Installing with Cloud Lifecycle Manager

SUSE OpenStack Cloud 8

Installing with Cloud Lifecycle Manager

SUSE OpenStack Cloud 8

Publication Date: 07/06/2018

SUSE LLC
10 Canal Park Drive
Suite 200
Cambridge MA 02141
USA
https://www.suse.com/documentation

Contents

Installation Overview

Before jumping into your installation, we recommend taking the time to read through our documentation to get an overview of the sample configurations SUSE OpenStack Cloud 8 offers. We have highly tuned example configurations for each of these Cloud models:

Name	Location
Book "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.3 "KVM Examples", Section 10.3.1 "Entry-Scale Cloud"*	`~/openstack/examples/entry-scale-kvm`
Book "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.3 "KVM Examples", Section 10.3.2 "Entry Scale Cloud with Metering and Monitoring Services"*	`~/openstack/examples/entry-scale-kvm-mml`
Book "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.4 "ESX Examples", Section 10.4.1 "Single-Region Entry-Scale Cloud with a Mix of KVM and ESX Hypervisors"*	`~/openstack/examples/entry-scale-esx-kvm`
Book "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.4 "ESX Examples", Section 10.4.2 "Single-Region Entry-Scale Cloud with Metering and Monitoring Services, and a Mix of KVM and ESX Hypervisors"*	`~/openstack/examples/entry-scale-esx-kvm-mml`
Book "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.5 "Swift Examples", Section 10.5.1 "Entry-scale Swift Model"*	`~/openstack/examples/entry-scale-swift`
Book "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Config-*	`~/openstack/examples/entry-scale-ironic-flat-network`

Name	Location
urations", Section 10.6 "Ironic Examples", Section 10.6.1 "Entry-Scale Cloud with Ironic Flat Network"	
Book "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.6 "Ironic Examples", Section 10.6.2 "Entry-Scale Cloud with Ironic Multi-Tenancy"*	`~/openstack/examples/entry-scale-ironic-multi-tenancy`
Book "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.3 "KVM Examples", Section 10.3.3 "Single-Region Mid-Size Model"*	`~/openstack/examples/mid-scale-kvm`

- *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.3 "KVM Examples", Section 10.3.1 "Entry-Scale Cloud"*
- *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.4 "ESX Examples"*
- *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.5 "Swift Examples", Section 10.5.1 "Entry-scale Swift Model"*
- *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.3 "KVM Examples", Section 10.3.3 "Single-Region Mid-Size Model"*

Using the Command-line

You should use the command-line if:

- You are installing a more complex or large-scale cloud.
- You need to use availability zones or the server groups functionality of the cloud model. Also see *Section 1, "For More Information"*.
- You want to customize the cloud configuration beyond the tuned defaults that SUSE provides out of the box.
- You need deeper customizations than are possible to express using the GUI.

Instructions for installing via the command-line are here:

- *Chapter 13, Installing Mid-scale and Entry-scale KVM*

1 For More Information

Other useful documents that will help you understand, plan, configure, and install your cloud are listed below.

- *Chapter 12, Using Git for Configuration Management*
- *Chapter 19, Troubleshooting the Installation*
- *Chapter 22, Cloud Verification*
- *Chapter 28, Other Common Post-Installation Tasks*

2 Note on Cloud Lifecycle Manager and Ardana

Ardana is an open-source project and a generalized lifecycle framework. Cloud Lifecycle Manager is based on Ardana, but delivers the lifecycle management functionality required by SUSE OpenStack Cloud 8. Due to this relationship, some Cloud Lifecycle Manager commands contain references to Ardana.

I Pre-Installation

1 Overview

To ensure that your environment meets the requirements of the cloud model you choose, see the check list in *Chapter 2, Pre-Installation Checklist*.

When you have decided on a configuration to choose for your cloud and you have gone through the pre-installation steps, you have two options for installation:

- You can use a GUI that runs in your Web browser.
- You can install via the command-line that exposes the full power and flexibility of SUSE OpenStack Cloud 8.

