of the OpenStack environments. The pipeline includes the OpenStack Tempest testing.

> Note
>
> The pipeline is provided as is. It contains default settings, examples, and templates that may need adjustment for a deployed environment. Mirantis does not support the third-party components like Tempest or Rally used in the CVP tooling.

Execute the CVP - Functional tests pipeline

This section instructs you on how to perform the OpenStack functional integration testing using the CVP - Functional tests Jenkins pipeline.

> Note
>
> Clone the cvp-configuration and tempest repositories to your local Gerrit and use them locally to add new or adjust the existing tests and fine-tune the Tempest configuration.

To perform functional integration testing of your deployment:

1. In your local cvp-configuration repository, inspect and modify the following items as required:

   - The Tempest configuration (tempest/tempest_ext.conf)
   - The skip list (tempest/skip-list* files)
   - The configure.sh setup script

   Tempest 18.0.0 and Rally 0.11.2 are default versions for CVP - Functional tests Jenkins pipeline job in the offline and online mode.

2. In a web browser, open http://<ip_address>:8081 to access the Jenkins web UI.

   > Note
   >
   > The IP address is defined in the classes/cluster/<cluster_name>/cicd/init.yml file of the Reclass model under the cicd_control_address parameter variable.

3. Log in to the Jenkins web UI as admin.

   > Note
   >
   > To obtain the password for the admin user, run the salt "cid*" pillar.data _param:jenkins_admin_password command from the Salt Master node.

4. In the global view, find the CVP - Functional tests pipeline.
5. Select the Build with Parameters option from the drop-down menu of the pipeline.
6. Configure the following parameters as required:

   CVP - Functional tests parameters

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>DEBUG_MODE</td><td>If checked, keeps the container after test is performed for the debugging purposes.</td></tr>
<tr><td>PROXY</td><td>If an environment uses HTTP or HTTPS proxy, verify that you specify it in this field as this proxy address will be used to clone the required repositories and install the Python requirements. For the offline mode, specify offline, no additional packages or modules will be pulled from the Internet.</td></tr>
<tr><td>SALT_MASTER_CREDENTIALS</td><td>Specifies the credentials to Salt API stored in Jenkins, included by default. See View credentials details used in Jenkins pipelines.</td></tr>
<tr><td>SALT_MASTER_URL</td><td>Specifies the reachable IP address of the Salt Master node and port on which Salt API listens. For example, http://172.18.170.28:6969
To determine on which port Salt API listens:
1. Log in to the Salt Master node.
2. Search for the port in the /etc/salt/master.d/_api.conf file.
3. Verify that the Salt Master node is listening on that port:
netstat -tunelp | grep <PORT>

Caution!
In the 2019.2.4 update, by default, the HTTPS (SSL) NGINX URL is used as SALT_MASTER_URL. For long-running pipelines, it may lead to a timeout error, usually after 10 minutes. Therefore, specify the Salt API native non-SSL URL as described above to prevent timeout errors. Starting from the 2019.2.5 update, Salt API native non-SSL URL is used by default.</td></tr>
<tr><td>SKIP_LIST_PATH</td><td>Optional. Specifies the path to the tempest skip list file located in the repository specified in TOOLS_REPO. The default value is cvp-configuration/tempest/skip-list.yaml. For the offline mode, when cvp-rally image is used, define /var/lib/cvp-configuration/tempest/skip-list.yaml. If you use a custom image, specify the path to the custom skip list file.</td></tr>
</table>

©2025, Mirantis Inc.

TARGET_NODE	Specifies the node to run the container with Tempest/Rally. Use the Jenkins slave as it has the Docker package. Note Starting from the MCP 2019.2.2 update, if the TARGET_NODE parameter is empty, the node with the gerrit:client pillar will be used, which is cid01 by default.
TEMPEST_ENDPOINT_TYPE	Sets the type of the Openstack endpoint to use during the test run.
TEMPEST_REPO	Specifies the Tempest repository to clone and use. By default, it is the upstream Tempest. Though, you can specify your customized tempest in a local or remote repository. For the full offline mode, specify /var/lib/tempest (a path inside a container) and cvp-rally image.
TEMPEST_TEST_PATTERN	Specifies the tests to run. See the Rally documentation for all available options.
TEST_IMAGE	Specifies the link to the Docker Rally-based image to use for running the container with testing tools. We recommend using the upstream Rally image xrally/xrally-openstack:0.11.2. For the full offline mode, use the cvp-rally image from the local Docker images mirror (Registry) or pull docker-prod-local.docker.mirantis.net/mirantis/cvp/cvp-rally:<mcp_version> locally.
TOOLS_REPO	Includes the URL or path to the repository where testing tools, scenarios, and configurations are located. By default, it is https://github.com/Mirantis/cvp-configuration. Specify your MCP version here. For example, -b release/2019.2.0 for Q4`18.  Caution! For the Q4`18 MCP release, the branch name format is release/2019.2.0 for the cvp-configuraiton repository. The old 2019.2.0 format is deprecated. To customize the configuration, clone the cvp-configuration repository to your local Gerrit and commit the changes. For the offline mode, specify /var/lib/cvp-configuration/configure.sh. Alternatively, use this repository from the offline image. If your image is fully configured, leave the field empty.

7. Click Build.

8. Verify the job status:

   - GREEN, SUCCESS

     Testing has beed performed successfully, no errors found.

   - YELLOW, UNSTABLE

     Some errors occurred during the test run. Proceed to Review the CVP - Functional tests pipeline results.

   - RED, FAILURE

     Testing has failed due to issues with the framework or/and pipeline configuration. Review the console output.

Review the CVP - Functional tests pipeline results

This section explains how to review the CVP Functional tests trace logs from the Jenkins web UI.

To review the CVP Functional tests results:

1. Log in to the Jenkins web UI.
2. Navigate to the build that you want to review.
3. Find Test Results at the bottom of the Build page.
4. Scroll down and click Show all failed tests to view the complete list of the failed tests.
5. Click on the test of concern for details.

> Note
>
> If + does not expand, use a different browser or a public URL for your Jenkins.

Add new tests to the CVP - Functional tests pipeline

This section includes the instruction on how to extend the predefined OpenStack functional integration tests.

To add custom functional integration tests:

1. Clone the openstack/tempest Git repository to your local Gerrit repository.
2. Check out the tempest repository from your local Gerrit.
3. If required, create a new directory.
4. Create a file for the new test or update the existing test file.
5. Add the test code. Refer to Tempest Test Writing Guide in the OpenStack official documentation.
6. Commit the changes to the local Gerrit.
7. To perform the newly added test, run the CVP - Functional tests pipeline as described in Execute the CVP - Functional tests pipeline, specifying the TEMPEST_TEST_PATTERN parameter according to your test/test class/suite name.

   > Note
   >
   > Verify, that the TEMPEST_REPO value matches your local Gerrit copy of tempest.

Perform OpenStack performance (load) testing

MCP DriveTrain enables you to perform the Rally-based baseline performance testing of your MCP OpenStack deployment using the CVP - Performance tests Jenkins pipeline.

> Note
>
> We recommend using the Rally image v0.11.2.
>
> If you need to customize Rally test scenarios, refer to the official Rally documentation.

> Note
>
> The pipeline is provided as is. It contains default settings, examples, and templates that may need adjustment for a deployed environment. Mirantis does not support the third-party components like Tempest or Rally used in the CVP tooling.

Install the Performance Jenkins plugin

Before you proceed, verify that the Performance Jenkins plugin is installed.

To install the Performance Jenkins plugin:

1. Log in to the Jenkins web UI.
2. Navigate to Manage Jenkins > Manage Plugins.
3. Under the Available tab, search for the Performance plugin, and check it.
4. Click Install without restart.

Execute the CVP - Performance tests pipeline

This section instructs you on how to perform the OpenStack performance (load) testing of your deployment using the CVP - Performance tests Jenkins pipeline.

> Note
>
> Clone the cvp-configuration to your local Gerrit and use it locally to add new or adjust the existing Rally scenarios.

To perform the OpenStack performance testing:

1. In your local cvp-configuration repository, inspect and modify the following items as required:
    - The Rally scenarios (rally/rally_scenarios* files)
    - The configure.sh setup script
2. In a web browser, open http://<ip_address>:8081 to access the Jenkins web UI.

    > Note
    >
    > The IP address is defined in the classes/cluster/<cluster_name>/cicd/init.yml file of the Reclass model under the cicd_control_address parameter variable.

3. Log in to the Jenkins web UI as admin.

    > Note
    >
    > To obtain the password for the admin user, run the salt "cid*" pillar.data _param:jenkins_admin_password command from the Salt Master node.

4. In the global view, find the CVP - Performance tests pipeline.
5. Select the Build with Parameters option from the drop-down menu of the pipeline.
6. Configure the following parameters as required:

CVP - Performance tests parameters

Parameter	Description

DEBUG_MODE	If checked, keeps the container after test is performed for the debugging purposes.
PROXY	If an environment uses HTTP or HTTPS proxy, verify that you specify it in this field as this proxy address will be used to clone the required repositories and install the Python requirements. For the offline mode, specify offline, no additional packages or modules will be pulled from the Internet.
RALLY_SCENARIO_FILE	Specifies the path to the Rally scenarios file located in the repository specified in TOOLS_REPO. The default value is cvp-configuration/rally/rally_scenarios.json. For the offline mode, when cvp-rally image is used, define /var/lib/cvp-configuration/rally/rally_scenarios.json. If you use a custom image, specify the path to the custom skip list file. The cvp-configuration repository contains a default set of scenarios for a dry run in rally/rally_scenarios.json and the same set but with 100 iterations and 10 threads in rally/rally_scenarios_100.json. More advanced scenarios with floating IPs and live migration are available in rally/rally_scenarios_fip_and_ubuntu.json and rally/rally_scenarios_fip_and_ubuntu_100.json.
SALT_MASTER_CREDENTIALS	Specifies the credentials to Salt API stored in Jenkins, included by default. See View credentials details used in Jenkins pipelines.

SALT_MASTER_URL	Specifies the reachable IP address of the Salt Master node and port on which Salt API listens. For example, http://172.18.170.28:6969 To determine on which port Salt API listens: 1. Log in to the Salt Master node. 2. Search for the port in the /etc/salt/master.d/_api.conf file. 3. Verify that the Salt Master node is listening on that port: `netstat -tunelp | grep <PORT>` **Caution!** In the 2019.2.4 update, by default, the HTTPS (SSL) NGINX URL is used as SALT_MASTER_URL. For long-running pipelines, it may lead to a timeout error, usually after 10 minutes. Therefore, specify the Salt API native non-SSL URL as described above to prevent timeout errors. Starting from the 2019.2.5 update, Salt API native non-SSL URL is used by default.
TARGET_NODE	Specifies the node to run the container with Tempest/Rally. Use the Jenkins slave as it has the Docker package. **Note** Starting from the MCP 2019.2.2 update, if the TARGET_NODE parameter is empty, the node with the gerrit:client pillar will be used, which is cid01 by default.
TEST_IMAGE	Specifies the link to the Docker Rally-based image to use for running the container with testing tools. We recommend using the upstream Rally image, xrally/xrally-openstack:0.11.2. For the full offline mode, use the cvp-rally image from the local Docker images mirror (Registry) or pull docker-prod-local.docker.mirantis.net/mirantis/cvp/cvp-rally:<mcp_version> locally.

TOOLS_REPO	Includes the URL or path to the repository where testing tools, scenarios, and configurations are located. By default, it is https://github.com/Mirantis/cvp-configuration. Specify your MCP version here. For example, -b release/2019.2.0 for Q4`18.  **Caution!** For the Q4`18 MCP release, the branch name format is release/2019.2.0 for the cvp-configuraiton repository. The old 2019.2.0 format is deprecated. To customize the configuration, clone the cvp-configuration repository to your local Gerrit and commit the changes. For the offline mode, specify /var/lib/cvp-configuration/configure.sh. Alternatively, use this repository from the offline image. If your image is fully configured, leave the field empty.

7. Click Build.
8. Verify the job status:
   - GREEN, SUCCESS

     Testing has beed performed successfully, no errors found.
   - YELLOW, UNSTABLE

     Some errors occurred during the test run. Proceed to Review the CVP - Performance pipeline tests results.
   - RED, FAILURE

     Testing has failed due to issues with the framework or/and pipeline configuration. Review the console output.

©2025, Mirantis Inc.

Review the CVP - Performance pipeline tests results

This section explains how to review the CVP Performance tests trace logs from the Jenkins web UI.

To review the CVP - Performance pipeline tests results:

1. Log in to the Jenkins web UI.
2. Navigate to the build that you want to review.
3. Find Test Results at the bottom of the Build page.
4. Scroll down and click Show all failed tests to view the complete list of the failed tests.
5. Click on the test of concern for details.

    Note

    If + does not expand, use a different browser or a public URL for your Jenkins.

Perform the OpenStack high availability testing

The OpenStack high availability testing is aimed to perform the non-functional failover testing of the Openstack Virtualized Control Plane (VCP) nodes such as the OpenStack controller, database, networking nodes. You can also check other nodes, for example, StackLight.

You can perform the high availability testing of your MCP OpenStack deployment using the CVP - HA tests Jenkins pipeline.

> Note
>
> The pipeline is provided as is. It contains default settings, examples, and templates that may need adjustment for a deployed environment. Mirantis does not support the third-party components like Tempest or Rally used in the CVP tooling.

Execute the CVP - HA tests pipeline

This section instructs you on how to perform non-functional failover testing of OpenStack nodes that include but are not limited to the ctl, ntw, and dbs virtual machines using the CVP - HA tests Jenkins pipeline.

> Note
>
> Clone the cvp-configuration and tempest repositories to your local Gerrit and use them locally to add new or adjust the existing tests and fine-tune the Tempest configuration.

To perform the non-functional failover testing of your OpenStack deployment:

1. In your local cvp-configuration repository, inspect and modify the following items as required:

    - The Tempest configuration
    - The skip list
    - The configure.sh setup script

    > Note
    >
    > We recommend running CVP - HA tests Jenkins pipeline after the functional/integration testing is completed and all issues are resolved.

2. In a web browser, open http://<ip_address>:8081 to access the Jenkins web UI.

    > Note
    >
    > The IP address is defined in the classes/cluster/<cluster_name>/cicd/init.yml file of the Reclass model under the cicd_control_address parameter variable.

3. Log in to the Jenkins web UI as admin.

    > Note
    >
    > To obtain the password for the admin user, run the salt "cid*" pillar.data _param:jenkins_admin_password command from the Salt Master node.

4. In the global view, find the CVP - HA tests pipeline.
5. Select the Build with Parameters option from the drop-down menu of the pipeline.
6. Configure the following parameters as required:

CVP - HA tests parameters

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>DEBUG_MODE</td><td>If checked, keeps the container after test is performed for the debugging purposes.</td></tr>
<tr><td>MANUAL_CONFIRMATION</td><td>If checked, you will be asked for a confirmation before any destructive actions such as a node reboot or shutdown.</td></tr>
<tr><td>PROXY</td><td>If an environment uses HTTP or HTTPS proxy, verify that you specify it in this field as this proxy address will be used to clone the required repositories and install the Python requirements. For the offline mode, specify offline.</td></tr>
<tr><td>RETRY_CHECK_STATUS</td><td>Specifies the number of retries to check the node status. If you have any issues with timeouts, increase the default 200 value.</td></tr>
<tr><td>SALT_MASTER_CREDENTIALS</td><td>Specifies the credentials to Salt API stored in Jenkins, included by default. See View credentials details used in Jenkins pipelines.</td></tr>
<tr><td>SALT_MASTER_URL</td><td>Specifies the reachable IP address of the Salt Master node and port on which Salt API listens. For example, http://172.18.170.28:6969
To determine on which port Salt API listens: Log in to the Salt Master node. Search for the port in the /etc/salt/master.d/_api.conf file. Verify that the Salt Master node is listening on that port:<pre><code>netstat -tunelp | grep <PORT></code></pre> Caution!
In the 2019.2.4 update, by default, the HTTPS (SSL) NGINX URL is used as SALT_MASTER_URL. For long-running pipelines, it may lead to a timeout error, usually after 10 minutes. Therefore, specify the Salt API native non-SSL URL as described above to prevent timeout errors. Starting from the 2019.2.5 update, Salt API native non-SSL URL is used by default.</td></tr>
</table>

SKIP_LIST_PATH	Optional. Specifies the path to the tempest skip list file located in the repository specified in TOOLS_REPO. The default value is cvp-configuration/tempest/skip-list.yaml. For the offline mode, when cvp-rally image is used, define /var/lib/cvp-configuration/tempest/skip-list.yaml. If you use a custom image, specify the path to the custom skip list file.
TARGET_NODES	Specifies OpenStack control plane nodes that will be under the HA test. For example, ctl* will include all nodes that have ctl at the beginning of their names (usually controllers).
TEMPEST_REPO	Specifies the Tempest repository to clone and use. By default, it is the upstream Tempest. Though, you can specify your customized tempest in a local or remote repository. For the full offline mode, specify /var/lib/tempest (a path inside a container) and cvp-rally image.
TEMPEST_TARGET_NODE	Specifies the node to run the container with Tempest/Rally. Use the Jenkins slave as it has the Docker package. Note Starting from the MCP 2019.2.2 update, if the TARGET_NODE parameter is empty, the node with the gerrit:client pillar will be used, which is cid01 by default.
TEMPEST_TEST_PATTERN	Specifies the tests to run. See the Rally documentation for all available options.
TEST_IMAGE	Specifies the link to the Docker Rally-based image to use for running the container with testing tools. We recommend using the upstream Rally image, xrally/xrally-openstack:0.11.2. For the full offline mode, use the cvp-rally image from the local Docker images mirror (Registry) or pull docker-prod-local.docker.mirantis.net/mirantis/cvp/cvp-rally:<mcp_version> locally.

<table>
<tr><td>TOOLS_REPO</td><td>Includes the URL or path to the repository where testing tools, scenarios, and configurations are located. By default, it is https://github.com/Mirantis/cvp-configuration. Specify your MCP version here. For example, -b release/2019.2.0 for Q4`18.<br><br><b>Caution!</b><br>For the Q4`18 MCP release, the branch name format is release/2019.2.0 for the cvp-configuraiton repository. The old 2019.2.0 format is deprecated.

To customize the configuration, clone the cvp-configuration repository to your local Gerrit and commit the changes.
For the offline mode, specify /var/lib/cvp-configuration/configure.sh. Alternatively, use this repository from the offline image.
If your image is fully configured, leave the field empty.</td></tr>
</table>

7. Click Build.
8. Verify the job stages statuses in the Stage view section:

   - GREEN, SUCCESS

     Testing has beed performed successfully, no errors found.
   - RED, FAILURE

     Tempest run has failed during one of the job stages causing the pipeline abort. Review the console output.

Add new tests to the CVP - HA tests pipeline

This section includes the instruction on how to extend the predefined OpenStack high availability tests.

To add custom OpenStack high availability tests:

1. Clone the openstack/tempest Git repository to your local Gerrit repository.
2. Check out the tempest repository from your local Gerrit.
3. If required, create a new directory.
4. Create a file for the new test or update the existing test file.
5. Add the test code. Refer to Tempest Test Writing Guide in the OpenStack official documentation.
6. Commit the changes to the local Gerrit.
7. To perform the newly added test, run the CVP - Functional tests pipeline as described in Execute the CVP - HA tests pipeline, specifying the TEMPEST_TEST_PATTERN parameter according to your test/test class/suite name.

   > Note
   >
   > Verify, that the TEMPEST_REPO value matches your local Gerrit copy of tempest.

©2025, Mirantis Inc.

Verify StackLight LMA

StackLight LMA testing through the Jenkins web UI is aimed to perform the basic verification of your StackLight LMA deployment helping to troubleshoot the StackLight LMA cluster health.

You can perform the StackLight LMA verification using the CVP - StackLight tests pipeline.

> Note
>
> The pipeline is provided as is. It contains default set of tests that may need adjustment for a deployed environment.

Execute the CVP - StackLight tests pipeline

This section instructs you on how to perform the StackLight LMA verification using the CVP - StackLight tests Jenkins pipeline.

> Note
>
> For MCP versions prior the 2019.2.4 update, clone the release/2019.2.0 branch of the stacklight-pytest repository to your local Gerrit and use it locally to add new or adjust the existing tests.

To perform the StackLight LMA verification:

1. In a web browser, open http://<ip_address>:8081 to access the Jenkins web UI.

   > Note
   >
   > The IP address is defined in the classes/cluster/<cluster_name>/cicd/init.yml file of the Reclass model under the cicd_control_address parameter variable.

2. Log in to the Jenkins web UI as admin.

   > Note
   >
   > To obtain the password for the admin user, run the salt "cid*" pillar.data _param:jenkins_admin_password command from the Salt Master node.

3. In the global view, find the CVP - StackLight tests pipeline.
4. Select the Build with Parameters option from the drop-down menu of the pipeline.
5. Configure the following parameters as required:

CVP - StackLight tests parameters

Parameter	Description
PROXY Removed since 2019.2.4 update	If an environment uses HTTP or HTTPS proxy, verify that you specify it in this field as this proxy address will be used to clone the required repositories and install the Python requirements.

