Copyright notice

2020 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:

<table>
<thead>
<tr>
<th>Revision date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 8, 2019</td>
<td>Q4`18 GA</td>
</tr>
</tbody>
</table>
List of Salt formulas supported in MCP

Salt formulas are pre-written Salt states. They are open-ended and can be used for such tasks as package installation, service configuration and starting, users and permissions setup, and others.

In MCP, the Salt formulas together with Salt are used as a configuration management tool that configures, deploys, and updates the MCP components. Each Salt formula defines a corresponding MCP component, such as the mysql formula for MySQL, rabbitmq formula for RabbitMQ, formulas for OpenStack services, and so on.

The Salt formulas supported in MCP includes:

- aodh
- apache
- aptcacher
- aptly
- artifactory
- avinetworks
- backupninja
- barbican
- baremetal-simulator
- bind
- cassandra
- ceilometer
- ceph
- cinder
- collectd
- designate
- docker
- dogtag
- elasticsearch
- etcd
- fluentd
- freeipa
- galera
- gerrit
- git
• glance
• glusterfs
• gnocchi
• grafana
• haproxy
• heat
• heka
• helm
• horizon
• influxdb
• iptables
• ironic
• isc-dhcp
• java
• jenkins
• keepalived
• keystone
• kibana
• kubernetes
• libvirt
• linux
• lldp
• logrotate
• maas
• memcached
• mongodb
• muranomysql
• neutron
• nginx
• nova
• ntp
• octavia
• opencontrail
- openldap
- openssh
- panko
- postgresql
- powerdns
- prometheus
- python
- rabbitmq
- reclass
- redis
- rsync
- rsyslog
- rundeck
- salt
- sensu
- sphinx
- statsd
- telegraf
- tftpd-hpa
- tinyproxy
- xtrabackup
- zookeeper
DEBMIRROR

Usage

This file provides the debmirror sample pillars configurations for different use cases.

See debmirror/schemas/*.yaml for all possible options A sample of one debmirror mirror configuration (Ubuntu):

```yaml
parameters:
debmirror:
  client:
    enabled: true
  mirrors:
    target01:
      enabled: true
      fetch_retry: 3
      http_proxy: "url"
      https_proxy: "url"
      ftp_proxy: "url"
      rsync_proxy: "url"
      no_proxy: ['val1', 'val2']
      force: False
      lock_target: True
      extra_flags: [ '--verbose', '--progress', '--nosource', '--no-check-gpg', '--rsync-extra=none' ]
      method: "rsync" # string
      arch: ['amd64']
      mirror_host: "mirror.mirantis.com" # rsync
      mirror_root: ':mirror/nightly/ubuntu/'
      target_dir: "/var/www/mirror/ubuntu/
      cache_dir: "/var/www/mirror/.cache/ubuntu"
      target01_log: "/var/www/mirror/target01_log.log"
      dist: [ xenial ] #, xenial-security, xenial-updates ]
      section: [ main ] #, multiverse, restricted, universe ]
      exclude_deb_section: [ 'games', gnome, Xfce, sound, electronics, graphics, hamradio , doc, localization, kde, video ]
    filter:
      00: "--exclude="/"
      01: "--exclude="/android**"
      02: "--exclude="/firefox**"
      03: "--exclude="/chromium-browser**"
      04: "--exclude="/ceph**"
      05: "--exclude="/*-wallpapers**"
      06: "--exclude="/language-pack-(?!en)**"
      07: "--include="/main(*)(.*manpages""
      08: "--include="/main(*)(.*python-.*doc""
      09: "--include="/main(*)(.*python-.*network"
```

The cache_dir parameter is optional and can be used to avoid extra disk space usage for repos, which can have same packages, by using hardlinks to files.

Metadata schema specifications for debmirror client

Core Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables debmirror processing.</td>
</tr>
</tbody>
</table>
mirrors | object | Set of mirror to sync For details, see: debmirror:mirror definition

debmirror:mirror definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dist</td>
<td>array</td>
<td>description_notset</td>
</tr>
<tr>
<td>target_dir</td>
<td>string</td>
<td>Destination folder for mirror</td>
</tr>
<tr>
<td>http_proxy</td>
<td>string</td>
<td>Specify proxy parameter.</td>
</tr>
<tr>
<td>ftp_proxy</td>
<td>string</td>
<td>Specify proxy parameter.</td>
</tr>
<tr>
<td>exclude_deb_section</td>
<td>array</td>
<td>Never download any files whose Debian Section (games, doc, oldlibs, science, etc.) match the regex.</td>
</tr>
<tr>
<td>rsync_proxy</td>
<td>string</td>
<td>Specify proxy parameter.</td>
</tr>
<tr>
<td>fetch_retry</td>
<td>integer</td>
<td>Number of retries, to fetch mirror. Works only with Salt 2017+.</td>
</tr>
<tr>
<td>force</td>
<td>boolean</td>
<td>Ignore lockfile</td>
</tr>
<tr>
<td>arch</td>
<td>array</td>
<td>description_notset</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Sorted list of any kind filtered options. Possible marks include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• --ignore=regex Never delete any files whose filenames match the regex.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• --exclude=regex Never download any files whose filenames match the regex.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• --include=regex Don’t exclude any files whose filenames match the regex.</td>
</tr>
<tr>
<td>mirror_root</td>
<td>string</td>
<td>Specifies the directory on the remote host that is the root of the Ubuntu archive. The root directory has a dists subdirectory.</td>
</tr>
<tr>
<td>no_proxy</td>
<td>array</td>
<td>Specifies list of host-excludes for proxy.</td>
</tr>
<tr>
<td>mirror_host</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>section</td>
<td>array</td>
<td>Specifies the section of Ubuntu to mirror.</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables exact mirror processing.</td>
</tr>
<tr>
<td>extra_flags</td>
<td>array</td>
<td>description_notset</td>
</tr>
<tr>
<td>lock_target</td>
<td>boolean</td>
<td>Creates lockfile inside target dir, to prevent future repo updates</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>https_proxy</td>
<td>string</td>
<td>Specifies proxy parameter</td>
</tr>
<tr>
<td>log_file</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>method</td>
<td>string</td>
<td>Specifies the method to download files. Currently, supported methods are ftp, http, https, and rsync. The file method is experimentally supported.</td>
</tr>
</tbody>
</table>
NTP

Usage

The Network Time Protocol (NTP) formula is used to properly synchronize services across the nodes. This file provides the sample configurations for different use cases.

• [Deprecated] NTP client configuration, should not be used if the stratum parameter exists:

```yaml
ntp:
  client:
    enabled: true
    strata:
      - ntp.cesnet.cz
      - ntp.nic.cz
```

• The NTP client extended definition with auth:

```yaml
ntp:
  client:
    enabled: true
    stratum:
      primary:
        server: ntp.cesnet.cz
        key_id: 1
      secondary:
        server: ntp.nic.cz
        key_id: 2
```

• The NTP client with MD5 auth configuration:

```yaml
ntp:
  client:
    enabled: true
    auth:
      enabled: true
      secrets:
        1:
          secret_type: 'M'
          secret: 'Runrabbitrundigthath'
          trustedkey: true
        2:
          secret_type: 'M'
          secret: 'Howiwishyouwereherew'
          trustedkey: true
  stratum:
    primary:
      server: ntp.cesnet.cz
```
• The NTP server with MD5 auth configuration:

```yaml
ntp:
  client:
    enabled: false
  server:
    enabled: true
    auth:
      enabled: true
  secrets:
    1:
      secret_type: 'M'
      secret: 'Runrabbitrundigthath'
      trustedkey: true
    2:
      secret_type: 'M'
      secret: 'Howiwishyouwereherew'
      trustedkey: true
stratum:
  primary:
    server: ntp.cesnet.cz
    key_id: 1
  secondary:
    server: ntp.nic.cz
    key_id: 2
```

• A cleaning up of the NTP configurations left by DHCP:

```yaml
ntp:
  client:
    enabled: true
  remove_dhcp_conf: true # default false
```

• The NTP server simple peering definition:

```yaml
ntp:
  server:
    peers:
      - 192.168.0.241
      - 192.168.0.242
```

• The NTP server extended peering definition:
ntp:
  server:
  peers:
    1:
      host: 192.168.31.1
    2:
      host: 192.168.31.2
    3:
      host: 192.168.31.3

- The NTP server definition enabling the listen and ignore actions on specific addresses:

  ntp:
    server:
      1:
        value: wildcard
        action: ignore
      2:
        value: ::1
        action: listen
      3:
        value: 192.168.31.1
        action: listen

Read more


**Metadata schema specifications for NTP client**

**Core Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode7</td>
<td>boolean</td>
<td>Enables mode7 for the NTP server.</td>
</tr>
<tr>
<td>remove_dhcp_conf</td>
<td>boolean</td>
<td>Forcibly remove “/var/lib/ntp/ntp.conf.dhcp” file. WA for issue</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=600661">https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=600661</a></td>
</tr>
<tr>
<td>stratum</td>
<td>object</td>
<td>List of NTP stratums to keep the time in sync. If define used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>instead of strata. For details, see: ntp:common:stratum definition</td>
</tr>
<tr>
<td>logfile</td>
<td>string</td>
<td>NTP log file path.</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables NTP client service.</td>
</tr>
<tr>
<td>strata</td>
<td>array</td>
<td>List of NTP stratums to keep the time in sync. For details, see: ntp:common:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stratum definition</td>
</tr>
<tr>
<td>secrets</td>
<td>object</td>
<td>Dict with secrets For details, see: ntp:common:secret definition</td>
</tr>
</tbody>
</table>
**Metadata schema specifications for NTP server**

### Core Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode7</td>
<td>boolean</td>
<td>Enables mode7 for the NTP server.</td>
</tr>
<tr>
<td>peers</td>
<td>array</td>
<td>List of peered NTP stratum services. For details, see: ntp:server:peer definition</td>
</tr>
<tr>
<td>remove_dhcp_conf</td>
<td>boolean</td>
<td>Forcibly remove “/var/lib/ntp/ntp.conf.dhcp” file. WA for issue <a href="https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=600661">https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=600661</a></td>
</tr>
<tr>
<td>orphan</td>
<td>number</td>
<td>Sets the orphan level of the NTP server.</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables NTP server service.</td>
</tr>
<tr>
<td>strata</td>
<td>array</td>
<td>List of NTP stratums to keep the time in sync. For details, see: ntp:common:strata definition</td>
</tr>
<tr>
<td>secrets</td>
<td>object</td>
<td>Dict with secrets For details, see: ntp:common:secret definition</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables NTP auth.</td>
</tr>
</tbody>
</table>
restrict | array | List of subnets that servers gives time to. For details, see: ntp:server:restrict definition
---|---|---
stratum | object | List of NTP stratum to keep the time in sync. If define used instead of strata For details, see: ntp:common:stratum definition
logfile | string | NTP log file path.

ntp:common:secret definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>secret_type</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>secret</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>trustedkey</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
</tbody>
</table>

ntp:common:interface definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>string</td>
<td>Determines the action for addresses which match</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>That parameter specifies a class of addresses, or a specific interface name, or an address. In the address case, prefixlen determines how many bits must match for this rule to apply. Ignore prevents opening matching addresses, drop causes ntpd to open the address and drop all received packets without examination.</td>
</tr>
</tbody>
</table>

ntp:common:stratum definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_id</td>
<td>integer</td>
<td>description_notset</td>
</tr>
<tr>
<td>server</td>
<td>string</td>
<td>description_notset</td>
</tr>
</tbody>
</table>

ntp:server:peer definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_id</td>
<td>integer</td>
<td>description_notset</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>description_notset</td>
</tr>
</tbody>
</table>

ntp:common:strata definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
</table>
ntp:common::strata

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet</td>
<td>string</td>
<td>IP address of the network</td>
</tr>
<tr>
<td>mask</td>
<td>string</td>
<td>Subnet mask of the network</td>
</tr>
<tr>
<td>options</td>
<td>string</td>
<td>Additional options passed to the net [notrap nomodify]</td>
</tr>
</tbody>
</table>
OPENSSH

Usage

OpenSSH is a free version of the SSH connectivity tools that technical users of the Internet rely on. The passwords of Telnet, remote login (rlogin), and File Transfer Protocol (FTP) users are transmitted across the Internet unencrypted. OpenSSH encrypts all traffic, including passwords, to effectively eliminate eavesdropping, connection hijacking, and other attacks. Additionally, OpenSSH provides secure tunneling capabilities and several authentication methods, and supports all SSH protocol versions.

This file provides the sample pillars configurations for different use cases.

OpenSSH client

• The OpenSSH client configuration with a shared private key:

```yaml
openssh:
  client:
    enabled: true
    use_dns: False
  user:
    root:
      enabled: true
      private_key:
        type: rsa
        key: ${_param:root_private_key}
      user: ${linux:system:user:root}
```

• The OpenSSH client configuration with an individual private key and known host:

```yaml
openssh:
  client:
    enabled: true
  user:
    root:
      enabled: true
      user: ${linux:system:user:root}
  known_hosts:
    - name: repo.domain.com
      type: rsa
      fingerprint_hash_type: sha256|md5
```

• The OpenSSH client configuration with keep alive settings:

```yaml
openssh:
  client:
    alive:
```
OpenSSH server

• The OpenSSH server simple configuration:

```
interval: 600
count: 3
```

```
openssh:
  server:
    enabled: true
    permit_root_login: true
    public_key_auth: true
    password_auth: true
    host_auth: true
    banner: Welcome to server!
    bind:
      address: 0.0.0.0
      port: 22
```

• The OpenSSH server configuration with auth keys for users:

```
openssh:
  server:
    enabled: true
    bind:
      address: 0.0.0.0
      port: 22

... user:
  newt:
    enabled: true
    user: ${linux:system:user:newt}
    public_keys:
      - ${public_keys:newt}
  root:
    enabled: true
    purge: true
    user: ${linux:system:user:root}
    public_keys:
      - ${public_keys:newt}
```

Note

Setting the purge parameter to true ensures that the exact authorized_keys contents will be filled explicitly from the model and undefined keys will be removed.
• The OpenSSH server configuration that binds OpenSSH on multiple addresses and ports:

```yaml
openssh:
  server:
    enabled: true
    binds:
      - address: 127.0.0.1
        port: 22
      - address: 192.168.1.1
        port: 2222
```

• The OpenSSH server with FreeIPA configuration:

```yaml
openssh:
  server:
    enabled: true
    bind:
      address: 0.0.0.0
      port: 22
      public_key_auth: true
      authorized_keys_command:
        command: /usr/bin/sss_ssh_authorizedkeys
        user: nobody
```

• The OpenSSH server configuration with keep alive settings:

```yaml
openssh:
  server:
    alive:
      keep: yes
      interval: 600
      count: 3

# will give you an timeout of 30 minutes (600 sec x 3)
```

• The OpenSSH server configuration with the DSA legacy keys enabled:

```yaml
openssh:
  server:
    dss_enabled: true
```

• The OpenSSH server configuration with the duo 2FA [https://duo.com/docs/duounix](https://duo.com/docs/duounix) with Match User 2FA can be bypassed for some accounts

```yaml
openssh:
  server:
    use_dns: false
    password_auth: false
```
**challenge_response_auth**: true

ciphers:

  **aes256-ctr**:
  enabled: true

  **aes192-ctr**:
  enabled: true

  **aes128-ctr**:
  enabled: true

authentication_methods:

  **publickey**:
  enabled: true

  **keyboard-interactive**:
  enabled: true

match_user:

  **jenkins**:
  authentication_methods:
    **publickey**:
    enabled: true

• OpenSSH server configuration supports AllowUsers, DenyUsers, AllowGroup, DenyGroups via allow_users, deny_users, allow_groups, deny_groups keys respectively.

For example, here is how to manage AllowUsers configuration item:

```yaml
openssh:
  server:
    allow_users:
      <user_name>:
        enabled: true
      <pattern_list_name>:
        enabled: true
        pattern: <pattern>
```

Elements of allow_users are either user names or pattern list names:

• `<user name>` goes to configurational file as is.

• `<pattern list name>` is not used directly - its main purpose is to provide a meaningful name for a pattern specified in ‘pattern’ key. Another advantage is that pattern can be overridden.

<enabled> by default is ‘true’.

See PATTERNS in ssh_config(5) for more information on what `<pattern>` is.

**CIS Compliance**

There is a number of configuration options that make the OpenSSH service compliant with CIS Benchmark. These options can be found under metadata/service/server/cis, and are not enabled by default. For each CIS item a comprehensive description is provided with the pillar data.

See also [https://www.cisecurity.org/cis-benchmarks/](https://www.cisecurity.org/cis-benchmarks/) for the details about CIS Benchmark.

Read more
• http://www.openssh.org/manual.html
• https://help.ubuntu.com/community/SSH/OpenSSH/Configuring
• http://www.zeitoun.net/articles/ssh-through-http-proxy/start

**Metadata Schema Specifications for OpenSSH client**

### Core Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>known_hosts</td>
<td>array</td>
<td>List of pre-defined known hosts for ssh access. For details, see: openssh_known_hosts_object definition</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables openssh client configuration.</td>
</tr>
<tr>
<td>user</td>
<td>object</td>
<td>Dict of openssh user’s, to be configured. Private</td>
</tr>
<tr>
<td>alive</td>
<td>object</td>
<td>Configure ServerAlive* option</td>
</tr>
</tbody>
</table>

**openssh_known_hosts_object definition**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>fingerprint</td>
<td>string</td>
<td>description_notset</td>
</tr>
</tbody>
</table>

**global_useradd_user definition**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shell</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>sudo</td>
<td>boolean</td>
<td>Allow user to use sudo</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>full_name</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>home</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>email</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>uid</td>
<td>integer</td>
<td>description_notset</td>
</tr>
</tbody>
</table>
**openssh_client_user definition**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_openssh_client_user</td>
<td>object</td>
<td>Define exactly one openssh user. Private</td>
</tr>
</tbody>
</table>

**Metadata schema specifications for OpenSSH server**

**Core Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>integer</td>
<td>Specifies the protocol versions sshd(8) supports. The possible values are “1” and “2”. Multiple versions must be comma-separated. The default is “2”. Protocol 1 suffers from a number of cryptographic weaknesses and should not be used. It is only offered to support legacy devices. Note that the order of the protocol list does not indicate preference, because the client selects among multiple protocol versions offered by the server. Specifying “2,1” is identical to “1,2”.</td>
</tr>
<tr>
<td>kerberos_auth</td>
<td>boolean</td>
<td>KerberosAuthentication Specifies whether the password provided by the user for PasswordAuthentication will be validated through the Kerberos KDC. To use this option, the server needs a Kerberos servtab which allows the verification of the KDC`s identity. The default is False (“no”).</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables / disabled specific algorithm.</td>
</tr>
<tr>
<td>force_command</td>
<td>string</td>
<td>Forces the execution of the command specified by ForceCommand, ignoring any command supplied by the client and ~/.ssh/rc if present.</td>
</tr>
<tr>
<td>syslog_facility</td>
<td>ERROR</td>
<td>SyslogFacility Gives the facility code that is used when logging messages from sshd(8). The possible values are: DAEMON, USER, AUTH, AUTHPRIV, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The default is AUTH.</td>
</tr>
<tr>
<td>public_key_auth</td>
<td>boolean</td>
<td>PubkeyAuthentication Specifies whether public key authentication is allowed. The default is True (“yes”).</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables / disabled specific method.</td>
</tr>
<tr>
<td>password_auth</td>
<td>boolean</td>
<td>PasswordAuthentication Specifies whether password authentication is allowed. The default is True (“yes”).</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>permit_user_environment</td>
<td>boolean</td>
<td>Specifies whether ~/.ssh/environment and environment= options in ~/.ssh/authorized_keys are processed by sshd(8). The default is False (“no”). Enabling environment processing may enable users to bypass access restrictions in some configurations using mechanisms such as LD_PRELOAD.</td>
</tr>
<tr>
<td>banner</td>
<td>string</td>
<td>Banner The contents of the specified file are sent to the remote user before authentication is allowed. If the argument is “none” then no banner is displayed. By default, no banner is displayed.</td>
</tr>
<tr>
<td>login_grace_time</td>
<td>integer</td>
<td>LoginGraceTime The server disconnects after this time if the user has not successfully logged in. If the value is 0, there is no time limit. The default is 120 seconds.</td>
</tr>
<tr>
<td>alive</td>
<td>object</td>
<td>Configure ClientAlive* option’s.</td>
</tr>
<tr>
<td>log_level</td>
<td>ERROR</td>
<td>LogLevel Gives the verbosity level that is used when logging messages from sshd(8). The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. The default is INFO. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of debugging output. Logging with a DEBUG level violates the privacy of users and is not recommended.</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>permit_empty_passwords</td>
<td>boolean</td>
<td>PermitEmptyPasswords When password authentication is allowed, it specifies whether the server allows login to accounts with empty password strings. The default is False (“no”).</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
<td>Specifies the local addresses sshd should listen on.</td>
</tr>
<tr>
<td>address</td>
<td>string</td>
<td>Specifies the port on which the server listens for connections. Multiple options are permitted.</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables openssh server configuration.</td>
</tr>
<tr>
<td>max_auth_tries</td>
<td>integer</td>
<td>MaxAuthTries Specifies the maximum number of authentication attempts permitted per connection. Once the number of failures reaches half this value, additional failures are logged. The default is 6.</td>
</tr>
<tr>
<td>use_dns</td>
<td>boolean</td>
<td>Specifies whether sshd should look up the remote host name, and to check that the resolved host name for the remote IP address maps back to the very same IP address</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables / disabled specific method.</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>host_auth</td>
<td>boolean</td>
<td>Specifies whether rhosts or /etc/hosts.equiv authentication together with successful public key client host authentication is allowed (host-based authentication). The default is False(&quot;no&quot;).</td>
</tr>
<tr>
<td>permit_root_login</td>
<td>boolean</td>
<td>Specifies whether root can log in using ssh(1). The argument must be “yes”, “prohibit-password”, “without-password”, “forced-commands-only”, or “no”. The default is “prohibit-password”. If this option is set to “prohibit-password” or “without-password”, password and keyboard-interactive authentication are disabled for root. If this option is set to “forced-commands-only”, root login with public key authentication will be allowed, but only if the command option has been specified (which may be useful for taking remote backups even if root login is normally not allowed). All other authentication methods are disabled for root. If this option is set to “no”, root is not allowed to log in. # TODO Currently its only boolean option, however, support for other # values has been added recently to sshd_config template, now # it may use both booleans and strings. # Now the next step is to update reclass models and switch # from boolean values to strings.</td>
</tr>
<tr>
<td>ignore_rhosts</td>
<td>boolean</td>
<td>Specifies that .rhosts and .shosts files will not be used in RhostsRSAAuthentication or HostbasedAuthentication. /etc/hosts.equiv and /etc/ssh/shosts.equiv are still used. The default is True (&quot;yes&quot;).</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables / disabled specific MAC algorithm.</td>
</tr>
<tr>
<td>challenge_response_auth</td>
<td>boolean</td>
<td>ChallengeResponseAuthentication controls support for the ‘keyboard-interactive’ authentication scheme defined in RFC-4256. The ‘keyboard-interactive’ authentication scheme could, in theory, ask a user any number of multi-faceted questions. It’s using for duo 2FA authorization.</td>
</tr>
<tr>
<td>user</td>
<td>object</td>
<td>List of openssh user’s, to be configured.</td>
</tr>
</tbody>
</table>

**global_useradd_user definition**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©2020, Mirantis Inc.
<table>
<thead>
<tr>
<th><strong>shell</strong></th>
<th>string</th>
<th>description_notset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td><strong>sudo</strong></td>
<td>boolean</td>
<td>Allow user to use sudo</td>
</tr>
<tr>
<td><strong>enabled</strong></td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td><strong>full_name</strong></td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td><strong>home</strong></td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td><strong>password</strong></td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td><strong>email</strong></td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td><strong>uid</strong></td>
<td>integer</td>
<td>description_notset</td>
</tr>
</tbody>
</table>
**APPLY**

**Usage**
The Apty formula configures and installs the Apty server and client.
The available states include:
  - aptly.server
  - aptly.publisher
The available metadata include:
  - metadata.aptly.server.single
  - metadata.aptly.client.publisher
This file provides the sample configurations for different use cases.
• Reclass examples:
  • The basic Aptly server configuration without repositories or mirrors:

    ```yaml
    classes:
    - service.aptly.server.single

    parameters:
    aptly:
      server:
        enabled: true
        secure: true
        gpg_keypair_id: A76882D3
        gpg_passphrase:
        gpg_public_key: |
          -----BEGIN PGP PUBLIC KEY BLOCK-----
          Version: GnuPG v1
          ...
        gpg_private_key: |
          -----BEGIN PGP PRIVATE KEY BLOCK-----
          Version: GnuPG v1
          ...
    
    • The definition of an s3 endpoint:

    parameters:
    aptly:
      server:
        endpoint:
          mys3endpoint:
            engine: s3
            awsAccessKeyId: xxxx
            awsSecretAccessKey: xxxx
            bucket: test

    • Pillar examples:
  • The Aptly server basic configuration:

    aptly:
      server:
        enabled: true
        repo:
          myrepo:
            distribution: trusty
            component: main
            architectures: amd64
            comment: "Custom components"
            publisher:
              component: mycomponent
distributions:
- nightly/trusty

• The Aptly server mirrors configuration:

```yaml
aptly:
  server:
    mirror:
      mirror_name:
        source: http://example.com/debian
        distribution: xenial
        components: main
        architectures: amd64
        gpgkeys: 460F3999
        filter: "!(Name (% *-dbg))"
        filter_with_deps: true
        publisher:
          component: example
          distributions:
            - xenial/repo/nightly
            - "s3:aptcdn:xenial/repo/nightly"
```

• The definition of the proxy environment variables in cron job for mirroring script:

```yaml
aptly:
  server:
    enabled: true
    ...  
    mirror_update:
      enabled: true
      http_proxy: "http://1.2.3.4:8000"
      https_proxy: "http://1.2.3.4:8000"
      ...
```

Read more

• http://www.aptly.info/doc/configuration/

**Metadata schema specifications for aptly publisher**

**Core properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>engine</td>
<td>string</td>
<td>Installation source for aptly publisher</td>
</tr>
<tr>
<td>image</td>
<td>string</td>
<td>Publisher full image name. Set if installation from docker is chosen</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>root_dir</td>
<td>string</td>
<td>Root directory</td>
</tr>
<tr>
<td>secure</td>
<td>boolean</td>
<td>Enable secure aptly server.</td>
</tr>
<tr>
<td>repo</td>
<td>object</td>
<td>Repo map where key is repo name and value is a list of repo properties.</td>
</tr>
<tr>
<td>gpg_public_key</td>
<td>string</td>
<td>Public key to PGP repository</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Host to bind aptly API service</td>
</tr>
<tr>
<td>port</td>
<td>['string', 'integer']</td>
<td>Port to bind aptly API service</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables aptly API service</td>
</tr>
<tr>
<td>gid</td>
<td>integer</td>
<td>Group id for aptly user</td>
</tr>
<tr>
<td>group</td>
<td>string</td>
<td>Group name for aptly</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>User name for aptly</td>
</tr>
<tr>
<td>uid</td>
<td>integer</td>
<td>User id for aptly user</td>
</tr>
<tr>
<td>mirror</td>
<td>array</td>
<td>Mirror map where key is mirror name and value is a list of mirror properties</td>
</tr>
<tr>
<td>https_proxy</td>
<td>string</td>
<td>HTTPS Proxy for apt mirror access</td>
</tr>
<tr>
<td>http_proxy</td>
<td>string</td>
<td>HTTP Proxy for apt mirror access</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables aptly mirror</td>
</tr>
<tr>
<td>hour</td>
<td>['string', 'integer']</td>
<td>Hour parameter in cron job for aptly mirror update</td>
</tr>
<tr>
<td>minute</td>
<td>['string', 'integer']</td>
<td>Minute parameter in cron job for aptly mirror update</td>
</tr>
<tr>
<td>no_config</td>
<td>boolean</td>
<td>Start service without config</td>
</tr>
</tbody>
</table>

**Metadata schema specifications for aptly server**

Core properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpg_keypair_id</td>
<td>string</td>
<td>GPG keypair id</td>
</tr>
<tr>
<td>gpg_passphrase</td>
<td>string</td>
<td>Password phrase for GPG key</td>
</tr>
<tr>
<td>public_key</td>
<td>string</td>
<td>Public key to PGP repository</td>
</tr>
<tr>
<td>private_key</td>
<td>string</td>
<td>GPG Private key</td>
</tr>
<tr>
<td>homedir</td>
<td>string</td>
<td>GPG home directory</td>
</tr>
<tr>
<td>http_proxy</td>
<td>string</td>
<td>HTTP proxy to use for keys download</td>
</tr>
<tr>
<td>keyring</td>
<td>string</td>
<td>Keyring for GPG</td>
</tr>
<tr>
<td>keypair_id</td>
<td>string</td>
<td>GPG keypair id</td>
</tr>
<tr>
<td>keyphrase</td>
<td>string</td>
<td>Password phrase for GPG key</td>
</tr>
<tr>
<td>keyserver</td>
<td>string</td>
<td>GPG key server</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables aptly server</td>
</tr>
<tr>
<td>home_dir</td>
<td>string</td>
<td>Home directory for aptly system user</td>
</tr>
<tr>
<td>engine</td>
<td>string</td>
<td>Installation source for aptly publisher. Can be one of ['pkg', 'docker']</td>
</tr>
<tr>
<td>image</td>
<td>string</td>
<td>Publisher full image name. Set if ‘source’ is ‘docker’</td>
</tr>
<tr>
<td>pkgs</td>
<td>array</td>
<td>List of packages to be installed. Set if ‘source’ is ‘pkg’</td>
</tr>
<tr>
<td>registry</td>
<td>string</td>
<td>Registry host for publisher image. Set if ‘source’ is ‘docker’</td>
</tr>
<tr>
<td>gpg_private_key</td>
<td>string</td>
<td>GPG Private key</td>
</tr>
</tbody>
</table>

**_aptly_repo_object definition**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment</td>
<td>string</td>
<td>Comment for repo description</td>
</tr>
<tr>
<td>publisher</td>
<td>ERROR</td>
<td>description_notset For details, see: _aptly_mirror</td>
</tr>
<tr>
<td>distribution</td>
<td>string</td>
<td>OS distribution</td>
</tr>
<tr>
<td>component</td>
<td>string</td>
<td>Component type</td>
</tr>
<tr>
<td>architectures</td>
<td>ERROR</td>
<td>description_notset For details, see: _architectures definition</td>
</tr>
</tbody>
</table>

**_aptly_mirror|repo_publisher_object definition**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>string</td>
<td>Publisher’s component</td>
</tr>
</tbody>
</table>
A table of distributions for publisher:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>publisher</td>
<td>object</td>
<td>Parameters of publish mirror For details, see: _aptly_mirror</td>
</tr>
<tr>
<td>source</td>
<td>string</td>
<td>Source url for apt mirror</td>
</tr>
<tr>
<td>udebs</td>
<td>boolean</td>
<td>Download .udeb packages</td>
</tr>
<tr>
<td>filter</td>
<td>string</td>
<td>Filter for packages in mirror</td>
</tr>
<tr>
<td>sources</td>
<td>boolean</td>
<td>Download source packages in addition to binary packages</td>
</tr>
<tr>
<td>filter_with_deps</td>
<td>string</td>
<td>When filtering, include dependencies of matching packages as well</td>
</tr>
<tr>
<td>gpgkeys</td>
<td>string</td>
<td>GPG keys for apt mirror</td>
</tr>
<tr>
<td>architectures</td>
<td>ERROR</td>
<td>description_notset For details, see: _architectures definition</td>
</tr>
<tr>
<td>components</td>
<td>string</td>
<td>Component's types</td>
</tr>
<tr>
<td>distribution</td>
<td>string</td>
<td>OS distribution</td>
</tr>
</tbody>
</table>

A table of architectures:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_architecture</td>
<td>string</td>
<td>Packages architecture</td>
</tr>
</tbody>
</table>
CINDER

Usage

Cinder provides an infrastructure for managing volumes in OpenStack. Originally, this project was the Nova component called nova-volume and starting from the Folsom OpenStack release it has become an independent project.

This file provides the sample configurations for different use cases:

- Pillar sample of a basic Cinder configuration:

  The pillar structure defines cinder-api and cinder-scheduler inside the controller role and cinder-volume inside the to volume role.

```yaml
  cinder:
    controller:
      enabled: true
      version: juno
      cinder_uid: 304
      cinder_gid: 304
      nas_secure_file_permissions: false
      nas_secure_file_operations: false
      cinder_internal_tenant_user_id: f46924c112a14c80ab0a24a613d95eef
      cinder_internal_tenant_project_id: b7455b8974bb4064ad2478c8f375eae6c
      default_volume_type: 7k2SaS
      enable_force_upload: true
      availability_zone_fallback: True

    database:
      engine: mysql
      host: 127.0.0.1
      port: 3306
      name: cinder
      user: cinder
      password: pwd

    identity:
      engine: keystone
      host: 127.0.0.1
      port: 35357
      tenant: service
      user: cinder
      password: pwd

    message_queue:
      engine: rabbitmq
      host: 127.0.0.1
      port: 5672
      user: openstack
      password: pwd
      virtual_host: /openstack

    client:
```

©2020, Mirantis Inc.
connection_params:
    connect_retries: 50
    connect_retry_delay: 1
backend:
    7k2_SAS:
        engine: storwize
        type_name: slow-disks
        host: 192.168.0.1
        port: 22
        user: username
        password: pass
        connection: FC/iSCSI
        multipath: true
        pool: SAS7K2
audit:
    enabled: false
    osapi_max_limit: 500
    barbican:
        enabled: true
cinder:
    enabled: true
    version: juno
    cinder_uid: 304
    cinder_gid: 304
    nas_secure_file_permissions: false
    nas_secure_file_operations: false
    cinder_internal_tenant_user_id: f46924c112a14c80ab0a24a613d95eef
    cinder_internal_tenant_project_id: b7455b8974bb4064ad247c8f375ea6e6c
    default_volume_type: 7k2SaS
    enable_force_upload: true
    my_ip: 192.168.0.254
database:
    engine: mysql
    host: 127.0.0.1
    port: 3306
    name: cinder
    user: cinder
    password: pwd
identity:
    engine: keystone
    host: 127.0.0.1
    port: 35357
    tenant: service
    user: cinder
    password: pwd
message_queue:
  engine: rabbitmq
  host: 127.0.0.1
  port: 5672
  user: openstack
  password: pwd
  virtual_host: '/openstack'

backend:
  7k2_SAS:
    engine: storwize
    type_name: 7k2 SAS disk
    host: 192.168.0.1
    port: 22
    user: username
    password: pass
    connection: FC/iSCSI
    multihost: true
    multipath: true
    pool: SAS7K2
  audit:
    enabled: false
  barbican:
    enabled: true

Volume vmware related options:

cinder:
  volume:
    backend:
      vmware:
        engine: vmware
        host_username: vmware
        host_password: vmware
        cluster_names: vmware_cluster01,vmware_cluster02

• The CORS parameters enablement:

cinder:
  controller:
    cors:
      expose_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
      allow_methods: GET,PUT,POST,DELETE,PATCH
      allow_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
      allow_credentials: True
      max_age: 86400
• The client-side RabbitMQ HA setup for the controller:

```yaml
cinder:
  controller:
    ....
    message_queue:
      engine: rabbitmq
      members:
        - host: 10.0.16.1
        - host: 10.0.16.2
        - host: 10.0.16.3
      user: openstack
      password: pwd
      virtual_host: '/openstack'
    ....
```

• The client-side RabbitMQ HA setup for the volume component

```yaml
cinder:
  volume:
    ....
    message_queue:
      engine: rabbitmq
      members:
        - host: 10.0.16.1
        - host: 10.0.16.2
        - host: 10.0.16.3
      user: openstack
      password: pwd
      virtual_host: '/openstack'
    ....
```

• Configuring TLS communications.

Note

By default, system-wide installed CA certs are used. Therefore, the cacert_file and cacert parameters are optional.

• RabbitMQ TLS:

```yaml
cinder:
  controller, volume:
    message_queue:
      port: 5671
      ssl:
        enabled: True
    (optional) cacert: cert body if the cacert_file does not exists
```
(optional) cacert_file: /etc/openstack/rabbitmq-ca.pem
(optional) version: TLSv1_2

- MySQL TLS:

```
cinder:
  controller:
    database:
      ssl:
        enabled: True
        (optional) cacert: cert body if the cacert_file does not exists
        (optional) cacert_file: /etc/openstack/mysql-ca.pem
```

- Openstack HTTPS API:

```
cinder:
  controller, volume:
    identity:
      protocol: https
      (optional) cacert_file: /etc/openstack/proxy.pem
    glance:
      protocol: https
      (optional) cacert_file: /etc/openstack/proxy.pem
```

- Cinder setup with zeroing deleted volumes:

```
cinder:
  controller:
    enabled: true
    wipe_method: zero
...
```

- Cinder setup with shredding deleted volumes:

```
cinder:
  controller:
    enabled: true
    wipe_method: shred
...
```

- Configuration of policy.json file:

```
cinder:
  controller:
    ....
    policy:
      'volume:delete': 'rule:admin_or_owner'
```
# Add key without value to remove line from policy.json
'volume:extend':

- Default Cinder backend lvm_type setup:

```yaml
# Type of LVM volumes to deploy; (default, thin, or auto). Auto defaults to thin if thin is supported.
lvm_type: auto
```

- Default Cinder setup with iSCSI target:

```yaml
cinder:
  controller:
    enabled: true
    version: mitaka
  default_volume_type: lvmdriver-1
  database:
    engine: mysql
    host: 127.0.0.1
    port: 3306
    name: cinder
    user: cinder
    password: pwd
  identity:
    engine: keystone
    host: 127.0.0.1
    port: 35357
    tenant: service
    user: cinder
    password: pwd
  message_queue:
    engine: rabbitmq
    host: 127.0.0.1
    port: 5672
    user: openstack
    password: pwd
    virtual_host: '/openstack'
  backend:
    lvmdriver-1:
      engine: lvm
      type_name: lvmdriver-1
      volume_group: cinder-volume
```

- Cinder setup for IBM Storwize:
cinder:
  volume:
    enabled: true
  backend:
    7k2_SAS:
      engine: storwize
      type_name: 7k2 SAS disk
      host: 192.168.0.1
      port: 22
      user: username
      password: pass
      connection: FC/iSCSI
      multipath: true
    pool: SAS7K2
  10k_SAS:
    engine: storwize
    type_name: 10k SAS disk
    host: 192.168.0.1
    port: 22
    user: username
    password: pass
    connection: FC/iSCSI
    multipath: true
    pool: SAS10K
  15k_SAS:
    engine: storwize
    type_name: 15k SAS
    host: 192.168.0.1
    port: 22
    user: username
    password: pass
    connection: FC/iSCSI
    multipath: true
    pool: SAS15K

• Cinder setup with NFS:

cinder:
  controller:
    enabled: true
  default_volume_type: nfs-driver
  backend:
    nfs-driver:
      engine: nfs
      type_name: nfs-driver
**volume_group**: cinder-volume
**path**: /var/lib/cinder/nfs
**devices**:  
- 172.16.10.110:/var/nfs/cinder
**options**: rw,sync

- **Cinder setup with NetApp:**

```yaml
cinder:
  controller:
    backend:
      netapp:
        engine: netapp
        type_name: netapp
        user: openstack
        vserver: vm1
        server_hostname: 172.18.2.3
        password: password
        storage_protocol: nfs
        transport_type: https
        lun_space_reservation: enabled
        use_multipath_for_image_xfer: True
        nas_secure_file_operations: false
        nas_secure_file_permissions: false
        devices:
          - 172.18.1.2:/vol_1
          - 172.18.1.2:/vol_2
          - 172.18.1.2:/vol_3
          - 172.18.1.2:/vol_4

linux:
  system:
    package:
      nfs-common:
        version: latest
```

- **Cinder setup with Hitachi VPS:**

```yaml
cinder:
  controller:
    enabled: true
    backend:
      hus100_backend:
        type_name: HUS100
        backend: hus100_backend
        engine: hitachi_vsp
        connection: FC
```

- **Cinder setup with Hitachi VPS with defined ldev range:**

```yaml
```
• Cinder setup with Ceph:

```yaml
  cinder:
  controller:
    enabled: true
  backend:
    ceph_backend:
      type_name: standard-iops
      backend: ceph_backend
      backend_host: ceph
      pool: volumes
      engine: ceph
      user: cinder
      secret_uuid: da74ccb7-aa59-1721-a172-0006b1aa4e3e
      client_cinder_key: AQDOavlU6BsSJhAAnpFR906mvdgdfRqLHwu0Uw==
      report_discard_supported: True
      image_volume_cache_enabled: False
```

Note

Ceph official documentation

• Cinder setup with HP3par:

```yaml
  cinder:
  controller:
    enabled: true
  backend:
    hp3par_backend:
      type_name: hp3par
      backend: hp3par_backend
      user: hp3paruser
      password: something
      url: http://10.10.10.10/api/v1
```
cpg: OpenStackCPG  
host: 10.10.10.10  
login: hp3paradmin  
sanpassword: something  
debug: True  
snapcpg: OpenStackSNAPCPG

• Cinder setup with Fujitsu Eternus:

```yaml
cinder:  
  volume:  
    enabled: true  
    backend:  
      10kThinPro:  
        type_name: 10kThinPro  
        engine: fujitsu  
        pool: 10kThinPro  
        host: 192.168.0.1  
        port: 5988  
        user: username  
        password: pass  
        connection: FC/iSCSI  
        name: 10kThinPro  
      10k_SAS:  
        type_name: 10k_SAS  
        pool: SAS10K  
        engine: fujitsu  
        host: 192.168.0.1  
        port: 5988  
        user: username  
        password: pass  
        connection: FC/iSCSI  
        name: 10k_SAS
```

• Cinder setup with IBM GPFS filesystem:

```yaml
cinder:  
  volume:  
    enabled: true  
    backend:  
      GPFS-GOLD:  
        type_name: GPFS-GOLD  
        engine: gdfs  
        mount_point: '/mnt/gpfs-openstack/cinder/gold'  
      GPFS-SILVER:  
        type_name: GPFS-SILVER  
        engine: gdfs  
        mount_point: '/mnt/gpfs-openstack/cinder/silver'
```
• Cinder setup with HP LeftHand:

```yaml
cinder:
  volume:
    enabled: true
  backend:
    HP-LeftHand:
      type_name: normal-storage
      engine: hp_lefthand
      api_url: 'https://10.10.10.10:8081/lhos'
      username: user
      password: password
      clustername: cluster1
      iscsi_chap_enabled: false
```

• Extra parameters for HP LeftHand:

```bash
cinder type-key normal-storage set hplh:data_pl=r-10-2 hplh:provisioning=full
```

• Cinder setup with Solidfire:

```yaml
cinder:
  volume:
    enabled: true
  backend:
    solidfire:
      type_name: normal-storage
      engine: solidfire
      san_ip: 10.10.10.10
      san_login: user
      san_password: password
      clustername: cluster1
      sf_emulate_512: false
      sf_api_port: 14443
      host: ctl01
      #for compatibility with old versions
      sf_account_prefix: PREFIX
```

• Cinder setup with Block Device driver:

```yaml
cinder:
  volume:
    enabled: true
  backend:
    bdd:
      engine: bdd
      enabled: true
      type_name: bdd
```
• Enable cinder-backup service for ceph

```yaml
devices:
- sdb
- sdc
- sdd
```

• Auditing filter (CADF) enablement:

```yaml
cinder:
  controller:
    enabled: true
  version: mitaka
  backup:
    engine: ceph
    ceph_conf: "/etc/ceph/ceph.conf"
    ceph_pool: backup
    ceph_stripe_count: 0
    ceph_stripe_unit: 0
    ceph_user: cinder
    ceph_chunk_size: 134217728
    restore_discard_excess_bytes: false

  volume:
    enabled: true
    version: mitaka
    backup:
      engine: ceph
      ceph_conf: "/etc/ceph/ceph.conf"
      ceph_pool: backup
      ceph_stripe_count: 0
      ceph_stripe_unit: 0
      ceph_user: cinder
      ceph_chunk_size: 134217728
      restore_discard_excess_bytes: false
```

```yaml
cinder:
  controller:
    audit:
      enabled: true

      filter_factory: 'keystonemiddleware.audit:filter_factory'
      map_file: '/etc/pycadf/cinder_api_audit_map.conf'

  volume:
    audit:
      enabled: true
```

```yaml```
filter_factory: 'keystonemiddleware.audit:filter_factory'
map_file: '/etc/pycadf/cinder_api_audit_map.conf'

- Cinder setup with custom availability zones:

```yaml
cinder:
  controller:
    default_availability_zone: my-default-zone
    storage_availability_zone: my-custom-zone-name

cinder:
  volume:
    default_availability_zone: my-default-zone
    storage_availability_zone: my-custom-zone-name
```

The default_availability_zone is used when a volume has been created, without specifying a zone in the create request as this zone must exist in your configuration.

The storage_availability_zone is an actual zone where the node belongs to and must be specified per each node.

- Cinder setup with custom non-admin volume query filters:

```yaml
cinder:
  controller:
    query_volume_filters:
      - name
      - status
      - metadata
      - availability_zone
      - bootable
```

- public_endpoint and osapi_volume_base_url:

  - public_endpoint
    Used for configuring versions endpoint
  - osapi_volume_base_URL
    Used to present Cinder URL to users

These parameters can be useful when running Cinder under load balancer in SSL.

