User Guide

User Guide

This section contains user tasks for your SUSE OpenStack Cloud 8 cloud.

Publication Date: 09/08/2022

SUSE LLC
1800 South Novell Place
Provo, UT 84606
USA
https://documentation.suse.com

Contents

I Cloud Admin User Guide

1 Using the Operations Console

1.1 Operations Console Overview

Often referred to as the Ops Console, you can use this web-based graphical user interface (GUI) to view data about your cloud infrastructure and ensure your cloud is operating correctly.

You can use the Operations Console for SUSE OpenStack Cloud 8 to view data about your SUSE OpenStack Cloud infrastructure in a web-based graphical user interface (GUI) and ensure your cloud is operating correctly. By logging on to the console, SUSE OpenStack Cloud administrators can manage data in the following ways: **Triage alarm notifications**.

- Alarm Definitions and notifications now have their own screens and are collected under the **Alarm Explorer** menu item which can be accessed from the Central Dashboard. Central Dashboard now allows you to customize the view in the following ways:
    - Rename or re-configure existing alarm cards to include services different from the defaults
    - Create a new alarm card with the services you want to select
    - Reorder alarm cards using drag and drop
    - View all alarms that have no service dimension now grouped in an **Uncategorized Alarms** card
    - View all alarms that have a service dimension that does not match any of the other cards -now grouped in an **Other Alarms** card
- You can also easily access alarm data for a specific component. On the Summary page for the following components, a link is provided to an alarms screen specifically for that component:
    - Compute Instances: *Section 1.3, "Managing Compute Hosts"*
    - Object Storage: *Section 1.4.4, "Alarm Summary"*

1.1.1 Monitor the environment by giving priority to alarms that take precedence.

Alarm Explorer now allows you to manage alarms in the following ways:

- Refine the monitoring environment by creating new alarms to specify a combination of metrics, services, and hosts that match the triggers unique to an environment
- Filter alarms in one place using an enumerated filter box instead of service badges
- Specify full alarm IDs as dimension key-value pairs in the form of **dimension=value**

1.1.2 Support Changes

- To resolve scalability issues, plain text search through alarm sets is no longer supported

The Business Logic Layer of Operations Console is a middleware component that serves as a single point of contact for the user interface to communicate with OpenStack services such as Monasca, Nova, and others.

1.2 Connecting to the Operations Console

Instructions for accessing the Operations Console through a web browser.

To connect to Operations Console, perform the following:

- Ensure your login has the required access credentials: *Section 1.2.1, "Required Access Credentials"*
- Connect through a browser: *Section 1.2.2, "Connect Through a Browser"*
- Optionally use a Hostname OR virtual IP address to access Operations Console: *Section 1.2.3, "Optionally use a Hostname OR virtual IP address to access Operations Console"*

Operations Console will always be accessed over port 9095.

1.2.1 Required Access Credentials

In previous versions of Operations Console you were required to have only the password for the Administrator account (admin by default). Now the Administrator user account must also have all of the following credentials:

Project	Domain	Role	Description
All projects	*not specific*	Admin	Admin role on at least one project
All projects	*not specific*	Admin	Admin role in default domain
Admin	default	Admin or Monasca-user	Admin or Monasca-user role on admin project

Important

If your login account has administrator role on the administrator project, then you only need to make sure you have the administrator role on the default domain.

Administrator account

During installation, an administrator account called `admin` is created by default.

Administrator password

During installation, an administrator password is randomly created by default. It is not recommend that you change the default password.

To find the randomized password:

1. To display the password, log on to the Cloud Lifecycle Manager and run:

```
cat ~/service.osrc
```

1.2.2 Connect Through a Browser

The following instructions will show you how to find the URL to access Operations Console. You will use SSH, also known as Secure Socket Shell, which provides administrators with a secure way to access a remote computer.

To access Operations Console:

1. Log in to the Cloud Lifecycle Manager.
2. Locate the URL or IP address for the Operations Console with the following command:

```
source ~/service.osrc && openstack endpoint list | grep opsconsole | grep admin
```

Sample output:

```
| 8ef10dd9c00e4abdb18b5b22adc93e87 | region1 | opsconsole | opsconsole | True |
 admin | https://192.168.24.169:9095/api/v1/
```

To access Operations Console, in the sample output, remove everything after port 9095 (api/v1/) and in a browser, type:

```
https://192.168.24.169:9095
```

1.2.3 Optionally use a Hostname OR virtual IP address to access Operations Console

Important

If you can access Operations Console using the above instructions, then you can skip this section. These steps provide an alternate way to access Operations Console if the above steps do not work for you.

To find your hostname OR IP address:

1. Navigate to and open in a text editor the following file:

   ```
   network_groups.yml
   ```

2. Find the following entry:

   ```
   external-name
   ```

3. If your administrator set a hostname value in the external-name field, you will use that hostname when logging in to Operations Console. or example, in a browser you would type:

   ```
   https://VIP:9095
   ```

4. If your administrator did not set a hostname value, then to determine the IP address to use, from your Cloud Lifecycle Manager, run:

   ```
   grep HZN-WEB /etc/hosts
   ```

   The output of that command will show you the virtual IP address you should use. For example, in a browser you would type:

   ```
   https://VIP:9095
   ```

1.3 Managing Compute Hosts

Operations Console (Ops Console) provides a graphical interface for you to add and delete compute hosts.

As your deployment grows and changes, you may need to add more compute hosts to increase your capacity for VMs, or delete a host to reallocate hardware for a different use. To accomplish these tasks, in previous versions of SUSE OpenStack Cloud you had to use the command line to update configuration files and run ansible playbooks. Now Operations Console provides a graphical interface for you to complete the same tasks quickly using menu items in the console.

Important

Do not refresh the Operations Console page or open Operations Console in another window during the following tasks. If you do, you will not see any notifications or be able to review the error log for more information. This would make troubleshooting difficult since you would not know the error that was encountered, or why it occurred.

Use Operations Console to perform the following tasks:

- Create a Compute Host: *Section 1.3.1, "Create a Compute Host"*
- Deactivate a Compute Host: *Section 1.3.2, "Deactivate a Compute Host"*
- Activate a Compute Host: *Section 1.3.3, "Activate a Compute Host"*
- Delete a Compute Host: *Section 1.3.4, "Delete a Compute Host"*

Important

To use Operations Console, you need to have the correct permissions and know the URL or VIP connected to Operations Console during installation. For steps on how to complete these tasks, see *Section 1.1, "Operations Console Overview"*.

1.3.1 Create a Compute Host

If you need to create additional compute hosts for more virtual machine capacity, you can do this easily on the Compute Hosts screen.

To add a compute host:

1. To open Operations Console, in a browser, enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   https://myardana.test:9095
   https://VIP:9095
   ```

2. On the **Home** screen, click the menu represented by 3 horizontal lines (☰).

3. From the menu that slides in on the left side, click **Compute**, and then **Compute Hosts**.

4. On the **Compute Hosts** page, click **Create Host**.
5. On the **Add & Activate Compute Host** tab that slides in from the right, enter the following information:

   Host ID
   : Cloud Lifecycle Manager model's server ID

   Host Role
   : Defined in the Cloud Lifecycle Manager model and cannot be modified in Operations Console

   Host Group
   : Defined in the Cloud Lifecycle Manager model and cannot be modified in Operations Console

   Host NIC Mapping
   : Defined in the Cloud Lifecycle Manager model and cannot be modified in Operations Console

   Encryption Key
   : If the configuration is encrypted, enter the encryption key here
6. Click *Create Host*, and in the confirmation screen that opens, click *Confirm*.
7. Wait for SUSE OpenStack Cloud to complete the pre deployment steps. This process can take up to 2 minutes.
8. If pre-deployment is successful, you will see a notification that deployment has started.

   ### Important

   If you receive a notice that pre-deployment did not complete successfully, read the notification explaining at which step the error occured. You can click on the error notification and see the ansible log for the configuration processor playbook. Then you can click **Create Host** in step 4 again and correct the mistake.
9. Wait for SUSE OpenStack Cloud to complete the deployments steps. This process can take up to 20 minutes.
10. If deployment is successful, you will see a notification and a new entry will appear in the compute hosts table.

Important

If you receive a notice that deployment did not complete successfully, read the notification explaining at which step the error occured. You can click on the error notification for more details.

1.3.2 Deactivate a Compute Host

If you have multiple compute hosts and for debugging reasons you want to disable them all except one, you may need to deactivate and then activate a compute host. If you want to delete a host, you will also have to deactivate it first. This can be done easily in the Operations Console.

Important

The host must be in the following state: **ACTIVATED**

To deactivate a compute host:

1. To open Operations Console, in a browser, enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   https://myardana.test:9095
   https://VIP:9095
   ```

2. On the **Home** screen, click the menu represented by 3 horizontal lines (☰).
3. From the menu that slides in on the left side, click **Compute**, and then **Compute Hosts**.
4. On the **Compute Hosts** page, in the row for the host you want to deactivate, click the details button (⋯).
5. Click **Deactivate**, and in the confirmation screen that opens, click **Confirm**.
6. Wait for SUSE OpenStack Cloud to complete the operation. This process can take up to 2 minutes.
7. If deactivation is successful, you will see a notification and in the compute hosts table the **STATE** will change to **DEACTIVATED**.

Important

If you receive a notice that the operation did not complete successfully, read the notification explaining at which step the error occured. You can click on the link in the error notification for more details. In the compute hosts table the **STATE** will remain **ACTIVATED**.

1.3.3 Activate a Compute Host

Important

The host must be in the following state: **DEACTIVATED**

To activate a compute host:

1. To open Operations Console, in a browser, enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   https://myardana.test:9095
   https://VIP:9095
   ```

2. On the **Home** screen, click the menu represented by 3 horizontal lines (☰).

3. From the menu that slides in on the left side, click **Compute**, and then **Compute Hosts**.

4. On the **Compute Hosts** page, in the row for the host you want to activate, click the details button (⋯).

5. Click **Activate**, and in the confirmation screen that opens, click **Confirm**.

6. Wait for SUSE OpenStack Cloud to complete the operation. This process can take up to 2 minutes.

7. If activation is successful, you will see a notification and in the compute hosts table the **STATE** will change to **ACTIVATED**.

Important

If you receive a notice that the operation did not complete successfully, read the notification explaining at which step the error occured. You can click on the link in the error notification for more details. In the compute hosts table the **STATE** will remain **DEACTIVATED**.

1.3.4 Delete a Compute Host

If you need to scale down the size of your current deployment to use the hardware for other purposes, you may want to delete a compute host.

Important

Complete the following steps before deleting a host:

- host must be in the following state: **DEACTIVATED**
- Optionally you can migrate the instance off the host to be deleted. To do this, complete the following sections in *Book "Operations Guide", Chapter 13 "System Maintenance", Section 13.1 "Planned System Maintenance", Section 13.1.3 "Planned Compute Maintenance", Section 13.1.3.5 "Removing a Compute Node"*:

  1. Disable provisioning on the compute host.
  2. Use live migration to move any instances on this host to other hosts.

To delete a compute host:

1. To open Operations Console, in a browser, enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   https://myardana.test:9095
   https://VIP:9095
   ```

2. On the **Home** screen, click the menu represented by 3 horizontal lines (☰).

3. From the menu that slides in on the left side, click **Compute**, and then **Compute Hosts**.
4. On the **Compute Hosts** page, in the row for the host you want to delete, click the details button (⋯).
5. Click **Delete**, and if the configuration is encrypted, enter the encryption key.
6. in the confirmation screen that opens, click **Confirm**.
7. In the compute hosts table you will see the **STATE** change to **Deleting**.
8. Wait for SUSE OpenStack Cloud to complete the operation. This process can take up to 2 minutes.
9. If deletion is successful, you will see a notification and in the compute hosts table the host will not be listed.

Important

If you receive a notice that the operation did not complete successfully, read the notification explaining at which step the error occured. You can click on the link in the error notification for more details. In the compute hosts table the **STATE** will remain **DEACTIVATED**.

1.3.5 For More Information

For more information on how to complete these tasks through the command line, see the following topics:

- *Book "Operations Guide", Chapter 13 "System Maintenance", Section 13.1 "Planned System Maintenance", Section 13.1.3 "Planned Compute Maintenance", Section 13.1.3.4 "Adding Compute Node"*
- *Book "Operations Guide", Chapter 13 "System Maintenance", Section 13.1 "Planned System Maintenance", Section 13.1.3 "Planned Compute Maintenance", Section 13.1.3.5 "Removing a Compute Node"*

1.4 Managing Swift Performance

In Operations Console you can monitor your Swift cluster to ensure long-term data protection as well as sufficient performance.

OpenStack Swift is an object storage solution with a focus on availability. While there are various mechanisms inside Swift to protect stored data and ensure a high availability, you must still closely monitor your Swift cluster to ensure long-term data protection as well as sufficient performance. The best way to manage Swift is to collect useful data that will detect possible performance impacts early on.

The new Object Summary Dashboard in Operations Console provides an overview of your Swift environment.

Important

If Swift is not installed and configured, you will not be able to access this dashboard. The Swift endpoint must be present in Keystone for the Object Summary to be present in the menu.

In Operations Console's object storage dashboard, you can easily review the following information:

- Performance Summary: *Section 1.4.1, "Performance Summary"*
- Inventory Summary: *Section 1.4.2, "Inventory Summary"*
- Capacity Summary: *Section 1.4.3, "Capacity Summary"*
- Alarm Summary: *Section 1.4.4, "Alarm Summary"*

1.4.1 Performance Summary

View a comprehensive summary of current performance values.

To access the object storage performance dashboard:

1. To open Operations Console, in a browser enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   https://myardana.test:9095
   https://VIP:9095
   ```

2. On the **Home** screen, click the menu represented by 3 horizontal lines (☰).
3. In the menu, click *Storage* › *Object Storage Summary*.

Performance data includes:

Healthcheck Latency from Monasca

This latency is the average time it takes for Swift to respond to a healthcheck, or ping, request. The swiftlm-uptime monitor program reports the value. A large difference between average and maximum may indicate a problem with one node.

Operational Latency from Monasca

Operational latency is the average time it takes for Swift to respond to an upload, download, or object delete request. The swiftlm-uptime monitor program reports the value. A large difference between average and maximum may indicate a problem with one node.

Service Availability

This is the availability over the last 24 hours as a percentage.

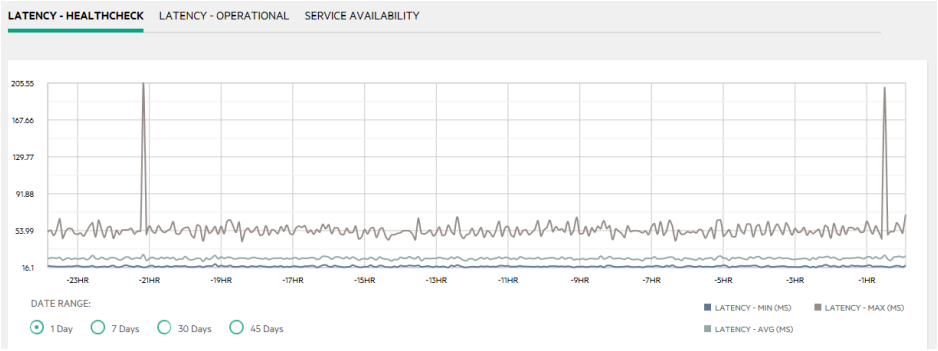
- **100%** - No outages in the last 24 hours
- **50%** - Swift was unavailable for a total of 12 hours in the last 24-hour period

Graph of Performance Over Time

Create a visual representation of performance data to see when Swift encountered longer-than-normal response times.

To create a graph:

1. Choose the length of time you want to graph in *Date Range*. This sets the length of time for the x-axis which counts backwards until it reaches the present time. In the example below, 1 day is selected, and so the x axis shows performance starting from 24 hours ago (-24) until the present time.
2. Look at the y-axis to understand the range of response times. The first number is the smallest value in the data collected from the backend, and the last number is the longest amount of time it took Swift to respond to a request. In the example below, the shortest time for a response from Swift was 16.1 milliseconds.
3. Look for spikes which represent longer than normal response times. In the example below, Swift experienced long response times 21 hours ago and again 1 hour ago.
4. Look for the latency value at the present time. The line running across the x-axis at 16.1 milliseconds shows you what the response time is currently.



1.4.2 Inventory Summary

Monitor details about all the Swift resources deployed in your cloud.

To access the object storage inventory screen:

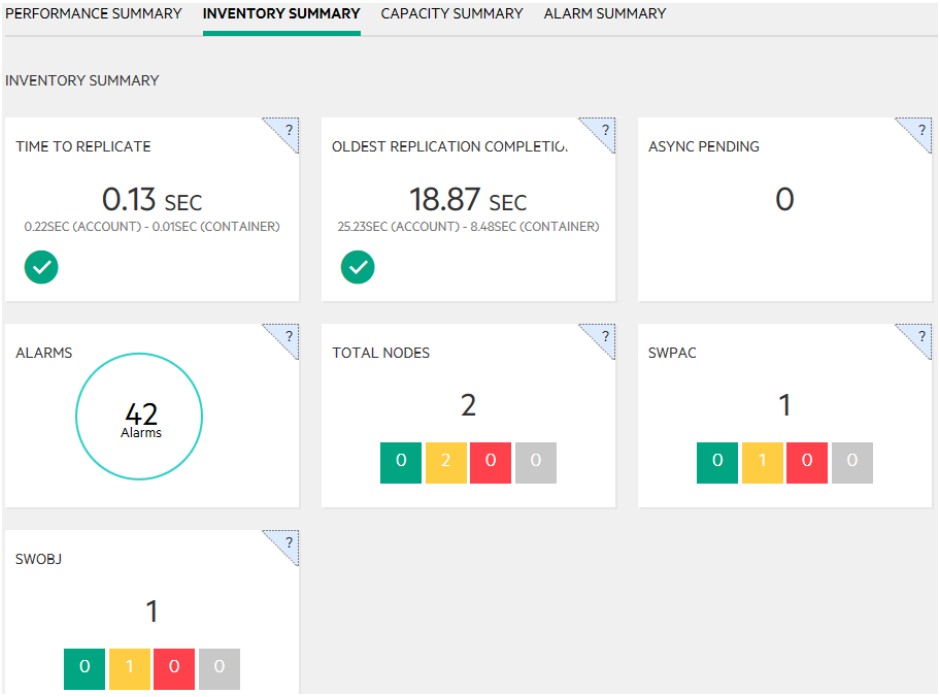
1. To open Operations Console, in a browser enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   https://myardana.test:9095
   https://VIP:9095
   ```

2. On the *Home* screen, click the menu represented by 3 horizontal lines (☰).

3. In the menu, click *Storage* › *Object Storage Summary*.

4. On the *Summary* page, click *Inventory Summary*.



General Swift metrics are available for the following attributes:

- *Time to replicate*: The average time in seconds it takes all hosts to complete a replication cycle.
- *Oldest replication*: The time in seconds that has elapsed since the object replication process completed its last replication cycle.
- *Async Pending*: This is the number of failed requests to add an entry in the container server's database.There is one async queue per Swift disk, and a cron job queries all Swift servers to calculate the total. When an object is uploaded into Swift, and it is successfully stored, a request is sent to the container server to add a new entry for the object in the database. If the container update fails, the request is stored in what Swift calls an Async Pending Queue.

  > **Important**
  >
  > On a public cloud deployment, this value can reach millions. If it continues to grow, it means that the container updates are not keeping up with the requests. It is also normal for it this number to grow if a node hosting the Swift container service is down.

- *Total number of alarms*: This number includes all nodes that host Swift services, including proxy, account, container, and object storage services.

- *Total nodes*: This number includes all nodes that host Swift services, including proxy, account, container, and object storage services. The number in the colored box represents the number of alarms in that state. The following colors are used to show the most severe alarm triggered on all nodes:

  Green

  Indicates all alarms are in a known and untriggered state. For example, if there are 5 nodes and they are all known with no alarms, you will see the number 5 in the green box, and a zero in all the other colored boxes.

  Yellow

  Indicates that some low or medium alarms have been triggered but no critical or high alarms. For example, if there are 5 nodes, and there are 3 nodes with untriggered alarms and 2 nodes with medium severity alarms, you will see the number 3 in the green box, the number 2 in the yellow box, and zeros in all the other colored boxes.

  Red

  Indicates at least one critical or high severity alarm has been triggered on a node. For example, if there are 5 nodes, and there are 3 nodes with untriggered alarms, 1 node with a low severity, and 1 node with a critical alarm, you will see the number 3 in the green box, the number 1 in the yellow box, the number 1 in the red box,and a zero in the gray box.

  Gray

  Indicates that all alarms on the nodes are unknown. For example, if there are 5 nodes with no data reported, you will see the number 5 in the gray box, and zeros in all the other colored boxes.

- *Cluster breakdown of nodes*: In the example screen above, the cluster consists of 2 nodes named SWPAC and SWOBJ. Click a node name to bring up more detailed information about that node.

1.4.3 Capacity Summary

Use this screen to view the size of the file system space on all nodes and disk drives assigned to Swift. Also shown is the remaining space available and the total size of all file systems used by Swift. Values are given in megabytes (MB).