SALT_MASTER_CREDENTIALS	Specifies the credentials to Salt API stored in Jenkins, included by default. See View credentials details used in Jenkins pipelines.
SALT_MASTER_URL	Specifies the reachable IP address of the Salt Master node and port on which Salt API listens. For example, http://172.18.170.28:6969 To determine on which port Salt API listens: 1. Log in to the Salt Master node. 2. Search for the port in the /etc/salt/master.d/_api.conf file. 3. Verify that the Salt Master node is listening on that port: `netstat -tunelp | grep <PORT>`
TESTS_REPO Removed since 2019.2.4 update	Specifies the repository with the Stacklight tests that can be either a Github URL or an internal Gerrit repository with custom tests. By default, it is http://gerrit.mcp.mirantis.com/mcp/stacklight-pytest -b release/2019.2.0 that clones the release/2019.2.0 branch of the stacklight-pytest repository.
TESTS_SET Removed since 2019.2.4 update	Specifies the name of the test to perform or directory to discover tests. By default, it is stacklight-pytest/stacklight_tests/tests/. Leave the field as is for a full test run or specify the test filename. For example, the <default_path>/test_logs.py will only run the test for Kibana.
TESTS_SETTINGS Removed since 2019.2.4 update	Includes additional environment variables that can be passed to the test framework to override the default configuration. Always specify the SL_AUTOCONF=True and PYTHONPATH="./stacklight-pytest" options. Use semicolon to separate variables. The export skipped_nodes=mtr01.local,log02.local string will force the pipeline to skip the mtr01 and log02 nodes. If you have the UPG nodes in your deployment, add them to the skip_nodes list. For example, skipped_nodes=upg01.<domain_name>.
EXTRA_PARAMS Added since 2019.2.4	Includes additional environment variables that can be passed to the test framework to override the default configuration. Always specify - SL_AUTOCONF=True. Added since 2019.2.5 Use the force_pull parameter to enable or disable the pull operation for an image before running the container. Set force_pull=false if image pulling is impossible or not required. In this case, the image may be manually uploaded to the target node.

6. Click Build.

7. Verify the job status:

   - GREEN, SUCCESS

     Testing has beed performed successfully, no errors found.
   - YELLOW, UNSTABLE

     Some errors occurred during the test run. Proceed to Review the CVP - StackLight pipeline tests results.
   - RED, FAILURE

     Testing has failed due to issues with the framework or/and pipeline configuration. Review the console output.

Review the CVP - StackLight pipeline tests results

This section explains how to review the CVP - StackLight tests trace logs from the Jenkins web UI.

To review the CVP - StackLight tests results:

1. Log in to the Jenkins web UI.
2. Navigate to the build that you want to review.
3. Find Test Results at the bottom of the Build page.
4. Scroll down and click Show all failed tests to view the complete list of the failed tests.
5. Click on the test of concern for details.

   > Note
   >
   > A test name corresponds to the test file path in the test source repository. For example, stacklight_tests.tests.prometheus.test_dashboards.

6. Review the lines beginning with the E letter in Stack trace.

   > Note
   >
   > If + does not expand, use a different browser or a public URL for your Jenkins.

Add new tests to the CVP - StackLight tests pipeline

This section includes the instruction on how to extend the predefined StackLight LMA tests.

> Note
>
> Starting from the MCP 2019.2.4 maintenance update, the CVP - Stacklight test does not depend on the external repository, adding custom StackLight LMA tests is deprecated.

To add custom StackLight LMA tests prior to the MCP 2019.2.4 update:

1. Clone the Mirantis/stacklight-pytest Git repository to a local Gerrit repository.
2. Check out the stacklight-pytest repository from you local Gerrit.
3. If required, create a new directory under stacklight-pytest/stacklight_tests/tests/.
4. Create the test_<test_file_name>.py file where the test_ prefix is mandatory.
5. Add the test code.
6. Commit your changes to the local Gerrit.
7. To perform the newly added test, run the CVP - StackLight tests pipeline as described in Execute the CVP - StackLight tests pipeline, specifying <default_path>/<new_folder_name>/test_<test_file_name> in the TESTS_SET field.

   > Note
   >
   > Verify that the TESTS_SET value matches your local Gerrit copy of the stacklight-pytest repository.

Perform the data plane networking testing with CVP Shaker

CVP Shaker verifies and measures the performance of the data plane networking of your MCP OpenStack deployment. This test suite is based on Shaker that is a wrapper around popular system network testing tools such as iperf, iperf3, and netperf.

To perform the OpenStack data plane networking test of your environment, use the CVP - Shaker network tests Jenkins pipeline.

> Warning
>
> This feature is available starting from the MCP 2019.2.3 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

Execute the CVP - Shaker network tests pipeline

This section instructs you on how to perform the data plane networking test of your deployment using the CVP - Shaker network tests Jenkins pipeline.

> Note
>
> Clone the cvp-shaker repository to your local Gerrit and use it locally to add new or adjust the existing scenarios and build a new Docker image with Shaker.

To perform the data plane networking test of your deployment:

1. In a web browser, open http://<ip_address>:8081 to access the Jenkins web UI.

   > Note
   >
   > The IP address is defined in the classes/cluster/<cluster_name>/cicd/init.yml file of the Reclass model under the cicd_control_address parameter variable.

2. Log in to the Jenkins web UI as Administrator.

   > Note
   >
   > To get the password, execute the following command on the Salt Master node:
   >
   > ```
   > salt-call pillar.data _param:jenkins_admin_password
   > ```

3. In the global view, find the CVP - Shaker network tests pipeline.
4. Select the Build with Parameters option from the drop-down menu of the pipeline.
5. Configure the following parameters as required:

CVP - Sanity checks parameters

Parameter	Description
IMAGE	The cvp-shaker Docker image (with all dependencies) that will be used during the test run. The default value is docker-prod-local.docker.mirantis.net/mirantis/cvp/cvp-shaker:<MCP_VERSION>. For the offline mode, use the URL from the local artifactory or offline image.

SALT_MASTER_CREDENTIALS	The credentials to Salt API stored in Jenkins, included by default. See View credentials details used in Jenkins pipelines for details.
SALT_MASTER_URL	The reachable IP address of the Salt Master node and port on which Salt API listens. For example, http://172.18.170.28:6969. To determine on which port Salt API listens: 1. Log in to the Salt Master node. 2. Search for the port in the /etc/salt/master.d/_api.conf file. 3. Verify that the Salt Master node is listening on that port: `netstat -tunelp | grep <PORT>`
SHAKER_PARAMS	The YAML context with parameters for running Shaker. See the description of the available options below. The SHAKER_SERVER_ENDPOINT option is mandatory, while others can be left with the default values.
SHAKER_PARAMS:SHAKER_SERVER_ENDPOINT	The address for the Shaker server connections in the form of host:port. The address should be accessible from the OpenStack public (floating) network and usually equals to a public-routable address of the CI/CD node. This is a mandatory option. **Caution!** The Shaker server address should belong to the CI/CD node that you start the job at and is accessible from the Openstack public (floating) network. Also, to be able to run some scenarios, you should provide compute nodes with the 4*(density count) gigabytes of free disk space.
SHAKER_PARAMS:SHAKER_SCENARIOS	Path to the Shaker scenarios in the cvp-shaker Docker image. Can be either a directory or a specific file. The main categories include: • scenarios/essential/l2 • scenarios/essential/l3 • scenarios/additional/cross_az • scenarios/additional/external • scenarios/additional/qos The default value is scenarios/essential that starts a comprehensive test of both L2 and L3 data plane networking.

SHAKER_PARAMS:SKIP_LIST	A comma-separated list of Shaker scenarios to skip directories or files inside the scenarios directory of the cvp-shaker Docker image. For example, dense_l2.yaml,full_l2.yaml,l3. Defaults to an empty string.
SHAKER_PARAMS:MATRIX	The matrix of extra parameters for the scenario. The value is specified in the JSON format. Defaults to an empty string. For example, to override a scenario duration, specify {time: 10}, or to override the list of hosts, define {host:[ping.online.net, iperf.eenet.ee]}. If several parameters are overridden, all combinations are tested. It is a required field for some of external-category scenarios when the host name with external iPerf3 server must be provided as a command-line parameter, for example, {host: 10.13.100.4}.
SHAKER_PARAMS:IMAGE_BUILDER	The list of the shaker-image-builder environment variables that includes: • SHAKER_FLAVOR_DISK • SHAKER_FLAVOR_RAM • SHAKER_FLAVOR_VCPUS • SHAKER_IMAGE_BUILDER_MODE Used for building an image of Shaker which will be used for running Shaker agents across the cluster. Leave these parameters commented out as the default settings should meet all the requirements for starting a test.
SHAKER_PARAMS:SHAKER	List of Shaker server environment variables including: • SHAKER_AGENT_JOIN_TIMEOUT • SHAKER_AGENT_LOSS_TIMEOUT • SCENARIO_AVAILABILITY_ZONE • SCENARIO_COMPUTE_NODES • SHAKER_EXTERNAL_NET You can define these parameters to alter the Shaker server environment variables. The SHAKER_EXTERNAL_NET variable should be set to the name of your OpenStack floating network, which defaults to public. The SCENARIO_AVAILABILITY_ZONE variable can be used when running the cross az scenarios category to override default availability zones set in the scenarios. All the other options in most cases can be left unchanged.

6. Click Build.
7. Verify the job status:

- GREEN, SUCCESS

  Testing has beed performed successfully, no errors found.
- RED, FAILURE

  Testing has failed due to issues with the framework or/and pipeline configuration. Review the console output.

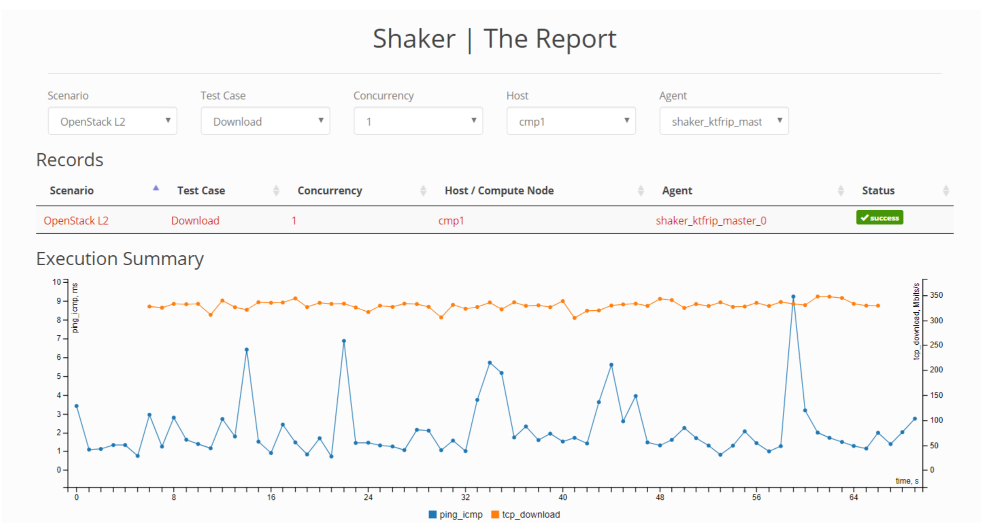
Review the CVP Shaker network test results

This section explains how to review the CVP - Shaker network tests trace logs and results from the Jenkins web UI.

To review the CVP Shaker test results:

1. Log in to the Jenkins web UI.
2. Navigate to the build that you want to review.
3. Find the shaker-report.html at the top of the Build page.
4. Download the report and open it.
5. Click on the scenario of concern for details. A scenario name corresponds to the scenario file path in the cvp-shaker source repository. For example, the OpenStack L3 East-West scenario name corresponds to cvp-shaker/scenarios/essential/l3/full_l3_east_west.yaml
6. Review the performance graphs or errors that appeared during the testing.

   For example:

7. (Optional) To view the log messages produced during the testing by Shaker, inspect shaker.log at the bottom of the Build page.

Troubleshooting

This section provides information on troubleshooting and known issues.

Generate a sosreport

Sosreport is an extensible and portable support data collection tool that creates diagnostic snapshots of the system, including the system log files and configuration details. Starting from MCP 2019.2.7 maintenance update, the tool works out of the box, does not require any mandatory configuration, and enables you to archive the obtained data and attach the archive to a Salesforce case.

Generate a sosreport prior to 2019.2.7

This section describes how to generate a sosreport for MCP versions prior to the MCP 2019.2.7 maintenance update. In this case, the sosreport tool works without Salt orchestration.

To generate a sosreport:

1. Log in to the Salt Master node.
2. Download the latest available sosreport package for the required nodes. For example:

   ```
   salt -C '<target>' cmd.run 'curl -o sosreport.deb http://mirror.mirantis.com/update/2019.2.0/extra/xenial/pool/main/s/sosreport/sosreport_<version>.deb  && dpkg -i sosreport.deb'
   ```

3. Run the sosreport command on the target nodes to generate a report for the entire system. For a list of possible options, run sosreport --help.

Generate a sosreport starting from 2019.2.7

> Note
>
> This feature is available starting from the MCP 2019.2.7 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

This section describes how to generate a sosreport starting from the MCP 2019.2.7 maintenance update. The tool does not require mandatory configuration. However, you can extend the sosreport tool as required.

To generate a sosreport:

1. Optional. Recommended. Configure the sosreport tool using one of the following options:

   - Configure the sosreport tool through the Reclass model:

     1. To save the configuration for all nodes, specify the following pillar in the cluster/<cluster_name>/infra/init.yml file. Otherwise, add the pillar to a particular node or the common component file.

        ```
        linux:
         system:
          sosreport:
           cmd_options:
            tmp-dir: /root/reportdir
            no_arg_opts: [ '-q' ]
           config_options:
            general:
             all-logs: true
            plugins:
             disabled: [ docker ]
            tunables:
             apache.log: true
        ```

        Parameters description:

        - cmd_options - defines additional arguments for a CLI and cmd call.
        - general - includes the parameters for the sos.conf general section.
        - plugins - defines the enabled or disabled plugins.
        - tunables - defines custom plugin options.
     2. Apply the changes from the Salt Master node:

        ```
        salt -C <target> saltutil.sync_all
        ```

©2025, Mirantis Inc.

   - Configure the sosreport tool by overriding the default settings from the Salt Master node. For example:

     ```
     salt -C '<target>' state.sls linux.system.sosreport.report pillar='{ \
     "sosreport" : { "ticket-number": 12345, "tmp-dir": "/root/reportdir2" } }'
     ```

     For a list of possible options, run sosreport --help.

2. Select from the following options:

   - To generate a sosreport on one or multiple target nodes:

     1. Log in to the Salt Master node.
     2. Run the following command:

        ```
        salt -C '<target>' state.sls linux.system.sosreport.report
        ```

   - To generate a sosreport on a particular node:

     1. Log in to the required node.
     2. Run the following command:

        ```
        salt-call state.sls linux.system.sosreport.report
        ```

Now, proceed to Collect and archive the reports.

Collect and archive the reports

> Note
>
> This feature is available starting from the MCP 2019.2.7 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

Once you obtain the data from the required nodes as described in Generate a sosreport starting from 2019.2.7, create a common archive with the obtained data and attach it to a Salesforce case.

To collect and archive the reports:

1. Log in to the Salt Master node.
2. Run the following command specifying the options as required:

   ```
   salt -C 'I@salt:master' state.sls linux.system.sosreport.collect pillar='{ \
   "sosreport_collect" : { "target": "<target>", "archiveName": "sosreport_<env_name>_<customer_name>_<SF_ticket_ID>" } }'
   ```

   Additionally, you can specify the following options:

   - nodeIp - to use an IP from another interface on the node. The IP must be available from the Salt minions.
   - port - to use NetCat in case the default 31337 port is busy.
   - reportWorkDir- to specify the directory to keep all reports for a specific case.

   As a result, the tool creates one common archive named sosreport_<env_name>_<customer>_<ticket>.tar.gz for all <target> nodes based on the parameters set through the model or pillar override and attaches it to the specified Salesforce ticket.

Troubleshooting

When collecting logs using the sosreport tool, the No space left on device while collecting plugin data exception may occur. Such exception occurs if the logs are large and indicates that the device with the temporary directory has no free space available. The temporary directory stores system reports before archiving and is set to /tmp by default.

To avoid the exception, manually change the directory through the model or CLI as described in the step 1 in Generate a sosreport starting from 2019.2.7. Additionally, you can enable a verbose output and interactive debugging using the Python debugger. For details, run sosreport --help.

Troubleshoot DriveTrain

This section instructs you on how to troubleshoot the DriveTrain issues.

- Glusterd service restart failure
- Failure to connect to Jenkins master
- NGINX gateway timeout during the DriveTrain upgrade
- SaltReqTimeoutError during the DriveTrain upgrade

Glusterd service restart failure

[32334] The Glusterd service does not restart automatically after its child processes failed or were unexpectedly killed.

To troubleshoot the issue:

1. Log in to a KVM node.
2. In the /lib/systemd/system/glusterd.service file, set the Restart option in the [Service] section:

   ```
   [Service]
   ...
   Restart=on-abort
   ...
   ```

   The recommended values include:

   - on-abort

     The service restarts only if the service process exits due to an uncaught signal not specified as a clean exit status.
   - on-failure

     The service restarts when the process exits with a non-zero exit code, is terminated by a signal including on core dump and excluding the aforementioned four signals, when an operation such as service reload times out, and when the configured watchdog timeout is triggered.
3. Apply the changes:

   ```
   systemctl daemon-reload
   ```

   > Note
   >
   > Re-apply the provided workaround if any of the GlusterFS packages has been re-installed or upgraded.

4. Perform the steps above on the remaining KVM nodes.

Failure to connect to Jenkins master

[34848] Jenkins slaves may be unable to connect to Jenkins master during the update of MCP versions prior to 2019.2.4 to MCP maintenance update 2019.2.8 or newer. The issue may be related to the Cross Site Request Forgery (CSRF) protection configuration.

To verify whether your deployment is affected:

1. From the Salt Master node, obtain the Jenkins credentials:

   ```
   salt -C 'I@jenkins:client and not I@salt:master' config.get jenkins
   ```

2. Run the following command:

   ```
   curl -u <jenkins_admin_user>:<jenkins_admin_password> '<jenkins_url>//crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)'
   ```

   If the system response is 404 Not found, proceed with the issue resolution below.

To apply the issue resolution:

1. Log in to the Jenkins web UI.
2. Navigate to Manage Jenkins > Configure Global Security.
3. Under CSRF Protection, select Prevent Cross Site Request Forgery exploits and Default Crumb Issuer.
4. Click Save.

Once done, Jenkins slaves automatically reconnect in a few seconds.

NGINX gateway timeout during the DriveTrain upgrade

[34798] The Deploy - upgrade MCP DriveTrain Jenkins pipeline job may fail with the Error with request: HTTP Error 504: Gateway Time-out error message. The issue may occur in huge environments when applying Salt states on all nodes due to a small NGINX timeout configured on the Salt Master node.

To apply the issue resolution:

Select one of the following options:

- In the Deploy - upgrade MCP DriveTrain Jenkins pipeline job, set the SALT_MASTER_URL parameter to the Salt API endpoint http://<cfg_node_ip>:6969.
- Increase the NGINX timeout:
  1. Open your Git project repository with the Reclass model on the cluster level.
  2. In classes/cluster/<cluster_name>/infra/config/init.yml, increase the NGINX timeout for the Salt API site. By default, the timeout is set to 600 seconds.

```yaml
parameters:
  nginx:
    server:
      site:
        nginx_proxy_salt_api:
          proxy:
            timeout: <timeout>
```

3. Refresh Salt pillars and apply the nginx state on Salt Master node:

```
salt -C 'I@salt:master' saltutil.refresh_pillar
salt -C 'I@salt:master' state.apply nginx
```

4. Commit the changes to your local repository.

SaltReqTimeoutError during the DriveTrain upgrade

[34114] The Deploy - upgrade MCP DriveTrain Jenkins pipeline job may fail with the SaltReqTimeoutError in master zmq thread error message when executing the salt.minion state on several minions at the same time. Adjust the Salt Master configuration to improve its performance.

To adjust the Salt Master configuration:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In cluster/<cluster_name>/infra/config/init.yml, increase the values for the following parameters as required:

   gather_job_timeout
   : The number of seconds to wait for the client to request information about the running jobs.

   worker_threads
   : The number of threads to start to receive commands and replies from minions.

   sock_pool_size
   : The pool size of Unix sockets. To avoid blocking waiting while writing data to a socket, a socket pool is supported for Salt applications. For example, a job or state with a large number of target host list can cause a long period of blocking waiting.

   zmq_backlog
   : The number of messages in the ZeroMQ backlog queue.

   For example:

```yaml
parameters:
  salt:
    master:
      worker_threads: 40
      opts:
```

```
gather_job_timeout: 100
sock_pool_size: 15
zmq_backlog: 3000
```

3. Log in to the Salt Master node.
4. Refresh Salt pillars and apply the salt.master state.

```
salt-call saltutil.refresh_pillar
salt-call state.apply salt.master
```

5. Commit the changes to your local repository.

Troubleshoot an MCP OpenStack environment

This section describes procedures helping to troubleshoot problems that may occur in an MCP OpenStack environment due to restarting the cloud environment after a power outage, for example.