Using the GUI

You should use the GUI if:

- You are not planning to deploy availability zones or use L3 segmentation in your initial deployment.
- You are satisfied with the tuned SUSE-default OpenStack configuration.

Instructions for installing via the GUI are here.

- *Chapter 9, Installing with the Install UI*

Note that reconfiguring your cloud, at the moment, can only be done via the command-line. The GUI installer is for initial installation only.

2 Pre-Installation Checklist

Important

The formatting of this page facilitates printing it out and using it to record details of your setup.

This checklist is focused on the Entry-scale KVM model but you can alter it to fit the example configuration you choose for your cloud.

2.1 BIOS and IPMI Settings

Ensure that the following BIOS and IPMI settings are applied to each bare-metal server:

#	Item
	Choose either UEFI or Legacy BIOS in the BIOS settings
	Verify the Date and Time settings in the BIOS. **Note** SUSE OpenStack Cloud installs and runs with UTC, not local time.
	Ensure that Wake-on-LAN is disabled in the BIOS
	Ensure that the NIC port to be used for PXE installation has PXE enabled in the BIOS
	Ensure that all other NIC ports have PXE disabled in the BIOS
	Ensure all hardware in the server not directly used by SUSE OpenStack Cloud is disabled

2.2 Network Setup and Configuration

Before installing SUSE OpenStack Cloud, the following networks must be provisioned and tested. The networks are not installed or managed by the Cloud. You must install and manage the networks as documented in *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”*.

Note that if you want a pluggable IPAM driver, it must be specified at install time. Only with a clean install of SUSE OpenStack Cloud 8 can you specify a different IPAM driver. If upgrading, you must use the default driver. More information can be found in *Book “Operations Guide”, Chapter 9 “Managing Networking”, Section 9.3 “Networking Service Overview”, Section 9.3.7 “Using IPAM Drivers in the Networking Service”*.

Use these checklists to confirm and record your network configuration information.

Router

The IP router used with SUSE OpenStack Cloud must support the updated of its ARP table through gratuitous ARP packets.

PXE Installation Network

When provisioning the IP range, allocate sufficient IP addresses to cover both the current number of servers and any planned expansion. Use the following table to help calculate the requirements:

Instance	Description	IPs
Deployer O/S		1
Controller server O/S (x3)		3
Compute servers (2nd thru 100th)	single IP per server	
block storage host servers	single IP per server	

#	Item	Value
	Network is untagged	
	No DHCP servers other than SUSE OpenStack Cloud are on the network	
	Switch PVID used to map any "internal" VLANs to untagged	

#	Item	Value
	Routable to the IPMI network	
	IP CIDR	

	IP Range (Usable IPs)	begin:
		end:

	Default IP Gateway	

Management Network

The management network is the backbone used for the majority of SUSE OpenStack Cloud management communications. Control messages are exchanged between the Controllers, Compute hosts, and Cinder backends through this network. In addition to the control flows, the management network is also used to transport Swift and iSCSI based Cinder block storage traffic between servers.

When provisioning the IP Range, allocate sufficient IP addresses to cover both the current number of servers and any planned expansion. Use the following table to help calculate the requirements:

Instance	Description	IPs
Controller server O/S (x3)		3
Controller VIP		1
Compute servers (2nd through 100th)	single IP per server	
VM servers	single IP per server	
VIP per cluster		

#	Item	Value
	Network is untagged	
	No DHCP servers other than SUSE OpenStack Cloud are on the network	

#	Item	Value
	Switch PVID used to map any "internal" VLANs to untagged	
	IP CIDR	

	IP Range (Usable IPs)	begin:
		end:

	Default IP Gateway	
	VLAN ID	

IPMI Network

The IPMI network is used to connect the IPMI interfaces on the servers that are assigned for use with implementing the cloud. This network is used by Cobbler to control the state of the servers during baremetal deployments.