To access the object storage alarm summary screen:

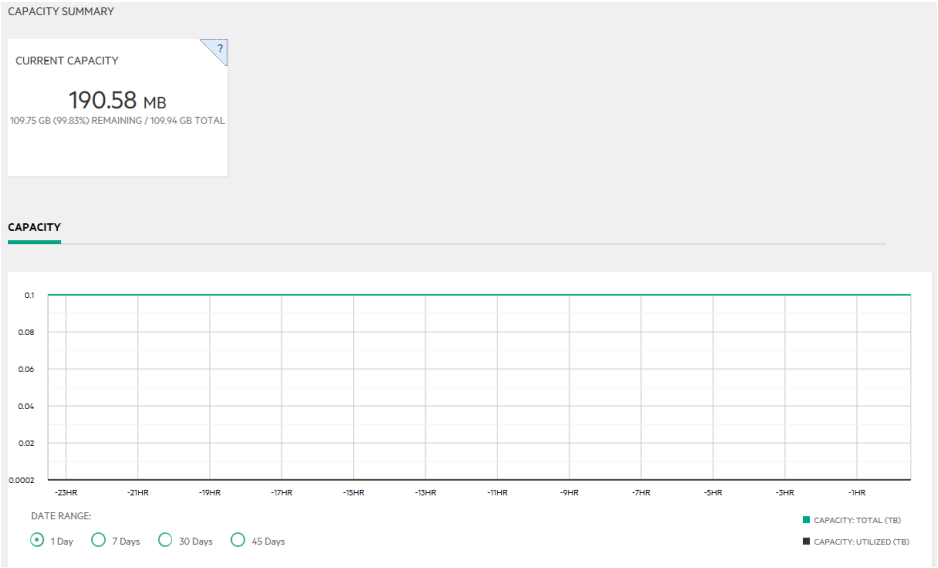
1. To open Operations Console, in a browser enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   ardana > https://myardana.test:9095
   https://VIP:9095
   ```

2. On the *Home* screen, click the menu represented by 3 horizontal lines (☰).

3. In the menu, click *Storage* › *Object Storage Summary*.

4. On the **Summary** page, click **Capacity Summary**.



1.4.4 Alarm Summary

Use this page to quickly see the most recent alarms and triage all alarms related to object storage.

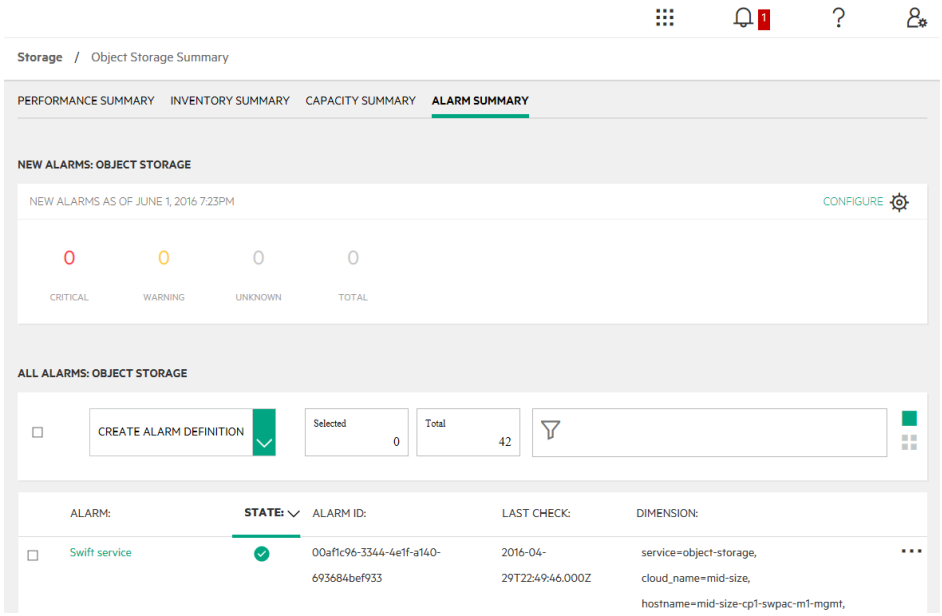
To access the object storage alarm summary screen:

1. To open Operations Console, in a browser enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   ardana > https://myardana.test:9095
   ```

```
https://VIP:9095
```

2. On the *Home* screen, click the menu represented by 3 horizontal lines (☰).
3. In the menu, click *Storage* › *Object Storage Summary*.
4. On the **Summary** page, click **Alarm Summary**.



Each row has a checkbox to allow you to select multiple alarms and set the same condition on them.

The *State* column displays a graphical indicator representing the state of each alarm:

- Green indicator: OK. Good operating state.
- Yellow indicator: Warning. Low severity, not requiring immediate action.
- Red indicator: Alarm. Varying severity levels and must be addressed.
- Gray indicator: Undetermined.

The *Alarm* column identifies the alarm by the name it was given when it was originally created.

The *Last Check* column displays the date and time the most recent occurrence of the alarm.

The *Dimension* column describes the components to check in order to clear the alarm.

The last column, depicted by three dots, reveals an *Actions* menu that allows you to choose:

- *View Details*, which opens a separate window that shows all the information from the table view and the alarm history.

Comments can be updated by clicking *Update Comment*. Click *View Alarm Definition* to go to the *Alarm Definition* tab showing that specific alarm definition.

1.5 Visualizing Data in Charts

Operations Console allows you to create a new chart and select the time range and the metric you want to chart, based on Monasca metrics.

Present data in a pictorial or graphical format to enable administrators and decision makers to grasp difficult concepts or identify new patterns.

Create new time-series graphs from *My Dashboard*.

My Dashboard also allows you to customize the view in the following ways:

- Include alarm cards from the Central Dashboard
- Customize graphs in new ways
- Reorder items using drag and drop

Plan for future storage

- Track capacity over time to predict with some degree of reliability the amount of additional storage needed.

Charts and graphs provide a quick way to visualize large amounts of complex data. It is especially useful when trying to find relationships and understand your data, which could include thousands or even millions of variables. You can create a new chart in Operations Console from **My Dashboard**.

The charts in Operations Console are based on Monasca data. When you create a new chart you will be able to select the time range and the metric you want to chart. The list of Metrics you can choose from is equivalent to using the **monasca metric-name-list** on the command line. After you select a metric, you can then specify a dimension, which is derived from the **monasca metric-list –name <metric_name>** command line results. The dimension list changes based on the selected metric.

This topic provides instructions on how to create a basic chart, and how to create a chart specifically to visualize your Cinder capacity.

- Create a Chart: *Section 1.5.1, “Create a Chart”*
- Chart Cinder Capacity: *Section 1.5.2, “Chart Cinder Capacity”*

1.5.1 Create a Chart

Create a chart to visually display data for up to 6 metrics over a period of time.

To create a chart:

1. To open Operations Console, in a browser, enter either the URL or Virtual IP connected to Operations Console.
   For example:

   ```
   https://myardana.test:9095
   https://VIP:9095
   ```

2. On the **Home** screen, click the menu represented by 3 horizontal lines (☰).

3. From the menu that slides in on the left, select **Home**, and then select **My Dashboard**.

4. On the **My Dashboard** screen, select **Create New Chart**.

5. On the **Add New Time Series Chart** screen, in **Chart Definition** complete any of the optional fields:

   Name

   Short description of chart.

   Time Range

   Specifies the interval between metric collection. The default is **1 hour**. Can be set to hours (1,2,4,8,24) or days (7,30,45).

   Chart Update Rate

   Collects metric data and adds it to the chart at the specified interval. The default is **1 minute**. Can be set to minutes (1,5,10,30) or 1 hour.

   Chart Type

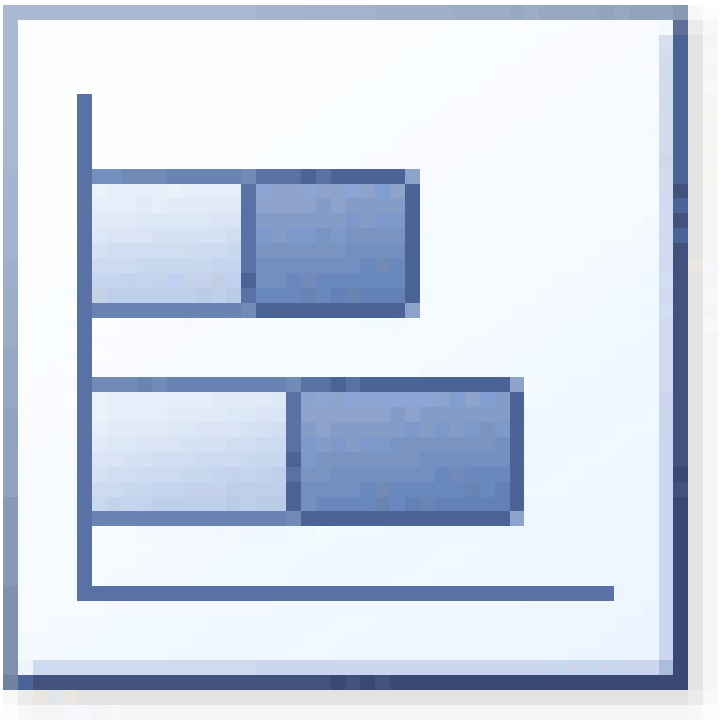
Determines how the data is displayed. The default type is **Line**. Can be set to the following values:

- 

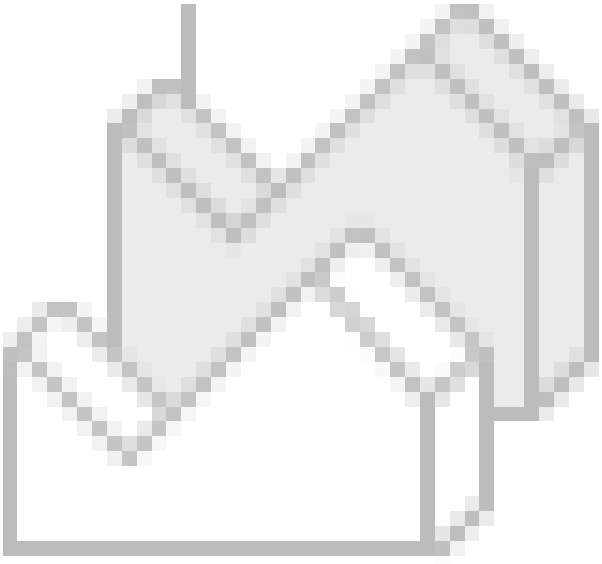
Line

- 

Bar

- 

Stacked Bar

- 

Area

Stacked Area

Chart Size

This controls the visual display of the chart width as it appears on **My Dashboard**. The default is **Small**. This field can be set to **Small** to display it at 50% or **Large** for 100%.

6. On the **Add New Time Series Chart** screen, in **Added Chart Data** complete the following fields:

Metric

In Monasca, a metric is a multi-dimensional description that consists of the following fields: name, dimensions, timestamp, value and value_meta. The pre-populated list is equivalent to using the **monasca metric-name-list** on the command line.

Dimension

The set of unique dimensions that are defined for a specific metric. Dimensions are a dictionary of key-value pairs. This pre-populated list is equivalent to using the **monasca metric-list –name <metric_name>** on the command line.

Function

Operations Console uses Monasca to provide the results of all mathematical functions. Monasca in turns uses Graphite to perform the mathematical calculations and return the results. The default is **AVG**. The **Function** field can be set to **AVG** (default), **MIN**, **MAX**. and **COUNT**. For more information on these functions, see the Graphite documentation at http://www.aosabook.org/en/graphite.html.

7. Click **Add Data To Chart**. To add another metric to the chart, repeat the previous step until all metrics are added. The maximum you can have in one chart is 6 metrics.

8. To create the chart, click **Create New Chart**.

After you click **Create New Chart**, you will be returned to *My Dashboard* where the new chart will be shown. From the *My Dashboard* screen you can use the menu in the top-right corner of the card to delete or edit the chart. You can also select an option to create a comma-delimited file of the data in the chart.

1.5.2 Chart Cinder Capacity

To visualize the use of storage capacity over time, you can create a chart that graphs the total block storage backend capacity. To find out how much of that total is being used, you can also create a chart that graphs the available block storage capacity.

Visualizing Cinder:

- Chart Total Capacity: *Section 1.5.3, "Chart Total Capacity"*
- Chart Available Capacity: *Section 1.5.4, "Chart Available Capacity"*

Important

The total and free capacity values are based on the available capacity reported by the Cinder backend. Be aware that some backends can be configured to thinly provision.

1.5.3 Chart Total Capacity

To chart the total block-storage backend capacity:

1. Log in to Operations Console.
2. Follow the steps in the previous instructions to start creating a chart.
3. To chart the total backend capacity, on the **Add New Time Series Chart** screen, in **Chart Definition** use the following settings:

Fiel	Setting
Met-rics	cinderlm.cinder.backend.total.size
Di-men-sion	any hostname. If multiple backends are available, select any one. The backends will all return the same metric data.

4. Add the data to the chart and click **Create**.

Example of a Cinder Total Capacity Chart:

1.5.4 Chart Available Capacity

To chart the available block-storage backend capacity:

1. Log in to Operations Console.
2. Follow the steps in the previous instructions to start creating a chart.
3. To chart the available backend capacity, on the **Add New Time Series Chart** screen, in **Chart Definition** use the following settings:

Fiel	Setting
Met-rics	cinderlm.cinder.backend.total.avail

Fie	Setting
Di- men- sion	any hostname. If multiple backends are available, select any one. The backends will all return the same metric data.

4. Add the data to the chart and click **Create**.

Example of a chart showing Cinder Available Capacity:

Important

The source data for the Capacity Summary pages is only refreshed at the top of each hour. This affects the latency of the displayed data on those pages.

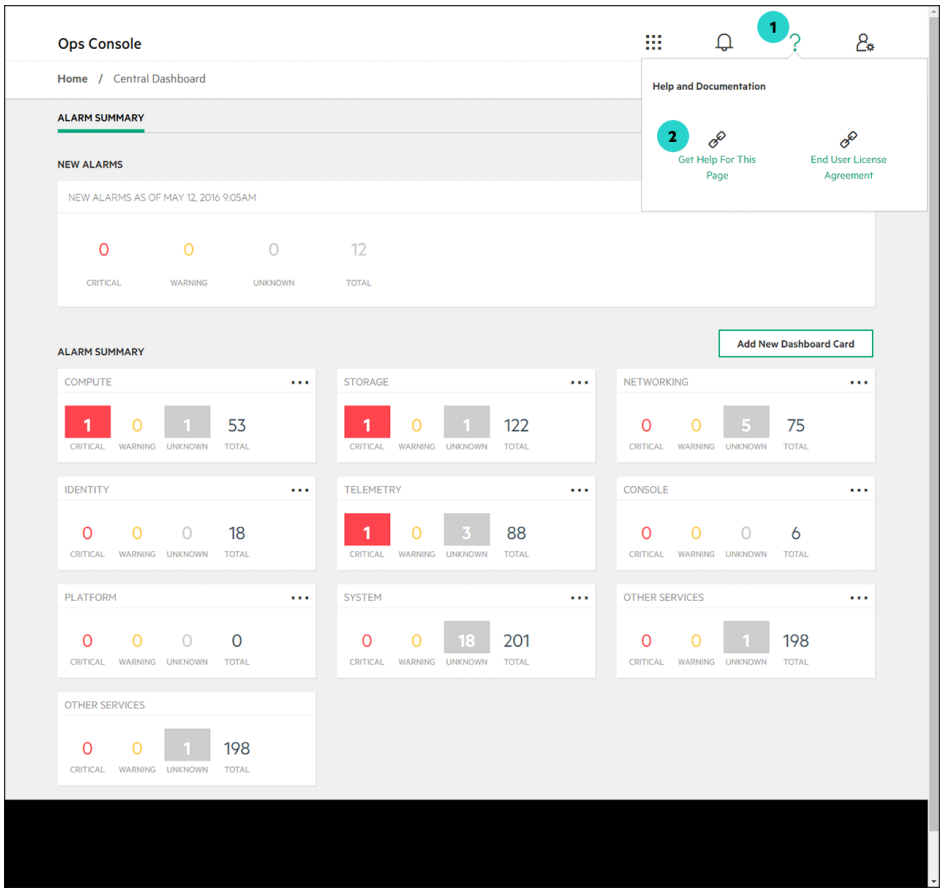
1.6 Getting Help with the Operations Console

On each of the Operations Console pages there is a help menu that you can click on to take you to a help page specific to the console you are currently viewing.

To reach the help page:

1. Click the help menu option in the upper-right corner of the page, depicted by the question mark seen in the screenshot below.

2. Click the *Get Help For This Page* link which will open the help page in a new tab in your browser.


Ops Console
1
Home / Central Dashboard
Help and Documentation
2
Get Help For This Page
End User License Agreement
ALARM SUMMARY
NEW ALARMS
NEW ALARMS AS OF MAY 12, 2016 9:05AM
0 CRITICAL
0 WARNING
0 UNKNOWN
12 TOTAL
ALARM SUMMARY
Add New Dashboard Card
COMPUTE
1 CRITICAL 0 WARNING 1 UNKNOWN 53 TOTAL
STORAGE
1 CRITICAL 0 WARNING 1 UNKNOWN 122 TOTAL
NETWORKING
0 CRITICAL 0 WARNING 5 UNKNOWN 75 TOTAL
IDENTITY
0 CRITICAL 0 WARNING 0 UNKNOWN 18 TOTAL
TELEMETRY
1 CRITICAL 0 WARNING 3 UNKNOWN 88 TOTAL
CONSOLE
0 CRITICAL 0 WARNING 0 UNKNOWN 6 TOTAL
PLATFORM
0 CRITICAL 0 WARNING 0 UNKNOWN 0 TOTAL
SYSTEM
0 CRITICAL 0 WARNING 18 UNKNOWN 201 TOTAL
OTHER SERVICES
0 CRITICAL 0 WARNING 1 UNKNOWN 198 TOTAL
OTHER SERVICES
0 CRITICAL 0 WARNING 1 UNKNOWN 198 TOTAL

2 Using the Dashboard

Often referred to as Horizon or the Horizon dashboard, you can use this console to manage resources on a domain and project level in a web-based graphical user interface (GUI).

Horizon is the OpenStack service that serves as the basis for the SUSE OpenStack Cloud dashboards.

The dashboards provide a web-based user interface to SUSE OpenStack Cloud services including Compute, Volume Operations, Networking, and Identity.

2.1 Accessing Horizon

To access the Horizon dashboard as a non cloud administrator, you should ask the cloud admin for the host name or public IP address of the dashboard, and for your user name and password. If you are a cloud admin, you can determine the URL and credentials by following the steps in *Chapter 3, Cloud Admin Actions with the Dashboard*.

Note

If you are authenticating against an external source such as an LDAP directory or via WebSSO you cannot change your password via the change password option in User settings.

Along the left side of the Horizon user interface are sections that provide access to Project and Settings sections. If your login credentials have been assigned the 'admin' role you will also see a separate Admin section that provides additional system-wide setting options.

Across the top are menus to switch between projects and menus where you can access user settings.

2.2 Browser support for Horizon

According to OpenStack.org, Horizon is tested and supported on the latest versions of Firefox and Chrome, and IE9+. Below is a summary of browser support for Horizon

Very good

Very well tested; should work as expected

Good

Moderately tested; should look nice and work fine, may have a few visual issues

Poor

Visually inferior

Broken

Essential functionality not working

No

Not supported

Browser	Status	Versions
Chrome	HPE Tested	43.0.2357.81
Firefox	HPE Tested	31 +
Firefox ESR	HPE Tested	31 +
Internet Explorer 11	HPE Tested	11.0.9600.18314
Internet Explorer 10	Not HPE Tested	
Internet Explorer 9	Not HPE Tested	
Internet Explorer 8 and below	Not supported	
Safari	Not HPE Tested	
Opera	Not HPE Tested	

Note

Please refer to the OpenStack Dashboard ("Horizon") wiki (https://wiki.openstack.org/wiki/Horizon) for additional compatibility and testing information.

At the bottom of this page, you can see some of the services you can manage right from the dashboard. Remember that Horizon runs over TLS so you will need to use the HTTPS protocol with your Horizon IP address.

2.3 Dashboard Use

As a Cloud or Domain admin of the SUSE OpenStack Cloud dashboard, you should follow the included recommendations for continued success with your cloud.

Important

- Do not create projects in the *default* domain. If you would like to create projects for your users, please create a new domain to host the new projects.
- If you are a Cloud admin, you may make changes to the *default* domain, however changes should be made from the command-line and not through the dashboard.
- If you are a Domain admin, you will need to modify the domain from the command-line and not from the dashboard.

2.4 Project dashboard

Use the **Project** dashboard to implement and build out your cloud. This dashboard contains tools to create virtual server instances, create and configure your network, configure access tools (such as key pairs and security groups) and cloud resource templates (stacks).

2.5 Admin dashboard

Use the **Admin** dashboard to view, allocate, and manage all resources within the cloud.

The Admin dashboard allows you to manage instances, define flavors, create and configure images, manage networks, view system resources, manage projects, and manage users.

2.6 Settings dashboard

Use the **Settings** dashboard to change your display language and settings, your time zone, and your password.

Click **Settings** in the user menu to display the Settings dashboard.

2.7 Accessing the Dashboard

1. Open a web browser that has both JavaScript and cookies enabled.
2. In the address bar, enter the host name or IP address for the dashboard.

   Note

   If a certificate warning appears when you try to access the URL for the first time, a self-signed certificate is in use. These certificates are not considered trustworthy by default. Verify the certificate or add an exception in the browser to bypass the warning. For information about managing certificates, see *Book "Security Guide", Chapter 7 "Transport Layer Security (TLS) Overview", Section 7.2 "TLS Configuration"*.

3. On the **Log In** page, enter your user name and password and then click **Connect**.
4. The UserID with the `admin` role is provided with a special tab called Admin. The menu options underneath this tab are similar to the non-admin user views but cover all projects as well as some additional panels not available to general users.
   The Overview page provides a summary by project of the current utilization of major OpenStack resources. The information on this screen is drawn directly from the Compute or Nova service API and provides the ability to filter the query by date range and an option to download results into a CSV file that is compatible with Microsoft Excel or can be imported into a database.
5. If you are a non-admin user, you will only be able to select a project from the list at the top of the screen.

2.8 Horizon Dashboard

There are several services you can manage in the Horizon dashboard.

Viewing System Information

The `System Information` panel provides a list of all currently installed OpenStack services and their operation status.

Additional panels on the `System Information` display sub-service status for services like Compute, Block Storage, and Network services. This are tabs for Compute Services, Block Storage Services, Network Agents, and Orchestration Services.

Managing Quotas

In the left hand menu, *Admin* › *Systems* › *Defaults* shows a list of Quota Names such as VCPUs, RAM, Instances, and storage. Click *Update Defaults* to manage Default Quotas.

Managing Routers

In the menu, *Project* › *Network* › *Routers* displays a list of routers. Routers connect end user networks together as well as to provide external connectivity. This panel shows all currently deployed routers and associated metadata. From here the system administrator can create, edit, and delete routers.

