Mirantis Container Cloud Deployment Guide

version latest

Contents

Copyright notice

Preface

This documentation provides information on how to deploy and operate Mirantis Container Cloud.

- About this documentation set
- Intended audience
- Conventions
- Technology Preview support scope
- Documentation history

About this documentation set

The documentation is intended to help operators understand the core concepts of the product.

The information provided in this documentation set is being constantly improved and amended based on the feedback and kind requests from our software consumers. This documentation set outlines description of the features that are supported within two latest Cloud Container minor releases, with a corresponding note Available since release.

The following table lists the guides included in the documentation set you are reading:

Guides list

Guide	Purpose
Reference Architecture	Learn the fundamentals of Container Cloud reference architecture to plan your deployment.
Deployment Guide	Deploy Container Cloud of a preferred configuration using supported deployment profiles tailored to the demands of specific business cases.
Operations Guide	Operate your Container Cloud deployment.
Release Compatibility Matrix	Deployment compatibility of the Container Cloud components versions for each product release.
Release Notes	Learn about new features and bug fixes in the current Container Cloud version as well as in the Container Cloud minor releases.

For your convenience, we provide all guides from this documentation set in HTML (default), single-page HTML, PDF, and ePUB formats. To use the preferred format of a guide, select the required option from the Formats menu next to the guide title on the Container Cloud documentation home page.

Intended audience

This documentation assumes that the reader is familiar with network and cloud concepts and is intended for the following users:

- Infrastructure Operator
  - Is member of the IT operations team
  - Has working knowledge of Linux, virtualization, Kubernetes API and CLI, and OpenStack to support the application development team
  - Accesses Mirantis Container Cloud and Kubernetes through a local machine or web UI
  - Provides verified artifacts through a central repository to the Tenant DevOps engineers
- Tenant DevOps engineer
  - Is member of the application development team and reports to line-of-business (LOB)
  - Has working knowledge of Linux, virtualization, Kubernetes API and CLI to support application owners
  - Accesses Container Cloud and Kubernetes through a local machine or web UI
  - Consumes artifacts from a central repository approved by the Infrastructure Operator

Conventions

This documentation set uses the following conventions in the HTML format:

Documentation conventions

Convention	Description
boldface font	Inline CLI tools and commands, titles of the procedures and system response examples, table titles.
monospaced font	Files names and paths, Helm charts parameters and their values, names of packages, nodes names and labels, and so on.
italic font	Information that distinguishes some concept or term.
Links	External links and cross-references, footnotes.
Main menu > menu item	GUI elements that include any part of interactive user interface and menu navigation.
Superscript	Some extra, brief information. For example, if a feature is available from a specific release or if a feature is in the Technology Preview development stage.
Note The Note block	Messages of a generic meaning that may be useful to the user.

Caution! The Caution block	Information that prevents a user from mistakes and undesirable consequences when following the procedures.
Warning The Warning block	Messages that include details that can be easily missed, but should not be ignored by the user and are valuable before proceeding.
Seealso The See also block	List of references that may be helpful for understanding of some related tools, concepts, and so on.
Learn more The Learn more block	Used in the Release Notes to wrap a list of internal references to the reference architecture, deployment and operation procedures specific to a newly implemented product feature.

Technology Preview support scope

This documentation set includes description of the Technology Preview features. A Technology Preview feature provide early access to upcoming product innovations, allowing customers to experience the functionality and provide feedback during the development process. Technology Preview features may be privately or publicly available and neither are intended for production use. While Mirantis will provide support for such features through official channels, normal Service Level Agreements do not apply. Customers may be supported by Mirantis Customer Support or Mirantis Field Support.

As Mirantis considers making future iterations of Technology Preview features generally available, we will attempt to resolve any issues that customers experience when using these features.

During the development of a Technology Preview feature, additional components may become available to the public for testing. Because Technology Preview features are being under development, Mirantis cannot guarantee the stability of such features. As a result, if you are using Technology Preview features, you may not be able to seamlessly upgrade to subsequent releases of that feature. Mirantis makes no guarantees that Technology Preview features will be graduated to a generally available product release.

The Mirantis Customer Success Organization may create bug reports on behalf of support cases filed by customers. These bug reports will then be forwarded to the Mirantis Product team for possible inclusion in a future release.

Documentation history

The documentation set refers to Mirantis Container Cloud GA as to the latest released GA version of the product. For details about the Container Cloud GA minor releases dates, refer to Container Cloud releases.

Introduction

Mirantis Container Cloud enables you to create, scale, and upgrade Kubernetes clusters on demand through a declarative API with a centralized identity and access management.

Container Cloud is installed once to deploy the management cluster. The management cluster is deployed through the bootstrap procedure on either the OpenStack, AWS, or bare metal provider. StackLight installs on both types of the clusters, management and managed, to provide metrics for each cluster separately. The baremetal-based deployment includes Ceph as a distributed storage system.

Deploy a baremetal-based management cluster

This section describes how to bootstrap a baremetal-based Mirantis Container Cloud management cluster.

Workflow overview

The bare metal management system enables the Infrastructure Operator to deploy Mirantis Container Cloud on a set of bare metal servers. It also enables Container Cloud to deploy managed clusters on bare metal servers without a pre-provisioned operating system.

The Infrastructure Operator performs the following steps to install Container Cloud in a bare metal environment:

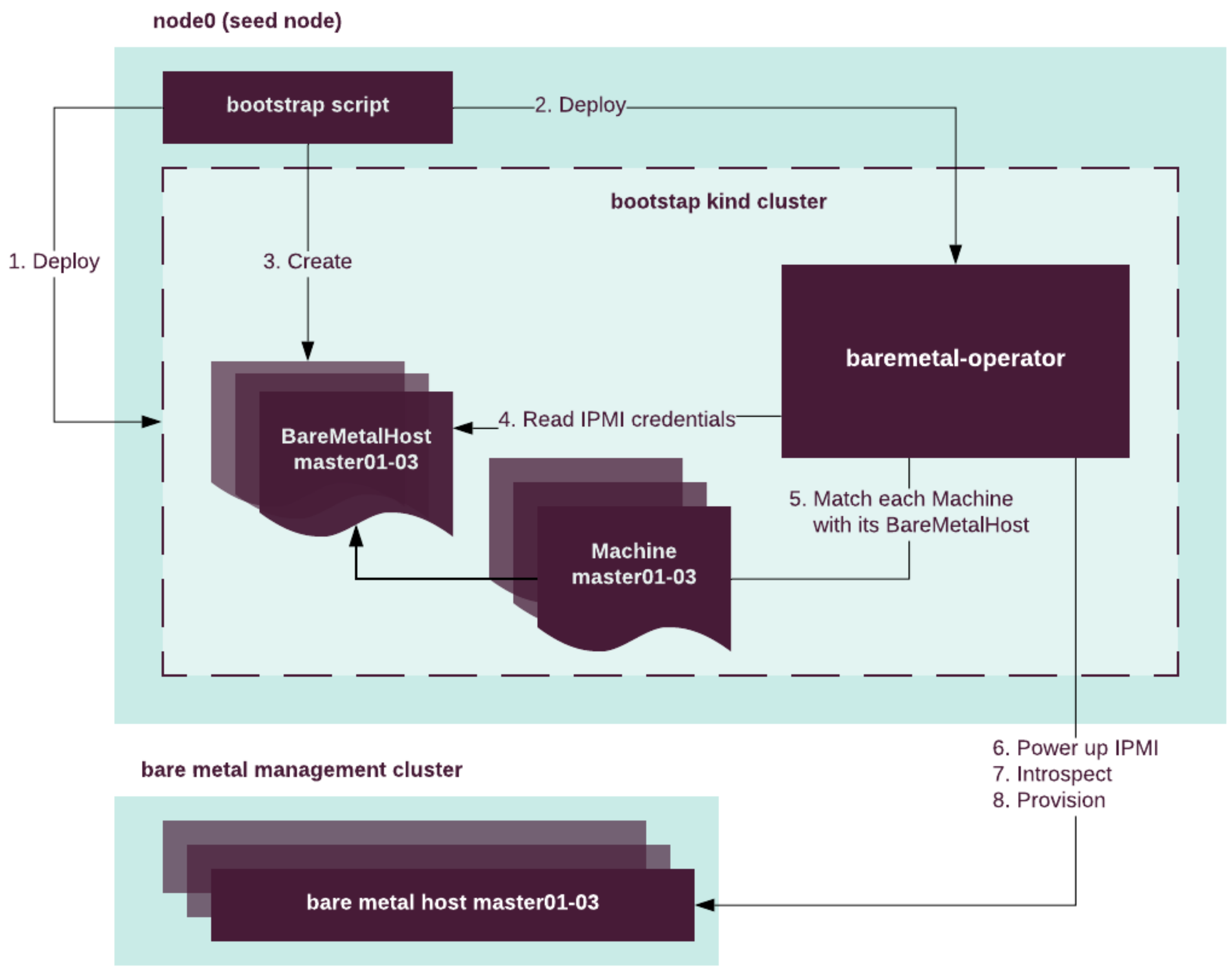
1. Install and connect hardware servers as described in Reference Architecture: Baremetal-based Container Cloud cluster.

   Caution!

   The baremetal-based Container Cloud does not manage the underlay networking fabric but requires specific network configuration to operate.

2. Install Ubuntu 18.04 on one of the bare metal machines to create a seed node and copy the bootstrap tarball to this node.
3. Obtain the Mirantis license file that will be required during the bootstrap.
4. Create the deployment configuration files that include the bare metal hosts metadata.
5. Validate the deployment templates using fast preflight or simulate the main stages of the management cluster deployment using full preflight.
6. Run the bootstrap script for the fully automated installation of the management cluster onto the selected bare metal hosts.

Using the bootstrap script, the Container Cloud bare metal management system prepares the seed node for the management cluster and starts the deployment of Container Cloud itself. The bootstrap script performs all necessary operations to perform the automated management cluster setup. The deployment diagram below illustrates the bootstrap workflow of a baremetal-based management cluster.



Bootstrap a management cluster

This section describes how to prepare and bootstrap a baremetal-based management cluster. The procedure includes:

- A runbook that describes how to create a seed node that is a temporary server used to run the management cluster bootstrap scripts.
- A step-by-step instruction how to prepare metadata for the bootstrap scripts and how to run them.