```yaml
cinder:
  controller:
    public_endpoint_address: https://${_param:cluster_domain}:8776
```

- Client role definition:

```yaml
cinder:
  client:
```
enabled: true
c
identity:
  host: 127.0.0.1
  port: 35357
  project: service
  user: cinder
  password: pwd
  protocol: http
  endpoint_type: internalURL
  region_name: RegionOne
  connection_params:
    connect_retries: 5
    connect_retry_delay: 1
  backend:
    ceph:
      type_name: standard-iops
      engine: ceph
      key:
        conn_speed: fibre-10G

• Barbican integration enablement:

  cinder:
    controller:
      barbican:
        enabled: true

• Keystone API version specification (v3 is default):

  cinder:
    controller:
      identity:
        api_version: v2.0

Enhanced logging with logging.conf

By default logging.conf is disabled. You can enable per-binary logging.conf by setting the following parameters:

• openstack_log_appender
  Set to true to enable log_config_append for all OpenStack services

• openstack_fluentd_handler_enabled
  Set to true to enable FluentHandler for all OpenStack services

• openstack_ossyslog_handler_enabled
  Set to true to enable OSSysLogHandler for all Openstack services

Only WatchedFileHandler, OSSysLogHandler, and FluentHandler are available.

To configure this functionality with pillar:
Enable x509 and ssl communication between Cinder and Galera cluster

By default communication between Cinder and Galera is unsecure.

You can set custom certificates in pillar:

cinder:
  controller:
    database:
      x509:
        enabled: True
  volume:
    database:
      x509:
        enabled: True

cinder:
  controller:
    database:
      x509:
        cacert: (certificate content)
        cert: (certificate content)
        key: (certificate content)
  volume:
    database:
For more details, see: OpenStack documentation.

Cinder service on compute node with memcached caching and security strategy:

```
cinder:
  volume:
    enabled: true
  ...
  cache:
    engine: memcached
    members:
    - host: 127.0.0.1
      port: 11211
    - host: 127.0.0.1
      port: 11211
    security:
      enabled: true
      strategy: ENCRYPT
      secret_key: secret
```

Cinder service on controller node with memcached caching and security strategy:

```
cinder:
  controller:
    enabled: true
  ...
  cache:
    engine: memcached
    members:
    - host: 127.0.0.1
      port: 11211
    - host: 127.0.0.1
      port: 11211
    security:
      enabled: true
      strategy: ENCRYPT
      secret_key: secret
```

Cinder service to define iscsi_helper for lvm backend:

```
cinder:
  volume:
  ...
```
backend:
  lvm:
    ...
    engine: lvm
    iscsi_helper: tgtadm

Cinder service to define scheduler_default_filters and which filter class names to use for filtering hosts when not specified in the request:

```yaml
  cinder:
    volume:
      ...
      scheduler_default_filters: (filters)
```

```yaml
  cinder:
    controller:
      ...
      scheduler_default_filters: (filters)
```

Upgrades

Each OpenStack formula provides a set of phases (logical blocks) that help to build a flexible upgrade orchestration logic for particular components. The table below lists the phases and their descriptions:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;app&gt;.upgrade.service_running</code></td>
<td>Ensure that all services for particular application are enabled for autostart and running</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.service_stopped</code></td>
<td>Ensure that all services for particular application disabled for autostart and dead</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.pkgs_latest</code></td>
<td>Ensure that packages used by particular application are installed to latest available version. This will not upgrade data plane packages like qemu and openvswitch as usually 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. Applying this state will not autostart service.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.render_config</code></td>
<td>Ensure configuration is rendered actual version.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.pre</code></td>
<td>We assume this state is applied on all nodes in the cloud before running upgrade. Only non destructive actions will be applied during this phase. Perform service built in service check like (keystone-manage doctor and nova-status upgrade)</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.pre</code></td>
<td>Mostly applicable for data plane nodes. During this phase resources will be gracefully removed from current node if it is allowed. Services for upgraded application will be set to admin disabled state to make sure node will not participate in resources scheduling. For example on gtw nodes this will set all agents to admin disable state and will move all routers to other agents.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade</code></td>
<td>This state will basically upgrade application on particular target. Stop services, render configuration, install new packages, run offline dbsync (for ctl), start services. Data plane should not be affected, only OpenStack Python services.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.post</code></td>
<td>Add services back to scheduling.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.post</code></td>
<td>This phase should be launched only when upgrade of the cloud is completed. Cleanup temporary files, perform other post upgrade tasks.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.verify</code></td>
<td>Here we will do basic health checks (API CRUD operations, verify do not have dead network agents/compute services)</td>
</tr>
</tbody>
</table>
DOCKER

Usage
Docker is a platform for developers and system administrators for developing, shipping, and running applications. Docker enables you to quickly assemble applications from components and eliminates the friction that can come when shipping the code. Also, with Docker, you get your code tested and deployed into production as fast as possible.

This file provides the sample configurations for different use cases.

Docker host configuration samples

- Docker host sample pillar configuration:

```yaml
docker:
  host:
    enabled: true
    options:
      bip: 172.31.255.1/16
      insecure-registries:
        - 127.0.0.1
        - 10.0.0.1
      log-driver: json-file
      log-opts:
        max-size: 50m
```

- Proxy configuration for Docker host:

```yaml
docker:
  host:
    proxy:
      enabled: true
    http: http://user:pass@proxy:3128
    https: http://user:pass@proxy:3128
    no_proxy:
      - localhost
      - 127.0.0.1
      - docker-registry
```

Docker Swarm configuration samples
Role can be master, manager, or worker. Master is the first manager that will initialize the swarm.

- Metadata for manager (the first node):
The token to join to the master node is obtained from grains using salt.mine. In case of any join_token undefined issues, verify that you have docker_swarm_grains available.

Docker client configuration samples

- Container:

```
docker:
  host:
    enabled: true
  swarm:
    role: worker
    master:
      host: 192.168.1.5
      port: 2377
```

- Docker compose:

```
The states providing this functionality include:

- docker.client.stack
```
• docker.client.compose

Stack is new and works with Docker Swarm Mode. Compose is legacy and works only if node is not a member of Swarm. Metadata for both states are similar and differs only in implementation.

• Stack:

```yaml
docker:
  client:
    stack:
      django_web:
        enabled: true
        update: true
        environment:
          SOMEVAR: somevalue
        version: "3.1"
      service:
        db:
          image: postgres
        web:
          image: djangoapp
        volumes:
          - /srv/volumes/django:/srv/django
        ports:
          - 8000:8000
        depends_on:
          - db
```

• Compose

You can install docker-compose using one of the following options:

• Distribution package (default)
• Using Pip
• Using Docker container

Install docker-compose using Docker (default is distribution package):

```bash
docker:
  client:
    compose:
      source:
        engine: docker
        image: docker/compose:1.8.0
    django_web:
      # Run up action, any positional argument to docker-compose CLI
      # If not defined, only docker-compose.yml is generated
      status: up
      # Run image pull every time state is run triggering container
      # restart in case it's changed
      pull: true
```
environment:
  SOMEVAR: somevalue
service:
  db:
    image: postgres
  web:
    image: djangoapp
volumes:
  - /srv/volumes/django:/srv/django
ports:
  - 8000:8000
depsends_on:
  - db

• Registry

docker:
  client:
    registry:
      target_registry: apt:5000
      image:
        - registry: docker
          name: compose:1.8.0
        - registry: tcpcloud
          name: jenkins:latest
        - registry: ""
          name: registry:2
      target_registry: myregistry

Docker Service configuration samples

To deploy service in Swarm mode, you can use docker.client.service:

parameters:
  docker:
    client:
      service:
        postgresql:
          environment:
            POSTGRES_USER: user
            POSTGRES_PASSWORD: password
            POSTGRES_DB: mydb
          restart:
            condition: on-failure
          image: "postgres:9.5"
          ports:
            - 5432:5432
          volume:
Docker Registry configuration samples

• Basic Docker Registry configuration:

```
data:
  type: bind
  source: /srv/volumes/postgresql/maas
  destination: /var/lib/postgresql/data
```

```
docker:
  registry:
    log:
      level: debug
      formatter: json
    cache:
      engine: redis
      host: localhost
    storage:
      engine: filesystem
      root: /srv/docker/registry
    bind:
      host: 0.0.0.0
      port: 5000
  hook:
    mail:
      levels:
        - panic
      # Options are rendered as yaml as is so use hook-specific options here
      options:
        smtp:
          addr: smtp.sendhost.com:25
          username: sendername
          password: password
          insecure: true
          from: name@sendhost.com
          to:
            - name@receivehost.com
```

• Docker login to private registry:

```
docker:
  host:
    enabled: true
  registry:
    first:
      address: private.docker.com
      user: username
```
password: password
second:
  address: private2.docker.com
  user: username2
  password: password2

Docker container service management configuration samples

• Start a service in a container:

```yaml
contrail_control_started:
dockerng_service.start:
  - container: f020d0d3efa8
  - service: contrail-control
```

or

```yaml
contrail_control_started:
dockerng_service.start:
  - container: contrail_controller
  - service: contrail-control
```

• Stop a service in a container:

```yaml
contrail_control_stoped:
dockerng_service.stop:
  - container: f020d0d3efa8
  - service: contrail-control
```

• Restart a service in a container:

```yaml
contrail_control_restart:
dockerng_service.restart:
  - container: f020d0d3efa8
  - service: contrail-control
```

• Enable a service in a container:

```yaml
contrail_control_enable:
dockerng_service.enable:
  - container: f020d0d3efa8
  - service: contrail-control
```

• Disable a service in a container:
contrail_control_disable:
dockerng_service_disable:
  - container: f020d0d3efa8
  - service: contrail-control

See also

- https://docs.docker.com/installation/ubuntu-linux/
- https://github.com/saltstack-formulas/docker-formula

**Metadata schema specifications for Docker client**

**Core properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>engine</td>
<td>string</td>
<td>Docker compose installation engine</td>
</tr>
<tr>
<td>image</td>
<td>string</td>
<td>Docker compose image</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>pkgs</td>
<td>array</td>
<td>List of Docker compose packages to be installed</td>
</tr>
<tr>
<td>base</td>
<td>string</td>
<td>base directory to store application compose files</td>
</tr>
<tr>
<td>container</td>
<td>object</td>
<td>Docker containers configuration</td>
</tr>
<tr>
<td>network</td>
<td>object</td>
<td>Docker networks configuration</td>
</tr>
<tr>
<td>service</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Docker client configuration</td>
</tr>
<tr>
<td>pkgs</td>
<td>array</td>
<td>List of Docker client packages to be installed</td>
</tr>
<tr>
<td>images</td>
<td>array</td>
<td>List of images to pull to the node</td>
</tr>
<tr>
<td>stack</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>target_registry</td>
<td>string</td>
<td>For details, see: _docker_service definition.</td>
</tr>
<tr>
<td>registry</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>target_registry</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------------------</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>pull</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>network</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>service</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>volume</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>environment</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>version</td>
<td>['number', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>user</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>config</td>
<td>object</td>
<td>description_notset</td>
</tr>
</tbody>
</table>

**Metadata schema specifications for Docker host**

**Core properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>string</td>
<td>docker service name</td>
</tr>
<tr>
<td>pkgs</td>
<td>array</td>
<td>List of Docker packages to be installed</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Docker host configuration</td>
</tr>
<tr>
<td>no_proxy</td>
<td>array</td>
<td>description_notset</td>
</tr>
<tr>
<td>http</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>https</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>insecure_registries</td>
<td>ERROR</td>
<td>description_notset</td>
</tr>
<tr>
<td>insecure_registries</td>
<td>ERROR</td>
<td>description_notset For details, see: _insecure_registries definition.</td>
</tr>
<tr>
<td>experimental</td>
<td>ERROR</td>
<td>description_notset For details, see: _experimental definition</td>
</tr>
<tr>
<td>experimental</td>
<td>ERROR</td>
<td>description_notset</td>
</tr>
<tr>
<td>registry</td>
<td>object</td>
<td>description_notset</td>
</tr>
</tbody>
</table>

_inperimental definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
_insecure_registries definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>insecure_registries</td>
<td>array</td>
<td>description_notset</td>
</tr>
</tbody>
</table>

**Metadata schema specifications for Docker registry**

Core properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatter</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>level</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>hooks</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>secret</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>engine</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>db</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>engine</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>root</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Docker registry configuration</td>
</tr>
<tr>
<td>pkgs</td>
<td>array</td>
<td>List of Docker registry packages to be installed</td>
</tr>
</tbody>
</table>

**Metadata schema specifications for Docker Swarm**

Core properties
<table>
<thead>
<tr>
<th>join_token</th>
<th>object</th>
<th>description_notset</th>
</tr>
</thead>
<tbody>
<tr>
<td>network</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>address</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>role</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Docker Swarm configuration</td>
</tr>
<tr>
<td>advertise_addr</td>
<td>string</td>
<td>description_notset</td>
</tr>
</tbody>
</table>
GALERA

Usage

Galera Cluster for MySQL is a true Multimaster Cluster based on synchronous replication. Galera Cluster is an easy-to-use, high-availability solution, which provides high system uptime, no data loss and scalability for future growth.

Sample pillars

Galera cluster master node

```yaml
galera:
  version:
    mysql: 5.6
    galera: 3
  master:
    enabled: true
    name: openstack
    bind:
      address: 192.168.0.1
      port: 3306
    members:
      - host: 192.168.0.1
        port: 4567
      - host: 192.168.0.2
        port: 4567
    admin:
      user: root
      password: pass
    sst:
      user: sstuser
      password: sstpassword
    database:
      name:
        encoding: 'utf8'
      users:
        name: 'username'
        password: 'password'
        host: 'localhost'
        rights: 'all privileges'
        database: '*,*'
```

Galera cluster slave node

```yaml
galera:
  slave:
    enabled: true
```

Mirantis Cloud Platform Salt Formulas Documentation
version latest

©2020, Mirantis Inc.
name: openstack
bind:
  address: 192.168.0.2
  port: 3306
members:
  - host: 192.168.0.1
    port: 4567
  - host: 192.168.0.2
    port: 4567
admin:
  user: root
  password: pass
sst:
  user: sstuser
  password: sstpassword

Enable TLS support:

galera:
  slave or master:
    ssl:
      enabled: True
ciphers:
    DHE-RSA-AES128-SHA:
      enabled: True
    DHE-RSA-AES256-SHA:
      enabled: True
    EDH-RSA-DES-CBC3-SHA:
      name: EDH-RSA-DES-CBC3-SHA
      enabled: True
    AES128-SHA:AES256-SHA:
      name: AES128-SHA:AES256-SHA
      enabled: True
    DES-CBC3-SHA:
      enabled: True
# path
cert_file: /etc/mysql/ssl/cert.pem
key_file: /etc/mysql/ssl/key.pem
ca_file: /etc/mysql/ssl/ca.pem

# content (not required if files already exists)
key: << body of key >>
cert: << body of cert >>
cacert_chain: << body of ca certs chain >>

Additional mysql users:
mysql:
server:
users:
  - name: clustercheck
    password: clustercheck
    database: '*.*'
    grants: PROCESS
  - name: inspector
    host: 127.0.0.1
    password: password
    databases:
      mydb:
        - database: mydb
        - table: mytable
        - grant_option: True
        - grants:
          - all privileges

Additional mysql SSL grants:

mysql:
server:
users:
  - name: clustercheck
    password: clustercheck
    database: '*.*'
    grants: PROCESS
    ssl_option:
      - SSL: True
      - X509: True
      - SUBJECT: <subject>
      - ISSUER: <issuer>
      - CIPHER: <cipher>

Additional check params:

galera:
clustercheck:
  - enabled: True
  - user: clustercheck
  - password: clustercheck
  - available_when_donor: 0
  - available_when_readonly: 1
  - port 9200

Configurable soft parameters
- `galera_innodb_buffer_pool_size`  
  Default is 3138M
- `galera_max_connections`  
  Default is 20000
- `galera_innodb_read_io_threads`  
  Default is 8
- `galera_innodb_write_io_threads`  
  Default is 8
- `galera_wsrep_slave_threads`  
  Default is 8
- `galera_xtrabackup_parallel`  
  Default is 4
- `galera_error_log_enabled`  
  Default is true
- `galera_error_log_path`  
  Default is /var/log/mysql/error.log

When the following parameters are set to 0, their defaults will be calculated automatically based on the number of CPU cores:

- `galera_innodb_read_io_threads`
- `galera_innodb_write_io_threads`
- `galera_wsrep_slave_threads`

Usage:

```
_param:
  galera_innodb_buffer_pool_size: 1024M
  galera_max_connections: 200
  galera_innodb_read_io_threads: 16
  galera_innodb_write_io_threads: 16
  galera_wsrep_slave_threads: 8
  galera_xtrabackup_parallel: 2
  galera_error_log_enabled: true
  galera_error_log_path: /var/log/mysql/error.log
```

Usage

MySQL Galera check scripts

```bash
mysql> SHOW STATUS LIKE 'wsrep%';
mysql> SHOW STATUS LIKE 'wsrep_cluster_size';
```

Galera monitoring command, performed from extra server
garbd -a gcomm://ipaddrofone:4567 -g my_wsrep_cluster -l /tmp/1.out -d

1. salt-call state.sls mysql
2. Comment everything starting wsrep* (wsrep_provider, wsrep_cluster, wsrep_sst)
3. Service mysql start
4. Run on each node mysql_secure_install and filling root password.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!

By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
... skipping.

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

5. Service mysql stop
6. Uncomment all wsrep* lines except first server, where leave only in my.cnf
   wsrep_cluster_address='gcomm://';
7. Start first node
8. Start third node which is connected to first one
9. Start second node which is connected to third one
10. After starting cluster, it must be change cluster address at first starting node without restart
    database and change config my.cnf.

  mysql> SET GLOBAL wsrep_cluster_address='gcomm://10.0.0.2';

Read more

- https://github.com/CaptTofu/ansible-galera
GERRIT

Usage
Gerrit provides web based code review and repository management for the Git version control system.

Sample pillars
Simple gerrit service

```yaml
  gerrit:
    server:
      enabled: true
      source:
        engine: http
        hash: 2e17064b8742c4622815593ec496c571
```

Full service setup

```yaml
  gerrit:
    server:
      enabled: true
      canonical_web_url: http://10.10.10.148:8082/
      email_private_key: 
      token_private_key: 
      initial_user:
        full_name: John Doe
        email: mail@jdoe.com
        username: jdoe
      plugin:
        download-commands:
          engine: gerrit
        replication:
          engine: gerrit
        reviewnotes:
          engine: gerrit
        singleusergroup:
          engine: gerrit
        ssh_rsa_key:
          -----BEGIN RSA PRIVATE KEY-----
          MIIEowIBAAKCAQEAs0Y8mxS3dfs5zG8Du5vdBkfOCOng1IEUmFZIirJ8oBgJOd54QgmkDFB7oP9eTCgz9k/rix1uJWhhVCMBzrWzH5IODO+tyy/tK66pv2BWtVfTDhBA
          nShOLDNbSIBaV8E/NcrbnQN+b0alp4N7rQnavkOYl+JQncKjz1csmCodirscB9Ojrdo6NG9olv9IQd/tDQxEeDyQkoW50aCEWcq7o+QaTzgnlrL+XZEzhzjdcvA9m8go...
          jvMXms60iD/A5OpG33LWHNNzQBP486SxG75LB+Xs5sp5j2/b7VF5LJLhpGiJv9Mk...
          -----END RSA PRIVATE KEY-----
          ssh_rsa_key_pub: ssh-rsa ...
          email: mail@domain.com
          auth:
            engine: HTTP
          source:
            engine: http
            address: https://gerrit-releases.storage.googleapis.com/gerrit-2.12.4.war
            hash: sha256=45786a920a929c6258de6461bcf03ddec8925577bd485905f102ceb6e5e1e47c
            receive_timeout: 5min
          sshd:
            threads: 64
            batch_threads: 16
            max_connections_per_user: 64
          database:
            engine: postgresql
            host: localhost
            port: 5432
            name: gerrit
            user: gerrit
            password: ${_param:postgresql_gerrit_password}
            pool_limit: 250
            pool_max_idle: 16
```

Gerrit LDAP authentification

```yaml
  gerrit:
    server:
      auth:
        engine: LDAP
        ldap_server: ldap://ldap.mycompany.net
        ldap_account_base: dc=company,dc=net
        ldap_group_base: ou=Groups,dc=company,dc=net
        ldap_account_pattern: uid=${username}
        ldap_group_pattern: (cn=${groupname})
        ldap_group_query: true
        ldap_group_member_pattern: (memberUid=${username})
```

Gerrit change auto abandon
Gerrit client enforcing groups

```yaml
gerrit:
    client:
        group:
            Admin001:
                description: admin 01
            Admin002:
                description: admin 02
```

Gerrit client enforcing users, install using pip

```yaml
gage:
    client:
        source:
            engine: pip
        user:
            jdoe:
                fullname: John Doe
                email: "jdoe@domain.com"
                ssh_key: ssh-rsa
                http_password: password
                groups:
                    - Admin001
```

Gerrit client enforcing projects

```yaml
gage:
    client:
        enabled: True
        server:
            host: 10.10.10.148
            user: newt
            key: |
                -----BEGIN RSA PRIVATE KEY-----
                MIIEowIBAAKCAQEAs0Y8mxS3dfs5zG8Du5vdBkfOCOnq11EUmFZlirJ8oBgLJ0d54
                OgmkDFB7o9eTCg9k/rix1uJWhhVCMBzrWzH5IODO+tyy/tK66pv2BwTvFTDhBA
                ...
                l1UrxQKBgEklBTuEiDRibKGXQBlAYvK2He09hWqptpt9/DVeI6s4A1bbTWDHyop
                jvMXms60ID/A5OpG33LWfNnzQBP4865xG75LB+Xs5sp5j2/b7VF5LJLhpGijv9Mk
                ydbuy8iuuvali2uf133kALLqnrWfTTYQQI1OfW5gl0v1L6kv94dU
                -----END RSA PRIVATE KEY-----
```
email: "Project Creator <infra@lists.domain.com>"
project:
  test_salt_project:
    enabled: true

Gerrit client enforcing project, full project example

```yaml
gerrit:
  client: True
  project:
    test_salt_project:
      enabled: true
      access:
        "refs/heads/*":
          actions:
            - name: abandon
c              group: openstack-salt-core
            - name: create
c              group: openstack-salt-release
          labels:
            - name: Code-Review
c              group: openstack-salt-core
c              score: -2..+2
            - name: Workflow
c              group: openstack-salt-core
c              score: -1..+1
        "refs/tags/*":
          actions:
            - name: pushSignedTag
c              group: openstack-salt-release
c              force: true
          inherit_access: All-Projects
          require_change_id: true
          require_agreement: true
          merge_content: true
          action: "fast forward only"
```

gerrit:
  client: True
  group:
    groupname:
      enabled: true
      members:
        - username
  account:
username:
  enabled: true
  full_name: User Example
  email: mail@newt.cz
  public_key: rsa
  http_password: passwd

Gerrit client proxy

gerrit:
  client:
    proxy:
      http_proxy: http://192.168.10.15:8000
      https_proxy: http://192.168.10.15:8000
      no_proxy: 192.168.10.90

Sample project access

[access "refs/**"]
  read = group Administrators
  read = group Anonymous Users
[access "refs/for/refs/**"]
  push = group Registered Users
  pushMerge = group Registered Users
[access "refs/heads/**"]
  create = group Administrators
  create = group Project Owners
  forgeAuthor = group Registered Users
  forgeCommitter = group Administrators
  forgeCommitter = group Project Owners
  push = group Administrators
  push = group Project Owners
  label-Code-Review = -2..+2 group Administrators
  label-Code-Review = -2..+2 group Project Owners
  label-Verified = -1..+1 group Non-Interactive Users
  submit = group Administrators
  submit = group Project Owners
  editTopicName = +force group Administrators
  editTopicName = +force group Project Owners
[access "refs/meta/config"]
  exclusiveGroupPermissions = read
  read = group Administrators
  read = group Project Owners
  push = group Administrators
  push = group Project Owners
  label-Code-Review = -2..+2 group Administrators
  label-Code-Review = -2..+2 group Project Owners
submit = group Administrators
submit = group Project Owners
[access "refs/tags/*"]
pushTag = group Administrators
pushTag = group Project Owners
pushSignedTag = +force group Administrators
pushSignedTag = group Project Owners
[label "Code-Review"]
function = MaxWithBlock
copyMinScore = true
value = -2 This shall not be merged
value = -1 I would prefer this is not merged as is
value = 0 No score
value = +1 Looks good to me, but someone else must approve
value = +2 Looks good to me, approved
[label "Verified"]
function = MaxWithBlock
copyMinScore = true
value = -1 Fails
value = 0 No score
value = +1 Verified

Gerrit replication enable

gerrit:
  server:
    plugin:
      replication:
        engine: gerrit
    replication:
      gerrit2.localdomain:
        remote_url: user@gerrit2.local.domain:/var/lib/gerrit
        remote_port: 22
        replication_user: gerrit2

For creating ssh keys use openssh state

Gerrit hide CI

gerrit:
  server:
    hideci:
      ci_user_name: ci_user

Read more

- [https://www.gerritcodereview.com/](https://www.gerritcodereview.com/)
• https://gerrit-review.googlesource.com/Documentation/
• https://github.com/openstack-infra/puppet-gerrit/
• https://gerrit-ci.gerritforge.com/
• https://github.com/morucci/exzuul
GLANCE

Usage

The Glance project provides services for discovering, registering, and retrieving virtual machine images. Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image.

Sample pillars

```yaml
glance:
  server:
    enabled: true
    version: juno
    workers: 8
  glance_uid: 302
  glance_gid: 302
  policy:
    publicize_image:
      - "role:admin"
      - "role:image_manager"
  database:
    engine: mysql
    host: 127.0.0.1
    port: 3306
    name: glance
    user: glance
    password: pwd
  identity:
    engine: keystone
    host: 127.0.0.1
    port: 35357
    tenant: service
    user: glance
    password: pwd
  message_queue:
    engine: rabbitmq
    host: 127.0.0.1
    port: 5672
    user: openstack
    password: pwd
    virtual_host: '/openstack'
  storage:
    engine: file
    images:
      - name: "CirrOS 0.3.1"
        format: qcow2
        file: cirros-0.3.1-x86_64-disk.img
```
**source**: http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img  
**public**: true  
**audit**:  
  **enabled**: false  
**api_limit_max**: 100  
**limit_param_default**: 50  
**barbican**:  
  **enabled**: true

The pagination is controlled by the `api_limit_max` and `limit_param_default` parameters as shown above:

- **api_limit_max**  
  Defines the maximum number of records that the server will return.

- **limit_param_default**  
  The default limit parameter that applies if the request didn’t define it explicitly.

Configuration of the policy.json file:

```

glance:
  server:
    ....
    policy:
      publicize_image: "role:admin"
      # Add key without value to remove line from policy.json
      add_member:
```

Keystone and cinder region

```

glance:
  server:
    enabled: true
    version: kilo
    ...
    identity:
      engine: keystone
      host: 127.0.0.1
      region: RegionTwo
      ...
```

Ceph integration glance

```

glance:
  server:
    enabled: true
    version: juno
    storage:
```
**engine**: rbd, http
**user**: glance
**pool**: images
**chunk_size**: 8
**client_glance_key**: AQDOavlU6BsSjAAOnpFR906mvpgdfRqLHu0Uw==

VMWare integration:

```yaml
glance:
server:
    storage:
        **engine**: vmware
        **default_store**: vsphere
        **vmware**:
            enabled: true
            **server_host**: 1.2.3.4
            **server_username**: vmware_username
            **server_password**: vmware_password
        **datstores**:
            **data1**:
                name: datastore_name1
                enabled: true
                **path**: datacenter_name
                **weight**: 10
            **data2**:
                name: datastore_name2
                enabled: true
                **path**: datacenter_name
```

RabbitMQ HA setup

```yaml
glance:
server:
    ....
    message_queue:
        **engine**: rabbitmq
        **members**:
            - host: 10.0.16.1
            - host: 10.0.16.2
            - host: 10.0.16.3
        **user**: openstack
        **password**: pwd
        virtual_host: '/openstack'
    ....
```

Quota Options
**glance:**

```yaml
server:

....
quota:
  image_member: -1
  image_property: 256
  image_tag: 256
  image_location: 15
  user_storage: 0
  ....
```

Configuring TLS communications

**Note**

By default, system wide installed CA certs are used, so cacert_file param is optional, as well as cacert.

- **RabbitMQ TLS**

```yaml
glance:
server:
  message_queue:
    port: 5671
    ssl:
      enabled: True
      (optional) cacert: cert body if the cacert_file does not exists
      (optional) cacert_file: /etc/openstack/rabbitmq-ca.pem
      (optional) version: TLSv1_2
```

- **MySQL TLS**

```yaml
glance:
server:
  database:
    ssl:
      enabled: True
      (optional) cacert: cert body if the cacert_file does not exists
      (optional) cacert_file: /etc/openstack/mysql-ca.pem
```

- **Openstack HTTPS API**

Set the https as protocol at glance:server sections:
glance:
  server:
    identity:
      protocol: https
      (optional) cacert_file: /etc/openstack/proxy.pem
    registry:
      protocol: https
      (optional) cacert_file: /etc/openstack/proxy.pem
    storage:
      engine: cinder, swift
      cinder:
        protocol: https
        (optional) cacert_file: /etc/openstack/proxy.pem
      swift:
        store:
          (optional) cafile: /etc/openstack/proxy.pem

Enable Glance Image Cache:

```yaml
glance:
  server:
    image_cache:
      enabled: true
      enable_management: true
      directory: /var/lib/glance/image-cache/
      max_size: 21474836480
```

Enable auditing filter (CADF):

```yaml
glance:
  server:
    audit:
      enabled: true

  filter_factory: 'keystonemiddleware.audit:filter_factory'
  map_file: '/etc/pycadf/glance_api_audit_map.conf'
```

Swift integration glance

```yaml
glance:
  server:
    enabled: true
    version: mitaka
    storage:
      engine: swift,http
```
Another way, which also supports multiple swift backends, can be configured like this:

```
swift:
store:
  auth:
    address: http://keystone.example.com:5000/v2.0
    version: 2
  endpoint_type: publicURL
  container: glance
  create_container_on_put: true
  retry_get_count: 5
  user: 2ec7966596504f59acc3a76b3b9d9291:glance-user
  key: someRandomPassword

glance:
server:
  enabled: true
  version: mitaka
storage:
  engine: swift,http
swift:
  store:
    endpoint_type: publicURL
    container: glance
    create_container_on_put: true
    retry_get_count: 5
    references:
      my_objectstore_reference_1:
        auth:
          address: http://keystone.example.com:5000/v2.0
          version: 2
          user: 2ec7966596504f59acc3a76b3b9d9291:glance-user
          key: someRandomPassword

Enable CORS parameters:

```

```
  expose_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
  allow_methods: GET,PUT,POST,DELETE,PATCH
  allow_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
  allow_credentials: True
  max_age: 86400
```

Enable Viewing Multiple Locations
If you want to expose all locations available (for example when you have multiple backends configured), then you can configure this like so:

```
glance:
  server:
    show_multiple_locations: True
    location_strategy: store_type
    store_type_preference: rbd,swift,file
```

Note
The show_multiple_locations option is deprecated since Newton and is planned to be handled by policy files only starting with the Pike release.

This feature is convenient in a scenario when you have swift and rbd configured and want to benefit from rbd enhancements.

Barbican integration glance

```
glance:
  server:
    barbican:
      enabled: true
```

Adding cron-job

```
glance:
  server:
    cron:
      cache_pruner:
        special_period: '@daily'
      cache_cleaner:
        hour: '5'
        minute: '30'
        daymonth: '*/2'
```

Image cache settings

```
glance:
  server:
    image_cache:
      max_size: 10737418240
      stall_time: 86400
      directory: '/var/lib/glance/image-cache/
```
Client role

Glance images

```yaml
glance:
  client:
    enabled: true
  server:
    profile_admin:
      image:
        cirros-test:
          visibility: public
          protected: false
          location: http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-i386-disk.img
```

Enhanced logging with logging.conf

By default logging.conf is disabled.

That is possible to enable per-binary logging.conf with new variables:

- `openstack_log_appender`
  Set to true to enable log_config_append for all OpenStack services
- `openstack_fluentd_handler_enabled`
  Set to true to enable FluentHandler for all Openstack services
- `openstack_ossyslog_handler_enabled`
  Set to true to enable OSSysLogHandler for all Openstack services

Only WatchedFileHandler, OSSysLogHandler, and FluentHandler are available.

Also, it is possible to configure this with pillar:

```yaml
glance:
  server:
    logging:
      log_appender: true
      log_handlers:
        watchedfile:
          enabled: true
        fluentd:
          enabled: true
        ossyslog:
          enabled: true
```

Enable x509 and ssl communication between Glance and Galera cluster

By default, communication between Glance and Galera is unsecure:
You can set custom certificates in pillar:

```yaml
glance:
  server:
    database:
      x509:
        enabled: True
```

You can read more about it here [https://docs.openstack.org/security-guide/databases/database-access-control.html](https://docs.openstack.org/security-guide/databases/database-access-control.html)

Glance services on controller node with memcached caching and security strategy:

```yaml
glance:
  server:
    enabled: true
    ...
  cache:
    engine: memcached
    members:
      - host: 127.0.0.1
        port: 11211
      - host: 127.0.0.1
        port: 11211
    security:
      enabled: true
      strategy: ENCRYPT
      secret_key: secret
```

Show all image locations when returning an image. This configuration option indicates whether to show all the image locations when returning image details to the user.

```yaml
glance:
  server:
    enabled: true
    ...
    show_multiple_locations: True
```

Usage
1. Import new public image:

   `glance image-create --name 'Windows 7 x86_64' --is-public true --container-format bare --disk-format qcow2 < ./win7.qcow2`

2. Change new image’s disk properties

   `glance image-update "Windows 7 x86_64" --property hw_disk_bus=ide`

3. Change new image’s NIC properties

   `glance image-update "Windows 7 x86_64" --property hw_vif_model=rtl8139`

**Upgrades**

Each OpenStack formula provides a set of phases (logical blocks) that help to build a flexible upgrade orchestration logic for particular components. The table below lists the phases and their descriptions:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;app&gt;.upgrade.service_running</td>
<td>Ensure that all services for particular application are enabled for autostart and running</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.service_stopped</td>
<td>Ensure that all services for particular application disabled for autostart and dead</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pkgs_latest</td>
<td>Ensure that packages used by particular application are installed to latest available version. This will not upgrade data plane packages like qemu and openvswitch as usually 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. Applying this state will not autostart service.</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.render_config</td>
<td>Ensure configuration is rendered actual version.</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pre</td>
<td>We assume this state is applied on all nodes in the cloud before running upgrade. Only non destructive actions will be applied during this phase. Perform service built in service check like (keystone-manage doctor and nova-status upgrade)</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.upgrade.pre</td>
<td>Mostly applicable for data plane nodes. During this phase resources will be gracefully removed from current node if it is allowed. Services for upgraded application will be set to admin disabled state to make sure node will not participate in resources scheduling. For example on gtw nodes this will set all agents to admin disable state and will move all routers to other agents.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade</code></td>
<td>This state will basically upgrade application on particular target. Stop services, render configuration, install new packages, run offline dbsync (for ctl), start services. Data plane should not be affected, only OpenStack Python services.</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.post</code></td>
<td>Add services back to scheduling.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.post</code></td>
<td>This phase should be launched only when upgrade of the cloud is completed. Cleanup temporary files, perform other post upgrade tasks.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.verify</code></td>
<td>Here we will do basic health checks (API CRUD operations, verify do not have dead network agents/compute services)</td>
</tr>
</tbody>
</table>

Read more

GLUSTERFS

Usage
Installs and configures GlusterFS server and client.

Available states

• glusterfs.server
  Sets up GlusterFS server (including both service and setup)

• glusterfs.server.service
  Sets up and start GlusterFS server service

• glusterfs.server.setup
  Sets up GlusterFS peers and volumes

• glusterfs.client
  Sets up GlusterFS client

Available metadata

• metadata.glusterfs.server
  Sets up basic server

• metadata.glusterfs.client
  Sets up client only

Example Reclass
Example for distributed Glance images storage where every control node is gluster peer.