Managing Host Aggregates

Selecting *Admin* › *Compute* › *Host Aggregates* displays a list of host aggregates along with Availability Zones and Hosts. You can manage Host Aggregates in the *Actions* drop-down box.

Hypervisor Status

Selecting *Admin* › *Compute* › *Hypervisors* displays a view of total consumed resources, the hypervisors currently running, the physical server for each hypervisor instance, the type of hypervisor, and the infrastructure resources consumed by the hypervisor.

The *Compute Host* tab lists the currently configured physical hosts, the Availability Zone of each host, status, and a button to *Disable Service*.

2.9 Other Panels

- The **Updates and Extensions** panel provides a list of available files that can be downloaded and installed. This view is also only available to UserIDs with the `admin` role.

  ### Note

  Administrators should be cautious to ensure that Horizon Extensions they install come from a trusted source and/or have undergone appropriate review and testing. Administrators should be aware of the risk of spear phishing, by which they may receive an offer to install an apparently useful application which also performs malicious actions.

- **Host aggregates** are a grouping of physical servers or hosts with associated metadata. A host can belong to multiple aggregates.

- Common use cases for host aggregates include supporting the scheduling of instances to a subset of hosts that have a specific capability or flavor such as a specific type of storage, lots of RAM, and/or large numbers of processors.
  Another use case of host aggregates is to support the arrangement of hosts into logical groups for load balancing and instance distribution. Host aggregates are configured and only viewable by system admins. The end user view of a host aggregate is called an Availability Zone. Availability zones are created via the Nova API and the host aggregates function. End users can use availability zones for similar use cases. For example, an application could be deployed on hosts in multiple availability zones. A load balancer can then be configured and the instances running each deployment of the application in each availability zone can be assigned to the load balancer thus providing a measure of failover and high availability support.
  Additional information on host aggregates and availability zones is available at http://blog.russellbryant.net/2013/05/21/availability-zones-and-host-aggregates-in-openstack-compute-nova/ and http://docs.openstack.org/openstack-ops/content/scaling.html.

- **The Instances panel** provides a view of all deployed end user instances across all projects, the images applied to those instances, the status of each instance, flavor type, and network information. This information is also supplied by the Nova API.

- **The Volumes panel** is displayed if the OpenStack Cinder service is available. Cinder provides block storage infrastructure. Block storage provides the ability for end users to attach storage as discrete drives to a compute instance. These drives are persistent and can be reassigned to a different instance. The Volumes tab on this view shows all currently created volumes and associated metadata and status, which physical host the volume is hosted on, and the ability for the admin to manage each volume.

- **Flavors** are virtual hardware templates. Flavors are used to define a virtual compute host. Flavors are assigned during the creation of a new instance. This panel displays all of the current flavor types and provides the ability for the admin to create, edit or delete flavors. Hosting organizations often use flavors as a form of billing unit; flavors with more resources are charged at a higher rate than flavors with fewer resources.

- **Images** are discrete software builds that can be loaded onto an instance. An image can be a standalone operating system implementation or a bundle of operating system and other pre-configured applications. This panel displays all currently available images and associated metadata and enables the system administrator to create, edit and delete images.

Important

If you encounter the following error:

```
ERROR: Unable to create new image. URL scheme not supported.
```

The error message indicates that the Glance service has not been configured to support the import of Glance images via the *http* option. This option can be enabled by your system administrator by enabling this function in the Glance configuration file.

- **Networks** are used by end users to provide connectivity to and between their deployed instances. This panel provides a list of all currently deployed networks and associated metadata and enables the system administrator the ability to created, edit, and delete networks.
- **Quotas** provide the means to limit resource usage. Quotas are assigned per project. This panel provides a view of all current quotas and enables a system administrator with the ability to edit quota line items.
- **The Identity panel** has two major panels. The Projects panel provides a list of all current projects and enables a system administrator the ability to create, modify, and delete panels and panel membership.
- **The Users panel** is available only to system administrators and provides the ability to create, modify, and delete users. These users are considered "local" to Keystone and are authenticated by Keystone.

3 Cloud Admin Actions with the Dashboard

The Horizon dashboard provides cloud admins a web GUI to perform domain admin tasks such as user and project administration and managing project quotas.

Cloud admins can use the dashboard GUI to perform domain admin tasks such as user and project administration and managing project quotas.

3.1 Cloud Admin

As a Cloud Admin, you can create new domains, projects, users or remove them. You can also give admin privileges to users you create.

3.2 Accessing the dashboard

Prior to accessing the dashboard you will need to retrieve your admin user password. For more details, see *Book "Operations Guide", Chapter 4 "Managing Identity", Section 4.6 "Retrieving the Admin Password"*.

If your administrator set a hostname value for `external_name` in your `network_groups.yml` file during the configuration process for your cloud then Horizon will be accessed over port 443 (https) on that hostname.

If your administrator did not set a hostname value then in order to determine which IP address to use to access Horizon you can use this command from your Cloud Lifecycle Manager node:

```
grep HZN-WEB /etc/hosts
```

The output of that command will show you the virtual IP address for Horizon that you should use.

The default username for the Administrator user is `admin`. You will use that along with the password you retrieved earlier and your domain name when logging in.

3.3 Cloud Admin Activities

You must log into the Dashboard via the default domain using a cloud admin username and password to create, modify or remove domains, projects or users. Look at the Project picker at the top of the Dashboard UI and ensure that the **Default** domain and the **admin** project BOTH have checks beside them. The **demo** project is selected by default, and this will cause issues if not changed.

3.4 Create a New Domain

You can create new domains as a Cloud Admin.

1. Select **Identity** and **Domains** in the sidebar
2. Click **Create Domain** from the list above the domains table
3. Enter a domain name and description.
4. The **Enabled** checkbox is checked by default
5. Click the **Create Domain** button

3.5 Set Domain context to the newly created Domain

After you have created a new domain as the Cloud Admin, set the context.

1. Select **Identity** and **Domains** in the sidebar.
2. Click **Set Domain Context** button on the newly created domain.

3.6 Create a New Project

You can create new projects.

1. Select **Identity** and **Projects** in the sidebar.
2. Click **Create Project** from the list above the projects table

3. The `Domain ID` and `Name` are immutably set to the new domain.
4. Enter a project name and description
5. The **Enabled** checkbox is checked by default.
6. Click *Create Project* button.
7. The project created will be displayed.

3.7 Create a New User

Warning

Do not create users in the default domain with admin privileges unless you want them to act as Cloud Admin users.

1. Select **Identity** and **Users** in the sidebar
2. Click **Create User** from the list above the users table
3. `Domain ID` and `Name` are immutably set to the new domain.
4. Enter a user name, description, email address, password.
5. The new project is preselected.
6. The role `_Member_` is preselected.
7. The `Enabled` checkbox is checked by default.
8. Click *Create User* button.
9. The user created will be displayed.

3.8 Remove a user from a project

1. Select **Identity** and **Projects** in the sidebar
2. Click **Manage Members** in the actions column for the project.

3. In the right column, click the minus ("") beside the user to remove them from the project
4. Click **Save** to save the change

3.9 Assign Admin role to User within the new Domain

You can assign admin privileges to a user within a domain.

Warning

Assigning admin privileges to a user in the default domain gives them Cloud Admin privileges.

1. Select **Identity** and **Domains** in the sidebar
2. Click **Manage Members** in the actions column for the domain
3. If necessary, in the left column, click the plus ("+") beside the user that should be the domain admin.
4. In the right column, click the dropdown beside the domain admin user and ensure that ONLY the **admin** role is selected by clicking it. Unselect any other roles selected by clicking them.
5. Click **Save** to save the change

3.10 Setting and managing quotas

Infrastructure resource quotas are also defined within the OpenStack cloud. Quotas are defined per service but can be assigned via the Horizon user interface in the Projects section; thus, this function is often associated with the Keystone API.

Quotas are assigned per each project.

Quotas can be managed using the CLI. Instructions can be found on OpenStack web site in Managing Quotas (http://docs.openstack.org/user-guide-admin/cli_set_quotas.html)

In Horizon, selecting *Admin* › *Systems* › *Defaults* from the menu provides a list of system quotas that can viewed and edited.

3.11 Sign out of the Dashboard

Sign out of the Dashboard by clicking *Sign Out*.

4 Cloud Admin Actions with the Command Line

Cloud admins can use the command line tools to perform domain admin tasks such as user and project administration.

4.1 Creating Additional Cloud Admins

You can create additional Cloud Admins to help with the administration of your cloud.

Keystone identity service query and administration tasks can be performed using the OpenStack command line utility. The utility is installed by the Cloud Lifecycle Manager onto the Cloud Lifecycle Manager.

Note

Keystone administration tasks should be performed by an *admin* user with a token scoped to the *default* domain via the Keystone v3 identity API. These settings are preconfigured in the file `~/keystone.osrc`. By default, `keystone.osrc` is configured with the admin endpoint of Keystone. If the admin endpoint is not accessible from your network, change `OS_AUTH_URL` to point to the public endpoint.

4.2 Command Line Examples

For a full list of OpenStackClient commands, see OpenStackClient Command List (http://docs.openstack.org/developer/python-openstackclient/command-list.html).

Sourcing the Keystone Administration Credentials

You can set the environment variables needed for identity administration by sourcing the `keystone.osrc` file created by the lifecycle manager:

```
source ~/keystone.osrc
```

List users in the default domain

These users are created by the Cloud Lifecycle Manager in the MySQL back end:

```
openstack user list
```

Example output:

```
$ openstack user list
+----------------------------------+------------------+
| ID                               | Name             |
+----------------------------------+------------------+
| 155b68eda9634725a1d32c5025b91919 | heat             |
| 303375d5e44d48f298685db7e6a4efce | octavia          |
| 40099e245a394e7f8bb2aa91243168ee | logging          |
| 452596adbf4d49a28cb3768d20a56e38 | admin            |
| 76971c3ad2274820ad5347d46d7560ec | designate        |
| 7b2dc0b5bb8e4ffb92fc338f3fa02bf3 | hlm_backup       |
| 86d345c960e34c9189519548fe13a594 | barbican         |
| 8e7027ab438c4920b5853d52f1e08a22 | nova_monasca     |
| 9c57dfff57e2400190ab04955e7d82a0 | barbican_service |
| a3f99bcc71b242a1bf79dbc9024eec77 | nova             |
| aeeb56fc4c4f40e0a6a938761f7b154a | glance-check     |
| af1ef292a8bb46d9a1167db4da48ac65 | cinder           |
| af3000158c6d4d3d9257462c9cc68dda | demo             |
| b41a7d0cb1264d949614dc66f6449870 | swift            |
| b78a2b17336b43368fb15fea5ed089e9 | cinderinternal   |
| bae1718dee2d47e6a75cd6196fb940bd | monasca          |
| d4b9b32f660943668c9f5963f1ff43f9 | ceilometer       |
| d7bef811fb7e4d8282f19fb3ee5089e9 | swift-monitor    |
| df58b381ca8a4c9bb42e371f2a78ef4f | freezer          |
| e22bbb2be91342fd9afa20baad4cd490 | neutron          |
| ec0ad2418a644e6b995d8af3eb5ff195 | glance           |
| ef16c37ec7a648338eaf53c029d6e904 | swift-dispersion |
| ef1a6daccb6f4694a27a1c41cc5e7a31 | glance-swift     |
| fed3a599b0864f5b80420c9e387b4901 | monasca-agent    |
+----------------------------------+------------------+
```

List domains created by the installation process:

```
openstack domain list
```

Example output:

```
$ openstack domain list
+----------------------------------+---------+---------
+-------------------------------------------------------------------+
| ID                               | Name    | Enabled | Description
                                   |
+----------------------------------+---------+---------
+-------------------------------------------------------------------+
| 6740dbf7465a4108a36d6476fc967dbd | heat    | True    | Owns users and projects created
 by heat                            |
```

```
| default                          | Default | True    | Owns users and tenants (i.e.
 projects) available on Identity API v2. |
+----------------------------------+---------+---------
+--------------------------------------------------------------------------+
```

List the roles:

```
openstack role list
```

Example output:

```
$ openstack role list
+----------------------------------+--------------------------+
| ID                               | Name                     |
+----------------------------------+--------------------------+
| 0be3da26cd3f4cd38d490b4f1a8b0c03 | designate_admin          |
| 13ce16e4e714473285824df8188ee7c0 | monasca-agent            |
| 160f25204add485890bc95a6065b9954 | key-manager:service-admin |
| 27755430b38c411c9ef07f1b78b5ebd7 | monitor                  |
| 2b8eb0a261344fbb8b6b3d5934745fe1 | key-manager:observer     |
| 345f1ec5ab3b4206a7bffdeb5318bd32 | admin                    |
| 49ba3b42696841cea5da8398d0a5d68e | nova_admin               |
| 5129400d4f934d4fbfc2c3dd608b41d9 | ResellerAdmin            |
| 60bc2c44f8c7460a9786232a444b56a5 | neutron_admin            |
| 654bf409c3c94aab8f929e9e82048612 | cinder_admin             |
| 854e542baa144240bfc761cdb5fe0c07 | monitoring-delegate      |
| 8946dbdfa3d346b2aa36fa5941b43643 | key-manager:auditor      |
| 901453d9a4934610ad0d56434d9276b4 | key-manager:admin        |
| 9bc90d1121544e60a39adbfe624a46bc | monasca-user             |
| 9fe2a84a3e7443ae868d1009d6ab4521 | service                  |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_                 |
| a24d4e0a5de14bffbe166bfd68b36e6a | swiftoperator            |
| ae088fcbf579425580ee4593bfa680e5 | heat_stack_user          |
| bfba56b2562942e5a2e09b7ed939f01b | KeystoneAdmin            |
| c05f54cf4bb34c7cb3a4b2b46c2a448b | glance_admin             |
| cd61735400ca4fa19e80cebe9970d9a7 | Member                   |
| fe010be5c57240db8f559e0114a380c1 | key-manager:creator      |
+----------------------------------+--------------------------+
```

List admin user role assignment within default domain:

```
openstack role assignment list --user admin --domain default
```

Example output:

```
# This indicates that the admin user is assigned the admin role within the default domain
ardana >  openstack role assignment list --user admin --domain default
```

```
+----------------------------------+----------------------------------+-------+---------
+---------+
| Role                             | User                             | Group | Project |
 Domain  |
+----------------------------------+----------------------------------+-------+---------
+---------+
| b398322103504546a070d607d02618ad | fed1c038d9e64392890b6b44c38f5bbb |       |         |
 default |
+----------------------------------+----------------------------------+-------+---------
+---------+
```

Create a new user in default domain:

```
openstack user create --domain default --password-prompt --email <email_address> --
description <description> --enable <username>
```

Example output showing the creation of a user named `testuser` with email address `test@example.com` and a description of `Test User`:

```
ardana >  openstack user create --domain default --password-prompt --email
 test@example.com --description "Test User" --enable testuser
User Password:
Repeat User Password:
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Test User                        |
| domain_id   | default                          |
| email       | test@example.com                 |
| enabled     | True                             |
| id          | 8aad69acacf0457e9690abf8c557754b |
| name        | testuser                         |
+-------------+----------------------------------+
```

Assign admin role for testuser within the default domain:

```
openstack role add admin --user <username> --domain default
openstack role assignment list --user <username> --domain default
```

Example output:

```
# Just for demonstration purposes - do not do this in a production environment!
ardana >  openstack role add admin --user testuser --domain default
ardana >  openstack role assignment list --user testuser --domain default
+----------------------------------+----------------------------------+-------+---------
+---------+
| Role                             | User                             | Group | Project |
 Domain  |
```

```
+----------------------------------+----------------------------------+-------+---------
+---------+
| b398322103504546a070d607d02618ad | 8aad69acacf0457e9690abf8c557754b |       |         |
 default |
+----------------------------------+----------------------------------+-------+---------
+---------+
```

4.3 Assigning the default service admin roles

The following examples illustrate how you can assign each of the new service admin roles to a user.

Assigning the glance_admin role

A user must have the role of admin in order to assign the glance_admin role. To assign the role, you will set the environment variables needed for the identity service administrator.

1. First, source the identity service credentials:

   ```
   source ~/keystone.osrc
   ```

2. You can add the glance_admin role to a user on a project with this command:

   ```
   openstack role add --user <username> --project <project_name> glance_admin
   ```

   Example, showing a user named `testuser` being granted the `glance_admin` role in the `test_project` project:

   ```
   openstack role add --user testuser --project test_project glance_admin
   ```

3. You can confirm the role assignment by listing out the roles:

   ```
   openstack role assignment list --user <username>
   ```

   Example output:

   ```
   ardana >  openstack role assignment list --user testuser
   +----------------------------------+----------------------------------+-------
   +----------------------------------+--------+-----------+
   | Role                             | User                             | Group |
    Project                          | Domain | Inherited |
   +----------------------------------+----------------------------------+-------
   +----------------------------------+--------+-----------+
   ```

```
| 46ba80078bc64853b051c964db918816 | 8bcfe10101964e0c8ebc4de391f3e345 |       |
 0ebbf7640d7948d2a17ac08bbbf0ca5b |        | False     |
+----------------------------------+----------------------------------+-------
+----------------------------------+--------+-----------+
```

4. Note that only the role ID is displayed. To get the role name, execute the following:

```
openstack role show <role_id>
```

   Example output:

```
ardana >  openstack role show 46ba80078bc64853b051c964db918816
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 46ba80078bc64853b051c964db918816 |
| name  | glance_admin                     |
+-------+----------------------------------+
```

5. To demonstrate that the user has Glance admin privileges, authenticate with those user creds and then upload and publish an image. Only a user with an admin role or glance_admin can publish an image.

   a. The easiest way to do this will be to make a copy of the `service.osrc` file and edit it with your user credentials. You can do that with this command:

```
cp ~/service.osrc ~/user.osrc
```

   b. Using your preferred editor, edit the `user.osrc` file and replace the values for the following entries to match your user credentials:

```
export OS_USERNAME=<username>
export OS_PASSWORD=<password>
```

   c. You will also need to edit the following lines for your environment:

```
## Change these values from 'unset' to 'export'
export OS_PROJECT_NAME=<project_name>
export OS_PROJECT_DOMAIN_NAME=Default
```

      Here is an example output:

```
unset OS_DOMAIN_NAME
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_VERSION=3
```

```
export OS_PROJECT_NAME=test_project
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USERNAME=testuser
export OS_USER_DOMAIN_NAME=Default
export OS_PASSWORD=testuser
export OS_AUTH_URL=http://192.168.245.9:35357/v3
export OS_ENDPOINT_TYPE=internalURL
# OpenstackClient uses OS_INTERFACE instead of OS_ENDPOINT
export OS_INTERFACE=internal
export OS_CACERT=/etc/ssl/certs/ca-certificates.crt
```

6. Source the environment variables for your user:

```
source ~/user.osrc
```

7. Upload an image and publicize it:

```
glance image-create --name "upload me" --visibility public --container-format bare
 --disk-format qcow2 --file uploadme.txt
```

Example output:

```
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | dd75c3b840a16570088ef12f6415dd15     |
| container_format | bare                                 |
| created_at       | 2016-01-06T23:31:27Z                 |
| disk_format      | qcow2                                |
| id               | cf1490f4-1eb1-477c-92e8-15ebbe91da03 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | upload me                            |
| owner            | bd24897932074780a20b780c4dde34c7     |
| protected        | False                                |
| size             | 10                                   |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2016-01-06T23:31:31Z                 |
| virtual_size     | None                                 |
| visibility       | public                               |
+------------------+--------------------------------------+
```

Note

You can use the command **`glance help image-create`** to get the full syntax for this command.

Assigning the nova_admin role

A user must have the role of admin in order to assign the nova_admin role. To assign the role, you will set the environment variables needed for the identity service administrator.