Prepare the seed node

Before installing Mirantis Container Cloud on a bare metal environment, complete the following preparation steps:

1. Verify that the hardware allocated for the installation meets the minimal requirements described in Reference Architecture: Requirements for a baremetal-based Container Cloud.

2. Install basic Ubuntu 18.04 server using standard installation images of the operating system on the bare metal seed node.
3. Log in to the seed node that is running Ubuntu 18.04.
4. Create a virtual bridge to connect to your PXE network on the seed node. Use the following netplan-based configuration file as an example:

```
# cat /etc/netplan/config.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    ens3:
        dhcp4: false
        dhcp6: false
  bridges:
      br0:
          addresses:
          # Please, adjust for your environment
          - 10.0.0.15/24
          dhcp4: false
          dhcp6: false
          # Please, adjust for your environment
          gateway4: 10.0.0.1
          interfaces:
          # Interface name may be different in your environment
          - ens3
          nameservers:
              addresses:
              # Please, adjust for your environment
              - 8.8.8.8
          parameters:
              forward-delay: 4
              stp: false
```

5. Apply the new network configuration using netplan:

```
sudo netplan apply
```

6. Verify the new network configuration:

```
sudo brctl show
```

Example of system response:

```
bridge name     bridge id               STP enabled     interfaces
br0             8000.fa163e72f146       no              ens3
```

Verify that the interface connected to the PXE network belongs to the previously configured bridge.

7. Install Docker version 18.09:

```
sudo apt install docker.io
```

8. Verify that your logged USER has access to the Docker daemon:

```
sudo usermod -aG docker $USER
```

9. Log out and log in again to the seed node to apply the changes.
10. Verify that Docker is configured correctly and has access to Container Cloud CDN. For example:

```
docker run --rm alpine sh -c "apk add --no-cache curl; \
curl https://binary.mirantis.com"
```

The system output must contain a json file with no error messages. In case of errors, follow the steps provided in Troubleshooting.

Proceed with Verify the seed node.

Verify the seed node

Before you proceed to bootstrapping the management cluster on bare metal, perform the following steps:

1. Verify that the seed node has direct access to the Baseboard Management Controller (BMC) of each baremetal host. All target hardware nodes must be in the power off state.

   For example, using the IPMI tool:

```
ipmitool -I lanplus -H 'IPMI IP' -U 'IPMI Login' -P 'IPMI password' \
chassis power status
```

   Example of system response:

```
Chassis Power is off
```

2. Verify that you configured each bare metal host as follows:
   - Enable the boot NIC support for UEFI load. Usually, at least the built-in network interfaces support it.
   - Enable the UEFI-LAN-OPROM support in BIOS -> Advanced -> PCIPCIe.
   - Enable the IPv4-PXE stack.
   - Set the UEFI-DISK => UEFI-PXE boot order.

- If your PXE network is not configured to use the first network interface, fix the UEFI-PXE boot order to make nodes discovering faster by selecting only one required network interface.
- Power off all bare metal hosts.

> Warning
>
> Only one Ethernet port on a host must be connected to the Common/PXE network at any given time. The physical address (MAC) of this interface must be noted and used to configure the BareMetalHost object describing the host.

Proceed with Prepare metadata and deploy the management cluster.

Prepare metadata and deploy the management cluster

Using the example procedure below, replace the addresses and credentials in the configuration YAML files with the data from your environment. Keep everything else as is, including the file names and YAML structure.

The overall network mapping scheme with all L2 parameters, for example, for a single 10.0.0.0/24 network, is described in the following table. The configuration of each parameter indicated in this table is described in the steps below.

Network mapping overview

Deployment file name	Parameters and values
cluster.yaml	• SET_LB_HOST=10.0.0.90 • SET_METALLB_ADDR_POOL=10.0.0.61-10.0.0.80
ipam-objects.yaml	• SET_IPAM_CIDR=10.0.0.0/24 • SET_PXE_NW_GW=10.0.0.1 • SET_PXE_NW_DNS=8.8.8.8 • SET_IPAM_POOL_RANGE=10.0.0.100-10.0.0.252 • SET_LB_HOST=10.0.0.90 • SET_METALLB_ADDR_POOL=10.0.0.61-10.0.0.80

bootstrap.sh	• KAAS_BM_PXE_IP=10.0.0.20 • KAAS_BM_PXE_MASK=24 • KAAS_BM_PXE_BRIDGE=br0 • KAAS_BM_BM_DHCP_RANGE=10.0.0.30,10.0.0.49 • KEYCLOAK_FLOATING_IP=10.0.0.70 • IAM_FLOATING_IP=10.0.0.71 • PROXY_FLOATING_IP=10.0.0.72

1. Log in to the seed node that you configured as described in Prepare the seed node.
2. Change to your preferred work directory, for example, your home directory:

```
cd $HOME
```

3. Download and run the Container Cloud bootstrap script to this directory:

```
wget https://binary.mirantis.com/releases/get_container_cloud.sh
chmod 0755 get_container_cloud.sh
./get_container_cloud.sh
```

4. Change the directory to the kaas-bootstrap folder created by the get_container_cloud.sh script:

```
cd kaas-bootstrap
```

5. Obtain your license file that will be required during the bootstrap. See step 3 in Getting Started with Mirantis Container Cloud.
6. Save the license file as mirantis.lic under the kaas-bootstrap directory.
7. Create a copy of the current templates directory for future reference.

```
mkdir templates.backup
cp -r templates/*  templates.backup/
```

8. Update the cluster definition template in templates/bm/cluster.yaml.template according to the environment configuration. Use the table below. Manually set all parameters that start with SET_. For example, SET_METALLB_ADDR_POOL.

Cluster template mandatory parameters

Parameter	**Description**	**Example value**

SET_LB_HOST	The IP address of the externally accessible API endpoint of the management cluster. This address must NOT be within the SET_METALLB_ADDR_POOL range but must be from the PXE network. External load balancers are not supported.	10.0.0.90
SET_METALLB_ADDR_POOL	The IP range to be used as external load balancers for the Kubernetes services with the LoadBalancer type. This range must be within the PXE network. The minimum required range is 19 IP addresses.	10.0.0.61-10.0.0.80
SET_IPAM_CIDR	The subnet mask to be used for network management. Must be minimum in the /24 network.	10.0.0.0/24

9. Inspect the default bare metal host profile definition in templates/bm/baremetalhostprofiles.yaml.template. If your hardware configuration differs from the reference, adjust the default profile to match. For details, see Customize the default bare metal host profile.
10. Update the bare metal hosts definition template in templates/bm/baremetalhosts.yaml.template according to the environment configuration. Use the table below. Manually set all parameters that start with SET_.

Bare metal hosts template mandatory parameters

Parameter	Description	Example value
SET_MACHINE_0_IPMI_USERNAME	The IPMI user name in the base64 encoding to access the BMC. [1]	dXNlcg== (base64 encoded user)
SET_MACHINE_0_IPMI_PASSWORD	The IPMI password in the base64 encoding to access the BMC. [1]	cGFzc3dvcmQ= (base64 encoded password)
SET_MACHINE_0_MAC	The MAC address of the first management master node in the PXE network.	ac:1f:6b:02:84:71
SET_MACHINE_0_BMC_ADDRESS	The IP address of the BMC endpoint for the first master node in the management cluster. Must be an address from the OOB network that is accessible through the PXE network default gateway.	192.168.100.11
SET_MACHINE_1_IPMI_USERNAME	The IPMI user name in the base64 encoding to access the BMC. [1]	dXNlcg== (base64 encoded user)
SET_MACHINE_1_IPMI_PASSWORD	The IPMI password in the base64 encoding to access the BMC. [1]	cGFzc3dvcmQ= (base64 encoded password)

SET_MACHINE_1_MAC	The MAC address of the second management master node in the PXE network.	ac:1f:6b:02:84:72
SET_MACHINE_1_BMC_ADDRESS	The IP address of the BMC endpoint for the second master node in the management cluster. Must be an address from the OOB network that is accessible through the PXE network default gateway.	192.168.100.12
SET_MACHINE_2_IPMI_USERNAME	The IPMI user name in the base64 encoding to access the BMC. [1]	dXNlcg== (base64 encoded user)
SET_MACHINE_2_IPMI_PASSWORD	The IPMI password in the base64 encoding to access the BMC. [1]	cGFzc3dvcmQ= (base64 encoded password)
SET_MACHINE_2_MAC	The MAC address of the third management master node in the PXE network.	ac:1f:6b:02:84:73
SET_MACHINE_2_BMC_ADDRESS	The IP address of the BMC endpoint for the third master node in the management cluster. Must be an address from the OOB network that is accessible through the PXE network default gateway.	192.168.100.13

1(1, 2, 3, 4, 5, 6) You can obtain the base64-encoded user name and password using the following command in your Linux console:

```
$ echo -n <username|password> | base64
```

11. Update the IP address pools definition template in templates/bm/ipam-objects.yaml.template according to the environment configuration. Use the table below. Manually set all parameters that start with SET_. For example, SET_IPAM_POOL_RANGE.

IP address pools template mandatory parameters

Parameter	Description	Example value
SET_IPAM_CIDR	The address of PXE network in CIDR notation. Must be minimum in the /24 network.	10.0.0.0/24
SET_PXE_NW_GW	The default gateway in the PXE network. Since this is the only network that Container Cloud will use, this gateway must provide access to: • The Internet to download the Mirantis artifacts • The OOB network of the Container Cloud cluster	10.0.0.1

SET_PXE_NW_DNS	An external (non-Kubernetes) DNS server accessible from the PXE network. This server will be used by the bare metal hosts in all Container Cloud clusters.	8.8.8.8
SET_IPAM_POOL_RANGE	This pool range includes addresses that will be allocated to bare metal hosts in all Container Cloud clusters. The size of this range limits the number of hosts that can be deployed by the instance of Container Cloud.	10.0.0.100-10.0.0.252
SET_LB_HOST [2]	The IP address of the externally accessible API endpoint of the management cluster. This address must NOT be within the SET_METALLB_ADDR_POOL range but must be from the PXE network. External load balancers are not supported.	10.0.0.90
SET_METALLB_ADDR_POOL [2]	The IP range to be used as external load balancers for the Kubernetes services with the LoadBalancer type. This range must be within the PXE network. The minimum required range is 19 IP addresses.	10.0.0.61-10.0.0.80

2(1, 2) Use the same value that you used for this parameter in the cluster.yaml.template file (see above).