```yaml
classes:
- service.glusterfs.server
- service.glusterfs.client

_param:
  cluster_node01_address: 192.168.1.21
  cluster_node02_address: 192.168.1.22
  cluster_node03_address: 192.168.1.23

parameters:
  glusterfs:
    server:
      peers:
        - ${param:cluster_node01_address}
        - ${param:cluster_node02_address}
        - ${param:cluster_node03_address}

  volumes:
```

©2020, Mirantis Inc.
glance:
  storage: /srv/glusterfs/glance
  replica: 3
  bricks:
    - ${_param:cluster_node01_address}:/srv/glusterfs/glance
    - ${_param:cluster_node02_address}:/srv/glusterfs/glance
    - ${_param:cluster_node03_address}:/srv/glusterfs/glance
  options:
    cluster.readdir-optimize: On
    nfs.disable: On
    network.remote-dio: On
    diagnostics.client-log-level: WARNING
    diagnostics.brick-log-level: WARNING
client:
  volumes:
    glance:
      path: /var/lib/glance/images
      server: ${_param:cluster_node01_address}
      user: glance
      group: glance

Example pillar

Server

  glusterfs:
    server:
      peers:
        - 192.168.1.21
        - 192.168.1.22
        - 192.168.1.23
      volumes:
        glance:
          storage: /srv/glusterfs/glance
          replica: 3
          bricks:
            - 172.168.1.21:/srv/glusterfs/glance
            - 172.168.1.21:/srv/glusterfs/glance
            - 172.168.1.21:/srv/glusterfs/glance
          enabled: true

Server with forced peer UUID (for peer recovery)

  glusterfs:
    server:
      recover_peers:
        kvm03.testserver.local:
Client

```
{
    enabled: true,
    uuid: ab6ac060-68f1-4f0b-8de4-70241dfb2279
}
```

glusterfs:
  client:
    volumes:
      glance:
        path: /var/lib/glance/images
        server: 192.168.1.21
        user: glance
        group: glance
        enabled: true

Read more

- [https://www.gluster.org/](https://www.gluster.org/)
HAPROXY

Usage
The reliable, high-performance TCP/HTTP load balancer.

Sample pillars
Simple admin listener:

```yaml
haproxy:
  proxy:
    enabled: True
    listen:
      admin_page:
        type: admin
        binds:
          - address: 0.0.0.0
            port: 8801
        user: fsdfdsfds
        password: dsfdsf
```

Simple stats listener:

```yaml
haproxy:
  proxy:
    enabled: True
    listen:
      admin_page:
        type: stats
        binds:
          - address: 0.0.0.0
            port: 8801
```

Sample pillar with admin:

```yaml
haproxy:
  proxy:
    enabled: True
    mode: http/tcp
    logging: syslog
    maxconn: 1024
    timeout:
      connect: 5000
      client: 50000
      server: 50000
    listen:
      https-in:
```
```yaml
binds:
- address: 0.0.0.0
  port: 443
servers:
- name: server1
  host: 10.0.0.1
  port: 8443
- name: server2
  host: 10.0.0.2
  port: 8443
params: 'maxconn 256'
```

Sample pillar with custom logging:

```yaml
haproxy:
proxy:
  enabled: True
  mode: http/tcp
  logging: syslog
  maxconn: 1024
  timeout:
    connect: 5000
    client: 50000
    server: 50000
listen:
  https-in:
    binds:
      address: 0.0.0.0
      port: 443
    servers:
      - name: server1
        host: 10.0.0.1
        port: 8443
      - name: server2
        host: 10.0.0.2
        port: 8443
    params: 'maxconn 256'
```

```yaml
haproxy:
proxy:
  enabled: true
  mode: tcp
  logging: syslog
  max_connections: 1024
listen:
  mysql:
    type: mysql
```
binds:
- **address**: 10.0.88.70
  - **port**: 3306

servers:
- **name**: node1
  - **host**: 10.0.88.13
  - **port**: 3306
  - **params**: check inter 15s fastinter 2s downinter 1s rise 5 fall 3
- **name**: node2
  - **host**: 10.0.88.14
  - **port**: 3306
  - **params**: check inter 15s fastinter 2s downinter 1s rise 5 fall 3 backup
- **name**: node3
  - **host**: 10.0.88.15
  - **port**: 3306
  - **params**: check inter 15s fastinter 2s downinter 1s rise 5 fall 3 backup

rabbitmq:
- **type**: rabbitmq

binds:
- **address**: 10.0.88.70
  - **port**: 5672

servers:
- **name**: node1
  - **host**: 10.0.88.13
  - **port**: 5673
  - **params**: check inter 5000 rise 2 fall 3
- **name**: node2
  - **host**: 10.0.88.14
  - **port**: 5673
  - **params**: check inter 5000 rise 2 fall 3 backup
- **name**: node3
  - **host**: 10.0.88.15
  - **port**: 5673
  - **params**: check inter 5000 rise 2 fall 3 backup

keystone-1:
- **type**: general-service

binds:
- **address**: 10.0.106.170
  - **port**: 5000

servers:
- **name**: node1
  - **host**: 10.0.88.13
  - **port**: 5000
  - **params**: check

haproxy:
- **proxy**:
  - **enabled**: true
mode: tcp
logging: syslog
max_connections: 1024
listen:
  mysql:
    type: mysql
    binds:
      - address: 10.0.88.70
        port: 3306
    servers:
      - name: node1
        host: 10.0.88.13
        port: 3306
        params: check inter 15s fastinter 2s downinter 1s rise 5 fall 3
      - name: node2
        host: 10.0.88.14
        port: 3306
        params: check inter 15s fastinter 2s downinter 1s rise 5 fall 3 backup
      - name: node3
        host: 10.0.88.15
        port: 3306
        params: check inter 15s fastinter 2s downinter 1s rise 5 fall 3 backup
  rabbitmq:
    type: rabbitmq
    binds:
      - address: 10.0.88.70
        port: 5672
    servers:
      - name: node1
        host: 10.0.88.13
        port: 5673
        params: check inter 5000 rise 2 fall 3
      - name: node2
        host: 10.0.88.14
        port: 5673
        params: check inter 5000 rise 2 fall 3 backup
      - name: node3
        host: 10.0.88.15
        port: 5673
        params: check inter 5000 rise 2 fall 3 backup
  keystone-1:
    type: general-service
    binds:
      - address: 10.0.106.170
        port: 5000
    servers:
      - name: node1
        host: 10.0.88.13
Sample pillar with port range and port offset:

This is useful in listen blocks for definition of multiple servers that differs only by port number in port range block. This situation can be result of multiple single-thread servers deployed in multi-core environment to better utilize the available cores.

For example, five contrail-api workers occupy ports 9100-9104. This can be achieved by using port_range_length in the pillar, port_range_length: 5 in this case. For skipping first worker (worker_id 0), because it has other responsibilities and to avoid overloading it by http requests use the port_range_start_offset in the pillar, port_range_start_offset: 1 in this case, it will only use ports 9101-9104 (skipping 9100).

- port_range_length parameter is used to calculate port range end
- port_range_start_offset will skip first n ports in port range

For backward compatibility, the name of the first server in port range has no pN suffix.

The sample will result in the following output:

```
listen contrail_api
  bind 172.16.10.252:8082
  balance leastconn
  server ntw01p1 172.16.10.95:9101 check inter 2000 rise 2 fall 3
  server ntw01p2 172.16.10.95:9102 check inter 2000 rise 2 fall 3
  server ntw01p3 172.16.10.95:9103 check inter 2000 rise 2 fall 3
  server ntw01p4 172.16.10.95:9104 check inter 2000 rise 2 fall 3
  server ntw02 172.16.10.96:9100 check inter 2000 rise 2 fall 3
  server ntw02p1 172.16.10.96:9101 check inter 2000 rise 2 fall 3
  server ntw02p2 172.16.10.96:9102 check inter 2000 rise 2 fall 3
  server ntw02p3 172.16.10.96:9103 check inter 2000 rise 2 fall 3
  server ntw02p4 172.16.10.96:9104 check inter 2000 rise 2 fall 3
  server ntw03 172.16.10.94:9100 check inter 2000 rise 2 fall 3
  server ntw03p1 172.16.10.94:9101 check inter 2000 rise 2 fall 3
  server ntw03p2 172.16.10.94:9102 check inter 2000 rise 2 fall 3
  server ntw03p3 172.16.10.94:9103 check inter 2000 rise 2 fall 3
  server ntw03p4 172.16.10.94:9104 check inter 2000 rise 2 fall 3
```

```
haproxy:
  proxy:
    listen:
      contrail_api:
        type: contrail-api
        service_name: contrail
        balance: leastconn
        binds:
```

```
- **address**: 10.10.10.10
  - **port**: 8082

**servers:**
- **name**: ntw01
  - **host**: 10.10.10.11
  - **port**: 9100
  - **port_range_length**: 5
  - **port_range_start_offset**: 1
  - **params**: check inter 2000 rise 2 fall 3
- **name**: ntw02
  - **host**: 10.10.10.12
  - **port**: 9100
  - **port_range_length**: 5
  - **port_range_start_offset**: 0
  - **params**: check inter 2000 rise 2 fall 3
- **name**: ntw03
  - **host**: 10.10.10.13
  - **port**: 9100
  - **port_range_length**: 5
  - **params**: check inter 2000 rise 2 fall 3

Sample pillar with a custom and more complex listener (for Artifactory and sub-domains for docker Registries):

```yaml
haproxy:
  proxy:
    listen:
      artifactory:
        mode: http
        options:
          - forwardfor
          - forwardfor header X-Real-IP
          - httpchk
          - httpclose
          - httplug
        sticks:
          - stick on src
          - stick-table type ip size 200k expire 2m
        acl:
          is_docker: "path_reg ^/v[12]/.*"
        http_request:
          - action: "set-path /artifactory/api/docker/%[req.hdr(host),lower,field(1,'.')]%[path]"
          - condition: "if is_docker"
        balance: source
        binds:
          - **address**: ${_param:cluster_vip_address}
            port: 8082
            ssl:
```

©2020, Mirantis Inc.
enabled: true
# This PEM file needs to contain key, cert, CA and possibly
# intermediate certificates
pem_file: /etc/haproxy/ssl/server.pem

servers:
- name: ${_param:cluster_node01_name}
  host: ${_param:cluster_node01_address}
  port: 8082
  params: check
- name: ${_param:cluster_node02_name}
  host: ${_param:cluster_node02_address}
  port: 8082
  params: backup check

You can also use multiple certificates for one listener, for example, when it is bind on multiple interfaces:

haproxy:
  proxy:
    listen:
      dummy_site:
        mode: http
        binds:
        - address: 127.0.0.1
          port: 8080
          ssl:
            enabled: true
            key: |
            my super secret key follows
            cert: |
            certificate
            chain: |
            CA chain (if any)
        - address: 127.0.1.1
          port: 8081
          ssl:
            enabled: true
            key: |
            my super secret key follows
            cert: |
            certificate
            chain: |
            CA chain (if any)

The definition above results in creation of /etc/haproxy/ssl/dummy_site directory with files 1-all.pem and 2-all.pem (per binds).

Sample pillar with a custom listener with HTTP-check options specified:
**haproxy:**

**proxy:**
- **enabled**: true

**forwardfor:**
- **enabled**: true
- **except**: 127.0.0.1
- **header**: X-Forwarded-For
- **if-none**: false

**listen:**
- **glance_api:**
  - **binds:**
    - **address**: 192.168.2.11
    - **port**: 9292
    - **ssl**:
      - **enabled**: true
      - **pem_file**: /etc/haproxy/ssl/all.pem
  - **http_request**:
    - **action**: set-header X-Forwarded-Proto https
  - **mode**: http
  - **options**:
    - httpchk GET /
    - httplog
    - httpclose
  - **servers**:
    - **host**: 127.0.0.1
    - **name**: ctl01
    - **params**: check inter 10s fastinter 2s downinter 3s rise 3 fall 3
    - **port**: 9292

Sample pillar with a custom listener with the tcp-check options specified (for Redis cluster with Sentinel):

**haproxy:**

**proxy:**
- **listen:**
  - **redis_cluster:**
    - **service_name**: redis
  - **health-check:**
    - **tcp**:
      - **enabled**: True
      - **options**:
        - 'send PING\r\n'
        - expect string +PONG
        - 'send info\ replication\r\n'
        - expect string role:master
        - 'send QUIT\r\n'
        - expect string +OK
  - **binds**:
Front-end for routing between exist listeners via URL with SSL and redirects. You can use one back end for several URLs.

```
- **address**: ${param:cluster_address}
  **port**: 6379

**servers**:
- **name**: ${param:cluster_node01_name}
  **host**: ${param:cluster_node01_address}
  **port**: 6379
  **params**: check inter 1s
- **name**: ${param:cluster_node02_name}
  **host**: ${param:cluster_node02_address}
  **port**: 6379
  **params**: check inter 1s
- **name**: ${param:cluster_node03_name}
  **host**: ${param:cluster_node03_address}
  **port**: 6379
  **params**: check inter 1s

**haproxy**:

**proxy**:

**listen**:

**service_proxy**:

- **mode**: http
- **balance**: source
- **format**: end

**binds**:
- **address**: ${param:haproxy_bind_address}
- **port**: 80
- **ssl**: ${param:haproxy_frontend_ssl}
- **ssl_port**: 443

**redirects**:
- **code**: 301
- **location**: domain.com/images

**conditions**:
- **type**: hdr_dom(host)
- **condition**: images.domain.com

**acls**:
- **name**: gerrit
  **conditions**:
  - **type**: hdr_dom(host)
    **condition**: gerrit.domain.com
- **name**: jenkins
  **conditions**:
  - **type**: hdr_dom(host)
    **condition**: jenkins.domain.com
- **name**: docker
  **conditions**:
  - **type**: hdr_dom(host)
    **condition**: docker.domain.com

**backend**: artifactroy
```
conditions:
- type: hdr_dom(host)
  condition: docker.domain.com

Enable customizable forwardfor option in the defaults section:

```
haproxy:
  proxy:
    enabled: true
    mode: tcp
    logging: syslog
    max_connections: 1024
    forwardfor:
      enabled: true
      except:
      header:  if-none: false
```

```
haproxy:
  proxy:
    enabled: true
    mode: tcp
    logging: syslog
    max_connections: 1024
    forwardfor:
      enabled: true
      except: 127.0.0.1
      header: X-Real-IP
      if-none: false
```

Sample pillar with multiprocess multicore configuration:

```
haproxy:
  proxy:
    enabled: True
    nbproc: 4
    cpu_map:  
      1: 0
      2: 1
      3: 2
      4: 3
    stats_bind_process: "1 2"
    mode: http/tcp
    logging: syslog
    maxconn: 1024
    timeout:  
      connect: 5000
```
client: 50000
server: 50000
listen:
  https-in:
    bind_process: "1 2 3 4"
    binds:
      - address: 0.0.0.0
        port: 443
    servers:
      - name: server1
        host: 10.0.0.1
        port: 8443
      - name: server2
        host: 10.0.0.2
        port: 8443
    params: 'maxconn 256'

Implement rate limiting, to prevent excessive requests. This feature only works if using format: end:

haproxy:
proxy:
  ...
listen:
  nova_metadata_api:
    ...
    options:
    - httpchk
    - httpclose
    - httplog
    rate_limit:
      duration: 900s
      enabled: true
      requests: 125
      track: content
    servers:
      ...
    type: http

Implement HAProxy configuration without specifying certain type or with type='None'. This approach allows you to set all major HAproxy parameters manually. Sample pillar:

haproxy:
proxy:
  listen:
    manila_api:
      type: None
mode: tcp
balance: roundrobin
timeout:
  check: 10
  client: 20
http_request:
  - action: "add-header X-Forwarded-Proto https"
  condition: "if { ssl_fc }"
options: ${_param:haproxy_https_check_options}
capture:
  - cookie ASPSESSION len 32
  - request header Host len 15
compression:
  - algo gzip
  - type text/html text/plain
declare_capture: request len 50
email_alert:
  - myhostname myserver
  - from server@localhost
  - level warning
errorfile:
  file_500:
    code: 500
    file: /tmp/error_500.log
  file_404:
    code: 400
    file: /tmp/error_400.log
max_keep_alive_queue: 100
maxconn: 10000
reqadd:
  - X-Proto: SSL if is-ssl
reqirep:
  - ^Host: www.mydomain.com Host:\ www
modify_headers:
  - reqallow ^Host:\ www\
  - reqdel ^Host:\ .*\.local
  - reqdeny ^Host:\ .*\.local
  - reqiallow ^Host:\ www\
  - reqidel ^Host:\ .*\.local
  - reqiarpit ^Host:\ .*\.local
  - reqtarpit ^Host:\ .*\.local
retries: 10
stats:
  - enable
  - auth admin1:AdMiN123
rate_limit_sessions: 1000
Implement rate limiting to prevent excessive requests using format: listen:

```yaml
haproxy:
  proxy:
    ...
    listen:
      nova_metadata_api:
        ...
        rate_limit:
          duration: 3s
          enabled: true
          requests: 60
          track: connection
          servers:
            ...
```

Implement rate limiting to prevent excessive requests using format: listen and acls/request/backend stick list:

```yaml
haproxy:
  proxy:
  listen:
    nova_metadata_api:
      options:
        - httplog
      rate_limit:
        enabled: true
        type: string
        len: 36
        size: 10m
        duration: 60s
      acls:
        101:
          enabled: true
          value: acl too_many_requests_3 sc0_gpc0_rate() gt 3
        102:
          enabled: true
          value: acl mark_seen sc0_inc_gpc0 gt 0
        110:
          enabled: true
          value: acl x_instance_id hdr(x-instance-id) -i 4777e8e0-16e8-46ce-a3fe-0a1ad9b3ebdc
        111:
          enabled: true
          value: acl x_instance_id hdr(x-instance-id) -i ca2395dd-f73f-4d43-8fe7-f7078a0920af
        201:
          enabled: true
          value: acl too_many_requests_6 sc0_gpc0_rate() gt 6
        202:
          enabled: true
          value: acl mark_seen sc0_inc_gpc0 gt 0
```
enabled: true
value: acl x_tenant_id hdr(x-tenant-id) -i 2b76cc56a437404bb8cb6cb20dbb0ea4

**tcp_request:**

001: 
  enabled: true  
  value: tcp-request inspect-delay 5s

101: 
  enabled: true
  value: tcp-request content track-sc0 hdr(x-instance-id) if ! too_many_requests_3

201: 
  enabled: true
  value: tcp-request content track-sc0 hdr(x-tenant-id) if ! too_many_requests_6

**use_backend:**

101: 
  enabled: true
  value: use_backend nova_metadata_api-rate_limit if mark_seen too_many_requests_3 x_instance_id

201: 
  enabled: true
  value: use_backend nova_metadata_api-rate_limit if mark_seen too_many_requests_6 x_tenant_id

---

**Read more**

- [https://gist.github.com/tomeduarte/6340205](https://gist.github.com/tomeduarte/6340205) - example on how to use a peer from within a config file (using Jinja)
- [http://youtu.be/jJJ8cfDjcTc?t=8m58s](http://youtu.be/jJJ8cfDjcTc?t=8m58s) - starting from the ninth minute, see an overview of a peer versus mine
- [https://github.com/russki/cluster-agents](https://github.com/russki/cluster-agents)
HEAT

Usage

Heat is the main project in the OpenStack Orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files that can be treated like code. A native Heat template format is evolving, but Heat also endeavors to provide compatibility with the AWS CloudFormation template format, so that many existing CloudFormation templates can be launched on OpenStack. Heat provides both an OpenStack-native ReST API and a CloudFormation-compatible Query API.

Sample pillars

Single Heat services on the controller node:

```yaml
heat:
  server:
    enabled: true
    version: icehouse
    region: RegionOne
    reauthentication_auth_method: trusts
    bind:
      metadata:
        address: 10.0.106.10
        port: 8000
        protocol: http
      waitcondition:
        address: 10.0.106.10
        port: 8000
        protocol: http
      watch:
        address: 10.0.106.10
        port: 8003
        protocol: http
      cloudwatch:
        host: 10.0.106.20
      api:
        host: 10.0.106.20
      api_cfn:
        host: 10.0.106.20
      database:
        engine: mysql
        host: 10.0.106.20
        port: 3306
        name: heat
        user: heat
        password: password
      identity:
```

©2020, Mirantis Inc.
Define server clients Keystone parameter:

```yaml
heat:
  server:
    clients:
      keystone:
        protocol: https
        host: 10.0.106.10
        port: 5000
        insecure: false
```

Server with auth_encryption_key defined:

```yaml
heat:
  server:
    ....
    auth_encryption_key: "KeyToEncrypt-hasToBeExact32Chars"
    ....
```

Enable CORS parameters:

```yaml
heat:
  server:
    cors:
      allowed_origin: https:localhost.local,http:localhost.local
      expose_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
      allow_methods: GET,PUT,POST,DELETE,PATCH
```
**allow_headers**: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token

**allow_credentials**: True

**max_age**: 86400

Heat client with specified Git templates:

```yaml
heat:
  client:
    enabled: true
  template:
    admin:
      domain: default
      source:
        engine: git
        address: git@repo.domain.com/admin-templates.git
        revision: master
    default:
      domain: default
      source:
        engine: git
        address: git@repo.domain.com/default-templates.git
        revision: master
```

Ceilometer notification:

```yaml
heat:
  server:
    enabled: true
    version: icehouse
    notification: true
```

Configuration of policy.json file:

```yaml
heat:
  server:
    ....
    policy:
      deny_stack_user: 'not role:heat_stack_user'
      'cloudformation:ValidateTemplate': 'rule:deny_stack_user'
      '# Add key without value to remove line from policy.json
      'cloudformation:DescribeStackResource':
```

Client-side RabbitMQ HA setup:

```yaml
heat:
  server:
    ....
```
message_queue:
  engine: rabbitmq
  members:
    - host: 10.0.16.1
    - host: 10.0.16.2
    - host: 10.0.16.3
  user: openstack
  password: pwd
  virtual_host: '/openstack'

Configuring TLS communications

Note
By default, system-wide installed CA certs are used, so the cacert_file and cacert parameters are optional.

• RabbitMQ TLS

```yaml
heat:
  server:
    message_queue:
      port: 5671
      ssl:
        enabled: True
        (optional) cacert: cert body if the cacert_file does not exists
        (optional) cacert_file: /etc/openstack/rabbitmq-ca.pem
        (optional) version: TLSv1_2
```

• MySQL TLS

```yaml
heat:
  server:
    database:
      ssl:
        enabled: True
        (optional) cacert: cert body if the cacert_file does not exists
        (optional) cacert_file: /etc/openstack/mysql-ca.pem
```

• Openstack HTTPS API

```yaml
heat:
  server:
    identity:
```
**protocol**: https
(\textit{optional]) \texttt{cacert\_file}: /etc/openstack/proxy.pem
\texttt{clients}:
\texttt{keystone}:
\texttt{protocol}: https
(\textit{optional}) \texttt{cacert\_file}: /etc/openstack/proxy.pem

Enhanced logging with \texttt{logging.conf}

By default \texttt{logging.conf} is disabled.

You can enable per-binary \texttt{logging.conf} with new variables:

- \texttt{openstack\_log\_appender}
  - Set to true to enable \texttt{log\_config\_append} for all OpenStack services
- \texttt{openstack\_fluentd\_handler\_enabled}
  - Set to true to enable FluentHandler for all Openstack services
- \texttt{openstack\_ossyslog\_handler\_enabled}
  - Set to true to enable OSSysLogHandler for all Openstack services

Only WatchedFileHandler, OSSysLogHandler, and FluentHandler are available.

Also, it is possible to configure this with pillar:

```
heat:
  server:
    logging:
      log_appender: true
      log_handlers:
        watchedfile:
          enabled: true
        fluentd:
          enabled: true
        ossyslog:
          enabled: true
```

Enable x509 and SSL communication between Heat and Galera cluster

By default communication between Heat and Galera is unsecure.

```
heat:
  server:
    database:
      x509:
        enabled: True
```

You can set custom certificates in pillar:
Heat services with Memcached caching and security strategy:

```
heat:
  server:
    database:
      x509:
        cacert: (certificate content)
        cert: (certificate content)
        key: (certificate content)

For more details, see: OpenStack documentation.
```

Upgrades

Each OpenStack formula provides a set of phases (logical blocks) that help to build a flexible upgrade orchestration logic for particular components. The table below lists the phases and their descriptions:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;app&gt;.upgrade.service_running</td>
<td>Ensure that all services for particular application are enabled for autostart and running</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.service_stopped</td>
<td>Ensure that all services for particular application disabled for autostart and dead</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pkgs_latest</td>
<td>Ensure that packages used by particular application are installed to latest available version. This will not upgrade data plane packages like qemu and openvswitch as usually 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. Applying this state will not autostart service.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.render_config</code></td>
<td>Ensure configuration is rendered actual version.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.pre</code></td>
<td>We assume this state is applied on all nodes in the cloud before running upgrade. Only non destructive actions will be applied during this phase. Perform service built in service check like (keystone-manage doctor and nova-status upgrade)</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.pre</code></td>
<td>Mostly applicable for data plane nodes. During this phase resources will be gracefully removed from current node if it is allowed. Services for upgraded application will be set to admin disabled state to make sure node will not participate in resources scheduling. For example on gtw nodes this will set all agents to admin disable state and will move all routers to other agents.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade</code></td>
<td>This state will basically upgrade application on particular target. Stop services, render configuration, install new packages, run offline dbsync (for ctl), start services. Data plane should not be affected, only OpenStack Python services.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.post</code></td>
<td>Add services back to scheduling.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.post</code></td>
<td>This phase should be launched only when upgrade of the cloud is completed. Cleanup temporary files, perform other post upgrade tasks.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.verify</code></td>
<td>Here we will do basic health checks (API CRUD operations, verify do not have dead network agents/compute services)</td>
</tr>
</tbody>
</table>
**HORIZON**

**Usage**
Horizon is the canonical implementation of OpenStack Dashboard, which provides a web-based user interface to OpenStack services including Nova, Swift, Keystone, etc.

**Sample pillars**
Simplest Horizon setup:

```yaml
horizon:
  server:
    enabled: true
    secret_key: secret
    host:
      name: cloud.lab.cz
  cache:
    engine: 'memcached'
    host: '127.0.0.1'
    port: 11211
    prefix: 'CACHE_HORIZON'
  api_versions:
    identity: 2
  identity:
    engine: 'keystone'
    host: '127.0.0.1'
    port: 5000
  mail:
    host: '127.0.0.1'
```

Multidomain setup for Horizon:

```yaml
horizon:
  server:
    enabled: true
    default_domain: MYDOMAIN
    multidomain: True
```

Simple branded Horizon:

```yaml
horizon:
  server:
    enabled: true
    branding: 'OpenStack Company Dashboard'
    default_dashboard: 'admin'
    help_url: 'http://doc.domain.com'
```
Horizon with policy files metadata. With source mine you can obtain real time policy file state from targeted node (OpenStack control node), provided you have policy file published to specified grain key. Source file will obtain static policy definition from formula files directory.

```yaml
horizon:
  server:
    enabled: true
  policy:
    identity:
      source: mine
      host: ctl01.my-domain.local
      name: keystone_policy.json
      grain_name: keystone_policy
      enabled: true
  compute:
    source: file
    name: nova_policy.json
    enabled: true
  network:
    source: file
    name: neutron_policy.json
    enabled: true
  image:
    source: file
    name: glance_policy.json
    enabled: true
  volume:
    source: file
    name: cinder_policy.json
    enabled: true
  telemetry:
    source: file
    name: ceilometer_policy.json
    enabled: true
  orchestration:
    source: file
    name: heat_policy.json
    enabled: true

Horizon with enabled SSL security (when SSL is realised by proxy):

```yaml
horizon:
  server:
    enabled: True
    secure: True

Horizon package setup with SSL:
Caution!

For the sake of backwards compatibility, the ssl_no_verify attribute defaults to true when horizon:server:identity:encryption is set to 'ssl'.

```yaml
horizon:
  server:
    enabled: true
    secret_key: MEGASECRET
    version: juno
    ssl_no_verify: false
    ssl:
      enabled: true
      authority: CA_Authority
  host:
    name: cloud.lab.cz
  cache:
    engine: 'memcached'
    host: '127.0.0.1'
    port: 11211
    prefix: 'CACHE_HORIZON'
  api_versions:
    identity: 2
    identity:
      engine: 'keystone'
      host: '127.0.0.1'
      port: 5000
    mail:
      host: '127.0.0.1'
```

Horizon with custom SESSION_ENGINE (default is signed_cookies, valid options are: signed_cookies, cache, file) and SESSION_TIMEOUT:

```yaml
horizon:
  server:
    enabled: True
    secure: True
    session:
      engine: 'cache'
      timeout: 43200
```

Multi-regional Horizon setup:

```yaml
horizon:
  server:
```

©2020, Mirantis Inc.
enabled: true
version: juno
secret_key: MEGASECRET
cache:
  engine: 'memcached'
  host: '127.0.0.1'
  port: 11211
  prefix: 'CACHE_HORIZON'
api_versions:
  identity: 2
  identity:
    engine: 'keystone'
    host: '127.0.0.1'
    port: 5000
mail:
  host: '127.0.0.1'
regions:
- name: cluster1
  address: http://cluster1.example.com:5000/v2.0
- name: cluster2
  address: http://cluster2.example.com:5000/v2.0

Configuration of LAUNCH_INSTANCE_DEFAULTS parameter:

horizon:
  server:
    launch_instance_defaults:
      config_drive: False
      enable_scheduler_hints: True
      disable_image: False
      disable_instance_snapshot: False
      disable_volume: False
      disable_volume_snapshot: False
      create_volume: False

Horizon setup with sensu plugin:

horizon:
  server:
    enabled: true
    version: juno
  sensu_api:
    host: localhost
    port: 4567
  plugin:
    monitoring:
      app: horizon_monitoring
      source:
type: git
dev: git@repo1.robotice.cz:django/horizon-monitoring.git
rev: develop

Sensu multi API:

```
horizon:
  server:
    enabled: true
    version: juno
    sensu_api:
      dc1:
        host: localhost
        port: 4567
      dc2:
        host: anotherhost
        port: 4567
```

Horizon setup with jenkins plugin:

```
horizon:
  server:
    enabled: true
    version: juno
    jenkins_api:
      url: https://localhost:8080
      user: admin
      password: pwd
      plugin:
        jenkins:
          app: horizon_jenkins
          source:
            type: pkg
```

Horizon setup with billometer plugin:

```
horizon:
  server:
    enabled: true
    version: juno
    billometer_api:
      host: localhost
      port: 9753
      api_version: 1
      plugin:
        billing:
          app: horizon_billing
```
source:
    type: git
    address: git@repo1.robotice.cz:django/horizon-billing.git
    rev: develop

Horizon setup with Contrail plugin:

horizon:
    server:
        enabled: true
        version: icehouse
        plugin:
            contrail:
                app: contrail_openstack_dashboard
                override: true
                source:
                    type: git
                    address: git@repo1.robotice.cz:django/horizon-contrail.git
                    rev: develop

Horizon setup with sentry log handler:

horizon:
    server:
        enabled: true
        version: juno
        ...  
        logging:
            engine: raven  
            dsn: http://pub:private@sentry1.test.cz/2

Multisite with Git source

Simple Horizon setup from Git repository:

horizon:
    server:
        enabled: true
        app:
            default:
                secret_key: MEGASECRET
                source:
                    engine: git
                    address: https://github.com/openstack/horizon.git
                    rev: stable/havana
                    cache:
                        engine: 'memcached'
host: '127.0.0.1'
port: 11211
prefix: 'CACHE_DEFAULT'

api_versions:
  identity: 2

identity:
  engine: 'keystone'
host: '127.0.0.1'
port: 5000

mail:
  host: '127.0.0.1'

Themed multisite setup:

horizon:
  server:
    enabled: true
  app:
    openstack1c:
      secret_key: MEGASECRET1
      source:
        engine: git
        address: https://github.com/openstack/horizon.git
        rev: stable/havana
      plugin:
        contrail:
          app: contrail_openstack_dashboard
          override: true
          source:
            type: git
            address: git@repo1.robotice.cz:django/horizon-contrail.git
            rev: develop
      theme:
        app: site1_theme
        source:
          type: git
          address: git@repo1.domain.com:django/horizon-site1-theme.git

    cache:
      engine: 'memcached'
      host: '127.0.0.1'
      port: 11211
      prefix: 'CACHE_SITE1'
      api_versions:
        identity: 2
        identity:
          engine: 'keystone'
          host: '127.0.0.1'
          port: 5000
mail:
  host: '127.0.0.1'

openstack2:
  secret_key: MEGASECRET2
  source:
    engine: git
    address: https://repo1.domain.com/openstack/horizon.git
    rev: stable/icehouse
  plugin:
    contrail:
      app: contrail_openstack_dashboard
      override: true
      source:
        type: git
        address: git@repo1.domain.com:django/horizon-contrail.git
        rev: develop
  monitoring:
    app: horizon_monitoring
    source:
      type: git
      address: git@domain.com:django/horizon-monitoring.git
      rev: develop
  theme:
    app: bootswatch_theme
    source:
      type: git
      address: git@repo1.robotice.cz:django/horizon-bootswatch-theme.git
      rev: develop
  cache:
    engine: 'memcached'
    host: '127.0.0.1'
    port: 11211
    prefix: 'CACHE_SITE2'
  api_versions:
    identity: 3
  identity:
    engine: 'keystone'
    host: '127.0.0.1'
    port: 5000
  mail:
    host: '127.0.0.1'

Set advanced theme options (for Horizon version OpenStack Mitaka and newer):

- Full example:

```yaml
horizon:
  server:
    themes:
```
• Minimal example:

```
horizon:
  server:
    theme:
      available:
        default:  # slug
          name: "Default"  # display name
          description: "Default style theme"
      material:
        name: "Material"
        description: "Google's Material Design style theme"
```

```
API versions override:

horizon:
  server:
    enabled: true
  app:
    openstack_api_override:
      secret_key: MEGASECRET1
      api_versions:
        identity: 3
        volume: 2
      source:
        engine: git
        address: https://github.com/openstack/horizon.git
        rev: stable/havana
```

```
Control dashboard behavior:

horizon:
  server:
    enabled: true
  app:
```
**openstack_dashboard_overrride:***

- **secret_key**: password
- **dashboards**:  
  - **settings**:  
    - **enabled**: true
  - **project**:  
    - **enabled**: false
    - **order**: 10
  - **admin**:  
    - **enabled**: false
    - **order**: 20
- **source**:  
  - **engine**: git
  - **address**: https://github.com/openstack/horizon.git
- **rev**: stable/juno

Enable WebSSO

Define a list of choices (supported choices are: oidc, saml2), credentials choice will be automatically appended and choice description is predefined.

WebSSO with credentials and saml2:

```
horizon:
  server:
    enabled: true
  websso:
    login_url: "WEBROOT + 'auth/login/'"
    logout_url: "WEBROOT + 'auth/logout/'"
    login_redirect_url: "WEBROOT + 'project/'"
    websso_choices:
      - saml2
```

Define a map of choices in the following format: 
`{"<choice_name>": {"description": "<choice_description>"}}`.

WebSSO with saml2 and credentials:

```
horizon:
  server:
    enabled: true
  websso:
    login_url: "WEBROOT + 'auth/login/'"
    logout_url: "WEBROOT + 'auth/logout/'"
    login_redirect_url: "WEBROOT + 'project/'"
    websso_choices:
      - saml2:
          description: "Security Assertion Markup Language"
```
credentials:
  description: "Keystone Credentials"

WebSSO with IDP mapping:

```
horizon:
  server:
    enabled: true
    websso:
      login_url: "WEBROOT + 'auth/login/'"
      logout_url: "WEBROOT + 'auth/logout/'"
      login_redirect_url: "WEBROOT + 'project/'"
    websso_choices:
      credentials:
        description: "Keystone Credentials"
      saml2:
        description: "Security Assertion Markup Language"
      oidc:
        description: "OpenID Connect"
      myidp_oidc:
        description: "Acme Corporation - OpenID Connect"
      myidp_saml2:
        description: "Acme Corporation - SAML2"
    idp_mapping:
      myidp_oidc:
        id: myidp
        protocol: oidc
      myidp_saml2:
        id: myidp
        protocol: saml2
```

Images upload mode

Horizon allows using different strategies when uploading images to Glance that are controlled by the `horizon:server:images_upload_mode` pillar. Possible options are direct, legacy, off. When direct mode is used, CORS have to be enabled on Glance side, and client should use modern browser.

```
horizon:
  server:
    images_upload_mode: "direct"
```

Images allow location

If set to True, this setting allows specifying an image location (URL) as the image source when creating or updating images. Depending on the Glance version, the ability to set an image location is controlled by policies and/or the Glance configuration. Therefore
IMAGES_ALLOW_LOCATION should only be set to True if Glance is configured to allow specifying a location.

```
horizon:
  server:
    images_allow_location: True
```

Custom django settings

Django has a ton of useful settings that might be tuned for particular use case. Cover them all in templated manner is not possible. This sections shows how to configure custom django setting via horizon metadata.

```
horizon:
  server:
    django_settings:
      CUSTOM_DJANGO_OPTION:
        enabled: true
        value: 'value'
```

Upgrades

Each OpenStack formula provides a set of phases (logical blocks) that help to build a flexible upgrade orchestration logic for particular components. The table below lists the phases and their descriptions:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;app&gt;.upgrade.service_running</td>
<td>Ensure that all services for particular application are enabled for autostart and running</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.service_stopped</td>
<td>Ensure that all services for particular application disabled for autostart and dead</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pkgs_latest</td>
<td>Ensure that packages used by particular application are installed to latest available version. This will not upgrade data plane packages like qemu and openvswitch as usually 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. Applying this state will not autostart service.</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.render_config</td>
<td>Ensure configuration is rendered actual version.</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pre</td>
<td>We assume this state is applied on all nodes in the cloud before running upgrade. Only non destructive actions will be applied during this phase. Perform service built in service check like (keystone-manage doctor and nova-status upgrade)</td>
</tr>
<tr>
<td>Formula</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.pre</code></td>
<td>Mostly applicable for data plane nodes. During this phase resources will be gracefully removed from current node if it is allowed. Services for upgraded application will be set to admin disabled state to make sure node will not participate in resources scheduling. For example on gtw nodes this will set all agents to admin disable state and will move all routers to other agents.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade</code></td>
<td>This state will basically upgrade application on particular target. Stop services, render configuration, install new packages, run offline dbsync (for ctl), start services. Data plane should not be affected, only OpenStack Python services.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.post</code></td>
<td>Add services back to scheduling.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.post</code></td>
<td>This phase should be launched only when upgrade of the cloud is completed. Cleanup temporary files, perform other post upgrade tasks.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.verify</code></td>
<td>Here we will do basic health checks (API CRUD operations, verify do not have dead network agents/compute services)</td>
</tr>
</tbody>
</table>

Seealso

- [https://github.com/openstack/horizon](https://github.com/openstack/horizon)
JENKINS

Usage

Jenkins CI is an open source automation server written in Java. Jenkins helps to automate the non-human part of software development process, with continuous integration and facilitating technical aspects of continuous delivery.

For more information, see https://jenkins.io/.

Setup jenkins client, works with Salt 2016.3+, supports pipeline workflow projects only for now.

Dependencies

To install on Ubuntu, you will need to add the jenkins debian repository to the target server. You can do this with the salt-formula-linux formula, with the following pillar data:

```
linux:
  system:
    enabled: true
  repo:
    jenkins:
      enabled: true
      source: "deb http://pkg.jenkins.io/debian-stable binary/
      key_url: "https://pkg.jenkins.io/debian/jenkins-ci.org.key"
```

This state will need to be applied before the jenkins state.

Using this formula

To use this formula, you must install the formula to your Salt Master as documented in saltstack formula docs

This formula is driven by pillar data, and can be used to install either a Jenkins Master or Client. See pillar data below for examples.

Sample pillars

Master role

Simple master with reverse proxy:

```
nginx:
  server:
    site:
      jenkins:
        enabled: true
        type: nginx_proxy
        name: jenkins
        proxy:
```
```
host: 127.0.0.1
port: 8080
protocol: http

host:
  name: jenkins.example.com
  port: 80

jenkins:
  master:
    mode: EXCLUSIVE
    java_args: -Xms256m -Xmx1g
    # Do not manage any xml config files via Salt, use UI instead
    # Including config.xml and any plugin xml's.
    no_config: true

slaves:
- name: slave01
  label: pbuilder
  executors: 2
- name: slave02
  label: image_builder
  mode: EXCLUSIVE
  executors: 2

views:
- name: "Package builds"
  regex: "debian-build-.*"
- name: "Contrail builds"
  regex: "contrail-build-.*"
- name: "Aptly"
  regex: "aptly-.*"

plugins:
- name: slack
- name: extended-choice-parameter
- name: rebuild
- name: test-stability

Jenkins master with experimental plugin source support:

jenkins:
  master:
    enabled: true

SMTP server settings:

jenkins:
  master:
    email:
      engine: "smtp"
      host: "smtp.domain.com"
```
user: "user@domain.cz"
password: "smtp-password"
port: 25

Script approvals from client:

jenkins:
  client:
    approved_scripts:
      - method groovy.json.JsonSlurperClassic.parseText java.lang.String

Script approvals:

jenkins:
  master:
    approved_scripts:
      - method groovy.json.JsonSlurperClassic.parseText java.lang.String

User enforcement:

jenkins:
  master:
    user:
      admin:
        api_token: xxxxxxxxxx
        password: admin_password
        email: admin@domain.com
      user01:
        api_token: xxxxxxxxxx
        password: user_password
        email: user01@domain.com

Agent (slave) role

jenkins:
  slave:
    master:
      host: jenkins.example.com
      port: 80
      protocol: http
    user:
      name: jenkins_slave
      password: dexiech6AepohthaiHook2iesh7ol5ook40v3leid3yekeid200Ng3Ee2oKeYo
    gpg:
      keypair_id: A76882D3
      public_key: |
          -----BEGIN PGP PUBLIC KEY BLOCK-----
... private_key: |
-----BEGIN PGP PRIVATE KEY BLOCK----- ...