1. First, source the identity service credentials:

```
source ~/keystone.osrc
```

2. You can add the glance_admin role to a user on a project with this command:

```
openstack role add --user <username> --project <project_name> nova_admin
```

Example, showing a user named `testuser` being granted the `glance_admin` role in the `test_project` project:

```
openstack role add --user testuser --project test_project nova_admin
```

3. You can confirm the role assignment by listing out the roles:

```
openstack role assignment list --user <username>
```

Example output:

```
ardana >  openstack role assignment list --user testuser
+----------------------------------+----------------------------------+-------
+----------------------------------+--------+-----------+
| Role                              | User                              | Group |
 Project                           | Domain | Inherited |
+----------------------------------+----------------------------------+-------
+----------------------------------+--------+-----------+
| 8cdb02bab38347f3b65753099f3ab73c | 8bcfe10101964e0c8ebc4de391f3e345 |       |
 0ebbf7640d7948d2a17ac08bbbf0ca5b |        | False     |
+----------------------------------+----------------------------------+-------
+----------------------------------+--------+-----------+
```

4. Note that only the role ID is displayed. To get the role name, execute the following:

```
openstack role show <role_id>
```

Example output:

```
ardana >  openstack role show 8cdb02bab38347f3b65753099f3ab73c
+-------+----------------------------------+
| Field | Value                            |
+-------+----------------------------------+
| id    | 8cdb02bab38347f3b65753099f3ab73c |
| name  | nova_admin                       |
+-------+----------------------------------+
```

5. To demonstrate that the user has Nova admin privileges, authenticate with those user creds and then upload and publish an image. Only a user with an admin role or glance_admin can publish an image.

   a. The easiest way to do this will be to make a copy of the `service.osrc` file and edit it with your user credentials. You can do that with this command:

   ```
   cp ~/service.osrc ~/user.osrc
   ```

   b. Using your preferred editor, edit the `user.osrc` file and replace the values for the following entries to match your user credentials:

   ```
   export OS_USERNAME=<username>
   export OS_PASSWORD=<password>
   ```

   c. You will also need to edit the following lines for your environment:

   ```
   ## Change these values from 'unset' to 'export'
   export OS_PROJECT_NAME=<project_name>
   export OS_PROJECT_DOMAIN_NAME=Default
   ```

   Here is an example output:

   ```
   unset OS_DOMAIN_NAME
   export OS_IDENTITY_API_VERSION=3
   export OS_AUTH_VERSION=3
   export OS_PROJECT_NAME=test_project
   export OS_PROJECT_DOMAIN_NAME=Default
   export OS_USERNAME=testuser
   export OS_USER_DOMAIN_NAME=Default
   export OS_PASSWORD=testuser
   export OS_AUTH_URL=http://192.168.245.9:35357/v3
   export OS_ENDPOINT_TYPE=internalURL
   # OpenstackClient uses OS_INTERFACE instead of OS_ENDPOINT
   ```

```
export OS_INTERFACE=internal
export OS_CACERT=/etc/ssl/certs/ca-certificates.crt
```

6. Source the environment variables for your user:

```
source ~/user.osrc
```

7. List all of the virtual machines in the project specified in user.osrc:

```
openstack server list
```

   Example output showing no virtual machines, because there are no virtual machines created on the project specified in the user.osrc file:

```
+--------------------------------------
+------------------------------------------------------+--------
+------------------------------------------------------------------+
| ID                                   | Name
           | Status | Networks
  |
+--------------------------------------
+------------------------------------------------------+--------
+------------------------------------------------------------------+
+--------------------------------------
+------------------------------------------------------+--------
+------------------------------------------------------------------+
```

8. For this demonstration, we do have a virtual machine associated with a different project and because your user has nova_admin permissions, you can view those virtual machines using a slightly different command:

```
openstack server list --all-projects
```

   Example output, now showing a virtual machine:

```
ardana >  openstack server list --all-projects
+--------------------------------------
+------------------------------------------------------+--------
+------------------------------------------------------------------+
| ID                                   | Name
           | Status | Networks
  |
+--------------------------------------
+------------------------------------------------------+--------
+------------------------------------------------------------------+
```

```
| da4f46e2-4432-411b-82f7-71ab546f91f3 | testvml
           | ACTIVE |
  |
+-------------------------------------
+-----------------------------------------------------+--------
+-------------------------------------------------------------+
```

9. You can also now delete virtual machines in other projects by using the `--all-tenants` switch:

   ```
   openstack server delete --all-projects <instance_id>
   ```

   Example, showing us deleting the instance in the previous step:

   ```
   openstack server delete --all-projects da4f46e2-4432-411b-82f7-71ab546f91f3
   ```

10. You can get a full list of available commands by using this:

    ```
    openstack -h
    ```

You can perform the same steps as above for the Neutron and Cinder service admin roles:

```
neutron_admin
cinder_admin
```

4.4 Customize policy.json on the Cloud Lifecycle Manager

One way to deploy `policy.json` for a service is by going to each of the target nodes and making changes there. This is not necessary anymore. This process has been streamlined and policy.json files can be edited on the Cloud Lifecycle Manager and then deployed to nodes. Please exercise caution when modifying policy.json files. It is best to validate the changes in a non-production environment before rolling out policy.json changes into production. It is not recommended that you make policy.json changes without a way to validate the desired policy behavior. Updated policy.json files can be deployed using the appropriate `<service_name>-reconfigure.yml` playbook.

4.5 Roles

Service roles represent the functionality used to implement the OpenStack role based access control (RBAC) model. This is used to manage access to each OpenStack service. Roles are named and assigned per user or group for each project by the identity service. Role definition and policy enforcement are defined outside of the identity service independently by each OpenStack service.

The token generated by the identity service for each user authentication contains the role(s) assigned to that user for a particular project. When a user attempts to access a specific OpenStack service, the role is parsed by the service, compared to the service-specific policy file, and then granted the resource access defined for that role by the service policy file.

Each service has its own service policy file with the `/etc/[SERVICE_CODENAME]/policy.json` file name format where `[SERVICE_CODENAME]` represents a specific OpenStack service name. For example, the OpenStack Nova service would have a policy file called `/etc/nova/policy.json`.

Service policy files can be modified and deployed to control nodes from the Cloud Lifecycle Manager. Administrators are advised to validate policy changes before checking in the changes to the site branch of the local git repository before rolling the changes into production. Do not make changes to policy files without having a way to validate them.

The policy files are located at the following site branch directory on the Cloud Lifecycle Manager.

```
~/openstack/ardana/ansible/roles/
```

For test and validation, policy files can be modified in a non-production environment from the `~/scratch/` directory. For a specific policy file, run a search for `policy.json`. To deploy policy changes for a service, run the service specific reconfiguration playbook (for example, `nova-reconfigure.yml`). For a complete list of reconfiguration playbooks, change directories to `~/scratch/ansible/next/ardana/ansible` and run this command:

```
ls –l | grep reconfigure
```

Note

Comments added to any `*.j2` files (including templates) must follow proper comment syntax. Otherwise you may see errors when running the config-processor or any of the service playbooks.

5 Installing the Command-Line Clients

During the installation, by default, the suite of OpenStack command-line tools are installed on the Cloud Lifecycle Manager and the control plane in your environment. This includes the OpenStack Command-Line Interface as well as the clients for the individual services such as the NovaClient, CinderClient, and SwiftClient. You can learn more about these in the OpenStack documentation here: OpenStack Command-Line Interface Reference (http://docs.openstack.org/cli-reference/).

If you wish to install the command-line interfaces on other nodes in your environment, there are two methods you can use to do so that we describe below.

5.1 Installing the CLI tools using the input model

During the initial install phase of your cloud you can edit your input model to request that the command-line clients be installed on any of the node clusters in your environment. To do so, follow these steps:

1. Log in to the Cloud Lifecycle Manager.
2. Edit your `control_plane.yml` file. Full path:

   ```
   ~/openstack/my_cloud/definition/data/control_plane.yml
   ```

3. In this file you will see a list of `service-components` to be installed on each of your clusters. These clusters will be divided per role, with your controller node cluster likely coming at the beginning. Here you will see a list of each of the clients that can be installed. These include:

   ```
   keystone-client
   glance-client
   cinder-client
   nova-client
   neutron-client
   swift-client
   heat-client
   openstack-client
   ceilometer-client
   monasca-client
   barbican-client
   ```

```
designate-client
```

4. For each client you want to install, specify the name under the `service-components` section for the cluster you want to install it on.
   So, for example, if you would like to install the Nova and Neutron clients on your Compute node cluster, you can do so by adding the `nova-client` and `neutron-client` services, like this:

```
      resources:
        - name: compute
          resource-prefix: comp
          server-role: COMPUTE-ROLE
          allocation-policy: any
          min-count: 0
          service-components:
            - ntp-client
            - nova-compute
            - nova-compute-kvm
            - neutron-l3-agent
            - neutron-metadata-agent
            - neutron-openvswitch-agent
            - neutron-lbaasv2-agent
            - nova-client
            - neutron-client
```

Note

This example uses the `entry-scale-kvm` sample file. Your model may be different so use this as a guide but do not copy and paste the contents of this example into your input model.

5. Commit your configuration to the local git repo, as follows:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "My config or other commit message"
```

6. Continue with the rest of your installation.

5.2 Installing the CLI tools using Ansible

At any point after your initial installation you can install the command-line clients on any of the nodes in your environment. To do so, follow these steps:

1. Log in to the Cloud Lifecycle Manager.
2. Obtain the hostname for the nodes you want to install the clients on by looking in your hosts file:

   ```
   cat /etc/hosts
   ```

3. Install the clients using this playbook, specifying your hostnames using commas:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts -e "install_package=<client_name>" client-
   deploy.yml -e "install_hosts=<hostname>"
   ```

   So, for example, if you would like to install the NovaClient on two of your Compute nodes with hostnames `ardana-cp1-comp0001-mgmt` and `ardana-cp1-comp0002-mgmt` you can use this syntax:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts -e "install_package=novaclient" client-
   deploy.yml -e "install_hosts=ardana-cp1-comp0001-mgmt,ardana-cp1-comp0002-mgmt"
   ```

4. Once the playbook completes successfully, you should be able to SSH to those nodes and, using the proper credentials, authenticate and use the command-line interfaces you have installed.

6 Creating a Highly Available Router

6.1 CVR and DVR High Available Routers

CVR (Centralized Virtual Routing) and DVR (Distributed Virtual Routing) are two types of technologies which can be used to provide routing processes in SUSE OpenStack Cloud 8. You can create Highly Available (HA) versions of CVR and DVR routers by using the options in the table below when creating your router.

The neutron command for creating a router `neutron router-create router_name --distributed=True|False --ha=True|False` requires administrative permissions. See the example in the next section, Creating a High Availability Router.

--distributed	--ha	Router Type	Description
False	False	CVR	Centralized Virtual Router
False	True	CVRHA	Centralized Virtual Router with L3 High Availability
True	False	DVR	Distributed Virtual Router without SNAT High Availability
True	True	DVRHA	Distributed Virtual Router with SNAT High Availability

6.2 Creating a High Availability Router

You can create a highly available router using the neutron command line interface.

1. To create the HA router simply add `--ha=True` to the neutron router-create command. If you want to also make the router distributed, add `--distributed=True`. In this example, a DVR SNAT HA router is created with the name `routerHA`.

   ```
   $ neutron router-create routerHA --distributed=True --ha=True
   ```

2. Set the gateway for the external network and add interface

```
$ neutron router-gateway-set routerHA <ext-net-id>
$ neutron router-interface-add routerHA <private_subnet_id>
```

3. Once the router is created, gateway set and interface attached, you now have a router with high availability.

6.3 Test Router for High Availability

You can demonstrate that the router is HA by running a continuous ping from a VM instance that is running on the private network to an external server such as a public DNS. As the ping is running, list the l3 agents hosting the router and identify the agent that is responsible for hosting the active router. Induce the failover mechanism by creating a catastrophic event like shutting down node hosting the l3 agent. Once the node is shut down, you will see that the ping from the VM to the external network continues to run as the backup l3 agent takes over. To verify the agent hosting the primary router has changed, list the agents hosting the router. You will see a different agent is now hosting the active router.

1. Boot an instance on the private network

```
$ nova boot --image <image_id> --flavor <flavor_id> --nic net_id=<private_net_id> --
key_name <key> VM1
```

2. Log into the VM using the ssh keys

```
ssh -i <key> <ipaddress of VM1>
```

3. Start a ping to X.X.X.X. While pinging, make sure there is no packet loss and leave the ping running.

```
$ ping X.X.X.X
```

4. Check which agent is hosting the active router

```
$ neutron l3-agent-list-hosting-router <router_id>
```

5. Shutdown the node hosting the agent.

6. Within 10 seconds, check again to see which L3 agent is hosting the active router

```
$ neutron l3-agent-list-hosting-router <router_id>
```

7. You will see a different agent.

II Project User Guide

7 Using Container as a Service (Magnum)

The SUSE OpenStack Cloud Magnum Service provides container orchestration engines such as Docker Swarm, Kubernetes, and Apache Mesos available as first class resources. SUSE OpenStack Cloud Magnum uses Heat to orchestrate an OS image which contains Docker and Kubernetes and runs that image in either virtual machines or bare metal in a cluster configuration.

7.1 Deploying a Kubernetes Cluster on Fedora Atomic

7.1.1 Prerequisites

These steps assume the following have been completed:

- The Magnum service has been installed. For more information, see *Book "Installing with Cloud Lifecycle Manager", Chapter 14 "Magnum Overview", Section 14.2 "Install the Magnum Service"*.
- Deploying a Kubernetes Cluster on Fedora Atomic requires the Fedora Atomic image **fedora-atomic-26-20170723.0.x86_64.qcow2** prepared specifically for the OpenStack release. You can download the **fedora-atomic-26-20170723.0.x86_64.qcow2** image from https://fedorapeople.org/groups/magnum/

7.1.2 Creating the Cluster

The following example is created using Kubernetes Container Orchestration Engine (COE) running on Fedora Atomic guest OS on SUSE OpenStack Cloud VMs.

1. As **stack** user, login to the lifecycle manager.
2. Source openstack admin credentials.

   ```
   $ source service.osrc
   ```

3. If you haven't already, download Fedora Atomic image, prepared for the Openstack Pike release.

```
$ wget https://download.fedoraproject.org/pub/alt/atomic/stable/
Fedora-Atomic-26-20170723.0/CloudImages/x86_64/images/Fedora-
Atomic-26-20170723.0.x86_64.qcow2
```

4. Create a Glance image.

```
$ glance image-create --name fedora-atomic-26-20170723.0.x86_64 --visibility public
 \
  --disk-format qcow2 --os-distro fedora-atomic --container-format bare \
  --file Fedora-Atomic-26-20170723.0.x86_64.qcow2 --progress
[=============================>] 100%
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 9d233b8e7fbb7ea93f20cc839beb09ab     |
| container_format | bare                                 |
| created_at       | 2017-04-10T21:13:48Z                 |
| disk_format      | qcow2                                |
| id               | 4277115a-f254-46c0-9fb0-fffc45d2fd38 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | fedora-atomic-26-20170723.0.x86_64   |
| os_distro        | fedora-atomic                        |
| owner            | 2f5b83ab49d54aaea4b39f5082301d09     |
| protected        | False                                |
| size             | 515112960                            |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2017-04-10T21:13:56Z                 |
| virtual_size     | None                                 |
| visibility       | public                               |
+------------------+--------------------------------------+
```

5. Create a Nova keypair.

```
$ test -f ~/.ssh/id_rsa.pub || ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub testkey
```

6. Create a Magnum cluster template.

```
$ magnum cluster-template-create --name my-template \
  --image-id 4277115a-f254-46c0-9fb0-fffc45d2fd38 \
  --keypair-id testkey \
  --external-network-id ext-net \
  --dns-nameserver 8.8.8.8 \
  --flavor-id m1.small \
```

```
  --docker-volume-size 5 \
  --network-driver flannel \
  --coe kubernetes \
  --http-proxy http://proxy.yourcompany.net:8080/ \
  --https-proxy http://proxy.yourcompany.net:8080/
```

Note

a. Use the *image_id* from `glance image-create` command output in the previous step.

b. Use your organization's DNS server. If the SUSE OpenStack Cloud public endpoint is configured with the hostname, this server should provide resolution for this hostname.

c. The proxy is only needed if public internet (for example, `https://discovery.etcd.io/` or `https://gcr.io/`) is not accessible without proxy.

7. Create cluster. The command below will create a minimalistic cluster consisting of a single Kubernetes Master (kubemaster) and single Kubernetes Node (worker, kubeminion).

```
$ magnum cluster-create --name my-cluster --cluster-template my-template --node-
count 1 --master-count 1
```

8. Immediately after issuing `cluster-create` command, cluster status should turn to **CREATE_IN_PROGRESS** and stack_id assigned.

```
$ magnum cluster-show my-cluster
+--------------------+------------------------------------------------------------+
| Property           | Value                                                      |
+--------------------+------------------------------------------------------------+
| status             | CREATE_IN_PROGRESS                                         |
| cluster_template_id | 245c6bf8-c609-4ea5-855a-4e672996cbbc                       |
| uuid               | 0b78a205-8543-4589-8344-48b8cfc24709                       |
| stack_id           | 22385a42-9e15-49d9-a382-f28acef36810                       |
| status_reason      | -                                                          |
| created_at         | 2017-04-10T21:25:11+00:00                                  |
| name               | my-cluster                                                 |
| updated_at         | -                                                          |
| discovery_url      | https://discovery.etcd.io/193d122f869c497c2638021eae1ab0f7 |
| api_address        | -                                                          |
| coe_version        | -                                                          |
| master_addresses   | []                                                         |
```

```
| create_timeout      | 60                                                                  |
| node_addresses      | []                                                                  |
| master_count        | 1                                                                   |
| container_version   | -                                                                   |
| node_count          | 1                                                                   |
+---------------------+---------------------------------------------------------------------+
```

9. You can monitor cluster creation progress by listing the resources of the Heat stack. Use the `stack_id` value from the `magnum cluster-status` output above in the following command:

```
$ heat resource-list -n2 22385a42-9e15-49d9-a382-f28acef36810
WARNING (shell) "heat resource-list" is deprecated, please use "openstack stack
 resource list" instead
+-------------------------------+-------------------------------------
+-----------------------------------+-------------------+----------------------
+-------------------------+
| resource_name                 | physical_resource_id                |
 resource_type                    | resource_status    | updated_time         |
 stack_name              |
+-------------------------------+-------------------------------------
+-----------------------------------+-------------------+----------------------
+-------------------------+
| api_address_floating_switch   | 06b2cc0d-77f9-4633-8d96-f51e2db1faf3 |
 Magnum::FloatingIPAddressSwitcher | CREATE_COMPLETE    | 2017-04-10T21:25:10Z | my-
cluster-z4aquda2mgpv |
| api_address_lb_switch         | 965124ca-5f62-4545-bbae-8d9cda7aff2e |
 Magnum::ApiGatewaySwitcher       | CREATE_COMPLETE    | 2017-04-10T21:25:10Z | my-
cluster-z4aquda2mgpv |
. . .
```

10. The cluster is complete when all resources show **CREATE_COMPLETE**.

11. Install kubectl onto your Cloud Lifecycle Manager.

```
$ export https_proxy=http://proxy.yourcompany.net:8080
$ wget https://storage.googleapis.com/kubernetes-release/release/v1.2.0/bin/linux/
amd64/kubectl
$ chmod +x ./kubectl
$ sudo mv ./kubectl /usr/local/bin/kubectl
```

12. Generate the cluster configuration using **`magnum cluster-config`**. If the CLI option `--tls-disabled` was not specified during cluster template creation, authentication in the cluster will be turned on. In this case, **`magnum cluster-config`** command will generate

client authentication certificate (`cert.pem`) and key (`key.pem`). Copy and paste **`magnum cluster-config`** output to your command line input to finalize configuration (that is, export KUBECONFIG environment variable).

```
$ mkdir my_cluster
$ cd my_cluster
/my_cluster $ ls
/my_cluster $ magnum cluster-config my-cluster
export KUBECONFIG=./config
/my_cluster $ ls
ca.pem cert.pem config key.pem
/my_cluster $ export KUBECONFIG=./config
/my_cluster $ kubectl version
Client Version: version.Info{Major:"1", Minor:"2", GitVersion:"v1.2.0",
 GitCommit:"5cb86ee022267586db386f62781338b0483733b3", GitTreeState:"clean"}
Server Version: version.Info{Major:"1", Minor:"2", GitVersion:"v1.2.0",
 GitCommit:"cffae0523cfa80ddf917aba69f08508b91f603d5", GitTreeState:"clean"}
```

13. Create a simple Nginx replication controller, exposed as a service of type NodePort.

```
$ cat >nginx.yml <<-EOF
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-controller
spec:
  replicas: 1
  selector:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
```

```
  ports:
  - port: 80
    nodePort: 30080
  selector:
    app: nginx
EOF

$ kubectl create -f nginx.yml
```

14. Check pod status until it turns from **Pending** to **Running**.

```
$ kubectl get pods
NAME                       READY    STATUS     RESTARTS    AGE
nginx-controller-5cmev     1/1      Running    0           2m
```

15. Ensure that the Nginx welcome page is displayed at port 30080 using the kubemaster floating IP.

```
$ http_proxy= curl http://172.31.0.6:30080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

16. If LBaaS v2 is enabled in SUSE OpenStack Cloud environment, and your cluster was created with more than one kubemaster, a new load balancer can be created to perform request rotation between several masters. For more information about LBaaS v2 support, see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 31 "Configuring Load Balancer as a Service"*.

7.2 Deploying a Kubernetes Cluster on CoreOS

7.2.1 Prerequisites

These steps assume the following have been completed:

- The Magnum service has been installed. For more information, see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 14 "Magnum Overview", Section 14.2 "Install the Magnum Service"*.
- Creating the Magnum cluster requires the CoreOS image for OpenStack. You can download compressed image file **coreos_production_openstack_image.img.bz2** from http://stable.release.core-os.net/amd64-usr/current/.

7.2.2 Creating the Cluster

The following example is created using Kubernetes Container Orchestration Engine (COE) running on CoreOS guest OS on SUSE OpenStack Cloud VMs.

1. Login to the Cloud Lifecycle Manager.
2. Source openstack admin credentials.

```
$ source service.osrc
```

3. If you haven't already, download CoreOS image that is compatible for the OpenStack release.

Note

The https_proxy is only needed if your environment requires a proxy.

```
$ export https_proxy=http://proxy.yourcompany.net:8080
$ wget https://stable.release.core-os.net/amd64-usr/current/
coreos_production_openstack_image.img.bz2
$ bunzip2 coreos_production_openstack_image.img.bz2
```

4. Create a Glance image.

```
$ glance image-create --name coreos-magnum --visibility public \
  --disk-format raw --os-distro coreos --container-format bare \
  --file coreos_production_openstack_image.img --progress
[=============================>] 100%
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 4110469bb15af72ec0cf78c2da4268fa     |
| container_format | bare                                 |
| created_at       | 2017-04-25T18:10:52Z                 |
| disk_format      | raw                                  |
| id               | c25fc719-2171-437f-9542-fcb8a534fbd1 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | coreos-magnum                        |
| os_distro        | coreos                               |
| owner            | 2f5b83ab49d54aaea4b39f5082301d09     |
| protected        | False                                |
| size             | 806551552                            |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2017-04-25T18:11:07Z                 |
| virtual_size     | None                                 |
| visibility       | public                               |
+------------------+--------------------------------------+
```

5. Create a Nova keypair.

```
$ test -f ~/.ssh/id_rsa.pub || ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub testkey
```

6. Create a Magnum cluster template.

```
$ magnum cluster-template-create --name my-coreos-template \
  --image-id c25fc719-2171-437f-9542-fcb8a534fbd1 \
  --keypair-id testkey \
  --external-network-id ext-net \
  --dns-nameserver 8.8.8.8 \
  --flavor-id m1.small \
  --docker-volume-size 5 \
  --network-driver flannel \
  --coe kubernetes \
  --http-proxy http://proxy.yourcompany.net:8080/ \
  --https-proxy http://proxy.yourcompany.net:8080/
```

Note

a. Use the *image_id* from `glance image-create` command output in the previous step.

b. Use your organization's DNS server. If the SUSE OpenStack Cloud public endpoint is configured with the hostname, this server should provide resolution for this hostname.

c. The proxy is only needed if public internet (for example, `https://discovery.etcd.io/` or `https://gcr.io/`) is not accessible without proxy.