#	Item	Value
	Network is untagged	
	Routable to the Management Network	
	IP Subnet	
	Default IP Gateway	

External API Network

The External network is used to connect OpenStack endpoints to an external public network such as a company's intranet or the public internet in the case of a public cloud provider.

When provisioning the IP Range, allocate sufficient IP addresses to cover both the current number of servers and any planned expansion. Use the following table to help calculate the requirements.

Instance	Description	IPs
Controller server O/S (x3)		3
Controller VIP		1

#	Item	Value
	VLAN Tag assigned:	
	IP CIDR	

	IP Range (Usable IPs)	begin:
		end:

	Default IP Gateway	
	VLAN ID	

External VM Network

The External VM network is used to connect cloud instances to an external public network such as a company's intranet or the public internet in the case of a public cloud provider. The external network has a predefined range of Floating IPs which are assigned to individual instances to enable communications to and from the instance to the assigned corporate intranet/internet. There should be a route between the External VM and External API networks so that instances provisioned in the cloud, may access the Cloud API endpoints, using the instance floating IPs.

#	Item	Value
	VLAN Tag assigned:	
	IP CIDR	

	IP Range (Usable IPs)	begin:
		end:

	Default IP Gateway	
	VLAN ID	

2.3 Cloud Lifecycle Manager

This server contains the SUSE OpenStack Cloud installer, which is based on Git, Ansible, and Cobbler.

#	Item	Value
	Disk Requirement: Single 8GB disk needed per the *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 2 "Hardware and Software Support Matrix"*	
	Section 3.3, "Installing the SUSE OpenStack Cloud Extension"	
	Ensure your local DNS nameserver is placed into your `/etc/resolv.conf` file	
	Install and configure NTP for your environment	
	Ensure your NTP server(s) is placed into your `/etc/ntp.conf` file	
	NTP time source:	

2.4 Information for the `nic_mappings.yml` Input File

Log on to each type of physical server you have and issue platform-appropriate commands to identify the `bus-address` and `port-num` values that may be required. For example, run the following command:

```
sudo lspci -D | grep -i net
```

and enter this information in the space below. Use this information for the `bus-address` value in your `nic_mappings.yml` file.

NIC Adapter PCI Bus Address Output

NIC Adapter PCI Bus Address Output

```

```

To find the `port-num` use:

```
cat /sys/class/net/<device name>/dev_port
```

where the 'device-name' is the name of the device **currently mapped** to this address, not necessarily the name of the device **to be mapped**. Enter the information for your system in the space below.

Network Device Port Number Output

```

```

2.5 Control Plane

The Control Plane consists of at least three servers in a highly available cluster that host the core SUSE OpenStack Cloud services including Nova, Keystone, Glance, Cinder, Heat, Neutron, Swift, Ceilometer, and Horizon. Additional services include mariadb, ip-cluster, apache2, rabbitmq, memcached, zookeeper, kafka, storm, monasca, logging, and cmc.

Note

To mitigate the “split-brain” situation described in *Book “Operations Guide”, Chapter 15 “Troubleshooting Issues”, Section 15.4 “Network Service Troubleshooting”* it is recommended that you have HA network configuration with Multi-Chassis Link Aggregation (MLAG) and NIC bonding configured for all the controllers to deliver system-level redundancy as well network-level resiliency. Also reducing the ARP timeout on the TOR switches will help.