Client role

Simple client with workflow job definition:

```
jenkins:
  client:
    master:
      host: jenkins.example.com
      port: 80
      protocol: http
    job:
      jobname:
        type: workflow
        param:
          bool_param:
            type: boolean
            description: true/false
            default: true
          string_param:
            type: string
            description: 1 liner
            default: default_string
          text_param:
            type: text
            description: multi-liner
            default: default_text
        jobname_scm:
          type: workflow-scm
          concurrent: false
          scm:
            type: git
            url: https://github.com/jenkinsci/docker.git
            branch: master
            script: Jenkinsfile
            github:
              url: https://github.com/jenkinsci/docker
              name: "Jenkins Docker Image"
        trigger:
          timer:
            dependency_job_names:
            - job1
            - job2
            spec: "H H ** **"
```
github:
pollscm:
  spec: "H/15 * * * *"
reverse:
  projects:
    - test1
    - test2
  state: SUCCESS
param:
  bool_param:
    type: boolean
    description: true/false
    default: true
  string_param:
    type: string
    description: 1 liner
    default: default_string
  text_param:
    type: text
    description: multi-liner
    default: default_text

Inline Groovy scripts:

jenkins:
  client:
    job:
      test_workflow_jenkins_simple:
        type: workflow
        display_name: Test jenkins simple workflow
        script:
          content: |
          node {
            stage 'Stage 1'
            echo 'Hello World 1'
            stage 'Stage 2'
            echo 'Hello World 2'
          }
      test_workflow_jenkins_input:
        type: workflow
        display_name: Test jenkins workflow inputs
        script:
          content: |
          node {
            stage 'Enter string'
            input message: 'Enter job parameters', ok: 'OK', parameters: [
              string(defaultValue: 'default', description: 'Enter a string.', name: 'string'),
            ]
stage 'Enter boolean'
input message: 'Enter job parameters', ok: 'OK', parameters: [
  booleanParam(defaultValue: false, description: 'Select boolean.', name: 'Bool'),
]
stage 'Enter text'
input message: 'Enter job parameters', ok: 'OK', parameters: [
  text(defaultValue: '', description: 'Enter multiline', name: 'Multiline')
]}

GIT controlled groovy scripts:

```groovy
jenkins:
  client:
    source:
      base:
        engine: git
        address: repo_url
        branch: branch
domain:
  client:
    source:
      base:
        engine: git
        address: domain_url
        branch: branch
  job:
  type: workflow
display_name: Test jenkins simple workflow
param:
  bool_param:
    type: boolean
    description: true/false
    default: true
script:
  repository: base
  file: workflows/test_workflow_jenkins_simple.groovy
test_workflow_jenkins_input:
  type: workflow
display_name: Test jenkins workflow inputs
script:
  repository: domain
  file: workflows/test_workflow_jenkins_input.groovy
test_workflow_jenkins_input_Jenkinsfile:
  type: workflow
display_name: Test jenkins workflow inputs (jenkinsfile)
script:
  repository: domain
  file: workflows/test_workflow_jenkins_input/Jenkinsfile
```

©2020, Mirantis Inc.
GIT controlled groovy script with shared libraries:

```groovy
jenkins:
  client:
    source:
      base:
        engine: git
        address: repo_url
        branch: branch
      domain:
        engine: git
        address: domain_url
        branch: branch
  job:
    test_workflow_jenkins_simple:
      type: workflow
      display_name: Test jenkins simple workflow
      param:
        bool_param:
          type: boolean
          description: true/false
          default: true
      script:
        repository: base
        file: workflows/test_workflow_jenkins_simple.groovy
      libs:
        - repository: base
          file: macros/cookiecutter.groovy
        - repository: base
          file: macros/git.groovy
```

Setting job max builds to keep (amount of last builds stored on Jenkins master)

```groovy
jenkins:
  client:
    job:
      my-amazing-job:
        type: workflow
        discard:
          build:
            keep_num: 5
            keep_days: 5
        artifact:
          keep_num: 6
          keep_days: 6
```

Using job templates in similar way as in jjb. For now just 1 defined param is supported:
jenkins:
  client:
    job_template:
      test_workflow_template:
        name: test-{{formula}}-workflow
        template:
          type: workflow
          display_name: Test jenkins {{name}} workflow
          param:
            repo_param:
              type: string
              default: repo/{{formula}}
          script:
            repository: base
            file: workflows/test_formula_workflow.groovy
          param:
            formula:
            - aodh
            - linux
            - openssh

Interpolating parameters for job templates:

```yaml
_param:
salt_formulas:
  - aodh
  - git
  - nova
  - xorg
jenkins:
  client:
    job_template:
      test_workflow_template:
        name: test-{{formula}}-workflow
        template:
          param:
            formula: ${_param:salt_formulas}
```

Or simply define multiple jobs and it's parameters to replace from template:

```yaml
jenkins:
  client:
    job_template:
      test_workflow_template:
        name: test-{{name}}-{{myparam}}
        template:
          ...
          param:
            formula: ${_param:salt_formulas}
```

©2020, Mirantis Inc.
jobs:
- name: firstjob
  myparam: dummy
- name: secondjob
  myparam: dummyaswell

Purging undefined jobs from Jenkins:

jenkins:
  client:
    purge_jobs: true
  job:
    my-amazing-job:
      type: workflow

Plugins management from client:

jenkins:
  client:
    plugin_remove_unwanted: false
    plugin_force_remove: false
  plugin:
    plugin1: 1.2.3
    plugin2:
    plugin3: {}
    plugin4:
      version: 3.2.1
      enabled: false
    plugin5: absent

Adding plugin params to job:

jenkins:
  client:
    job:
      my_plugin_parametrized_job:
        plugin_properties:
          throttleconcurrents:
            enabled: True
            max_concurrent_per_node: 3
            max_concurrent_total: 1
          throttle_option: category #one of project (default or category)
          categories:
            - my_throttle_category
        plugin:
          throttle-concurrents:

LDAP configuration (depends on LDAP plugin):
jenkins:
  client:
    security:
      ldap:
        server: 1.2.3.4
        root_dn: dc=foo,dc=com
        user_search_base: cn=users,cn=accounts
        manager_dn: ""
        manager_password: password
        user_search: ""
        group_search_base: ""
        inhibit_infer_root_dn: false

Matrix configuration (depends on auth-matrix plugin):

jenkins:
  client:
    security:
      matrix:
        # set true for use ProjectMatrixAuthStrategy instead of GlobalMatrixAuthStrategy
        project_based: false
        permissions:
          Jenkins:
            # administrator access
            ADMINISTER:
              - admin
            # read access (anonymous too)
            READ:
              - anonymous
              - user1
              - user2
            # agents permissions
            MasterComputer:
              BUILD:
                - user3
            # jobs permissions
            hudson:
              model:
                Item:
                  BUILD:
                    - user4

Common matrix strategies
Views enforcing from client:

jenkins:
  client:
view:
  my-list-view:
    enabled: true
    type: ListView
    include_regex: ".*"
  my-view:
    # set false to disable
    enabled: true
    type: MyView

View specific params:

  • include_regex for ListView and CategorizedJobsView
  • categories for CategorizedJobsView

Categorized views:

jenkins:
  client:
    view:
      my-categorized-view:
        enabled: true
        type: CategorizedJobsView
        include_regex: ".*"
        categories:
          - group_regex: "aptly-.*-nightly-testing"
            naming_rule: "Nightly -> Testing"
          - group_regex: "aptly-.*-nightly-production"
            naming_rule: "Nightly -> Production"

Credentials enforcing from client:

jenkins:
  client:
    credential:
      cred_first:
        username: admin
        password: password
      cred_second:
        username: salt
        password: password
      cred_with_key:
        username: admin
        key: SOMESSHKEY
      cred_with_text_secret:
        secret: SOMETEXTSECRET
      cred_with_secret_file:
        filename: somefile.json
```yaml
content: |
  { "Hello": "world!" }
```

Users enforcing from client:

```yaml
jenkins:
  client:
    user:
      admin:
        password: admin_password
        admin: true
      user01:
        password: user_password
```

Node enforcing from client using JNLP launcher:

```yaml
jenkins:
  client:
    node:
      node01:
        remote_home: /remote/home/path
        desc: node-description
        num_executors: 1
        node_mode: Normal
        ret_strategy: Always
        labels:
          - example
          - label
        launcher:
          type: jnlp
```

Node enforcing from client using SSH launcher:

```yaml
jenkins:
  client:
    node:
      node01:
        remote_home: /remote/home/path
        desc: node-description
        num_executors: 1
        node_mode: Normal
        ret_strategy: Always
        labels:
          - example
          - label
        launcher:
          type: ssh
```
Configure Jenkins master:

```yaml
jenkins:
  client:
    node:
      master:
        num_executors: 1
        node_mode: Normal # or Exclusive
        labels:
          - example
          - label
```

Setting node labels:

```yaml
jenkins:
  client:
    label:
      node-name: node-name
      lbl_text: label-offline
      append: false # set true for label append instead of replace
```

SMTP server settings from client:

```yaml
jenkins:
  client:
    smtp:
      host: "smtp.domain.com"
      username: "user@domain.cz"
      password: "smtp-password"
      port: 25
      ssl: false
      reply_to: reply_to@address.com
```

Jenkins admin user email enforcement from client:

```yaml
jenkins:
  client:
    smtp:
      admin_email: "My Jenkins <jenkins@myserver.com>"
```

Slack plugin configuration:
Pipeline global libraries setup:

```yaml
jenkins:
  client:
    slack:
      team_domain: example.com
      token: slack-token
      room: slack-room
      token_credential_id: cred_id
      send_as: Some slack user
```

Artifactory server enforcing:

```yaml
jenkins:
  client:
    artifactory:
      my-artifactory-server:
        enabled: true
        url: https://path-to-my-library
        credential_id: github
```

Jenkins Global env properties enforcing:

```yaml
jenkins:
  client:
    globalenvprop:
      OFFLINE_DEPLOYMENT:
        enabled: true
        name: "OFFLINE_DEPLOYMENT" # optional, default using dict key
        value: "true"
```

Throttle categories management from client (requires Throttle Concurrent Builds plugin):

```yaml
jenkins:
  client:
    throttle_category:
      'My First Category':
```
max_total: 2
max_per_node: 1
'My Second Category':
  max_total: 5
  max_per_node: 2
  max_per_label:
    'node_label_1': 1
    'node_label_2': 2
'My Category To Remove:
  enabled: false

Jira sites management from client (requires JIRA plugin):

# Remove all sites
jenkins:
  client:
    jira:
      enabled: False

jenkins:
  client:
    jira:
      sites:
        'http://my.jira.site/':
          link_url: 'http://alternative.link/
          http_auth: false
          use_wiki_notation: false
          record_scm: false
          disable_changelog: false
          issue_pattern: ''
          any_build_result: false
          user: 'username'
          password: 'passwd'
          conn_timeout: 10
          visible_for_group: ''
          visible_for_project: ''
          timestamps: false
          timestamp_format: ''

Gerrit trigger plugin configuration:

jenkins:
  client:
    gerrit:
      server1:
        host: "gerrit.domain.local"
        port: 29418
username: "jenkins"
email: "jenkins@domain.local"
auth_key_file: "/var/jenkins_home/.ssh/id_rsa"
frontendURL: "https://gerrit.domain.local"
build_current_patches_only: true
abort_new_patchsets: false
abort_manual_patchsets: false
abort_same_topic: false
authkey: |
  SOMESSHKEY

server2:
  host: "gerrit2.domain.local"
  port: 29418
  username: "jenkins"
  email: "jenkins@domain.local"
  auth_key_file: "/var/jenkins_home/.ssh/id_rsa"
  frontendURL: "https://gerrit2.domain.local"
  build_current_patches_only: true
  abort_new_patchsets: false
  abort_manual_patchsets: false
  abort_same_topic: false
  authkey: |
    SOMESSHKEY

CSRF Protection configuration:

jenkins:
  client:
    security:
      csrf:
        enabled: true
        proxy_compat: false

Agent to Master Access Control:

jenkins:
  client:
    security:
      agent2master:
        enabled: true
        whitelisted: ""
        file_path_rules: ""

Content Security Policy configuration:

jenkins:
  client:
security:
csp: "sandbox; default-src 'none'; img-src 'self'; style-src 'self';"

Usage

1. Generate password hash:
   
   `echo -n "salt{plainpassword}" | openssl dgst -sha256`

2. Place in the configuration salt:hashpassword.

Read more

- [https://wiki.jenkins-ci.org/display/JENKINS/Use+Jenkins](https://wiki.jenkins-ci.org/display/JENKINS/Use+Jenkins)

**Metadata schema specifications for Jenkins client**

Core properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>sites</td>
<td>object</td>
<td>Jira sites to configure</td>
</tr>
<tr>
<td>node</td>
<td>object</td>
<td>Jenkins slave nodes config</td>
</tr>
<tr>
<td>trigger_gerrit_server</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>patternProperties</td>
<td>ERROR</td>
<td>description_notset</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Jenkins client</td>
</tr>
<tr>
<td>purge_jobs</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>username</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>charset</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>ssl</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>reply_to</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>admin_email</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>jenkins_jobs_root</td>
<td>string</td>
<td>Root folder for jenkins jobs</td>
</tr>
<tr>
<td>jenkins_source_root</td>
<td>string</td>
<td>Root folder for jenkins source repositories</td>
</tr>
<tr>
<td>job_status</td>
<td>object</td>
<td>description_notset</td>
</tr>
<tr>
<td>plugin_remove_unwanted</td>
<td>boolean</td>
<td>Whether to remove not listed plugins</td>
</tr>
<tr>
<td>job</td>
<td>object</td>
<td>Jenkins jobs configuration For details, see: _job definition</td>
</tr>
<tr>
<td>patternProperties</td>
<td>ERROR</td>
<td>description_notset</td>
</tr>
<tr>
<td>flowdurability_level</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>token_credential_id</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>team_domain</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>token</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>send_as</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>room</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>approved_scripts</td>
<td>array</td>
<td>NO REF Jenkins approved scripts for use in pipelines</td>
</tr>
<tr>
<td>plugin_force_remove</td>
<td>boolean</td>
<td>Force removing plugins recursively with all dependent plugins</td>
</tr>
<tr>
<td>job_template</td>
<td>object</td>
<td>Job templates definition</td>
</tr>
<tr>
<td>lib</td>
<td>object</td>
<td>Jenkins libraries configuration</td>
</tr>
<tr>
<td>plugin</td>
<td>array</td>
<td>Jenkins global environment properties</td>
</tr>
<tr>
<td>patternProperties</td>
<td>ERROR</td>
<td>description_notset</td>
</tr>
<tr>
<td>gerrit</td>
<td>object</td>
<td>Gerrit configuration in jenkins</td>
</tr>
<tr>
<td>label</td>
<td>object</td>
<td>Map of jenkins slaves and labels</td>
</tr>
<tr>
<td>patternProperties</td>
<td>ERROR</td>
<td>description_notset</td>
</tr>
<tr>
<td>css_url</td>
<td>string</td>
<td>Url or path to theme CSS files</td>
</tr>
<tr>
<td>js_url</td>
<td>boolean</td>
<td>Url or path to theme JS files</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Jenkins master host to connect to</td>
</tr>
<tr>
<td>protocol</td>
<td>string</td>
<td>Protocol to connect to jenkins master</td>
</tr>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>Jenkins master port to connect to</td>
</tr>
</tbody>
</table>
### pkgs
- **Type:** array
- **Description:** List of Jenkins master packages to be installed

### file_path_rules
- **Type:** boolean
- **Description:**

### whitelisted
- **Type:** string
- **Description:**

### manager_password
- **Type:** string
- **Description:**

### inhibit_infer_root_dn
- **Type:** string
- **Description:**

### manager_dn
- **Type:** string
- **Description:**

### group_search_base
- **Type:** string
- **Description:**

### root_dn
- **Type:** string
- **Description:**

### server
- **Type:** string
- **Description:** LDAP server url

### user_search
- **Type:** string
- **Description:**

### user_search_base
- **Type:** string
- **Description:**

### csp
- **Type:** string
- **Description:** CSP security policy

### proxy_compatible
- **Type:** boolean
- **Description:**

### enabled
- **Type:** boolean
- **Description:**

### project_base
- **Type:** boolean
- **Description:** Flag if it is GlobalMatrix security or ProjectMatrix security

### permissions
- **Type:** string
- **Description:** Map of security matrix permissions

### artifactory
- **Type:** object
- **Description:** Artifactory configuration in jenkins

### throttle_category
- **Type:** object
- **Description:** Concurrent build configuration

### view
- **Type:** object
- **Description:** Jenkins views configuration

---

**_job definition_**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>branches</td>
<td>array</td>
<td>description_notset</td>
</tr>
<tr>
<td>refspec</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>script</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>depth</td>
<td>['integer', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>shallow</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>no_tags</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>honor_refspector</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>reference</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>branch</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>credentials</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>remote_name</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>wipe_workspace</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>limit_one_job_with_matching_params</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>throttle_option</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>max_concurrent_per_node</td>
<td>['integer', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>categories</td>
<td>array</td>
<td>description_notset</td>
</tr>
<tr>
<td>max_concurrent_total</td>
<td>['integer', 'string']</td>
<td>description_notset</td>
</tr>
<tr>
<td>display_name</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>repository</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>file</td>
<td>string</td>
<td>Repository to checkout workflow file</td>
</tr>
<tr>
<td>auth_token</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>param</td>
<td>object</td>
<td>Job parameters</td>
</tr>
<tr>
<td>quiet_period</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>concurrent</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>sandbox</td>
<td>boolean</td>
<td>description_notset</td>
</tr>
<tr>
<td>trigger</td>
<td>object</td>
<td>Jenkins job trigger configuration</td>
</tr>
</tbody>
</table>
libs  array  description_notset
keep_days  ['integer', 'string']  description_notset
keep_num  ['integer', 'string']  description_notset
keep_days  ['integer', 'string']  description_notset
keep_num  ['integer', 'string']  description_notset
type  string  description_notset

**Metadata schema specifications for Jenkins job_builder**

Core properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>string</td>
<td>Base configuration folder for Jenkins Job Builder</td>
</tr>
<tr>
<td>conf</td>
<td>string</td>
<td>Folder for jenkins_jobs.ini file</td>
</tr>
<tr>
<td>engine</td>
<td>string</td>
<td>Installation source for Jenkins Job Builder. Can be one of ['pkg', 'pip']</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Path to Jenkins Job Builder configuration file</td>
</tr>
<tr>
<td>branch</td>
<td>string</td>
<td>Branch of the remote repository with Jenkins Job builder configuration</td>
</tr>
<tr>
<td>address</td>
<td>string</td>
<td>Address of the remote repository with Jenkins Job builder configuration</td>
</tr>
<tr>
<td>pkgs</td>
<td>array</td>
<td>NO REF List of packages to be installed. Set if 'source' is 'pkg'</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Jenkins Job Builder installation</td>
</tr>
</tbody>
</table>

**Metadata schema specifications for Jenkins master**

Core properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>Jenkins master http port</td>
</tr>
<tr>
<td>java_args</td>
<td>string</td>
<td>Java args for Jenkins master process</td>
</tr>
<tr>
<td>views</td>
<td>array</td>
<td>Jenkins views parameters For details, see: _views definition</td>
</tr>
<tr>
<td>sudo</td>
<td>boolean</td>
<td>Enables nopasswd sudo for jenkins system user</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>user</td>
<td>object</td>
<td>Jenkins user parameters For details, see: _user definition</td>
</tr>
<tr>
<td>plugins</td>
<td>array</td>
<td>NO REF Jenkins plugin parameters</td>
</tr>
<tr>
<td>home</td>
<td>string</td>
<td>Jenkins master home directory to store configuration</td>
</tr>
<tr>
<td>approved_scripts</td>
<td>array</td>
<td>NO REF List of approved scripts</td>
</tr>
<tr>
<td>no_config</td>
<td>boolean</td>
<td>Do not configure jenkins master</td>
</tr>
<tr>
<td>service</td>
<td>string</td>
<td>Jenkins service name</td>
</tr>
<tr>
<td>update_site_url</td>
<td>string</td>
<td>Jenkins master update center url</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Jenkins master configuration</td>
</tr>
<tr>
<td>pkgs</td>
<td>array</td>
<td>NO REF List of Jenkins master packages to be installed</td>
</tr>
<tr>
<td>slaves</td>
<td>array</td>
<td>Jenkins slaves parameters For details, see: _slaves definition</td>
</tr>
<tr>
<td>config</td>
<td>string</td>
<td>Path to jenkins master configuration file</td>
</tr>
<tr>
<td>engine</td>
<td>string</td>
<td>Jenkins email engine</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Jenkins email host</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Jenkins email user password</td>
</tr>
<tr>
<td>user</td>
<td>string</td>
<td>Jenkins email user</td>
</tr>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>Jenkins email port</td>
</tr>
<tr>
<td>mode</td>
<td>string</td>
<td>Jenkins master mode</td>
</tr>
</tbody>
</table>

_user definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public_keys</td>
<td>array</td>
<td>Jenkins user public keys</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Jenkins user password</td>
</tr>
<tr>
<td>email</td>
<td>string</td>
<td>Jenkins user email</td>
</tr>
<tr>
<td>api_token</td>
<td>string</td>
<td>Jenkins user API token</td>
</tr>
</tbody>
</table>

_slaves definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>executors</td>
<td>integer</td>
<td>Jenkins slave num of executors</td>
</tr>
<tr>
<td>mode</td>
<td>string</td>
<td>Jenkins slave mode</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Jenkins slave name</td>
</tr>
<tr>
<td>label</td>
<td>string</td>
<td>Jenkins slave label</td>
</tr>
</tbody>
</table>
### _views definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>regex</td>
<td>string</td>
<td>Jenkins regex for jobs under view</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Jenkins view name</td>
</tr>
</tbody>
</table>

### Metadata schema specifications for Jenkins slave

#### Core properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public_key</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>private_key</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>keypair_id</td>
<td>string</td>
<td>description_notset</td>
</tr>
<tr>
<td>service</td>
<td>string</td>
<td>Jenkins slave service name</td>
</tr>
<tr>
<td>sudo</td>
<td>boolean</td>
<td>Enables nopasswd sudo for Jenkins slave user</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Jenkins slave configuration</td>
</tr>
<tr>
<td>ccachedir</td>
<td>string</td>
<td>GPG keypair id for Jenkins slave</td>
</tr>
<tr>
<td>mirrorsite</td>
<td>string</td>
<td>Site mirror for pbuilder</td>
</tr>
<tr>
<td>usenetwork</td>
<td>boolean</td>
<td>Use network in Pbuilder</td>
</tr>
<tr>
<td>aptcache</td>
<td>string</td>
<td>Pbuilder apt cache directory</td>
</tr>
<tr>
<td>buildresult</td>
<td>string</td>
<td>Pbuilder build result</td>
</tr>
<tr>
<td>othermirror</td>
<td>ERROR</td>
<td>description_notset For details, see: _othermirror definition</td>
</tr>
<tr>
<td>buildplace</td>
<td>string</td>
<td>Pbuilder build place folder</td>
</tr>
<tr>
<td>keyring</td>
<td>string</td>
<td>Mirror keyring</td>
</tr>
<tr>
<td>aptcachehardlink</td>
<td>boolean</td>
<td>True if apt cache directory is hard link</td>
</tr>
<tr>
<td>parallel</td>
<td>['boolean', 'integer']</td>
<td>Number of parallel threads for Pbuilder. Set to false to use default (num of cpu)</td>
</tr>
<tr>
<td>components</td>
<td>array</td>
<td>Pbuilder components</td>
</tr>
<tr>
<td>os</td>
<td>object</td>
<td>OS mirror parameters for Pbuilder For details, see: _os_parameters definition</td>
</tr>
<tr>
<td>eatmydata</td>
<td>boolean</td>
<td>Install eatmydata as extra package</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>Keystone server url</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Keystone server user password</td>
</tr>
<tr>
<td>user</td>
<td>string</td>
<td>Keystone server user</td>
</tr>
<tr>
<td>tenant</td>
<td>string</td>
<td>Keystone server user tenant</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Jenkins slave user name</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Jenkins slave user name</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Jenkins master host to connect to</td>
</tr>
<tr>
<td>protocol</td>
<td>string</td>
<td>Protocol to connect to Jenkins master</td>
</tr>
<tr>
<td>port</td>
<td>['integer', 'string']</td>
<td>Jenkins master port to connect to</td>
</tr>
<tr>
<td>init_script</td>
<td>string</td>
<td>Path to Jenkins slave init script</td>
</tr>
<tr>
<td>config</td>
<td>string</td>
<td>Path to Jenkins slave configuration file</td>
</tr>
<tr>
<td>hostname</td>
<td>string</td>
<td>Jenkins slave hostname</td>
</tr>
<tr>
<td>pkgs</td>
<td>array</td>
<td>List of packages to be installed</td>
</tr>
</tbody>
</table>

_os_parameters definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_os_parameters</td>
<td>object</td>
<td>Map of OS and its distribution parameters For details, see: _os_distribution_parameters definition</td>
</tr>
</tbody>
</table>

_othermirror definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>string</td>
<td>Mirror url</td>
</tr>
<tr>
<td>dist</td>
<td>string</td>
<td>Mirror dist</td>
</tr>
<tr>
<td>trusted</td>
<td>boolean</td>
<td>Trusted mirror or not</td>
</tr>
<tr>
<td>components</td>
<td>array</td>
<td>Mirror components</td>
</tr>
</tbody>
</table>

_os_distribution_parameters definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mirrorsite</td>
<td>string</td>
<td>Site mirror for pbuilder</td>
</tr>
<tr>
<td>extrapackages</td>
<td>array</td>
<td>Distribution extra packages</td>
</tr>
<tr>
<td>keyring</td>
<td>string</td>
<td>Keyring for distribution mirror</td>
</tr>
<tr>
<td>arch</td>
<td>string</td>
<td>Distribution architecture</td>
</tr>
<tr>
<td>eatmydata</td>
<td>boolean</td>
<td>Install eatmydata as extra package</td>
</tr>
<tr>
<td>othermirror</td>
<td>ERROR</td>
<td>description_notset For details, see: _othermirror definition</td>
</tr>
</tbody>
</table>
KEEPALIVED

Usage
Keepalived is a routing software written in C. The main goal of this project is to provide simple and robust facilities for loadbalancing and high-availability to Linux system and Linux based infrastructures. Loadbalancing framework relies on well-known and widely used Linux Virtual Server (IPVS) kernel module providing Layer4 loadbalancing. Keepalived implements a set of checkers to dynamically and adaptively maintain and manage loadbalanced server pool according their health. On the other hand high-availability is achieved by VRRP protocol. VRRP is a fundamental brick for router failover. In addition, Keepalived implements a set of hooks to the VRRP finite state machine providing low-level and high-speed protocol interactions. Keepalived frameworks can be used independently or all together to provide resilient infrastructures.

Sample pillar
Simple virtual IP on an interface:

```
keepalived:
cluster:
  enabled: True
instance:
  VIP1:
    nopreempt: True
    priority: 100 (highest priority must be on primary server, different for cluster members)
    virtual_router_id: 51
    auth_type: AH
    password: pass
    address: 192.168.10.1
    interface: eth0
  VIP2:
    nopreempt: True
    priority: 150 (highest priority must be on primary server, different for cluster members)
    virtual_router_id: 52
    auth_type: PASS
    password: pass
    address: 10.0.0.5
    interface: eth1
```

Multiple virtual IPs on single interface:

```
keepalived:
cluster:
  enabled: True
instance:
  VIP1:
    nopreempt: True
    priority: 100 (highest priority must be on primary server, different for cluster members)
virtual_router_id: 51
password: pass
addresses:
- 192.168.10.1
- 192.168.10.2
interface: eth0

Use unicast:

keepalived:
  cluster:
    enabled: True
  instance:
    VIP1:
      nopreempt: True
      priority: 100 (highest priority must be on primary server, different for cluster members)
      virtual_router_id: 51
      password: pass
      address: 192.168.10.1
      interface: eth0
      unicast_src_ip: 172.16.10.1
      unicast_peer:
        172.16.10.2
        172.16.10.3

Disable nopreempt mode to have Master. Highest priority is taken in all cases:

keepalived:
  cluster:
    enabled: True
  instance:
    VIP1:
      nopreempt: False
      priority: 100 (highest priority must be on primary server, different for cluster members)
      virtual_router_id: 51
      password: pass
      addresses:
        - 192.168.10.1
        - 192.168.10.2
      interface: eth0

Notify action in keepalived:

keepalived:
  cluster:
    enabled: True
  instance:
**VIP1:**

- `nopreempt`: True
- `notify_action`:
  - `master`:
    - `/usr/bin/docker start jenkins`
    - `/usr/bin/docker start gerrit`
  - `backup`:
    - `/usr/bin/docker stop jenkins`
    - `/usr/bin/docker stop gerrit`
- `fault`:
  - `/usr/bin/docker stop jenkins`
  - `/usr/bin/docker stop gerrit`
- `priority`: 100 # highest priority must be on primary server, different for cluster members
- `virtual_router_id`: 51
- `password`: pass
- `addresses`:
  - 192.168.10.1
  - 192.168.10.2
- `interface`: eth0

Track/vrrp scripts for keepalived instance:

**keepalived**:

- `enabled`: True
- `instance`:

**VIP2**:

- `priority`: 100
- `virtual_router_id`: 10
- `password`: pass
- `addresses`:
  - 192.168.11.1
  - 192.168.11.2
- `interface`: eth0
- `track_script`: check_haproxy

**VIP3**:

- `priority`: 100
- `virtual_router_id`: 11
- `password`: pass
- `addresses`:
  - 192.168.10.1
  - 192.168.10.2
- `interface`: eth0
- `track_script`:
  - `check_random_exit`:
    - `interval`: 10
  - `check_port`: 50
vrrp_scripts:
check_haproxy:
  name: check_pidof
  args:
    - haproxy
check_mysql_port:
  name: check_port
  args:
    - 3306
    - TCP
    - 4
check_ssh:
  name: check_port
  args: "22"
check_mysql_cluster:
  args:
    # github: olafz/percona-clustercheck
    # <user> <pass> <available_when_donor=0|1> <log_file> <available_when_readonly=0|1> <defaults_extra_file>
    - clustercheck
    - clustercheck
    - available_when_donor=0
    - available_when_readonly=0
check_random_exit:
  interval: 10
  timeout: 5
  content: |
    #!/bin/bash
    exit $(($RANDOM%2))
  weight: 50

Read more

- https://raymii.org/s/tutorials/Keepalived-Simple-IP-failover-on-Ubuntu.html
**KEYSTONE**

**Usage**

Keystone provides authentication, authorization and service discovery mechanisms via HTTP primarily for use by projects in the OpenStack family. It is most commonly deployed as an HTTP interface to existing identity systems, such as LDAP.

From Kilo release Keystone v3 endpoint has definition without version in url

<table>
<thead>
<tr>
<th>id</th>
<th>region</th>
<th>publicurl</th>
<th>internalurl</th>
<th>adminurl</th>
<th>service_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>91663a8d...494</td>
<td>RegionOne</td>
<td><a href="http://10.0.150.37:5000/">http://10.0.150.37:5000/</a></td>
<td><a href="http://10.0.150.37:5000/">http://10.0.150.37:5000/</a></td>
<td><a href="http://10.0.150.37:35357/">http://10.0.150.37:35357/</a></td>
<td>0fd2dba...9c9</td>
</tr>
</tbody>
</table>

Sample pillars

Caution!

When you use localhost as your database host (keystone:server: atabase:host), sqlalchemy will try to connect to /var/run/mysql/ mysqld.sock, may cause issues if you located your mysql socket elsewhere

Full stacked Keystone:

```yaml
keystone:
  server:
    enabled: true
    version: juno
    service_token: 'service_token'
    service_tenant: service
    service_password: 'servicepwd'
    admin_tenant: admin
    admin_name: admin
    admin_password: 'adminpwd'
    admin_email: stackmaster@domain.com
    enable_proxy_headers_parsing: True
  roles:
    - admin
    - Member
    - image_manager
  bind:
    address: 0.0.0.0
    private_address: 127.0.0.1
    private_port: 35357
```
**public_address**: 127.0.0.1
**public_port**: 5000
**api_version**: 2.0
**region**: RegionOne
**database**:
  - **engine**: mysql
  - **host**: '127.0.0.1'
  - **name**: 'keystone'
  - **password**: 'LfTno5mYdZmRfoPV'
  - **user**: 'keystone'

Keystone public HTTPS API:

```yaml
keystone:
  server:
    enabled: true
    version: juno
    ...
  services:
    - name: nova
      type: compute
      description: OpenStack Compute Service
      user:
        name: nova
        password: password
      bind:
        public_address: cloud.domain.com
        public_protocol: https
        public_port: 8774
        internal_address: 10.0.0.20
        internal_port: 8774
        admin_address: 10.0.0.20
        admin_port: 8774
```

Keystone with custom policies. Keys with specified rules are created or set to this value if they already exists. Keys with no value (like our existing_rule) are deleted from the policy file:

```yaml
keystone:
  server:
    enabled: true
    policy:
      new_rule: "rule:admin_required"
      existing_rule:
```

Keystone memcached storage for tokens:
keystone:
    server:
      enabled: true
      version: juno
      ...
    token_store: cache
    cache:
      engine: memcached
      host: 127.0.0.1
      port: 11211
    services:
      ...

Keystone clustered memcached storage for tokens:

keystone:
    server:
      enabled: true
      version: juno
      ...
    token_store: cache
    cache:
      engine: memcached
      members:
      - host: 192.160.0.1
        port: 11211
      - host: 192.160.0.2
        port: 11211
    services:
      ...

Keystone client:

keystone:
    client:
      enabled: true
      server:
        host: 10.0.0.2
        public_port: 5000
        private_port: 35357
        service_token: 'token'
        admin_tenant: admin
        admin_name: admin
        admin_password: 'passwd'

Keystone cluster
keystone:
  control:
    enabled: true
  provider:
    os15_token:
      host: 10.0.0.2
      port: 35357
      token: token
    os15_tcp_core_stg:
      host: 10.0.0.5
      port: 5000
      tenant: admin
      name: admin
      password: password

Keystone fernet tokens for OpenStack Kilo release:

keystone:
  server:
    ... tokens:
      engine: fernet
      max_active_keys: 3
    ...

Keystone auth methods:

keystone:
  server:
    ... auth_methods:
      - external
      - password
      - token
      - oauth1
    ...

Keystone domain with LDAP backend, using SQL for role/project assignment:

keystone:
  server:
    domain:
      external:
        description: "Testing domain"
        backend: ldap
        assignment:
          backend: sql
ldap:
  url: "ldaps://idm.domain.com"
  suffix: "dc=cloud,dc=domain,dc=com"
  # Will bind as uid=keystone,cn=users,cn=accounts,dc=cloud,dc=domain,dc=com
  uid: keystone
  password: password

Use driver aliases for drivers instead of class path’s:

keystone:
  server:
    domain:
      test:
        description: "Test domain"
        backend: ldap
        assignment:
          backend: sql
          driver: sql
        identity:
          backend: ldap
          driver: keystone.identity.backends.ldap.Identity
  ldap:
    url: "ldaps://idm.domain.com"
    ...

Using LDAP backend for default domain:

keystone:
  server:
    backend: ldap
    assignment:
      backend: sql
  ldap:
    url: "ldaps://idm.domain.com"
    suffix: "dc=cloud,dc=domain,dc=com"
    # Will bind as uid=keystone,cn=users,cn=accounts,dc=cloud,dc=domain,dc=com
    uid: keystone
    password: password

Using LDAP backend for default domain with user_enabled field emulation:

keystone:
  server:
    backend: ldap
    assignment:
      backend: sql
  ldap:

url: "ldap://idm.domain.com"
suffix: "ou=Openstack Service Users,o=domain.com"
bind_user: keystone
password: password

# Define LDAP "group" object class and "membership" attribute
group_objectclass: groupOfUniqueNames
group_member_attribute: uniqueMember

# User will receive "enabled" attribute basing on membership in "os-user-enabled" group
user_enabled_emulation: True
user_enabled_emulation_dn: "cn=os-user-enabled,ou=Openstack,o=domain.com"
user_enabled_emulation_use_group_config: True

If the members of the group objectclass are user IDs rather than DNs, set group_members_are_ids to true. This is the case when using posixGroup as the group objectclass and OpenDirectory:

keystone:
  server:
    backend: ldap
    assignment:
      backend: sql
  ldap:
    url: "ldaps://idm.domain.com"
    suffix: "dc=cloud,dc=domain,dc=com"
    # Will bind as uid=keystone,cn=users,cn=accounts,dc=cloud,dc=domain,dc=com
    uid: keystone
    password: password
    group_members_are_ids: True

Simple service endpoint definition (defaults to RegionOne):

keystone:
  server:
    service:
      ceilometer:
        type: metering
        description: OpenStack Telemetry Service
        user:
          name: ceilometer
          password: password
          bind:
            ...

Region-aware service endpoints definition:

keystone:
  server:
service:
ceilometer_region01:
  service: ceilometer
  type: metering
  region: region01
  description: OpenStack Telemetry Service
  user:
    name: ceilometer
    password: password
  bind:
    ...
ceilometer_region02:
  service: ceilometer
  type: metering
  region: region02
  description: OpenStack Telemetry Service
  bind:
    ...

Enable Ceilometer notifications:

keystone:
  server:
    notification: true
    message_queue:
      engine: rabbitmq
      host: 127.0.0.1
      port: 5672
      user: openstack
      password: password
      virtual_host: '/openstack'
      ha_queues: true

Client-side RabbitMQ HA setup:

keystone:
  server:
    ....
    message_queue:
      engine: rabbitmq
      members:
        - host: 10.0.16.1
        - host: 10.0.16.2
        - host: 10.0.16.3
      user: openstack
      password: pwd
      virtual_host: '/openstack'
    ....
Client-side RabbitMQ TLS configuration:

By default system-wide CA certs are used. Nothing should be specified except ssl.enabled.

```yaml
keystone:
  server:
    ....
    message_queue:
      ssl:
        enabled: True
```

Use cacert_file option to specify the CA-cert file path explicitly:

```yaml
keystone:
  server:
    ....
    message_queue:
      ssl:
        enabled: True
        cacert_file: /etc/ssl/rabbitmq-ca.pem
```

To manage content of the cacert_file use the cacert option:

```yaml
keystone:
  server:
    ....
    message_queue:
      ssl:
        enabled: True
        cacert: |
          -----BEGIN CERTIFICATE-----
          ...
          -----END CERTIFICATE-----

        cacert_file: /etc/openstack/rabbitmq-ca.pem
```

**Note**
- The message_queue.port is set to 5671 (AMQPS) by default if ssl.enabled=True.
- Use message_queue.ssl.version if you need to specify protocol version. By default, is TLSv1 for python < 2.7.9 and TLSv1_2 for version above.

Enable CADF audit notification:
keystone:
  server:
    notification: true
    notification_format: cadf

Run Keystone under Apache:

keystone:
  server:
    service_name: apache2
apache:
  server:
    enabled: true
    default_mpm: event
  site:
    keystone:
      enabled: true
      type: keystone
      name: wsgi
      host:
        name: ${linux:network:fqdn}
    modules:
      - wsgi

Enable SAML2 Federated keystone:

keystone:
  server:
    auth_methods:
    - password
    - token
    - saml2
  federation:
    saml2:
      protocol: saml2
      remote_id_attribute: Shib-Identity-Provider
      shib_url_scheme: https
      shib_compat_valid_user: 'on'
      federation_driver: keystone.contrib.federation.backends.sql.Federation
      federated_domain_name: Federated
      trusted_dashboard:
        - https://${_param:cluster_public_host}/horizon/auth/websso/
apache:
  server:
    pkgs:
    - apache2
    - libapache2-mod-shib2
  modules:
Enable OIDC Federated Keystone:

```yaml
keystone:
  server:
    auth_methods:
      - password
      - token
      - oidc
  federation:
    oidc:
      protocol: oidc
      remote_id_attribute: HTTP_OIDC_ISS
      remote_id_attribute_value: https://accounts.google.com
      oidc_claim_prefix: "OIDC-
      oidc_response_type: id_token
      oidc_scope: "openid email profile"
      oidc_provider_metadata_url: https://accounts.google.com/.well-known/openid-configuration
      oidc_client_id: <openid_client_id>
      oidc_client_secret: <openid_client_secret>
      oidc_crypto_passphrase: openstack
      oidc_redirect_uri: https://key.example.com:5000/v3/auth/OS-FEDERATION/websso/oidc/redirect
      oidc_oauth_introspection_endpoint: https://www.googleapis.com/oauth2/v1/tokeninfo
      oidc_oauth_introspection_token_param_name: access_token
      oidc_oauth_remote_user_claim: user_id
      oidc_ssl_validate_server: 'off'
  federated_domain_name: Federated
  federation_driver: keystone.contrib.federation.backends.sql.Federation
  trusted_dashboard:
    - https://${_param:cluster_public_host}/auth/websso/

apache:
  server:
    pkgs:
      - apache2
      - libapache2-mod-auth-openidc
    modules:
      - wsgi
      - auth_openidc
```

Note

Ubuntu Trusty repository doesn’t contain libapache2-mod-auth-openidc package. Additional repository should be added to the source list.

Use a custom identity driver with custom options:
keystone:
  server:
    backend: k2k
    k2k:
      auth_url: 'https://keystone.example.com/v2.0'
      read_user: 'example_user'
      read_pass: 'password'
      read_tenant_id: 'admin'
      identity_driver: 'sql'
      id_prefix: 'k2k:'
      domain: 'default'
      caching: true
      cache_time: 600

Enable CORS parameters:

keystone:
  server:
    cors:
      expose_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
      allow_methods: GET,PUT,POST,DELETE,PATCH
      allow_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
      allow_credentials: True
      max_age: 86400

Keystone client

Service endpoints enforcement with service token:

keystone:
  client:
    enabled: true
  server:
    keystone01:
      admin:
        host: 10.0.0.2
        port: 35357
        token: 'service_token'
      service:
        nova:
          type: compute
          description: OpenStack Compute Service
          endpoints:
            - region: region01
              public_address: 172.16.10.1
              public_port: 8773
              public_path: '/v2'
**internal_address**: 172.16.10.1  
**internal_port**: 8773  
**internal_path**: '/v2'  
**admin_address**: 172.16.10.1  
**admin_port**: 8773  
**admin_path**: '/v2'

Project, users, roles enforcement with admin user:

```
keystone:
  client:
    enabled: true
  server:
    keystone01:
      admin:
        host: 10.0.0.2
        port: 5000
        project: admin
        user: admin
        password: 'passwd'
        region_name: RegionOne
        protocol: https
      roles:
        - admin
        - member
  project:
    tenant01:
      description: "test env"
      quota:
        instances: 100
        cores: 24
        ram: 151200
        floating_ips: 50
        fixed_ips: -1
        metadata_items: 128
        injected_files: 5
        injected_file_content_bytes: 10240
        injected_file_path_bytes: 255
        key_pairs: 100
        security_groups: 20
        security_group_rules: 40
        server_groups: 20
        server_group_members: 20
      user:
        user01:
          email: jdoe@domain.com
          is_admin: true
          password: some
```
**user02:**
- **email:** jdoe2@domain.com
- **password:** some
- **roles:**
  - custom-roles

**Multiple servers example:**

```yaml
keystone:
  client:
    enabled: true
  server:
    keystone01:
      admin:
        host: 10.0.0.2
        port: 5000
        project: 'admin'
        user: admin
        password: 'workshop'
        region_name: RegionOne
        protocol: https
    keystone02:
      admin:
        host: 10.0.0.3
        port: 5000
        project: 'admin'
        user: admin
        password: 'workshop'
        region_name: RegionOne
```

**Tenant quotas:**

```yaml
keystone:
  client:
    enabled: true
  server:
    keystone01:
      admin:
        host: 10.0.0.2
        port: 5000
        project: admin
        user: admin
        password: 'passwd'
        region_name: RegionOne
        protocol: https
      roles:
        - admin
        - member
```

---

©2020, Mirantis Inc.
project:
tenant01:
    description: "test env"
    quota:
        instances: 100
        cores: 24
        ram: 151200
        floating_ips: 50
        fixed_ips: -1
        metadata_items: 128
        injected_files: 5
        injected_file_content_bytes: 10240
        injected_file_path_bytes: 255
        key_pairs: 100
        security_groups: 20
        security_group_rules: 40
        server_groups: 20
        server_group_members: 20

Extra config params in keystone.conf (since Mitaka release):

keystone:
    server:
        ....
        extra_config:
            ini_section1:
                param1: value
                param2: value
            ini_section2:
                param1: value
                param2: value
        ....

Configuration of policy.json file:

keystone:
    server:
        ....
        policy:
            admin_or_token_subject: 'rule:admin_required or rule:token_subject'

Manage os-cloud-config yml with keystone.client:

keystone:
    client:
        os_client_config:
            enabled: true
```yaml
cfgs:
  root:
    file: /root/.config/openstack/clouds.yml
    content:
      clouds:
        admin_identity:
          region_name: RegioneOne
        auth:
          username: admin
          password: secretpassword
          user_domain_name: Default
          project_name: admin
          project_domain_name: Default
          auth_url: "http://1.2.3.4:5000"

Setting up default admin project name and domain:
```

```yaml
keystone:
  server:
    ....
    admin_project:
      name: "admin"
      domain: "default"
```

Enhanced logging with logging.conf

By default logging.conf is disabled. That is possible to enable per-binary logging.conf with new variables:

- openstack_log_appender
  Set to true to enable log_config_append for all OpenStack services
- openstack_fluentd_handler_enabled
  Set to true to enable FluentHandler for all Openstack services
- openstack_ossyslog_handler_enabled
  Set to true to enable OSSysLogHandler for all Openstack services

Only WatchedFileHandler, OSSysLogHandler, and FluentHandler are available.

Also, it is possible to configure this with pillar:

```yaml
keystone:
  server:
    logging:
      log_appender: true
      log_handlers:
        watchedfile:
          enabled: true
```
Usage

1. Apply the keystone.client.service state.
2. Apply the keystone.client state.

Fernet-keys rotation without gluster

In the future fernet keys supposed to be rotated with rsync+ssh instead of using glusterfs. By default it is assumed that the script will run on primary control node (ctl01) and will rotate and transfer fernet keys to secondary controller nodes (ctl02, ctl03). Following parameter should be set on cluster level:

keystone_node_role

and fernet_rotation_driver should be set to ‘rsync’

By default this parameter is set to “secondary” on system level along with other parameters:

```yaml
keystone:
  server:
    role: ${_param:keystone_node_role}
  tokens:
    fernet_sync_nodes_list:
    control02:
      name: ctl02
      enabled: True
    control03:
      name: ctl03
      enabled: True
    fernet_rotation_driver: rsync
```

Prior to running keystone salt states ssh key should be generated and its public part should be placed on secondary controllers. It can be accomplished by running following orchestration state before keystone states:

```
salt-run state.orchestrate keystone.orchestrate.deploy
```

Currently the default fernet rotation driver is a shared filesystem

Enable x509 and SSL communication between Keystone and Galera cluster

By default communication between Keystone and Galera is unsecure.
keystone:
server:
database:
x509:
enabled: True

You able to set custom certificates in pillar:

keystone:
server:
database:
x509:
cacert: (certificate content)
cert: (certificate content)
key: (certificate content)

You can read more about it here:
https://docs.openstack.org/security-guide/databases/database-access-control.html

Upgrades
Each OpenStack formula provides a set of phases (logical blocks) that help to build a flexible upgrade orchestration logic for particular components. The table below lists the phases and their descriptions:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;app&gt;.upgrade.service_running</td>
<td>Ensure that all services for particular application are enabled for autostart and running</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.service_stopped</td>
<td>Ensure that all services for particular application disabled for autostart and dead</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pkgs_latest</td>
<td>Ensure that packages used by particular application are installed to latest available version. This will not upgrade data plane packages like qemu and openvswitch as usually 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. Applying this state will not autostart service.</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.render_config</td>
<td>Ensure configuration is rendered actual version.</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pre</td>
<td>We assume this state is applied on all nodes in the cloud before running upgrade. Only non destructive actions will be applied during this phase. Perform service built in service check like (keystone-manage doctor and nova-status upgrade)</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.pre</code></td>
<td>Mostly applicable for data plane nodes. During this phase resources will be gracefully removed from current node if it is allowed. Services for upgraded application will be set to admin disabled state to make sure node will not participate in resources scheduling. For example on gtw nodes this will set all agents to admin disable state and will move all routers to other agents.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade</code></td>
<td>This state will basically upgrade application on particular target. Stop services, render configuration, install new packages, run offline dbsync (for ctl), start services. Data plane should not be affected, only OpenStack Python services.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.post</code></td>
<td>Add services back to scheduling.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.post</code></td>
<td>This phase should be launched only when upgrade of the cloud is completed. Cleanup temporary files, perform other post upgrade tasks.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.verify</code></td>
<td>Here we will do basic health checks (API CRUD operations, verify do not have dead network agents/compute services)</td>
</tr>
</tbody>
</table>
LINUX

Linux Fomula

Linux Operating Systems:

• Ubuntu
• CentOS
• RedHat
• Fedora
• Arch

Sample pillars

Linux System

Basic Linux box

```
linux:
  system:
    enabled: true
    name: 'node1'
    domain: 'domain.com'
    cluster: 'system'
    environment: prod
    timezone: 'Europe/Prague'
    utc: true
```

Linux with system users, some with password set:

```
Warning
If no password variable is passed, any predefined password will be removed.
```

```
linux:
  system:
    ...
  user:
    jdoe:
      name: 'jdoe'
      enabled: true
      sudo: true
      shell: /bin/bash
      full_name: 'John Doe'
```
Configure password expiration parameters

The following login.defs parameters can be overridden per-user:

- PASS_MAX_DAYS
- PASS_MIN_DAYS
- PASS_WARN_DAYS
- INACTIVE

```
linux:
    system:
        ...
        user:
            jdoe:
                name: 'jdoe'
                enabled: true
            ...
            maxdays: <PASS_MAX_DAYS>
            mindays: <PASS_MIN_DAYS>
            warndays: <PASS_WARN_DAYS>
            inactdays: <INACTIVE>
```

Configure sudo for users and groups under /etc/sudoers.d/. This ways linux.system.sudo pillar map to actual sudo attributes:

```
# simplified template:
Cmds_Alias {{ alias }}={{ commands }}
```
```plaintext
{{ user }} {{ hosts }} = ({{ runas }}) NOPASSWD: {{ commands }}
%{{ group }} {{ hosts }} = ({{ runas }}) NOPASSWD: {{ commands }}

# when rendered:
saltuser1 ALL=(ALL) NOPASSWD: ALL

linux:
  system:
    sudo:
      enabled: true
  aliases:
    host:
      LOCAL:
      - localhost
      PRODUCTION:
      - db1
      - db2
  runas:
    DBA:
      - postgres
      - mysql
    SALT:
      - root
  command:
    # Note: This is not 100% safe when ALL keyword is used, user still may modify configs and hide his actions.
    # Best practice is to specify full list of commands user is allowed to run.
    SUPPORT_RESTRICTED:
      - /bin/vi /etc/sudoers*
      - /bin/vim /etc/sudoers*
      - /bin/nano /etc/sudoers*
      - /bin/emacs /etc/sudoers*
      - /bin/su - root
      - /bin/su -
      - /bin/su
      - /usr/sbin/visudo
    SUPPORT_SHELLS:
      - /bin/sh
      - /bin/ksh
      - /bin/bash
      - /bin/rbash
      - /bin/bash
      - /bin/zsh
      - /bin/csh
      - /bin/fish
      - /bin/tcsh
      - /usr/bin/login
      - /usr/bin/su
      - /usr/su

ALL_SALT_SAFE:
  - /usr/bin/salt state*
  - /usr/bin/salt service*
  - /usr/bin/salt pillar*
  - /usr/bin/salt grains*
```
- /usr/bin/salt saltutil*
- /usr/bin/salt-call state*
- /usr/bin/salt-call service*
- /usr/bin/salt-call pillar*
- /usr/bin/salt-call grains*
- /usr/bin/salt-call saltutil*

**SALT_TRUSTED:**
- /usr/bin/salt*

**users:**

# saltuser1 with default values: saltuser1 ALL=(ALL) NOPASSWD: ALL
saltuser1: {}

saltuser2:
  hosts:
  - LOCAL

# User Alias DBA

**DBA:**
  hosts:
  - ALL
  commands:
  - ALL_SALT_SAFE

**groups:**

db-ops:
  hosts:
  - ALL
  - '!PRODUCTION'
  runas:
  - DBA
  commands:
  - /bin/cat *
  - /bin/less *
  - /bin/ls *

salt-ops:
  hosts:
  - 'ALL'
  runas:
  - SALT
  commands:
  - SUPPORT_SHELLS

salt-ops-2nd:
  name: salt-ops
  nopasswd: false
  setenv: true # Enable sudo -E option
  runas:
  - DBA
  commands:
  - ALL
  - '!SUPPORT_SHELLS'
  - '!SUPPORT_RESTRICTED'
Linux with package, latest version:

```yaml
linux:
  system:
    ...
  package:
    package-name:
      version: latest
```

Linux with package from certai repo, version with no upgrades:

```yaml
linux:
  system:
    ...
  package:
    package-name:
      version: 2132.323
      repo: 'custom-repo'
      hold: true
```

Linux with package from certai repo, version with no GPG verification:

```yaml
linux:
  system:
    ...
  package:
    package-name:
      version: 2132.323
      repo: 'custom-repo'
      verify: false
```

Linux with autoupdates (automatically install security package updates):

```yaml
linux:
  system:
    ...
  autoupdates:
    enabled: true
    mail: root@localhost
    mail_only_on_error: true
    remove_unused_dependencies: false
    automatic_reboot: true
    automatic_reboot_time: "02:00"
```

Managing cron tasks

There are two data structures that are related to managing cron itself and cron tasks:
`linux:
  system:
    cron:

and

`linux:
  system:
    job:

`linux:system:cron` manages cron packages, services, and `'/etc/cron.allow'` file. ‘deny’ files are managed the only way - we’re ensuring they are absent, that’s a requirement from CIS 5.1.8

‘cron’ pillar structure is the following:

`linux:
  system:
    cron:
      enabled: true
      pkgs: [ <cron packages> ]
      services: [ <cron services> ]
      user:
        <username>:
          enabled: true

To add user to `'/etc/cron.allow'` use ‘enabled’ key as shown above.

`'/etc/cron.deny'` is not managed as CIS 5.1.8 requires it was removed.

A user would be ignored if any of the following is true: * user is disabled in `linux:system:user:<username>` * user is disabled in `linux:system:cron:user:<username>`

`linux:system:job` manages individual cron tasks.

By default, it will use name as an identifier, unless identifier key is explicitly set or False (then it will use Salt’s default behavior which is identifier same as command resulting in not being able to change it):

`linux:
  system:
    ...
    job:
      cmd1:
        command: '/cmd/to/run'
        identifier: cmd1
        enabled: true
        user: 'root'`
Managing ‘at’ tasks

Pillar for managing at tasks is similar to one for cron tasks:

```yaml
linux:
  system:
    at:
      enabled: true
      pkgs: [ <at packages> ]
      services: [ <at services> ]
    user:
      <username>:
        enabled: true
```

To add a user to ‘/etc/at.allow’ use ‘enabled’ key as shown above.

‘/etc/at.deny’ is not managed as CIS 5.1.8 requires it was removed.

A user will be ignored if any of the following is true: * user is disabled in
  `linux:system:user:<username>` * user is disabled in `linux:system:at:user:<username>`

Linux security limits (limit sensu user memory usage to max 1GB):

```yaml
linux:
  system:
    ...
    limit:
      sensu:
        enabled: true
        domain: sensu
        limits:
          - type: hard
            item: as
            value: 1000000
```

Enable autologin on tty1 (may work only for Ubuntu 14.04):

```yaml
linux:
  system:
    console:
      tty1:
        autologin: root
      # Enable serial console
    tty50:
      autologin: root
```
rate: 115200
term: xterm

To disable set autologin to false.

Set policy-rc.d on Debian-based systems. Action can be any available command in while true loop and case context. Following will disallow dpkg to stop/start services for the Cassandra package automatically:

```
linux:
  system:
    policyrcd:
      - package: cassandra
        action: exit 101
      - package: '*'
        action: switch
```

Set system locales:

```
linux:
  system:
    locale:
      en_US.UTF-8:
        default: true
      "cs_CZ.UTF-8 UTF-8":
        enabled: true
```

Systemd settings:

```
linux:
  system:
    ...
    systemd:
      system:
        Manager:
          DefaultLimitNOFILE: 307200
          DefaultLimitNPROC: 307200
        user:
          Manager:
            DefaultLimitCPU: 2
            DefaultLimitNPROC: 4
```

Ensure presence of directory:

```
linux:
  system:
    directory:
```
Ensure presence of file by specifying its source:

```yaml
linux:
  system:
    file:
      /tmp/test.txt:
        source: http://example.com/test.txt
        user: root #optional
        group: root #optional
        mode: 700 #optional
        dir_mode: 700 #optional
        encoding: utf-8 #optional
        hash: <<hash>> or <<URI to hash>> #optional
        makedirs: true #optional

linux:
  system:
    file:
      test.txt:
        name: /tmp/test.txt
        source: http://example.com/test.txt

linux:
  system:
    file:
      test2:
        name: /tmp/test2.txt
        source: http://example.com/test2.jinja
        template: jinja
```

Ensure presence of file by specifying its contents:

```yaml
linux:
  system:
    file:
      /tmp/test.txt:
        contents:
          line1
          line2
```
Ensure presence of file to be serialized through one of the serializer modules (see: https://docs.saltstack.com/en/latest/ref/serializers/all/index.html):

```yaml
linux:
  system:
    file:
      /tmp/test.txt:
        contents_grains: motd
```

Kernel

Install always up to date LTS kernel and headers from Ubuntu Trusty:

```yaml
linux:
  system:
    kernel:
      type: generic
      lts: trusty
      headers: true
```

Load kernel modules and add them to /etc/modules:

```yaml
linux:
  system:
    kernel:
      modules:
        - nf_conntrack
        - tp_smapi
        - 8021q
```

Configure or blacklist kernel modules with additional options to /etc/modprobe.d following example will add /etc/modprobe.d/nf_conntrack.conf file with line options nf_conntrack hashsize=262144:

‘option’ can be a mapping (with ‘enabled’ and ‘value’ keys) or a scalar.
Example for ‘scalar’ option value:

```yaml
linux:
  system:
    kernel:
      module:
        nf_contrack:
          option:
            hashsize: 262144
```

Example for ‘mapping’ option value:

```yaml
linux:
  system:
    kernel:
      module:
        nf_contrack:
          option:
            hashsize: true
            value: 262144
```

Note

The enabled key is optional and is true by default.

Blacklist a module:

```yaml
linux:
  system:
    kernel:
      module:
        nf_contrack:
          blacklist: true
```

A module can have a number of aliases, wildcards are allowed. Define an alias for a module:

```yaml
linux:
  system:
    kernel:
      module:
        nf_contrack:
          alias:
            nfct:
```

enabled: true
"nf_conn*:"
 enabled: true

Note
The enabled key is mandatory as no other keys exist.

Execute custom command instead of ‘insmod’ when inserting a module:

```yaml
linux:
  system:
    kernel:
      module:
        nf_conntrack:
          install:
            enabled: true
            command: /bin/true
```

Note
The enabled key is optional and is true by default.

Execute custom command instead of ‘rmmod’ when removing a module:

```yaml
linux:
  system:
    kernel:
      module:
        nf_conntrack:
          remove:
            enabled: true
            command: /bin/true
```

Note
The enabled key is optional and is true by default.

Define module dependencies:
linux:
  system:
    kernel:
      module:
        nf_conntrack:
          softdep:
            pre:
              1:
                enabled: true
                value: a
              2:
                enabled: true
                value: b
              3:
                enabled: true
                value: c
            post:
              1:
                enabled: true
                value: x
              2:
                enabled: true
                value: y
              3:
                enabled: true
                value: z

Note
The enabled key is optional and is true by default.

Install specific kernel version and ensure all other kernel packages are not present. Also install extra modules and headers for this kernel:

linux:
  system:
    kernel:
      type: generic
      extra: true
      headers: true
      version: 4.2.0-22

Systcl kernel parameters:
Configure kernel boot options:

```yaml
linux:
  system:
    kernel:
      boot_options:
        - elevator=deadline
        - spectre_v2=off
        - nopti
```

CPU

Enable cpufreq governor for every cpu:

```yaml
linux:
  system:
    cpu:
      governor: performance
```

CGROUPS

Setup linux cgroups:

```yaml
linux:
  system:
    cgroup:
      enabled: true
      group:
        ceph_group_1:
          controller:
            cpu:
              shares:
                value: 250
              usage:
                value: 0
            cpuset:
              cpus:
                value: 1,2,3
              memory:
```
limit_in_bytes:
  value: 2G
memsw.limit_in_bytes:
  value: 3G
mapping:
  subjects:
    - '@ceph'
generic_group_1:
  controller:
    cpu:
      shares:
        value: 250
    cpuacct:
      usage:
        value: 0
mapping:
  subjects:
    - '*:firefox'
    - 'student:cp'

---

Shared libraries

Set additional shared library to Linux system library path:

```
linux:
  system:
    ld:
      library:
        java:
          - /usr/lib/jvm/jre-openjdk/lib/amd64/server
          - /opt/java/jre/lib/amd64/server
```

Certificates

Add certificate authority into system trusted CA bundle:

```
linux:
  system:
    ca_certificates:
      mycert: |
        -----BEGIN CERTIFICATE-----
        MIICPDCAaUHEHEC65B0Q2Sk0tjKewPMur8wDQYJKoZIhvcNAQECBQAwXzELMAkG
        A1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBMbMuMTcwNQYDVQQLEy5DbGFz
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFoXDTI4MDgwMTIzNTk1OVowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
        cyAzIyFB1YmxpYyBQcmiltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTk2
        MDEyOTAwMDAwMFowXzELMAkGA1UEBhMCVVMxExEw
        dQYJKoZIhvcNAQE nEwDQYJKoZIhvcNAQELBQAwXzELMAkGA1UEBhMCVVMx
```
Sysfs

Install sysfs utils and set sysfs attributes:

```yaml
linux:
  system:
    sysfs:
      scheduler:
        block/sda/queue/scheduler: deadline
        power:
          mode:
            power/state: 0660
          owner:
            power/state: "root:power"
        devices/system/cpu/cpu0/cpufreq/scaling_governor: powersave
```

Optional: You can also use list that will ensure order of items.

```yaml
linux:
  system:
    sysfs:
      scheduler:
        block/sda/queue/scheduler: deadline
        power:
          - mode:
            power/state: 0660
          - owner:
            power/state: "root:power"
        devices/system/cpu/cpu0/cpufreq/scaling_governor: powersave
```

Sysfs definition with disabled automatic write. Attributes are saved to configuration, but are not applied during the run. They will be applied automatically after the reboot.

```yaml
linux:
  system:
    sysfs:
      enable_apply: false
      scheduler:
        block/sda/queue/scheduler: deadline
```

©2020, Mirantis Inc.
Huge Pages give a performance boost to applications that intensively deal with memory allocation/deallocation by decreasing memory fragmentation:

```
linux:
  system:
    kernel:
      hugepages:
        small:
          size: 2M
          count: 107520
          mount_point: /mnt/hugepages_2MB
          mount: false/true # default is true (mount immediately) / false (just save in the fstab)
        large:
          default: true # default automatically mounted
          size: 1G
          count: 210
          mount_point: /mnt/hugepages_1GB
```

Note
Not recommended to use both pagesizes concurrently.

Intel SR-IOV
PCI-SIG Single Root I/O Virtualization and Sharing (SR-IOV) specification defines a standardized mechanism to virtualize PCIe devices. The mechanism can virtualize a single PCIe Ethernet controller to appear as multiple PCIe devices:

```
local: |
  #!/bin/sh -e
  # Enable 7 VF on eth1
  echo 7 > /sys/class/net/eth1/device/sriov_numvfs; sleep 2; ifup -a
  exit 0
```
Isolate CPU options

Remove the specified CPUs, as defined by the cpu_number values, from the general kernel SMP balancing and scheduler algorithms. The only way to move a process onto or off an isolated CPU is via the CPU affinity syscalls. cpu_number begins at 0, so the maximum value is 1 less than the number of CPUs on the system:

```
linux:
  system:
    kernel:
      isolcpu: 1,2,3,4,5,6,7 # isolate first cpu 0
```

Repositories

RedHat-based Linux with additional OpenStack repo:

```
linux:
  system:
    ...
  repo:
    rdo-icehouse:
      enabled: true
      pgpcheck: 0
```

Ensure system repository to use czech Debian mirror (default: true) Also pin it’s packages with priority 900:

```
linux:
  system:
    repo:
      debian:
        default: true
        source: "deb http://ftp.cz.debian.org/debian/ jessie main contrib non-free"
        # Import signing key from URL if needed
        key_url: "http://dummy.com/public.gpg"
        pin:
          - pin: 'origin "ftp.cz.debian.org"
            priority: 900
            package: '*'
```

Sometimes better to use one pining rule file, to decrease mistaken ordering. You can use those option system:apt:preferences, which would add opts into /etc/apt/preferences file:

```
parameters:
  linux:
    system:
      apt:
```
**preferences:**
  **enabled:** true
**rules:**
  **100:**
    **enabled:** true
    **name:** 'some origin pin'
    **pin:** 'release o=Debian'
    **priority:** 1100
    **package:** '*'

If you need to add multiple pin rules for one repo, please use new, ordered definition format ('pinning' definition will be in priority to use):

**linux:**
  **system:**
  **repo:**
    **mcp_saltstack:**
      **source:** "deb [arch=amd64] http://repo.saltstack.com/apt/ubuntu/16.04/amd64/2017.7/ xenial main"
      **architectures:** amd64
      **clean_file:** true
      **pinning:**
        **10:**
          **enabled:** true
          **pin:** 'release o=SaltStack'
          **priority:** 50
          **package:** 'libsodium18'
        **20:**
          **enabled:** true
          **pin:** 'release o=SaltStack'
          **priority:** 1100
          **package:** 'i*'

**Note**

For old Ubuntu releases (<xenial) extra packages for apt transport, like apt-transport-https may be required to be installed manually. (Chicken-eggs issue: we need to install packages to reach repo from where they should be installed) Otherwise, you still can try ‘fortune’ and install prereq.packages before any repo configuration, using list of requires in map.jinja.

Disabling any prerequisite packages installation:

You can simply drop any package pre-installation (before system.linux.repo will be processed) via cluster lvl:
Package manager proxy global setup:

```
linux:
system:
  ...  
  repo:
    apt-mk:
      source: "deb http://apt-mk.mirantis.com/ stable main salt"
  ...  
    proxy:
      pkg:
        enabled: true
        ftp: ftp://ftp-proxy-for-apt.host.local:2121
  ...  
# NOTE: Global defaults for any other component that configure proxy on the system.
#       If your environment has just one simple proxy, set it on linux:system:proxy.
#     # fall back system defaults if linux:system:proxy:pkg has no protocol specific entries
# as for https and http
ftp: ftp://proxy.host.local:2121
http: http://proxy.host.local:3142
https: https://proxy.host.local:3143
```

Package manager proxy setup per repository:

```
linux:
system:
  ...  
  repo:
    debian:
      source: "deb http://apt-mk.mirantis.com/ stable main salt"
  ...  
    apt-mk:
      source: "deb http://apt-mk.mirantis.com/ stable main salt"
      # per repository proxy
      proxy:
        enabled: true
      http: http://maas-01:8080
      https: http://maas-01:8080
  ...  
  proxy:
    # package manager fallback defaults
    # used if linux:system:repo:apt-mk:proxy has no protocol specific entries
    pkg:
```
enabled: true
ftp: ftp://proxy.host.local:2121
#http: http://proxy.host.local:3142
#https: https://proxy.host.local:3143

...  
# global system fallback system defaults
ftp: ftp://proxy.host.local:2121
http: http://proxy.host.local:3142
https: https://proxy.host.local:3143

Remove all repositories:

```yaml
linux:
  system:
    purge_repos: true
```

Refresh repositories metada, after configuration:

```yaml
linux:
  system:
    refresh_repos_meta: true
```

Setup custom apt config options:

```yaml
linux:
  system:
    apt:
      config:
        compression-workaround:
        "Acquire::CompressionTypes::Order": "gz"
        docker-clean:
        "DPkg::Post-Invoke":
        - "rm -f /var/cache/apt/archives/*.deb /var/cache/apt/archives/partial/*.deb /var/cache/apt/*.bin || true"
        "APT::Update::Post-Invoke":
        - "rm -f /var/cache/apt/archives/*.deb /var/cache/apt/archives/partial/*.deb /var/cache/apt/*.bin || true"
```

RC
rc.local example

```bash
linux:
  system:
    rc:
      local: |
        #!/bin/sh -e
        #
        # rc.local
        #
        # This script is executed at the end of each multiuser runlevel.
```
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
exit 0

Prompt
Setting prompt is implemented by creating `/etc/profile.d/prompt.sh`. Every user can have different prompt:

```bash
linux:
  system:
    prompt:
      root: \n\\033[0;37m\\D{%y/%m/%d %H:%M:%S} $(hostname -f)\\n\\e[0m\\n\\[\e[1;31m\\]
      default: \n\\D{%y/%m/%d %H:%M:%S} $(hostname -f)\\n\\u@\\h:\w
```

On Debian systems, to set prompt system-wide, it’s necessary to remove setting PS1 in `/etc/bash.bashrc` and `~/.bashrc`, which comes from `/etc/skel/.bashrc`. This formula will do this automatically, but will not touch existing user’s `~/.bashrc` files except root.

Bash
Fix bash configuration to preserve history across sessions like ZSH does by default:

```bash
linux:
  system:
    bash:
      preserve_history: true
```

Login banner message
/`etc/issue` is a text file which contains a message or system identification to be printed before the login prompt. It may contain various `@`char and char sequences, if supported by the getty-type program employed on the system.

Setting logon banner message is easy:

```bash
linux:
  system:
    banner:
      enabled: true
      contents: |
      UNAUTHORIZED ACCESS TO THIS SYSTEM IS PROHIBITED

      You must have explicit, authorized permission to access or configure this device. Unauthorized attempts and actions to access or use this system may
result in civil and/or criminal penalties.
All activities performed on this system are logged and monitored.

Message of the day
pam_motp from package libpam-modules is used for dynamic messages of the day. Setting custom motd will clean up existing ones.
Setting static motd will replace existing /etc/motd and remove scripts from /etc/update-motd.d.
Setting static motd:

```
linux:
  system:
    motd: |
      UNAUTHORIZED ACCESS TO THIS SYSTEM IS PROHIBITED

      You must have explicit, authorized permission to access or configure this device. Unauthorized attempts and actions to access or use this system may result in civil and/or criminal penalties.
      All activities performed on this system are logged and monitored.
```

Setting dynamic motd:

```
linux:
  system:
    motd:
      - release: |
        #!/bin/sh
        [ -r /etc/lsb-release ] && . /etc/lsb-release
        if [ -z "$DISTRIBUTION" ] && [ -x /usr/bin/lsb_release ]; then
          # Fall back to using the very slow lsb_release utility
          DISTRIB_DESCRIPTION=$(lsb_release -s -d)
        fi
        printf "Welcome to %s (%s %s %s)\n" "$DISTRIBUTION" "$(uname -o)" "$(uname -r)" "$(uname -m)"
      - warning: |
        #!/bin/sh
        printf "This is [company name] network.
        Unauthorized access strictly prohibited.
```

Services
Stop and disable the linux service:

```
linux:
  system:
    service:
      apt-daily.timer:
        status: dead
```
Override systemd service unit:

```
parameters:

linux:
  system:
    service:
      tgt:
        name: tgt
        status: running
        enabled: True
        override:
          50:
            target: tgt.service.d
            name: bind
            content: |
            [Service]
            ExecStart=
            ExecStart=/usr/sbin/tgtd -f --iscsi portal=${_param:single_address}:3260
```

Possible statuses are dead (disable service by default), running (enable service by default), enabled, disabled:

Linux with the atop service:

```
linux:
  system:
    atop:
      enabled: true
      interval: 20
      logpath: "/var/log/atop"
      outfile: "/var/log/atop/daily.log"
```

Linux with the mcelog service:

```
linux:
  system:
    mcelog:
      enabled: true
      logging:
        syslog: true
        syslog_error: true
```

RHEL / CentOS

Currently, update-motd is not available for RHEL. So there is no native support for dynamic motd. You can still set a static one, with a different pillar structure:
Haveged

If you are running headless server and are low on entropy, you may set up Haveged:

```yaml
linux:
  system:
    motd: |
      This is [company name] network.
      Unauthorized access strictly prohibited.

haveged:
  enabled: true
```

Linux network

Linux with network manager:

```yaml
linux:
  network:
    enabled: true
    network_manager: true
```

Execute `linux.network.interface` state without `ifupdown` activity:

```bash
salt-call linux.network.interface pillar='{"linux":{"network":{"noifupdown":true}}}'
```

Linux with default static network interfaces, default gateway interface and DNS servers:

```yaml
linux:
  network:
    enabled: true
    interface:
      eth0:
        enabled: true
        type: eth
        address: 192.168.0.102
        netmask: 255.255.255.0
        gateway: 192.168.0.1
        name_servers:
          - 8.8.8.8
          - 8.8.4.4
        mtu: 1500
```

Linux with bonded interfaces and disabled NetworkManager:
Linux:

```yaml
linux:
  network:
    enabled: true
    interface:
      eth0:
        type: eth
      ...
      eth1:
        type: eth
      ...
    bond0:
      enabled: true
      type: bond
      address: 192.168.0.102
      netmask: 255.255.255.0
      mtu: 1500
      use_in:
        - interface: ${linux:interface:eth0}
        - interface: ${linux:interface:eth0}
    network_manager:
      disable: true
```

Linux with VLAN interface_params:

```yaml
linux:
  network:
    enabled: true
    interface:
      vlan69:
        type: vlan
        use_interfaces:
          - interface: ${linux:interface:bond0}
```

Linux with wireless interface parameters:

```yaml
linux:
  network:
    enabled: true
    gateway: 10.0.0.1
    default_interface: eth0
    interface:
      wlan0:
        type: eth
        wireless:
          essid: example
          key: example_key
          security: wpa
          priority: 1
```
Linux networks with routes defined:

```yaml
linux:
  network:
    enabled: true
    gateway: 10.0.0.1
    default_interface: eth0
  interface:
    eth0:
      type: eth
      route:
        default:
          address: 192.168.0.123
          netmask: 255.255.255.0
          gateway: 192.168.0.1
```

Native Linux Bridges:

```yaml
linux:
  network:
    interface:
      eth1:
        enabled: true
        type: eth
        proto: manual
        up_cmds:
          - ip address add 0/0 dev $IFACE
          - ip link set $IFACE up
        down_cmds:
          - ip link set $IFACE down
    br-ex:
      enabled: true
      type: bridge
      address: ${linux:network:host:public_local:address}
      netmask: 255.255.255.0
      use_interfaces:
        - eth1
```

Open vSwitch Bridges:

```yaml
linux:
  network:
    bridge: openvswitch
  interface:
    eth1:
      enabled: true
      type: eth
      proto: manual
```
**up_cmds:**
- ip address add 0/0 dev $IFACE
- ip link set $IFACE up

**down_cmds:**
- ip link set $IFACE down

**br-ex:**
- **enabled:** true
- **type:** bridge
- **address:** ${linux:network:host:public_local:address}
- **netmask:** 255.255.255.0

**use_interfaces:**
- eth1

**br-prv:**
- **enabled:** true
- **type:** ovs_bridge
- **mtu:** 65000

**br-ens7:**
- **enabled:** true
- **name:** br-ens7
- **type:** ovs_bridge
- **proto:** manual
- **mtu:** 9000

**use_interfaces:**
- ens7

**patch-br-ens7-br-prv:**
- **enabled:** true
- **name:** ens7-prv
- **ovs_type:** ovs_port
- **type:** ovs_port
- **bridge:** br-ens7
- **port_type:** patch
- **peer:** prv-ens7
- **tag:** 109 # [ ] to unset a tag
- **mtu:** 65000

**patch-br-prv-br-ens7:**
- **enabled:** true
- **name:** prv-ens7
- **bridge:** br-prv
- **ovs_type:** ovs_port
- **type:** ovs_port
- **port_type:** patch
- **peer:** ens7-prv
- **tag:** 109
- **mtu:** 65000

**ens7:**
- **enabled:** true
- **name:** ens7
- **proto:** manual
- **ovs_port_type:** OVSPort
Debian manual proto interfaces

When you are changing interface proto from static in up state to manual, you may need to flush ip addresses. For example, if you want to use the interface and the ip on the bridge. This can be done by setting the ipflush_onchange to true.

```yaml
linux:
  network:
    interface:
      eth1:
        enabled: true
        type: eth
        proto: manual
        mtu: 9100
        ipflush_onchange: true
```

Debian static proto interfaces

When you are changing interface proto from dhcp in up state to static, you may need to flush ip addresses and restart interface to assign ip address from a managed file. For example, if you want to use the interface and the ip on the bridge. This can be done by setting the ipflush_onchange with combination restart_on_ipflush param set to true.

```yaml
linux:
  network:
    interface:
      eth1:
        enabled: true
        type: eth
        proto: static
        address: 10.1.0.22
        netmask: 255.255.255.0
        ipflush_onchange: true
        restart_on_ipflush: true
```

Concatinating and removing interface files

Debian based distributions have `/etc/network/interfaces.d/` directory, where you can store configuration of network interfaces in separate files. You can concatenate the files to the defined destination when needed, this operation removes the file from the `/etc/network/interfaces.d/`. If you just need to remove iface files, you can use the remove_iface_files key.

```yaml
linux:
  network:
    concat_iface_files:
```

---

©2020, Mirantis Inc.
- `src`: `/etc/network/interfaces.d/50-cloud-init.cfg`
  
  - `dst`: '/etc/network/interfaces'
  
  - `remove_iface_files`:
    - '/etc/network/interfaces.d/90-custom.cfg'

Configure DHCP client

None of the keys is mandatory, include only those you really need. For full list of available options under `send`, `supersede`, `prepend`, `append` refer to `dhcp-options(5)`.

```
linux:
  network:
    dhclient:
      enabled: true
    backoff_cutoff: 15
    initial_interval: 10
    reboot: 10
    retry: 60
    select_timeout: 0
    timeout: 120
    send:
      - `option`: host-name
        `declaration`: "= gethostname()"
    supersede:
      - `option`: host-name
        `declaration`: "spaceship"
      - `option`: domain-name
        `declaration`: "domain.home"
    #- option: arp-cache-timeout
    #  declaration: 20
    prepend:
      - `option`: domain-name-servers
        `declaration`:
          - 8.8.8.8
          - 8.8.4.4
      - `option`: domain-search
        `declaration`:
          - example.com
          - eng.example.com
    # append:
    #  - option: domain-name-servers
    #    declaration: 127.0.0.1
    # ip or subnet to reject dhcp offer from
    reject:
      - 192.33.137.209
      - 10.0.2.0/24
    request:
      - subnet-mask
      - broadcast-address
```
- time-offset
- routers
- domain-name
- domain-name-servers
- domain-search
- host-name
- dhcp6.name-servers
- dhcp6.domain-search
- dhcp6.fqdn
- dhcp6.sntp-servers
- netbios-name-servers
- netbios-scope
- interface-mtu
- rfc3442-classless-static-routes
- ntp-servers

require:
- subnet-mask
- domain-name-servers

# if per interface configuration required add below

interface:
  ens2:
    initial_interval: 11
    reject:
      - 192.33.137.210
  ens3:
    initial_interval: 12
    reject:
      - 192.33.137.211

Linux network systemd settings:

```yaml
linux:
  network:
    ...
  systemd:
    link:
      10-iface-dmz:
        Match:
          MACAddress: c8:5b:67:fa:1a:af
          OriginalName: eth0
        Link:
          Name: dmz0
    netdev:
      20-bridge-dmz:
        match:
          name: dmz0
        network:
          mescription: bridge
```
**bridge**: br-dmz0

**network**:  
# works with lowercase, keys are by default capitalized
**40-dhcp**:  
**match**:  
**name**:  
**network**:  
**DHCP**: yes

Configure global environment variables

Use `/etc/environment` for static system wide variable assignment after boot. Variable expansion is frequently not supported.

```
linux:  
    system:  
        env:  
            BOB_VARIABLE: Alice  
...  
            BOB_PATH:  
                - /srv/alice/bin  
                - /srv/bob/bin  
...  
            ftp_proxy: none  
            https_proxy: ${linux:system:proxy:https}  
            no_proxy:  
                - 192.168.0.80  
                - 192.168.1.80  
                - .domain.com  
                - .local  
...  
# NOTE: global defaults proxy configuration.  
proxy:  
    ftp: ftp://proxy.host.local:2121  
    http: http://proxy.host.local:3142  
    https: https://proxy.host.local:3143  
    noproxy:  
                - .domain.com  
                - .local  
```

Configure the profile.d scripts

The profile.d scripts are being sourced during `.sh` execution and support variable expansion in opposite to `/etc/environment` global settings in `/etc/environment`.  

```
linux:  
    system:  
        profile:  
```
Configure login.defs parameters

```bash
locales: |
    export LANG=C
    export LC_ALL=C
...
vi_flavors.sh: |
    export PAGER=view
    export EDITOR=vim
    alias vi=vim
shell_locales.sh: |
    export LANG=en_US
    export LC_ALL=en_US.UTF-8
shell_proxies.sh: |
    export FTP_PROXY=ftp://127.0.3.3:2121
    export NO_PROXY='.local'
```

<opt_name> is a configurational option defined in ‘man login.defs’. <opt_name> is case sensitive, should be UPPERCASE only!

Linux with hosts

Parameter purge_hosts will enforce whole /etc/hosts file, removing entries that are not defined in model except defaults for both IPv4 and IPv6 localhost and hostname as well as FQDN.

We recommend using this option to verify that /etc/hosts is always in a clean state. However it is not enabled by default for security reasons.

```bash
linux:
    network:
        purge_hosts: true
        host:
            # No need to define this one if purge_hosts is true
            hostname:
                address: 127.0.1.1
                names:
                    - ${linux:network:fqdn}
                    - ${linux:network:hostname}
            node1:
                address: 192.168.10.200
                names:
                    - node2.domain.com
```
Linux with hosts collected from mine

All DNS records defined within infrastructure are passed to the local hosts records or any DNS server. Only hosts with the grain parameter set to true will be propagated to the mine.

```yaml
linux:
  network:
    purge_hosts: true
    mine_dns_records: true
    host:
      node1:
        address: 192.168.10.200
        grain: true
        names: 
          - node2.domain.com
          - service2.domain.com
```

Set up resolvconf's basic resolver info, e.g. nameservers, search/domain and options:

```yaml
linux:
  network:
    resolv:
      dns:
        - 8.8.4.4
        - 8.8.8.8
      domain: my.example.com
      search:
        - my.example.com
        - example.com
      options:
        - ndots: 5
        - timeout: 2
        - attempts: 2
```

Set up custom TX queue length for tap interfaces:

```yaml
linux:
  network:
    tap_custom_txqueuelen: 10000
```

DPDK OVS interfaces
DPDK OVS NIC

```yaml
linux:
  network:
    bridge: openvswitch
    dpdk:
      enabled: true
      driver: uio/vfio
    openvswitch:
      pmd_cpu_mask: "0x6"
      dpdk_socket_mem: "1024,1024"
      dpdk_lcore_mask: "0x400"
      memory_channels: 2
  interface:
    dpkd0:
      name: ${_param:dpdk_nic}
      pci: 0000:06:00.0
      driver: igb_uio/vfio-pci
      enabled: true
      type: dpdk_ovs_port
      n_rxq: 2
      pmd_rxq_affinity: "0:1,1:2"
      bridge: br-prv
      mtu: 9000
      br-prv:
        enabled: true
        type: dpdk_ovs_bridge

DPDK OVS Bond

```
```yaml
  type: dpdk_ovs_port
  n_rxq: 2
  pmd_rxq_affinity: "0:1,1:2"
  mtu: 9000

  dpdk_first_nic:
    name: ${_param:primary_first_nic}
    pci: 0000:05:00.0
    driver: igb_uio/vfio-pci
    bond: dpdkbond0
    enabled: true
    type: dpdk_ovs_port
    n_rxq: 2
    pmd_rxq_affinity: "0:1,1:2"
    mtu: 9000

  dpdkbond0:
    enabled: true
    bridge: br-prv
    type: dpdk_ovs_bond
    mode: active-backup
    br-prv:
      enabled: true
      type: dpdk.ovs.bridge

DPDK OVS LACP Bond with vlan tag

  linux:
    network:
      bridge: openvswitch
    dpdk:
      enabled: true
      driver: uio
    openvswitch:
      pmd_cpu_mask: "0x6"
      dpdk_socket_mem: "1024,1024"
      dpdk_lcore_mask: "0x400"
      memory_channels: "2"
    interface:
      eth3:
        enabled: true
        type: eth
        proto: manual
        name: ${_param:tenant_first_nic}
      eth4:
        enabled: true
        type: eth
        proto: manual
        name: ${_param:tenant_second_nic}
    dpdk0:
```
DPDK OVS bridge for VXLAN

If VXLAN is used as tenant segmentation, IP address must be set on br-prv.

```yaml
name: ${_param:tenant_first_nic}
pci: "0000:81:00.0"
driver: igb_uio
bond: bond1
enabled: true
type: dpdk_ovs_port
n_rxq: 2
dpdk1:
  name: ${_param:tenant_second_nic}
  pci: "0000:81:00.1"
  driver: igb_uio
  bond: bond1
  enabled: true
  type: dpdk_ovs_port
  n_rxq: 2
bond1:
  enabled: true
  bridge: br-prv
  type: dpdk_ovs_bond
  mode: balance-slb
br-prv:
  enabled: true
  type: dpdk_ovs_bridge
  tag: ${_param:tenant_vlan}
  address: ${_param:tenant_address}
  netmask: ${_param:tenant_network_netmask}
```

DPDK OVS bridge with Linux network interface

```yaml
linux:
  network:
    ...
    interface:
      br-prv:
        enabled: true
        type: dpdk_ovs_bridge
        address: 192.168.50.0
        netmask: 255.255.255.0
        tag: 101
        mtu: 9000
```
interface:
  eth0:
    type: eth
    ovs_bridge: br-prv
...
  br-prv:
    enabled: true
    type: dpdk_ovs_bridge
...