7. Create cluster. The command below will create a minimalistic cluster consisting of a single Kubernetes Master (kubemaster) and single Kubernetes Node (worker, kubeminion).

```
$ magnum cluster-create --name my-coreos-cluster --cluster-template my-coreos-
template --node-count 1 --master-count 1
```

8. Almost immediately after issuing `cluster-create` command, cluster status should turn to **CREATE_IN_PROGRESS** and stack_id assigned.

```
$ magnum cluster-show my-coreos-cluster
+---------------------+------------------------------------------------------------+
| Property            | Value                                                      |
+---------------------+------------------------------------------------------------+
| status              | CREATE_IN_PROGRESS                                         |
| cluster_template_id | c48fa7c0-8dd9-4da4-b599-9e62dc942ca5                       |
| uuid                | 6b85e013-f7c3-4fd3-81ea-4ea34201fd45                       |
| stack_id            | c93f873a-d563-4721-9bd9-3bae2340750a                       |
| status_reason       | -                                                          |
| created_at          | 2017-04-25T22:38:43+00:00                                  |
| name                | my-coreos-cluster                                          |
| updated_at          | -                                                          |
| discovery_url       | https://discovery.etcd.io/6e4c0e5ff5e5b9872173d06880886a0c |
| api_address         | -                                                          |
| coe_version         | -                                                          |
| master_addresses    | []                                                         |
| create_timeout      | 60                                                         |
| node_addresses      | []                                                         |
| master_count        | 1                                                          |
| container_version   | -                                                          |
| node_count          | 1                                                          |
```

```
+--------------------+--------------------------------------------------------+
```

9. You can monitor cluster creation progress by listing the resources of the Heat stack. Use the `stack_id` value from the `magnum cluster-status` output above in the following command:

```
$ heat resource-list -n2 c93f873a-d563-4721-9bd9-3bae2340750a
WARNING (shell) "heat resource-list" is deprecated, please use "openstack stack
 resource list" instead
+--------------------------------
+-------------------------------------------------------------------------------------------
+-------------------------------------------------------------------
-------------------------------------------------------------------
+-------------------+---------------------
+-------------------------------------------------------------------------+
| resource_name                  | physical_resource_id
                                   | resource_type
                                                                  | resource_status
 | updated_time         | stack_name
              |
+--------------------------------
+-------------------------------------------------------------------------------------------
+-------------------------------------------------------------------
-------------------------------------------------------------------
+-------------------+---------------------
+-------------------------------------------------------------------------+
| api_address_switch             |
                                   | Magnum::ApiGatewaySwitcher
                                                                  | INIT_COMPLETE
 | 2017-04-25T22:38:42Z | my-coreos-cluster-mscybll54eoj
              |
| api_listener                   | 1dc1c599-b552-4f03-94e9-936fbb889741
                                   | Magnum::Optional::Neutron::LBaaS::Listener
                                                                  | CREATE_COMPLETE
 | 2017-04-25T22:38:42Z | my-coreos-cluster-mscybll54eoj
              |
| api_loadbalancer               | bf1d8c64-a75f-4eec-8f2f-373d49aee581
                                   | Magnum::Optional::Neutron::LBaaS::LoadBalancer
                                                                  | CREATE_COMPLETE
 | 2017-04-25T22:38:42Z | my-coreos-cluster-mscybll54eoj
              |
. . .
```

10. The cluster is complete when all resources show **CREATE_COMPLETE**.

11. Install kubectl onto your Cloud Lifecycle Manager.

```
$ export https_proxy=http://proxy.yourcompany.net:8080
$ wget https://storage.googleapis.com/kubernetes-release/release/v1.2.0/bin/linux/
amd64/kubectl
$ chmod +x ./kubectl
$ sudo mv ./kubectl /usr/local/bin/kubectl
```

12. Generate the cluster configuration using **`magnum cluster-config`**. If the CLI option `--tls-disabled` was not specified during cluster template creation, authentication in the cluster will be turned on. In this case, **`magnum cluster-config`** command will generate client authentication certificate (`cert.pem`) and key (`key.pem`). Copy and paste **`magnum cluster-config`** output to your command line input to finalize configuration (that is, export KUBECONFIG environment variable).

```
$ mkdir my_cluster
$ cd my_cluster
/my_cluster $ ls
/my_cluster $ magnum cluster-config my-cluster
export KUBECONFIG=./config
/my_cluster $ ls
ca.pem cert.pem config key.pem
/my_cluster $ export KUBECONFIG=./config
/my_cluster $ kubectl version
Client Version: version.Info{Major:"1", Minor:"2", GitVersion:"v1.2.0",
 GitCommit:"5cb86ee022267586db386f62781338b0483733b3", GitTreeState:"clean"}
Server Version: version.Info{Major:"1", Minor:"2", GitVersion:"v1.2.0",
 GitCommit:"cffae0523cfa80ddf917aba69f08508b91f603d5", GitTreeState:"clean"}
```

13. Create a simple Nginx replication controller, exposed as a service of type NodePort.

```
$ cat >nginx.yml <<-EOF
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-controller
spec:
  replicas: 1
  selector:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
```

```
          image: nginx
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  ports:
  - port: 80
    nodePort: 30080
  selector:
    app: nginx
EOF

$ kubectl create -f nginx.yml
```

14. Check pod status until it turns from **Pending** to **Running**.

```
$ kubectl get pods
NAME                       READY    STATUS     RESTARTS    AGE
nginx-controller-5cmev     1/1      Running    0           2m
```

15. Ensure that the Nginx welcome page is displayed at port 30080 using the kubemaster floating IP.

```
$ http_proxy= curl http://172.31.0.6:30080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

16. If LBaaS v2 is enabled in SUSE OpenStack Cloud environment, and your cluster was created with more than one kubemaster, a new load balancer can be created to perform request rotation between several masters. For more information about LBaaS v2 support, see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 31 "Configuring Load Balancer as a Service"*.

7.3 Deploying a Docker Swarm Cluster on Fedora Atomic

7.3.1 Prerequisites

These steps assume the following have been completed:

- The Magnum service has been installed. For more information, see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 14 "Magnum Overview", Section 14.2 "Install the Magnum Service"*.
- Deploying a Docker Swarm Cluster on Fedora Atomic requires the Fedora Atomic image **fedora-atomic-26-20170723.0.x86_64.qcow2** prepared specifically for the OpenStack Pike release. You can download the **fedora-atomic-26-20170723.0.x86_64.qcow2** image from https://download.fedoraproject.org/pub/alt/atomic/stable/Fedora-Atomic-26-20170723.0/CloudImages/x86_64/

7.3.2 Creating the Cluster

The following example is created using Kubernetes Container Orchestration Engine (COE) running on Fedora Atomic guest OS on SUSE OpenStack Cloud VMs.

1. As **stack** user, login to the lifecycle manager.
2. Source openstack admin credentials.

```
$ source service.osrc
```

3. If you haven't already, download Fedora Atomic image, prepared for Openstack Pike release.

```
$ wget https://download.fedoraproject.org/pub/alt/atomic/stable/
Fedora-Atomic-26-20170723.0/CloudImages/x86_64/images/Fedora-
Atomic-26-20170723.0.x86_64.qcow2
```

4. Create a Glance image.

```
$ glance image-create --name fedora-atomic-26-20170723.0.x86_64 --visibility public
 \
```

```
  --disk-format qcow2 --os-distro fedora-atomic --container-format bare \
  --file Fedora-Atomic-26-20170723.0.x86_64.qcow2 --progress
[=============================>] 100%
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 9d233b8e7fbb7ea93f20cc839beb09ab     |
| container_format | bare                                 |
| created_at       | 2017-04-10T21:13:48Z                 |
| disk_format      | qcow2                                |
| id               | 4277115a-f254-46c0-9fb0-fffc45d2fd38 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | fedora-atomic-26-20170723.0.x86_64   |
| os_distro        | fedora-atomic                        |
| owner            | 2f5b83ab49d54aaea4b39f5082301d09     |
| protected        | False                                |
| size             | 515112960                            |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2017-04-10T21:13:56Z                 |
| virtual_size     | None                                 |
| visibility       | public                               |
+------------------+--------------------------------------+
```

5. Create a Nova keypair.

```
$ test -f ~/.ssh/id_rsa.pub || ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub testkey
```

6. Create a Magnum cluster template.

Note

The `--tls-disabled` flag is not specified in the included template. Authentication via client certificate will be turned on in clusters created from this template.

```
$  magnum cluster-template-create --name my-swarm-template \
  --image-id 4277115a-f254-46c0-9fb0-fffc45d2fd38 \
  --keypair-id testkey \
  --external-network-id ext-net \
  --dns-nameserver 8.8.8.8 \
  --flavor-id m1.small \
  --docker-volume-size 5 \
  --network-driver docker \
```

```
  --coe swarm \
  --http-proxy http://proxy.yourcompany.net:8080/ \
  --https-proxy http://proxy.yourcompany.net:8080/
```

Note

a. Use the `image_id` from `glance image-create` command output in the previous step.

b. Use your organization's DNS server. If the SUSE OpenStack Cloud public endpoint is configured with the hostname, this server should provide resolution for this hostname.

c. The proxy is only needed if public internet (for example, `https://discovery.etcd.io/` or `https://gcr.io/`) is not accessible without proxy.

7. Create cluster. The command below will create a minimalistic cluster consisting of a single Kubernetes Master (kubemaster) and single Kubernetes Node (worker, kubeminion).

```
$ magnum cluster-create --name my-swarm-cluster --cluster-template my-swarm-template
 \
  --node-count 1 --master-count 1
```

8. Immediately after issuing `cluster-create` command, cluster status should turn to **CREATE_IN_PROGRESS** and `stack_id` assigned.

```
$ magnum cluster-show my-swarm-cluster
+---------------------+------------------------------------------------------------+
| Property            | Value                                                      |
+---------------------+------------------------------------------------------------+
| status              | CREATE_IN_PROGRESS                                         |
| cluster_template_id | 17df266e-f8e1-4056-bdee-71cf3b1483e3                       |
| uuid                | c3e13e5b-85c7-44f4-839f-43878fe5f1f8                       |
| stack_id            | 3265d843-3677-4fed-bbb7-e0f56c27905a                       |
| status_reason       | -                                                          |
| created_at          | 2017-04-21T17:13:08+00:00                                  |
| name                | my-swarm-cluster                                           |
| updated_at          | -                                                          |
| discovery_url       | https://discovery.etcd.io/54e83ea168313b0c2109d0f66cd0aa6f |
| api_address         | -                                                          |
| coe_version         | -                                                          |
| master_addresses    | []                                                         |
| create_timeout      | 60                                                         |
```

```
| node_addresses      | []                                                          |
| master_count        | 1                                                           |
| container_version   | -                                                           |
| node_count          | 1                                                           |
+---------------------+-------------------------------------------------------------+
```

9. You can monitor cluster creation progress by listing the resources of the Heat stack. Use the `stack_id` value from the `magnum cluster-status` output above in the following command:

```
$ heat resource-list -n2 3265d843-3677-4fed-bbb7-e0f56c27905a
WARNING (shell) "heat resource-list" is deprecated, please use "openstack stack
 resource list" instead
+-------------------+-----------------------------------
+-------------------------------------------+----------------
+---------------------+------------------------------+
| resource_name      | physical_resource_id                | resource_type
                     | resource_status | updated_time         | stack_name
          |
|-------------------+-----------------------------------
+-------------------------------------------+----------------
+---------------------+------------------------------+
| api_address_switch | 430f82f2-03e3-4085-8c07-b4a6b6d7e261 |
 Magnum::ApiGatewaySwitcher                 | CREATE_COMPLETE | 2017-04-21T17:13:07Z
 | my-swarm-cluster-j7gbjcxaremy |
| api_listener       | a306fd54-e569-4673-8fd3-7ae5ebdf53c3 |
 Magnum::Optional::Neutron::LBaaS::Listener | CREATE_COMPLETE | 2017-04-21T17:13:07Z
 | my-swarm-cluster-j7gbjcxaremy |
. . .
```

10. The cluster is complete when all resources show **CREATE_COMPLETE**. You can also obtain the floating IP address once the cluster has been created.

```
$ magnum cluster-show my-swarm-cluster
+---------------------+--------------------------------------------------------------+
| Property            | Value                                                        |
+---------------------+--------------------------------------------------------------+
| status              | CREATE_COMPLETE                                              |
| cluster_template_id | 17df266e-f8e1-4056-bdee-71cf3b1483e3                         |
| uuid                | c3e13e5b-85c7-44f4-839f-43878fe5f1f8                         |
| stack_id            | 3265d843-3677-4fed-bbb7-e0f56c27905a                         |
| status_reason       | Stack CREATE completed successfully                          |
| created_at          | 2017-04-21T17:13:08+00:00                                    |
| name                | my-swarm-cluster                                             |
| updated_at          | 2017-04-21T17:18:26+00:00                                    |
| discovery_url       | https://discovery.etcd.io/54e83ea168313b0c2109d0f66cd0aa6f |
```

```
| api_address          | tcp://172.31.0.7:2376                                        |
| coe_version          | 1.0.0                                                        |
| master_addresses     | ['172.31.0.7']                                               |
| create_timeout       | 60                                                           |
| node_addresses       | ['172.31.0.5']                                               |
| master_count         | 1                                                            |
| container_version    | 1.9.1                                                        |
| node_count           | 1                                                            |
+----------------------+--------------------------------------------------------------+
```

11. Generate and sign client certificate using `magnum cluster-config` command.

```
$ mkdir my_swarm_cluster
$ cd my_swarm_cluster/
~/my_swarm_cluster $ magnum cluster-config my-swarm-cluster
{'tls': True, 'cfg_dir': '.', 'docker_host': u'tcp://172.31.0.7:2376'}
~/my_swarm_cluster $ ls
ca.pem  cert.pem  key.pem
```

12. Copy generated certificates and key to ~/.docker folder on first cluster master node.

```
$ scp -r ~/my_swarm_cluster fedora@172.31.0.7:.docker
ca.pem                                        100% 1066     1.0KB/s   00:00
key.pem                                       100% 1679     1.6KB/s   00:00
cert.pem                                      100% 1005     1.0KB/s   00:00
```

13. Login to first master node and set up cluster access environment variables.

```
$ ssh fedora@172.31.0.7
[fedora@my-6zxz5ukdu-0-bvqbsn2z2uwo-swarm-master-n6wfplu7jcwo ~]$ export
 DOCKER_TLS_VERIFY=1
[fedora@my-6zxz5ukdu-0-bvqbsn2z2uwo-swarm-master-n6wfplu7jcwo ~]$ export
 DOCKER_HOST=tcp://172.31.0.7:2376
```

14. Verfy that the swarm container is up and running.

```
[fedora@my-6zxz5ukdu-0-bvqbsn2z2uwo-swarm-master-n6wfplu7jcwo ~]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
 STATUS              PORTS               NAMES
fcbfab53148c        swarm:1.0.0         "/swarm join --addr 1"   24 minutes ago
  Up 24 minutes       2375/tcp            my-xggjts5zbgr-0-d4qhxhdujh4q-swarm-node-
vieanhwdonon.novalocal/swarm-agent
```

15. Deploy a sample docker application (nginx) and verify that Nginx is serving requests at port 8080 on worker node(s), on both floating and private IPs:

```
[fedora@my-6zxz5ukdu-0-bvqbsn2z2uwo-swarm-master-n6wfplu7jcwo ~]$ docker run -itd -p
 8080:80 nginx
192030325fef0450b7b917af38da986edd48ac5a6d9ecb1e077b017883d18802

[fedora@my-6zxz5ukdu-0-bvqbsn2z2uwo-swarm-master-n6wfplu7jcwo ~]$ docker port
 192030325fef
80/tcp -> 10.0.0.11:8080

[fedora@my-6zxz5ukdu-0-bvqbsn2z2uwo-swarm-master-n6wfplu7jcwo ~]$ curl
 http://10.0.0.11:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
...
[fedora@my-6zxz5ukdu-0-bvqbsn2z2uwo-swarm-master-n6wfplu7jcwo ~]$ curl
 http://172.31.0.5:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
...
```

16. If LBaaS v2 is enabled in your SUSE OpenStack Cloud environment, a new load balancer can be created to perform request rotation between several masters. For more information about LBaaS v2 support, see *Book "*Installing with Cloud Lifecycle Manager*", Chapter 31 "Configuring Load Balancer as a Service"*.

7.4 Deploying an Apache Mesos Cluster on Ubuntu

7.4.1 Prerequisites

These steps assume the following have been completed:

- The Magnum service has been installed. For more information, see *Book "Installing with Cloud Lifecycle Manager", Chapter 14 "Magnum Overview", Section 14.2 "Install the Magnum Service"*.
- Deploying an Apache Mesos Cluster requires the Fedora Atomic image that is compatible for the OpenStack release. You can download the **ubuntu-mesos-latest.qcow2** image from https://fedorapeople.org/groups/magnum/

7.4.2 Creating the Cluster

The following example is created using Kubernetes Container Orchestration Engine (COE) running on Fedora Atomic guest OS on SUSE OpenStack Cloud VMs.

1. As **stack** user, login to the lifecycle manager.
2. Source openstack admin credentials.

```
$ source service.osrc
```

3. If you haven't already, download Fedora Atomic image that is compatible for the OpenStack release.

Note

The `https_proxy` is only needed if your environment requires a proxy.

```
$ https_proxy=http://proxy.yourcompany.net:8080 wget https://fedorapeople.org/
groups/magnum/ubuntu-mesos-latest.qcow2
```

4. Create a Glance image.

```
$ glance image-create --name ubuntu-mesos-latest --visibility public --disk-format
 qcow2 --os-distro ubuntu --container-format bare --file ubuntu-mesos-latest.qcow2
 --progress
```

```
[==============================>] 100%
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 97cc1fdb9ca80bf80dbd6842aab7dab5     |
| container_format | bare                                 |
| created_at       | 2017-04-21T19:40:20Z                 |
| disk_format      | qcow2                                |
| id               | d6a4e6f9-9e34-4816-99fe-227e0131244f |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | ubuntu-mesos-latest                  |
| os_distro        | ubuntu                               |
| owner            | 2f5b83ab49d54aaea4b39f5082301d09     |
| protected        | False                                |
| size             | 753616384                            |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2017-04-21T19:40:32Z                 |
| virtual_size     | None                                 |
| visibility       | public                               |
+------------------+--------------------------------------+
```

5. Create a Nova keypair.

```
$ test -f ~/.ssh/id_rsa.pub || ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub testkey
```

6. Create a Magnum cluster template.

```
$ magnum cluster-template-create --name my-mesos-template \
  --image-id d6a4e6f9-9e34-4816-99fe-227e0131244f \
  --keypair-id testkey \
  --external-network-id ext-net \
  --dns-nameserver 8.8.8.8 \
  --flavor-id m1.small \
  --docker-volume-size 5 \
  --network-driver docker \
  --coe mesos \
  --http-proxy http://proxy.yourcompany.net:8080/ \
  --https-proxy http://proxy.yourcompany.net:8080/
```

Note

1. Use the *image_id* from `glance image-create` command output in the previous step.
2. Use your organization's DNS server. If the SUSE OpenStack Cloud public endpoint is configured with the hostname, this server should provide resolution for this hostname.
3. The proxy is only needed if public internet (for example, `https://discovery.etcd.io/` or `https://gcr.io/`) is not accessible without proxy.

7. Create cluster. The command below will create a minimalistic cluster consisting of a single Kubernetes Master (kubemaster) and single Kubernetes Node (worker, kubeminion).

```
$ magnum cluster-create --name my-mesos-cluster --cluster-template my-mesos-template
 --node-count 1 --master-count 1
```

8. Immediately after issuing `cluster-create` command, cluster status should turn to **CREATE_IN_PROGRESS** and stack_id assigned.

```
$ magnum cluster-show my-mesos-cluster
+---------------------+--------------------------------------+
| Property            | Value                                |
+---------------------+--------------------------------------+
| status              | CREATE_IN_PROGRESS                   |
| cluster_template_id | be354919-fa6c-4db8-9fd1-69792040f095 |
| uuid                | b1493402-8571-4683-b81e-ddc129ff8937 |
| stack_id            | 50aa20a6-bf29-4663-9181-cf7ba3070a25 |
| status_reason       | -                                    |
| created_at          | 2017-04-21T19:50:34+00:00            |
| name                | my-mesos-cluster                     |
| updated_at          | -                                    |
| discovery_url       | -                                    |
| api_address         | -                                    |
| coe_version         | -                                    |
| master_addresses    | []                                   |
| create_timeout      | 60                                   |
| node_addresses      | []                                   |
| master_count        | 1                                    |
| container_version   | -                                    |
| node_count          | 1                                    |
```

```
+--------------------+------------------------------------+
```

9. You can monitor cluster creation progress by listing the resources of the Heat stack. Use the `stack_id` value from the `magnum cluster-status` output above in the following command:

```
$ heat resource-list -n2 50aa20a6-bf29-4663-9181-cf7ba3070a25
WARNING (shell) "heat resource-list" is deprecated, please use "openstack stack
 resource list" instead
+------------------------------+-------------------------------------
+----------------------------------+-----------------+---------------------
+------------------------------+
| resource_name                | physical_resource_id                |
 resource_type                    | resource_status | updated_time         |
 stack_name                   |
+------------------------------+-------------------------------------
+----------------------------------+-----------------+---------------------
+------------------------------+
| add_proxy_master             | 10394a74-1503-44b4-969a-44258c9a7be1 |
 OS::Heat::SoftwareConfig         | CREATE_COMPLETE | 2017-04-21T19:50:33Z | my-
mesos-cluster-w2trq7m46qus |
| add_proxy_master_deployment  |                                      |
 OS::Heat::SoftwareDeploymentGroup | INIT_COMPLETE   | 2017-04-21T19:50:33Z | my-
mesos-cluster-w2trq7m46qus |
...
```

10. The cluster is complete when all resources show **CREATE_COMPLETE**.