12. Optional. Skip this step to use the default password password in the Container Cloud web UI.

    Configure the IAM parameters:

    1. Create hashed passwords for every IAM role: reader, writer, and operator for bare metal deployments:

       ```
       ./bin/hash-generate -i 27500
       ```

       The hash-generate utility requests you to enter a password and outputs the parameters required for the next step. Save the password that you enter in a secure location. This password will be used to access the Container Cloud web UI with a specific IAM role.

       Example of system response:

       ```
       passwordSalt: 6ibPZdUfQK8PsOpSmyVJnA==
       passwordHash: 23W1l65FBdI3NL7LMiUQG9Cu62bWLTqIsOgdW8xNsqw=
       passwordHashAlgorithm: pbkdf2-sha256
       passwordHashIterations: 27500
       ```

       Run the tool several times to generate hashed passwords for every IAM role.

2. Open templates/cluster.yaml.template for editing.
3. In the initUsers section, add the following parameters for each IAM role that you generated in the previous step:
   - passwordSalt - base64-encoded randomly generated sequence of bytes.
   - passwordHash - base64-encoded password hash generated using passwordHashAlgorithm with passwordHashIterations. Supported algorithms include pbkdf2-sha256 and pbkdf-sha512.

13. Optional. Configure external identity provider for IAM.

14. Configure the Ceph cluster:

    1. Configure dedicated networks for Ceph components. Set up the disk configuration according to your hardware node specification in templates/bm/kaascephcluster.yaml.template. Also, verify that the storageDevices section has a valid list of the HDD device names and each device is empty, that is, no file system is present on it. To enable all LCM features of Ceph controller, set manageOsds to true:

```
...
manageOsds: true
...
# This part of KaaSCephCluster should contain valid networks definition
# For production environment, hostNetwork should be always true
network:
  hostNetwork: true
  clusterNet: 10.10.10.0/24
  publicNet: 10.10.11.0/24
...
nodes:
  master-0:
  ...
    # This part of KaaSCephCluster should contain valid device names
    storageDevices:
    - name: sdc
      config:
        deviceClass: hdd
    # Each storageDevices dicts can have several devices
    storageDevices:
    - name: sdc
      config:
        deviceClass: hdd
    # All devices for Ceph also should be described to ``wipe`` in
    # ``baremetalhosts.yaml.template``
    - name: sdd
      config:
       deviceClass: hdd
    # Do not to include first devices here (like vda or sda)
    # because they will be allocated for operating system
...
```

2. Verify that the machine names in the spec:nodes structure are relevant to the metadata:name structure data in machines.yaml.template.

15. Verify that the kaas-bootstrap directory contains the following files:

```
# tree  ~/kaas-bootstrap
  ~/kaas-bootstrap/
  ....
  ├── bootstrap.sh
  ├── kaas
  ├── mirantis.lic
  ├── releases
  ...
  ├── templates
  ....
  │    ├── bm
  │    │    ├── baremetalhostprofiles.yaml.template
  │    │    ├── baremetalhosts.yaml.template
  │    │    ├── cluster.yaml.template
  │    │    ├── ipam-objects.yaml.template
  │    │    ├── kaascephcluster.yaml.template
  │    │    └── machines.yaml.template
  ....
  ├── templates.backup
      ....
```

16. Export all required parameters using the table below.

```
export KAAS_BM_ENABLED="true"
#
export KAAS_BM_PXE_IP="10.0.0.20"
export KAAS_BM_PXE_MASK="24"
export KAAS_BM_PXE_BRIDGE="br0"
#
export KAAS_BM_BM_DHCP_RANGE="10.0.0.30,10.0.0.49"
#
export KEYCLOAK_FLOATING_IP="10.0.0.70"
export IAM_FLOATING_IP="10.0.0.71"
export PROXY_FLOATING_IP="10.0.0.72"
export KAAS_BM_FULL_PREFLIGHT="true"
```

Bare metal prerequisites data

Parameter	Description	Example value

KAAS_BM_PXE_IP	The provisioning IP address. This address will be assigned to the interface of the seed node defined by the KAAS_BM_PXE_BRIDGE parameter (see below). The PXE service of the bootstrap cluster will use this address to network boot the bare metal hosts for the management cluster.	10.0.0.20
KAAS_BM_PXE_MASK	The CIDR prefix for the PXE network. It will be used with all of the addresses below when assigning them to interfaces.	24
KAAS_BM_PXE_BRIDGE	The PXE network bridge name. The name must match the name of the bridge created on the seed node during the Prepare the seed node stage.	br0
KAAS_BM_BM_DHCP_RANGE	The start_ip and end_ip addresses must be within the PXE network. This range will be used by Dnsmasq to provide IP addresses for nodes during provisioning.	10.0.0.30,10.0.0.49
KEYCLOAK_FLOATING_IP	The spec.loadBalancerIP address for the Keycloak service. This address must be within the METALLB_ADDR_POOL range.	10.0.0.70
IAM_FLOATING_IP	The spec.loadBalancerIP address for the IAM service. This address must be within the METALLB_ADDR_POOL range.	10.0.0.71
PROXY_FLOATING_IP	The spec.loadBalancerIP address for the Squid service. This address must be within the METALLB_ADDR_POOL range.	10.0.0.72
KAAS_BM_FULL_PREFLIGHT	The verification preflight check to validate the deployment before bootstrap: • Recommended. If set to true, the full preflight command will run up to 15 minutes to simulate the main stages of the management cluster deployment and to verify that the cluster network and nodes in the BareMetalHost template are configured correctly. • If set to false, the fast preflight command will execute a quick IPMI check to ensure that the template parameters for the Baseboard Management Controller (BMC) access credentials and PXE service are configured correctly.	true

17. Run the verification preflight script to validate the deployment templates configuration:

    ```
    ./bootstrap.sh preflight
    ```

    The command outputs a human-readable report with the verification details:

    - If you run the full preflight command, the report includes information whether the nodes successfully passed the inspection stage and outputs the ICMP results of the networks verification.

      The following example illustrates a positive system response where each node IP and seed addresses have dict with the standard ICMP result:

      ```
      {"10.0.0.20":{"packet_loss":"0","packets_received":"10","packets_transmitted":"10","roundtrip_avg":"2.2381ms","roundtrip_max"     :"12.144668ms","roundtrip_min":"884.641Bµs","roundtrip_stddev":"3.31024ms"},\
      "10.0.0.200":{"packet_loss":"0","packets_received     ":"10","packets_transmitted":"10","roundtrip_avg":"4.066847ms","roundtrip_max":"12.185964ms","roundtrip_min":"1.146483ms","ro     undtrip_stddev":"4.267026ms"},\
      "10.0.0.201":{"packet_loss":"0","packets_received":"10","packets_transmitted":"10","roundtrip_a     vg":"3.658907ms","roundtrip_max":"19.389166ms","roundtrip_min":"1.286625ms","roundtrip_stddev":"5.285539ms"},\
      "10.0.0.202":{"p     acket_loss":"0","packets_received":"10","packets_transmitted":"10","roundtrip_avg":"411.068Bµs","roundtrip_max":"577.381Bµs",     "roundtrip_min":"302.042Bµs","roundtrip_stddev":"92.759Bµs"}}
      ```

      The following example illustrates a negative system response when a node is unreachable:

      ```
      min package test: ping loss 100.0 for 10.0.0.20 higher than acceptable package loss 30.0
      ```

    - If you run the fast preflight command, the report includes the list of verified bare metal nodes and their Chassis Power status. This status is based on the deployment templates configuration used during the verification.

      Note

      During the fast preflight validation, a non-voting warning check is preformed to verify whether the MAC address is LAN1 or LAN2 and ensure that the network scheme type is correct on a node being verified.

      If the system outputs a warning about the MAC address not being from any LANs, verify that you correctly set boot-pxe-mac-address in baremetalhosts.yaml.template according to the hardware documentation of your bare metal node.

    Caution!

    If the report contains information about missing dependencies or incorrect configuration, fix the issues before proceeding to the next step.

18. Run the bootstrap script:

```
./bootstrap.sh all
```

Warning

During the bootstrap process, do not manually restart or power off any of the bare metal hosts.

19. When the bootstrap is complete, collect and save the following management cluster details in a secure location:

    - The kubeconfig file located in the same directory as the bootstrap script. This file contains the admin credentials for the management cluster.
    - The private SSH key openstack_tmp located in ~/.ssh/ for access to the management cluster nodes.

      Note

      The SSH key name openstack_tmp is the same for all cloud providers. This name will be changed in one of the following Container Cloud releases to avoid confusion with a cloud provider name and its related SSH key name.

    - The URL and credentials for the Container Cloud web UI. The system outputs these details when the bootstrap completes.
    - The Keycloak URL that the system outputs when the bootstrap completes. The admin password for Keycloak is located in kaas-bootstrap/passwords.yml along with other IAM passwords.

    Note

    When the bootstrap is complete, the bootstrap cluster resources are freed up.

Seealso

- Operations Guide: Connect to a Container Cloud cluster
- Operations Guide: Remove a management cluster

Customize the default bare metal host profile

This section describes the bare metal host profile settings and instructs how to configure this profile before deploying Mirantis Container Cloud on physical servers.

The bare metal host profile is a Kubernetes custom resource. It allows the Infrastructure Operator to define how the storage devices and the operating system are provisioned and configured.

The bootstrap templates for a bare metal deployment include the template for the default BareMetalHostProfile object in the following file that defines the default bare metal host profile:

```
templates/bm/baremetalhostprofiles.yaml.template
```

The customization procedure of BareMetalHostProfile is almost the same for the management and managed clusters, with the following differences:

- For a management cluster, the customization automatically applies to machines during bootstrap. And for a managed cluster, you apply the changes using kubectl before creating a managed cluster.
- For a management cluster, you edit the default baremetalhostprofiles.yaml.template. And for a managed cluster, you create a new BareMetalHostProfile with the necessary configuration.

For the procedure details, see Operations Guide: Create a custom bare metal host profile. Use this procedure for both types of clusters considering the differences described above.

Deploy an OpenStack-based management cluster

This section describes how to bootstrap an OpenStack-based Mirantis Container Cloud management cluster.