If your MCP OpenStack environment is OpenContrail-based, refer to Troubleshoot OpenContrail to troubleshoot networking.

Troubleshoot the system

This section describes how to verify the Linux system status.

To perform the Linux status check:

1. Log in to the OpenStack controller node that you want to troubleshoot.
2. Verify that there is enough space on the disk:

   ```
   df -h
   ```

   Example system response:

   ```
   Filesystem      Size  Used Avail Use% Mounted on
   /dev/sda1        20G  4.5G   15G  25% /
   tmpfs           939M  328K  939M   1% /dev/shm
   /dev/sdd1       3.9G   11M  3.7G   1% /tmp
   /dev/sdc1       9.8G  102M  9.2G   2% /var/log
   /dev/sde        3.9G   34M  3.6G   1% /var/log/audit
   tmpfs           512M     0  512M   0% /config
   ```

3. Check the available inodes:

   ```
   df -i
   ```

   Example system response:

   ```
   Filesystem       Inodes  IUsed    IFree IUse% Mounted on
   /dev/sda1      14589952 387481 14202471    3% /
   none            2038052     13  2038039    1% /sys/fs/cgroup
   udev            2035377    518  2034859    1% /dev
   tmpfs           2038052    591  2037461    1% /run
   none            2038052      5  2038047    1% /run/lock
   none            2038052    118  2037934    1% /run/shm
   none            2038052     35  2038017    1% /run/user
   ```

4. Verify that there is enough amount of memory:

   ```
   free m
   ```

   Example system response:

   ```
                total       used       free     shared    buffers     cached
   Mem:       1922208    1839352      82856        500     415068     525944
   -/+ buffers/cache:     898340    1023868
   Swap:      4193276       1544    4191732
   ```

5. Verify if there are no processes that use all available memory/CPU:

```
htop
```

Example system response:

```
top - 11:46:28 up 40 days, 22:35,  3 users,  load average: 0.00, 0.00, 0.03
Tasks: 218 total,   1 running, 217 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.8%us,  1.5%sy,  0.0%ni, 95.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1922208k total,  1829060k used,    93148k free,   415108k buffers
Swap:  4193276k total,     1544k used,  4191732k free,   526056k cached
```

Troubleshoot the OpenStack services

Depending on your needs, inspect the OpenStack services logging information:

- On the OpenStack controller nodes:

```
tail -fn0 /var/log/{nova,cinder,glance,keystone,neutron,contrail,cassandra,rabbitmq}/*.log \
| egrep 'ERROR|WARNING|REFUSED|EXCEPTION|TRACE|error'
```

- On the OpenStack compute nodes:

```
tail -fn0 /var/log/{nova,contrail}/*.log | \
egrep 'ERROR|TRACE|WARNING|REFUSED|EXCEPTION|error|SHUTDOWN'
```

To ensure that an OpenStack service is up and running, verify the service status on every controller node. Some OpenStack services require additional verification on the OpenStack non-controller nodes.

Before you can proceed with troubleshooting, export all credentials from keystonerc file on each controller node to be able to manage your OpenStack environment:

Verifying the OpenStack services status

Service name	Verification procedure
Glance	Use the glance image-list command. The output should contain the table with the images list. The images status should be active.
Nova	Use the nova service-list. The output should contain the table with the Nova services list. The services status should be enabled, their state should be up.
Cinder	Use the cinder-manage service list. The output should contain the table with the Cinder services list. The services status should be enabled.

©2025, Mirantis Inc.

Troubleshoot RabbitMQ

To troubleshoot RabbitMQ:

1. Log in to any OpenStack messaging node.
2. Verify the RabbitMQ cluster status:

```
rabbitmqctl cluster_status
```

3. Verify the AMQP messaging RabbitMQ status:

```
curl -s -o rabbitmqadmin http://$(hostname -s):15672/cli/rabbitmqadmin;
chmod 0755 ./rabbitmqadmin
adm_creds=$(salt-call pillar.get rabbitmq:server:admin --out=yaml \
  | egrep ':\ (.+)' | tr '\n' ' ' | sed 's/name: /-u /; s/password: /-p /')
./rabbitmqadmin -H $(hostname -s) $adm_creds show overview
./rabbitmqadmin -H $(hostname -s) $adm_creds list queues
./rabbitmqadmin -H $(hostname -s) $adm_creds list queues vhost name node messages \
  message_stats.publish_details.rate | grep -v " 0"
./rabbitmqadmin -H $(hostname -s) $adm_creds -V /openstack list queues vhost name \
node messages message_stats.publish_details.rate | grep -v " 0"
```

4. List the number of messages and consumers for each queue. Select vhost, otherwise, you will work with the default queue that is /. The number of messages should be low and number of consumers above 0:

```
rabbitmqctl list_queues -p /openstack messages consumers name
```

5. Print stuck nodes if any:

```
rabbitmqctl eval 'rabbit_diagnostics:maybe_stuck().'
```

Troubleshot RabbitMQ non-functional queue bindings

Caution!

The following procedure has been tested with clustered RabbitMQ series 3.8. For the RabbitMQ series 3.6, as well as for a nonclustered RabbitMQ, you may need to adjust the procedure.

Note

This feature is available starting from the MCP 2019.2.15 maintenance update. Before using the feature, follow the steps described in Apply maintenance updates.

In case of network hiccups or instability, a RabbitMQ cluster may not pass messages between some exchanges and their respective queues even if the corresponding bindings are present and visible in the RabbitMQ Management Plugin UI. Due to concurrent handling of self-healing actions and OpenStack clients recreating queues during reconnection, some bindings that route messages from exchanges to queues may be created dysfunctional. This issue also applies to queue policies. For example, queues can be created without the needed policies, which can lead to an unexpected behavior.

This section describes how to identify and fix dysfunctional bindings. The solution includes a helper script that sends test messages to exchanges with the identified bindings. If a message cannot be delivered with a particular binding, the script prints out a message about a potentially dysfunctional binding.

Warning

Perform the following procedure only if the RabbitMQ cluster is healthy, the self-healing is over, and cluster_status indicates that the cluster is running all the nodes it was configured with. To verify the RabbitMQ cluster health, run the following command:

```
rabbitmqctl cluster_status
```

If the output indicates that the cluster is running less nodes than it was configured with and shows active partitions that will not be dismissed, or if the command errors out showing that the RabbitMQ process is not running, proceed with Restart RabbitMQ with clearing the Mnesia database instead.

To identify and fix RabbitMQ dysfunctional queues:

©2025, Mirantis Inc.

1. Log in to the msg01 node.
2. Download the script:

   ```
   wget "https://review.fuel-infra.org/gitweb?p=tools/sustaining.git;a=blob_plain;f=scripts/rabbitmq_check.py;hb=HEAD" -O ./rabbitmq_check.py
   ```

3. Obtain the RabbitMQ admin credentials:

   ```
   salt-call -l quiet pillar.items rabbitmq:server:admin
   ```

4. Obtain the IP address where the RabbitMQ Management Plugin is listening for incoming connections:

   ```
   ss -lnt '( sport = :15672 )'
   ```

5. Run the script replacing %IP% and %PASSWORD% with the previously obtained parameters:

   ```
   python3 ./rabbitmq_check.py -u admin -p %PASSWORD% -H %IP% -V "/openstack" check
   ```

   If the script returns output with Possibly unreachable binding [...] ..., discrepancies have been found. Proceed with the steps below.
6. Restart the service that runs the particular exchange-queue pair. For example, if a conductor binding is faulty, restart the nova-conductor service on all OpenStack controller nodes.
7. If the service restart did not fix the issue, drop the state of exchanges and queues:
   1. Log in to the Salt Master node.
   2. Disconnect the clients from RabbitMQ to remove the interference with inter-cluster operations:

      ```
      salt -C 'I@rabbitmq:server' cmd.run 'iptables -I INPUT -p tcp --dport 5671:5672 -j REJECT'
      ```

   3. Stop the RabbitMQ application without shutting down the entire Erlang machine:

      ```
      salt -C 'I@rabbitmq:server' cmd.run 'rabbitmqctl stop_app'
      ```

   4. Wait for 1-2 minutes and then start the RabbitMQ application:

      ```
      salt -C 'I@rabbitmq:server' cmd.run 'rabbitmqctl start_app"'
      ```

   5. Verify that the RabbitMQ cluster is healthy:

      ```
      salt -C 'I@rabbitmq:server' cmd.run 'rabbitmqctl cluster_status'
      ```

   6. Remove the iptables rules to allow clients to connect back to RabbitMQ:

```
salt -C 'I@rabbitmq:server' cmd.run 'iptables -D INPUT -p tcp --dport 5671:5672 -j REJECT'
```

If none of the approaches fixes the issue, proceed with Restart RabbitMQ with clearing the Mnesia database.

Troubleshoot MongoDB

This section instructs you on how to troubleshoot the MongoDB failures.

To troubleshoot MongoDB:

1. Log in to the MongoDB shell as Administrator into the admin collection:

   ```
   mongo -uadmin -p$(salt-call pillar.get mongodb:server:admin:password|tail -1|tr -d ' ') admin
   ```

2. Verify the replica set configuration and status:

   ```
   rs.conf()
   rs.status()
   ```

3. To force synchronization of one of the replicas, delete data in the replica directory and start MongoDB:

   > **Caution!**
   >
   > Proceed with caution to avoid any data loss.

   ```
   service mongodb stop
   mv /var/lib/mongodb /var/lib/mongodb.backup
   mkdir /var/lib/mongodb
   chown mongodb:mongodb /var/lib/mongodb
   service mongodb start
   ```

©2025, Mirantis Inc.

Troubleshoot a Galera cluster

A Galera cluster is a true synchronous multi-master replication system. Any or all nodes composing the cluster can be used as master at any time meaning that there is no failover in the traditional MySQL master-slave sense. Though, you may encounter issues with your Galera cluster after restarting the cluster during maintenance or upgrade.

This section instructs you on how to troubleshoot the Galera cluster as well as how to restart, restore, and rejoin nodes to the cluster.

Verify a Galera cluster status

You can verify the status of a MySQL Galera cluster either manually from the Salt Master node or automatically using Jenkins.

To verify a MySQL Galera cluster status, select from the following options:

- Verify the Galera cluster status automatically using the Verify and Restore Galera cluster Jenkins pipeline as desribed in Restore a Galera cluster and database automatically.
- Verify the Galera cluster status manually:

  1. Log in to the Salt Master node.
  2. Apply the following state:

     ```
     salt -C 'I@galera:master' mysql.status | \
     grep -EA1 'wsrep_(local_state_c|incoming_a|cluster_size)'
     ```

     Example of system response:

     ```
     wsrep_cluster_size:
         3

     wsrep_incoming_addresses:
         192.168.2.52:3306,192.168.2.53:3306,192.168.2.51:3306

     wsrep_local_state_comment:
         Synced
     ```

Restore a Galera cluster

This section instructs you on how to restore the Galera cluster either using the dedicated Jenkins pipeline or manually. The automatic restoration procedure included in this section presupposes that only 1 Galera node is down or the data is corrupted. If you experience a loss of multiple nodes, apply the manual procedure adjusted to the needs of your deployment.

©2025, Mirantis Inc.

Prepare for a Galera cluster restoration

The restoration of the Galera cluster is fully dependent on the correct Xtrabackup configuration. The service misconfiguration or skipping the data backup stage may lead to the restoration failure.

To prepare for the Galera cluster restoration:

1. Log in to the Salt Master node.
2. Configure Xtrabackup by setting the parameters as required in cluster/openstack/database/init.yml. For example:

   > Note
   >
   > You will need to revert the Xtrabackup configuration after the Galera cluster is restored.

   ```
   parameters:
     xtrabackup:
       client:
         enabled: true
         restore_full_latest: 1
         restore_from: remote
   ```

   The available values for the configurable parameters include:

   - restore_full_latest can have the following values:

     1
       Restore the database from the last complete backup and its increments.

     2
       Restore the database from the second latest complete backup and its increments.

   - restore_from can have the following values:

     local
       Restore from the local storage

     remote
       Use scp to get the files from the xtrabackup server.
3. Create a data backup using the Xtrabackup service as described in Create an instant backup of a MySQL database. The backup data will be used after the restart is completed.
4. Proceed with one of the following options:

   - Restore a Galera cluster and database automatically
   - Restore a Galera cluster manually

Restore a Galera cluster and database automatically

The Galera cluster ensures that the OpenStack services are operable. In case of a cluster outage, the number of manual steps to start the cluster, as well as ensuring the necessary access can significantly delay the restoration of services and is prone to operator errors. Therefore, to reduce the complexity of the procedure and support greater scalability, MCP provides the automatic way to verify and restore the Galera cluster in your deployment.

This section describes how to verify the status of a Galera cluster and restore it using the Verify and Restore Galera cluster Jenkins pipeline. Use the automatic restoration procedure only if 1 Galera node is down or the data is corrupted. Otherwise, apply the manual procedure adjusted to the needs of your deployment as described in Restore a Galera cluster manually.

> Note
>
> This feature is available starting from the MCP 2019.2.5 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

> Note
>
> The Verify and Restore Galera cluster Jenkins pipeline restores the Galera cluster with the provided configuration and does not fix the issues caused by cluster misconfiguration.

To restore the Galera cluster and database automatically:

1. Log in to the Jenkins web UI.
2. Open the Verify and Restore Galera cluster pipeline.
3. Specify the required parameters:

Parameter	Description and values
SALT_MASTER_URL	Add the IP address of your Salt Master node host and the salt-api port. For example, http://172.18.170.27:6969.
CREDENTIALS_ID	Add credentials_id as credentials for the connection.
RESTORE_TYPE	Check ONLY_RESTORE if manual backup has been performed already. The created backup will be used during the restoration. Check BACKUP_AND_RESTORE if backup has not been performed and is required to be performed during the pipeline run.
ASK_CONFIRMATION	Set to False if you do not want the pipeline to wait for a manual confirmation before running the restoration. Defaults to True.

CHECK_TIME_SYNC	Set to False if you do not want the pipeline to verify the time synchronization accross the nodes. Defaults to True.
VERIFICATION_RETRIES	Specify the number of retries of the verification process after the restoration was performed. The value should be increased for the bigger clusters as it may take more time for such clusters to come up and synchronize. Defaults to 5.

4. Click Deploy.

   The pipeline workflow:

   1. The verification stage:

      1. Obtaining and parsing the result of the mysql.status call.
      2. Formatting and printing a result report to the user.

      Example of a verification report:

      ```
      CLUSTER STATUS REPORT: 6 expected values, 0 warnings and 1 error found:

      [OK    ] Cluster status: Primary (Expected: Primary)
      [OK    ] Master node status: true (Expected: ON or true)
      [OK    ] Master node status comment: Synced (Expected: Joining or Waiting on SST
               or Joined or Synced or Donor)
      [OK    ] Master node connectivity: true (Expected: ON or true)
      [OK    ] Average size of local reveived queue: 0.166667 (Expected: below 0.5)
               (Value above 0 means that the node cannot apply write-sets as fast
               as it receives them, which can lead to replication throttling)
      [OK    ] Average size of local send queue: 0.010204 (Expected: below 0.5)
               (Value above 0 indicate replication throttling or network throughput
               issues, such as a bottleneck on the network link.)

      [  ERROR] Current cluster size: 2 (Expected: 3)

      Errors found.

      There's something wrong with the cluster, do you want to run a restore?

      Are you sure you want to run a restore? Click to confirm
      Proceed or Abort
      ```

   2. Optional. The backup stage:

      Running the Galera database backup pipeline. For the pipeline workflow, see Create an instant backup of a MySQL database automatically.

   3. The restoration stage:

      1. If Proceed is selected, the restoration stage will continue. Otherwise, it will abort.
      2. The last shutdown node will be used as a source of truth.

4. The verification stage:

   Verifying the status of the cluster.

5. After the restoration is finalized, verify that all nodes are back and the cluster is working.
6. Revert the changes made in the cluster/openstack/database/init.yml file in the step 2 during Prepare for a Galera cluster restoration.

Restore a Galera cluster manually

As the manual procedure of the Galera cluster restoration is prone to operator errors, we recommend restoring the cluster using the Jenkins pipeline as described in Restore a Galera cluster and database automatically. Though, if you cannot apply the automatic procedure to your deployment for some reason, use the manual instruction included in this section.

> **Note**
>
> To restore a Galera database using the Jenkins pipeline, see Restore a Galera cluster and database automatically.

To restore a Galera cluster manually:

1. On all Galera dbs nodes:
   1. Stop all the MySQL processes.
   2. Verify that the MySQL processes are stopped:

      ```
      ps aux | grep mysql
      ```

   3. Identify the last shutdown Galera node:
      1. In the /var/lib/mysql/grastate.dat file on every Galera node, compare the seqno value. The Galera node that contains the maximum seqno value is the last shutdown node.
      2. If the seqno value is equal on all three nodes, identify the node on which the /var/lib/mysql/gvwstate.dat file exists. The Galera node that contains this file is the last shutdown node.
   4. Remove the grastate.dat and ib_logfiles:

      ```
      rm /var/lib/mysql/grastate.dat
      rm /var/lib/mysql/ib_logfile*
      ```

2. Log in to the last shutdown Galera node.
3. In /etc/mysql/my.cnf:
   1. Comment out the wsrep_cluster_address line:

      ```
      ...
      #wsrep_cluster_address="gcomm://192.168.0.1,192.168.0.2,192.168.0.3"
      ...
      ```

   2. Add the wsrep_cluster_address parameter without any IP address specified.

```
...
wsrep_cluster_address=gcomm://
...
```

4. Start MySQL:

```
service mysql start
```

5. Validate the current status of the Galera cluster:

```
salt-call mysql.status | grep -A1 wsrep_cluster_size
```

6. Start MySQL on the second Galera node. Wait until the node re-joins the cluster.
7. When the cluster size equals to two, start MySQL on the third node.
8. Verify the MySQL status. The cluster size should be equal to three:

```
salt-call mysql.status | grep -A1 wsrep_cluster_size
```

9. Log in to the last shutdown Galera node.
10. In /etc/mysql/my.cnf:
    1. Uncomment the line with the wsrep_cluster_address parameter:

    ```
    ...
    wsrep_cluster_address="gcomm://192.168.0.1,192.168.0.2,192.168.0.3"
    ...
    ```

    2. Remove the wsrep_cluster_address parameter without any IP address specified.
11. Verify that the /etc/salt/.galera_bootstrap file exists on every dbs node. Otherwise, create one:

```
touch /etc/salt/.galera_bootstrap
rw-rr- 1 root root 0 Feb 12 08:31 /etc/salt/.galera_bootstrap
```

12. Revert the changes made in the cluster/openstack/database/init.yml file in the step 2 during Prepare for a Galera cluster restoration.

Restart a Galera cluster

This section instructs you on how to restart the whole Galera cluster. You may need to restart the whole cluster if all three Galera nodes fail to start.

Before you restart the Galera cluster, create a data backup using the Xtrabackup service as described in Create an instant backup of a MySQL database.

> **Caution!**
>
> We recommend that you always back up your MySQL database before performing any changes to the Galera cluster, even if you do not plan to restore the backup data afterwards.

To restart the Galera cluster, use Restore a Galera cluster manually.

Rejoin a MySQL node

The MySQL Galera cluster contains three nodes. If one node fails, the MySQL database does not have an outage. Solve the problem by restarting the mysql service on the failed node. If the node cannot be rejoined to the cluster after restart, remove the following files from the /var/lib/mysql/ directory and restart the mysql service again:

```
rm -rf /var/lib/mysql/grastate*
rm -rf /var/lib/mysql/ib_log*
service mysql start
```

Troubleshoot GlusterFS

This section describes how to verify the GlusterFS services status and troubleshoot the GlusterFS-related issues if any.

> **Caution!**
>
> If you need to reboot the kvm nodes that host the GlusterFS cluster, do not proceed with rebooting the next kvm node before you make sure that the replication status of the GlusterFS services and volumes is healthy after the rebooting of the first kvm node.

Verify a GlusterFS cluster status

If you update or upgrade your MCP cluster, the kvm nodes that usually host the GlusterFS cluster require a reboot for the updates to be applied. In such cases, you have to verify the GlusterFS cluster status after reboot of every node before rebooting the next kvm node.