```
$ magnum cluster-show my-mesos-cluster
+---------------------+--------------------------------------+
| Property            | Value                                |
+---------------------+--------------------------------------+
| status              | CREATE_COMPLETE                      |
| cluster_template_id | 9e942bfa-2c78-4837-82f5-6bea88ba1bf9 |
| uuid                | 9d7bb502-8865-4cbd-96fa-3cd75f0f6945 |
| stack_id            | 339a72b4-a131-47c6-8d10-365e6f6a18cf |
| status_reason       | Stack CREATE completed successfully  |
| created_at          | 2017-04-24T20:54:31+00:00            |
| name                | my-mesos-cluster                     |
| updated_at          | 2017-04-24T20:59:18+00:00            |
| discovery_url       | -                                    |
| api_address         | 172.31.0.10                          |
| coe_version         | -                                    |
| master_addresses    | ['172.31.0.10']                      |
| create_timeout      | 60                                   |
| node_addresses      | ['172.31.0.5']                       |
```

```
| master_count        | 1                                    |
| container_version   | 1.9.1                                |
| node_count          | 1                                    |
+---------------------+--------------------------------------+
```

11. Verify that Marathon (https://mesosphere.github.io/marathon/) web console is available at http://${MASTER_IP}:8080/, and Mesos (http://mesos.apache.org/documentation/latest/) UI is available at http://${MASTER_IP}:5050/

```
$ https_proxy=http://proxy.yourcompany.net:8080 curl -LO \
  https://storage.googleapis.com/kubernetes-release/release/v1.2.0/bin/linux/amd64/
kubectl
$ chmod +x ./kubectl
$ sudo mv ./kubectl /usr/local/bin/kubectl
```

12. Create an example Mesos application.

```
$ mkdir my_mesos_cluster
$ cd my_mesos_cluster/
$ cat > sample.json <<-EOFc
{
  "id": "sample",
  "cmd": "python3 -m http.server 8080",
  "cpus": 0.5,
  "mem": 32.0,
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "python:3",
      "network": "BRIDGE",
      "portMappings": [
        { "containerPort": 8080, "hostPort": 0 }
      ]
    }
  }
}
EOF
```

```
$ curl -s -X POST -H "Content-Type: application/json" \
  http://172.31.0.10:8080/v2/apps -d@sample.json | json_pp
{
   "dependencies" : [],
   "healthChecks" : [],
   "user" : null,
   "mem" : 32,
   "requirePorts" : false,
```

```
  "tasks" : [],
  "cpus" : 0.5,
  "upgradeStrategy" : {
     "minimumHealthCapacity" : 1,
     "maximumOverCapacity" : 1
  },
  "maxLaunchDelaySeconds" : 3600,
  "disk" : 0,
  "constraints" : [],
  "executor" : "",
  "cmd" : "python3 -m http.server 8080",
  "id" : "/sample",
  "labels" : {},
  "ports" : [
     0
  ],
  "storeUrls" : [],
  "instances" : 1,
  "tasksRunning" : 0,
  "tasksHealthy" : 0,
  "acceptedResourceRoles" : null,
  "env" : {},
  "tasksStaged" : 0,
  "tasksUnhealthy" : 0,
  "backoffFactor" : 1.15,
  "version" : "2017-04-25T16:37:40.657Z",
  "uris" : [],
  "args" : null,
  "container" : {
     "volumes" : [],
     "docker" : {
        "portMappings" : [
           {
              "containerPort" : 8080,
              "hostPort" : 0,
              "servicePort" : 0,
              "protocol" : "tcp"
           }
        ],
        "parameters" : [],
        "image" : "python:3",
        "forcePullImage" : false,
        "network" : "BRIDGE",
        "privileged" : false
     },
     "type" : "DOCKER"
  },
```

```
   "deployments" : [
      {
         "id" : "6fbe48f0-6a3c-44b7-922e-b172bcae1be8"
      }
   ],
   "backoffSeconds" : 1
}
```

13. Wait for sample application to start. Use REST API or Marathon web console to monitor status:

```
$ curl -s http://172.31.0.10:8080/v2/apps/sample | json_pp
{
   "app" : {
      "deployments" : [],
      "instances" : 1,
      "tasks" : [
         {
            "id" : "sample.7fdd1ee4-29d5-11e7-9ee0-02427da4ced1",
            "stagedAt" : "2017-04-25T16:37:40.807Z",
            "version" : "2017-04-25T16:37:40.657Z",
            "ports" : [
               31827
            ],
            "appId" : "/sample",
            "slaveId" : "21444bc5-3eb8-49cd-b020-77041e0c88d0-S0",
            "host" : "10.0.0.9",
            "startedAt" : "2017-04-25T16:37:42.003Z"
         }
      ],
      "upgradeStrategy" : {
         "maximumOverCapacity" : 1,
         "minimumHealthCapacity" : 1
      },
      "storeUrls" : [],
      "requirePorts" : false,
      "user" : null,
      "id" : "/sample",
      "acceptedResourceRoles" : null,
      "tasksRunning" : 1,
      "cpus" : 0.5,
      "executor" : "",
      "dependencies" : [],
      "args" : null,
      "backoffFactor" : 1.15,
      "ports" : [
         10000
```

```
      ],
      "version" : "2017-04-25T16:37:40.657Z",
      "container" : {
         "volumes" : [],
         "docker" : {
            "portMappings" : [
               {
                  "servicePort" : 10000,
                  "protocol" : "tcp",
                  "hostPort" : 0,
                  "containerPort" : 8080
               }
            ],
            "forcePullImage" : false,
            "parameters" : [],
            "image" : "python:3",
            "privileged" : false,
            "network" : "BRIDGE"
         },
         "type" : "DOCKER"
      },
      "constraints" : [],
      "tasksStaged" : 0,
      "env" : {},
      "mem" : 32,
      "disk" : 0,
      "labels" : {},
      "tasksHealthy" : 0,
      "healthChecks" : [],
      "cmd" : "python3 -m http.server 8080",
      "backoffSeconds" : 1,
      "maxLaunchDelaySeconds" : 3600,
      "versionInfo" : {
         "lastConfigChangeAt" : "2017-04-25T16:37:40.657Z",
         "lastScalingAt" : "2017-04-25T16:37:40.657Z"
      },
      "uris" : [],
      "tasksUnhealthy" : 0
   }
}
```

14. Verify that deployed application is responding on automatically assigned port on floating IP address of worker node.

```
$ curl http://172.31.0.5:31827
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/
strict.dtd">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
...
```

15. If LBaaS v2 is enabled in SUSE OpenStack Cloud environment, a new load balancer can be created to perform request rotation between several masters. For more information about LBaaS v2 support, see *Book "Installing with Cloud Lifecycle Manager", Chapter 31 "Configuring Load Balancer as a Service"*.

7.5 Creating a Magnum Cluster with the Dashboard

You can alternatively create a cluster template and cluster with the Magnum UI in Horizon. The example instructions below demonstrate how to deploy a Kubernetes Cluster using the Fedora Atomic image. Other deployments such as Kubernetes on CoreOS, Docker Swarm on Fedora, and Mesos on Ubuntu all follow the same set of instructions mentioned below with slight variations to their parameters. You can determine those parameters by looking at the previous set of CLI instructions in the **`magnum cluster-template-create`** and **`magnum cluster-create`** commands.

7.5.1 Prerequisites

- Magnum must be installed before proceeding. For more information, see *Book "Installing with Cloud Lifecycle Manager", Chapter 14 "Magnum Overview", Section 14.2 "Install the Magnum Service"*.

Important

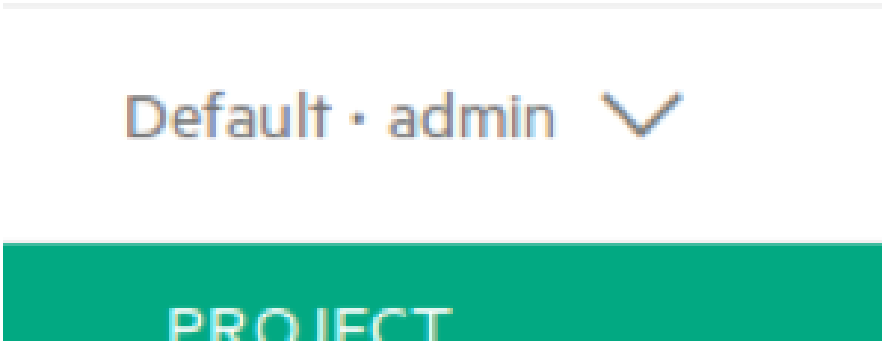
Pay particular attention to `external-name:` in `data/network_groups.yml`. This cannot be set to the default `myardana.test` and must be a valid DNS-resolvable FQDN. If you do not have a DNS-resolvable FQDN, remove or comment out the `external-name` entry and the public endpoint will use an IP address instead of a name.

- The image for which you want to base your cluster on must already have been uploaded into glance. See the previous CLI instructions regarding deploying a cluster on how this is done.

7.5.2 Creating the Cluster Template

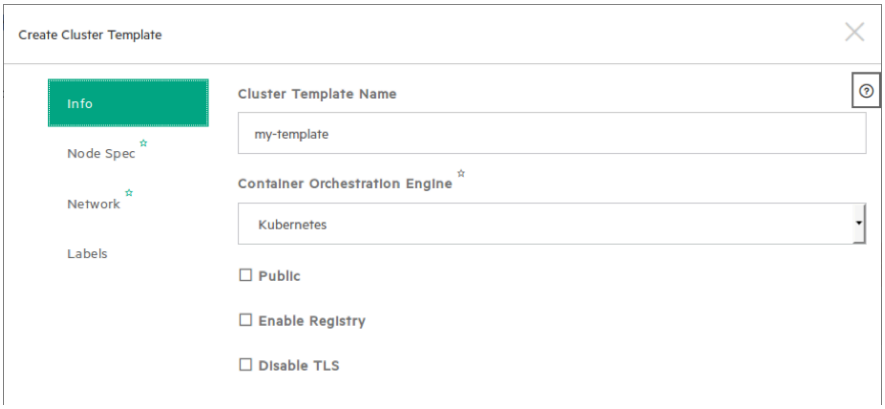
You will need access to the Dashboard to create the cluster template. If you have not accessed the Horizon Dashboard before or you are unfamiliar with it, see *Chapter 2, Using the Dashboard* and *Chapter 3, Cloud Admin Actions with the Dashboard* for more information.

1. Open a web browser that has both JavaScript and cookies enabled. In the address bar, enter the host name or IP address for the dashboard.
2. On the *Log In* page, enter your user name and password and then click *Connect*.
3. Make sure you are in the appropriate domain and project in the left pane. Below is an example image of the drop-down box:

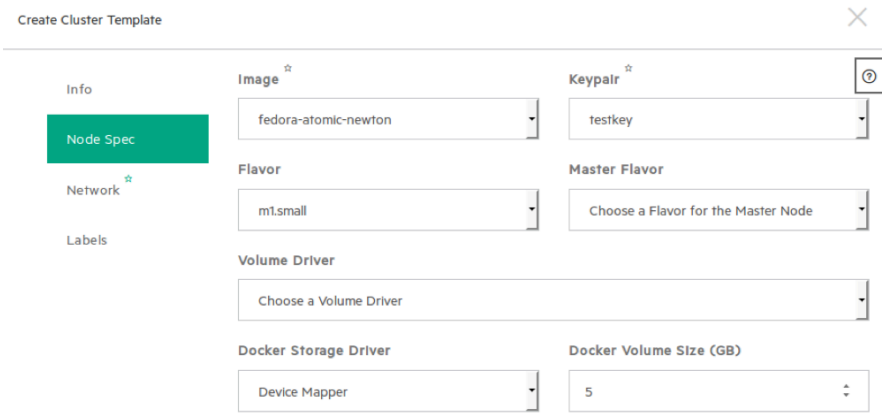


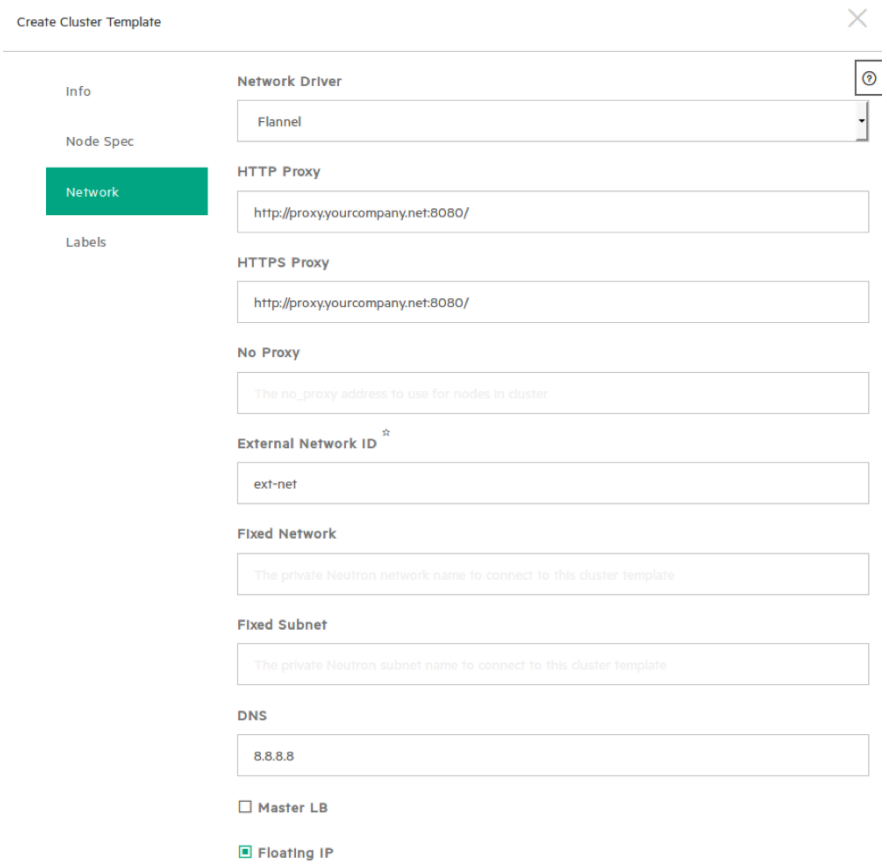
4. A key pair is required for cluster template creation. It is applied to VMs created during the cluster creation process. This allows SSH access to your cluster's VMs. If you would like to create a new key pair, do so by going to *Project* › *Compute* › *Access & Security* › *Key Pairs*.

5. Go to *Project* › *Container Infra* › *Cluster Templates*. Insert `CLUSTER_NAME` and click on + *Create Cluster Template* with the following options:



- *Public* - makes the template available for others to use.
- *Enable Registry* - creates and uses a private docker registry backed by OpenStack Swift in addition to using the public docker registry.
- *Disable TLS* - turns off TLS encryption. For Kubernetes clusters which use client certificate authentication, disabling TLS also involves disabling authentication.



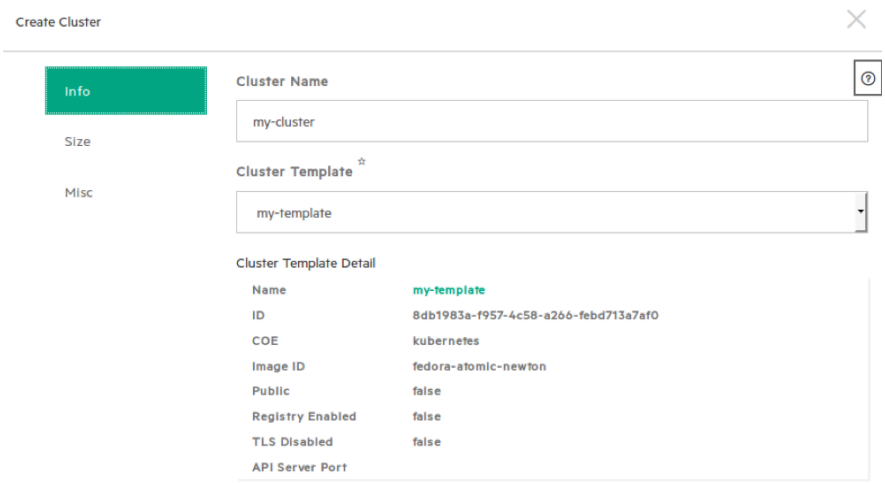


- Proxies are only needed if the created VMs require a proxy to connect externally.
- *Master LB* – This should be turned on if LbaaS v2 (Octavia) is configured in your environment. This option will create load balancers in front of the cluster's master nodes and distribute the requests to the Kubernetes API and etcd across the master nodes.
- *Floating IP* – This assigns floating IPs to the cluster nodes when the cluster is being created. This should be selected if you wish to ssh into the cluster nodes, perform diagnostics and additional tuning to Kubernetes.

6. Click the *Submit* button to create the cluster template and you should see *my-template* in the list of templates.

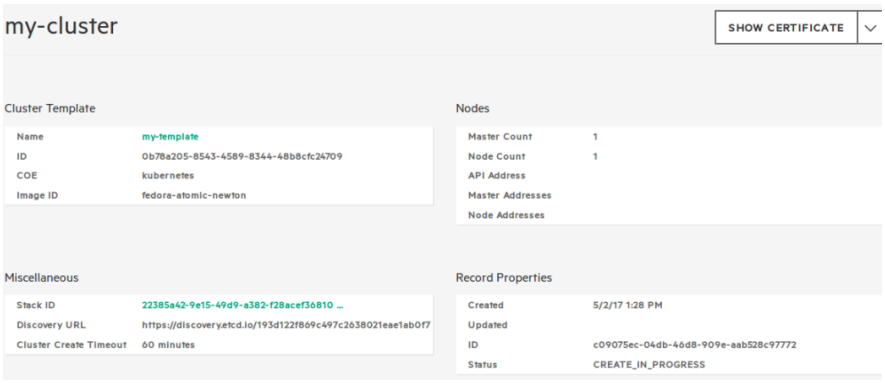
7.5.3 Creating the Cluster

1. Click *Create Cluster* for *my-template* or go to *Project* › *Container Infra* › *Clusters* and click + *Create Cluster* with the following options.



2. Click *Create* to start the cluster creation process.

3. Click *Clusters* in the left pane to see the list of clusters. You will see *my-cluster* in this list. If you select *my-cluster*, you will see additional information regarding your cluster.



8 Creating a Private Network

8.1 Prerequisites

These steps assume the following have been completed:

- Your Administrator has created a valid external network for your environment. See *Book "Installing with Cloud Lifecycle Manager", Chapter 27 "UI Verification", Section 27.4 "Creating an External Network"* for more details.
- Your Administrator has provided you with the IP address or hostname for the Horizon Dashboard. See *Chapter 3, Cloud Admin Actions with the Dashboard* for more details.

You will want to use the user credentials that your Administrator has provided to you when you access the dashboard.

8.2 Creating a Router

1. Access Horizon and log in with your credentials. Contact your Administrator if you do not know your login credentials.
2. Open the Create Router wizard by following these steps:
    a. Navigate to **Routers** under the Network menu.
    b. Press the **Create Router** button.
3. Give your router a name and then click the Create Router button again:
    a. Select a **Router Name**.
    b. Select an **External Network** from the drop down menu.
    c. Press the **Create Router** button.

8.3 Creating a Network and Subnet

1. Access Horizon and log in with your credentials. Contact your Administrator if you do not know your login credentials.
2. Open the Create Network wizard by following these steps:
    a. Navigate to **Networks** under the Network menu.
    b. Press the **Create Network** button.
3. On the first window of the create network wizard:
    a. Choose a **Network Name**.
    b. Select the **Next** button.
4. On the second window of the create network wizard:
    a. Choose a **Subnet Name**.
    b. Enter a **Network Address** (CIDR) for your subnet.
    c. Select the **Next** button.
5. On the final window of the Create Network wizard:
    a. Ensure that **Enable DHCP** is checked.
    b. For a basic network setup, leave the **Allocation Pools**, **DNS Name Servers**, and **Host Routes** fields blank.
    c. Select *Create* when finished.
6. The next step is to attach your subnet to your router.
    a. Navigate to *Routers* under the `Network` menu.
    b. Click on your router name to bring up its settings.
7. Create the interface:
    a. Select the **Interfaces** tab.
    b. Click the **Add Interface** button.

8. Complete the interface creation:

    a. Select your subnet from the drop down menu.

    b. select the **Add Interface** button.

Warning

Do not enter a value for the `IP Address (Optional)` field as this can cause other issues.

9. To confirm your router is setup properly you can:

    a. Click on the **Network Topology** menu option:

    b. Ensure it looks correct.

9 Creating a Key Pair

Key pairs are used to provide SSH access to Nova compute instances. These steps will show you how to create them with either the Horizon dashboard UI or the command-line tools.

For more details about access and security for Nova, see Configure access and security for instances (http://docs.openstack.org/user-guide/cli_nova_configure_access_security_for_instances.html).

9.1 Creating a Key Pair using Horizon

1. Log in to your Horizon dashboard.
2. Under the **Project** menu, navigate to **Access and Security**.
3. Navigate to the **Key Pairs** tab and select the **Create Key Pair** button.
4. Enter in a unique name for your key pair and then select the **Create Key Pair** button.
5. Your key pair will be created and the public key portion of the key (`.pem`) will be automatically downloaded to your local computer. You will then have the option to select this key pair when creating new Nova compute instances.

9.2 Creating a Key Pair using the Command Line

You can utilize either the OpenStack unified command-line tool or the NovaClient command-line tool to create a keypair. These steps assume you have the tool installed and that you have credentials to request an authentication token.

For full details on these command-line clients and installation instructions, see OpenStack Command-Line Interface Reference (http://docs.openstack.org/cli-reference/content/)

Using the OpenStack CLI

The format of the command-line call for this is:

```
openstack keypair create --public-key <file> <name>
```

where:

Value	Description
--public-key <file>	This is an optional field. If used, this will be the path to your public key which will be used when creating the key pair.
<name>	This will be the unique name for your key pair.

Using the Nova CLI

The format of the command-line call for this is:

```
nova keypair-add --public-key <file> <name>
```

where:

Note

You can use `nova help keypair-add` to get the syntax for this command.

Value	Description
--public-key <file>	This is an optional field. If used, this will be the path to your public key which will be used when creating the key pair.
<name>	This will be the unique name for your key pair.

10 Creating and Uploading a Glance Image

This guide will assist you in obtaining, creating, or modifying cloud images for your Image (Glance) repository and uploading them for use.

10.1 How to Curate Your Own Images

OpenStack has created a guide to show you how to obtain, create, and modify images that will be compatible with your SUSE OpenStack Cloud cloud:

OpenStack Virtual Machine Image Guide (http://docs.openstack.org/image-guide/content/)

10.2 Example: Uploading a Cirros Linux Image for Use

These steps assume you have a user account setup within Keystone that has access to upload images to the Glance repository. Contact your Administrator if you do not have these permissions or if you are unsure.

1. Download the Cirros image from the internet:

```
wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
```

2. Upload that file to Glance using the GlanceClient CLI:

```
glance \
  --os-username <username> \
  --os-password <password> \
  --os-tenant-name <project name> \
  --os-auth-url <identity endpoint> \
  --os-endpoint-type internalURL \
  image-create
     --name cirros-0.3.3-x86_64 \
     --container-format bare \
     --disk-format qcow2 \
     --visibility public \
     --file <path to Cirros image file>
```

10.3 Using Horizon to Upload an Image

It is possible to use the Horizon UI to create images for use in your cloud. These steps will show you how.

To successfully create large images, select the image format first, then add image name, image source and visibility.

Performing the steps out of this order will cause OS image creation to fail.

Important

By default, the HTTP upload option will not work when uploading images. To utilize this option you will need your cloud administrator to enable this option. See *Book "Operations Guide", Chapter 15 "Troubleshooting Issues", Section 15.5 "Troubleshooting the Image (Glance) Service"* for more details.

1. Log in to the Horizon UI.
2. In the menu, select *Project* › *Compute* › *Images* and click the *Create Image* button:
3. Fill in the details for your new image and then click the **Create Image** button:

11 Creating a Load Balancer with the Dashboard

In SUSE OpenStack Cloud 8 you can create a Load Balancer with the Load Balancer Panel in the Dashboard.

Follow the steps below to create the load balancer, listener, pool, add members to the pool and create the health monitor.

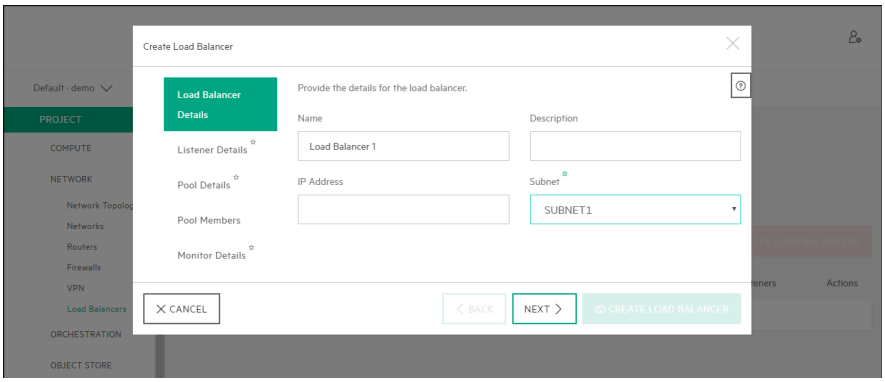
Note

Optionally, you may add members to the load balancer pool after the load balancer has been created.

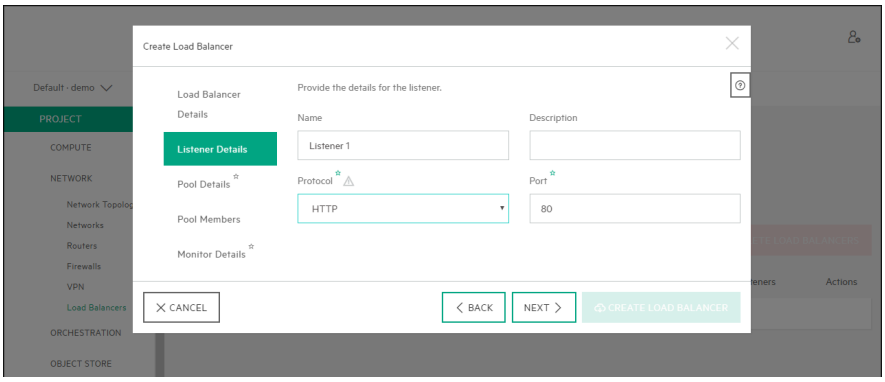
1. **Login to the Dashboard**
   Login into the Dashboard using your domain, user account and password.

2. **Navigate and Create Load Balancer**
   Once logged into the Dashboard, navigate to the *Load Balancers* panel by selecting *Project* › *Network* › *Load Balancers* in the navigation menu, then select *Create Load Balancer* from the Load Balancers page.

3. **Create Load Balancer**
   Provide the Load Balancer details, Load Balancer Name, Description (optional), IP Address and Subnet. When complete, select Next.

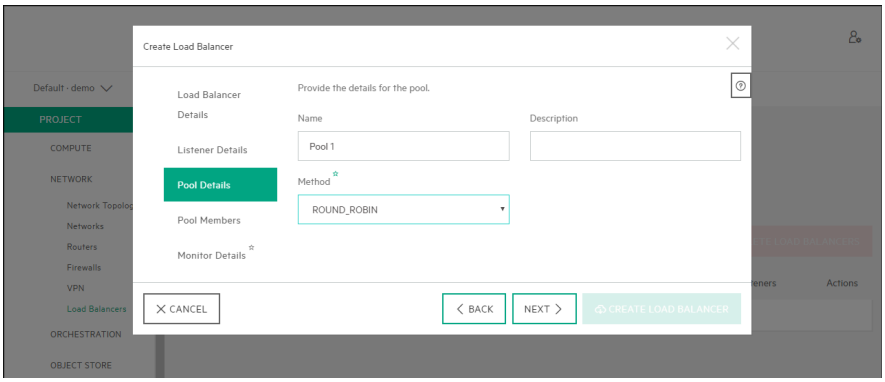


4. **Create Listener**
   Provide a Name, Description, Protocol (HTTP, TCP, TERMINATED_HTTPS) and Port for the Load Balancer Listener.



5. **Create Pool**

   Provide the Name, Description and Method (LEAST_CONNECTIONS, ROUND_ROBIN, SOURCE_IP) for the Load Balancer Pool.

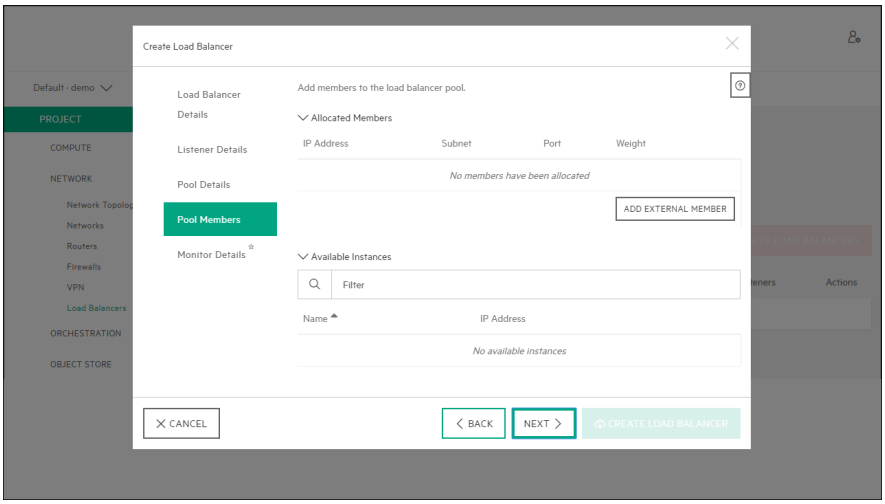


6. **Add Pool Members**

   Add members to the Load Balancer Pool.

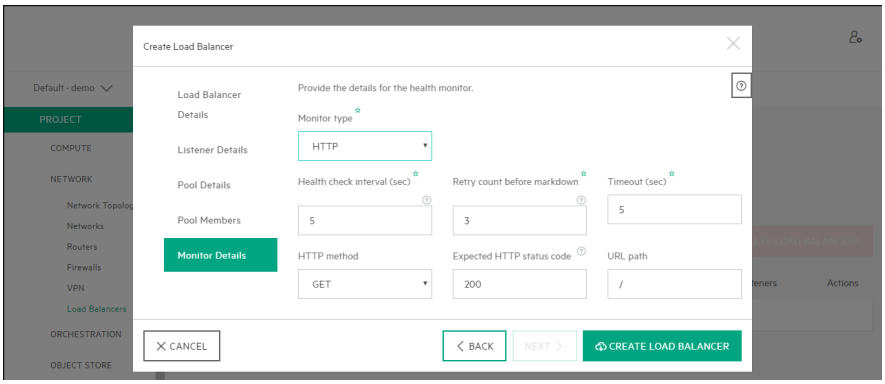
Note

Optionally, you may add members to the load balancer pool after the load balancer has been created.



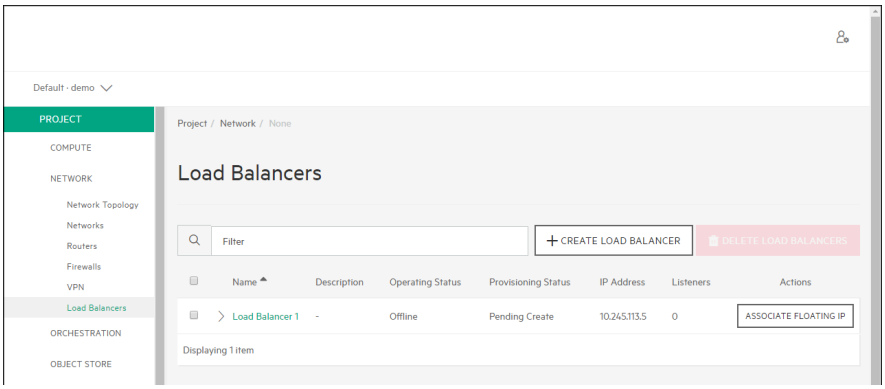
7. **Create Health Monitor**

   Create Health Monitor by providing the Monitor type (HTTP, PING, TCP), the Health check interval, Retry count, timeout, HTTP Method, Expected HTTP status code and the URL path. Once all fields are filled, select **Create Load Balancer**.



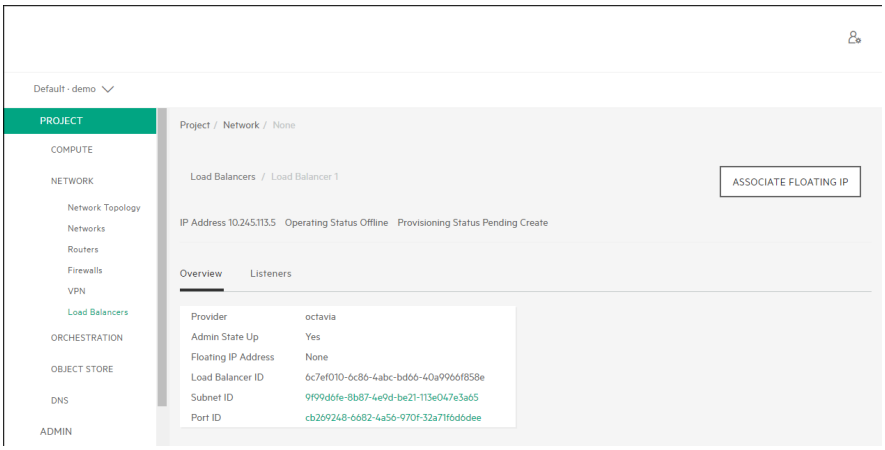
8. **Load Balancer Provisioning Status**

   Clicking on the Load Balancers tab again will provide the status of the Load Balancer. The Load Balancer will be in **Pending Create** until the Load Balancer is created, at which point the Load Balancer will change to an **Active** state.



9. **Load Balancer Overview**

   Once **Load Balancer 1** has been created, it will appear in the Load Balancers list. Click the Load Balancer 1, it will show the Overview. In this view, you can see the Load Balancer Provider type, the Admin State, Floating IP, Load Balancer, Subnet and Port ID's.



12 Using Load Balancing as a Service (LBaaS)

Load Balancing as a Service (LBaaS) is an advanced networking service that allows load balancing of multi-node environments. It provides the ability to spread requests across multiple servers thereby reducing the load on any single server. The following examples depict usage of the various OpenStack command-line interfaces. The Load Balancer v1 API is also accessible via the Horizon web interface if the v1 API is enabled. The Load Balancer v2 API does not currently have a representative Horizon web interface. The v2 API is targeted to have a Horizon web interface in a future SUSE OpenStack Cloud release. This document describes the configuration for LBaaS v1 and v2.

You can create TLS enabled Load Balancers in SUSE OpenStack Cloud 8 by following the steps labeled for TLS Load Balancers. You cannot enable TLS with v1 Load Balancers, only v2 Load Balancers can be enabled with TLS.

Important

When Barbican is not installed by default, you have to manually install Barbican and redeploy neutron.

SUSE OpenStack Cloud 8 can support either LBaaS v1 or LBaaS v2 to allow for wide ranging customer requirements. Check with your administrator for the version that is installed before starting your configuration.

12.1 Configuration

SUSE OpenStack Cloud 8 LBaaS Configuration

Create Private Network for LBaaS

You can create the new network and router by executing the following command from the Cloud Lifecycle Manager or a shell with access to the API nodes.

- As a cloud admin, run the following commands to create a private network and a router.

```
neutron net-create private
neutron subnet-create --name sub private 10.1.0.0/24 --gateway 10.1.0.1
neutron router-create --distributed false router
neutron router-interface-add router sub
neutron router-gateway-set router ext-net
```

Start Virtual Machines

1. Add security group rules.

   Note

   In the example below the load balancer is tested on port 80. If you need to test the load balancer on a different port you will need to create a security group rule for your port number.

```
neutron security-group-rule-create default --protocol icmp
neutron security-group-rule-create default --protocol tcp --port-range-min 22 --
port-range-max 22
neutron security-group-rule-create default --protocol tcp --port-range-min 80 --
port-range-max 80
```

2. Start two VMs on the private network.

```
## Start vm1
nova boot --flavor 1 --image <image> --nic net-id=$(neutron net-list | awk '/
private/ {print $2}') vm1

## start vm2
nova boot --flavor 1 --image <image> --nic net-id=$(neutron net-list | awk '/
private/ {print $2}') vm2
```

3. Check if the VMs are active.

```
nova list
```

For TLS Load Balancers - Create Certificate Chain and Key

1. From a computer with access to the Barbican and labs API, run the following sequence of commands

```
openssl genrsa -des3 -out ca.key 1024
openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
openssl x509  -in  ca.crt -out ca.pem
openssl genrsa -des3 -out ca-int_encrypted.key 1024
openssl rsa -in ca-int_encrypted.key -out ca-int.key
openssl req -new -key ca-int.key -out ca-int.csr -subj "/CN=ca-int@acme.com"
openssl x509 -req -days 3650 -in ca-int.csr -CA ca.crt -CAkey ca.key -set_serial 01
 -out ca-int.crt
```

```
openssl genrsa -des3 -out server_encrypted.key 1024
openssl rsa -in server_encrypted.key -out server.key
openssl req -new -key server.key -out server.csr -subj "/CN=server@acme.com"
openssl x509 -req -days 3650 -in server.csr -CA ca-int.crt -CAkey ca-int.key -
set_serial 01 -out server.crt
```

2. For SNI, create another chain with a different CN

```
openssl genrsa -des3 -out ca2.key 1024
openssl req -new -x509 -days 3650 -key ca2.key -out ca2.crt
openssl x509  -in  ca2.crt -out ca2.pem
openssl genrsa -des3 -out ca-int_encrypted2.key 1024
openssl rsa -in ca-int_encrypted2.key -out ca-int2.key
openssl req -new -key ca-int2.key -out ca-int2.csr -subj "/CN=ca-int-
test2@stacme.com"
openssl x509 -req -days 3650 -in ca-int2.csr -CA ca2.crt -CAkey ca2.key -set_serial
 01 -out ca-int2.crt
openssl genrsa -des3 -out server_encrypted2.key 1024
openssl rsa -in server_encrypted2.key -out server2.key
openssl req -new -key server2.key -out server2.csr -subj "/CN=test2@stacme.com"
openssl x509 -req -days 3650 -in server2.csr -CA ca-int2.crt -CAkey ca-int2.key -
set_serial 01 -out server2.crt
```

For TLS Load Balancers - Barbican Secrets and Containers

1. Source the `barbican.osrc` file from the lifecycle manager node. If you need to perform this operation on a different computer make sure that the OpenStack user account uploading the certs has the keymanager-admin role and is in the admin tenant (see *Book "Operations Guide", Chapter 4 "Managing Identity", Section 4.5 "Configuring the Identity Service"* for a list of roles and tenants). LBaaS will only be able to access certificates stored in the admin tenant.

```
barbican secret store --payload-content-type='text/plain' --name='certificate' --
payload="$(cat server.crt)"
barbican secret store --payload-content-type='text/plain' --name='private_key' --
payload="$(cat server.key)"
barbican secret container create --name='tls_container' --type='certificate' --
secret="certificate=$(barbican secret list | awk '/ certificate / {print $2}')" --
secret="private_key=$(barbican secret list | awk '/ private_key / {print $2}')"
```

Warning

Do not delete the certificate container associated with your Load Balancer listeners before deleting the Load Balancers themselves. If you delete the certificate first, future operations on your Load Balancers and failover will cease to function.

2. Add the second certificate chain to test SNI

```
barbican secret store --payload-content-type='text/plain' --name='certificate2' --
payload="$(cat server2.crt)"
barbican secret store --payload-content-type='text/plain' --name='private_key2' --
payload="$(cat server2.key)"
barbican secret container create --name='tls_container2' --type='certificate' --
secret="certificate=$(barbican secret list | awk '/ certificate2 / {print $2}')" --
secret="private_key=$(barbican secret list | awk '/ private_key2 / {print $2}')"
```

3. Find the Octavia user id. You will add the id as a barbican acl user for the containers and keys.

```
source keystone.osrc
openstack user list
```

4. Get the container and secret hrefs.

```
source barbican.osrc
barbican secret list
barbican secret container list
```

5. Add the acl user obtained in step 3 to the hrefs obtained in step 4 by executing `barbican acl user add --user <user id of Octavia> <href>`. In the example below, the Octavia user, `66649a0863b64275bc3bffb50e3d76c8` is being added as the Barbican acl user for the containers and keys:

```
barbican acl user add --user 66649a0863b64275bc3bffb50e3d76c8
 https://10.242.124.130:9311/v1/containers/7ebcd4fa-e96a-493d-b1ee-260914d3cbeb
barbican acl user add --user 66649a0863b64275bc3bffb50e3d76c8
 https://10.242.124.130:9311/v1/secrets/d3c9584c-a43c-4fc1-bfa9-ebcafee57059
barbican acl user add --user 66649a0863b64275bc3bffb50e3d76c8
 https://10.242.124.130:9311/v1/secrets/0b958aa8-49d2-40aa-82dd-5660e012b3a3
```

Create Load Balancer v2

Note

The creation of the Load Balancer requires a tenant network and not an external network.

1. Create the new load balancer using the `lbaas-loadbalancer-create` command and giving the load balancer a name and subnet.

```
source barbican.osrc
neutron lbaas-loadbalancer-create --name lb sub
```

2. Create a new listener. If you are enabling TLS, use the second example, if you are enabling TLS and SNI, use the third example.

Note

Use unique port numbers for each listener. This example uses 80, 443 and 444.

a. Create a new listener for the load balancer without TLS using the `lbaas-listener-create` command and giving the listener the name of the load balancer, the protocol, the protocol port and a name for the listener.

```
neutron lbaas-listener-create --loadbalancer lb --protocol HTTP --protocol-
port 80 --name listener
```

b. Create a new listener for the load balancer with TLS and no SNI using the `lbaas-listener-create` command and giving the listener the name of the load balancer, the protocol, the protocol port, the name for the listener and the default TLS container.

```
neutron lbaas-listener-create --loadbalancer lb --protocol-port 443 --protocol
 TERMINATED_HTTPS --name tls_listener --default-tls-container-ref=$(barbican
 secret container list | awk '/ tls_container / {print $2}')
```

c. Create a new listener for the load balancer with TLS and SNI using the `lbaas-listener-create` command and giving the listener the name of the load balancer, the protocol, the protocol port, the name for the listener, the default TLS container and the SNI container.

```
neutron lbaas-listener-create --loadbalancer lb --protocol-port 444 --protocol
 TERMINATED_HTTPS --name sni_listener --default-tls-container-ref=$(barbican
```

```
 secret container list | awk '/ tls_container / {print $2}') --sni-container-
refs $(barbican secret container list | awk '/ tls_container2 / {print $2}'
```

d. For each listener, create a new pool for the load balancer using the `lbaas-pool-create` command. Creating a new pool requires the load balancing algorithm, the name of the listener, the protocol and a name for the pool. In the example below we show the command for the listener named `listener`. You need to repeat that for the tls_listener and sni_listener as well. Make sure to specify different names for each pool.

```
neutron lbaas-pool-create --lb-algorithm ROUND_ROBIN --listener listener --
protocol HTTP --name <pool name>
```

e. You can add members to the load balancer pool by running the `lbaas-member-create` command. The command requires the subnet, IP address, protocol port and the name of the pool for each virtual machine you would like to include into the load balancer pool. It is important to note that this will need to be repeated for each pool created above.

```
neutron lbaas-member-create --subnet sub --address <ip address vm1> --
protocol-port <port> <pool name>
neutron lbaas-member-create --subnet sub --address <ip address vm2> --
protocol-port <port> <pool name>
```

f. Display the current state of the load balancer and values with `lbaas-loadbalancer-show`.

```
neutron lbaas-loadbalancer-show lb
```

g. You need to assign the floating IP to lbaas VIP so it can be accessed from the external network.

```
fixedip_vip=$(neutron lbaas-loadbalancer-list | awk '/lb/ {print $6}')
portuuid_vip=$(neutron port-list | grep $fixedip_vip | awk '{print $2}')
```

h. Create and associate the floating IP address to lbaas VIP address.