Workflow overview

The Infrastructure Operator performs the following steps to install Mirantis Container Cloud on an OpenStack-based environment:

1. Prepare an OpenStack environment with the requirements described in Reference Architecture: OpenStack-based cluster requirements.
2. Prepare the bootstrap node using Prerequisites.
3. Obtain the Mirantis license file that will be required during the bootstrap.
4. Prepare the OpenStack clouds.yaml file.
5. Create and configure the deployment configuration files that include the cluster and machines metadata.
6. Run the bootstrap script for the fully automated installation of the management cluster.

For more details, see Bootstrap a management cluster.

Prerequisites

Before you start with bootstrapping the OpenStack-based management cluster, complete the following prerequisite steps:

1. Verify that your planned cloud meets the reference hardware bill of material and software requirements as described in Reference Architecture: Requirements for an OpenStack-based Mirantis Container Cloud.
2. Log in to any personal computer or VM running Ubuntu 18.04 that you will be using as the bootstrap node.
3. Install Docker version 18.09:

   ```
   sudo apt install docker.io
   ```

4. Grant your USER access to the Docker daemon:

   ```
   sudo usermod -aG docker $USER
   ```

5. Log off and log in again to the bootstrap node to apply the changes.
6. Verify that Docker is configured correctly and has access to Container Cloud CDN. For example:

```
docker run --rm alpine sh -c "apk add --no-cache curl; \
curl https://binary.mirantis.com"
```

The system output must contain no error records. In case of issues, follow the steps provided in Troubleshooting.

7. Proceed to Bootstrap a management cluster.

Bootstrap a management cluster

After you complete the prerequisite steps described in Prerequisites, proceed with bootstrapping your OpenStack-based Mirantis Container Cloud management cluster.

To bootstrap an OpenStack-based management cluster:

1. Log in to the bootstrap node running Ubuntu 18.04 that is configured as described in Prerequisites.
2. Download and run the Container Cloud bootstrap script:

   ```
   wget https://binary.mirantis.com/releases/get_container_cloud.sh
   chmod 0755 get_container_cloud.sh
   ./get_container_cloud.sh
   ```

3. Change the directory to the kaas-bootstrap folder created by the get_container_cloud.sh script.
4. Obtain your license file that will be required during the bootstrap. See step 3 in Getting Started with Mirantis Container Cloud.
5. Save the license file as mirantis.lic under the kaas-bootstrap directory.
6. Log in to the OpenStack Horizon.
7. In the Project section, select API Access.
8. In the right-side drop-down menu Download OpenStack RC File, select OpenStack clouds.yaml File.
9. Add the downloaded clouds.yaml file to the directory with the bootstrap.sh script.
10. In clouds.yaml, add the password field with your OpenStack password under the clouds/openstack/auth section.

    Example:

    ```
    clouds:
      openstack:
        auth:
          auth_url: https://auth.openstack.example.com:5000/v3
          username: your_username
          password: your_secret_password
          project_id: your_project_id
          user_domain_name: your_ldap_password
        region_name: RegionOne
        interface: public
        identity_api_version: 3
    ```

11. Verify access to the target cloud endpoint from Docker. For example:

    ```
    docker run --rm alpine sh -c "apk add --no-cache curl; \
    curl https://auth.openstack.example.com:5000/v3"
    ```

The system output must contain no error records. In case of issues, follow the steps provided in Troubleshooting.

12. In templates/machines.yaml.template, modify the spec:providerSpec:value sections for set: master and set: node by substituting the flavor and image parameters with the corresponding values of your OpenStack cluster. For example:

    ```
    spec:
      providerSpec:
        value:
          apiVersion: "openstackproviderconfig.k8s.io/v1alpha1"
          kind: "OpenstackMachineProviderSpec"
          flavor: kaas.minimal
          image: bionic-server-cloudimg-amd64-20190612
    ```

    Also, modify other parameters as required.

13. Modify the templates/cluster.yaml.template parameters to fit your deployment. For example, add the corresponding values for cidrBlocks in the spec::clusterNetwork::services section.

    Note

    The passwordSalt and passwordHash values for the IAM roles are automatically re-generated during the IAM configuration described in the next step.

14. Optional. Skip this step to use the default password password in the Container Cloud web UI.

    Configure the IAM parameters:

    1. Create hashed passwords for every IAM role: reader, writer, and operator for bare metal deployments:

       ```
       ./bin/hash-generate -i 27500
       ```

       The hash-generate utility requests you to enter a password and outputs the parameters required for the next step. Save the password that you enter in a secure location. This password will be used to access the Container Cloud web UI with a specific IAM role.

       Example of system response:

       ```
       passwordSalt: 6ibPZdUfQK8PsOpSmyVJnA==
       passwordHash: 23W1l65FBdI3NL7LMiUQG9Cu62bWLTqIsOgdW8xNsqw=
       passwordHashAlgorithm: pbkdf2-sha256
       passwordHashIterations: 27500
       ```

Run the tool several times to generate hashed passwords for every IAM role.

2. Open templates/cluster.yaml.template for editing.
3. In the initUsers section, add the following parameters for each IAM role that you generated in the previous step:

    - passwordSalt - base64-encoded randomly generated sequence of bytes.
    - passwordHash - base64-encoded password hash generated using passwordHashAlgorithm with passwordHashIterations. Supported algorithms include pbkdf2-sha256 and pbkdf-sha512.

15. Optional. Configure external identity provider for IAM.
16. Run the bootstrap script:

```
./bootstrap.sh all
```

17. When the bootstrap is complete, collect and save the following management cluster details in a secure location:

    - The kubeconfig file located in the same directory as the bootstrap script. This file contains the admin credentials for the management cluster.
    - The private SSH key openstack_tmp located in ~/.ssh/ for access to the management cluster nodes.

    > Note
    >
    > The SSH key name openstack_tmp is the same for all cloud providers. This name will be changed in one of the following Container Cloud releases to avoid confusion with a cloud provider name and its related SSH key name.

    - The URL and credentials for the Container Cloud web UI. The system outputs these details when the bootstrap completes.
    - The Keycloak URL that the system outputs when the bootstrap completes. The admin password for Keycloak is located in kaas-bootstrap/passwords.yml along with other IAM passwords.

    > Note
    >
    > When the bootstrap is complete, the bootstrap cluster resources are freed up.

18. In case of deployment issues, collect and inspect the bootstrap and management cluster logs as described in Troubleshooting.

19. Optional. Deploy an additional regional cluster as described in Deploy an additional regional cluster.

Now, you can proceed with operating your management cluster using the Container Cloud web UI and deploying managed clusters as described in Create an OpenStack-based managed cluster.

Seealso

- Operations Guide: Connect to a Container Cloud cluster
- Operations Guide: Remove a management cluster

Deploy an AWS-based management cluster

This section describes how to bootstrap a Mirantis Container Cloud management cluster that is based on the Amazon Web Services (AWS) cloud provider.

Workflow overview

The Infrastructure Operator performs the following steps to install Mirantis Container Cloud on an AWS-based environment:

1. Prepare an AWS environment with the requirements described in Reference Architecture: AWS-based Container Cloud cluster requirements.
2. Prepare the bootstrap node as per Prerequisites.
3. Obtain the Mirantis license file that will be required during the bootstrap.
4. Prepare the AWS environment credentials.
5. Create and configure the deployment configuration files that include the cluster and machines metadata.
6. Run the bootstrap script for the fully automated installation of the management cluster.

For more details, see Bootstrap a management cluster.

Prerequisites

Before you start with bootstrapping the AWS-based management cluster, complete the following prerequisite steps:

1. Inspect the Requirements for an AWS-based Container Cloud cluster to understand the potential impact of the Container Cloud deployment on your AWS cloud usage.
2. Log in to any personal computer or VM running Ubuntu 18.04 that you will be using as the bootstrap node.
3. If you use a newly created VM, run:

   ```
   sudo apt-get update
   ```

4. Install Docker version 18.09:

   ```
   sudo apt install docker.io
   ```

5. Grant your USER access to the Docker daemon:

   ```
   sudo usermod -aG docker $USER
   ```

6. Log out and log in again to the bootstrap node to apply the changes.
7. Verify that Docker is configured and works correctly. For example:

```
docker run --rm alpine sh -c "apk add --no-cache curl; \
curl https://binary.mirantis.com"
```

The system output must contain no error records. In case of issues, follow the steps provided in Troubleshooting.

8. Proceed to Bootstrap a management cluster.

Bootstrap a management cluster

After you complete the prerequisite steps described in Prerequisites, proceed with bootstrapping your AWS-based Mirantis Container Cloud management cluster.

To bootstrap an AWS-based management cluster:

1. Log in to the bootstrap node running Ubuntu 18.04 that is configured as described in Prerequisites.
2. Download and run the Container Cloud bootstrap script:

   ```
   wget https://binary.mirantis.com/releases/get_container_cloud.sh

   chmod 0755 get_container_cloud.sh

   ./get_container_cloud.sh
   ```

3. Change the directory to the kaas-bootstrap folder created by the get_container_cloud.sh script.
4. Obtain your license file that will be required during the bootstrap. See step 3 in Getting Started with Mirantis Container Cloud.
5. Save the license file as mirantis.lic under the kaas-bootstrap directory.
6. Verify access to the target cloud endpoint from Docker. For example:

   ```
   docker run --rm alpine sh -c "apk add --no-cache curl; \
   curl https://ec2.amazonaws.com"
   ```

   The system output must contain no error records. In case of issues, follow the steps provided in Troubleshooting.
7. In templates/aws/machines.yaml.template, modify the spec:providerSpec:value section by substituting the ami:id parameter with the corresponding value for Ubuntu 18.04 from the required AWS region. For example:

   ```
   spec:
     providerSpec:
       value:
         apiVersion: aws.kaas.mirantis.com/v1alpha1
         kind: AWSMachineProviderSpec
         instanceType: c5d.2xlarge
         ami:
           id: ami-033a0960d9d83ead0
   ```

   Also, modify other parameters as required.
8. Optional. In templates/aws/cluster.yaml.template, modify the default AWS instance types and AMIs configuration for further creation of managed clusters:

```
providerSpec:
  value:
    ...
    kaas:
      ...
      regional:
      - provider: aws
        helmReleases:
          - name: aws-credentials-controller
            values:
              config:
                allowedInstanceTypes:
                  minVCPUs: 8
                  # in MiB
                  minMemory: 16384
                  # in GB
                  minStorage: 120
                  supportedArchitectures:
                  - "x86_64"
                  filters:
                  - name: instance-storage-info.disk.type
                    values:
                      - "ssd"
                allowedAMIs:
                -
                  - name: name
                    values:
                    - "ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20200729"
                  - name: owner-id
                    values:
                    - "099720109477"
```

Also, modify other parameters as required.