To verify a GlusterFS cluster status:

1. Log in to the Salt Master node.
2. Verify that the GlusterFS server status is healthy:

   ```
   salt -C 'I@glusterfs:server' cmd.run "gluster peer status"
   ```

   Example of system response:

   ```
   kvm01.cookied-cicd-bm-os-contrail40-maas.local:
   Number of Peers: 2

   Hostname: 10.167.8.243
   Uuid: 5ac8ee70-40a3-44c4-8ec8-967a0584a0d4
   State: Peer in Cluster (Connected)
   ```

```
Hostname: 10.167.8.242
Uuid: 8ce5fd88-2cae-47b5-a397-447874686406
State: Peer in Cluster (Connected)

kvm02.cookied-cicd-bm-os-contrail40-maas.local:
Number of Peers: 2

Hostname: 10.167.8.243
Uuid: 5ac8ee70-40a3-44c4-8ec8-967a0584a0d4
State: Peer in Cluster (Connected)

Hostname: kvm01.cookied-cicd-bm-os-contrail40-maas.local
Uuid: 6bf7eccb-69a2-42bd-b033-50d683120130
State: Peer in Cluster (Connected)

kvm03.cookied-cicd-bm-os-contrail40-maas.local:
Number of Peers: 2

Hostname: 10.167.8.242
Uuid: 8ce5fd88-2cae-47b5-a397-447874686406
State: Peer in Cluster (Connected)

Hostname: kvm01.cookied-cicd-bm-os-contrail40-maas.local
Uuid: 6bf7eccb-69a2-42bd-b033-50d683120130
State: Peer in Cluster (Connected)
```

3. Verify that the GlusterFS client status is healthy:

```
salt -C 'I@glusterfs:client and not I@glusterfs:server' test.ping
```

Example of system response:

```
cid03.cookied-cicd-bm-os-contrail40-maas.local:
    True
cid02.cookied-cicd-bm-os-contrail40-maas.local:
    True
cid01.cookied-cicd-bm-os-contrail40-maas.local:
    True
ctl03.cookied-cicd-bm-os-contrail40-maas.local:
    True
prx02.cookied-cicd-bm-os-contrail40-maas.local:
    True
prx01.cookied-cicd-bm-os-contrail40-maas.local:
    True
ctl01.cookied-cicd-bm-os-contrail40-maas.local:
```

```
    True
ctl02.cookied-cicd-bm-os-contrail40-maas.local:
    True
```

4. If any of the above commands fail, refer to the GlusterFS official documentation to troubleshoot the required services.
5. Verify the GlusterFS volumes status as described in Verify the GlusterFS volumes status.

Verify the GlusterFS volumes status

This section describes how to verify the status of the GlusterFS volumes and troubleshoot issues if any.

To verify the GlusterFS volumes status:

1. Log in to the Salt Master node.
2. Verify the GlusterFS volumes status:

   ```
   salt -C 'I@glusterfs:server' cmd.run "gluster volume status all"
   ```

3. If the system output contains issues, such as in the example below, or/and the volume status cannot be retrieved, refer to the GlusterFS official documentation to resolve the issues.

   Example of system response:

   ```
   kvm01.cookied-cicd-bm-os-contrail40-maas.local:
   Another transaction is in progress for aptly. Please try again after sometime.
   Another transaction is in progress for gerrit. Please try again after sometime.
   Another transaction is in progress for keystone-credential-keys. Please try again after sometime.
   Another transaction is in progress for mysql. Please try again after sometime.
   Another transaction is in progress for registry. Please try again after sometime.
   Locking failed on 10.167.8.243. Please check log file for details.
   ```

4. Inspect the GlusterFS server logs for volumes at /var/log/glusterfs/bricks/srv-glusterfs-<volume name>.log on the kvm nodes.
5. In case of any issues with the replication status of GlusterFS, stop all the GlusterFS volume-related services to prevent data corruption and immediately proceed with a troubleshooting to restore the volume in question.
6. If you need to reboot a kvm that hosts GlusterFS, verify that all GlusterFS clients have volumes mounted after a node reboot:

   1. Log in to the Salt Master node.
   2. Identify the GlusterFS VIP address:

```
salt-call pillar.get
```

Example of system response:

```
_param:infra_kvm_address
local:
    10.167.8.240
```

3. Verify that all GlusterFS clients have volumes mounted. For example:

```
salt -C 'I@glusterfs:client and not I@glusterfs:server' cmd.run "mount | grep 10.167.8.240"
```

Example of system response:

```
prx02.cookied-cicd-bm-os-contrail40-maas.local:
    10.167.8.240:/salt_pki on /srv/salt/pki type fuse.glusterfs
(rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
ctl01.cookied-cicd-bm-os-contrail40-maas.local:
    10.167.8.240:/keystone-credential-keys on /var/lib/keystone/credential-keys type fuse.glusterfs
(rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
    10.167.8.240:/keystone-keys on /var/lib/keystone/fernet-keys type fuse.glusterfs
(rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
cid03.cookied-cicd-bm-os-contrail40-maas.local:
prx01.cookied-cicd-bm-os-contrail40-maas.local:
    10.167.8.240:/salt_pki on /srv/salt/pki type fuse.glusterfs
(rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
cid01.cookied-cicd-bm-os-contrail40-maas.local:
cid02.cookied-cicd-bm-os-contrail40-maas.local:
ctl03.cookied-cicd-bm-os-contrail40-maas.local:
cfg01.cookied-cicd-bm-os-contrail40-maas.local:
    10.167.8.240:/salt_pki on /srv/salt/pki type fuse.glusterfs
    (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072)
ctl02.cookied-cicd-bm-os-contrail40-maas.local:
```

In the example above, several VMs do not have the GlusterFS volumes mounted, for example, cid03, cid01, cid02. In such case, reboot the corresponding VMs and verify the status again.

4. Verify that all mounted volumes identified in the previous step match the pillar information on the corresponding VM. For example:

```
salt prx02.cookied-cicd-bm-os-contrail40-maas.local pillar.get glusterfs:client:volumes
```

Example of system response:

```
----------
    salt_pki:
    ----------
    opts:
        defaults,backup-volfile-servers=10.167.8.241:10.167.8.242:10.167.8.243
    path:
        /srv/salt/pki
    server:
        10.167.8.240
```

In the output above, the server IP address and the only mounted salt_pki volume information match the output for the prx02 VM shown in the previous step.

Caution!

Do not proceed to reboot the next kvm node that hosts the GlusterFS cluster before all volumes are mounted on VMs of the first kvm node.

Troubleshoot an instance failure

Depending on your needs, debug an instance failure examining the following log files:

- /var/log/nova/nova-scheduler.log
- /var/log/nova/nova-compute.log

To delete the instance in the error state:

```
nova reset-state
nova reset-state --active
nova delete
```

Increase the size limit for uploading a Glance image

The maximum size of a Glance image is limited to 30 GB on the system level of the Reclass model in /nginx/server/proxy/openstack/glance.yml. Due to this limitation, you may receive the 500 Internal Server Error from NGINX once the upload size of a Glance image reaches 30 GB. Also, in such cases, the prx node may fail if the disk size is less than 30 GB.

To increase the upload size limit of a Glance image:

1. Open your Git project repository with the Reclass model on the cluster level.
2. In openstack/proxy.yml, add the following parameters under nginx:server:site:

   ```
   ...
   nginx_proxy_openstack_api_glance:
     proxy:
       request_buffer: false
       size: 100000m
   ```

3. Log in to the Salt Master node.
4. Apply the following state:

   ```
   salt -C 'I@nginx:server' state.sls nginx.server
   ```

Clearing ephemeral LVM volumes using shred consumes huge hardware resources

> Note
>
> This feature is available starting from the MCP 2019.2.4 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

To prevent excessive disk consumption while clearing ephemeral LVM volumes using shred, you can set the ionice level for the ephemeral LVM volume shred operation in nova-compute.

Setting of the ionice level described below makes sense if:

- nova:compute:lvm:ephemeral is set to True
- nova:compute:lvm:volume_clear is set to zero or shred

To set the ionice level:

1. Log in to the Salt Master node.
2. In classes/cluster/<cluster_name>/openstack/compute.yml, set the level for volume_clear_ionice_level as required:

   ```
   nova:
     compute:
       lvm:
         volume_clear_ionice_level: <level>
   ```

   Possible <level> values are as follows:

   - idle - to use the idle scheduling class. This option impacts system performance the least with a downside of increased time for a volume clearance.
   - From 0 to 7 - to use the best-effort scheduling class. Set the priority level to the specified number.
   - No value - not to set the I/O scheduling class explicitly. Mirantis does not recommend using no value since this is the most aggressive option in terms of system performance impact.
3. Apply the changes:

   ```
   salt -C 'I@nova:compute' state.sls nova.compute
   ```

©2025, Mirantis Inc.

Insufficient OVS timeouts causing instance traffic losses

If you receive the OVS timeout errors in the neutron-openvswitch-agent logs, such as ofctl request <...> timed out: Timeout: 10 seconds or Commands [<ovsdbap...>] exceeded timeout 10 seconds, you can configure the OVS timeout parameters as required depending on the number of the OVS ports on the gtw in your cloud. For example, if you have more than 1000 ports per a gtw node, Mirantis recommends changing the OVS timeouts as described below. The same procedure can be applied to the compute nodes if required.

> Warning
>
> This feature is available starting from the MCP 2019.2.3 maintenance update. Before enabling the feature, follow the steps described in Apply maintenance updates.

To increase OVS timeouts on the gateway nodes:

1. Log in to the Salt Master node.
2. Open /srv/salt/reclass/classes/cluster/<cluster_name>/openstack/gateway.yml for editing.
3. Add the following snippet to the parameters section of the file with the required values.

   ```
   neutron:
    gateway:
      of_connect_timeout: 60
      of_request_timeout: 30
      ovs_vsctl_timeout: 30  # Pike
      ovsdb_timeout: 30  # Queens and beyond
   ```

4. Apply the following state:

   ```
   salt -C 'I@neutron:gateway' state.sls neutron
   ```

5. Verify whether the Open vSwitch logs contain the Datapath Invalid and no response to inactivity probe errors:

   - In the neutron-openvswitch-agent logs, for example:

     ```
     ERROR ... ofctl request <...> error Datapath Invalid 64183592930369: \
     InvalidDatapath: Datapath Invalid 64183592930369
     ```

   - In openvswitch/ovs-vswitchd.log, for example:

     ```
     ERR|br-tun<->tcp:127.0.0.1:6633: no response to inactivity probe \
     after 5 seconds, disconnecting
     ```

If the logs contain such errors, increase inactivity probes for the OVS bridge controllers and OVS manager:

1. Log in to any gtw node.
2. Run the following commands:

```
ovs-vsctl set controller br-int inactivity_probe=60000
ovs-vsctl set controller br-tun inactivity_probe=60000
ovs-vsctl set controller br-floating inactivity_probe=60000
```

3. Identify the OVS manager ID:

```
ovs-vsctl list manager
```

4. Run the following command:

```
ovs-vsctl set manager <ovs_manager_id> inactivity_probe=30000
```

To increase OVS timeouts on the compute nodes:

1. Log in to the Salt Master node.
2. Open /srv/salt/reclass/classes/cluster/<cluster_name>/openstack/compute.yml for editing.
3. Add the following snippet to the parameters section of the file with the required values.

```
neutron:
 compute:
   of_connect_timeout: 60
   of_request_timeout: 30
   ovs_vsctl_timeout: 30  # Pike
   ovsdb_timeout: 30  # Queens and beyond
```

4. Apply the following state:

```
salt -C 'I@neutron:compute' state.sls neutron
```

5. Verify whether the Open vSwitch logs contain the Datapath Invalid and no response to inactivity probe errors:

   - In the neutron-openvswitch-agent logs, for example:

```
ERROR ... ofctl request <...> error Datapath Invalid 64183592930369: \
InvalidDatapath: Datapath Invalid 64183592930369
```

- In openvswitch/ovs-vswitchd.log, for example:

```
ERR|br-tun<->tcp:127.0.0.1:6633: no response to inactivity probe \
after 5 seconds, disconnecting
```

If the logs contain such errors, increase inactivity probes for the OVS bridge controllers and OVS manager:

1. Log in to the target cmp node.
2. Run the following commands:

```
ovs-vsctl set controller br-int inactivity_probe=60000
ovs-vsctl set controller br-tun inactivity_probe=60000
ovs-vsctl set controller br-floating inactivity_probe=60000
```

3. Identify the OVS manager ID:

```
ovs-vsctl list manager
```

4. Run the following command:

```
ovs-vsctl set manager <ovs_manager_id> inactivity_probe=30000
```

Avoiding ARP flux on the multi-interface Linux hosts

A Linux host with two or more Ethernet interfaces on the same subnet may cause the Address Resolution Protocol (ARP) flux issue when a machine responds to the ARP requests from both Ethernet interfaces of the same subnet.

To avoid the ARP flux issue:

1. Log in to the affected node.
2. Set the following network parameters:

   ```
   sysctl -w net.ipv4.conf.all.arp_ignore=2
   sysctl -w net.ipv6.conf.all.arp_ignore=2
   ```

   This configuration enables responding to the ARP requests only if the target IP is a local address that is configured on the incoming interface and, together with the sender IP address, is part of the same subnet on this interface.

Running the controller node after outage causes errors

If the OpenStack controller node was affected by a host or network outage and was unable to communicate with other nodes of the cluster, you may occasionally receive 401 Unauthorized errors from Keystone. To resolve the issue, synchronize the Keystone fernet tokens and credentials with other OpenStack controller nodes of the cluster.

To synchronize the Keystone fernet tokens and credentials:

1. Log in to the affected OpenStack controller node.
2. Synchronize the Keystone fernet tokens:

   ```
   su -c "/var/lib/keystone/keystone_keys_rotate.sh -s -t fernet" keystone
   ```

3. If the OpenStack controller node was unavailable at 12:00 a.m., also synchronize the Keystone credentials:

   ```
   su -c "/var/lib/keystone/keystone_keys_rotate.sh -s -t credential" keystone
   ```

Manually remove a broken Swift container

All Swift container operations such as updating a container metadata or removing a container fail if the name of the Swift container includes a forward slash (/) character. Typically, Swift does not accept container names that begin with the forward slash character. However, if you have such containers, you must remove them manually as described below. For example, you may get the following errors when trying to remove a bucket:

```
radosgw-admin bucket rm --bucket='{bucket_name}'
2020-04-23 21:17:37.581515 7f8880047e40 -1 ERROR: get_bucket_instance_from_oid failed: -2
2020-04-23 21:17:37.581535 7f8880047e40  0 could not get bucket info for bucket={bucket_name}
```

> Note
>
> RADOS Gateway uses the term bucket to describe a data container since it may be construed as the equivalent of the term container used within Swift.

> Warning
>
> The following procedure includes changes in an underlying RADOS GATEWAY object and can cause a major malfunction of Swift if performed incorrectly.

To remove a broken Swift container manually:

1. Optional. Identify the ID of the user who created the affected bucket. Run radosgw-admin user check against each user causing an inconsistency error.

   ```
   radosgw-admin user list | grep \" | cut -d\" -f2 | while read uid; do echo '--- $uid'; rdosgw-admin user check --uid $uid; done
   --- {user_id}
   ```

   Example of system response:

   ```
   --- {user_id1}
   --- {user_id2}
   --- {user_id3}
   --- {user_id4}
   --- {user_id5}
   --- {user_id6}
   --- {user_id7}
   2020-04-23 21:17:37.581515 7f8880047e40 -1 ERROR: get_bucket_instance_from_oid failed: -2
   2020-04-23 21:17:37.581535 7f8880047e40  0 could not get bucket info for bucket={bucket_name}
   --- {user_id8}
   --- {user_id9}
   ```

©2025, Mirantis Inc.

2. Remove the link between the user and the bucket:

```
rados -p default.rgw.meta -N users.uid rmomapkey {user_id}.buckets {bucket_name}
```

3. List all metadata objects for the broken bucket. The output contains a unique ID of the RADOS Gateway object.

```
rados -p default.rgw.meta ls --all | grep {bucket_name}
```

Example of system response:

```
root 8ea1a1093ba949deb63010c23caaff5a/{bucket_name}
root .bucket.meta.8ea1a1093ba949deb63010c23caaff5a:{bucket_name}:4d1cb33f-c11c-42ea-8f84-08f1ac56907a.96818.3
```

4. Remove the object using the bucket ID from the previous step. For example:

```
rados -p default.rgw.meta -N root rm 8ea1a1093ba949deb63010c23caaff5a/{bucket_name}
rados -p default.rgw.meta -N root rm .bucket.meta.8ea1a1093ba949deb63010c23caaff5a:{bucket_name}:4d1cb33f-c11c-42ea-8f84-08f1ac56907a.96818.3
```

Troubleshoot OpenContrail

This section includes workarounds for the OpenContrail-related issues in a running MCP cluster.

Troubleshoot the OpenStack-specific issues

This section includes the OpenStack-specific troubleshooting procedures for the OpenContrail services.

Troubleshoot Cassandra for OpenContrail 4.x

In case of issues with Cassandra not starting in OpenContrail 4.x, the system response of the doctrail analyticsdb contrail-status command on the nal nodes can be as follows:

```
== Contrail Database ==
contrail-database:          active
kafka:                      active
contrail-database-nodemgr:  initializing (Cassandra state detected DOWN.
                            Disk space for analytics db not retrievable.)
```

Workaround:

1. Remove all files from the /var/lib/cassandra/ folder:

   ```
   rm -rf /var/lib/cassandra/*
   ```

2. Log in to the nal node in question.
3. Restart the service:

   ```
   doctrail analyticsdb service contrail-database restart
   ```

4. Verify the Cassandra status is active:

   ```
   doctrail analyticsdb contrail-status
   ```

5. Verify that the replication status of Cassandra is UN:

   ```
   doctrail analyticsdb nodetool status
   ```

   Example of system response:

   ```
   Datacenter: datacenter1
   =======================
   Status=Up/Down
   |/ State=Normal/Leaving/Joining/Moving
   --  Address        Load       Tokens  Owns  Host ID                               Rack
   UN  172.17.98.161  255.56 MB  256     ?     eafd7b42-b977-44fa-8c5d-6bed193ac87f  rack1
   UN  172.17.98.163  389.98 MB  256     ?     89493967-bfdb-4300-bb39-08fb6f6d63f1  rack1
   UN  172.17.98.162  292.7 MB   256     ?     91444200-c370-4a6e-b7f8-cab0e78f39eb  rack1
   ```

Troubleshoot Cassandra for OpenContrail 3.2

In case of issues with the OpenContrail 3.2 Cassandra database connection, the system response of the contrail-status command can be as follows:

```
== Contrail Analytics ==
supervisor-analytics:        active
contrail-alarm-gen           active
contrail-analytics-api       active
contrail-analytics-nodemgr   active
contrail-collector           initializing (Database:ctl02:Global connection down)
contrail-query-engine        timeout
contrail-snmp-collector      active
contrail-topology            active
```

Workaround:

1. Restart the supervisor-analytics service:

   ```
   service supervisor-analytics restart
   ```

2. Verify the status of the Neutron API:

   ```
   neutron net-list
   +--------------------------+-------------------------+---------+
   | id                       | name                    | subnets |
   +--------------------------+-------------------------+---------+
   | d6638b91-4e1d-4214-... | ip-fabric               |         |
   | ce66ee12-71b4-44ea-... | __link_local__          |         |
   | d452af3a-3b9f-442e-... | default-virtual-network |         |
   +--------------------------+-------------------------+---------+
   ```

3. Restart nova-api to reflect installed OpenContrail dependencies:

   ```
   salt 'ctl*' service.restart nova-api
   ctl02.workshop.cloudlab.cz:
       True
   ctl03.workshop.cloudlab.cz:
       True
   ctl01.workshop.cloudlab.cz:
       True
   ```

4. Verify the system response of the nova list command:

   ```
   nova list
   ```

Now, the OpenStack and OpenContrail controller nodes are properly deployed.

TCP checksum errors on compute nodes

In the following cases, TCP-based services may not work on VMs or have TCP checksum errors increasing in the output of the dropstats command:

- If the deployment has nested VMs.
- If VMs are running in the ESXi hypervisor.
- If the Network Interface Cards (NICs) do not support the IP checksum calculation and generate an incorrect checksum. For example, the Broadcom Corporation NetXtreme BCM5719 Gigabit Ethernet PCIe NIC cards.

To identify the issue:

1. Inspect the output of the dropstats command that shows the number of Checksum errors.
2. Inspect the output of the tcpdump command for a specific NIC. For example, for enp2s0f1. If you find cksum incorrect entires, the issue exists in your environment.

   ```
   tcpdump -v -nn -l -i enp2s0f1.1162 host 10.0.2.162 | grep -i incorrect
   ```

   Example of system response:

   ```
   tcpdump: listening on enp2s0f1.1162, link-type EN10MB (Ethernet), capture size 262144 bytes
   10.254.19.231.80 > 192.168.100.3.45506: Flags [S.], cksum 0x43bf (incorrect -> 0xb8dc), \
   seq 1901889431, ack 1081063811, win 28960, options [mss 1420,sackOK,\
   TS val 456361578 ecr 41455995,nop,wscale 7], length 0
   10.254.19.231.80 > 192.168.100.3.45506: Flags [S.], cksum 0x43bf (incorrect -> 0xb8dc), \
   seq 1901889183, ack 1081063811, win 28960, options [mss 1420,sackOK,\
   TS val 456361826 ecr 41455995,nop,wscale 7], length 0
   10.254.19.231.80 > 192.168.100.3.45506: Flags [S.], cksum 0x43bf (incorrect -> 0xb8dc), \
   seq 1901888933, ack 1081063811, win 28960, options [mss 1420,sackOK,\
   TS val 456362076 ecr 41455995,nop,wscale 7], length 0
   ```

3. If you do not find the checksum errors using the tcpdump command, inspect the output of the flow -l command that shows the information about a drop for unknown reason.