TABLE 2.1: CONTROL PLANE 1

#	Item	Value
	Disk Requirement: 3x 512 GB disks (or enough space to create three logical drives with that amount of space)	
	Ensure the disks are wiped	
	MAC address of first NIC	
	A second NIC, or a set of bonded NICs are required	
	IPMI IP address	
	IPMI Username/Password	

TABLE 2.2: CONTROL PLANE 2

#	Item	Value
	Disk Requirement: 3x 512 GB disks (or enough space to create three logical drives with that amount of space)	
	Ensure the disks are wiped	
	MAC address of first NIC	
	A second NIC, or a set of bonded NICs are required	
	IPMI IP address	
	IPMI Username/Password	

TABLE 2.3: CONTROL PLANE 3

#	Item	Value
	Disk Requirement: 3x 512 GB disks (or enough space to create three logical drives with that amount of space)	
	Ensure the disks are wiped	

#	Item	Value
	MAC address of first NIC	
	A second NIC, or a set of bonded NICs are required	
	IPMI IP address	
	IPMI Username/Password	

2.6 Compute Hosts

One or more KVM Compute servers will be used as the compute host targets for instances.

#	Item
	Disk Requirement: 2x 512 GB disks (or enough space to create three logical drives with that amount of space)
	A NIC for PXE boot and a second NIC, or a NIC for PXE and a set of bonded NICs are required
	Ensure the disks are wiped

Table to record your Compute host details:

ID	NIC MAC Address	IPMI Username/Password	IMPI IP Address	CPU/Mem/Disk

2.7 Storage Hosts

Three or more servers with local disk volumes to provide Cinder block storage resources.

Note

The cluster created from block storage nodes must allow for quorum. In other words, the node count of the cluster must be 3, 5, 7, or another odd number.

#	Item
	Disk Requirement: 3x 512 GB disks (or enough space to create three logical drives with that amount of space) The block storage appliance deployed on a host is expected to consume ~40 GB of disk space from the host root disk for ephemeral storage to run the block storage virtual machine.
	A NIC for PXE boot and a second NIC, or a NIC for PXE and a set of bonded NICs are required
	Ensure the disks are wiped

Table to record your block storage host details:

ID	NIC MAC Address	IPMI Username/Password	IPMI IP Address	CPU/Mem/Disk	Data Volume

2.8 Additional Comments

This section is for any additional information that you deem necessary.

3 Installing the Cloud Lifecycle Manager server

In this chapter you will learn how to install the Cloud Lifecycle Manager from scratch. It will run on SUSE Linux Enterprise Server 12 SP3 and include the SUSE OpenStack Cloud extension and, optionally, the Subscription Management Tool (SMT) server.

3.1 Starting the Operating System Installation

Start the installation by booting into the SUSE Linux Enterprise Server 12 SP3 installation system. For an overview of a default SUSE Linux Enterprise Server installation, refer to the SUSE Linux Enterprise Server *Installation Quick Start* (http://www.suse.com/documentation/sles-12/book_quickstarts/data/art_sle_installquick.html) ↗. Detailed installation instructions (http://www.suse.com/documentation/sles-12/book_sle_deployment/data/cha_inst.html) ↗ are available in the SUSE Linux Enterprise Server *Deployment Guide*. Both documents are available at http://www.suse.com/documentation/sles-12/ ↗.

The following sections will only cover the differences from the default installation process.

3.2 Registration and Online Updates

Registering SUSE Linux Enterprise Server 12 SP3 during the installation process is required for getting product updates and for installing the SUSE OpenStack Cloud extension. Refer to the SUSE Customer Center Registration (http://www.suse.com/documentation/sles-12/book_sle_deployment/data/sec_i_yast2_conf_manual_cc.html) ↗ section of the SUSE Linux Enterprise Server 12 SP3 *Deployment Guide* for further instructions.

After a successful registration you will be asked whether to add the update repositories. If you agree, the latest updates will automatically be installed, ensuring that your system is on the latest patch level after the initial installation. We strongly recommend adding the update repositories immediately. If you choose to skip this step you need to perform an online update later, before starting the installation.

Note: SUSE Login Required

To register a product, you need to have a SUSE login. If you do not have such a login, create it at http://www.suse.com/login.