9. Generate the AWS Access Key ID with Secret Access Key for the admin user and select the AWS default region name. For details, see AWS General Reference: Programmatic access.

10. Export the following parameters by adding the corresponding values for the AWS admin credentials created in the previous step:

```
export KAAS_AWS_ENABLED=true
export AWS_SECRET_ACCESS_KEY=XXXXXXX
export AWS_ACCESS_KEY_ID=XXXXXXX
export AWS_DEFAULT_REGION=us-east-2
```

11. Create the AWS CloudFormation template for IAM policy:

```
./kaas bootstrap aws policy
```

12. Generate the AWS Access Key ID with Secret Access Key for the bootstrapper.cluster-api-provider-aws.kaas.mirantis.com user, that was created in the previous step, and select the AWS default region name.
13. Export the AWS bootstrapper.cluster-api-provider-aws.kaas.mirantis.com user credentials that were created in the previous step:

    ```
    export KAAS_AWS_ENABLED=true
    export AWS_SECRET_ACCESS_KEY=XXXXXXX
    export AWS_ACCESS_KEY_ID=XXXXXXX
    export AWS_DEFAULT_REGION=us-east-2
    ```

14. Optional. Skip this step to use the default password password in the Container Cloud web UI.

    Configure the IAM parameters:

    1. Create hashed passwords for every IAM role: reader, writer, and operator for bare metal deployments:

       ```
       ./bin/hash-generate -i 27500
       ```

       The hash-generate utility requests you to enter a password and outputs the parameters required for the next step. Save the password that you enter in a secure location. This password will be used to access the Container Cloud web UI with a specific IAM role.

       Example of system response:

       ```
       passwordSalt: 6ibPZdUfQK8PsOpSmyVJnA==
       passwordHash: 23W1l65FBdI3NL7LMiUQG9Cu62bWLTqIsOgdW8xNsqw=
       passwordHashAlgorithm: pbkdf2-sha256
       passwordHashIterations: 27500
       ```

       Run the tool several times to generate hashed passwords for every IAM role.
    2. Open templates/cluster.yaml.template for editing.
    3. In the initUsers section, add the following parameters for each IAM role that you generated in the previous step:
       - passwordSalt - base64-encoded randomly generated sequence of bytes.
       - passwordHash - base64-encoded password hash generated using passwordHashAlgorithm with passwordHashIterations. Supported algorithms include pbkdf2-sha256 and pbkdf-sha512.
15. Optional. Configure external identity provider for IAM.
16. Run the bootstrap script:

    ```
    ./bootstrap.sh all
    ```

17. When the bootstrap is complete, collect and save the following management cluster details in a secure location:

    - The kubeconfig file located in the same directory as the bootstrap script. This file contains the admin credentials for the management cluster.
    - The private SSH key openstack_tmp located in ~/.ssh/ for access to the management cluster nodes.

      > Note
      >
      > The SSH key name openstack_tmp is the same for all cloud providers. This name will be changed in one of the following Container Cloud releases to avoid confusion with a cloud provider name and its related SSH key name.

    - The URL and credentials for the Container Cloud web UI. The system outputs these details when the bootstrap completes.
    - The Keycloak URL that the system outputs when the bootstrap completes. The admin password for Keycloak is located in kaas-bootstrap/passwords.yml along with other IAM passwords.

    > Note
    >
    > When the bootstrap is complete, the bootstrap cluster resources are freed up.

18. In case of deployment issues, collect and inspect the bootstrap and management cluster logs as described in Troubleshooting.
19. Optional. Deploy an additional regional cluster of a different provider type as described in Deploy an additional regional cluster.

Now, you can proceed with operating your management cluster using the Container Cloud web UI and deploying managed clusters as described in Create an AWS-based managed cluster.

> Seealso
>
> - Operations Guide: Connect to a Container Cloud cluster
> - Operations Guide: Remove a management cluster

Deploy a VMWare vSphere-based management cluster

Caution!

This feature is available as Technology Preview. Use such configuration for testing and evaluation purposes only. For details about the Mirantis Technology Preview support scope, see the Preface section of this guide.

Caution!

This feature is available starting from the Container Cloud release 2.2.0.

This section describes how to bootstrap a VMWare vSphere-based Mirantis Container Cloud management cluster.

Workflow overview

Perform the following steps to install Mirantis Container Cloud on a VMWare vSphere-based environment:

1. Prepare a VMWare vSphere environment with the requirements described in Reference Architecture: VMWare vSphere-based cluster requirements.
2. Prepare the bootstrap node as described in Prerequisites.
3. Obtain the Mirantis license file to use during the bootstrap.
4. Set up the VMWare accounts for deployment as described in VMWare deployment users.
5. Prepare the OVF template for the management cluster nodes using OVF template requirements.
6. Create and configure the deployment configuration files that include the cluster and machines metadata.
7. Run the bootstrap script for the fully automated installation of the management cluster.

For more details, see Bootstrap a management cluster.

Prerequisites

Before bootstrapping a VMWare vSphere-based management cluster, complete the following prerequisite steps:

1. Verify that your planned cloud meets the reference hardware bill of material and software requirements as described in Reference Architecture: Requirements for a VMWare vSphere-based Container Cloud cluster.
2. Log in to any personal computer or VM running Ubuntu 18.04 that you will be using as the bootstrap node.
3. Install Docker version 18.09:

   ```
   sudo apt install docker.io
   ```

4. Grant your USER access to the Docker daemon:

   ```
   sudo usermod -aG docker $USER
   ```

5. Log off and log in again to the bootstrap node to apply the changes.
6. Verify that Docker is configured correctly and has access to Container Cloud CDN. For example:

   ```
   docker run --rm alpine sh -c "apk add --no-cache curl; \
   curl https://binary.mirantis.com"
   ```

   The system output must contain no error records. In case of issues, follow the steps provided in Troubleshooting.
7. Prepare the VMWare deployment user setup and permissions.
8. Prepare the VMWare OVF template.

Prepare the VMWare deployment user setup and permissions

To deploy Mirantis Container Cloud on a VMWare vSphere-based environment, prepare the following VMWare accounts:

1. Create a read-only virt-who user on the vCenter Server. The virt-who user requires at least read-only access to all objects in the vCenter Data Center.

   The virt-who service on RHEL machines will be provided with the virt-who user credentials in order to properly manage RHEL subscriptions.

   For details on how to create the virt-who user, refer to the official RedHat Customer Portal documentation.
2. Add the following privileges to the cluster-api user:

Note

Container Cloud uses two separate vSphere accounts for the Cluster API related operations (create and delete VMs) and for storage operations (dynamic PVC provision). But also it can be one user that has both privileges sets.

Privilege	Permission
Content Library	• Download files • Read storage • Sync library item
Datastore	• Allocate space • Browse datastore • Update virtual machine files • Update virtual machine metadata
Folder	• Create folder • Rename folder
Global	Cancel task
Host	Local operations: • Create virtual machine • Delete virtual machine • Reconfigure virtual machine
Network	Assign network
Network	Assign virtual machine to resource pool
Scheduled task	• Create tasks • Modify task • Remove task • Run task
Sessions	• Validate session • View and stop sessions
Storage views	View
Tasks	• Create task • Update task

Virtual machine permissions

Privilege	Permission
Change configuration	• Acquire disk lease • Add existing disk • Add new disk • Add or remove device • Advanced configuration • Change CPU count • Change Memory • Change Settings • Change Swapfile placement • Change resource • Configure Host USB device • Configure Raw device • Configure managedBy • Display connection settings • Extend virtual disk • Modify device settings • Query Fault Tolerance compatibility • Query unowned files • Reload from path • Remove disk • Rename • Reset guest information • Set annotation • Toggle disk change tracking • Toggle fork parent • Upgrade virtual machine compatibility

Edit inventory	• Create from existing • Create new • Move • Register • Remove • Unregister
Interaction	• Connect devices • Console interaction • Power off • Power on • Reset • Suspend
Provisioning	• Allow disk access • Allow file access • Allow read-only disk access • Allow virtual machine download • Allow virtual machine files upload • Clone template • Clone virtual machine • Create template from virtual machine • Customize guest • Deploy template • Mark as template • Mark as virtual machine • Modify customization specification • Promote disks • Read customization specifications
Snapshot management	• Create snapshot • Remove snapshot • Rename snapshot • Revert to snapshot
vSphere replication	Monitor replication

3. Add the following privileges to the storage user:

Privilege	Permission
Cns	Searchable
Content Library	View configuration settings
Datastore	• Allocate space • Browse datastore • Configure datastore • Low level file operations • Remove file • Update virtual machine files • Update virtual machine metadata
Folder	• Create folder • Delete folder • Move folder • Rename folder
Host	• Configuration: • Storage partition configuration • Local operations: • Create virtual machine • Delete virtual machine • Reconfigure virtual machine
ImageBuilder	• Depot: • Create • Delete • Profile: • Create • Delete • Export
Host profile	View

Resource	• Assign vApp to resource pool • Assign virtual machine to resource pool • Migrate powered off virtual machine • Migrate powered on virtual machine • Modify resource pool • Query vMotion
Scheduled task	• Create tasks • Modify task • Run task
Sessions	• Validate session • View and stop sessions
Datastore cluster	Configure a datastore cluster
Profile-driven storage	Profile-driven storage view
Storage views	• Configure service • View

Virtual machine permissions

Privilege	**Permission**

Change configuration	• Acquire disk lease • Add existing disk • Add new disk • Add or remove device • Advanced configuration • Change CPU count • Change Memory • Change Settings • Change Swapfile placement • Change resource • Configure Host USB device • Configure Raw device • Configure managedBy • Display connection settings • Extend virtual disk • Modify device settings • Query Fault Tolerance compatibility • Query unowned files • Reload from path • Remove disk • Rename • Reset guest information • Set annotation • Toggle disk change tracking • Toggle fork parent • Upgrade virtual machine compatibility

Provisioning	• Allow disk access • Allow file access • Allow read-only disk access • Allow virtual machine download • Allow virtual machine files upload • Clone template • Clone virtual machine • Create template from virtual machine • Customize guest • Deploy template • Mark as template • Mark as virtual machine • Modify customization specification • Promote disks • Read customization specifications
Snapshot management	• Create snapshot • Remove snapshot • Rename snapshot • Revert to snapshot

Now, proceed to Prepare the VMWare OVF template.