Linux storage

Linux with mounted Samba:

```
linux:
  storage:
    enabled: true
    mount:
      samba1:
        - enabled: true
        - path: /media/myuser/public/
        - device: //192.168.0.1/storage
        - file_system: cifs
        - options: guest,uid=myuser,iocharset=utf8,file_mode=0777,dir_mode=0777,noperm
```

NFS mount:

```
linux:
  storage:
    enabled: true
    mount:
      nfs_glance:
        enabled: true
        path: /var/lib/glance/images
        device: 172.16.10.110:/var/nfs/glance
        file_system: nfs
        opts: rw,sync
```

File swap configuration:

```
linux:
  storage:
    enabled: true
    swap:
      file:
        enabled: true
        engine: file
```
device: /swapfile
size: 1024

Partition swap configuration:

linux:
  storage:
    enabled: true
  swap:
    partition:
      enabled: true
      engine: partition
      device: /dev/vg0/swap

LVM group vg1 with one device and data volume mounted into /mnt/data.

parameters:
  linux:
    storage:
      mount:
        data:
          enabled: true
          device: /dev/vg1/data
          file_system: ext4
          path: /mnt/data
    lvm:
      vg1:
        enabled: true
        devices:
          - /dev/sdb
        volume:
          data:
            size: 40G
            mount: ${linux:storage:mount:data}

# When set they will take precedence over filters aget from volume groups.

lvm_filters:
  10:
    enabled: True
    value: "a|loop|"
  20:
    enabled: True
    value: "r|/dev/hdc|"
  30:
    enabled: True
    value: "a|/dev/ide|"
  40:
    enabled: True
    value: "r|.*|"
Create partitions on disk. Specify size in MB. It expects empty disk without any existing partitions. Set startsector=1 if you want to start partitions from 2048.

```yaml
linux:
  storage:
    disk:
      first_drive:
        startsector: 1
        name: /dev/loop1
        type: gpt
      partitions:
        - size: 200 # size in MB
          type: fat32
        - size: 300 # size in MB
          mkfs: True
          type: xfs
      /dev/vda1:
        partitions:
          - size: 5
            type: ext2
          - size: 10
            type: ext4
```

Multipath with Fujitsu Eternus DXL:

```yaml
parameters:
  linux:
    storage:
      multipath:
        enabled: true
        blacklist_devices:
          - /dev/sda
          - /dev/sdb
      backends:
        - fujitsu_eternus_dx1
```

Multipath with Hitachi VSP 1000:

```yaml
parameters:
  linux:
    storage:
      multipath:
        enabled: true
        blacklist_devices:
          - /dev/sda
          - /dev/sdb
      backends:
        - hitachi_vsp1000
```
Multipath with IBM Storwize:

```yaml
parameters:
  linux:
    storage:
      multipath:
        enabled: true
      blacklist_devices:
        - /dev/sda
        - /dev/sdb
    backends:
      - ibm_storwize
```

Multipath with multiple backends:

```yaml
parameters:
  linux:
    storage:
      multipath:
        enabled: true
      blacklist_devices:
        - /dev/sda
        - /dev/sdb
        - /dev/sdc
        - /dev/sdd
    backends:
      - ibm_storwize
      - fujitsu_ eternus_dxl
      - hitachi_vsp1000
```

PAM LDAP integration:

```yaml
parameters:
  linux:
    system:
      auth:
        enabled: true
      mkhomedir:
        enabled: true
        umask: 0027
    ldap:
      enabled: true
      binddn: cn=bind,ou=service_users,dc=example,dc=com
      bindpw: secret
      uri: ldap://127.0.0.1
      base: ou=users,dc=example,dc=com
      ldap_version: 3
      pagesize: 65536
```
referrals: off
filter:
  passwd: (&(&(objectClass=person)(uidNumber=*)))(unixHomeDirectory=*))
  shadow: (&(&(objectClass=person)(uidNumber=*)))(unixHomeDirectory=*))
  group: (&(objectClass=group)(gidNumber=*))

PAM duo 2FA integration

parameters:
  linux:
    system:
      auth:
        enabled: true
      duo:
        enabled: true
        duo_host: localhost
        duo_ikey: DUO-INTEGRATION-KEY
        duo_skey: DUO-SECRET-KEY

duo package version may be specified (optional)

  linux:
    system:
      package:
        duo-unix:
          version: 1.10.1-0

Disabled multipath (the default setup):

parameters:
  linux:
    storage:
      multipath:
        enabled: false

Linux with local loopback device:

  linux:
    storage:
      loopback:
        disk1:
          file: /srv/disk1
          size: 50G

External config generation
You are able to use config support metadata between formulas and only generate configuration files for external use, for example, Docker, and so on.

```yaml
parameters:
  linux:
    system:
      config:
        pillar:
          jenkins:
            master:
              home: /srv/volumes/jenkins
              approved_scripts:
                - method java.net.URL openConnection
              credentials:
                - type: username_password
                  scope: global
                  id: test
                  desc: Testing credentials
                  username: test
                  password: test

Netconsole Remote Kernel Logging

Netconsole logger can be configured for the configfs-enabled kernels (CONFIG_NETCONSOLE_DYNAMIC must be enabled). The configuration applies both in runtime (if network is already configured), and on-boot after an interface initialization.

Note

- Receiver can be located only on the same L3 domain (or you need to configure gateway MAC manually).
- The Receiver MAC is detected only on configuration time.
- Using broadcast MAC is not recommended.

```yaml
parameters:
  linux:
    system:
      netconsole:
        enabled: true
        port: 514 (optional)
        loglevel: debug (optional)
        target:
          192.168.0.1:
            interface: bond0
            mac: "ff:ff:ff:ff:ff:ff" (optional)
```
Check network params on the environment

Grab nics and nics states

```bash
salt osd001\* net_checks.get_nics
```

Example of system output:

```
osd001.domain.com:
  |- bond0
  |  - None
  |  - 1e:c8:64:42:23:b9
  |  - 0
  |  - 1500
  |- bond1
  |  - None
  |  - 3c:fd:fe:27:3b:00
  |  - 1
  |  - 9100
  |- forty1
  |  - None
  |  - 3c:fd:fe:27:3b:00
  |  - 1
  |  - 9100
  |- forty2
  |  - None
  |  - 3c:fd:fe:27:3b:02
  |  - 1
  |  - 9100
```

Grab 10G nics PCI addresses for hugepages setup

```bash
salt cmp001\* net_checks.get_ten_pci
```

Example of system output:

```
cmp001.domain.com:
  |- ten1
  |  - 0000:19:00.0
  |- ten2
  |  - 0000:19:00.1
```
Grab ip address for an interface

```
salt cmp001\* net_checks.get_ip iface=one4
```

Example of system output:

```
cmp001.domain.com:
  10.200.177.101
```

Grab ip addresses map

```
salt-call net_checks.nodes_addresses
```

Example of system output:

```
local:
  - cid01.domain.com
    - pxe
      - 10.200.177.91
    - control
      - 10.200.178.91
  - cmn02.domain.com
    - storage_access
      - 10.200.181.67
    - pxe
      - 10.200.177.67
    - control
      - 10.200.178.67
  - cmp010.domain.com
  - cmn02.domain.com
```

---

©2020, Mirantis Inc.
Verify full mesh connectivity

```
salt-call net_checks.ping_check
```

Example of positive system output:

```
['PASSED']
[INFO    ] ['PASSED']
local: True
```

Example of system output in case of failure:

```
FAILED
[ERROR   ] FAILED
['control: 10.0.1.92 -> 10.0.1.224: Failed']
['control: 10.0.1.93 -> 10.0.1.224: Failed']
['control: 10.0.1.51 -> 10.0.1.224: Failed']
['control: 10.0.1.102 -> 10.0.1.224: Failed']
['control: 10.0.1.13 -> 10.0.1.224: Failed']
['control: 10.0.1.81 -> 10.0.1.224: Failed']
local: False
```

For this feature to work, please mark addresses with some role. Otherwise ‘default’ role is assumed and mesh would consist of all addresses on the environment.

Mesh mark is needed only for interfaces which are enabled and have ip address assigned.

Checking dhcp pxe network meaningless, as it is used for salt master vs minion communications, therefore treated as checked.

```
parameters:
  linux:
    network:
      interface:
```
ens3:
   enabled: true
   type: eth
   proto: static
   address: ${_param:deploy_address}
   netmask: ${_param:deploy_network_netmask}
   gateway: ${_param:deploy_network_gateway}
   mesh: pxe

Check pillars for ip address duplicates

salt-call net_checks.verify_addresses

Example of positive system output:

['PASSED']
[INFO    ] ['PASSED']
local:
   True

Example of system output in case of failure:

FAILED. Duplicates found
[ERROR   ] FAILED. Duplicates found
['gtw01.domain.com', 'gtw02.domain.com', '10.0.1.224']
[ERROR   ] ['gtw01.domain.com', 'gtw02.domain.com', '10.0.1.224']
local:
   False

Generate csv report for the env

salt -C 'kvm* or cmp* or osd*' net_checks.get_nics_csv
   | grep '^\' | sed 's/\s/ */g' | grep -Ev ^server
   | sed '1 i server,nic_name,ip_addr,mac_addr,link,mtu,chassis_id,chassis_name,port_mac,port_descr'

Example of system output:

server,nic_name,ip_addr,mac_addr,link,mtu,chassis_id,chassis_name,port_mac,port_descr
cmp010.domain.com,bond0,None,b4:96:91:10:5b:3a,1,1500,,
cmp010.domain.com,bond0.21,10.200.178.110,b4:96:91:10:5b:3a,1,1500,,
cmp010.domain.com,bond0.22,10.200.179.110,b4:96:91:10:5b:3a,1,1500,,
cmp010.domain.com,bond1,None,3c:fd:fe:34:ad:22,0,1500,,
cmp010.domain.com,bond1.24,10.200.181.110,3c:fd:fe:34:ad:22,0,1500,,
cmp010.domain.com,fourty5,3c:fd:fe:34:ad:20,0,9000,,
cmp010.domain.com,fourty6,3c:fd:fe:34:ad:22,0,9000,,
cmp010.domain.com,one1,3c:fd:fe:34:ad:20,0,9000,,
cmp010.domain.com,one2,3c:fd:fe:34:ad:20,0,9000,,

©2020, Mirantis Inc.
Usage

Set MTU of the eth0 network interface to 1400:

```
ip link set dev eth0 mtu 1400```

Read more

- [https://www.archlinux.org/](https://www.archlinux.org/)
MAAS

Usage
Metal as a Service

Sample pillars

Single MAAS service:

```yaml
maas:
  server:
    enabled: true
```

Single MAAS region service [single UI/API]:

```yaml
maas:
  salt_master_ip: 192.168.0.10
  region:
    upstream_proxy:
      address: 10.0.0.1
      port: 8080
      user: username  #OPTIONAL
      password: password  #OPTIONAL
    theme: mirantis
  bind:
    host: 192.168.0.10:5240
    port: 5240
  admin:
    username: exampleuser
    password: examplepassword
    email: email@example.com
  database:
    engine: null
    host: localhost
    name: maasdb
    password: qwqwqw
    username: maas
    enabled: true
  user: mirantis
  token: "89EgtWkX45ddjMYpuL:SqvJxfG87Dr6kVf4Wp:5WLfbUgmm9XOtjxm3V2LUUy7bpCmqmnk"
  fabrics:
    fabric1:
      name: 'tf2'
      description: "Test fabric"
    fabric2:
      name: 'tf2'
      description: "Test fabric2"
  deploy_network:
    name: 'deploy_network'
    description: Fabric for deploy_network
  vlans:
    0:
      name: 'vlan 0'
      description: Deploy VLAN
```
mtu: 1500
dhcp: true

# FIXME: after refactoring domain module, it should be
# fixed exactly for FQDN, not only 'hostname'
primary_rack: "$(linux:network:hostname)"

subnets:
    subnet1:
        fabric: ${maas:region:fabrics:deploy_network:name}
        cidr: 2.2.3.0/24
        gateway_ip: 2.2.3.2
        vlan: 150

        ipranges:
            1:
                start: "2.2.3.20"
                end: "2.2.3.40"
                type: dynamic

            2:
                start: "2.2.3.250"
                end: "2.2.3.250"
                type: reserved

dhcp_snippets:
    test-snippet:
        value: option bootfile-name "tftp://192.168.0.10/snippet";
        description: Test snippet
        enabled: true

    subnet: subnet1
    boot_sources_delete_all_others: true

    boot_sources:
        resources_mirror:
            keyring_file: /usr/share/keyrings/ubuntu-cloudimage-keyring.gpg

    boot_sources_selections:
        xenial:
            url: "http://images.maas.io/ephemeral-v3/" # should be same in boot_sources, or other already defined.
            os: "ubuntu"
            release: "xenial"
            arches: "amd64"
            subarches: "*="
            labels: "*="

    package_repositories:
        Saltstack:
            distributions:
                - trusty
                - main
            arches: amd64
            key: "-----BEGIN PGP PUBLIC KEY BLOCK-----

                mQENBF0pvpgBCADkP656H41i8fpplEEB8leLhugyC2rTEwwSclb8tQNYtUiGdna9
                fuBmScum8uQTrEF5+Um5zkwC7EXTdH1co/+V/fp0txl4X04kcugZefVm5ERfVS
                MA==
                =dtMN

            -----END PGP PUBLIC KEY BLOCK-----"
            enabled: true

    machines:
        machine1_new_schema:
pxe_interface_mac: "11:22:33:44:55:66" # Node will be identified by those mac
interfaces:
  nic01: # could be any, used for iterate only
    type: eth # Not-implemented
    name: eth0 # Override default nic name. Interface to rename will be identified by mac
    mode: "static"
    ip: "2.2.3.19" # ip should be out of reserved subnet range, but still in subnet range
    subnet: "subnet1"
    gateway: "2.2.3.2" # override default gateway from subnet

nic02:
  type: eth # Not-implemented
  subnet: "subnet2"
  mode: "dhcp"

power_parameters:
  power_type: ipmi
  power_address: '192.168.10.10'
  power_user: bmc_user
  power_pass: bmc_password
  # power_password: bmc_password # Old format, please use new one
  #Optional (for legacy HW)
  power_driver: LAN
  distro_series: xenial
  hwe_kernel: hwe-16.04

machine1_old_schema:
interface:
  mac: "11:22:33:44:55:88" # Node will be identified by those mac
  mode: "static"
  ip: "2.2.3.15"
  subnet: "subnet1"
  gateway: "2.2.3.2"

power_parameters:
  power_type: ipmi
  power_address: '192.168.10.10'
  power_user: bmc_user
  power_pass: bmc_password
  # power_password: bmc_password # Old format, please use new one
  #Optional (for legacy HW)
  power_driver: LAN
  distro_series: xenial
  hwe_kernel: hwe-16.04

virsh_example:
pxe_interface_mac: "52:54:00:00:01:01"
interfaces:
  nic01:
    type: eth
    name: eth0
    mac: "52:54:00:00:01:01"

subnet: "${maas:region:subnets:deploy_network:name}" # globally used subnet
mode: "dhcp"

power_parameters:
  power_type: virsh
  power_address: "qemu+tcp://my-kvm-node-hostname/system"
  power_id: "kvm01-pxe01"

devices:
  machine1-ipmi:
    interface:
      ip_address: 192.168.10.10
      subnet: cidr:192.168.10.0/24
commissioning_scripts: /etc/maas/files/commissioning_scripts/00-maas-05-simplify-network-interfaces
maas_config:
  # domain: mydomain.local # This function broken
http_proxy: http://192.168.0.10:3142
commissioning_distro_series: xenial
default_distro_series: xenial
default_osystem: 'ubuntu'
default_storage_layout: lvm
disk_erase_with_secure_erase: true
dnssec_validation: 'no'
enable_third_party_drivers: true
maas_name: cfg01
network_discovery: 'enabled'
active_discovery_interval: '600'
ntp_external_only: true
ntp_servers: 10.10.11.23 10.10.11.24
upstream_dns: 192.168.12.13
enable_http_proxy: true
default_min_hwe_kernel: '
sshprefs:
- 'ssh-rsa ASD...........dfsadfsblah@blah'

Update VLAN:

Note
Vid 0 has default name untagged in the MAAS UI.

maas:
  region:
  fabrics:
    test-fabric:
      description: "Test fabric"
      vlan:
        0:
          description: "Your VLAN 0"
          dhcp: True
        13:
          description: "Your VLAN 13"
          dhcp: False

Create disk schema per machine via maas/client.sls with default lvm schema + default values.

Note
This should be used mostly for custom root partitioning and RAID configuration. For not-root partitions, use salt-formula-linux.
maas:
  region:
    machines:
      server1:
        disk_layout:
          type: lvm
          root_size: 20G
          root_device: vda
          volume_group: vg1
          volume_name: root
          volume_size: 8
          bootable_device: vda

FLAT layout with custom root size:

maas:
  region:
    machines:
      server2:
        disk_layout:
          type: flat
          root_size: 20
          physical_device: vda
          bootable_device: vda

Size specification with % char used is not yet supported.

maas:
  region:
    machines:
      server3:
        disk_layout:
          type: flat
          bootable_device: sda
disk:
  sda:
    type: physical
    partition_schema:
      part1:
        size: 100%
        type: ext4
        mount: '/'

Define more complex layout:

maas:
  region:
machines:
  server3:
    disk_layout:
      type: custom
      bootable_device: vda
disk:
  vda:
    type: physical
    partition_schema:
      part1:
        size: 10G
        type: ext4
        mount: '/'
      part2:
        size: 2G
      part3:
        size: 3G
  vdc:
    type: physical
    partition_schema:
      part1:
        size: 100G
  vdd:
    type: physical
    partition_schema:
      part1:
        size: 100G
raid0:
  type: raid
  level: 10
  devices:
    - vde
    - vdf
  partition_schema:
    part1:
      size: 10G
    part2:
      size: 2G
    part3:
      size: 3G
raid1:
  type: raid
  level: 1
  partitions:
    - vdc-part1
    - vdd-part1
volume_group2:
  type: lvm
  devices:
- raid1
  volume:
    tmp:
      size: 5G
type: ext4
mount: '/tmp'
log:
  size: 7G
  type: ext4
  mount: '/var/log'

Raid setup, 4x HDD:

maas:
  region:
    machines:
      serverWithRaidExample:
        disk_layout:
          type: custom
          bootable_device: sda
disk:
  md0:
    type: raid
    level: 1
devices:
    - sda
    - sdb
    partition_schema:
      part1:
        size: 230G
type: ext4
mount: /
md1:
  type: raid
  level: 1
devices:
  - sdc
  - sdd
  partition_schema:
    part1:
      size: 1890G
type: ext4
mount: /var/lib/libvirt

Raid + LVM setup, 2xSSD + 2xHDD:
Note
This setup lacks the ability run state twice, as of now when disk_partition_present is called, it tries blindly to delete the partition and then recreated. That fails as MAAS rejects remove partition used in RAID/LVM.

```yaml
maas:
  region:
    machines:
      serverWithRaidExample2:
        disk_layout:
          type: custom
          #bootable_device: vgssd-root
        disk:
          sda: &maas_disk_physical_ssd
            type: physical
            partition_schema:
              part1:
                size: 239G
          sdb: *maas_disk_physical_ssd
          sdc: &maas_disk_physical_hdd
            type: physical
            partition_schema:
              part1:
                size: 1990G
          sdd: *maas_disk_physical_hdd
        md0:
          type: raid
          level: 1
          partitions:
            - sda-part1
            - sdb-part1
        md1:
          type: raid
          level: 1
          partitions:
            - sdc-part1
            - sdd-part1
        vgssd:
          type: lvm
          devices:
            - md0
          volume:
            root:
              size: 230G
              type: ext4
```
```yaml
  mount: '/'
vghdd:
  type: lvm
devices:
  - md1
  volume:
    libvirt:
      size: 1800G
type: ext4
    mount: '/var/lib/libvirt'
```

LVM setup using partition

```yaml
maas:
  region:
    machines:
      serverWithLvmExample3:
        disk_layout:
          type: custom
          bootable_device: sda
disk:
  sda:
    type: physical
    partition_schema:
      part1:
        size: 50G
      part2:
        mount: /var/lib/libvirt/images/
        size: 10G
        type: ext4
    vg0:
      partitions:
        - sda-part1
type: lvm
volume:
  root:
    mount: /
    size: 40G
type: ext4
```

Setup image mirror (MAAS boot resources):

```yaml
maas:
mirror:
  enabled: true
image:
  sections:
    bootloaders:
```
Usage of local deb repos and curtin-based variables.

Dict of variables curtin_vars:amd64:xenial: format, which will be passed only to /etc/maas/preseeds/curtin_userdata_amd64_generic_xenial accordingly.

```yaml
keyring: /usr/share/keyrings/ubuntu-cloudimage-keyring.gpg
upstream: http://images.maas.io/ephemeral-v3/daily/
count: 1
# i386 need for pxe
filters: ['arch~(i386|amd64)', 'os~(grub*|pxelinux)']
xenial:
  keyring: /usr/share/keyrings/ubuntu-cloudimage-keyring.gpg
  upstream: http://images.maas.io/ephemeral-v3/daily/
count: 1
  filters: ['release~(xenial)', 'arch~(amd64)', 'subarch~(generic|hwe-16.04$|ga-16.04)']
count: 1
```

Single MAAS cluster service [multiple racks]
maas:
  cluster:
    enabled: true
    role: master/slave

maas:
  cluster:
    enabled: true
    role: master/slave

MAAS region service with backup data:

maas:
  region:
    database:
      initial_data:
        source: cfg01.local
        host: 192.168.0.11

MAAS service power_parameters defintion with OpenStack Nova power_type:

maas:
  region:
    machines:
      cmp1:
        power_type: nova
        power_parameters: # old style, deprecated
          power_nova_id: hostuuid
          power_os_tenantname: tenant
          power_os_username: user
          power_os_password: password
          power_os_authurl: http://url

maas:
  region:
    machines:
      cmp1:
        power_type: nova
        power_parameters: # new style
          nova_id: hostuuid
          os_tenantname: tenant
          os_username: user
          os_password: password
          os_authurl: http://url

Ext pillar from MAAS address pool
Set up the Salt Master node:

```
salt:
  master:
  ext_pillars:
  
  module: cmd_json
params: /usr/share/salt-formulas/env/_modules/maas-IPAM.py --address_pool ${salt:master:pillar:data_dir}/classes/cluster/${_param:cluster_name}/infra/address_pool.yml

salt-call state.apply salt.master
salt '*' saltutil.refresh_pillar
```

Update `infra/address_pool.yml`:

```
parameters:
address_pool:
  external:
    dns_server01: 8.8.8.8
    dns_server02: 8.8.4.4
    upstream_ntp_server: 193.27.208.100
    remote_rsyslog_host: 127.0.0.3

deploy_network:
  address: 192.168.0.0
  netmask: 255.255.255.0
  gateway: 192.168.0.1
  prefix: 24
  vlan: 0

# Static reservation which interfere with maas reserve pool
reserved:
  cmp001_deploy_address: 192.168.0.101
  cmp002_deploy_address: 192.168.0.102
  infra_config_deploy_address: 192.168.0.253
  infra_kvm_node01_deploy_address: 192.168.0.241
  infra_kvm_node02_deploy_address: 192.168.0.242
  infra_kvm_node03_deploy_address: 192.168.0.243
  infra_kvm_node04_deploy_address: 192.168.0.244
  infra_kvm_node05_deploy_address: 192.168.0.245
  infra_kvm_node06_deploy_address: 192.168.0.246
  ldap_ip_address: 192.168.0.249

pool:
# Static reservation out of maas reserved pool
aptly_server_deploy_address: 192.168.0.252
# Dynamic serialization
  cicd_control_node01_deploy_address: dummy
  cicd_control_node02_deploy_address: dummy
  cicd_control_node03_deploy_address: dummy
# Release IP address
openstack_share_node02_proxy_address: ""

cluster_networks:
deploy_network:
  name: 'deploy_network'
```
**cidr**: `${address_pool:deploy_network:address}/${address_pool:deploy_network:prefix}`  
**fabric**: deploy_fabric  
**vlan**: `${address_pool:deploy_network:vlan}`  
**gateway_ip**: `${address_pool:deploy_network:gateway}`  

**ipranges**:  
1:  
  **start**: 192.168.0.30  
  **end**: 192.168.0.80  
  **type**: dynamic  
  **comment**: 'dynamic range'  
2:  
  **start**: 192.168.0.1  
  **end**: 192.168.0.29  
  **type**: reserved  
  **comment**: 'infra reserve'  

**control_network**:  
  **name**: 'control_network'  
  **cidr**: `${address_pool:control_network:address}/${address_pool:control_network:prefix}`  
  **fabric**: control_fabric  
  **vlan**: `${address_pool:control_network:vlan}`  
  **gateway_ip**: `${address_pool:control_network:address}`

---

**Update maas.yml:**

```yaml
maas:
  region:
    fabrics:
      deploy_fabric:
        name: ${cluster_networks:deploy_network:fabric}
        description: 'Fabric for deploy_network'
        vlans:
          0:
            name: 'lan 0'
            description: Deploy VLAN
            dhcp: true
            primary_rack: "${linux:network:hostname}"
      control_fabric:
        name: 'control_fabric'
        description: 'Fabric for control_network'
        vlans:
          0:
            name: ${cluster_networks:control_network:fabric}
            description: Control VLAN
            dhcp: false
            primary_rack: "${linux:network:hostname}"
      mesh_fabric:
        name: ${cluster_networks:mesh_network:fabric}
        description: 'Fabric for mesh_network'
```
```yaml
vlans:
  0:
    name: 'mesh_network'
    description: Mesh VLAN
    dhcp: false
    primary_rack: "${linux:network:hostname}"

subnets:
  deploy_network: ${cluster_networks:deploy_network}
  control_network: ${cluster_networks:control_network}
  mesh_network: ${cluster_networks:mesh_network}
  proxy_network: ${cluster_networks:proxy_network}
```

Populate MAAS with networks:

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

Serialize IP addresses using MAAS network pools:

```
salt-call maasng.sync_address_pool
```

Verify pillar override works:

```
salt-call pillar.get address_pool:deploy_network:pool:openstack_share_node02_deploy_address
```

# Sample output:
# local:
#   192.168.0.81

Test pillars

Mind the PostgreSQL and rsyslog .sls. Database and syslog service are required for MAAS to properly install and work.

- https://github.com/salt-formulas/salt-formula-rsyslog/tree/master/tests/pillar

Module function example

Wait for status of selected machine's:

```
> cat maas/machines/wait_for_machines_ready.sls
...

wait_for_machines_ready:
  module.run:
    - name: maas.wait_for_machine_status
    - kwargs:
      machines:
```
If module run w/o any extra paremeters, `wait_for_machines_ready` will wait for defined in salt machines. In this case, it is usefull to skip some machines:

```
> cat maas/machines/wait_for_machines_deployed.sls

wait_for_machines_ready:
  module.run:
    - name: maas.wait_for_machine_status
      - timeout: 1200 # in seconds
      - kwargs:
        req_status: "Deployed"
        ignore_machines:
          - kvm01 # in case it's broken or whatever
      - require:
        - cmd: maas_login_admin
```

List of available `req_status` defined in global variable:

```
STATUS_NAME_DICT = dict(
    (0, 'New'), (1, 'Commissioning'), (2, 'Failed commissioning'),
    (3, 'Missing'), (4, 'Ready'), (5, 'Reserved'), (10, 'Allocated'),
    (9, 'Deploying'), (6, 'Deployed'), (7, 'Retired'), (8, 'Broken'),
    (11, 'Failed deployment'), (12, 'Releasing'),
    (13, 'Releasing failed'), (14, 'Disk erasing'),
    (15, 'Failed disk erasing'))
```

Read more

- [https://maas.io/](https://maas.io/)
MEMCACHED

Usage

Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering.

Sample metadata

```yaml
memcached:
  server:
    enabled: true
    cache_size: 64
    slabsize: 1m
    bind:
      address: 0.0.0.0
      port: 11211
      protocol: tcp
```

Enable/Disable tcp/udp listener

```yaml
memcached:
  server:
    enabled: true
    cache_size: 64
    slabsize: 2m
    threads: 1
    bind:
      address: 0.0.0.0
      port: 11211
    proto:
      tcp:
        enabled: True
      udp:
        enabled: True
```

Note

The following pillar option is deprecated and does not affect any functionality:

```yaml
bind:
  protocol: tcp
```
Read more

- http://memcached.org/

**Metadata schema specifications for Memcached server**

Core properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache_size</td>
<td>integer</td>
<td>Size for cache, tells Memcached how much RAM to use for item storage (in megabytes).</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables Memcached server service.</td>
</tr>
<tr>
<td>slabsize</td>
<td>string</td>
<td>Set size of each slab page.</td>
</tr>
<tr>
<td>threads</td>
<td>integer</td>
<td>Number of threads to use to process incoming requests.</td>
</tr>
<tr>
<td>address</td>
<td>string</td>
<td>IP address to listen on.</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
<td>Connection port to use.</td>
</tr>
<tr>
<td>proto</td>
<td>object</td>
<td>Listen on TCP/UDP port.</td>
</tr>
</tbody>
</table>
NGINX

Usage

Nginx is an open source reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols, as well as a load balancer, HTTP cache, and a web server (origin server). The nginx project started with a strong focus on high concurrency, high performance and low memory usage.

Sample pillars

Gitlab server setup:

```yaml
nginx:
  server:
    enabled: true
    bind:
      address: '0.0.0.0'
    ports:
      - 80
  site:
    gitlab_domain:
      enabled: true
      type: gitlab
      name: domain
      ssl:
        enabled: true
        key: |
        -----BEGIN RSA PRIVATE KEY-----
        ...
        cert: |
        xyz
        chain: |
        my_chain..
      host:
        name: gitlab.domain.com
        port: 80
```

Simple static HTTP site:

```yaml
nginx:
  server:
  site:
    nginx_static_site01:
      enabled: true
      type: nginx_static
      name: site01
      host:
```
Simple load balancer:

```yaml
name: gitlab.domain.com
port: 80

nginx:
server:
  upstream:
    horizon-upstream:
      backend1:
        address: 10.10.10.113
        port: 8078
        opts: weight=3
      backend2:
        address: 10.10.10.114

site:
  nginx_proxy_openstack_web:
    enabled: true
    type: nginx_proxy
    name: openstack_web
    proxy:
      upstream_proxy_pass: http://horizon-upstream
    host:
      name: 192.168.0.1
      port: 31337
```

Static site with access policy:

```yaml
nginx:
server:
  site:
    nginx_static_site01:
      enabled: true
      type: nginx_static
      name: site01
      access_policy:
        allow:
        - 192.168.1.1/24
        - 127.0.0.1
        deny:
        - 192.168.1.2
        - all
    host:
      name: gitlab.domain.com
      port: 80
```

Simple TCP/UDP proxy:
nginx:
  server:
    stream:
      rabbitmq:
        host:
          port: 5672
        backend:
          server1:
            address: 10.10.10.113
            port: 5672
            least_conn: true
            hash: "$remote_addr consistent"
    unbound:
      host:
        bind: 127.0.0.1
        port: 53
        protocol: udp
        backend:
          server1:
            address: 10.10.10.113
            port: 5353

Simple HTTP proxy:

nginx:
  server:
    site:
      nginx_proxy_site01:
        enabled: true
        type: nginx_proxy
        name: site01
        proxy:
          host: local.domain.com
          port: 80
          protocol: http
        host:
          name: gitlab.domain.com
          port: 80

Simple HTTP proxy with multiple locations:

Note
If proxy part is defined and location is missing /, the proxy part is used. If the / location is defined, it overrides the proxy part.
**nginx:**

```
server:
  site:
    nginx_proxy_site01:
      enabled: true
      type: nginx_proxy
      name: site01
      location:
        /internal/: 
          host: 172.120.10.200
          port: 80
          protocol: http
        /doc/:
          host: 172.10.10.200
          port: 80
          protocol: http
        host:
          name: gitlab.domain.com
          port: 80
```
nginx:
server:
site:
    nginx_proxy_site02:
        enabled: true
        type: nginx_proxy
        name: site02
        proxy:
            websocket: true
            host: local.domain.com
            port: 80
            protocol: http
        host:
            name: gitlab.domain.com
            port: 80

Content filtering proxy:

nginx:
server:
    enabled: true
site:
    nginx_proxy_site03:
        enabled: true
        type: nginx_proxy
        name: site03
        proxy:
            host: local.domain.com
            port: 80
            protocol: http
            filter:
                search: https://www.domain.com
                replace: http://10.10.10.10
        host:
            name: gitlab.domain.com
            port: 80

Proxy with access policy:

nginx:
server:
site:
    nginx_proxy_site01:
        enabled: true
        type: nginx_proxy
        name: site01
        access_policy:
            allow:
Use nginx ngx_http_map_module that creates variables whose values depend on values of other variables.

```
nginx:
  server:
    enabled: true
  map:
    enabled: true
    items:
      mymap:
        enabled: true
        string: input_string
        variable: output_map_variable
        body:
          default:
            value: ""
      example.com:
        value: '1'
      example.org:
        value: '2'
```

Use nginx ngx_http_geo_module module that creates variables with values depending on the client IP address.

```
nginx:
  server:
    enabled: true
  geo:
    enabled: true
    items:
      my_geo_map:
        enabled: true
        variable: output_get_variable
        body:
          default:
```

- 192.168.1.1/24
- 127.0.0.1
deny:
- 192.168.1.2
- all
proxy:
  host: local.domain.com
  port: 80
  protocol: http
  host:
    name: gitlab.domain.com
  port: 80
Use ngx_http_limit_req_module module that is used to limit the request processing rate per a defined key, in particular, the processing rate of requests coming from a single IP address. The limitation is done using the leaky bucket method. The limit_req_module might be configured globally or applied to specific nginx site.

```nginx
server:
  limit_req_module:
  limit_req_zone:
    global_limit_ip_zone:
      key: global_limit_ip_var
      size: 10m
      rate: '1r/s'
      limit_req_status: 503
      limit_req:
        global_limit_zone:
          burst: 5
          enabled: true
```

There is an example to to limit requests to all sites based on IP. In the following example all clients are limited except of 10.12.100.1 with 1 req per second.

1. Create geo instance that will match IP and set limit_action var. “0” - is unlimited, 1 - limited
2. Create a global_geo_limiting_map that will map ip_limit_key to ip_limit_action
3. Create global limit_req_zone called global_limit_zone that limits number of requests to 1r/s
4. Apply global_limit_zone globally to all requests with 5 req burst.

```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:
    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 request limiting to particular site only limit_req should be applied on site level, for example:

nginx:
  server:
    site:
      nginx_proxy_openstack_api_keystone:
        limit_req_module:
          limit_req:
            global_limit_zone:
              burst: 5
              enabled: true

Use ngx_http_limit_conn_module module that is used to set the shared memory zone and the maximum allowed number of connections for a given key value. The limit_conn_module might be configured globally or applied to specific nginx site.

nginx:
  server:
limit_conn_module:
limit_conn_zone:
  global_limit_conn_zone:
    key: 'binary_remote_addr'
    size: 10m
  limit_conn_status: 503
limit_conn:
  global_limit_conn_zone:
    connection: 50
    enabled: true

To apply connection limiting to particular site only limit_conn should be applied on site level, for example:

nginx:
  server:
    site:
      nginx_proxy_openstack_web:
        limit_conn_module:
          limit_conn:
            global_limit_conn_zone:
              connection: 25
              enabled: true

Gitlab server with user for basic auth:

nginx:
  server:
    enabled: true
  user:
    username1:
      enabled: true
      password: magicunicorn
    htpasswd: htpasswd-site1
    username2:
      enabled: true
      password: magicunicorn

Proxy buffering:

nginx:
  server:
    enabled: true
    bind:
      address: '0.0.0.0'
      ports:
        - 80
site:
gitlab_proxy:
  enabled: true
type: nginx_proxy
proxy:
  request_buffer: false
  buffer:
    number: 8
    size: 16
host:
  name: gitlab.domain.com
  port: 80

Let's Encrypt:

nginx:
server:
  enabled: true
bind:
  address: '0.0.0.0'
ports:
  - 443
site:
  gitlab_domain:
    enabled: true
type: gitlab
name: domain
ssl:
  enabled: true
  engine: letsencrypt
host:
  name: gitlab.domain.com
  port: 443

SSL using already deployed key and cert file.

Note
The cert file should already contain CA cert and complete chain.
Advanced SSL configuration, more information about SSL option may be found at http://nginx.org/en/docs/http/ngx_http_ssl_module.html

Note
Prior to nginx 1.11.0, only one type of ecdh curve can be applied in ssl_ecdh_curve directive.

if mode = secure or mode = normal and ciphers or protocols are set, they should have type string. If mode = manual, their type should be dict as shown below.

```
nginx:
server:
  enabled: true
site:
  mysite:
    ssl:
      enabled: true
      mode: 'manual'
      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
protocols:
  TLS1:
    name: 'TLSv1'
    enabled: True
  TLS1_1:
    name: 'TLSv1.1'
    enabled: True
  TLS1_2:
    name: 'TLSv1.2'
    enabled: False
ciphers:
  ECDHE_RSA_AES256_GCM_SHA384:
    name: 'ECDHE-RSA-AES256-GCM-SHA384'
    enabled: True
  ECDHE_ECDSA_AES256_GCM_SHA384:
    name: 'ECDHE-ECDSA-AES256-GCM-SHA384'
    enabled: True
buffer_size: '16k'
crl:
```

```
file: '/etc/ssl/crl.pem'
enabled: False
dhparam:
```
enabled: True
numbits: 2048
use_dsaparam: True
ecdh_curve:
  secp384r1:
    name: 'secp384r1'
    enabled: False
  secp521r1:
    name: 'secp521r1'
    enabled: True
password_file:
  content: 'testcontent22'
  enabled: True
  file: '/etc/ssl/password.key'
prefer_server_ciphers: 'on'
ticket_key:
  enabled: True
  numbytes: 48
resolver:
  address: '127.0.0.1'
  valid_seconds: '500'
  timeout_seconds: '60'
session_tickets: 'on'
stapling: 'off'
stapling_file: '/path/to/stapling/file'
stapling_responder: 'http://ocsp.example.com/
stapling_verify: 'on'
verify_client: 'on'
client_certificate:
  file: '/etc/ssl/client_cert.pem'
  enabled: False
verify_depth: 1
session_cache: 'shared:SSL:15m'
session_timeout: '15m'
strict_transport_security:
  max_age: 16000000
  include_subdomains: False
  always: true
  enabled: true

Setting custom proxy headers:

nginx:
  server:
    enabled: true
  site:
    custom_headers:
      type: nginx_proxy
proxy_set_header:
  Host:
    enabled: true
    value: "$host:8774"
X-Real-IP:
  enabled: true
  value: '$remote_addr'
X-Forwarded-For:
  enabled: true
  value: '$proxy_add_x_forwarded_for'
X-Forwarded-Proto:
  enabled: true
  value: '$scheme'
X-Forwarded-Port:
  enabled: true
  value: '$server_port'

Define site catalog indexes:

nginx:
  server:
    enabled: true
  site:
    nginx_catalog:
      enabled: true
      type: nginx_static
      name: server
      indexes:
        - index.htm
        - index.html
      host:
        name: 127.0.0.1
        port: 80

Define site catalog autoindex:

nginx:
  server:
    enabled: true
  site:
    nginx_catalog:
      enabled: true
      type: nginx_static
      name: server
      autoindex: True
      host:
        name: 127.0.0.1
        port: 80
Nginx stats server (required by collectd nginx plugin) (DEPRECATED):

```yaml
nginx:
    server:
        enabled: true
        site:
            nginx_stats_server:
                enabled: true
                type: nginx_stats
                name: server
                host:
                    name: 127.0.0.1
                    port: 8888
```

or:

```yaml
nginx:
    server:
        enabled: true
        site:
            nginx_stats_server:
                enabled: true
                root: disabled
                indexes: []
                stats: True
                type: nginx_static
                name: stat_server
                host:
                    name: 127.0.0.1
                    address: 127.0.0.1
                    port: 8888
```

Nginx configured to wait for another service/s before starting (currently only with systemd):

```yaml
nginx:
    server:
        wait_for_service:
            - foo-bar.mount
        enabled: true
        site:
            ...
```

Read more

- [http://wiki.nginx.org/Main](http://wiki.nginx.org/Main)
• https://mozilla.github.io/server-side-tls/ssl-config-generator/
NEUTRON

Usage

Neutron is an OpenStack project to provide networking as a service between interface devices (e.g., vNICs) managed by other Openstack services (e.g., nova).

Starting with the Folsom release, Neutron is a core and supported part of the OpenStack platform (for Essex, we were an incubated project, which means use is suggested only for those who really know what they’re doing with Neutron).

Sample pillars

Neutron Server on the controller node

```
neutron:
  server:
    enabled: true
    version: mitaka
    allow_pagination: true
    pagination_max_limit: 100
    api_workers: 2
    rpc_workers: 2
    rpc_state_report_workers: 2
    root_helper_daemon: false
    dhcp_lease_duration: 600
    firewall_driver: iptables_hybrid
    agent_boot_time: 180
    agent_down_time: 30
    dhcp_agents_per_network: 2
    allow_automatic_dhcp_failover: true
    bind:
      address: 172.20.0.1
      port: 9696
    database:
      engine: mysql
      host: 127.0.0.1
      port: 3306
      name: neutron
      user: neutron
      password: pwd
    identity:
      engine: keystone
      host: 127.0.0.1
      port: 35357
      user: neutron
      password: pwd
      tenant: service
      endpoint_type: internal
```
message_queue:
  engine: rabbitmq
  host: 127.0.0.1
  port: 5672
  user: openstack
  password: pwd
  virtual_host: '/openstack'
  rpc_conn_pool_size: 30
  rpc_thread_pool_size: 100
  rpc_response_timeout: 120
metadata:
  host: 127.0.0.1
  port: 8775
  insecure: true
  proto: https
  password: pass
  workers: 2
  audit:
    enabled: false

Note
The pagination is useful to retrieve a large bunch of resources, because a single request may fail (timeout). This is enabled with both parameters allow_pagination and pagination_max_limit as shown above.

Configuration of policy.json file:

```json
neutron:
  server:
    ....
policy:
  create_subnet: 'rule:admin_or_network_owner'
  get_network:queue_id: 'rule:admin_only'
# Add key without value to remove line from policy.json
  'create_network:shared':

Neutron LBaaSv2 enablement
```

```json
neutron:
  server:
    lbaas:
      enabled: true
      providers:
```
octavia:
  engine: octavia
  driver_path: 'neutron_lbaas.drivers.octavia.driver.OctaviaDriver'
  base_url: 'http://127.0.0.1:9876'
avi_adc:
  engine: avinetworks
  driver_path: 'avi_lbaasv2.avi_driver.AviDriver'
  controller_address: 10.182.129.239
  controller_user: admin
  controller_password: Cloudlab2016
  controller_cloud_name: Default-Cloud
avi_adc2:
  engine: avinetworks

Note
If the Contrail backend is set, Opencontrail loadbalancer would be enabled automatically.
In this case lbaas should disabled in pillar:

neutron:
  server:
    lbaas:
      enabled: false

Neutron FWaaSv1 enablement

neutron:
  fwaas:
    enabled: true
    version: ocata
    api_version: v1

Enable CORS parameters

neutron:
  server:
    cors:
      allowed_origin: https:localhost.local,http:localhost.local
      expose_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
      allow_methods: GET,PUT,POST,DELETE,PATCH
      allow_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
allow_credentials: True
max_age: 86400

Neutron VXLAN tenant networks with Network nodes
With DVR for East-West and Network node for North-South.
This use case describes a model utilising VxLAN overlay with DVR. The DVR routers will only be utilized for traffic that is router within the cloud infrastructure and that remains encapsulated. External traffic will be routed to via the network nodes.
The intention is that each tenant will require at least two (2) vrouteers one to be utilised

Neutron Server:

neutron:
  server:
    version: mitaka
    path_mtu: 1500
    bind:
      address: 172.20.0.1
      port: 9696
    database:
      engine: mysql
      host: 127.0.0.1
      port: 3306
      name: neutron
      user: neutron
      password: pwd
    identity:
      engine: keystone
      host: 127.0.0.1
      port: 35357
      user: neutron
      password: pwd
      tenant: service
      endpoint_type: internal
    message_queue:
      engine: rabbitmq
      host: 127.0.0.1
      port: 5672
      user: openstack
      password: pwd
      virtual_host: '/openstack'
    global_physnet_mtu: 9000
  l3_ha: False # Which type of router will be created by default
  dvr: True # disabled for non DVR use case
  backend:
    engine: ml2
  tenant_network_types: "flat,vxlan"
```
external_mtu: 9000
mechanism:
  ovs:
    driver: openvswitch

Network Node:

neutron:
gateway:
  enabled: True
  version: mitaka
  report_interval: 10
  dhcp_lease_duration: 600
  firewall_driver: iptables_hybrid
message_queue:
  engine: rabbitmq
  host: 127.0.0.1
  port: 5672
  user: openstack
  password: pwd
  virtual_host: '/openstack'
rpc_conn_pool_size: 300
rpc_thread_pool_size: 2048
rpc_response_timeout: 3600
local_ip: 192.168.20.20 # br-mesh ip address
dvr: True # disabled for non DVR use case
agent_mode: dvr_snat
metadata:
  host: 127.0.0.1
  password: pass
backend:
  engine: ml2
  tenant_network_types: "flat,vxlan"
mechanism:
  ovs:
    driver: openvswitch
agents:
  dhcp:
    ovs_use_veth: False

Compute Node:

neutron:
compute:
  enabled: True
  version: mitaka
  message_queue:
    engine: rabbitmq
```
Setting mac base address

By default neutron uses fa:16:3f:00:00:00 basement for mac generator. One can set it’s own mac base both for dvr and nondvr cases.

NOTE: dvr_base_mac and base_mac SHOULD differ.