Workaround:

Turn off the transmit (TX) offloading on all compute nodes for the problematic NIC used by vRouter:

1. Run the following command:

   ```
   ethtool -K <interface_name> tx off
   ```

2. Verify the status of the TX checksumming:

   ```
   ethtool -k <interface_name>
   ```

   Example of system response:

```
tx-checksumming: off
tx-checksum-ipv4: off
tx-checksum-ipv6: off
tx-checksum-sctp: off
tcp-segmentation-offload: off
tx-tcp-segmentation: off [requested on]
tx-tcp6-segmentation: off [requested on]
```

Seealso

Juniper documentation

Troubleshoot HAProxy and LBaaS configuration

The issues with the creation of network namespaces may occur due to an incorrect configuration of HAProxy and LBaaS or due to contrail-svc-monitor not working properly.

Workaround:

1. Restart the contrail-svc-monitor service by restarting the supervisor-config service group:

   ```
   service supervisor-config restart
   ```

2. Verify the OpenContrail status:

   ```
   contrail-status
   ```

   Example of the system response extract:

   ```
   == Contrail Config ==
   supervisor-config:           active
   contrail-api:0               active
   contrail-config-nodemgr      active
   contrail-device-manager      active
   contrail-discovery:0         active
   contrail-schema              active
   contrail-svc-monitor         active
   ifmap                        active
   ```

3. Verify the HAProxy configuration created by LBaaS in /var/lib/contrail/loadbalancer/haproxy/${LB_UUID}/.
4. Verify that HAProxy version >= 1.5 and iproute2 version >= 3.10.0 are installed on all compute nodes.

©2025, Mirantis Inc.

Neutron network ports for LBaaS are not deleted through Heat

If the Neutron network ports for LBaaS are not deleted through Heat, it may occur due to an incorrect Reclass configuration of the OpenStack ctl node that runs the Neutron server.

Workaround:

1. In your project repository, change the directory to /srv/salt/reclass/classes/cluster/<cluster_name>/openstack/.
2. In the classes section of the control.yml file, verify that the following parameter is defined for the ctl node running the Neutron server:

   ```
   - system.opencontrail.client.cluster
   ```

3. For the Contrail-specific Heat templates, define the following parameter in the classes section of the /srv/salt/reclass/classes/cluster/<cluster_name>/openstack/control.yml file:

   ```
   - system.heat.server.resource.contrail
   ```

Troubleshoot a VM network outage

This section explains how to troubleshoot a VM network outage.

To troubleshoot a VM network outage:

1. Verify the disk space, CPU load, and RAM on the VM in question.
2. Verify that the VM is enabled and has all interfaces up. You can do this using the Horizon Dashboard in the Admin > Instances tab or using CLI:

   ```
   # Get VM status
   nova list --all-tenants | grep <vmname>

   # Get hypervisor name
   nova show <vmname>
   ```

3. Ping the default gateway using ip r and ip a* and other VMs on the same network to identify whether it is a global, VM-related, or hypervisor-related problem.

   Each VM has a virtual gateway usually at the first address. Pinging of a virtual gateway means that network connection between the VM and the hypervisor vRouter is not broken. This can show you a broken network connection inside the VM.

   If you can ping the default gateway, but not anything outside or you cannot ping other VMs inside the virtual network, it can be a hypervisor-related issue. If it is the case, follow the steps below:

   1. Log in to Horizon.
   2. Identify the OpenContrail controller node that runs the VM hypervisor.
   3. Log in to that OpenContrail controller node.
   4. Verify the status of the supervisor-vrouter service using the contrail-status command.
   5. If the supervisor-vrouter status is inactive:

      1. Restart supervisor-vrouter.
      2. Inspect the /var/log/contrail/contrail-vrouter* logs.
4. Verify that the VM unavailability is not caused by the firewall rules set in Security Groups or Network Policies in Horizon. Verify the security groups associated with the VM and the network policies attached to the virtual network.
5. Verify the peering status in the OpenContrail web UI navigating to Monitor > Control Nodes > Choose of them > Peers. The status should be Established, in sync. If it is not the case, select from the following options:

   - Verify the availability of network devices. If the network devices are in the expected status (active by default), you can restart all OpenContrail controller nodes in sequence. Though, verify that at least two OpenContrail controller nodes are up and in a correct state. You should never restart all nodes at once.
   - Restart the supervisor-control service and verify whether it is in the active status using the contrail-status command.

6. Verify the OpenContrail status on the OpenContrail node that runs the hypervisor with the VM.

   If the contrail-status output contains some services not in the active status, restart the supervisor-vrouter process and verify the status again.

7. To identify the problem with the vRouter on the hypervisor or a VM inside the network setup, ping another VM on the same hypervisor or link-Local.

   You can also ping the link-local IP address. To get this address, select from the following options:

   * Using the OpenContrail web UI, find the VM details on the Contrail Dashboard -> vRouter with VM page.
   * Using CLI, run the following command on the OpenStack compute node in question:

     ```
     netstat -r
     ```

     Example of system response:

     ```
     Kernel IP routing table
     Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
     default         10.0.106.1      0.0.0.0         UG        0 0          0 vhost0
     localnet        *               255.255.255.0   U         0 0          0 vhost0
     169.254.0.3     *               255.255.255.255 UH        0 0          0 vhost0
     169.254.0.4     *               255.255.255.255 UH        0 0          0 vhost0
     169.254.0.5     *               255.255.255.255 UH        0 0          0 vhost0
     ```

   * Using ssh to the VM from the hypervisor, run the following command:

     ```
     vif --list
     ```

     Example of system response:

     ```
     Vrouter Interface Table

     Flags: P=Policy, X=Cross Connect, S=Service Chain, Mr=Receive Mirror
            Mt=Transmit Mirror, Tc=Transmit Checksum Offload, L3=Layer 3, L2=Layer 2
            D=DHCP, Vp=Vhost Physical, Pr=Promiscuous, Vnt=Native Vlan Tagged
            Mnp=No MAC Proxy

     vif0/0      OS: bond0.3034 (Speed 20000, Duplex 1)
                 Type:Physical HWaddr:a4:1f:72:0a:93:8c IPaddr:0
                 Vrf:0 Flags:TcL3L2Vp MTU:1514 Ref:22
                 RX packets:9294622  bytes:1402159738 errors:0
                 TX packets:14035541  bytes:10121866276 errors:0
     ```

```
vif0/1      OS: vhost0
            Type:Host HWaddr:a4:1f:72:0a:93:8c IPaddr:0
            Vrf:0 Flags:L3L2 MTU:1514 Ref:3
            RX packets:9649123  bytes:9390647810 errors:0
            TX packets:5671040  bytes:993417128 errors:0
```

8. Inspect the OpenContrail log files in /var/log/contrail/* and the /var/log/contrail/contrail-vrouter*.log file to debug vRouter, in particular.
9. If the problem is related to the networking inside the VM, verify the interface configuration, the DNS resolv.conf file, routing table, and so on.

Connectivity-related OpenContrail issues

This section outlines different use cases related to the connectivity issues in an OpenContrail cluster.

Connection from a VM to the Internet does not work

This section describes how to troubleshoot connectivity failures from a VM to the Internet.

Verify the OpenContrail schema flapping

The connection issues from a VM to the Internet may be caused by the OpenContrail schema flapping. The contrail-schema may be flapping, for example, if at least one OpenContrail object has a non-ASCII character in its name.

To verify that OpenContrail schema is not flapping:

1. Log in to any Mirantis OpenContrail controller ntw node.
2. Run one of the following commands:

   - Using the contrail-status command:

     ```
     watch -n 5 "contrail-status -d | grep schema"
     ```

     Example of system response:

     ```
     Every 5.0s: contrail-status -d | grep schema       Thu Nov 23 09:33:35 2017

     contrail-schema              backup           pid 28441, uptime 0 days, 00:00:31
     ```

     The contrail-schema status must be backup or active.

   - Using the contrail-schema status command:

     ```
     watch -n 5 "service contrail-schema status"
     ```

     Example of system response:

     ```
     Every 5.0s: service contrail-schema status          Thu Nov 23 09:36:28 2017

     contrail-schema                 RUNNING    pid 28441, uptime 0 days, 00:00:24
     ```

     The contrail-schema status must be RUNNNING.

     If the status is other than RUNNING, verify the contrail-schema logs to fix the issue. See step 5 for details.
3. Repeat the previous step on all ntw nodes.
4. Using the commands from the step 2, verify uptime and pid of contrail-schema. If pid is changing or uptime is reset every one or two minutes, schema is flappig.
5. If the OpenContrail schema is flapping, inspect its logs located in the /var/log/contrail/contrail-schema-stdout.log and /var/log/contrail/contrail-schema.log directories.
   In the logs, if the to_bgp.py or config_db.py file are affected, it may be caused by a non-ASCII character in the name of some object. For example, tenant, network, subnet names, and so on.

Verify the OpenStack security groups

If you have connectivity failures between VMs, you may need to troubleshoot the OpenStack security groups. You can verify the OpenStack security groups of a VM using CLI or Horizon.

To verify the OpenStack security groups using CLI:

1. Log in to any OpenStack controller node using CLI.
2. List the IDs of the security groups:

```
openstack security group list --project <project_id>

+--------------------------------------+------------+------------------------+---------+
| ID                                   | Name       | Description            | Project |
+--------------------------------------+------------+------------------------+---------+
| 16463a93-6d87-4e2f-8f5b-8954ed6a243b | default    | Default security group |         |
| 3b2490a1-8efb-4208-a202-455710088ac8 | other_secg | Security group         |         |
+--------------------------------------+------------+------------------------+---------+
```

3. Verify the rules of the security group. For example

```
openstack security group show 16463a93-6d87-4e2f-8f5b-8954ed6a243b

+---------------+------------------------------------------------------------------------------+
|Field          |Value                                                                         |
+---------------+------------------------------------------------------------------------------+
|created_at     |None                                                                          |
|description    |Default security group                                                        |
|id             |16463a93-6d87-4e2f-8f5b-8954ed6a243b                                          |
|name           |default                                                                       |
|project_id     |cf9b8bd8667b4b53a65192a486c4ab9c                                              |
|revision_number|None                                                                          |
|rules          |direction='egress', ethertype='IPv4', id='97b1a242-ef00-4ff3-87f0-63c405c73570', |
|               |port_range_max='65535', protocol='any', remote_ip_prefix='0.0.0.0/0'          |
|               |direction='egress', ethertype='IPv6', id='9e037646-d99c-4e00-baf1-b422fa839253', |
|               |port_range_max='65535', protocol='any', remote_ip_prefix='::/0'               |
|               |direction='ingress', ethertype='IPv4', id='5a4ac047-a410-4e03-a41f-261de909ed0a',|
|               |port_range_max='65535', protocol='any', remote_ip_prefix='0.0.0.0/0'          |
|updated_at     |None                                                                          |
+---------------+------------------------------------------------------------------------------+
```

To verify the OpenStack security groups using Horizon:

1. Log in to Horizon.
2. Go to Project > Network > Security Groups.
3. Select the security group that is used by the VM in question.
4. Add a new test rule that allows all ingress/egress traffic.
5. Test the connectivity between two VMs.

6. If connectivity fails, follow the steps described in Verify the IP address and default gateway on a VM.

Verify the IP address and default gateway on a VM

This section describes how to verify and troubleshoot issues with the IP address and the default gateway of a VM.

To verify the IP address and default gateway on a VM:

1. Log in to the VM in question.
2. Run the following command:

   ```
   ip a
   ```

   Example of system response:

   ```
   1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
       inet 127.0.0.1/8 scope host lo
          valid_lft forever preferred_lft forever
       inet6 ::1/128 scope host
          valid_lft forever preferred_lft forever
   3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
       link/ether ac:de:48:92:9d:69 brd ff:ff:ff:ff:ff:ff
       inet 10.167.4.21/24 brd 10.167.4.255 scope global eth1
          valid_lft forever preferred_lft forever
       inet6 fe80::aede:48ff:fe92:9d69/64 scope link
          valid_lft forever preferred_lft forever
   ```

   - If Ethernet does not have an IP address, run the following command:

     ```
     dhclient eth1
     ```

   - If Ethernet has an IP address with the /32 mask:
     1. Restart the VM.
     2. Verify ifmap-server.
   - If the interface is down, run the following command:

     ```
     ifcongfig eth1 up
     ```

3. Display the default gateway:

   ```
   route -n
   ```

   Example of system response:

   ```
   Kernel IP routing table
   Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
   ```

```
0.0.0.0         10.167.4.1      0.0.0.0         UG    0      0        0 eth1
10.167.4.0      0.0.0.0         255.255.255.0   U     0      0        0 eth1
```

4. Ping the default gateway.

Verify the IP and DNS traffic

This section describes how to troubleshoot issues with the IP and DNS traffic on a VM.

To verify the IP traffic:

1. Log in to the VM in question.
2. Ping any IP, for example, Google DNS 8.8.8.8:

   ```
   ping 8.8.8.8 -c 3
   ```

   Example of system response:

   ```
   PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
   64 bytes from 8.8.8.8: icmp_seq=1 ttl=59 time=4.12 ms
   64 bytes from 8.8.8.8: icmp_seq=2 ttl=59 time=4.37 ms
   64 bytes from 8.8.8.8: icmp_seq=3 ttl=59 time=4.03 ms

   --- 8.8.8.8 ping statistics ---
   3 packets transmitted, 3 received, 0% packet loss, time 2002ms
   rtt min/avg/max/mdev = 4.038/4.180/4.377/0.161 ms
   ```

3. If pinging does not work, use the traceroute command on the same address to identify the issue.

To verify the DNS traffic:

1. Log in to the VM in question.
2. Run the following command:

   ```
   dig google.com
   ```

   Example of system response:

   ```
   ; <<>> DiG 9.9.5-3ubuntu0.16-Ubuntu <<>> google.com
   ;; global options: +cmd
   ;; Got answer:
   ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15428
   ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 5

   ;; OPT PSEUDOSECTION:
   ; EDNS: version: 0, flags:; udp: 4096
   ;; QUESTION SECTION:
   ;google.com.                    IN      A

   ;; ANSWER SECTION:
   google.com.             300     IN      A       216.58.201.110
   ```

```
;; Query time: 20 msec
;; SERVER: 172.17.32.194#53(172.17.32.194)
;; WHEN: Thu Nov 23 14:14:41 UTC 2017
;; MSG SIZE  rcvd: 191
```

In the output above, the VM resolved the google.com domain to the IP address 216.58.201.110.

3. If the domain to the IP address is not resolvable:

    1. Run the following command to identify when the resolve of the DNS server stops:

    ```
    dig google.com +trace
    ```

    2. Dig a different server name. For example:

    ```
    dig  google.com @8.8.8.8
    ```

Verify the vhost0 interface on a compute node

If you have connectivity failures from a VM to the Internet, it may be caused by issues with the vhost0 interface on a compute node.

To verify the vhost0 interface on a compute node:

1. Log in to the OpenStack compute node with a failed VM.
2. Verify the status of the vhost0 interface:

   ```
   ifconfig vhost0
   ```

   Example of system response:

   ```
   vhost0    Link encap:Ethernet  HWaddr 0c:c4:7a:15:f8:f9
             inet addr:10.167.4.101  Bcast:10.167.4.255  Mask:255.255.255.0
             inet6 addr: fe80::ec4:7aff:fe15:f8f9/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
             RX packets:45371419 errors:0 dropped:0 overruns:0 frame:0
             TX packets:11440313 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:4598915477 (4.5 GB)  TX bytes:13876287735 (13.8 GB)
   ```

3. If the interface does not exist:
   1. Verify the settings of the vhost0 interface in /etc/network/interfaces.
   2. Verify the vRouter kernel modules that are currently being used:

      ```
      lsmod | grep -i -e contrail -e router
      ```

4. If the interface does not have an IP address:
   1. Verify the settings of the vhost0 interface in /etc/network/interfaces.
   2. Run the following command:

      ```
      ifconfig vhost0 down && ifconfig vhost0 up
      ```

©2025, Mirantis Inc.

Verify the floating IP

This section describes how to verify the floating IP on an MX Series router and a virtual router.

The floating IP verification process is as follows:

1. Verify a route propagation to the gateway.
2. Ping next hop.
3. Verify flow on the vRouter.
4. Verify duplicated floating IPs.

To verify a route propagation on the vMX/vSRX routers:

> Note
>
> Use the Junos CLI during the verification procedure.

1. Log in to the vsrx1 router.
2. Run the following command:

   ```
   root@vsrx1> run show route table public.inet.0
   ```

   Example of system response:

   ```
   public.inet.0: 8 destinations, 13 routes (8 active, 0 holddown, 0 hidden)
   + = Active Route, - = Last Active, * = Both

   0.0.0.0/0          *[Static/5] 4w6d 22:49:22
                    > to 172.17.32.193 via ge-0/0/0.0
   172.17.32.192/26   *[Direct/0] 4w6d 22:49:22
                    > via ge-0/0/0.0
   172.17.32.240/32   *[Local/0] 4w6d 22:49:22
                      Local via ge-0/0/0.0
   <floating_ip0>/32  *[BGP/170] 4w3d 00:41:07, MED 100, localpref 200, from 10.167.4.22
                      AS path: ?
                    > via gr-0/0/0.32769, Push 40
                    [BGP/170] 3w3d 00:30:48, MED 100, localpref 200, from 10.167.4.23
                      AS path: ?
                    > via gr-0/0/0.32769, Push 40
   ```

3. In the output of the previous command, find the floating IP address that you want to debug.
4. Run the following command to output details of the floating IP in question. For example:

   ```
   root@vsrx1> show route <floating_ip0>/32 detail
   ```

5. Ping the floating IP in question using the following command:

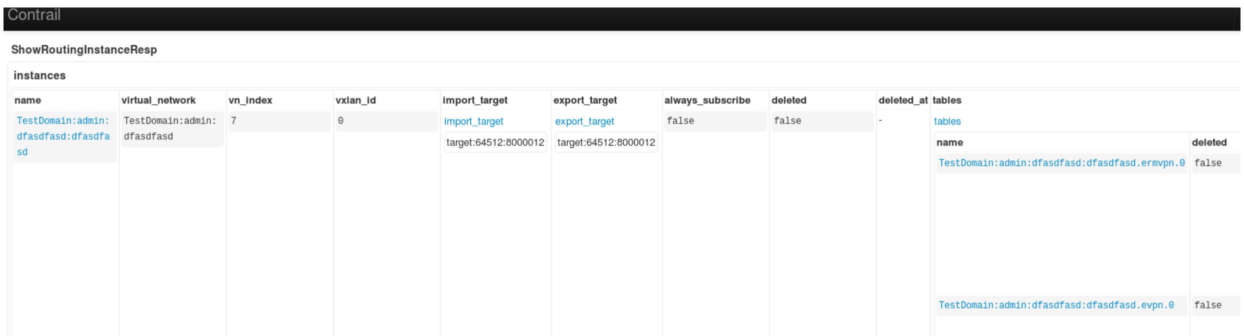
```
root@vsrx1> ping <floating_ip0> routing-instance public count 10
```

To check a route exchange on the Mirantis OpenContrail controller nodes:

1. Validate the presence of the routing instance for each virtual network in the following sample system:

```
http://<ntw_node_vip>:8083/Snh_ShowRoutingInstanceReq?name=
```

2. Using the link from the previous step, find public network with floating IP pool.
3. Using the same link, find and click the public:inet.0 table.
4. On the opened page, find and click a floating IP that you want to check. For example:



For a detailed MX troubleshooting procedure, see: Troubleshoot the vMX router.

To verify forward and reverse flow on a vRouter

The OpenContrail vRouter is located on the compute cmp nodes. Use the following steps to troubleshoot the OpenContrail user issues related to service chaining, communication issues between virtual machines, between two virtual networks, and so on.

1. Log in to any compute node of your MCP cluster. For example, cmp002.
2. Run the following command:

```
flow -l | grep '<floating_ip>\|192.168.0.100'
```

Example of system response:

```
 Index              Source:Port/Destination:Port                     Proto(V)
-----------------------------------------------------------------------------------
492152<=>1500364      <floating_ip>:792                              1 (5)
                      192.168.0.100:0
(Gen: 1, K(nh):83, Action:F, Flags:, QOS:-1, S(nh):83,  Stats:487/519142,
```

```
1500364<=>492152      192.168.0.100:792                          1 (5)
                      <floating_ip>:0
(Gen: 1, K(nh):83, Action:F, Flags:, QOS:-1, S(nh):35,  Stats:487/519142,
```

For a detailed forward and reverse flow troubleshooting procedure, see: Troubleshoot a VM forward and reverse flow.