Prepare the VMWare OVF template

To deploy Mirantis Container Cloud on a VMWare vSphere-based environment, the VMWare OVF template for cluster machines must be prepared according to the following requirements:

1. The VMware Tools package is installed.
2. The cloud-init utility is installed and configured with the specific VMwareGuestInfo datasource.
3. The virt-who service is enabled and configured to connect to the VMware vCenter Server to properly apply the RHEL subscriptions on the nodes.

The following procedure describes how to meet the requirements above.

To prepare the VMWare OVF template:

1. Run a virtual machine on the VMware Datacenter from the official RHEL 7.8 server image. Specify the amount of resources that will be used in the Container Cloud setup. A minimal resources configuration must match the requirements for a VMWare vSphere-based Container Cloud cluster.

2. Select minimal setup in the VM installation configuration. Create a user with root or sudo permissions to access the machine.
3. Log in to the VM when it starts.
4. Attach your RHEL license for Virtual Datacenters to the VM:

   ```
   subscription-manager register
   # automatic subscription selection:
   subscription-manager attach --auto
   # or specify pool id:
   subscription-manager attach --pool=<POOL_ID>
   # verify subscription status
   subscription-manager status
   ```

5. Select from the following options:

   - Prepare the operating system automatically:

     1. Download the automation script:

        ```
        curl https://gerrit.mcp.mirantis.com/plugins/gitiles/kubernetes/vmware-guestinfo/+/refs/tags/  v1.1.1/install.sh?format=TEXT | \
        base64 -d > install.sh
        chmod +x install.sh
        ```

     2. Export the VMware vCenter Server credentials of the read-only user. For example:

        ```
        export VC_SERVER='vcenter1.example.com'
        export VC_USER='domain\vmware_read_only_username'
        export VC_PASSWORD='password!23'
        export VC_OWNER='1234567'
        # optional parameter:
        export VC_HYPERVISOR_ID=hostname
        export VC_FILTER_HOSTS="esx1.example.com, esx2.example.com"
        export VCENTER_CONFIG_PATH="/etc/virt-who.d/vcenter.conf"
        ```

     3. Run the installation script:

        ```
        ./install.sh
        ```

   - Prepare the operating system manually:

     1. Install open-vm-tools:

        ```
        yum install open-vm-tools -y
        ```

     2. Install and configure cloud-init:

        1. Download the VMwareGuestInfo datasource files:

           ```
           curl https://gerrit.mcp.mirantis.com/plugins/gitiles/kubernetes/vmware-guestinfo/+/refs/tags/v1.1.1/DataSourceVMwareGuestInfo.py?format=TEXT | \
           base64 -d > DataSourceVMwareGuestInfo.py
           curl https://gerrit.mcp.mirantis.com/plugins/gitiles/kubernetes/vmware-guestinfo/+/refs/tags/v1.1.1/99-DataSourceVMwareGuestInfo.cfg?format=TEXT | \
           base64 -d > 99-DataSourceVMwareGuestInfo.cfg
           ```

        2. Add 99-DataSourceVMwareGuestInfo.cfg to /etc/cloud/cloud.cfg.d/.

3. Depending on the Python version on the VM operating system, add DataSourceVMwareGuestInfo.py to the cloud-init sources folder.
4. Obtain the cloud-init folder on RHEL:

```
yum install cloud-init -y
python -c 'import os; from cloudinit import sources; print(os.path.dirname(sources.__file__));'
```

3. Prepare the virt-who user configuration:

Note

For details about the virt-who user creation, see Prepare the VMWare deployment user setup and permissions.

1. Install virt-who:

```
yum install virt-who -y
cp /etc/virt-who.d/template.conf /etc/virt-who.d/vcenter.conf
```

2. Set up the file content using the following example:

```
[vcenter]
type=esx
server=vcenter1.example.com
username=domain\vmware_read_only_username
encrypted_password=bd257f93d@482B76e6390cc54aec1a4d
owner=1234567
hypervisor_id=hostname
filter_hosts=esx1.example.com, esx2.example.com
```

virt-who configuration parameters

Parameter	Description
[vcenter]	Name of VMware datacenter.
type=esx	Specifies the connection of the defined virt-who user to the VMware vCenter Server.
server	The FQDN of the vCenter Server.
username	The virt-who user name on the vCenter Server with the read-only access.
encrypted_password	The virt-who password encrypted by the virt-who-password utility using the virt-who-password -p <password> command.
owner	The organization that the hypervisors belong to.

hypervisor_id	Specifies how to identify the hypervisors. Use a host name to provide meaningful host names to the Subscription Management. Alternatively, use uuid or hwuuid to avoid duplication in case of hypervisor renaming.
filter_hosts	List of hypervisors that never run RHEL VMs. Such hypervisors do not have to be reported by virt-who.

6. Remove the RHEL subscription from the node.

```
subscription-manager remove --all
subscription-manager unregister
subscription-manager clean
```

7. Shut down the VM.
8. Create an OVF template from the VM.

Now, proceed to Bootstrap a management cluster.

Bootstrap a management cluster

Caution!

This feature is available as Technology Preview. Use such configuration for testing and evaluation purposes only. For details about the Mirantis Technology Preview support scope, see the Preface section of this guide.

Caution!

This feature is available starting from the Container Cloud release 2.2.0.

After you complete the prerequisite steps described in Prerequisites, proceed with bootstrapping your VMWare vSphere-based Mirantis Container Cloud management cluster.

To bootstrap a VMWare vSphere-based management cluster:

1. Log in to the bootstrap node running Ubuntu 18.04 that is configured as described in Prerequisites.
2. Download and run the Container Cloud bootstrap script:

   ```
   wget https://binary.mirantis.com/releases/get_container_cloud.sh
   chmod 0755 get_container_cloud.sh
   ./get_container_cloud.sh
   ```

3. Change the directory to the kaas-bootstrap folder created by the get_container_cloud.sh script.
4. Obtain your license file that will be required during the bootstrap. See step 3 in Getting Started with Mirantis Container Cloud.
5. Save the license file as mirantis.lic under the kaas-bootstrap directory.
6. In templates/vsphere/rhellicenses.yaml.template, set the user name and password of your RedHat Customer Portal account associated with your RHEL license for Virtual Datacenters. Optionally, specify the subscription allocation pools to use for the RHEL subscriptions activation. If you leave the pool field empty, subscription-manager will automatically select the licenses for machines.
7. In templates/vsphere/machines.yaml.template, modify the spec:providerSpec:value section using the following configuration file extract as an example:

   ```
   spec:
     providerSpec:
   ```

```
  value:
    apiVersion: vsphere.cluster.k8s.io/v1alpha1
    kind: VsphereMachineProviderSpec
    sshUserName: SSH_USER_NAME
    rhelLicense: kaas-mgmt-rhel-license
    datacenter: SET_VSPHERE_DATACENTER_PATH
    network:
      devices:
      - networkName: SET_VSPHERE_NETWORK_PATH
        dhcp4: true
        dhcp6: false
    template: SET_VSPHERE_TEMPLATE_PATH
```

Replace the values in caps lock with the corresponding values of your environment. The default SSH user name is cloud-user. Also, modify other parameters as required.

8. Modify the templates/vsphere/cluster.yaml.template parameters to fit your deployment. For example, add the corresponding values for cidrBlocks in the spec::clusterNetwork::services section.

   Note

   The passwordSalt and passwordHash values for the IAM roles are automatically re-generated during the IAM configuration described in the next step.

9. Modify vsphere-config.yaml.template:

vSphere configuration data

Parameter	Description
SET_VSPHERE_SERVER_IP	IP address of the VMware vCenter Server.
SET_VSPHERE_CAPI_PROVIDER_USERNAME	vSphere Cluster API provider user name. For details, see Prepare the VMWare deployment user setup and permissions.
SET_VSPHERE_CAPI_PROVIDER_PASSWORD	vSphere Cluster API provider password.
SET_VSPHERE_CLOUD_PROVIDER_USERNAME	VMware deployment user name. For details, see Prepare the VMWare deployment user setup and permissions.
SET_VSPHERE_CLOUD_PROVIDER_PASSWORD	VMware deployment user password.

SET_VSPHERE_DATACENTER_NAME	VMware datacenter name.
SET_VSPHERE_DATACENTER_PATH	VMware datacenter path.
SET_VSPHERE_DATASTORE_NAME	VMware datastore name.
SET_VSPHERE_NETWORK_PATH	Path to network for cluster machines.
SET_VSPHERE_RESOURCE_POOL_PATH	Path to a resource pool in which VMs will be created.
SET_VSPHERE_MACHINES_FOLDER	Path to a folder where the cluster machines metadata will be stored.
SET_VSPHERE_SERVER_INSECURE	Flag that controls validation of the vSphere Server certificate.
SET_VSPHERE_TEMPLATE_PATH	Path to the prepared OVF template for cluster machines.

10. Optional. Skip this step to use the default password password in the Container Cloud web UI.

    Configure the IAM parameters:

    1. Create hashed passwords for every IAM role: reader, writer, and operator for bare metal deployments:

       ```
       ./bin/hash-generate -i 27500
       ```

       The hash-generate utility requests you to enter a password and outputs the parameters required for the next step. Save the password that you enter in a secure location. This password will be used to access the Container Cloud web UI with a specific IAM role.

       Example of system response:

       ```
       passwordSalt: 6ibPZdUfQK8PsOpSmyVJnA==
       passwordHash: 23W1l65FBdI3NL7LMiUQG9Cu62bWLTqIsOgdW8xNsqw=
       passwordHashAlgorithm: pbkdf2-sha256
       passwordHashIterations: 27500
       ```

       Run the tool several times to generate hashed passwords for every IAM role.

    2. Open templates/cluster.yaml.template for editing.
    3. In the initUsers section, add the following parameters for each IAM role that you generated in the previous step:

       - passwordSalt - base64-encoded randomly generated sequence of bytes.

   - passwordHash - base64-encoded password hash generated using passwordHashAlgorithm with passwordHashIterations. Supported algorithms include pbkdf2-sha256 and pbkdf-sha512.