```
neutron floatingip-create ext-net --port-id $portuuid_vip
```

- A complete list of the Load Balancer v2 API commands can be found at: https://wiki.openstack.org/wiki/Neutron/LBaaS/API_2.0
- Additional Load Balancer v2 API examples can be found at: http://docs.openstack.org/mitaka/networking-guide/adv-config-lbaas.html
- Instructions on how to terminate TLS certificates on a deployed Load Balancer can be found at: https://wiki.openstack.org/wiki/Neutron/LBaaS/API_2.0#Create_a_Listener

Create Load Balancer v1

Note

v1 Load Balancers cannot be enabled with TLS.

1. Create the load balancer pool with `lb-pool-create` giving it a method, name, protocol and subnet.

```
neutron lb-pool-create --lb-method ROUND_ROBIN --name pool --protocol HTTP --subnet-
id $(neutron subnet-list | awk '/sub/ {print $2}')
```

2. Create load balancing members with `lb-member-create` providing the IP address, protocol and load balancing pool name to each member.

```
neutron lb-member-create --address <ip address vm1> --protocol-port <port> pool
neutron lb-member-create --address <ip address vm2> --protocol-port <port> pool
```

3. Create the vip with `lb-vip-create` giving it a name, protocol, protocol port and a subnet.

```
neutron lb-vip-create --name vip --protocol-port <port> --protocol HTTP --subnet-id
 $(neutron subnet-list | awk '/sub/ {print $2}') pool
```

4. You can check to see if the load balancer is active with `lb-vip-show`

```
neutron lb-vip-show vip
```

Validate LBaaS Functionality

Note

You should perform the following steps from a node that has a route to the private network. Using the examples from above, 10.1.0.0/24 should be reachable.

1. SSH into both vm1 and vm2 in two separate windows and make them listen on your configured port.

2. From one window.

```
ssh cirros@<ip address vm1>
pass: <password>
```

3. From another window.

```
ssh cirros@<ip address vm2>
pass: <password>
```

4. Start running web servers on both of the virtual machines.

5. Create a webserv.sh script with below contents. Use the <port> from the member creation step.

```
$ vi webserv.sh

#!/bin/bash

MYIP=$(/sbin/ifconfig eth0|grep 'inet addr'|awk -F: '{print $2}'| awk '{print $1}');
while true; do
    echo -e "HTTP/1.0 200 OK

Welcome to $MYIP" | sudo nc -l -p <port>
done

## Give it Exec rights
$ chmod 755 webserv.sh

## Start webserver
$ ./webserv.sh
```

6. Open a separate window. From the respective source node in external network (in case of accessing LBaaS VIP thorough its FIP) or in private network (in case of no FIP), add the respective IP address to the no_proxy env variable, if required. You can get the *VIP* from the `neutron lbaas-loadbalancer-list` for LBaaS v2 and `neutron lb-vip-list` for LBaaS v1.

7. Run the following commands to test load balancing. In this example, the VIP IP address is 10.1.0.7 and when executing curl against the VIP, the responses are returned from the load balanced services.

```
$ export no_proxy=$no_proxy,10.1.0.7

## Curl the VIP
$ curl 10.1.0.7
Welcome to 10.1.0.4

$ curl 10.1.0.7
Welcome to 10.1.0.5

$ curl 10.1.0.7
Welcome to 10.1.0.4
```

Verify SNI

You can verify SNI by running the following command from a node with access to the private network. You can get the VIP from the neutron `lbaas-loadbalancer-list` for LBaaS v2.

```
openssl s_client -servername test2@stacme.com -connect <vip of lb>:444
```

Certificate information will print to the screen. You should verify that the CN matches the CN you passed to `-servername`. In the example below the CN matches the servername passed from above.

```
subject=/CN=test2@stacme.com
issuer=/CN=ca-int-test2@stacme.com
```

12.2 For More Information

For more information on the neutron command-line interface (CLI) and load balancing, see the OpenStack networking command-line client reference: http://docs.openstack.org/cli-reference/content/neutronclient_commands.html

13 Using Load Balancing as a Service with Orchestration Service

13.1 Orchestration Service

HPE Orchestration service, based on OpenStack Heat, enables the design and coordination of multiple composite cloud applications using templates.

You can use the Orchestration Service templates to describe and execute OpenStack API calls to create running cloud applications. The templates can be used to describe relationships and resources to help manage your cloud infrastructure and can also help you with the creation of resource types.

13.2 Orchestration Service support for LBaaS v2

In SUSE OpenStack Cloud, the Orchestration Service provides support for LBaaS v2, which means users can create LBaaS v2 resources using Orchestration.

The OpenStack documentation for LBaaSv2 resource plugins is available at following locations.

- Neutron LBaaS v2 LoadBalancer: http://docs.openstack.org/developer/heat/template_guide/openstack.html#OS::Neutron::LBaaS::LoadBalancer
- Neutron LBaaS v2 Listener: http://docs.openstack.org/developer/heat/template_guide/openstack.html#OS::Neutron::LBaaS::Listener
- Neutron LBaaS v2 Pool: http://docs.openstack.org/developer/heat/template_guide/openstack.html#OS::Neutron::LBaaS::Pool
- Neutron LBaaS v2 Pool Member: http://docs.openstack.org/developer/heat/template_guide/openstack.html#OS::Neutron::LBaaS::PoolMember
- Neutron LBaaS v2 Health Monitor: http://docs.openstack.org/developer/heat/template_guide/openstack.html#OS::Neutron::LBaaS::HealthMonitor

13.3 Limitations

In order to avoid stack-create timeouts when using load balancers, it is recommended that no more than 100 load balancers be created at a time using stack-create loops. Larger numbers of load balancers could reach quotas and/or exhaust resources resulting in the stack create-timeout.

13.4 More Information

For more information on configuring and using the SUSE OpenStack Cloud Load Balancing as a Service, see: *Book "Operations Guide", Chapter 9 "Managing Networking", Section 9.3 "Networking Service Overview", Section 9.3.8 "Configuring Load Balancing as a Service (LBaaS)"* and *Chapter 12, Using Load Balancing as a Service (LBaaS)*.

For more information on the neutron command-line interface (CLI) and load balancing, see the OpenStack networking command-line client reference: http://docs.openstack.org/cli-reference/content/neutronclient_commands.html

For more information on Heat see: http://docs.openstack.org/developer/heat

14 Using Firewall as a Service (FWaaS)

The Firewall as a Service (FWaaS) provides the ability to assign network-level, port security for all traffic entering and existing a tenant network. More information on this service can be found via the public OpenStack documentation located at http://specs.openstack.org/openstack/neutron-specs/specs/api/firewall_as_a_service__fwaas_.html. The following documentation provides command-line interface example instructions for configuring and testing a firewall. The Firewall as a Service can also be configured and managed by the Horizon web interface.

FWaaS is implemented directly in the L3 agent (*neutron-l3-agent*), however if VPNaaS is enabled, FWaaS is implemented in the VPNaaS agent (*neutron-vpn-agent*). Because FWaaS does not use a separate agent process or start a specific service, there currently are no Monasca alarms for it.

If DVR is enabled, the firewall service currently does not filter traffic between OpenStack private networks, also known as *east-west traffic* and will only filter traffic from external networks, also known as *north-south traffic*.

14.1 Prerequisites

SUSE OpenStack Cloud must be installed.

14.2 SUSE OpenStack Cloud 8 FWaaS Configuration

Check for an enabled firewall.

1. You should check to determine if the firewall is enabled. The output of the *neutron ext-list* should contain a firewall entry.

   ```
   neutron ext-list
   ```

2. Assuming the external network is already created by the admin, this command will show the external network.

   ```
   neutron net-list
   ```

Create required assets.

Before creating firewalls, you will need to create a network, subnet, router, security group rules, start an instance and assign it a floating IP address.

1. Create the network, subnet and router.

```
neutron net-create private
neutron subnet-create --name sub private 10.0.0.0/24 --gateway 10.0.0.1
neutron router-create router
neutron router-interface-add router sub
neutron router-gateway-set router ext-net
```

2. Create security group rules. Security group rules filter traffic at VM level.

```
neutron security-group-rule-create default --protocol icmp
neutron security-group-rule-create default --protocol tcp --port-range-min 22 --
port-range-max 22
neutron security-group-rule-create default --protocol tcp --port-range-min 80 --
port-range-max 80
```

3. Boot a VM.

```
NET=$(neutron net-list | awk '/private/ {print $2}')
nova boot --flavor 1 --image <image> --nic net-id=$NET vm1 --poll
```

4. Verify if the instance is ACTIVE and is assigned an IP address.

```
nova list
```

5. Get the port id of the vm1 instance.

```
fixedip=$(nova list | awk '/vm1/ {print $12}' | awk -F '=' '{print $2}' | awk -F ','
 '{print $1}')
vmportuuid=$(neutron port-list | grep $fixedip | awk '{print $2}')
```

6. Create and associate a floating IP address to the vm1 instance.

```
neutron floatingip-create ext-net --port-id $vmportuuid
```

7. Verify if the floating IP is assigned to the instance. The following command should show an assigned floating IP address from the external network range.

```
nova show vm1
```

8. Verify if the instance is reachable from the external network. SSH into the instance from a node in (or has route to) the external network.

```
ssh cirros@FIP-VM1
password: <password>
```

Create and attach the firewall.

Note

By default, an internal "drop all" rule is enabled in IP tables if none of the defined rules match the real-time data packets.

1. Create new firewall rules using `firewall-rule-create` command and providing the protocol, action (allow, deny, reject) and name for the new rule.
   Firewall actions provide rules in which data traffic can be handled. An **allow** rule will allow traffic to pass through the firewall, **deny** will stop and prevent data traffic from passing through the firewall and **reject** will reject the data traffic and return a *destination-unreachable* response. Using **reject** will speed up failure detection time dramatically for legitimate users, since they will not be required to wait for retransmission timeouts or submit retries. Some customers should stick with **deny** where prevention of port scanners and similar methods may be attempted by hostile attackers. Using **deny** will drop all of the packets, making it more difficult for malicious intent. The firewall action, **deny** is the default behavior.
   The example below demonstrates how to allow icmp and ssh while denying access to http. See the `neutron` command-line reference at http://docs.openstack.org/cli-reference/content/neutronclient_commands.html on additional options such as source IP, destination IP, source port and destination port.

   #### Note

   You can create a firewall rule with an identical name and each instance will have a unique id associated with the created rule, however for clarity purposes this is not recommended.

```
neutron firewall-rule-create --protocol icmp --action allow --name allow-icmp
neutron firewall-rule-create --protocol tcp --destination-port 80 --action deny --
name deny-http
```

```
neutron firewall-rule-create --protocol tcp --destination-port 22 --action allow --
name allow-ssh
```

2. Once the rules are created, create the firewall policy by using the `firewall-policy-create` command with the `--firewall-rules` option and rules to include in quotes, followed by the name of the new policy. The order of the rules is important.

```
neutron firewall-policy-create --firewall-rules "allow-icmp deny-http allow-ssh"
 policy-fw
```

3. Finish the firewall creation by using the `firewall-create` command, the policy name and the new name you want to give to your new firewall.

```
neutron firewall-create policy-fw --name user-fw
```

4. You can view the details of your new firewall by using the `firewall-show` command and the name of your firewall. This will verify that the status of the firewall is ACTIVE.

```
neutron firewall-show user-fw
```

Verify the FWaaS is functional.

1. Since allow-icmp firewall rule is set you can ping the floating IP address of the instance from the external network.

```
ping <FIP-VM1>
```

2. Similarly, you can connect via ssh to the instance due to the allow-ssh firewall rule.

```
ssh cirros@<FIP-VM1>
password: <password>
```

3. Run a web server on vm1 instance that listens over port 80, accepts requests and sends a WELCOME response.

```
$ vi webserv.sh

#!/bin/bash

MYIP=$(/sbin/ifconfig eth0|grep 'inet addr'|awk -F: '{print $2}'| awk '{print $1}');
while true; do
  echo -e "HTTP/1.0 200 OK

Welcome to $MYIP" | sudo nc -l -p 80
```

```
done

# Give it Exec rights
$ chmod 755 webserv.sh

# Execute the script
$ ./webserv.sh
```

4. You should expect to see curl fail over port 80 because of the deny-http firewall rule. If curl succeeds, the firewall is not blocking incoming http requests.

```
curl -vvv <FIP-VM1>
```

Warning

When using reference implementation, new networks, FIPs and routers created after the Firewall creation will not be automatically updated with firewall rules. Thus, execute the firewall-update command by passing the current and new router Ids such that the rules are reconfigured across all the routers (both current and new).

For example if router-1 is created before and router-2 is created after the firewall creation

```
$ neutron firewall-update –router <router-1-id> –router <router-2-id> <firewall-
name>
```

14.3 More Information

Firewalls are based in IPtable settings.

Each firewall that is created is known as an instance.

A firewall instance can be deployed on selected project routers. If no specific project router is selected, a firewall instance is automatically applied to all project routers.

Only 1 firewall instance can be applied to a project router.

Only 1 firewall policy can be applied to a firewall instance.

Multiple firewall rules can be added and applied to a firewall policy.

Firewall rules can be shared across different projects via the Share API flag.

Firewall rules supersede the Security Group rules that are applied at the Instance level for all traffic entering or leaving a private, project network.

For more information on the neutron command-line interface (CLI) and firewalls, see the OpenStack networking command-line client reference: http://docs.openstack.org/cli-reference/content/neutronclient_commands.html

15 Using VPN as a Service (VPNaaS)

SUSE OpenStack Cloud 8 VPNaaS Configuration

This document describes the configuration process and requirements for the SUSE OpenStack Cloud 8 Virtual Private Network (VPN) as a Service module.

15.1 Prerequisites

1. SUSE OpenStack Cloud must be installed.
2. Before setting up VPNaaS, you will need to have created an external network and a subnet with access to the internet. Information on how to create the external network and subnet can be found in *Section 15.4, "More Information"*.
3. You should assume 172.16.0.0/16 as the ext-net CIDR in this document.

15.2 Considerations

Using the Neutron plugin-based VPNaaS causes additional processes to be run on the Network Service Nodes. One of these processes, the ipsec charon process from StrongSwan, runs as root and listens on an external network. A vulnerability in that process can lead to remote root compromise of the Network Service Nodes. If this is a concern customers should consider using a VPN solution other than the Neutron plugin-based VPNaaS and/or deploying additional protection mechanisms.

15.3 Configuration

Setup Networks You can setup VPN as a Service (VPNaaS) by first creating networks, subnets and routers using the `neutron` command line. The VPNaaS module enables the ability to extend access between private networks across two different SUSE OpenStack Cloud clouds or between a SUSE OpenStack Cloud cloud and a non-cloud network. VPNaaS is based on the open source software application called StrongSwan. StrongSwan (more information available at http://www.strongswan.org/) is an IPsec implementation and provides basic VPN gateway functionality.

Note

You can execute the included commands from any shell with access to the service APIs. In the included examples, the commands are executed from the lifecycle manager, however you could execute the commands from the controller node or any other shell with aforementioned service API access.

Note

The use of floating IP's is not possible with the current version of VPNaaS when DVR is enabled. Ensure that no floating IP is associated to instances that will be using VPNaaS when using a DVR router. Floating IP associated to instances are ok when using CVR router.

1. From the Cloud Lifecycle Manager, create first private network, subnet and router assuming that *ext-net* is created by admin.

```
neutron net-create privateA
neutron subnet-create --name subA privateA 10.1.0.0/24 --gateway 10.1.0.1
neutron router-create router1
neutron router-interface-add router1 subA
neutron router-gateway-set router1 ext-net
```

2. Create second private network, subnet and router.

```
neutron net-create privateB
neutron subnet-create --name subB privateB 10.2.0.0/24 --gateway 10.2.0.1
neutron router-create router2
neutron router-interface-add router2 subB
neutron router-gateway-set router2 ext-net
```

PROCEDURE 15.1: STARTING VIRTUAL MACHINES

1. From the Cloud Lifecycle Manager run the following to start the virtual machines. Begin with adding secgroup rules for SSH and ICMP.

```
neutron security-group-rule-create default --protocol icmp
neutron security-group-rule-create default --protocol tcp --port-range-min 22 --
port-range-max 22
```

2. Start the virtual machine in the privateA subnet. Using *nova images-list*, use the image id to boot image instead of the image name. After executing this step, it is recommended that you wait approximately 10 seconds to allow the virtual machine to become active.

```
NETA=$(neutron net-list | awk '/privateA/ {print $2}')
nova boot --flavor 1 --image <id> --nic net-id=$NETA vm1
```

3. Start the virtual machine in the privateB subnet.

```
NETB=$(neutron net-list | awk '/privateB/ {print $2}')
nova boot --flavor 1 --image <id> --nic net-id=$NETB vm2
```

4. Verify private IP's are allocated to the respective vms. Take note of IP's for later use.

```
nova show vm1
nova show vm2
```

PROCEDURE 15.2: CREATE VPN

1. You can set up the VPN by executing the below commands from the lifecycle manager or any shell with access to the service APIs. Begin with creating the policies with `vpn-ikepolicy-create` and `vpn-ipsecpolicy-create` .

```
neutron vpn-ikepolicy-create ikepolicy
neutron vpn-ipsecpolicy-create ipsecpolicy
```

2. Create the VPN service at router1.

```
neutron vpn-service-create --name myvpnA --description "My vpn service" router1 subA
```

3. Wait at least 5 seconds and then run `ipsec-site-connection-create` to create a ipsec-site connection. Note that `--peer-address` is the assign ext-net IP from router2 and `--peer-cidr` is subB cidr.

```
neutron ipsec-site-connection-create --name vpnconnection1 --vpnservice-id myvpnA \
--ikepolicy-id ikepolicy --ipsecpolicy-id ipsecpolicy --peer-address 172.16.0.3 \
--peer-id 172.16.0.3 --peer-cidr 10.2.0.0/24 --psk secret
```

4. Create the VPN service at router2.

```
neutron vpn-service-create --name myvpnB --description "My vpn serviceB" router2
 subB
```

5. Wait at least 5 seconds and then run `ipsec-site-connection-create` to create a ipsec-site connection. Note that `--peer-address` is the assigned ext-net IP from router1 and `--peer-cidr` is subA cidr.

```
neutron ipsec-site-connection-create --name vpnconnection2 --vpnservice-id myvpnB \
--ikepolicy-id ikepolicy --ipsecpolicy-id ipsecpolicy --peer-address 172.16.0.2 \
--peer-id 172.16.0.2 --peer-cidr 10.1.0.0/24 --psk secret
```

6. On the Cloud Lifecycle Manager, run the `ipsec-site-connection-list` command to see the active connections. Be sure to check that the vpn_services are ACTIVE. You can check this by running `vpn-service-list` and then checking ipsec-site-connections status. You should expect that the time for both vpn-services and ipsec-site-connections to become ACTIVE could take as long as 1 to 3 minutes.

```
neutron ipsec-site-connection-list
+--------------------------------------+---------------+-------------
+---------------+------------+-----------+--------+
| id                                   | name          | peer_address | peer_cidrs
   | route_mode | auth_mode | status |
+--------------------------------------+---------------+-------------
+---------------+------------+-----------+--------+
| 1e8763e3-fc6a-444c-a00e-426a4e5b737c | vpnconnection2 | 172.16.0.2   |
 "10.1.0.0/24" | static     | psk       | ACTIVE |
| 4a97118e-6d1d-4d8c-b449-b63b41e1eb23 | vpnconnection1 | 172.16.0.3   |
 "10.2.0.0/24" | static     | psk       | ACTIVE |
+--------------------------------------+---------------+-------------
+---------------+------------+-----------+--------+
```

Verify VPN In the case of non-admin users, you can verify the VPN connection by pinging the virtual machines.

1. Check the VPN connections.

Note

vm1-ip and vm2-ip denotes private IP's for vm1 and vm2 respectively. The private IPs are obtained, as described in of *Step 4*. If you are unable to SSH to the private network due to a lack of direct access, the VM console can be accessed through Horizon.

```
ssh cirros@vm1-ip
password: <password>
```

```
# ping the private IP address of vm2
ping ###.###.###.###
```

2. In another terminal.

```
ssh cirros@vm2-ip
password: <password>

# ping the private IP address of vm1
ping ###.###.###.###
```

3. You should see ping responses from both virtual machines.

As the admin user, you should check to make sure that a route exists between the router gateways. Once the gateways have been checked, packet encryption can be verified by using traffic analyzer (tcpdump) by tapping on the respective namespace (qrouter-* in case of non-DVR and snat-* in case of DVR) and tapping the right interface (qg-***).

Note

When using DVR namespaces, all the occurrences of qrouter-xxxxxx in the following commands should be replaced with respective snat-xxxxxx.

1. Check the if the route exists between two router gateways. You can get the right qrouter namespace id by executing *sudo ip netns*. Once you have the qrouter namespace id, you can get the interface by executing *sudo ip netns qrouter-xxxxxxxx ip addr* and from the result the interface can be found.

```
sudo ip netns
sudo ip netns exec qrouter-<router1 UUID> ping <router2 gateway>
sudo ip netns exec qrouter-<router2 UUID> ping <router1 gateway>
```

2. Initiate a tcpdump on the interface.

```
sudo ip netns exec qrouter-xxxxxxxx tcpdump -i qg-xxxxxx
```

3. Check the VPN connection.

```
ssh cirros@vm1-ip
password: <password>

# ping the private IP address of vm2
```

```
ping ###.###.###.###
```

4. Repeat for other namespace and right tap interface.

```
sudo ip netns exec qrouter-xxxxxxxx tcpdump -i qg-xxxxxx
```

5. In another terminal.

```
ssh cirros@vm2-ip
password: <password>

# ping the private IP address of vm1
ping ###.###.###.###
```

6. You will find encrypted packets containing 'ESP' in the tcpdump trace.

15.4 More Information

VPNaaS currently only supports Pre-shared Keys (PSK) security between VPN gateways. A different VPN gateway solution should be considered if stronger, certificate-based security is required.

For more information on the neutron command-line interface (CLI) and VPN as a Service (VPNaaS), see the OpenStack networking command-line client reference: http://docs.openstack.org/cli-reference/content/neutronclient_commands.html

For information on how to create an external network and subnet, see the OpenStack manual: http://docs.openstack.org/user-guide/dashboard_create_networks.html