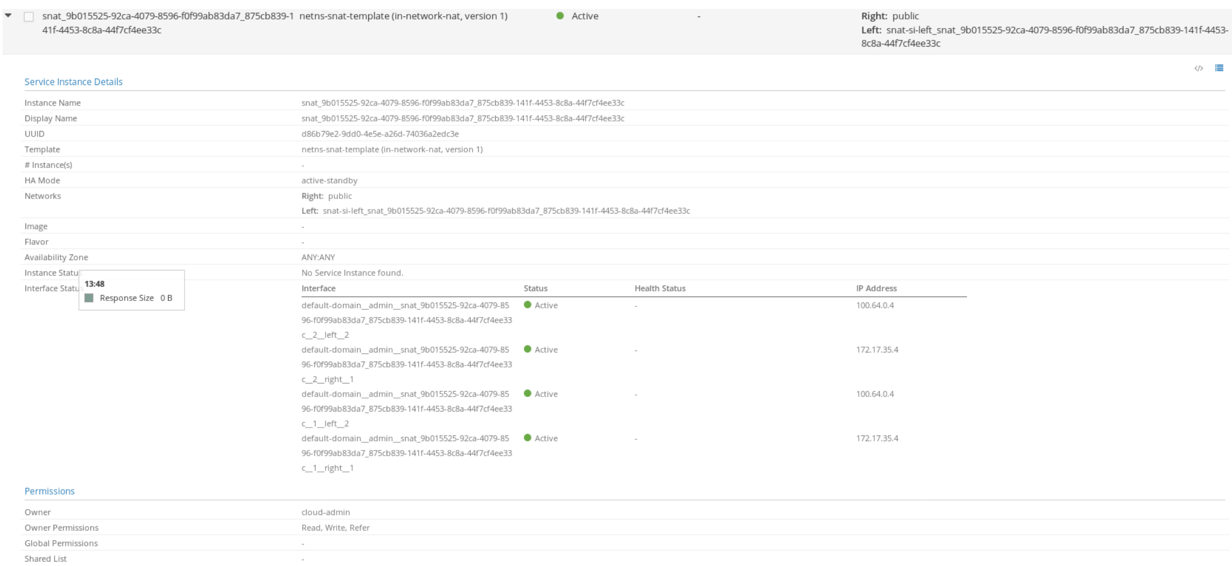
Verify SNAT and other services instances

The connectivity issues from a VM to the Internet may be related to the issues with the Source Network Address Translation (SNAT) router instance.

To verify an SNAT instance:

1. Log in to the OpenContrail web UI.
2. Go to Configure > Services > Service Instances.
3. Find an SNAT instance that is used by the VM in question. Here you can see the current status of the SNAT router instance.
4. In the Service Instance Details window, inspect the SNAT router instance and its interfaces' statuses.
5. In the same window, verify the floating IP address of the active network interface.

   Example of an SNAT instance details:

6. Follow the steps described in Verify the OpenContrail svc-monitor.
7. Follow the steps described in Verify the default route in VRF.
8. Follow the steps described in Troubleshoot a VM forward and reverse flow.

Connection from the Internet to a VM does not work

This section describes how to troubleshoot connectivity failures from the Internet to a VM.

©2025, Mirantis Inc.

Verify the OpenStack security groups

If you have connectivity failures between VMs, you may need to troubleshoot the OpenStack security groups. You can verify the OpenStack security groups of a VM using CLI or Horizon.

To verify the OpenStack security groups using CLI:

1. Log in to any OpenStack controller node using CLI.
2. List the IDs of the security groups:

```
openstack security group list --project <project_id>

+--------------------------------------+------------+------------------------+---------+
| ID                                   | Name       | Description            | Project |
+--------------------------------------+------------+------------------------+---------+
| 16463a93-6d87-4e2f-8f5b-8954ed6a243b | default    | Default security group |         |
| 3b2490a1-8efb-4208-a202-455710088ac8 | other_secg | Security group         |         |
+--------------------------------------+------------+------------------------+---------+
```

3. Verify the rules of the security group. For example

```
openstack security group show 16463a93-6d87-4e2f-8f5b-8954ed6a243b

+---------------+------------------------------------------------------------------------------+
|Field          |Value                                                                         |
+---------------+------------------------------------------------------------------------------+
|created_at     |None                                                                          |
|description    |Default security group                                                        |
|id             |16463a93-6d87-4e2f-8f5b-8954ed6a243b                                          |
|name           |default                                                                       |
|project_id     |cf9b8bd8667b4b53a65192a486c4ab9c                                              |
|revision_number|None                                                                          |
|rules          |direction='egress', ethertype='IPv4', id='97b1a242-ef00-4ff3-87f0-63c405c73570', |
|               |port_range_max='65535', protocol='any', remote_ip_prefix='0.0.0.0/0'          |
|               |direction='egress', ethertype='IPv6', id='9e037646-d99c-4e00-baf1-b422fa839253', |
|               |port_range_max='65535', protocol='any', remote_ip_prefix='::/0'               |
|               |direction='ingress', ethertype='IPv4', id='5a4ac047-a410-4e03-a41f-261de909ed0a',|
|               |port_range_max='65535', protocol='any', remote_ip_prefix='0.0.0.0/0'          |
|updated_at     |None                                                                          |
+---------------+------------------------------------------------------------------------------+
```

To verify the OpenStack security groups using Horizon:

1. Log in to Horizon.
2. Go to Project > Network > Security Groups.
3. Select the security group that is used by the VM in question.
4. Add a new test rule that allows all ingress/egress traffic.
5. Test the connectivity between two VMs.

6. If connectivity fails, follow the steps described in Verify the IP address and default gateway on a VM.

Verify the IP address and default gateway on a VM

This section describes how to verify and troubleshoot issues with the IP address and the default gateway of a VM.

To verify the IP address and default gateway on a VM:

1. Log in to the VM in question.
2. Run the following command:

   ```
   ip a
   ```

   Example of system response:

   ```
   1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
       inet 127.0.0.1/8 scope host lo
          valid_lft forever preferred_lft forever
       inet6 ::1/128 scope host
          valid_lft forever preferred_lft forever
   3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
       link/ether ac:de:48:92:9d:69 brd ff:ff:ff:ff:ff:ff
       inet 10.167.4.21/24 brd 10.167.4.255 scope global eth1
          valid_lft forever preferred_lft forever
       inet6 fe80::aede:48ff:fe92:9d69/64 scope link
          valid_lft forever preferred_lft forever
   ```

   - If Ethernet does not have an IP address, run the following command:

     ```
     dhclient eth1
     ```

   - If Ethernet has an IP address with the /32 mask:

     1. Restart the VM.
     2. Verify ifmap-server.

   - If the interface is down, run the following command:

     ```
     ifcongfig eth1 up
     ```

3. Display the default gateway:

   ```
   route -n
   ```

   Example of system response:

   ```
   Kernel IP routing table
   Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
   ```

```
0.0.0.0         10.167.4.1      0.0.0.0         UG    0      0        0 eth1
10.167.4.0      0.0.0.0         255.255.255.0   U     0      0        0 eth1
```

4. Ping the default gateway.

Verify the IP and DNS traffic

This section describes how to troubleshoot issues with the IP and DNS traffic on a VM.

To verify the IP traffic:

1. Log in to the VM in question.
2. Ping any IP, for example, Google DNS 8.8.8.8:

   ```
   ping 8.8.8.8 -c 3
   ```

   Example of system response:

   ```
   PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
   64 bytes from 8.8.8.8: icmp_seq=1 ttl=59 time=4.12 ms
   64 bytes from 8.8.8.8: icmp_seq=2 ttl=59 time=4.37 ms
   64 bytes from 8.8.8.8: icmp_seq=3 ttl=59 time=4.03 ms

   --- 8.8.8.8 ping statistics ---
   3 packets transmitted, 3 received, 0% packet loss, time 2002ms
   rtt min/avg/max/mdev = 4.038/4.180/4.377/0.161 ms
   ```

3. If pinging does not work, use the traceroute command on the same address to identify the issue.

To verify the DNS traffic:

1. Log in to the VM in question.
2. Run the following command:

   ```
   dig google.com
   ```

   Example of system response:

   ```
   ; <<>> DiG 9.9.5-3ubuntu0.16-Ubuntu <<>> google.com
   ;; global options: +cmd
   ;; Got answer:
   ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15428
   ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 5

   ;; OPT PSEUDOSECTION:
   ; EDNS: version: 0, flags:; udp: 4096
   ;; QUESTION SECTION:
   ;google.com.                    IN      A

   ;; ANSWER SECTION:
   google.com.             300     IN      A       216.58.201.110
   ```

```
;; Query time: 20 msec
;; SERVER: 172.17.32.194#53(172.17.32.194)
;; WHEN: Thu Nov 23 14:14:41 UTC 2017
;; MSG SIZE  rcvd: 191
```

In the output above, the VM resolved the google.com domain to the IP address 216.58.201.110.

3. If the domain to the IP address is not resolvable:

   1. Run the following command to identify when the resolve of the DNS server stops:

      ```
      dig google.com +trace
      ```

   2. Dig a different server name. For example:

      ```
      dig  google.com @8.8.8.8
      ```

Verify the vhost0 interface on a compute node

If you have connectivity failures from a VM to the Internet, it may be caused by issues with the vhost0 interface on a compute node.

To verify the vhost0 interface on a compute node:

1. Log in to the OpenStack compute node with a failed VM.
2. Verify the status of the vhost0 interface:

   ```
   ifconfig vhost0
   ```

   Example of system response:

   ```
   vhost0    Link encap:Ethernet  HWaddr 0c:c4:7a:15:f8:f9
             inet addr:10.167.4.101  Bcast:10.167.4.255  Mask:255.255.255.0
             inet6 addr: fe80::ec4:7aff:fe15:f8f9/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:9000  Metric:1
             RX packets:45371419 errors:0 dropped:0 overruns:0 frame:0
             TX packets:11440313 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:4598915477 (4.5 GB)  TX bytes:13876287735 (13.8 GB)
   ```

3. If the interface does not exist:

   1. Verify the settings of the vhost0 interface in /etc/network/interfaces.
   2. Verify the vRouter kernel modules that are currently being used:

      ```
      lsmod | grep -i -e contrail -e router
      ```

4. If the interface does not have an IP address:

   1. Verify the settings of the vhost0 interface in /etc/network/interfaces.
   2. Run the following command:

      ```
      ifconfig vhost0 down && ifconfig vhost0 up
      ```

Verify the floating IP

This section describes how to verify the floating IP on an MX Series router and a virtual router.

The floating IP verification process is as follows:

1. Verify a route propagation to the gateway.
2. Ping next hop.
3. Verify flow on the vRouter.
4. Verify duplicated floating IPs.

To verify a route propagation on the vMX/vSRX routers:

> Note
>
> Use the Junos CLI during the verification procedure.

1. Log in to the vsrx1 router.
2. Run the following command:

   ```
   root@vsrx1> run show route table public.inet.0
   ```

   Example of system response:

   ```
   public.inet.0: 8 destinations, 13 routes (8 active, 0 holddown, 0 hidden)
   + = Active Route, - = Last Active, * = Both

   0.0.0.0/0          *[Static/5] 4w6d 22:49:22
                    > to 172.17.32.193 via ge-0/0/0.0
   172.17.32.192/26   *[Direct/0] 4w6d 22:49:22
                    > via ge-0/0/0.0
   172.17.32.240/32   *[Local/0] 4w6d 22:49:22
                      Local via ge-0/0/0.0
   <floating_ip0>/32  *[BGP/170] 4w3d 00:41:07, MED 100, localpref 200, from 10.167.4.22
                      AS path: ?
                    > via gr-0/0/0.32769, Push 40
                    [BGP/170] 3w3d 00:30:48, MED 100, localpref 200, from 10.167.4.23
                      AS path: ?
                    > via gr-0/0/0.32769, Push 40
   ```

3. In the output of the previous command, find the floating IP address that you want to debug.
4. Run the following command to output details of the floating IP in question. For example:

   ```
   root@vsrx1> show route <floating_ip0>/32 detail
   ```

5. Ping the floating IP in question using the following command:

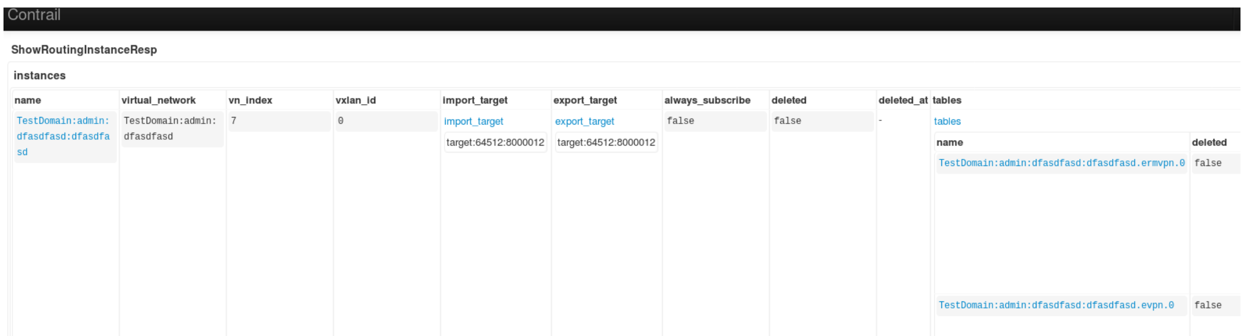
```
root@vsrx1> ping <floating_ip0> routing-instance public count 10
```

To check a route exchange on the Mirantis OpenContrail controller nodes:

1. Validate the presence of the routing instance for each virtual network in the following sample system:

```
http://<ntw_node_vip>:8083/Snh_ShowRoutingInstanceReq?name=
```

2. Using the link from the previous step, find public network with floating IP pool.
3. Using the same link, find and click the public:inet.0 table.
4. On the opened page, find and click a floating IP that you want to check. For example:



For a detailed MX troubleshooting procedure, see: Troubleshoot the vMX router.

To verify forward and reverse flow on a vRouter

The OpenContrail vRouter is located on the compute cmp nodes. Use the following steps to troubleshoot the OpenContrail user issues related to service chaining, communication issues between virtual machines, between two virtual networks, and so on.

1. Log in to any compute node of your MCP cluster. For example, cmp002.
2. Run the following command:

```
flow -l | grep '<floating_ip>\|192.168.0.100'
```

Example of system response:

```
 Index              Source:Port/Destination:Port                      Proto(V)
-----------------------------------------------------------------------------------
492152<=>1500364      <floating_ip>:792                                1 (5)
                      192.168.0.100:0
(Gen: 1, K(nh):83, Action:F, Flags:, QOS:-1, S(nh):83,  Stats:487/519142,
```

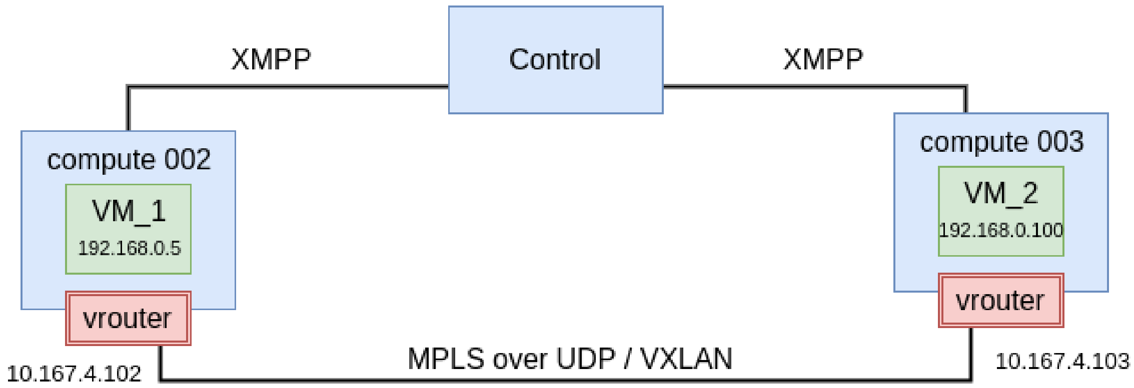
```
1500364<=>492152      192.168.0.100:792                     1 (5)
                      <floating_ip>:0
(Gen: 1, K(nh):83, Action:F, Flags:, QOS:-1, S(nh):35,  Stats:487/519142,
```

For a detailed forward and reverse flow troubleshooting procedure, see: Troubleshoot a VM forward and reverse flow.

Connection between VMs does not work

This section describes how to troubleshoot the connectivity issues between VMs.

In this section, the following example of communication between VM_1 and VM_2 is described:



To troubleshoot the connection between VMs:

1. Log in to an OpenStack compute node. For example, cmp002.
2. Verify the flow between VMs. For example:

```
flow -l | grep '192.168.0.5\|192.168.0.100'
```

   Example of system response:

```
Index                Source:Port/Destination:Port                      Proto(V)
-----------------------------------------------------------------------------------
492152<=>1500364      192.168.0.5:792                                   1 (5)
                      192.168.0.100:0
(Gen: 1, K(nh):83, Action:F, Flags:, QOS:-1, S(nh):83,  Stats:487/519142,


1500364<=>492152      192.168.0.100:792                                 1 (5)
                      192.168.0.5:0
(Gen: 1, K(nh):83, Action:F, Flags:, QOS:-1, S(nh):35,  Stats:487/519142,
```

3. Use the output from the previous step to get the information on VRF by ID. For example, for S(nh):83 and S(nh):35:

```
nh --get 83

Id:83         Type:Encap         Fmly: AF_INET  Rid:0  Ref_cnt:4         Vrf:5
              Flags:Valid, Policy,
              EncapFmly:0806 Oif:8 Len:14
```

```
              Encap Data: 02 29 64 b3 e2 f4 00 00 5e 00 01 00 08 00

nh --get 35

Id:35         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:4850       Vrf:0
              Flags:Valid, MPLSoUDP,
              Oif:0 Len:14 Flags Valid, MPLSoUDP,  Data:0c c4 7a 50 27 88 0c c4 7a 17 99 5d 08 00
              Vrf:0  Sip:10.167.4.102  Dip:10.167.4.103
```

4. Verify the routing table for VRF (routing instance). For example:

```
rt --dump 5 | grep 192.168.0.5/32
```

Example of system response:

```
192.168.0.5/32       32       P       -       83       2:29:64:b3:e2:f4(240568)
```

```
rt --dump 5 | grep 192.168.0.100/32
```

Example of system response:

```
192.168.0.100/32       32       LP       44       35       2:d1:32:42:b5:87(149684)
```

If the above procedure does not resolve the connectivity issue, proceed with the following steps:

1. Verify that a network policy is created using the OpenContrail web UI in Configure > Networking > Policies.
2. Verify that a network policy is assigned to virtual networks (VNs) using OpenContrail web UI in Configure > Networking > Network > Edit Network > Network Policy(s).
3. Verify the rules of security group(s).
4. Verify that VNs are assigned to a correct security group.
5. Verify whether the prefixes were exchanged in the routing table using the rt --dump <VRF_ID> command.
6. Verify MTU on a physical interface used by vRouter.
7. Verify the MTU jumbo frames in the underlay network.

Seealso

Troubleshoot a VM forward and reverse flow

VM does not have link-local (169.254.x.x) address upon boot

To troubleshoot the problem:

1. Verify the hypervisor at <hypervisor_ip>:8085/Snh_ItfReq to identify whether the configuration is present. For example:

   ```
   http://10.92.249.119:8085/Snh_ItfReq?name=
   ```

   If the configuration is missing, it means that the hypervisor did not receive it from the OpenContrail controller node.

2. Verify api-server at <ip>:8082/virtual-machine-interface/<uuid> to identify whether routing_instance_refs are present. For example:

   ```
   curl -u admin:secret123 http://localhost:8095/virtual-machine-interface/39f | \
   python -m json.tool > /tmp/vmis
   ```

3. If routing_instance_refs are missing:

   1. Verify that contrail-schema is running on at least one OpenContrail controller node:

      ```
      contrail-status
      ```

   2. Verify that VMI is present in ifmap on all OpenStack controller nodes:

      ```
      ifmap-view localhost 8443 visual visual | grep <vmi-name>
      ```

      If the VMI is missing in ifmap:

      1. Verify that RabbitMQ is clustered successfully and all API servers are connected to it:

         ```
         rabbitmqctl cluster_status
         ```

      2. Verify that contrail-api introspects the port at 8084 for rest/db/messagebus/ifmap.

   3. To verify why contrail-schema does not establish the link between VMI and RI, inspect /var/log/contrail/schema.err.

   4. Inspect the OpenContrail logs:

      ```
      contrail-logs --object-type config --object-values
      contrail-logs --object-type config --object-id <id_from_above_command>
      ```

4. Inspect the latest log from OpenContrail to stdout and other logs using the contrail-logs command.

5. Verify the connection status, for example:

```
http://<config-node-where-schema-active>:8084/Snh_SandeshUVECacheReq?x=NodeStatus
```

Troubleshoot SNAT and LBaaS namespaces

SNAT or LBaaS provisions two namespaces in the active-standby mode on two random OpenStack compute nodes. To display the namespaces, use the ip netns command.

The namespaces with vrouter-UUID represent SNAT and the namespaces with vrouter-UUID-UUID represent LBaaS. Namespaces can be managed as standard Linux networking using the netns command.

Slow response time from JMX Exporter

The third-party jmx-exporter service used by StackLight for exporting the OpenContrail Cassandra metrics may have a slow response time on the ntw node where the Cassandra backup is enabled. Usually, this is the ntw01 node.

You may also detect the following symptoms of the issue:

- Grafana does not display metrics for Cassandra.
- The PrometheusTargetDown alert for the jmx_cassandra_exporter job appears in the FIRING state for the ntw0x node.
- The contrail-database-nodemgr service status is initializing.