11. Optional. Configure external identity provider for IAM.
12. Run the bootstrap script:

    ```
    ./bootstrap.sh all
    ```

13. When the bootstrap is complete, collect and save the following management cluster details in a secure location:

    - The kubeconfig file located in the same directory as the bootstrap script. This file contains the admin credentials for the management cluster.
    - The private SSH key openstack_tmp located in ~/.ssh/ for access to the management cluster nodes.

      Note

      The SSH key name openstack_tmp is the same for all cloud providers. This name will be changed in one of the following Container Cloud releases to avoid confusion with a cloud provider name and its related SSH key name.

    - The URL and credentials for the Container Cloud web UI. The system outputs these details when the bootstrap completes.
    - The Keycloak URL that the system outputs when the bootstrap completes. The admin password for Keycloak is located in kaas-bootstrap/passwords.yml along with other IAM passwords.

    Note

    When the bootstrap is complete, the bootstrap cluster resources are freed up.

Now, you can proceed with operating your management cluster using the Container Cloud web UI and deploying managed clusters as described in Create a VMWare vSphere-based managed cluster.

Seealso

- Operations Guide: Connect to a Container Cloud cluster
- Operations Guide: Remove a management cluster

Deploy an additional regional cluster

After you bootstrap a management cluster of the required cloud provider type, you can deploy an additional regional cluster of the same or different provider type.

Supported combinations of providers types for management and regional clusters

	Bare metal regional cluster	**AWS regional cluster**	**OpenStack regional cluster**
Bare metal management cluster	✗	✗	✓
AWS management cluster	✗	✓	✓
OpenStack management cluster	✗	✗	✓

Multi-regional deployment enables you to create managed clusters of several provider types using one management cluster. For example, you can bootstrap an AWS-based management cluster and deploy an OpenStack-based regional cluster on this management cluster. Such cluster enables creation of OpenStack-based and AWS-based managed clusters with Kubernetes deployments.

Note

The integration of baremetal-based support for deploying additional regional clusters is in the final development stage and will be announced separately in one of the upcoming Mirantis Container Cloud releases.

Note

The integration of VMWare vSphere-based support for deploying additional regional clusters is in the development stage and will be announced separately in one of the future Container Cloud releases.

This section describes how to deploy an additional OpenStack or AWS-based regional cluster on an existing management cluster.

Deploy an AWS-based regional cluster

If you want to deploy AWS-based managed clusters of different configurations, deploy an additional regional cluster with specific settings that differ from the AWS-based management cluster configuration.

To deploy an AWS-based regional cluster:

1. Log in to the node where you bootstrapped a management cluster.
2. Prepare the AWS configuration for the new regional cluster:
    1. Verify access to the target cloud endpoint from Docker. For example:

        ```
        docker run --rm alpine sh -c "apk add --no-cache curl; \
        curl https://ec2.amazonaws.com"
        ```

        The system output must contain no error records. In case of issues, follow the steps provided in Troubleshooting.
    2. Change the directory to the kaas-bootstrap folder.
    3. In templates/aws/machines.yaml.template, modify the spec:providerSpec:value section by substituting the ami:id parameter with the corresponding value for Ubuntu 18.04 from the required AWS region. For example:

        ```
        spec:
          providerSpec:
            value:
              apiVersion: aws.kaas.mirantis.com/v1alpha1
              kind: AWSMachineProviderSpec
              instanceType: c5d.2xlarge
              ami:
                id: ami-033a0960d9d83ead0
        ```

        Also, modify other parameters as required.
    4. Optional. In templates/aws/cluster.yaml.template, modify the default configuration of the AWS instance types and AMI IDs for further creation of managed clusters:

        ```
        providerSpec:
            value:
              ...
              kaas:
                ...
                regional:
                - provider: aws
                  helmReleases:
                    - name: aws-credentials-controller
                      values:
                        config:
                          allowedInstanceTypes:
        ```

```
              minVCPUs: 8
              # in MiB
              minMemory: 16384
              # in GB
              minStorage: 120
              supportedArchitectures:
              - "x86_64"
              filters:
              - name: instance-storage-info.disk.type
                values:
                  - "ssd"
            allowedAMIs:
            -
              - name: name
                values:
                - "ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20200729"
              - name: owner-id
                values:
                - "099720109477"
```

Also, modify other parameters as required.

5. Generate the AWS Access Key ID with Secret Access Key for the bootstrapper.cluster-api-provider-aws.kaas.mirantis.com user, that was created in the previous step, and select the AWS default region name.
6. Export the AWS bootstrapper.cluster-api-provider-aws.kaas.mirantis.com user credentials that were created in the previous step:

```
export KAAS_AWS_ENABLED=true
export AWS_SECRET_ACCESS_KEY=XXXXXXX
export AWS_ACCESS_KEY_ID=XXXXXXX
export AWS_DEFAULT_REGION=us-east-2
```

3. Export the following parameters:

```
export KUBECONFIG=<pathToMgmtClusterKubeconfig>
export REGIONAL_CLUSTER_NAME=<newRegionalClusterName>
export REGION=<NewRegionName>
```

Substitute the parameters enclosed in angle brackets with the corresponding values of your cluster.

4. Run the regional cluster bootstrap script:

```
./bootstrap.sh deploy_regional
```

Note

When the bootstrap is complete, obtain and save in a secure location the kubeconfig-<regionalClusterName> file located in the same directory as the bootstrap script. This file contains the admin credentials for the regional cluster.

The workflow of the regional cluster bootstrap script

#	Description
1	Prepare the bootstrap cluster for the new regional cluster.
2	Load the updated Container Cloud CRDs for Credentials, Cluster, and Machines with information about the new regional cluster to the management cluster.
3	Connect to each machine of the management cluster through SSH.
4	Wait for the Machines and Cluster objects of the new regional cluster to be ready on the management cluster.
5	Load the following objects to the new regional cluster: Secret with the management cluster kubeconfig and ClusterRole for the Container Cloud provider.
6	Forward the bootstrap cluster endpoint to helm-controller.
7	Wait for all CRDs to be available and verify the objects created using these CRDs.
8	Pivot the cluster API stack to the regional cluster.
9	Switch the LCM agent from the bootstrap cluster to the regional one.
10	Wait for the Container Cloud components to start on the regional cluster.

Now, you can proceed with deploying the managed clusters of supported provider types as described in Create and operate a managed cluster.

Seealso

Operations Guide: How to remove a regional cluster

Deploy an OpenStack-based regional cluster

You can deploy an additional regional OpenStack-based cluster on top of the AWS, bare metal, or OpenStack management cluster to create managed clusters of several provider types if required.

To deploy an OpenStack-based regional cluster:

1. Log in to the node where you bootstrapped a management cluster.
2. Prepare the OpenStack configuration for a new regional cluster:
    1. Log in to the OpenStack Horizon.
    2. In the Project section, select API Access.
    3. In the right-side drop-down menu Download OpenStack RC File, select OpenStack clouds.yaml File.
    4. Add the downloaded clouds.yaml file to the directory with the bootstrap.sh script.
    5. In clouds.yaml, add the password field with your OpenStack password under the clouds/openstack/auth section.

       Example:

       ```
       clouds:
         openstack:
           auth:
             auth_url: https://auth.openstack.example.com:5000/v3
             username: your_username
             password: your_secret_password
             project_id: your_project_id
             user_domain_name: your_ldap_password
           region_name: RegionOne
           interface: public
           identity_api_version: 3
       ```
    6. Verify access to the target cloud endpoint from Docker. For example:

       ```
       docker run --rm alpine sh -c "apk add --no-cache curl; \
       curl https://auth.openstack.example.com:5000/v3"
       ```

       The system output must contain no error records. In case of issues, follow the steps provided in Troubleshooting.
    7. Change the directory to the kaas-bootstrap folder.
    8. In templates/machines.yaml.template, modify the spec:providerSpec:value sections for set: master and set: node by substituting the flavor and image parameters with the corresponding values of your OpenStack cluster. For example:

       ```
       spec:
         providerSpec:
           value:
             apiVersion: "openstackproviderconfig.k8s.io/v1alpha1"
             kind: "OpenstackMachineProviderSpec"
       ```

```
    flavor: kaas.minimal
    image: bionic-server-cloudimg-amd64-20190612
```

Also, modify other parameters as required.

9. Modify the templates/cluster.yaml.template parameters to fit your deployment. For example, add the corresponding values for cidrBlocks in the spec::clusterNetwork::services section.

   Note

   The passwordSalt and passwordHash values for the IAM roles are automatically re-generated during the IAM configuration described in the next step.

3. Clean up the environment configuration:

   1. If you are deploying the regional cluster on top of a baremetal-based management cluster, unset the following parameters:

      ```
      unset KAAS_BM_ENABLED KAAS_BM_FULL_PREFLIGHT KAAS_BM_PXE_IP \
          KAAS_BM_PXE_MASK KAAS_BM_PXE_BRIDGE KAAS_BM_BM_DHCP_RANGE \
          TEMPLATES_DIR
      ```

   2. If you are deploying the regional cluster on top of an AWS-based management cluster, unset the KAAS_AWS_ENABLED parameter:

      ```
      unset KAAS_AWS_ENABLED
      ```

4. Export the following parameters:

   ```
   export KUBECONFIG=<pathToMgmtClusterKubeconfig>
   export REGIONAL_CLUSTER_NAME=<newRegionalClusterName>
   export REGION=<NewRegionName>
   ```

   Substitute the parameters enclosed in angle brackets with the corresponding values of your cluster.

5. Run the regional cluster bootstrap script:

   ```
   ./bootstrap.sh deploy_regional
   ```

Note

When the bootstrap is complete, obtain and save in a secure location the kubeconfig-<regionalClusterName> file located in the same directory as the bootstrap script. This file contains the admin credentials for the regional cluster.

The workflow of the regional cluster bootstrap script

#	Description
1	Prepare the bootstrap cluster for the new regional cluster.
2	Load the updated Container Cloud CRDs for Credentials, Cluster, and Machines with information about the new regional cluster to the management cluster.
3	Connect to each machine of the management cluster through SSH.
4	Wait for the Machines and Cluster objects of the new regional cluster to be ready on the management cluster.
5	Load the following objects to the new regional cluster: Secret with the management cluster kubeconfig and ClusterRole for the Container Cloud provider.
6	Forward the bootstrap cluster endpoint to helm-controller.
7	Wait for all CRDs to be available and verify the objects created using these CRDs.
8	Pivot the cluster API stack to the regional cluster.
9	Switch the LCM agent from the bootstrap cluster to the regional one.
10	Wait for the Container Cloud components to start on the regional cluster.