```yaml
neutron:
  server:
    base_mac: fa:16:3f:00:00:00
    dvr_base_mac: fa:16:3f:a0:00:00

gateways:

neutron:
  gateway:
    base_mac: fa:16:3f:00:00:00
    dvr_base_mac: fa:16:3f:a0:00:00

compute nodes:

neutron:
  compute:
    base_mac: fa:16:3f:00:00:00
    dvr_base_mac: fa:16:3f:a0:00:00
```
Disable physnet1 bridge

By default we have external access turned on, so among any physnets in your reclass there would be additional one: physnet1, which is mapped to br-floating

If you need internal nets only without this bridge, remove br-floating and configurations mappings. Disable mappings for this bridge on neutron-servers:

```yaml
neutron:
  server:
    external_access: false
```

gateways:

```yaml
neutron:
  gateway:
    external_access: false
```

compute nodes:

```yaml
neutron:
  compute:
    external_access: false
```

Add additional bridge mappings for OVS bridges

By default we have external access turned on, so among any physnets in your reclass there would be additional one: physnet1, which is mapped to br-floating

If you need to add extra non-default bridge mappings they can be defined separately for both gateways and compute nodes:

gateways:

```yaml
neutron:
  gateway:
    bridge_mappings:
      physnet4: br-floating-internet
```

compute nodes:

```yaml
neutron:
  compute:
    bridge_mappings:
      physnet4: br-floating-internet
```

Specify different mtu values for different physnets

Neutron Server:
**neutron:**

```
server:
  version: mitaka
  backend:
    external_mtu: 1500
    tenant_net_mtu: 9000
    ironic_net_mtu: 9000
```

Neutron VXLAN tenant networks with Network Nodes (non DVR)

This **section describes a network solution that utilises VxLAN overlay** networks without DVR with all routers being managed on the network nodes.

Neutron Server:

```
neutron:
  server:
    version: mitaka
    bind:
      address: 172.20.0.1
      port: 9696
    database:
      engine: mysql
      host: 127.0.0.1
      port: 3306
      name: neutron
      user: neutron
      password: pwd
    identity:
      engine: keystone
      host: 127.0.0.1
      port: 35357
      user: neutron
      password: pwd
      tenant: service
    endpoint_type: internal
    message_queue:
      engine: rabbitmq
      host: 127.0.0.1
      port: 5672
      user: openstack
      password: pwd
      virtual_host: '/openstack'
    global_physnet_mtu: 9000
    l3_ha: True
    dvr: False
    backend:
      engine: ml2
      tenant_network_types= "flat,vxlan"
```
external_mtu: 9000
mechanism:
  ovs:
    driver: openvswitch

Network Node:

neutron:
gateway:
  enabled: True
  version: mitaka
message_queue:
  engine: rabbitmq
  host: 127.0.0.1
  port: 5672
  user: openstack
  password: pwd
  virtual_host: '/openstack'
local_ip: 192.168.20.20 # br-mesh ip address
  dvr: False
agent_mode: legacy
availability_zone: az1
metadata:
  host: 127.0.0.1
  password: pass
backend:
  engine: ml2
  tenant_network_types: "flat,vxlan"
mechanism:
  ovs:
    driver: openvswitch

Compute Node:

neutron:
compute:
  enabled: True
  version: mitaka
message_queue:
  engine: rabbitmq
  host: 127.0.0.1
  port: 5672
  user: openstack
  password: pwd
  virtual_host: '/openstack'
local_ip: 192.168.20.20 # br-mesh ip address
  external_access: False
dvr: False
Neutron VXLAN tenant networks with Network Nodes with DVR

With DVR for East-West and North-South, DVR everywhere, Network node for SNAT.

This section describes a network solution that utilises VxLAN overlay networks with DVR with North-South and East-West. Network Node is used only for SNAT.

Neutron Server:

```yaml
backend:
  engine: ml2
  tenant_network_types: "flat,vxlan"
mechanism:
  ovs:
    driver: openvswitch

neutron:
  server:
    version: mitaka
    bind:
      address: 172.20.0.1
      port: 9696
    database:
      engine: mysql
      host: 127.0.0.1
      port: 3306
      name: neutron
      user: neutron
      password: pwd
    identity:
      engine: keystone
      host: 127.0.0.1
      port: 35357
      user: neutron
      password: pwd
      tenant: service
    endpoint_type: internal
    message_queue:
      engine: rabbitmq
      host: 127.0.0.1
      port: 5672
      user: openstack
      password: pwd
      virtual_host: '/openstack'
  global_physnet_mtu: 9000
  l3_ha: False
  dvr: True
  backend:
    engine: ml2
```
tenant_network_types= "flat,vxlan"
external_mtu: 9000
mechanism:
  ovs:
    driver: openvswitch

Configuring networking-generic-switch ml2 plugin used for bare-metal integration:

neutron:
  server:
    backend:
      mechanism:
        ngs:
          driver: genericswitch
        n_g_s:
          enabled: true
          coordination:
            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

Network Node:

neutron:
  gateway:
    enabled: True
    version: mitaka
    message_queue:
      engine: rabbitmq
      host: 127.0.0.1
      port: 5672
      user: openstack
      password: pwd
      virtual_host: '/openstack'
    local_ip: 192.168.20.20  # br-mesh ip address
    dvr: True
    agent_mode: dvr_snat
    availability_zone: az1
metadata:
    host: 127.0.0.1
    password: pass
backend:
    engine: ml2
    tenant_network_types: "flat,vxlan"
mechanism:
    ovs:
        driver: openvswitch

Compute Node:

neutron:
    compute:
        enabled: True
        version: mitaka
        message_queue:
            engine: rabbitmq
            host: 127.0.0.1
            port: 5672
            user: openstack
            password: pwd
            virtual_host: '/openstack'
    local_ip: 192.168.20.20 # br-mesh ip address
dvr: True
    external_access: True
    agent_mode: dvr
    availability_zone: az1
metadata:
    host: 127.0.0.1
    password: pass
backend:
    engine: ml2
    tenant_network_types: "flat,vxlan"
mechanism:
    ovs:
        driver: openvswitch

Sample Linux network configuration for DVR:

linux:
    network:
        bridge: openvswitch
        interface:
            eth1:
                enabled: true
                type: eth
                mtu: 9000
proto: manual
eth2:
  enabled: true
  type: eth
  mtu: 9000
  proto: manual
eth3:
  enabled: true
  type: eth
  mtu: 9000
  proto: manual
br-int:
  enabled: true
  mtu: 9000
  type: ovs_bridge
br-floating:
  enabled: true
  mtu: 9000
  type: ovs_bridge
float-to-ex:
  enabled: true
  type: ovs_port
  mtu: 65000
  bridge: br-floating
br-mgmt:
  enabled: true
  type: bridge
  mtu: 9000
  address: ${_param:single_address}
  netmask: 255.255.255.0
  use_interfaces:
    - eth1
br-mesh:
  enabled: true
  type: bridge
  mtu: 9000
  address: ${_param:tenant_address}
  netmask: 255.255.255.0
  use_interfaces:
    - eth2
br-ex:
  enabled: true
  type: bridge
  mtu: 9000
  address: ${_param:external_address}
  netmask: 255.255.255.0
  use_interfaces:
    - eth3
**use_ovs_ports:**
- float-to-ex

Additional VXLAN tenant network settings
The default multicast group of 224.0.0.1 only multicasts to a single subnet. Allow overriding it to allow larger underlay network topologies.

**Neutron Server:**

```yaml
neutron:
  server:
    vxlan:
      group: 239.0.0.0/8
      vni_ranges: "2:65535"
```

**Neutron VLAN tenant networks with Network Nodes**

**VLAN tenant provider**

**Neutron Server only:**

```yaml
neutron:
  server:
    version: mitaka
    ... 
    global_physnet_mtu: 9000 
    l3_ha: False 
    dvr: True 
    backend:
      engine: ml2
      tenant_network_types: "flat,vlan" # Can be mixed flat,vlan,vxlan 
      tenant_vlan_range: "1000:2000" 
      external_vlan_range: "100:200" # Does not have to be defined.
      external_mtu: 9000
      mechanism:
        ovs:
          driver: openvswitch
```

**Compute node:**

```yaml
neutron:
  compute:
    version: mitaka
    ... 
    dvr: True 
    agent_mode: dvr 
    external_access: False 
    backend:
```

©2020, Mirantis Inc.
Page 258
engine: ml2
tenant_network_types: "flat,vlan"  # Can be mixed flat,vlan,vxlan
mechanism:
  ovs:
    driver: openvswitch

Neutron with explicit physical networks

Neutron Server only:

neutron:
  server:
    version: ocata
    ...  
  backend:
    engine: ml2
    tenant_network_types: "flat,vlan"  # Can be mixed flat,vlan,vxlan
    ...  
    # also need to configure corresponding bridge_mappings on
    # compute and gateway nodes
    flat_networks_default: '*'  # '*' to allow arbitrary names or '' to disable
    physnets:  # only listed physnets will be configured (overrides physnet1/2/3)
      external:
        mtu: 1500
        types:
          - flat  # possible values - 'flat' or 'vlan'
    sriov_net:
      mtu: 9000  # Optional, defaults to 1500
      vlan_range: '100:200,300:400'  # Optional
      types:
        - vlan
    ext_net2:
      mtu: 1500
      types:
        - flat
        - vlan
    mechanism:
      ovs:
        driver: openvswitch

Advanced Neutron Features (DPDK, SR-IOV)

Neutron OVS DPDK

Enable datapath netdev for neutron openvswitch agent:

neutron:
  server:
    version: mitaka
... 

dpdk: True
...

neutron:
compute:
    version: mitaka
dpdk: True
vhost_mode: client  # options: client|server (default)
vhost_socket_dir: /var/run/openvswitch
backend:
    engine: ml2
...
mechanism:
    ovs:
        driver: openvswitch

Neutron OVS SR-IOV:

neutron:
server:
    version: mitaka
backend:
    engine: ml2
...
mechanism:
    ovs:
        driver: openvswitch
    sriov:
        driver: sriovnicswitch
        # Driver w/ highest number will be placed ahead in the list (default is 0).
        # It's recommended for SR-IOV driver to set an order >0 to get it
        # before (for example) the opendaylight one.
        order: 9

neutron:
compute:
    version: mitaka
...
backend:
    engine: ml2
tenant_network_types: "flat,vlan"  # Can be mixed flat,vlan,vxlan
sriov:
    nic_one:
        devname: eth1
        physical_network: physnet3
mechanism:
### Neutron with LinuxBridge Agents

```yaml
neutron:
  server:
    firewall_driver: iptables
    backend:
      mechanism:
        lb:
          driver: linuxbridge
    ....
  compute:
    backend:
      mechanism:
        lb:
          driver: linuxbridge
    ....
  gateway:
    backend:
      mechanism:
        lb:
          driver: linuxbridge
    agents:
      dhcp:
        interface_driver: linuxbridge
      l3:
        interface_driver: linuxbridge
```

### Neutron with VLAN-aware-VMs

```yaml
neutron:
  server:
    vlan_aware_vms: true
    ....
  compute:
    vlan_aware_vms: true
    ....
  gateway:
    vlan_aware_vms: true
```

### Neutron with BGP VPN (BaGPipe driver)

```yaml
neutron:
  server:
```
```yaml
version: pike
bgp_vpn:
  enabled: true
  driver: bagpipe  # Options: bagpipe/opencontrail/opendaylight[_v2]

compute:
  version: pike
  bgp_vpn:
    enabled: true
    driver: bagpipe  # Options: bagpipe/opencontrail/opendaylight[_v2]
  bagpipe:
    local_address: 192.168.20.20  # IP address for mpls/gre tunnels
    peers: 192.168.20.30  # IP addresses of BGP peers
    autonomous_system: 64512  # Autonomous System number
    enable_rtc: True  # Enable RT Constraint (RFC4684)
  backend:
    ovs_extension:  # for OVS agent only, not supported in SRIOV agent
    bagpipe_bgpvpn:
      enabled: True

Neutron with DHCP agent on compute node

neutron:
  ...
  compute:
    dhcp_agent_enabled: true
  ...

Neutron with DHCP agent disabled on gateway node

neutron:
  ...
  gateway:
    dhcp_agent_enabled: false
  ...

Neutron with metadata agent on compute node

neutron:
  ...
  compute:
    metadata_agent_enabled: true
  ...

Neutron with OVN
Control node:
```
neutron:
server:
  backend:
    engine: ovn
    mechanism:
      ovn:
        driver: ovn
        tenant_network_types: "geneve,flat"
      ovn:
        ovn_l3_scheduler: leastloaded # valid options: chance, leastloaded
        neutron_sync_mode: repair # valid options: log, off, repair
        metadata_enabled: True
    ovn_ctl_opts:
      db-nb-create-insecure-remote: 'yes'
      db-sb-create-insecure-remote: 'yes'

Compute node:

neutron:
compute:
  local_ip: 10.2.0.105
  controller_vip: 10.1.0.101
  external_access: false
backend:
  engine: ovn
  ovsdb_connection: tcp:127.0.0.1:6640
metadata:
  enabled: true
  ovsdb_server_iface: ptcp:6640:127.0.0.1
host: 10.1.0.101
password: unsegreto

Neutron L2 Gateway

Control node:

neutron:
server:
  version: pike
l2gw:
  enabled: true
  periodic_monitoring_interval: 5
  quota_l2_gateway: 20
  # service_provider=<service_type>:<name>:<driver>[:default]
  service_provider: L2GW:OpenDaylight:networking_odl.l2gateway.driver.OpenDaylightL2gwDriver:default
backend:
  engine: ml2

Network/Gateway node:
neutron:
gateway:
  version: pike
l2gw:
  enabled: true
db: true
socket_timeout: 20
ovsdb_hosts:
  # <ovsdb_name>: <ip address>:<port>
  # - ovsdb_name: a user defined symbolic identifier of physical switch
  # - ip address: the address or dns name for the OVSDB server (i.e. pointer to the switch)
  ovsdb1: 10.164.5.33:6632
  ovsdb2: 10.164.4.33:6632

OpenDaylight integration

Control node:

neutron:
server:
  backend:
    opendaylight: true
    router: odl-router_v2
    host: 10.20.0.77
    rest_api_port: 8282
    user: admin
    password: admin
    ovsdb_connection: tcp:127.0.0.1:6639
    ovsdb_interface: native
    enable_websocket: true
    enable_dhcp_service: false
    mechanism:
      ovs:
        driver: opendaylight_v2
        order: 1

Network/Gateway node:

neutron:
gateway:
  backend:
    router: odl-router_v2
    ovsdb_connection: tcp:127.0.0.1:6639
    ovsdb_interface: native
    opendaylight:
      ovsdb_server_iface: ptcp:6639:127.0.0.1
      ovsdb_odl_iface: tcp:10.20.0.77:6640
Compute node:

```yaml
neutron:
  compute:
    opendaylight:
      ovsdb_server_iface: ptcp:6639:127.0.0.1
      ovsdb_odl_iface: tcp:10.20.0.77:6640
      tunnel_ip: 10.1.0.105
      provider_mappings: physnet1:br-floating
```

Service Function Chaining Extension (SFC)

```yaml
neutron:
  server:
    sfc:
      enabled: true
      sfc_drivers:
        - ovs # valid options: ovs, odl, ovn (not implemented yet)
      flow_classifier_drivers:
        - ovs # valid options: see above
```

Neutron Server

Neutron Server with OpenContrail:

```yaml
neutron:
  server:
    backend:
      contrail:
        engine: contrail_discovery_host
        host: 8082
        port: 8082
        user: admin
        password: pass
        tenant: admin
        token: token
```

Neutron Server with Midonet:
neutron:
  server:
    backend:
      engine: midonet
      host: midonet_api_host
      port: 8181
      user: admin
      password: password

Neutron Server with NSX:

neutron:
  server:
    backend:
      engine: vmware
      core_plugin: vmware_nsxv3
      vmware:
        nsx:
          extension_drivers:
            - vmware_nsxv3_dns
          v3:
            api_password: nsx_password
            api_user: nsx_username
            api_managers:
              01:
                scheme: https
                host: 192.168.10.120
                port: '443'
            insecure: true

Neutron Keystone region:

neutron:
  server:
    enabled: true
    version: kilo
    ...
  identity:
    region: RegionTwo
  ...
  compute:
    region: RegionTwo
  ...

Client-side RabbitMQ HA setup:
neutron:
  server:
    ....
    message_queue:
      engine: rabbitmq
      members:
        - host: 10.0.16.1
        - host: 10.0.16.2
        - host: 10.0.16.3
      user: openstack
      password: pwd
      virtual_host: '/openstack'
    ....

Configuring TLS communications

Note
By default, system-wide installed CA certs are used, so cacert_file param is optional, as well as cacert.

- RabbitMQ TLS

  neutron:
    server, gateway, compute:
      message_queue:
        port: 5671
        ssl:
          enabled: True
          (optional) cacert: cert body if the cacert_file does not exists
          (optional) cacert_file: /etc/openstack/rabbitmq-ca.pem
          (optional) version: TLSv1_2

- MySQL TLS

  neutron:
    server:
      database:
        ssl:
          enabled: True
          (optional) cacert: cert body if the cacert_file does not exists
          (optional) cacert_file: /etc/openstack/mysql-ca.pem

- Openstack HTTPS API
**Neutron**:

```
server:
  identity:
    protocol: https
  (optional) cacert_file: /etc/openstack/proxy.pem
```

Enable auditing filter, ie: CADF:

```
neutron:
  server:
    audit:
      enabled: true
      ...
    filter_factory: 'keystonemiddleware.audit:filter_factory'
    map_file: '/etc/pycadf/neutron_api_audit_map.conf'
  ...
compute:
  audit:
    enabled: true
    ...
  filter_factory: 'keystonemiddleware.audit:filter_factory'
  map_file: '/etc/pycadf/neutron_api_audit_map.conf'
  ...
```

Neutron with security groups disabled:

```
neutron:
  server:
    security_groups_enabled: False
  ...
compute:
  security_groups_enabled: False
  ...
gateway:
  security_groups_enabled: False
```

Neutron Client

Neutron networks:

```
neutron:
  client:
    enabled: true
  server:
    identity:
      endpoint_type: internalURL
      network:
        inet1:
```

©2020, Mirantis Inc.
tenant: demo
shared: False
admin_state_up: True
router_external: True
provider_physical_network: inet
provider_network_type: flat
provider_segmentation_id: 2
subnet:
  inet1-subnet1:
    cidr: 192.168.90.0/24
    enable_dhcp: False
inet2:
  tenant: admin
  shared: False
  router_external: True
  provider_network_type: "vlan"
subnet:
  inet2-subnet1:
    cidr: 192.168.92.0/24
    enable_dhcp: False
  inet2-subnet2:
    cidr: 192.168.94.0/24
    enable_dhcp: True
identity1:
  network:
    ...

Neutron routers:

neutron:
  client:
    enabled: true
  server:
    identity:
      endpoint_type: internalURL
    router:
      inet1-router:
        tenant: demo
        admin_state_up: True
        gateway_network: inet
        interfaces:
          - inet1-subnet1
          - inet1-subnet2
    identity1:
      router:
    ...

Neutron security groups:
neutron:
  client:
    enabled: true
  server:
    identity:
      endpoint_type: internalURL
      security_group:
        security_group1:
          tenant: demo
          description: security group 1
          rules:
            - direction: ingress
              ethertype: IPv4
              protocol: TCP
              port_range_min: 1
              port_range_max: 65535
              remote_ip_prefix: 0.0.0.0/0
            - direction: ingress
              ethertype: IPv4
              protocol: UDP
              port_range_min: 1
              port_range_max: 65535
              remote_ip_prefix: 0.0.0.0/0
            - direction: ingress
              protocol: ICMP
              remote_ip_prefix: 0.0.0.0/0
    identity1:
      security_group:
      ...
Note
The network must have flag router:external set to True. Instance port in the stated subnet will be associated with the dynamically generated floating IP.

Enable Neutron extensions (QoS, DNS, etc.)

```yaml
neutron:
  server:
    backend:
      extension:
        dns:
          enabled: True
          host: 127.0.0.1
          port: 9001
          protocol: http
          ...
        qos
          enabled: True
```

Different Neutron extensions for different agents

```yaml
neutron:
  server:
    backend:
      extension: # common extensions for OVS and SRIOV agents
        dns:
          enabled: True
          ...
        qos
          enabled: True
      ovs_extension: # OVS specific extensions
        bagpipe_bgpvpn:
          enabled: True
      sriov_extension: # SRIOV specific extensions
        dummy:
          enabled: True
```

Neutron with Designate

```yaml
neutron:
  server:
    backend:
      extension:
        dns:
```

©2020, Mirantis Inc.
**enabled**: True  
**host**: 127.0.0.1  
**port**: 9001  
**protocol**: http

Enable RBAC for OpenContrail engine

```yaml
neutron:
  server:
    backend:
      engine: contrail
      rbac:
        enabled: True
```

Enhanced logging with logging.conf

By default logging.conf is disabled.

That is possible to enable per-binary logging.conf with new variables:

- `openstack_log_appender`  
  Set to true to enable log_config_append for all OpenStack services

- `openstack_fluentd_handler_enabled`  
  Set to true to enable FluentHandler for all Openstack services

- `openstack_ossyslog_handler_enabled`  
  Set to true to enable OSSysLogHandler for all Openstack services.

Only WatchedFileHandler, OSSysLogHandler, and FluentHandler are available.

Also it is possible to configure this with pillar:

```yaml
neutron:
  server:
    logging:
      log_appender: true
      log_handlers:
        watchedfile:
          enabled: true
        fluentd:
          enabled: true
        ossyslog:
          enabled: true
    ....
compute:
  logging:
    log_appender: true
    log_handlers:
      watchedfile:
```
enabled: true
fluentd:
  enabled: true
ossyslog:
  enabled: true

.....
gateway:
  logging:
    log_appender: true
    log_handlers:
      watchedfile:
        enabled: true
fluentd:
  enabled: true
ossyslog:
  enabled: true

Logging levels pillar example:

neutron:
  server:
    logging:
      log_appender: true
      loggers:
        root:
          level: 'DEBUG'
        neutron:
          level: 'DEBUG'
        amqplib:
          level: 'DEBUG'
        sqlalchemy:
          level: 'DEBUG'
        boto:
          level: 'DEBUG'
        suds:
          level: 'DEBUG'
        eventletwsgi:
          level: 'DEBUG'

....

Neutron server with Memcached caching and security strategy:

neutron:
  server:
    enabled: true
    ...
    cache:
      engine: memcached
Upgrades

Each OpenStack formula provides a set of phases (logical blocks) that help to build a flexible upgrade orchestration logic for particular components. The table below lists the phases and their descriptions:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;app&gt;.upgrade.service_running</td>
<td>Ensure that all services for particular application are enabled for autostart and running</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.service_stopped</td>
<td>Ensure that all services for particular application are disabled for autostart and dead</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pkgs_latest</td>
<td>Ensure that packages used by particular application are installed to latest available version. This will not upgrade data plane packages like qemu and openvswitch as usually 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. Applying this state will not autostart service.</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.render_config</td>
<td>Ensure configuration is rendered actual version.</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.pre</td>
<td>We assume this state is applied on all nodes in the cloud before running upgrade. Only non destructive actions will be applied during this phase. Perform service built in service check like (keystone-manage doctor and nova-status upgrade)</td>
</tr>
<tr>
<td>&lt;app&gt;.upgrade.upgrade.pre</td>
<td>Mostly applicable for data plane nodes. During this phase resources will be gracefully removed from current node if it is allowed. Services for upgraded application will be set to admin disabled state to make sure node will not participate in resources scheduling. For example on gtw nodes this will set all agents to admin disable state and will move all routers to other agents.</td>
</tr>
</tbody>
</table>
This state will basically upgrade application on particular target. Stop services, render configuration, install new packages, run offline dbsync (for ctl), start services. Data plane should not be affected, only OpenStack Python services.

Add services back to scheduling.

This phase should be launched only when upgrade of the cloud is completed. Cleanup temporary files, perform other post upgrade tasks.

Here we will do basic health checks (API CRUD operations, verify do not have dead network agents/compute services)

Enable x509 and SSL communication between Neutron and Galera cluster
By default communication between Neutron and Galera is unsecure.

```
neutron:
  server:
    database:
      x509:
        enabled: True
```

You able to set custom certificates in pillar:

```
neutron:
  server:
    database:
      x509:
        cacert: (certificate content)
        cert: (certificate content)
        key: (certificate content)
```

You can read more about it here: https://docs.openstack.org/security-guide/databases/database-access-control.html
NOVA

Usage

OpenStack Nova provides a cloud computing fabric controller, supporting a wide variety of virtualization technologies, including KVM, Xen, LXC, VMware, and more. In addition to its native API, it includes compatibility with the commonly encountered Amazon EC2 and S3 APIs.

Sample pillars

Controller nodes

Nova services on the controller node:

```yaml
nova:
  controller:
    version: juno
    enabled: true
    security_group: true
    cpu_allocation_ratio: 8.0
    ram_allocation_ratio: 1.0
    disk_allocation_ratio: 1.0
    cross_az_attach: false
    workers: 8
    report_interval: 60
    dhcp_domain: novalocal
    vif_plugging_timeout: 300
    vif_plugging_is_fatal: false
    consoleauth:
      token_ttl: 600
    bind:
      public_address: 10.0.0.122
      public_name: openstack.domain.com
      novncproxy_port: 6080
    database:
      engine: mysql
      host: 127.0.0.1
      port: 3306
      name: nova
      user: nova
      password: pwd
    identity:
      engine: keystone
      host: 127.0.0.1
      port: 35357
      user: nova
      password: pwd
      tenant: service
```
message_queue:
  engine: rabbitmq
  host: 127.0.0.1
  port: 5672
  user: openstack
  password: pwd
  virtual_host: '/openstack'
pci:
  alias:
    alias1:
      device_type: "type-PF"
      name: "a1"
      product_id: "154d"
      vendor_id: "8086"
network:
  engine: neutron
  host: 127.0.0.1
  port: 9696
  extension_sync_interval: 600
identity:
  engine: keystone
  host: 127.0.0.1
  port: 35357
  user: neutron
  password: pwd
  tenant: service
metadata:
  password: password
audit:
  enabled: false
  osapi_max_limit: 500
  barbican:
    enabled: true

Nova services from custom package repository:

nova:
  controller:
    version: juno
    source:
      engine: pkg
      address: http://...

Client-side RabbitMQ HA setup:

nova:
  controller:
message_queue:
    engine: rabbitmq
    members:
        - host: 10.0.16.1
        - host: 10.0.16.2
        - host: 10.0.16.3
    user: openstack
    password: pwd
    virtual_host: '/openstack'

Enable auditing filter, i.e: CADF:

```yaml
nova:
    controller:
        audit:
            enabled: true

        filter_factory: 'keystonemiddleware.audit:filter_factory'
        map_file: '/etc/pycadf/nova_api_audit_map.conf'
```

Enable CORS parameters:

```yaml
nova:
    controller:
        cors:
            expose_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
            allow_methods: GET,PUT,POST,DELETE,PATCH
            allow_headers: X-Auth-Token,X-Openstack-Request-Id,X-Subject-Token
            allow_credentials: True
            max_age: 86400
```

Configuration of the policy.json file:

```yaml
nova:
    controller:
        ....
        policy:
            context_is_admin: 'role:admin or role:administrator'
            'compute:create': 'rule:admin_or_owner'

        # Add key without value to remove line from policy.json
        'compute:create:attach_network':
```

Enable Barbican integration:
Define aliases for PCI devices:

```yaml
nova:
  controller:
    ...
    pci:
      alias: alias1:
        device_type: "type-PF"
        name: "a1"
        product_id: "154d"
        vendor_id: "806"
```

Enable cells update:

```yaml
nova:
  controller:
    update_cells: true
```

Configuring TLS communications

```yaml
nova:
  compute:
    message_queue:
```

• RabbitMQ TLS
**port**: 5671
**ssl**:  
**enabled**: True  
(optional) **cacert**: cert body if the cacert_file does not exists 
(optional) **cacert_file**: /etc/openstack/rabbitmq-ca.pem  
(optional) **version**: TLSv1.2

- **MySQL TLS**
  
  **nova**:  
  **controller**:  
  **database**:  
  **ssl**:  
  **enabled**: True  
  (optional) **cacert**: cert body if the cacert_file does not exists  
  (optional) **cacert_file**: /etc/openstack/mysql-ca.pem

- **Openstack HTTPS API**
  
  Set the https as protocol at nova:compute and nova:controller sections:

  **nova**:  
  **controller**:  
  **identity**:  
  **protocol**: https  
  (optional) **cacert_file**: /etc/openstack/proxy.pem  
  **network**:  
  **protocol**: https  
  (optional) **cacert_file**: /etc/openstack/proxy.pem  
  **glance**:  
  **protocol**: https  
  (optional) **cacert_file**: /etc/openstack/proxy.pem

  **nova**:  
  **compute**:  
  **identity**:  
  **protocol**: https  
  (optional) **cacert_file**: /etc/openstack/proxy.pem  
  **network**:  
  **protocol**: https  
  (optional) **cacert_file**: /etc/openstack/proxy.pem  
  **image**:  
  **protocol**: https  
  (optional) **cacert_file**: /etc/openstack/proxy.pem  
  **ironic**:  
  **protocol**: https  
  (optional) **cacert_file**: /etc/openstack/proxy.pem
Note
Barbican, Cinder, and placement url endpoints are discovering using service catalog.

Compute nodes
Nova controller services on compute node:

```yaml
nova:
  compute:
    version: juno
    enabled: true
    cross_az_attach: false
    disk_cachemodes: network=writeback,block=none
    availability_zone: availability_zone_01
    aggregates:
      - hosts_with_fc
      - hosts_with_ssd
    security_group: true
    resume_guests_state_on_host_boot: False
    preallocate_images: space  # Default is 'none'
    my_ip: 10.1.0.16
    vif_plugging_timeout: 300
    vif_plugging_is_fatal: false
    bind:
      vnc_address: 172.20.0.100
      vnc_port: 6080
      vnc_name: openstack.domain.com
      vnc_protocol: http
    database:
      engine: mysql
      host: 127.0.0.1
      port: 3306
      name: nova
      user: nova
      password: pwd
    identity:
      engine: keystone
      host: 127.0.0.1
      port: 35357
      user: nova
      password: pwd
      tenant: service
    message_queue:
      engine: rabbitmq
      host: 127.0.0.1
      port: 5672
```
```yaml
user: openstack
group: openstack
password: pwd
virtual_host: '/openstack'
image:
  engine: glance
  host: 127.0.0.1
  port: 9292
pci:
  alias:
    alias1:
      device_type: "type-PF"
      name: "a1"
      product_id: "154d"
      vendor_id: "8086"
network:
  engine: neutron
  host: 127.0.0.1
  port: 9696
  identity:
    engine: keystone
    host: 127.0.0.1
    port: 35357
    user: neutron
    password: pwd
    tenant: service
qemu:
  max_files: 4096
  max_processes: 4096
  host: node-12.domain.tld
```

Compute with VMware driver. Each VMware cluster requires a separate process of nova-compute. Each process should have uniq host identifier. However, multiple computes might be running on single host. It is not recommended to have multiple computes running on different hosts that manage the same VMware cluster. To achieve this, Pacemaker/Corosync or Keepalived might be used.

```yaml
nova:
  compute:
    compute_driver: VMwareVCDriver
    vmware:
      host_username: vmware
      host_password: vmware
      cluster_name: vmware_cluster01
      host_ip: 1.2.3.4
```

Group and user to be used for QEMU processes run by the system instance:
nova:  
compute:  
   enabled: true  
...
qemu:  
   user: nova  
   group: cinder  
   dynamic_ownership: 1

Group membership for user nova (upgrade related):

nova:  
compute:  
   enabled: true  
...
user:  
   groups:  
      - libvirt

Nova services on compute node with OpenContrail:

nova:  
compute:  
   enabled: true  
...
   networking: contrail

Nova services on compute node with memcached caching and security strategy:

nova:  
compute:  
   enabled: true  
...
   cache:  
      engine: memcached  
      members:  
         - host: 127.0.0.1  
         port: 11211  
         - host: 127.0.0.1  
         port: 11211  
   security:  
      enabled: true  
      strategy: ENCRYPT  
      secret_key: secret

Client-side RabbitMQ HA setup:
Nova:
  compute:
    ....
    message_queue:
      engine: rabbitmq
      members:
        - host: 10.0.16.1
        - host: 10.0.16.2
        - host: 10.0.16.3
      user: openstack
      password: pwd
      virtual_host: '/openstack'
    ....

Nova with ephemeral configured with Ceph:

nova:
  compute:
    enabled: true
    ...
  ceph:
    ephemeral: yes
    rbd_pool: nova
    rbd_user: nova
    secret_uuid: 03006edd-d957-40a3-ac4c-26cd254b3731
  ....

Nova with ephemeral configured with LVM:

nova:
  compute:
    enabled: true
    ...
  lvm:
    ephemeral: yes
    images_volume_group: nova vg

linux:
  storage:
    lvm:
      nova vg:
        name: nova vg
        devices:
          - /dev/sdf
          - /dev/sdd
          - /dev/sdg
          - /dev/sde
          - /dev/sdc
- /dev/sdj
- /dev/sdh

Enable Barbican integration:

```yaml
nova:
  compute:
  ....
  barbican:
    enabled: true
```

Define aliases for PCI devices:

```yaml
nova:
  compute:
  ...
  pci:
    alias:
      alias1:
        device_type: "type-PF"
        name: "a1"
        product_id: "154d"
        vendor_id: "8086"
```

Nova metadata custom bindings:

```yaml
nova:
  controller:
    enabled: true
  ...
  metadata:
    bind:
      address: 1.2.3.4
      port: 8776
```

Define multipath for nova compute:

```yaml
nova:
  compute:
  ....
  libvirt:
    volume_use_multipath: True
```

Client role
Nova configured with NFS:
nova:
  compute:
    instances_path: /mnt/nova/instances

linux:
  storage:
    enabled: true
  mount:
    nfs_nova:
      enabled: true
      path: ${nova:compute:instances_path}
      device: 172.31.35.145:/data
      file_system: nfs
      opts: rw,vers=3

Nova flavors:

nova:
  client:
    enabled: true
  server:
    identity:
      flavor:
        flavor1:
          flavor_id: 10
          ram: 4096
          disk: 10
          vcpus: 1
        flavor2:
          flavor_id: auto
          ram: 4096
          disk: 20
          vcpus: 2
        identity1:
          flavor:
            ...

Availability zones:

nova:
  client:
    enabled: true
  server:
    identity:
      availability_zones:
        - availability_zone_01
        - availability_zone_02
Aggregates:

```yaml
nova:
  client:
    enabled: true
  server:
    identity:
      aggregates:
        - aggregate1
        - aggregate2
```

Upgrade levels:

```yaml
nova:
  controller:
    upgrade_levels:
    compute: juno

nova:
  compute:
    upgrade_levels:
    compute: juno
```

SR-IOV

Add PciPassthroughFilter into scheduler filters and NICs on specific compute nodes:

```yaml
nova
  controller:

nova:
  compute:
    sriov:
      nic_one:
        devname: eth1
        physical_network: physnet1
```

Note

Parameters located under nova:compute:sriov:<nic_name> are copied to passthrough_whitelist parameter into nova.conf file in appropriate format.

CPU pinning & Hugepages

CPU pinning of virtual machine instances to dedicated physical CPU cores. Hugepages mount point for libvirt.
Custom Scheduler filters

If you have a custom filter, that needs to be included in the scheduler, then you can include it like so:

```
nova:
  controller:
    scheduler_custom_filters:
      - my_custom_filter
```

Then add your custom filter on the end (make sure to include all other ones that you need as well):

```
scheduler_default_filters:
  - DifferentHostFilter
  - SameHostFilter
  - ... 
  - MyCustomFilter
```

Hardware Trip/Unmap Support

To enable TRIM support for ephemeral images (thru nova managed images), libvirt has this option:

```
nova:
  compute:
    libvirt:
      hw_disk_discard: unmap
```

To actually utilize this feature, the following metadata must be set on the image as well, so the SCSI unmap is supported:

```
glance image-update --property hw_scsi_model=virtio-scsi <image>
glance image-update --property hw_disk_bus=scsi <image>
```

Scheduler Host Manager

Specify a custom host manager.

libvirt CPU mode

Allow setting the model of CPU that is exposed to a VM. This allows for better support live migration between hypervisors with different hardware, among other things. Defaults to host-passthrough.

```
nova:
  controller:
    scheduler_host_manager: ironic_host_manager

compute:
  cpu_mode: host-model
```

Nova compute cpu model

```
nova:
  compute:
    cpu_mode: custom
```
libvirt:
  cpu_model: IvyBridge

RNG (Random Number Generator) device path
The path to an RNG (Random Number Generator) device that will be used as the source of
entropy on the host.
The recommended source of entropy is /dev/urandom.
Permitted options include /dev/random, /dev/urandom, and /dev/hwrng.
Default value is /dev/urandom.

nova:
  controller:
    libvirt:
      rng_dev_path: /dev/urandom
  compute:
    libvirt:
      rng_dev_path: /dev/urandom

Nova compute workarounds
Live snapshotting is disabled by default in nova. To enable this, it needs a manual switch.
From manual:

When using libvirt 1.2.2 live snapshots fail intermittently under load
(likely related to concurrent libvirt/qemu operations). This config
option provides a mechanism to disable live snapshot, in favor of cold
snapshot, while this is resolved. Cold snapshot causes an instance
outage while the guest is going through the snapshotting process.

For more information, refer to the bug report:

https://bugs.launchpad.net/nova/+bug/1334398

Configurable pillar data:

nova:
  compute:
    workaround:
      disable_libvirt_livesnapshot: False

Config drive options
See example below on how to configure the options for the config drive:
**nova**:  

```yaml
compute:
  config_drive:
    forced: True  # Default: True
    cdrom: True  # Default: False
    format: iso9660  # Default: vfat
    inject_password: False  # Default: False
```

**Number of concurrent live migrates**

Default is to have no concurrent live migrations (so 1 live-migration at a time).

Excerpt from config options page [https://docs.openstack.org/ocata/config-reference/compute/config-options.html](https://docs.openstack.org/ocata/config-reference/compute/config-options.html):

Maximum number of live migrations to run concurrently. This limit is enforced to avoid outbound live migrations overwhelming the host/network and causing failures. It is not recommended that you change this unless you are very sure that doing so is safe and stable in your environment.

Possible values:

- 0 : treated as unlimited.
- Negative value defaults to 0.
- Any positive integer representing maximum number of live migrations to run concurrently.

To configure this option:

```yaml
nova:
  compute:
    max_concurrent_live_migrations: 1  # (1 is the default)
```

**Live migration with auto converge**

Auto converge throttles down CPU if a progress of on-going live migration is slow [https://docs.openstack.org/ocata/config-reference/compute/config-options.html](https://docs.openstack.org/ocata/config-reference/compute/config-options.html):

```yaml
nova:
  compute:
    libvirt:
      live_migration_permit_auto_converge: False  # (False is the default)
```

```yaml
nova:
  controller:
    libvirt:
      live_migration_permit_auto_converge: False  # (False is the default)
```

**Enhanced logging with logging.conf**

By default logging.conf is disabled.
That is possible to enable per-binary logging.conf with new variables:

- openstack_log_appender
  Set to true to enable log_config_append for all OpenStack services
- openstack_fluentd_handler_enabled
  Set to true to enable FluentHandler for all Openstack services
- openstack_ossyslog_handler_enabled
  Set to true to enable OSSysLogHandler for all Openstack services

Only WatchedFileHandler, OSSysLogHandler, and FluentHandler are available. Also it is possible to configure this with pillar:

```yaml
nova:
  controller:
    logging:
      log_appender: true
      log_handlers:
        watchedfile:
          enabled: true
        fluentd:
          enabled: true
        ossyslog:
          enabled: true

compute:
  logging:
    log_appender: true
    log_handlers:
      watchedfile:
        enabled: true
      fluentd:
        enabled: true
      ossyslog:
        enabled: true
```

The log level might be configured per logger by using the following pillar structure:

```yaml
nova:
  compute:
    logging:
      loggers:
        <logger_name>:
          level: WARNING
```

©2020, Mirantis Inc.
Configure syslog parameters for libvirtd

To configure syslog parameters for libvirtd the below pillar structure should be used with values which are supported by libvirtd. These values might be known from the documentation.