Workaround:

1. Log in to the ntw01 node.
2. Verify that the Cassandra snapshots are automatically backed up in /var/backups/cassandra/. Otherwise, manually back them up in /var/lib/cassandra/data.
3. Clear the Cassandra snapshots. For example:
   - For OpenContrail 4.x:

   ```
   doctrail controller nodetool -h localhost -p 7198 clearsnapshot
   ```

   - For OpenContrail 3.2:

   ```
   nodetool -h localhost -p 7198 clearsnapshot
   ```

4. If clearing of snapshots does not resolve the issue, increase the scrape_interval and scrape_timeout values for jmx_cassandra_exporter:
   1. Open your Git project repository with the Reclass model on the cluster level.
   2. In cluster/<cluster_name>/stacklight/server.yml, modify the scrape parameters. For example:

      ```
      prometheus:
        server:
          target:
            static:
              jmx_cassandra_exporter:
                scheme: http
                metrics_path: /metrics
                honor_labels: False
                scrape_interval: 60s
                scrape_timeout: 60s
      ```

   3. Log in to the Salt Master node.

©2025, Mirantis Inc.

4. Apply the changes to the Reclass model:

```
salt 'cfg01*' state.apply reclass.storage
salt '*' saltutil.sync_all
```

5. Apply the following state:

```
salt -C 'I@prometheus:server' state.sls prometheus
```

6. Connect to Grafana as described in Connect to Grafana.
7. Navigate to the Cassandra dashboard.
8. Verify that the rate_interval value is more than 1m.

©2025, Mirantis Inc.

FIP is not associated to an LB port through Terraform

If you use Terraform for creating resources in OpenStack, due to a limitation, Terraform cannot associate a floating IP (FIP) address to a load balancer (LB) port managed by the Avi Vantage tool. This limitation is related to the following resource in terraform-openstack-provider:

```
resource "openstack_networking_floatingip_associate_v2" "association" {
  floating_ip = "<Floating-ip address>"
  port_id = "<LoadBalancer port id>"
}
```

Workaround:

Associate a floating IP address to an LB port using Horizon or the OpenStack CLI by running the following command:

```
neutron floatingip-associate <floating_ip_id> <lb_port_id>
```

Seealso

Terraform issue

Troubleshoot OpenContrail 3.2

This section includes procedures to troubleshoot the OpenContrail 3.2 services.

Perform initial troubleshooting

This section describes basic troubleshooting steps for the OpenContrail-related services.

To perform initial troubleshooting:

1. Verify the NTP peers on every node of your MCP cluster:

   ```
   ntpq -p
   ```

   Example of system response:

   ```
        remote           refid      st t when poll reach   delay   offset  jitter
   ==============================================================================
   +tik.cesnet.cz   195.113.144.238  2 u  728 1024  377    4.645   -0.199   0.545
   *netopyr.hanacke .GPS.            1 u 1604 1024  276   14.931   -0.021   0.373
   ```

   If at least one of peers has * before its name, time is synchronized. Otherwise, inspect the /etc/ntp.conf file .

   Example of an ntp.conf file

   ```
   # Associate to cloud NTP pool servers
   server ntp.cesnet.cz iburst
   server pool.ntp.org

   # Only allow read-only access from localhost
   restrict default noquery nopeer
   restrict 127.0.0.1
   restrict ::1

   # Location of drift file
   driftfile /var/lib/ntp/ntp.drift
   logfile /var/log/ntp.log
   ```

2. Verify the disk space, Inode, RAM, and CPU usage on every OpenContrail node. The total amount of used resources in the output must be maximum 90%.

   - To verify the disk space:

     ```
     df -h
     ```

     Example of system response:

     ```
     Filesystem      Size  Used Avail Use% Mounted on
     udev            3.9G   12K  3.9G   1% /dev
     tmpfs           799M  380K  798M   1% /run
     /dev/vda1        48G  5.7G   41G  13% /
     none            4.0K     0  4.0K   0% /sys/fs/cgroup
     ```

```
none            5.0M     0  5.0M   0% /run/lock
none            3.9G   12K  3.9G   1% /run/shm
none            100M     0  100M   0% /run/user
```

- To verify the Inode usage:

  ```
  df -i
  ```

  Example of system response:

  ```
  Filesystem       Inodes   IUsed    IFree IUse% Mounted on
  udev            2032563     533  2032030    1% /dev
  tmpfs           2037690     781  2036909    1% /run
  /dev/sda1       6250496 1396006  4854490   23% /
  tmpfs           2037690     304  2037386    1% /dev/shm
  tmpfs           2037690       6  2037684    1% /run/lock
  tmpfs           2037690      18  2037672    1% /sys/fs/cgroup
  /dev/sda6      53821440  731583 53089857    2% /home
  cgmfs           2037690      14  2037676    1% /run/cgmanager/fs
  tmpfs           2037690      44  2037646    1% /run/user/1000
  ```

- To verify RAM usage:

  ```
  free -h
  ```

  Example of system response:

  ```
               total       used       free     shared    buffers     cached
  Mem:          7.8G       7.3G       501M       416K       239M       2.6G
  -/+ buffers/cache:       4.5G       3.3G
  Swap:           0B         0B         0B
  ```

- To verify CPU usage:

  ```
  cat /proc/stat | grep cpu | awk \
  '{unit=100/($1+$2+$3+$4+$5+$6+$7+$8+$9+$10); print $1 "\tidle: "  $5*unit  "%"}'
  ```

  Example of system response:

  ```
  cpu     idle: 94.1113%
  cpu0    idle: 94.3852%
  cpu1    idle: 92.851%
  cpu2    idle: 94.0428%
  cpu3    idle: 94.1673%
  ```

```
cpu4    idle: 94.2658%
cpu5    idle: 94.3526%
cpu6    idle: 94.4082%
cpu7    idle: 94.4092%
```

3. Verify MTU and the status of interfaces on all OpenContrail nodes:

   ```
   ip link
   ```

   Example of system response:

   ```
   1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1
       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
       link/ether ac:de:48:b0:2d:3e brd ff:ff:ff:ff:ff:ff
   3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
       link/ether ac:de:48:a8:7a:09 brd ff:ff:ff:ff:ff:ff
   ```

4. Verify whether the current number of files opened by Linux kernel is not over-limited:

   ```
   cat /proc/sys/fs/file-nr
   ```

   Example of system response:

   ```
   17736    0    1609849
   ```

OpenContrail services are down

If any OpenContrail service fails, start with a basic troubleshooting as described below. If it does not help, proceed with troubleshooting a specific service as required.

To preform basic service troubleshooting:

1. Log in to any OpenContrail controller node.
2. Verify the status of the service in question using the service <service_name> status command.
3. Inspect the logs of the service in question in /var/log/contrail/<service-name>.log.
4. Restart the service using the service <service_name> restart command.
5. Verify whether the service is not flapping. For example, see: Verify the OpenContrail svc-monitor.

Verify contrail-vrouter-agent

While troubleshooting OpenContrail, you may have issues with the contrail-vrouter-agent configuration that fails to be created.

To verify the contrail-vrouter-agent configuration:

1. Log in to any OpenStack compute node.
2. Verify whether the global vRouter configuration is created:

   ```
   contrail-status
   ```

   Example of system response:

   ```
   == Contrail vRouter ==
   supervisor-vrouter:          active
   contrail-vrouter-agent       initializing (No Configuration for self)
   contrail-vrouter-nodemgr     active
   ```

   If the status is active, the configuration is created successfully. If the contrail-vrouter-agent status is initializing (No Configuration for self), proceed to the next step.
3. Use either the OpenContrail web UI or SaltStack commands to create the global vRouter configuration:

   - Using the OpenContrail web UI:

     1. Log in to the OpenContrail web UI as admin.
     2. Go to Configure > Infrastructure > Global Config.
     3. On the right side above the table, click Edit. If the global configuration is missing, the window with default values opens.
     4. Click Save.
   - Using SaltStack:

     1. Log in to the Salt Master node.
     2. Depending on your deployment, apply one of the following states:

        - For Kubernetes:

          ```
          salt 'ctl01*' state.sls opencontrail.client
          ```
        - For OpenStack:

          ```
          salt 'ntw01*' state.sls opencontrail.client
          ```
     3. Verify if global-vrouter-config was created:

- For Kubernetes:

```
salt 'ctl01*' contrail.global_vrouter_config_list
```

- For OpenStack:

```
salt 'ntw01*' contrail.global_vrouter_config_list
```

> Note
>
> If the output is empty, add system.opencontrail.client.resource.global_vrouter_config as the last system class in /srv/salt/reclass/classes/cluster/k8s-ha-contrail-40/opencontrail/control.yml and reapply the opencontrail.client state for Kubernetes or OpenStack as described in the step 2.

4. Verify the contrail-vrouter-agent status:

```
salt 'cmp*' cmd.run "contrail-status"
```

5. If global-vrouter-config is still missing, perform the following manual steps:

   1. Log in to any OpenContrail controller node, for example, ntw01.
   2. Run the following command:

   ```
   salt-call contrail.global_vrouter_config_create name="global-vrouter-config" \
   parent_type="global-system-config" encap_priority="MPLSoUDP,MPLSoGRE" \
   vxlan_vn_id_mode="automatic" \
   fq_names=['default-global-system-config','default-global-vrouter-config']
   ```

4. Verify that the global vRouter configuration is created successfully and is in the active status using the contrail-status command.

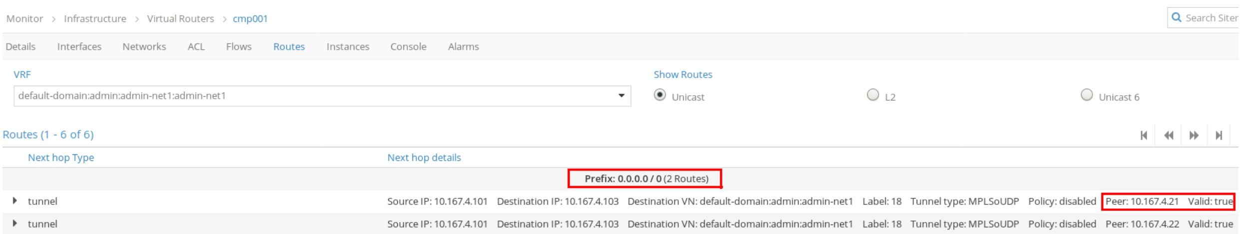
Verify the default route in VRF

Usually, if an instance can ping the default gateway, for example, vRouter, but does not have access to the Internet or cannot ping an IP address outside the cloud, you should verify the default route in virtual routing and forwarding (VRF).

To verify the default route in VRF:

1. Log in to the OpenContrail web UI.
2. Go to Monitor > Infrastructure > Virtual Routers > cmp00x > Routes, where cmp00x is the name of the compute node in question.
3. From the VRF drop-down list, select the VRF of the virtual network in question.

   The following example displays the default route 0.0.0.0/0 with two valid peer routes:

4. Log in to any Mirantis OpenContrail analytics node, for example, nal01.
5. Verify the status of the OpenContrail analytics services:

   ```
   contrail-status
   ```

   Example of system response:

   ```
   == Contrail Analytics ==
   supervisor-analytics:        active
   contrail-alarm-gen           active
   contrail-analytics-api       active
   contrail-analytics-nodemgr   active
   contrail-collector           active
   contrail-query-engine        active
   contrail-snmp-collector      active
   contrail-topology            active

   == Contrail Supervisor Database ==
   supervisor-database:         active
   contrail-database            active
   contrail-database-nodemgr    active
   kafka                        active
   ```

6. If some service is not in the active status, fix the issue and restart the service using the service <service_name> restart.

7. Verify the contrail-svc-monitor status:

    1. Log in to the Mirantis OpenContrail controller ntw node.
    2. Verify that the contrail-svc-monitor state is active using the contrail-status command.

       Example of system response:

       ```
       == Contrail Config ==
       supervisor-config:           active
       contrail-api:0               active
       contrail-config-nodemgr      active
       contrail-device-manager      active
       contrail-discovery:0         active
       contrail-schema              initializing
       contrail-svc-monitor         active
       ifmap                        active
       ```

    3. If the contrail-svc-monitor state is inactive or initializing with an error message:

        1. Log in to the Mirantis OpenContrail controller node where contrail-svc-monitor is in the active state. For example, to ntw01.
        2. Restart the service using the following command. On the remaining ntw nodes, the contrail-svc-monitor state must be backup.

           ```
           service contrail-svc-monitor restart
           ```

        3. If the state is still inactive or initializing with an error message, recreate the gateway:

            1. Log in to the Horizon web UI.
            2. Go to Project > Network > Routers.
            3. On the right side of the router, click Clear Gateway.
            4. On the right side of the router, click Set Gateway.
            5. In the External Network field, specify the network to which the router will connect.
            6. Click Set Gateway.

8. Verify the SNAT and other services instances as described in Verify SNAT and other services instances.

Verify the OpenContrail svc-monitor

The contrail-svc-monitor service listens to the configuration changes of the service templates and service instances as well as spawns and monitors virtual machines for the firewall, analyzer services, and so on. In the multi-node deployments, it works in the active/backup mode.

This section describes how to verify if the svc-monitor is flapping.

To verify that svc-monitor is not flapping:

1. Log in to the Salt Master node.
2. Run the following command every 20 seconds many times, until the contrail-svc-monitor status changes to active on one ntw node and to backup on other ntw nodes:

   ```
   salt 'ntw0*' cmd.run 'contrail-status -d | grep monitor'
   ```

   Example of system response:

   ```
   ntw01.mcp11-opencontrail.local:
   contrail-svc-monitor         backup     pid 28442, uptime 7 days, 5:45:53
   ntw03.mcp11-opencontrail.local:
   contrail-svc-monitor         backup     pid 31439, uptime 7 days, 5:46:30
   ntw02.mcp11-opencontrail.local:
   contrail-svc-monitor         active     pid 7069, uptime 7 days, 5:46:35
   ```

If contrail-svc-monitor service with active state during tests and uptime of all processes is not restarted, then the contrail-svc-monitor is not flapping. Otherwise, proceed to Verify the default route in VRF.

©2025, Mirantis Inc.

Verify route target references

Updating OpenContrail from version 3.1.1 to version 3.2.3 can cause inconsistencies in the OpenContrail configuration database. These inconsistencies may cause connectivity issues from or to a VM, between VMs, and so on. Verifying route target references may help to resolve such issues.

To verify route target references:

1. Log in to any Mirantis OpenContrail controller ntw node.
2. Start contrail-api-cli. For details, see Use the OpenContrail API client.
3. Run the following command using the OpenContrail controller node VIP. For example, 10.167.4.20:

   ```
   contrail-api-cli --host 10.167.4.20 --port 9100 shell
   ```

4. Verify whether route-target exists in more than one project:

   ```
   10.167.4.21:/> cat route-target/* | jq  -c 'if(.routing_instance_back_refs[0]? ) \
   then if ([.routing_instance_back_refs[].to[1]] | unique | length)>1 \
   then [.display_name,[.routing_instance_back_refs[].to[1]]] | \
   unique else empty end else empty end'
   ```

   Example of system response:

   ```
   ["target:64512:8000001",["TestingTenant","admin"]]
   ["target:64512:8000002",["admin","demo-jdc"]]
   ```

   In the output, the list of suspected route-targets is displayed with names of referenced projects.

5. Identify the UUID of the suspected route-target using its name. For example:

   ```
   10.167.4.21:/> ls route-target -l | grep target:64512:8000002
   ```

   Example of system response:

   ```
   route-target/8ed6241c-a1e7-4abf-85c0-3afc70a0b3f0  target:64512:8000002
   ```

6. Verify the status of route-target using the UUID of the suspected route-target. For example:

   ```
   10.167.4.21:/> cat route-target/8ed6241c-a1e7-4abf-85c0-3afc70a0b3f0
   ```

   Example of system response:

```
{
"display_name": "target:64512:8000002",
"fq_name": [
  "target:64512:8000002"
],
...
},
"routing_instance_back_refs": [
  {
    "attr": {
      "import_export": null
    },
    "href": "http://10.167.4.21:9100/routing-instance/cb6a97c8-aca9-4dd1-af00-84368c46d784",
    "to": [
      "default-domain",
      "admin",
      "admin-net2",
      "admin-net2"
    ],
    "uuid": "cb6a97c8-aca9-4dd1-af00-84368c46d784"
  },
  {
    "attr": {
      "import_export": null
    },
    "href": "http://10.167.4.21:9100/routing-instance/4db24ca5-7f10-42b8-b7dc-7c2d12a9d6a3",
    "to": [
      "TestDomain",
      "demo-jdc",
      "jdc-net03",
      "jdc-net03"
    ],
    "uuid": "4db24ca5-7f10-42b8-b7dc-7c2d12a9d6a3"
  }
],
"uuid": "8ed6241c-a1e7-4abf-85c0-3afc70a0b3f0"
}
```

In the output above, the routing_instance_back_refs section contains the route-target with to back references to different routing instances in two different projects. If you know what back reference is incorrect, delete it as described below.

To delete a back reference:

Select from the following options:

- Edit the corresponding JSON records directly.
- Use the ln command:

  1. Exit from contrail-api-cli.

2. Start contrail-api-cli with the --schema-version parameter. For more information, see Use the OpenContrail API client.
3. Remove the back reference of route-target. For example:

```
10.167.4.21:/> ln -r route-target/8ed6241c-a1e7-4abf-85c0-3afc70a0b3f0 \
 routing-instance/cb6a97c8-aca9-4dd1-af00-84368c46d784
```

4. Verify route-target again using the procedure above.

Verify ifmap-server

OpenContrail uses the standard Interface for Metadata Access Point (IF-MAP) mechanism for the configuration data distribution among OpenContrail configuration and control nodes.

To verify ifmap-server:

1. Log in to any OpenContrail controller node.
2. Verify that the virtual machine interface (VMI) is present in ifmap.
    1. Select from the following options:
        - Run the following command:

          ```
          ifmap-view visual visual 2>&1  | grep "contrail:virtual-machine-interface"
          ```

          Example of system response:

          ```
          'contrail:virtual-machine-interface:default-domain:TestingTenant:071b9c54-25ed-41cf-ab89-4f60b60e0e66',
          'contrail:virtual-machine-interface:default-domain:TestingTenant:7c05158a-5c4b-458c-bb88-e6de2146856e',
          'contrail:virtual-machine-interface:default-domain:TestingTenant:3187786b-6f11-4c3b-9d0d-3ce9d02de4cf',
          'contrail:virtual-machine-interface:default-domain:TestingTenant:3187786b-6f11-4c3b-9d0d-3ce9d02de4cf',
          ```
        - Verify the virtual machine interfaces directly by ID. For example:

          ```
          ifmap-view visual visual 2>&1  | grep 071b9c54-25ed-41cf-ab89-4f60b60e0e66
          ```

          Example of system response:

          ```
          'contrail:virtual-machine-interface:default-domain:TestingTenant:071b9c54-25ed-41cf-ab89-4f60b60e0e66'
          ```
    2. If the record of the virtual machine interface in question is missing on the OpenContrail controller node, restart the ifmap-server and contrail-config-nodemgr services on this node:

       ```
       service ifmap-server restart
       service supervisor-config restart
       ```
    3. Verify the number of result items:

       ```
       ifmap-view 10.10.10.201 visual visual  2>&1 | grep <virtual_machine>
       ```

       Example of system response:

       ```
       INFO: Number of result items for search = 72
       INFO: Properties on identifier = ['{http://www.contrailsystems.com/vnc_cfg.xsd}id-perms']
       INFO: Links from identifier = ['contrail:domain:default-domain',
       ```

The number of result items should be the same on all OpenContrail controller nodes.

4. If the nodes have a different number of result items, restart the ifmap-server and contrail-config-nodemgr services on these nodes:

```
service ifmap-server restart
service supervisor-config restart
```

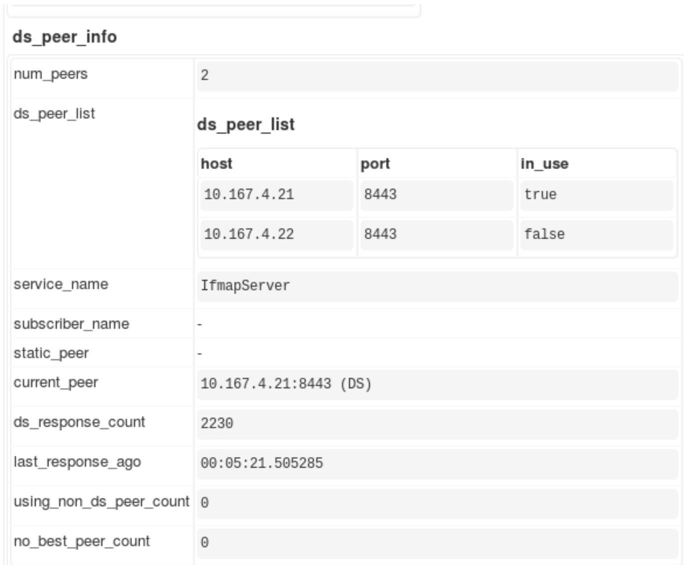
3. Verify the authentication parameters defined in /etc/ifmap-server/basicauthusers.properties. Each IF-MAP client requires a unique user name. All OpenContrail controller nodes must have unique IF-MAP client IDs.