Now, you can proceed with deploying the managed clusters of supported provider types as described in Create and operate a managed cluster.

Seealso

Operations Guide: How to remove a regional cluster

Troubleshooting

This section provides solutions to the issues that may occur while deploying a management cluster.

Collect the bootstrap logs

If the bootstrap script fails during the deployment process, collect and inspect the bootstrap and management cluster logs.

To collect the bootstrap logs:

1. Log in to your local machine where the bootstrap script was executed.
2. Run the following command:

   ```
   ./bootstrap.sh collect_logs
   ```

   The logs are collected in the directory where the bootstrap script is located.

Starting from the Container Cloud release 2.2.0, the logs structure is as follows:

- <output_dir>/<cluster_name>/events.log - human-readable table that contains information about the cluster events
- <output_dir>/<cluster_name>/system - system logs
- <output_dir>/<cluster_name>/objects/cluster - logs of the non-namespaced Kubernetes objects
- <output_dir>/<cluster_name>/objects/namespaced - logs of the namespaced Kubernetes objects
- <output_dir>/<cluster_name>/objects/namespaced/<namespace>/core/pods - pods logs from a specified Kubernetes namespace

Depending on the type of issue found in logs, apply the corresponding fixes. For example, if you detect the LoadBalancer ERROR state errors during the bootstrap of an OpenStack-based management cluster, contact your system administrator to fix the issue. To troubleshoot other issues, refer to the corresponding section in Troubleshooting.

DNS settings

If you have issues related to the DNS settings, the following error message may occur:

```
curl: (6) Could not resolve host
```

The issue may occur if a VPN is used to connect to the cloud or a local DNS forwarder is set up.

The workaround is to change the default DNS settings for Docker:

1. Log in to your local machine.
2. Identify your internal or corporate DNS server address:

```
systemd-resolve --status
```

3. Create or edit /etc/docker/daemon.json by specifying your DNS address:

```
{
 "dns": ["<YOUR_DNS_ADDRESS>"]
}
```

4. Restart the Docker daemon:

```
sudo systemctl restart docker
```

Default network address

If you have issues related to the default network address configuration, cURL either hangs or the following error occurs:

```
curl: (7) Failed to connect to xxx.xxx.xxx.xxx port xxxx: Host is unreachable
```

The issue may occur because the default Docker network address 172.17.0.0/16 overlaps with your cloud address or other addresses of the network configuration.

Workaround:

1. Log in to your local machine.
2. Verify routing to the IP addresses of the target cloud endpoints:
   1. Obtain the IP address of your target cloud. For example:

      ```
      nslookup auth.openstack.example.com
      ```

      Example of system response:

      ```
      Name:   auth.openstack.example.com
      Address: 172.17.246.119
      ```

   2. Verify that this IP address is not routed through docker0 but through any other interface, for example, ens3:

      ```
      ip r get 172.17.246.119
      ```

      Example of the system response if the routing is configured correctly:

      ```
      172.17.246.119 via 172.18.194.1 dev ens3 src 172.18.1.1 uid 1000
        cache
      ```

Example of the system response if the routing is configured incorrectly:

```
172.17.246.119 via 172.18.194.1 dev docker0 src 172.18.1.1 uid 1000
  cache
```

3. If the routing is incorrect, change the IP address of the default Docker bridge:

    1. Create or edit /etc/docker/daemon.json by adding the "bip" option:

    ```
    {
      "bip": "192.168.91.1/24"
    }
    ```

    2. Restart the Docker daemon:

    ```
    sudo systemctl restart docker
    ```

TLS handshake timeout

If you execute the bootstrap.sh script from an OpenStack VM that is running on the OpenStack environment used for bootstrapping the management cluster, the following error messages may occur that can be related to the MTU settings discrepancy:

```
curl: (35) OpenSSL SSL_connect: SSL_ERROR_SYSCALL in connection to server:port

Failed to check if machine "<machine_name>" exists:
failed to create provider client ... TLS handshake timeout
```

To identify whether the issue is MTU-related:

1. Log in to the OpenStack VM in question.
2. Compare the MTU outputs for the docker0 and ens3 interfaces:

    ```
    ip addr
    ```

    Example of system response:

    ```
    3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500...
    ...
    2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450...
    ```

    If the MTU output values differ for docker0 and ens3, proceed with the workaround below. Otherwise, inspect the logs further to identify the root cause of the error messages.

Workaround:

1. In your OpenStack environment used for Mirantis Container Cloud, log in to any machine with CLI access to OpenStack. For example, you can create a new Ubuntu VM (separate from the bootstrap VM) and install the python-openstackclient package on it.
2. Change the vXLAN MTU size for the VM to the required value depending on your network infrastructure and considering your physical network configuration, such as Jumbo frames, and so on.

   ```
   openstack network set --mtu <YOUR_MTU_SIZE> <network-name>
   ```

3. Stop and start the VM in Nova.
4. Log in to the bootstrap VM dedicated for the management cluster.
5. Re-execute the bootstrap.sh script.

Configure external identity provider for IAM

This section describes how to configure authentication for Mirantis Container Cloud depending on the external identity provider type integrated to your deployment.

Configure LDAP for IAM

If you integrate LDAP for IAM to Mirantis Container Cloud, add the required LDAP configuration to cluster.yaml.template during the bootstrap of the management cluster.

> Note
>
> The example below defines the recommended non-anonymous authentication type. If you require anonymous authentication, replace the following parameters with authType: "none":
>
> ```
> authType: "simple"
> bindCredential: ""
> bindDn: ""
> ```

To configure LDAP for IAM:

1. Select from the following options:
   - For a baremetal-based management cluster, open the templates/bm/cluster.yaml.template file for editing.
   - For an OpenStack management cluster, open the templates/cluster.yaml.template file for editing.
   - For an AWS-based management cluster, open the templates/aws/cluster.yaml.template file for editing.
2. Configure the keycloak:userFederation:providers: and keycloak:userFederation:mappers: sections as required:

   > Note
   >
   > Verify that the userFederation section is located on the same level as the initUsers section.

   ```
   spec:
     providerSpec:
       value:
         kaas:
           management:
             helmReleases:
   ```

```
      - name: iam
        values:
          keycloak:
            userFederation:
              providers:
                - displayName: "<LDAP_NAME>"
                  providerName: "ldap"
                  priority: 1
                  fullSyncPeriod: -1
                  changedSyncPeriod: -1
                  config:
                    pagination: "true"
                    debug: "false"
                    searchScope: "1"
                    connectionPooling: "true"
                    usersDn: "<DN>" # "ou=People, o=<ORGANIZATION>, dc=<DOMAIN_COMPONENT>"
                    userObjectClasses: "inetOrgPerson,organizationalPerson"
                    usernameLDAPAttribute: "uid"
                    rdnLDAPAttribute: "uid"
                    vendor: "ad"
                    editMode: "READ_ONLY"
                    uuidLDAPAttribute: "uid"
                    connectionUrl: "ldap://<LDAP_DNS>"
                    syncRegistrations: "false"
                    authType: "simple"
                    bindCredential: ""
                    bindDn: ""
              mappers:
                - name: "username"
                  federationMapperType: "user-attribute-ldap-mapper"
                  federationProviderDisplayName: "<LDAP_NAME>"
                  config:
                    ldap.attribute: "uid"
                    user.model.attribute: "username"
                    is.mandatory.in.ldap: "true"
                    read.only: "true"
                    always.read.value.from.ldap: "false"
                - name: "full name"
                  federationMapperType: "full-name-ldap-mapper"
                  federationProviderDisplayName: "<LDAP_NAME>"
                  config:
                    ldap.full.name.attribute: "cn"
                    read.only: "true"
                    write.only: "false"
                - name: "last name"
                  federationMapperType: "user-attribute-ldap-mapper"
                  federationProviderDisplayName: "<LDAP_NAME>"
                  config:
```

```
              ldap.attribute: "sn"
              user.model.attribute: "lastName"
              is.mandatory.in.ldap: "true"
              read.only: "true"
              always.read.value.from.ldap: "true"
          - name: "email"
            federationMapperType: "user-attribute-ldap-mapper"
            federationProviderDisplayName: "<LDAP_NAME>"
            config:
              ldap.attribute: "mail"
              user.model.attribute: "email"
              is.mandatory.in.ldap: "false"
              read.only: "true"
              always.read.value.from.ldap: "true"
```

Now, return to the bootstrap instruction depending on the provider type of your management cluster.

Configure Google OAuth IdP for IAM

Caution!

The instruction below applies to the DNS-based management clusters. If you bootstrap a non-DNS-based management cluster, configure Google OAuth IdP for Keycloak after bootstrap using the official Keycloak documentation.

If you integrate Google OAuth external identity provider for IAM to Mirantis Container Cloud, create the authorization credentials for IAM in your Google OAuth account and configure cluster.yaml.template during the bootstrap of the management cluster.

To configure Google OAuth IdP for IAM:

1. Create Google OAuth credentials for IAM:
    1. Log in to your https://console.developers.google.com.
    2. Navigate to Credentials.
    3. In the APIs Credentials menu, select OAuth client ID.
    4. In the window that opens:
        1. In the Application type menu, select Web application.
        2. In the Authorized redirect URIs field, type in <keycloak-url>/auth/realms/iam/broker/google/endpoint, where <keycloak-url> is the corresponding DNS address.
        3. Press Enter to add the URI.

   4. Click Create.

   A page with your client ID and client secret opens. Save these credentials for further usage.

2. Log in to the bootstrap node.
3. Select from the following options:
   - For a baremetal-based management cluster, open the templates/bm/cluster.yaml.template file for editing.
   - For an OpenStack management cluster, open the templates/cluster.yaml.template file for editing.
   - For an AWS-based management cluster, open the templates/aws/cluster.yaml.template file for editing.
4. In the keycloak:externalIdP: section, add the following snippet with your credentials created in previous steps:

```
keycloak:
  externalIdP:
    google:
      enabled: true
      config:
        clientId: <Google_OAuth_client_ID>
        clientSecret: <Google_OAuth_client_secret>
```

Now, return to the bootstrap instruction depending on the provider type of your management cluster.