```
loggers:
  <logger_name>:
    level: WARNING
```

Logging controls:

Logging level: 4 errors, 3 warnings, 2 information, 1 debug basically 1 will log everything possible log_level = 3

Logging filters:

A filter allows to select a different logging level for a given category of logs.

The format for a filter is one of:

- x:name
- x:+name
  where name is a string which is matched against source file name, e.g., remote, qemu, or util/json, the optional + prefix tells libvirt to log stack trace for each message matching name, and x is the minimal level where matching messages should be logged:

- 1: DEBUG
- 2: INFO
- 3: WARNING
- 4: ERROR

Multiple filter can be defined in a single @filters, they just need to be separated by spaces.

For example, to only get warning or errors from the remote layer and only errors from the event layer: log_filters="3:remote 4:event"

Logging outputs:

An output is one of the places to save logging information The format for an output can be:

- x:stderr
  Output goes to stderr
• **x:syslog:name**
  Use syslog for the output and use the given name as the ident

• **x:file:file_path**
  output to a file, with the given filepath
  In all case the x prefix is the minimal level, acting as a filter

  • 1: DEBUG
  • 2: INFO
  • 3: WARNING
  • 4: ERROR

Multiple output can be defined, they just need to be separated by spaces. For example, to log all warnings and errors to syslog under the libvirt dident: log_outputs="3:syslog:libvirt"

Log debug buffer size: default 64 The daemon keeps an internal debug log buffer which will be dumped in case of crash or upon receiving a SIGUSR2 signal. This setting allows to override the default buffer size in kilobytes. If value is 0 or less the debug log buffer is deactivated

log_buffer_size = 64

To configure the logging parameters for QEMU, the below pillar structure and logging parameters should be used:

```yaml
nova:
  compute:
    qemu:
      logging:
        handler: logd
      virtlog:
        enabled: true
        level: 4
        filters: '3:remote 3:event'
        outputs: '4:syslog:virtlogd'
        max_clients: 512
        max_size: 2097100
        max_backups: 2
```

Inject password to VM

By default nova blocks up any inject to VM because inject_partition param is equal to -2. If you want to inject password to VM, you will need to define inject_partition greater or equal to -1 and define inject_password to True

For example:

```yaml
nova:
  compute:
    inject_partition: '-1'
    inject_password: True
```
Allow the injection of an admin password for instance only at create and rebuild process.
There is no agent needed within the image to do this. If libguestfs is available on the host, it will be used. Otherwise nbd is used. The file system of the image will be mounted and the admin password, which is provided in the REST API call will be injected as password for the root user. If no root user is available, the instance won’t be launched and an error is thrown. Be aware that the injection is not possible when the instance gets launched from a volume.

Possible values:

- **True**
  Allows the injection

- **False (default)**
  Disallows the injection. Any via the REST API provided admin password will be silently ignored.

Related options:

- **inject_partition**
  Decides about the discovery and usage of the file system. It also can disable the injection at all. (boolean value)

You can read more about injecting the administrator password here: https://docs.openstack.org/nova/queens/admin/admin-password-injection.html

Enable libvirt control channel over TLS

By default TLS is disabled.

Enable TLS transport:

```yaml
compute:
  libvirt:
    tls:
      enabled: True
```

You able to set custom certificates in pillar:

```yaml
nova:
  compute:
    libvirt:
      tls:
        key: (certificate content)
        cert: (certificate content)
        cacert: (certificate content)
        client:
          key: (certificate content)
          cert: (certificate content)
```

Controlling access by tls_allowed_dn_list. Enable an access control list of client certificate Distinguished Names (DNs) which can connect to the TLS port on this server. The default is that DNs are not checked. This list may contain wildcards such as
“C=GB,ST=London,L=London,O=Libvirt Project,CN=*” See the POSIX fnmatch function for the format of the wildcards. Note that if this is an empty list, no client can connect. Note also that GnuTLS returns DNs without spaces after commas between the fields (and this is what we check against), but the openssl x509 tool shows spaces.

```yaml
nova:
  compute:
    libvirt:
      tls:
        tls_allowed_dn_list:
          host1:
            enabled: true
            value: 'C=foo,CN=cmp1'
          host2:
            enabled: true
            value: 'C=foo,CN=cmp2'
```

You can read more about live migration over TLS here: https://wiki.libvirt.org/page/TLSCreateServerCerts

Enable transport + authentication for VNC over TLS

# Only for Queens. Communication between noVNC proxy service and QEMU

By default communication between nova-novncproxy and qemu service is unsecure.

```yaml
compute:
  qemu:
    vnc:
      tls:
        enabled: True
```

```yaml
controller:
  novncproxy:
    # This section responsible for communication between noVNC proxy and client machine
    tls:
      enabled: True
    # This section responsible for communication between nova-novncproxy and qemu service
    vencrypt:
      tls:
        enabled: True
```

You can set custom certificates in pillar:

```yaml
nova:
  compute:
    qemu:
      vnc: 
```

©2020, Mirantis Inc.
You can read more about it here: https://docs.openstack.org/nova/queens/admin/remote-console-access.html

Enable communication between noVNC proxy and client machine over TLS
By default communication between noVNC proxy and client machine is unsecure.

You can read more about it here: https://docs.openstack.org/mitaka/config-reference/dashboard/configure.html

Enable x509 and ssl communication between Nova and Galera cluster
By default communication between Nova and Galera is unsecure.
You can set custom certificates in pillar:

```yaml
nova:
  controller:
    database:
      x509:
        enabled: True
        cacert: (certificate content)
        cert: (certificate content)
        key: (certificate content)
```

You can read more about it here: https://docs.openstack.org/security-guide/databases/database-access-control.html

### Upgrades

Each OpenStack formula provides a set of phases (logical blocks) that help to build a flexible upgrade orchestration logic for particular components. The table below lists the phases and their descriptions:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;app&gt;.upgrade.service_running</code></td>
<td>Ensure that all services for particular application are enabled for autostart and running</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.service_stopped</code></td>
<td>Ensure that all services for particular application disabled for autostart and dead</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.pkgs_latest</code></td>
<td>Ensure that packages used by particular application are installed to latest available version. This will not upgrade data plane packages like qemu and openvswitch as usually 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. Applying this state will not autostart service.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.render_config</code></td>
<td>Ensure configuration is rendered actual version.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.pre</code></td>
<td>We assume this state is applied on all nodes in the cloud before running upgrade. Only non destructive actions will be applied during this phase. Perform service built in service check like (keystone-manage doctor and nova-status upgrade)</td>
</tr>
</tbody>
</table>

©2020, Mirantis Inc.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.pre</code></td>
<td>Mostly applicable for data plane nodes. During this phase resources will be gracefully removed from current node if it is allowed. Services for upgraded application will be set to admin disabled state to make sure node will not participate in resources scheduling. For example on gtw nodes this will set all agents to admin disable state and will move all routers to other agents.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade</code></td>
<td>This state will basically upgrade application on particular target. Stop services, render configuration, install new packages, run offline dbsync (for ctl), start services. Data plane should not be affected, only OpenStack Python services.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.upgrade.post</code></td>
<td>Add services back to scheduling.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.post</code></td>
<td>This phase should be launched only when upgrade of the cloud is completed. Cleanup temporary files, perform other post upgrade tasks.</td>
</tr>
<tr>
<td><code>&lt;app&gt;.upgrade.verify</code></td>
<td>Here we will do basic health checks (API CRUD operations, verify do not have dead network agents/compute services)</td>
</tr>
</tbody>
</table>
OPENLDAP

Usage

Sample pillars

Client

```yaml
openldap:
  client:
    server:
      basedn: dc=example,dc=local
      host: ldap.example.local
      tls: true
      port: 389
    auth:
      user: cn=admin,dc=example,dc=local
      password: dummypass
  entry:
    people:
      type: ou
      classes:
        - top
        - organizationalUnit
    entry:
      jdoe:
        type: cn
        # Change attributes that already exists with different content
        action: replace
        # Delete all other attributes
        purge: true
        attr:
          uid: jdoe
          uidNumber: 20001
          gidNumber: 20001
          gecos: John Doe
          givenName: John
          sn: Doe
          homeDirectory: /home/jdoe
          loginShell: /bin/bash
          classes:
            - posixAccount
            - inetOrgPerson
            - top
            - ldapPublicKey
            - shadowAccount
      karelin: 
```
# Simply remove cn=karel

type: cn
enabled: false

Read more

Python

Usage
Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

Available metadata

- service.environment.environment
  Basic Python environment
- service.environment.development
  Python development environment
- python.environment.django
  Python Django environment

Sample pillars
Simple Python environment:

```python
python:
  environment:
    enabled: true
```

Development Python environment:

```python
python:
  environment:
    enabled: true
    module:
      development: true
```

Python django environment:

```python
python:
  environment:
    enabled: true
    module:
      django: true
```
Using offline mirrors:

```yaml
python:
  environment:
    enabled: true
  user:
    root:
      pypi_user: user
      pypi_password: password
  pypi_mirror:
    protocol: http
    host: pypi.local
    port: 8084
    upstream_fallback: true
    user: user
    password: password
```

Read more

- [https://www.python.org/](https://www.python.org/)
RABBITMQ

Usage

RabbitMQ is a complete and highly reliable enterprise messaging system based on the emerging AMQP standard.

Sample pillars

Standalone broker

RabbitMQ as AMQP broker with admin user and vhosts:

```yaml
rabbitmq:
  server:
    enabled: true
    memory:
      vm_high_watermark: 0.4
    bind:
      address: 0.0.0.0
      port: 5672
    secret_key: rabbit_master_cookie
  admin:
    name: adminuser
    password: pwd
  plugins:
    - amqp_client
    - rabbitmq_management
  host:
    '/monitor':
      enabled: true
      user: monitor
      password: password
```

RabbitMQ as a STOMP broker:

```yaml
rabbitmq:
  server:
    enabled: true
    secret_key: rabbit_master_cookie
  bind:
    address: 0.0.0.0
    port: 5672
  host:
    '/monitor':
      enabled: true
      user: monitor
      password: password
```
plugins_runas_user: rabbitmq
plugins:
- rabbitmq_stomp

RabbitMQ cluster
RabbitMQ as base cluster node:

rabbitmq:
  server:
    enabled: true
    bind:
      address: 0.0.0.0
      port: 5672
    secret_key: rabbit_master_cookie
  admin:
    name: adminuser
    password: pwd
  cluster:
    enabled: true
    role: master
    mode: disc
    members:
    - name: openstack1
      host: 10.10.10.212
    - name: openstack2
      host: 10.10.10.213

HA Queues definition:

rabbitmq:
  server:
    enabled: true
    ...
  host:
    '/monitor':
      enabled: true
      user: 'monitor'
      password: 'password'
      policies:
      - name: HA
        pattern: '^(!amq\.)\.*$
        definition: '{"ha-mode": "all"}'

Enable TLS support
To enable support of TLS for rabbitmq-server you need to provide a path to cacert, server cert and private key:
rabbitmq:
  server:
    enabled: true
  ... 
  ssl:
    enabled: True
    key_file: /etc/rabbitmq/ssl/key.pem
    cert_file: /etc/rabbitmq/ssl/cert.pem
    ca_file: /etc/rabbitmq/ssl/ca.pem

To manage content of these files you can either use the following options:

rabbitmq:
  server:
    enabled: true
  ... 
  ssl:
    enabled: True
    key_file: /etc/rabbitmq/ssl/key.pem
    key:
      -----BEGIN RSA PRIVATE KEY-----
       ...
      -----END RSA PRIVATE KEY------

    ca_file: /etc/rabbitmq/ssl/ca.pem
    cacert_chain:
      -----BEGIN CERTIFICATE-----
       ...
      -----END CERTIFICATE------

    cert_file: /etc/rabbitmq/ssl/cert.pem
    cert:
      -----BEGIN CERTIFICATE-----
       ...
      -----END CERTIFICATE------

Or you can use the salt.minion.cert salt state which creates all required files according to defined reclass model. See https://github.com/Mirantis/reclass-system-salt-model/tree/master/salt/minion/cert/rabbitmq for details. In this case you need just to enable ssl and nothing more:

rabbitmq:
  server:
    enabled: true
  ... 
  ssl:
    enabled: True
Default port for TLS is 5671:

```
rabbitmq:
  server:
  bind:
    ssl:
      port: 5671
```

**Usage**

Check cluster status, example shows running cluster with 3 nodes: ctl-1, ctl-2, ctl-3

```
> rabbitmqctl cluster_status

Cluster status of node 'rabbit@ctl-1' ...
{nodes,[{disc,['rabbit@ctl-1','rabbit@ctl-2','rabbit@ctl-3']}]},
{running_nodes,['rabbit@ctl-3','rabbit@ctl-2','rabbit@ctl-1']},
{partitions,[]}
...done.
```

Setup management user:

```
> rabbitmqctl add_vhost vhost
> rabbitmqctl add_user user alive
> rabbitmqctl set_permissions -p vhost user "." "." "." "." "."
> rabbitmqctl set_user_tags user management
```

EPD process is Erlang Port Mapper Daemon. It’s a feature of the Erlang runtime that helps Erlang nodes to find each other. It’s a pretty tiny thing and doesn’t contain much state (other than “what Erlang nodes are running on this system?”) so it’s not a huge deal for it to still be running.

Although it’s running as user rabbitmq, it was started automatically by the Erlang VM when we started. We’ve considered adding “epmd -kill” to our shutdown script - but that would break any other Erlang apps running on the system; it’s more “global” than RabbitMQ.

**Read more**

- [http://docs.saltstack.com/ref/states/all/salt.states.rabbitmq_user.html](http://docs.saltstack.com/ref/states/all/salt.states.rabbitmq_user.html)

**Clustering**

- [http://www.rabbitmq.com/clustering.html#auto-config](http://www.rabbitmq.com/clustering.html#auto-config)
• https://github.com/jesusaurus/hpcs-salt-state/tree/master/rabbitmq
**RECLASS**

**Usage**

Reclass is an external node classifier (ENC) as can be used with automation tools, such as Puppet, Salt, and Ansible. It is also a stand-alone tool for merging data sources recursively.

**Sample metadata**

Install sources from [repository, git, pip]:

```yaml
salt:
  source:
    engine: pkg
...
  source:
    engine: git
    repo: git+https://github.com/salt-formulas/reclass
    branch: master
...
  source:
    engine: pip
...
```

If reclass is pre-installed, set the engine to None to avoid updates:

```yaml
salt:
  source:
    engine: None
```

Reclass storage with data fetched from git:

- See tests/pillar/storage_git.sls

Reclass storage with local data source:

- See tests/pillar/storage_local.sls

Reclass storage with archive data source:

- See tests/pillar/storage_archive.sls

Reclass storage with archive data source with content hash check:

- See tests/pillar/storage_archive_public.sls

Reclass model with single node definition:
See tests/pillar/generate_single.sls

Reclass model with multiple node defined:
See tests/pillar/generate_multi.sls

Reclass model with multiple node defined and interpolation enabled:
See tests/pillar/generate_multi_interpolate.sls

Reclass storage with simple class mappings:
See tests/pillar/class_mapping.sls

Reclass models with dynamic node classification
See tests/pillar/node_classify.sls

Classify node after creation and unclassify on node deletion:

```
salt:
  master:
    reactor:
      reclass/minion/classify:
      - salt://reclass/reactor/node_register.sls
      reclass/minion/declassify:
      - salt://reclass/reactor/node_unregister.sls
```

Event to trigger the node classification:
```
salt-call event.send 'reclass/minion/classify' "{'node_master_ip': '$config_host', 'node_ip': '$node_ip', 'node_domain': '$node_domain', 'node_cluster': '$node_cluster', 'node_hostname': '$node_hostname', 'node_os': '$node_os'}"
```

Note
You can send any parameters in the event payload, all will be checked against dynamic node classification conditions.
Both actions will use the minion ID as the node_name to be updated.

Confirmation of node classification
Currently, Salt does not allow getting confirmation on minion upon successful reactor execution on event. However, there can be issues with reactor in Salt 2017.7 (https://github.com/saltstack/salt/issues/47539) or reactor register state can fail if pillar failed to
render, so node registration confirmation may be needed. To enable this functionality, add the `node_confirm_registration` parameter to event data with value true:

```
salt-call event.send 'reclass/minion/classify' "{'node_master_ip': '${config_host}', 'node_ip': '${node_ip}', 'node_domain': '${node_domain}', 'node_cluster': '${node_cluster}', 'node_hostname': '${node_hostname}', 'node_os': '${node_os}', node_confirm_registration: true}""
```

Then on minion side execute:

```
salt-call mine.get 'salt:master' ${minion_id}_classified pillar
```

If true is returned, then registration has passed successfully.

Event to trigger the node declassification:

```
salt-call event.send 'reclass/minion/declassify'
```

Nodes definitions generator

Generate nodes definitions by running:

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

Remove unnecessary files from nodes/_generated:

```
reclass:
  storage:
    reclass_nodes_cleanup: true
```

Static node definition:

```
reclass:
  storage:
    enabled: true
    node:
      openstack_benchmark_node01:
        classes:
          - cluster.example.openstack.benchmark
        domain: example.com
        name: bmk01
        params:
          linux_system_codename: xenial
          salt_master_host: 192.168.0.253
          single_address: 192.168.2.95
```

Multiple nodes definitions (using generator):

```
reclass:
  storage:
    enabled: true
```
Multiple nodes definitions (using generator) with IP address comprehension. Ranges are named and formatting symbol of the same name is replaced by IP address from the corresponding range:

```yaml
node:
  openstack_compute_rack01:
    classes:
    - cluster.example.openstack.compute
    domain: example.com
    name: cmp<<count>>
    params:
      linux_system_codename: xenial
      salt_master_host: 192.168.0.253
    repeat:
      start: 1
      count: 50
      digits: 3
      params:
        single_address:
          start: 101
          value: 192.168.2.<<count>>
```

```yaml
reclass:
  storage:
    enabled: true
  node:
    openstack_compute_rack01:
      classes:
      - cluster.example.openstack.compute
      domain: example.com
      name: cmp<<count>>
      params:
        linux_system_codename: xenial
        salt_master_host: 192.168.0.253
      repeat:
        ip_ranges:
          single_address: '172.16.10.97-172.16.10.98'
          tenant_address: '172.16.20.97-172.16.20.98'
        network_ranges:
          sriov_address: '10.10.0.1/24-10.10.50.1/24'
          start: 1
          count: 50
          digits: 3
          params:
            single_address:
              start: 101
              value: 192.168.2.<<single_address>>
    tenant_address:
```
**start:** 101
**value:** 192.168.2.<<tenant_address>>

Read more

- [http://reclass.pantsfullofunix.net/index.html](http://reclass.pantsfullofunix.net/index.html)
- [http://reclass.pantsfullofunix.net/operations.html](http://reclass.pantsfullofunix.net/operations.html)
SALT

Usage
Salt is a new approach to infrastructure management. Easy enough to get running in minutes, scalable enough to manage tens of thousands of servers, and fast enough to communicate with them in seconds.

Salt delivers a dynamic communication bus for infrastructures that can be used for orchestration, remote execution, configuration management and much more.

Sample metadata
Salt Master
Salt master with base formulas and pillar metadata back end:


Salt master with reclass ENC metadata back end:


Salt master with Architect ENC metadata back end:

```yaml
salt:
  master:
    enabled: true
    pillar:
      engine: architect
      project: project-name
      host: architect-api
      port: 8181
      username: salt
      password: password
```

Salt master with multiple ext_pillars:

```yaml
salt:
  master:
    enabled: true
  pillar:
    engine: salt
    source:
      engine: local
  ext_pillars:
    1:
      module: cmd_json
```
```yaml
params: "echo {"arg": "val"}"
2:
  module: cmd_yaml
  params: /usr/local/bin/get_yml.sh
```

Salt master with API:


Salt master with defined user ACLs:


Salt master with preset minions:

```yaml
salt:
  master:
    enabled: true
    minions:
      - name: 'node1.system.location.domain.com'
```

Salt master with pip based installation (optional):

```yaml
salt:
  master:
    enabled: true
  ...  
  source: pkg
  engine: pip
  version: 2016.3.0rc2
```

Install formula through system package management:

```yaml
salt:
  master:
    enabled: true
  ...  
  environment:
    prd:
      keystone:
        source: pkg
        name: salt-formula-keystone
      nova:
        source: pkg
        name: salt-formula-keystone
        version: 0.1+0~20160818133412.24~1.gbp6e1ebb
```
Formula `postresql` is installed latest version and the formulas without version are installed in one call to `aptpkg` module. If the version attribute is present `sls` iterates over formulas and take action to install specific version or remove it. The version attribute may have these values `[latest|purged|removed|<VERSION>]`.

Clone master branch of `keystone` formula as local feature branch:

```
salt:
  master:
    enabled: true
...
  environment:
    dev:
      formula:
        keystone:
          source: git
          address: git@github.com:openstack/salt-formula-keystone.git
          revision: master
          branch: feature
```

Salt master with specified formula refs (for example, for Gerrit review):

```
salt:
  master:
    enabled: true
...
  environment:
    dev:
      formula:
        keystone:
          source: git
          address: https://git.openstack.org/openstack/salt-formula-keystone
          revision: refs/changes/56/123456/1
```

Salt master logging configuration:

```
salt:
  master:
    enabled: true
  log:
    level: warning
    file: '/var/log/salt/master'
    level_logfile: warning
```
Salt minion logging configuration:

```yaml
salt:
  minion:
    enabled: true
    log:
      level: info
      file: '/var/log/salt/minion'
      level_logfile: warning
```

Salt master with logging handlers:

```yaml
salt:
  master:
    enabled: true
  handler:
    handler01:
      engine: udp
      bind:
        host: 127.0.0.1
        port: 9999
    handler02:
      engine: zmq
      bind:
        host: 127.0.0.1
        port: 9999
```

Salt engine definition for saltgraph metadata collector:

```yaml
salt:
  master:
    engine:
      graph_metadata:
        engine: saltgraph
        host: 127.0.0.1
        port: 5432
        user: salt
        password: salt
        database: salt
```

Salt engine definition for Architect service:
salt:
master:
  engine:
    architect:
      engine: architect
      project: project-name
      host: architect-api
      port: 8181
      username: salt
      password: password

Salt engine definition for sending events from docker events:

salt:
master:
  engine:
    docker_events:
      docker_url: unix://var/run/docker.sock

Salt master peer setup for remote certificate signing:

salt:
master:
  peer:
    ":.*": 
      - x509.sign_remote_certificate

Salt master backup configuration:

salt:
master:
  backup: true
  initial_data:
    engine: backupninja
    home_dir: remote-backup-home-dir
    source: backup-node-host
    host: original-salt-master-id

Configure verbosity of state output (used for salt command):

salt:
master:
  state_output: changes

Pass pillar render error to minion log:
Note
When set to False this option is great for debugging. However it is not recommended for any production environment as it may contain templating data as passwords, and so on, that minion should not expose.

```yaml
salt:
  master:
    pillar_safe_render_error: False
```

Enable Windows repository support:

```yaml
salt:
  master:
    win_repo:
      source: git
      address: https://github.com/saltstack/salt-winrepo-ng
      revision: master
```

Configure a gitfs_remotes resource:

```yaml
salt:
  master:
    gitfs_remotes:
      salt_formula:
        url: https://github.com/salt-formulas/salt-formula-salt.git
        enabled: true
        params:
          base: master
```

Read more about gitfs resource options in the official Salt documentation.

Event/Reactor systems
Salt to synchronize node pillar and modules after start:

```yaml
salt:
  master:
    reactor:
      salt/minion/*/start:
        - salt://salt/reactor/node_start.sls
```

Trigger basic node install:
salt:
master:
  reactor:
    salt/minion/install:
      - salt://salt/reactor/node_install.sls

Sample event to trigger the node installation:
salt-call event.send 'salt/minion/install'

Run any defined orchestration pipeline:
salt:
master:
  reactor:
    salt/orchestrate/start:
      - salt://salt/reactor/orchestrate_start.sls

Event to trigger the orchestration pipeline:
salt-call event.send 'salt/orchestrate/start' "{'orchestrate': 'salt/orchestrate/infra_install.sls'}"

Synchronise modules and pillars on minion start:
salt:
master:
  reactor:
    'salt/minion/*/start':
      - salt://salt/reactor/minion_start.sls

Add and/or remove the minion key:
salt:
master:
  reactor:
    salt/key/create:
      - salt://salt/reactor/key_create.sls
    salt/key/remove:
      - salt://salt/reactor/key_remove.sls

Event to trigger the key creation:
salt-call event.send 'salt/key/create' \n> "{'node_id': 'id-of-minion', 'node_host': '172.16.10.100', 'orch_post_create': 'kubernetes.orchestrate.compute_install', 'post_create_pillar': {'node_name': 'id-of-minion'}}"
Note

You can add pass additional orch_pre_create, orch_post_create, orch_pre_remove or orch_post_remove parameters to the event to call extra orchestrate files. This can be useful for example for registering/unregistering nodes from the monitoring alarms or dashboards.

The key creation event needs to be run from other machine than the one being registered.

Event to trigger the key removal:

```
salt-call event.send 'salt/key/remove'
```

Control VM provisioning:

```yaml
_param:
  vcp_links: &vcp_links
  - type: phy
    id: ens2
    name: ens2
  private-ipv4: &private-ipv4
  - id: private-ipv4
    type: ipv4
    link: ens2
    netmask: 255.255.255.0
    routes:
      - gateway: 192.168.0.1
        netmask: 0.0.0.0
        network: 0.0.0.0

_virt:
  disk:
    three_disks:
      - system:
          size: 4096
          image: ubuntu.qcow
      - repository_snapshot:
          size: 8192
          image: snapshot.qcow
      - cinder-volume:
          size: 2048

_nic:
  control:
    - name: nic01
      bridge: br-pxe
      model: virtio
    - name: nic02
      bridge: br-cp
```
model: virtio
- name: nic03
  bridge: br-store-front
model: virtio
- name: nic04
  bridge: br-public
model: virtio
- name: nic05
  bridge: br-prv
model: virtio
  virtualport:
    type: openvswitch

salt:
  control:
    enabled: true
    virt_enabled: true
  size:
    medium_three_disks:
      cpu: 2
      ram: 4
    disk_profile: three_disks
cluster:
  mycluster:
    domain: neco.virt.domain.com
    engine: virt

# Cluster global settings
rng: false
enable_vnc: True
seed: cloud-init
cloud_init:
  user_data:
    disable_ec2_metadata: true
    resize_rootfs: True
    timezone: UTC
    ssh_deletekeys: True
    ssh_genkeytypes: ['rsa', 'dsa', 'ecdsa']
    ssh_svcname: ssh
    locale: en_US.UTF-8
    disable_root: true
    apt_preserve_sources_list: false
    apt:
      sources_list: ""
      sources:
        ubuntu.list:
          source: ${linux:system:repo:ubuntu:source}
        mcp_saltstack.list:
          source: ${linux:system:repo:mcp_saltstack:source}

node:
There are two methods to seed an initial Salt minion configuration to Libvirt VMs: mount a disk and update a filesystem or create a ConfigDrive with a Cloud-init config. This is controlled by the “seed” parameter on cluster and node levels. When set to _True_ or “qemu-nbd”, the old method of mounting a disk will be used. When set to “cloud-init”, the new method will be used. When set to _False_, no seeding will happen. The default value is _True_, meaning the “qemu-nbd” method will be used. This is done for backward compatibility and may be changed in future.
The recommended method is to use Cloud-init. It’s controlled by the “cloud_init” dictionary on cluster and node levels. Node level parameters are merged on top of cluster level parameters. The Salt Minion config is populated automatically based on a VM name and config settings of the minion who is actually executing a state. To override them, add the “salt_minion” section into the “user_data” section as shown above. It is possible to disable Cloud-init by setting “cloud_init.enabled” to _False_.

To enable Redis plugin for the Salt caching subsystem, use the below pillar structure:

```yaml
salt:
  master:
    cache:
      plugin: redis
      host: localhost
      port: 6379
      db: '0'
      password: pass_word
      bank_prefix: 'MCP'
      bank_keys_prefix: 'MCPKEY'
      key_prefix: 'KEY'
      separator: '@'
```

Jinja options

Use the following options to update default Jinja renderer options. Salt recognize Jinja options for templates and for the sls files.

For full list of options, see Jinja documentation: [http://jinja.pocoo.org/docs/api/#high-level-api](http://jinja.pocoo.org/docs/api/#high-level-api)
line_comment_prefix: '##'

# for .sls state files
jinja_sls: *jinja_env

With the line_statement/comment* _prefix options enabled following code statements are valid:

```jinja
%- set myvar = 'one'

## You can mix even with '{%'
{% set myvar = 'two' %} ## comment
%- set mylist = ['one', 'two', 'three'] ## comment

## comment
%- for item in mylist: ## comment
  {{- item }}
%- endfor
```

Encrypted pillars

Note

NACL and the below configuration will be available in Salt > 2017.7.

External resources:

- Tutorial to configure the Salt and Reclass ext_pillar and NAACL:
  http://apealive.net/post/2017-09-salt-nacl-ext-pillar/
- SaltStack documentation: https://docs.saltstack.com/en/latest/ref/modules/all/salt.modules.nacl.html

Configure salt NACL module:

```
pip install --upgrade libnacl==1.5.2
salt-call --local nacl.keygen /etc/salt/pki/master/nacl

local:
  saved sk_file:/etc/salt/pki/master/nacl  pk_file: /etc/salt/pki/master/nacl.pub
```

salt:
  master:
    pillar:
      reclass: *reclass
      nacl:
NACL encrypt secrets:

```
salt-call --local nacl.enc 'my_secret_value' pk_file=/etc/salt/pki/master/nacl
hXTkJpC1hcKMS7yZVGESutWrkvzusXfETXkacSkIixYjFWDIMJmR37MImthdIgjXpg4f2AlBKb8tc9Woma7q
# or
salt-run nacl.enc 'myotherpass'
ADDFD0Rav6p6+63sojl7Htfrcnp5rrDVyeE4BSPO7ipq8fZuLDIVAzQl4PCbDqi+Fau5KD3/j/E+Pw=
```

NACL encrypted values on pillar:

Use Boxed syntax NACL[CryptedValue=] to encode value on pillar:

```
my_pillar:
  my_nacl:
    key0: unencrypted_value
    key1: NACL[hXTkJpC1hcKMS7yZVGESutWrkvzusXfETXkacSkIixYjFWDIMJmR37MImthdIgjXpg4f2AlBKb8tc9Woma7q]
```

NACL large files:

```
salt-call nacl.enc_file /tmp/cert.crt out=/srv/salt/env/dev/cert.nacl
# or more advanced

cert=$(cat /tmp/cert.crt)
salt-call --out=newline_values_only nacl.enc_pub data="$cert" > /srv/salt/env/dev/cert.nacl
```

NACL within template/native pillars:

```
pillarexample:
  user: root
  password1: {{salt.nacl.dec('DRB7Q6/X5gGSRCTpZyxS6hlbWj0lU+uAvryou3vJ4=')|json}}
  cert_key: {{salt.nacl.dec_file('/srv/salt/env/dev/certs/example.com/cert.nacl')|json}}
  cert_key2: {{salt.nacl.dec_file('salt://certs/example.com/cert2.nacl')|json}}
```

Salt Syndic

The master of masters:

```
salt:
  master:
    enabled: true
    order_masters: True
```
Lower syndicated master:

```yaml
salt:
syndic:
  enabled: true
master:
  host: master-of-master-host
timeout: 5
```

Syndicated master with multiple master of masters:

```yaml
salt:
syndic:
  enabled: true
masters:
- host: master-of-master-host1
- host: master-of-master-host2
timeout: 5
```

Salt Minion

Minion ID by default triggers dependency on Linux formula, as it uses fqdn configured from `linux.system.name` and `linux.system.domain` pillar. To override, provide exact minion ID you require. The same can be set for master ID rendered at master.conf.

```yaml
salt:
minion:
  id: minion1.production
master:
  id: master.production
```

Simplest Salt minion setup with central configuration node:

```
```

Multi-master Salt minion setup:

```
```

Salt minion with salt mine options:

```
https://github.com/salt-formulas/salt-formula-salt/blob/master/tests/pillar/minion_mine.sls
```

Salt minion with graphing dependencies:

```
https://github.com/salt-formulas/salt-formula-salt/blob/master/tests/pillar/minion_graph.sls
```
Salt minion behind HTTP proxy:

```yaml
salt:
  minion:
    proxy:
      host: 127.0.0.1
      port: 3128
```

Salt minion to specify non-default HTTP backend. The default tornado backend does not respect HTTP proxy settings set as environment variables. This is useful for cases where you need to set no_proxy lists.

```yaml
salt:
  minion:
    backend: urllib2
```

Salt minion with PKI certificate authority (CA):

https://github.com/salt-formulas/salt-formula-salt/blob/master/tests/pillar/minion_pki_ca.sls

Salt minion using PKI certificate


Salt minion trust CA certificates issued by salt CA on a specific host (ie: salt-master node):

```yaml
salt:
  minion:
    trusted_ca_minions:
      - cfg01
```

Salt Minion Proxy

Salt proxy pillar:

```yaml
salt:
  minion:
    proxy_minion:
      master: localhost
device:
      vsrx01.mydomain.local:
        enabled: true
        engine: napalm
csr1000v.mydomain.local:
        enabled: true
        engine: napalm
```
Proxy pillar for IOS device:

```yaml
proxy:
    proxytype: napalm
    driver: ios
    host: csr1000v.mydomain.local
    username: root
    passwd: r00tme
```

Note
This is pillar of the the real salt-minion

Proxy pillar for JunOS device:

```yaml
proxy:
    proxytype: napalm
    driver: junos
    host: vsrx01.mydomain.local
    username: root
    passwd: r00tme
    optional_args:
        config_format: set
```

Note
This is pillar of the node thats not able to run salt-minion itself.

Salt SSH
Salt SSH with sudoer using key:

```
https://github.com/salt-formulas/salt-formula-salt/blob/master/tests/pillar/master_ssh_minion_key.sls
```

Salt SSH with sudoer using password:

©2020, Mirantis Inc.
Salt SSH with root using password:

Salt control (cloud/kvm/docker)
Salt cloud with local OpenStack provider:
https://github.com/salt-formulas/salt-formula-salt/blob/master/tests/pillar/control_cloud_openstack.sls

Salt cloud with Digital Ocean provider:
https://github.com/salt-formulas/salt-formula-salt/blob/master/tests/pillar/control_cloud_digitalocean.sls

Salt virt with KVM cluster:
https://github.com/salt-formulas/salt-formula-salt/blob/master/tests/pillar/control_virt.sls

Salt virt with custom destination for image file:
https://github.com/salt-formulas/salt-formula-salt/blob/master/tests/pillar/control_virt_custom.sls

Salt shared library
This formula includes ‘sharedlib’ execution module which is a kind of ‘library’ of function and / or classes to be used in Jinja templates or directly as execution module.

‘sharedlib’ implements a loader that is able to scan nested directories and import Python classes / functions from nested modules. Salt doesn’t allow this as it only imports top-level modules:
https://github.com/saltstack/salt/issues/37273

‘sharedlib’ implements 4 main functions:

• ‘sharedlib.list’ - search and print functions / classes found in nested directories
• ‘sharedlib.info’ - print docstring of a function (if it exists)
• ‘sharedlib.get’ - get function / class object, but not execute it immediately
• ‘sharedlib.call’ - get function / class and execute / initialize it with arguments given.

Each of the commands above also have it’s own docstring so it’s possible to use them on a system:

```
# salt-call sys.doc sharedlib.list
local:
    --------
sharedlib.list:
```

List available functions.

.. code-block::

    salt-call sharedlib.list

Usage examples:

```
# salt-call sharedlib.list
local:
    ---------
    sharedlib.list:
        ---------
        classes:
        - misc.Test
        - misc2.Test
        functions:
        - misc.cast_dict_keys_to_int
```

```
# salt-call sharedlib.info misc.cast_dict_keys_to_int
local:
    ---------
    sharedlib.info:
        ---------
        misc.cast_dict_keys_to_int:

Return a dictionary with keys casted to int.
This usually is required when you want sort the dict later.

Jinja example:

```
.. code-block:: jinja

    {%- set ruleset = salt['sharedlib.call']('misc.cast_dict_keys_to_int', c.get('ruleset', {})) %}
```

```
.. code-block:: jinja

    {%- set func = salt['sharedlib.get']('misc.cast_dict_keys_to_int') %}
    {%- for c_name, c in t.chains.items() %}
    {%- set ruleset = func(c.get('ruleset', {})) %}
    {%- for rule_id, r in ruleset | dictsort %}
    ...
    {%- endfor %}
```

Usage

Working with salt-cloud:
salt-cloud -m /path/to/map --assume-yes

Debug LIBCLOUD for salt-cloud connection:

```bash
export LIBCLOUD_DEBUG=/dev/stderr; salt-cloud --list-sizes provider_name --log-level all
```

Read more

- [https://github.com/saltstack-formulas/salt-formula](https://github.com/saltstack-formulas/salt-formula)

salt-cloud

- [http://www.blog.sandro-mathys.ch/2013/07/setting-user-password-when-launching.html](http://www.blog.sandro-mathys.ch/2013/07/setting-user-password-when-launching.html)
- [http://docs.saltstack.com/topics/cloud/digitalocean.html](http://docs.saltstack.com/topics/cloud/digitalocean.html)

### Metadata schema specifications for Salt minion

#### Core properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>masters</td>
<td>array</td>
<td>List of Salt masters to connect to. For details, see: Master definition</td>
</tr>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Enables the Salt minion role.</td>
</tr>
</tbody>
</table>

#### Master definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>

©2020, Mirantis Inc.
| master | string | Hostname or IP address of the masters server |
**SPHINX**

**Usage**

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license. It was originally created for the new Python documentation, and it has excellent facilities for the documentation of Python projects. The C/C++ projects are already supported as well, and it is planned to add special support for other languages as well.

Sample pillars

Sample documentation with local source:

```python
sphinx:
    server:
      enabled: true
    doc:
      board:
        builder: 'html'
        source:
          engine: local
          path: '/path/to/sphinx/documentation'
```

Sample documentation with Git source:

```python
sphinx:
    server:
      enabled: true
    doc:
      board:
        builder: 'html'
        source:
          engine: git
          address: 'git@repo1.domain.com/repo.git'
          revision: master
```

Sample documentation with Reclass source:

```python
sphinx:
    server:
      enabled: true
    doc:
      board:
        builder: 'html'
        source:
          engine: reclass
```
Sample documentation with pillar-schema source:

```yaml
sphinx:
  server:
    enabled: true
  doc:
    schemas_doc:
      author: Author
      year: Year
      version: Version
      builder: 'html'
      source:
        engine: pillar-schema
```

Read more

- [http://sphinx-doc.org/tutorial.html](http://sphinx-doc.org/tutorial.html)
XTRABACKUP

Usage

Xtrabackup allows you to backup and restore databases from full backups or full backups and its incrementals.

Sample pillars

Backup client with ssh/rsync remote host:

```yaml
xtrabackup:
  client:
    enabled: true
    full_backups_to_keep: 3
    hours_before_full: 48
    hours_before_incr: 12
    database:
      user: username
      password: password
    target:
      host: cfg01
```

Note

The full_backups_to_keep parameter states how many backup will be stored locally on xtrabackup client. More options to relocate local backups can be done using salt-formula-backupninja.

Backup client using DB API instead of socket (still needs to be run on the same server as DB):

```yaml
xtrabackup:
  client:
    enabled: true
    full_backups_to_keep: 3
    hours_before_full: 48
    hours_before_incr: 12
    database:
      user: username
      password: password
      host: localhost
      port: 3306
    target:
      host: cfg01
```
Note
DB user username must have “RELOAD” and “REPLICATION CLIENT” privileges on all databases.

Backup client with local backup only:

```
xtrabackup:
  client:
    enabled: true
    full_backups_to_keep: 3
    hours_before_full: 48
    hours_before_incr: 12
  database:
    user: username
    password: password
```

Note
The full_backups_to_keep parameter states how many backup will be stored locally on xtrabackup client.

Backup client with ssh/rsync to remote host with compression, IO throttling and non-default backup directory on server:

```
xtrabackup:
  client:
    enabled: true
    full_backups_to_keep: 3
    hours_before_full: 48
    hours_before_incr: 12
    compression: true
    compression_threads: 2
    throttle: 20
  database:
    user: username
    password: password
  target:
    host: cfg01
  server:
    enabled: false
    backup_dir: /srv/backup
```
Note
More options to relocate local backups can be done using salt-formula-backupninja.

Note
If the server section is omitted, backups will be made to default location, same on both
cclient and server side.

Backup client at exact times:

```yaml
xtrabackup:
  client:
    enabled: true
    full_backups_to_keep: 3
    incr_before_full: 3
    backup_dir: /var/backups/mysql/xtrabackup
    backup_times:
      day_of_week: 0
      hour: 4
      minute: 52
    compression: true
    compression_threads: 2
    database:
      user: user
      password: password
    target:
      host: host01
```

Note
Parameters in backup_times section can be used to set up exact time the cron job should
be executed. In this example, the backup job would be executed every Sunday at 4:52
AM. If any of the individual backup_times parameters is not defined, the default * value
will be used. For example, if minute parameter is *, it will run the backup every minute,
which is usually not desired.

Available parameters include:

- day_of_week
- day_of_month
• month
• hour
• minute.

See the crontab reference for further info on how to set these parameters.

Note
Please be aware that only backup_times section OR hours_before_full(incremental) can be defined. If both are defined. The backup_times section will be preferred.

Note
New parameter incr_before_full needs to be defined. This number sets number of incremental backups to be run, before a full backup is performed.

Backup server rsync and non-default backup directory:

```yaml
xtrabackup:
  server:
    enabled: true
    hours_before_full: 48
    full_backups_to_keep: 5
  key:
    xtrabackup_pub_key:
      enabled: true
      key: key
    backup_dir: /srv/backup
```

Note
The hours_before_full parameter should have the same value as is stated on xtrabackup client.
Note
If the backup_dir argument is omitted backups will be made to default location, same on both client and server side.

Backup server without strict client restriction:

```
xtrabackup:
  server:
    restrict_clients: false
```

Backup server at exact times:

```
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
```

Note
Parameters in backup_times section can be used to set up exact time the cron job should be executed. In this example, the backup job would be executed every Sunday at 4:52 AM. If any of the individual backup_times parameters is not defined, the default * value will be used. For example, if minute parameter is *, it will run the backup every minute, which is usually not desired.

See the crontab reference for further info on how to set these parameters.

Note
Please be aware that only backup_times section OR hours_before_full(incr) can be defined. If both are defined. The backup_times section will be preferred.
New parameter incr_before_full needs to be defined. This number sets number of incremental backups to be run, before a full backup is performed.

Client restore from local backups:

```yaml
xtrabackup:
  client:
    enabled: true
    full_backups_to_keep: 5
    hours_before_full: 48
    hours_before_incr: 12
    restore_full_latest: 1
    restore_from: local
    compression: true
    compressThreads: 2
    database:
      user: username
      password: password
    target:
      host: cfg01
    qpress:
      source: tar
      name: url
```

Note

restore_full_latest param with a value of 1 means to restore db from the last full backup and its increments. 2 would mean to restore second latest full backup and its increments.

Client restore from remote backups:

```yaml
xtrabackup:
  client:
    enabled: true
    full_backups_to_keep: 5
    hours_before_full: 48
    hours_before_incr: 12
    restore_full_latest: 1
    restore_from: remote
    compression: true
```
compressThreads: 2

database:
  user: username
  password: password

target:
  host: cfg01
  qpress:
    source: tar
  name: url

Note

The restore_full_latest parameter with a value of 1 means to restore db from the last full backup and its increments. 2 would mean to restore second latest full backup and its increments

Read more

- https://labs.riseup.net/code/projects/xtrabackup/wiki/Configuration
- http://www.debian-administration.org/articles/351
- https://github.com/riseuplabs/puppet-xtrabackup
- http://www.ushills.co.uk/2008/02/backup-with-xtrabackup.html