   Example:

```
test:test
test2:test2
test3:test3
api-server:api-server
schema-transformer:schema-transformer
svc-monitor:svc-monitor
control-user:control-user-passwd
control-node-1:control-node-1
control-node-2:control-node-2
control-node-3:control-node-3
dhcp:dhcp
visual:visual
sensor:sensor
```

4. Verify the ifmap-server logs in the following directories:
   - /var/log/contrail/ifmap-server.log
   - /var/log/contrail/ifmap-stdout.log
5. Verify the introspect of the IF-MAP server over HTTP on the port 8083. Use the following introspect address format: http://<ntw0x_ip_address>:8083/ifmap_server_show.xml.
   1. In the IFMapPeerServerInfoReq section, click Send or open the following address in a browser: http://<ntw0x_ip_address>:8083/Snh_IFMapPeerServerInfoReq.

      The following screen shows an example of the IF-MAP server information:



ds_peer_info

num_peers	2
ds_peer_list	**ds_peer_list** (see below)
service_name	IfmapServer
subscriber_name	-
static_peer	-
current_peer	10.167.4.21:8443 (DS)
ds_response_count	2230
last_response_ago	00:05:21.505285
using_non_ds_peer_count	0
no_best_peer_count	0

ds_peer_list

host	port	in_use
10.167.4.21	8443	true
10.167.4.22	8443	false

2. In the ds_peer_info and ds_peer_list sections, verify the information about all IF-MAP peers. In the example above, two OpenContrail controller nodes are displayed. The value of one peer node must be true in the In_use column.

6. Repeat the steps above on the remaining OpenContrail controller nodes.

Troubleshoot the vMX router

MCP uses OpenContrail with vMX routers in its cloud deployments as they provide a rich set of features particularly beneficial for NFV use cases and which allow easy network scale-out.

> Note
>
> - AS (Autonomous System) number is a 2/4 byte identifier for a network segment/organization
> - OpenContrail supports only 2-byte AS numbers (1-65534)
> - Typically, the private AS numbers are being used (64512-65534)
> - The AS number must be the same on the MX and contrail controllers
> - MX uplink peers might be in different ASNes

> Warning
>
> For this section vSRX has been used instead of vMX, but the process is same for both of them.

To troubleshoot the vMX router:

1. Log in to a Mirantis OpenContrail controller ntw node.
2. Verify the BGP Routers configuration using the Introspect section in web UI. Web UI is accessible directly or through HAProxy with the port 9100.

   ```
   curl http://control01:8082/bgp-routers
   ```

   The command above returns a list of all routers defined in the OpenContrail cluster.

   Example of system response:

   ```
   {
    "bgp-routers": [
      {
       "uuid": "443af522-2463-4960-a2b3-77b6b6a46fef",
       "fq_name": [
         "default-domain",
         "default-project",
         "ip-fabric",
         "__default__",
         "ntw03"
       ],
   ```

```
      "href": "http://10.167.4.20:8082/bgp-router/443af522-2463-4960-a2b3-77b6b6a46fef"
    },
    ...
    {
      "uuid": "8af298c2-2a12-452b-86ce-54bff3ce2d9d",
      "fq_name": [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "vsrx1"
      ],
      "href": "http://10.167.4.20:8082/bgp-router/8af298c2-2a12-452b-86ce-54bff3ce2d9d"
    }
  ]
}
```


3. Display the detailed information about the vRouter using the URL from the command above:

```
curl http://10.167.4.20:8082/bgp-router/443af522-2463-4960-a2b3-77b6b6a46fef
```

Example of system response:


```
{
  "bgp-router": {
    "name": "vsrx1",
    ...
    "fq_name": [
      "default-domain",
      "default-project",
      "ip-fabric",
      "__default__",
      "vsrx1"
    ],
    "bgp_router_refs": [
      {
        "uuid": "fb39d3e8-6be0-4e69-b13d-bd69c1685c6c",
        "attr": {
          "session": ....
        },
        "href": "http://10.167.4.22:9100/bgp-router/fb39d3e8-6be0-4e69-b13d-bd69c1685c6c",
        "to": [
          "default-domain",
          "default-project",
          "ip-fabric",
          "__default__",
```

```
        "ntw03"
      ]
    },
    {
      "uuid": "426affc9-b05c-47d8-b0ba-ffe72e59d984",
      "attr": {
        "session": .....
      },
      "href": "http://10.167.4.22:9100/bgp-router/426affc9-b05c-47d8-b0ba-ffe72e59d984",
      "to": [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "ntw02"
      ]
    },
    {
      "uuid": "50f1a77e-9807-4024-b889-f771f2b97835",
      "attr": {
        "session": .....
      },
      "href": "http://10.167.4.22:9100/bgp-router/50f1a77e-9807-4024-b889-f771f2b97835",
      "to": [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "ntw01"
      ]
    }
  ],
  "display_name": "vsrx1",
  "uuid": "2097b2c0-65ac-4c2f-ab0b-aaef2bf9e95a",
  "parent_uuid": "8356722f-02d9-4a57-baaf-1f5013e263f5",
  "parent_href": "http://10.167.4.22:9100/routing-instance/8356722f-02d9-4a57-baaf-1f5013e263f5",
  "parent_type": "routing-instance",
  "bgp_router_parameters": {
    "address_families": {
      "family": [
        "route-target",
        "inet-vpn",
        "e-vpn",
        "inet6-vpn"
      ]
    },
    "autonomous_system": 64512,
```

```
    "hold_time": 0,
    "identifier": "10.167.4.100",
    "router_type": "router",
    "source_port": null,
    "gateway_address": null,
    "vendor": "mx",
    "admin_down": false,
    "ipv6_gateway_address": null,
    "port": 179,
    "local_autonomous_system": null,
    "auth_data": null,
    "address": "10.167.4.100"
   },
   "perms2": {
    "share": [],
    "global_access": 0,
    "owner_access": 7,
    "owner": "cloud-admin"
   },
   "href": "http://10.167.4.22:9100/bgp-router/2097b2c0-65ac-4c2f-ab0b-aaef2bf9e95a"
  }
}
```


In the output above, you can verify such important parameters as autonomous_system, vendor, and others.

4. Log in to the vMX/vSRX router.
5. Verify peer BGR routers and the AS number:

```
root@vsrx1% cli
root@vsrx1> show bgp summary
```

Example of system response:

```
Groups: 1 Peers: 3 Down peers: 0
Unconfigured peers: 3
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l3vpn.0           54         27          0          0          0          0
Peer               AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped...
10.167.4.21         64512      66176      66588       0       0     3w1d23h Establ
  bgp.l3vpn.0: 0/0/0/0
10.167.4.22         64512     117437     100892       0       0     4w6d20h Establ
  bgp.l3vpn.0: 27/27/27/0
  public.inet.0: 5/5/5/0
10.167.4.23         64512      85912      69311       0       0     3w2d21h Establ
  bgp.l3vpn.0: 0/27/27/0
  public.inet.0: 0/5/5/0
```

6. View the current configuration:

```
root@vsrx1> show configuration routing-options
```

Example of system response:

```
route-distinguisher-id 10.109.3.250;
autonomous-system 64512;
dynamic-tunnels {
    dynamic_overlay_tunnels {
        source-address 10.167.4.100;
        gre;
        destination-networks {
            10.109.3.0/24;
            172.16.10.0/24;
            10.167.4.0/24;
        }
    }
}
```

> Note
>
> The command above returns current configuration. When you want to view the latest changes, use compare rollback:
>
> ```
> root@vsrx1> show configuration | compare rollback 1
>
> -   source-address 10.167.4.20;
> +    source-address 10.167.4.100;
> ```
>
> The number after rollback signifies the number of commit to which to compare this configuration.

7. If you have a BGP peer down error with incorrect family:

    1. Verify the BGP peer UVE:

    ```
    curl http://nal01:9081/analytics/uves/bgp-peers
    ```

    User Visible Entities are OpenContrail resources, such as virtual network, virtual machines, vrouter, routing-instances, and so on. UVE APIs are used to query these resources.

8. Search for the vMX/vSRX BGP peer by name in the list.

   In the sample output, families is the family advertised by the peer and configured_families is what is provisioned. In the sample output, the families configured on the peer have a mismatch, thus the peer does not move to an established state. You can verify it in the peer UVE.

9. Fix the families mismatch in the sample by updating the configuration on the MX Series router, using Junos CLI:

   ```
   set protocols bgp group contrail-control-nodes family inet-vpn unicast
   ```

10. After committing the CLI configuration, the peer comes up. Verify it with UVE.

11. Verify the peer status on the MX router using Junos CLI:

    ```
    run show bgp neighbor 10.167.4.21
    ```

12. Check the router in MX/vSRX:

    - Use Junos CLI show commands from the router to check the route.

    ```
    root@vsrx1> run show route table public.inet.0

    public.inet.0: 8 destinations, 13 routes (8 active, 0 holddown, 0 hidden)
    + = Active Route, - = Last Active, * = Both

    0.0.0.0/0          *[Static/5] 4w6d 22:49:22
                        > to 172.17.32.193 via ge-0/0/0.0
    172.17.32.192/26   *[Direct/0] 4w6d 22:49:22
                        > via ge-0/0/0.0
    172.17.32.240/32   *[Local/0] 4w6d 22:49:22
                          Local via ge-0/0/0.0
    <floating_ip0>/32  *[BGP/170] 4w3d 00:41:07, MED 100, localpref 200, from 10.167.4.22
                          AS path: ?
                        > via gr-0/0/0.32769, Push 40
                        [BGP/170] 3w3d 00:30:48, MED 100, localpref 200, from 10.167.4.23
                          AS path: ?
                        > via gr-0/0/0.32769, Push 40
    <floating_ip1>/32  *[BGP/170] 3w3d 00:28:16, MED 100, localpref 200, from 10.167.4.22
                          AS path: ?
                        > via gr-0/0/0.32770, Push 19
                        [BGP/170] 3w3d 00:30:48, MED 100, localpref 200, from 10.167.4.23
                          AS path: ?
                        > via gr-0/0/0.32770, Push 19
    <floating_ip2>/32  *[BGP/170] 4w5d 23:22:58, MED 100, localpref 200, from 10.167.4.22
                          AS path: ?
                        > via gr-0/0/0.32769, Push 29
                        [BGP/170] 3w3d 00:30:48, MED 100, localpref 200, from 10.167.4.23
    ```

```
                    AS path: ?
                  > via gr-0/0/0.32769, Push 29
<floating_ip2>/32  *[BGP/170] 2d 01:50:04, MED 100, localpref 200, from 10.167.4.22
                    AS path: ?
                  > via gr-0/0/0.32770, Push 37
                  [BGP/170] 2d 01:50:04, MED 100, localpref 200, from 10.167.4.23
                    AS path: ?
                  > via gr-0/0/0.32770, Push 37
<floating_ip4>/32  *[BGP/170] 2d 01:31:11, MED 100, localpref 200, from 10.167.4.22
                    AS path: ?
                  > via gr-0/0/0.32770, Push 39
                  [BGP/170] 2d 01:31:11, MED 100, localpref 200, from 10.167.4.23
                    AS path: ?
                  > via gr-0/0/0.32770, Push 39
```

In the output above, you can find the floating IP address what you want to debug.

13. To view the detailed output, run:

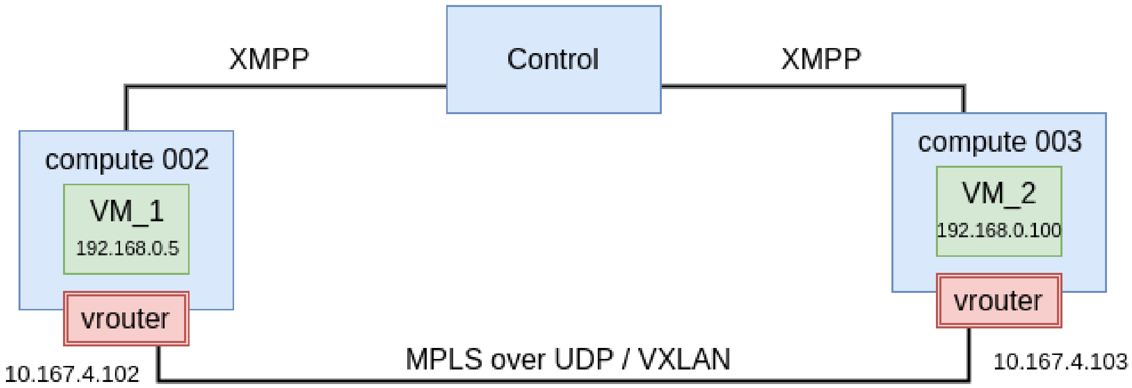
    ```
    root@vsrx1> show route 172.17.35.8/32 detail
    ```

14. Proceed to Troubleshoot a VM forward and reverse flow.

Troubleshoot a VM forward and reverse flow

This section describes how to receive the forward and reverse flow record information of VMs from their respective vRouter compute nodes. This information helps troubleshooting communication issues between virtual machines.

The image below displays an example of communication between VM_1 and VM_2:



The following table shows known information about troubleshooted flows described in this section:

Value name	VM_1	VM_2
prefix local	192.168.0.5/32	192.168.0.100/32
prefix remote	192.168.0.100/32	192.168.0.5/32

To show flows between VMs:

Run the following command on one of the hosts:

- For compute 002:

```
root@cmp002:~# flow -l | grep '192.168.0.5\|192.168.0.100'
```

Example of system response:

```
 Index             Source:Port/Destination:Port                     Proto(V)
-----------------------------------------------------------------------------------
492152<=>1500364       192.168.0.5:792                                    1 (5)
                       192.168.0.100:0
(Gen: 1, K(nh):83, Action:F, Flags:, QOS:-1, S(nh):83,  Stats:487/519142,


1500364<=>492152       192.168.0.100:792                                  1 (5)
                       192.168.0.5:0
(Gen: 1, K(nh):83, Action:F, Flags:, QOS:-1, S(nh):35,  Stats:487/519142,
```

- For compute 003:

```
root@cmp002:~# flow -l | grep '192.168.0.5\|192.168.0.100'
# OR
root@cmp002:~# flow --match "192.168.0.5,192.168.0.100"
```

Example of system response:

```
 Index              Source:Port/Destination:Port                 Proto(V)
---------------------------------------------------------------------------
1274292<=>1451388      192.168.0.5:792                              1 (1)
                       192.168.0.100:0
(Gen: 15, K(nh):98, Action:F, Flags:, QOS:-1, S(nh):63,  Stats:983/1047878,

1451388<=>1274292      192.168.0.100:792                            1 (1)
                       192.168.0.5:0
(Gen: 9, K(nh):98, Action:F, Flags:, QOS:-1, S(nh):98,  Stats:983/1047878,
```

In the output, you can see the ICMP request and response.

Pay attention to the following important parts of the output:

Action
: If you get Action: D(Sg), there might be an issue in security groups.

K(nh)
: It means that next hops are assigned to prefixes. It represents nh-id for the flow.

S(nh)
: This is the next hop used for RPF checks. When a flow is being setup, agent will do route lookup for the source IP and sets up the rpf-nh in the flow. The type of NH used depends on matched route for the source IP.

The following table describes the abbreviations used in the flow output:

Parameter	Abbreviation description
Action	• F=Forward • D=Drop N=NAT(S=SNAT, D=DNAT, Ps=SPAT, Pd=DPAT, L=Link Local Port)
Other	• K(nh)=Key_Nexthop • S(nh)=RPF_Nexthop

Flags	• E=Evicted • Ec=Evict Candidate • N=New Flow • M=Modified Dm=Delete Marked
TCP(r=reverse)	• S=SYN • F=FIN • R=RST • C=HalfClose • E=Established • D=Dead

Now, we have the following information about troubleshooted flows described in this section:

Value name	VM_1	VM_2
prefix local	192.168.0.5/32	192.168.0.100/32
prefix remote	192.168.0.100/32	192.168.0.5/32
nh local	83	98
nh remote	35	63

To get VRF ID:

Run the following commands:

- For compute 002:

```
root@cmp002:~# nh --get 83

Id:83         Type:Encap          Fmly: AF_INET  Rid:0  Ref_cnt:4          Vrf:5
              Flags:Valid, Policy,
              EncapFmly:0806 Oif:8 Len:14
              Encap Data: 02 29 64 b3 e2 f4 00 00 5e 00 01 00 08 00

root@cmp002:~# nh --get 35

Id:35         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:4850       Vrf:0
              Flags:Valid, MPLSoUDP,
              Oif:0 Len:14 Flags Valid, MPLSoUDP,  Data:0c c4 7a 50 27 88 0c c4 7a 17 99 5d 08 00
              Vrf:0  Sip:10.167.4.102  Dip:10.167.4.103
```

- For compute 003:

```
root@cmp003:~# nh --get 98
```

```
Id:98        Type:Encap         Fmly: AF_INET  Rid:0  Ref_cnt:5         Vrf:1
             Flags:Valid, Policy,
             EncapFmly:0806 Oif:13 Len:14
             Encap Data: 02 d1 32 42 b5 87 00 00 5e 00 01 00 08 00

root@cmp003:~# nh --get 63

Id:63        Type:Tunnel        Fmly: AF_INET  Rid:0  Ref_cnt:20        Vrf:0
             Flags:Valid, MPLSoUDP,
             Oif:0 Len:14 Flags Valid, MPLSoUDP,  Data:0c c4 7a 17 99 5d 0c c4 7a 50 27 88 08 00
             Vrf:0  Sip:10.167.4.103  Dip:10.167.4.102
```

Using the outputs above, we add the following information about the troubleshooted flows described in this section:

Value name	VM_1	VM_2
prefix local	192.168.0.5/32	192.168.0.100/32
prefix remote	192.168.0.100/32	192.168.0.5/32
nh local	83	98
nh remote	35	63
VRF local	5	1
VRF remote	0	0

To identify tap interface used by VMs:

1. Log in to an OpenStack controller node.
2. Identify the tap interface used by VMs:

```
neutron port-list | grep 192.168.0.5 | awk '{print $2}' | cut -c 1-11 | awk '{print "tap"$1}'

tap2964b3e2-f4

neutron port-list | grep 192.168.0.100 | awk '{print $2}' | cut -c 1-11 | awk '{print "tap"$1}'

tapd13242b5-87
```

You cam also get this information from the compute cmp node using the 0if value of the nh --get XY command output. For example:

```
nh --get 83
```

Example of system response:

```
Id:83        Type:Encap         Fmly: AF_INET  Rid:0  Ref_cnt:4         Vrf:5
             Flags:Valid, Policy,
```

```
        EncapFmly:0806 Oif:8 Len:14
        Encap Data: 02 29 64 b3 e2 f4 00 00 5e 00 01 00 08 00
```

```
vif --get 8
```

Example of system response:

```
Vrouter Interface Table

Flags: P=Policy, X=Cross Connect, S=Service Chain, Mr=Receive Mirror
       Mt=Transmit Mirror, Tc=Transmit Checksum Offload, L3=Layer 3, L2=Layer 2
       D=DHCP, Vp=Vhost Physical, Pr=Promiscuous, Vnt=Native Vlan Tagged
       Mnp=No MAC Proxy, Dpdk=DPDK PMD Interface, Rfl=Receive Filtering Offload, Mon=Interface is Monitored
       Uuf=Unknown Unicast Flood, Vof=VLAN insert/strip offload, Df=Drop New Flows, Proxy=MAC Requests Proxied Always

vif0/8      OS: tap2964b3e2-f4
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0
            Vrf:5 Flags:PL3L2D QOS:-1 Ref:5
            RX packets:7221  bytes:5897180 errors:0
            TX packets:7229  bytes:6023816 errors:0
            Drops:15
```

You can compare the Vrf parameter of this output with the output for Id:83.

Using the outputs above, we add the following information about the troubleshooted flows described in this section:

Value name	VM_1	VM_2
prefix local	192.168.0.5/32	192.168.0.100/32
prefix remote	192.168.0.100/32	192.168.0.5/32
nh local	83	98
nh remote	35	63
VRF local	5	1
VRF remote	0	0
Tap Interface	tap2964b3e2-f4	tapd13242b5-87

To verify a routing table for VRF (routing instance):

Show or dump the routing table of the required VRF using the rt command

- For compute 002:

```
root@cmp002:~# rt --dump 5 | grep 192.168.0.5/32
192.168.0.5/32         32            P          -        83        2:29:64:b3:e2:f4(240568)

root@cmp002:~# rt --dump 5 | grep 192.168.0.100/32
192.168.0.100/32       32            LP         44       35        2:d1:32:42:b5:87(149684)
```

- For compute 003:

```
root@cmp003:~# rt --dump 1 | grep 192.168.0.5/32
192.168.0.5/32         32          LP        33      63       2:29:64:b3:e2:f4(159964)

root@cmp003:~# rt --dump 1 | grep 192.168.0.100/32
192.168.0.100/32       32           P         -      98       2:d1:32:42:b5:87(227736)
```

If the above procedure does not resolve the issues, proceed with the following steps:

- Verify that a network policy is created.
- Verify that a network policy is assigned to VMs.
- Verify the rules of a security group.
- Verify that virtual networks (VNs) are assigned a correct security group.
- Verify that the routing table using the rt --dump <VRF_ID> if prefixes were exchanged.
- Verify MTU on physical interface used by vRouter.
- Verify the MTU jumbo frames in the underlay network.

Seealso

OpenContrail operations