3.3 Installing the SUSE OpenStack Cloud Extension

SUSE OpenStack Cloud is an extension to SUSE Linux Enterprise Server. Installing it during the SUSE Linux Enterprise Server installation is the easiest and recommended way to set up the Cloud Lifecycle Manager. To get access to the extension selection dialog, you need to register SUSE Linux Enterprise Server 12 SP3 during the installation. After a successful registration, the SUSE Linux Enterprise Server 12 SP3 installation continues with the *Extension & Module Selection*. Choose *SUSE OpenStack Cloud 8* and provide the registration key you obtained by purchasing SUSE OpenStack Cloud. The registration and the extension installation require an Internet connection.

Alternatively, install the SUSE OpenStack Cloud after the SUSE Linux Enterprise Server 12 SP3 installation via *YaST* › *Software* › *Add-On Products*. For details, refer to the section Installing Modules and Extensions from Online Channels (https://www.suse.com/documentation/sles-12/book_sle_deployment/data/sec_add-ons_extensions.html) of the SUSE Linux Enterprise Server 12 SP3 *Deployment Guide*.

3.4 Partitioning

Create a custom partition setup using the *Expert Partitioner*. The following setup is required:

- An LVM setup with no encryption, containing:
  - a volume group named `ardana-vg` on the first disk (`/dev/sda`)
  - a volume named `root` with a size of 50GB and an ext4 filesystem
- no separate mount point for `/home`
- no swap partition or file

3.5 Creating a User

Setting up Cloud Lifecycle Manager requires a regular user which you can set up during the installation. You are free to choose any available user name except for `ardana`, since the `ardana` user is reserved by SUSE OpenStack Cloud.

3.6 Installation Settings

In the final installation step, *Installation Settings*, you need to adjust the software selection for your Cloud Lifecycle Manager setup. For more information refer to the Installation Settings (http://www.suse.com/documentation/sles-12/book_sle_deployment/data/sec_i_yast2_proposal.html) ↗ section of the SUSE Linux Enterprise Server 12 SP3 *Deployment Guide*.

Important: Disable Default Firewall

The default firewall must be disabled, as SUSE OpenStack Cloud enables its own firewall during deployment. You can disable the default firewall in the *Installation Settings* screen of the installation wizard.

3.6.1 Software Selection

Installing a minimal base system is sufficient to set up the Administration Server. The following patterns are the minimum required:

- *Base System*
- *Minimal System (Appliances)*
- *Meta Package for Pattern cloud-ardana* (in case you have chosen to install the SUSE OpenStack Cloud Extension)
- *Subscription Management Tool* (optional, also see *Tip: Installing a Local SMT Server (Optional)*)
- *YaST2 configuration packages*

Tip: Installing a Local SMT Server (Optional)

If you do not have a SUSE Manager or SMT server in your organization, or are planning to manually update the repositories required for deployment of the SUSE OpenStack Cloud nodes, you need to set up an SMT server on the Administration Server. Choose the pattern *Subscription Management Tool* in addition to the patterns listed above to install the SMT server software.

4 Installing and Setting Up an SMT Server on the Cloud Lifecycle Manager server (Optional)

One way to provide the repositories needed to set up the nodes in SUSE OpenStack Cloud is to install a Subscription Management Tool (SMT) server on the Cloud Lifecycle Manager server, and then mirror all repositories from SUSE Customer Center via this server. Installing an SMT server on the Cloud Lifecycle Manager server is optional. If your organization already provides an SMT server or a SUSE Manager server that can be accessed from the Cloud Lifecycle Manager server, skip this step.

Important: Use of SMT Server and Ports

When installing an SMT server on the Cloud Lifecycle Manager server, use it exclusively for SUSE OpenStack Cloud. To use the SMT server for other products, run it outside of SUSE OpenStack Cloud. Make sure it can be accessed from the Cloud Lifecycle Manager for mirroring the repositories needed for SUSE OpenStack Cloud.

When the SMT server is installed on the Cloud Lifecycle Manager server, Cloud Lifecycle Manager provides the mirrored repositories on port `79`.

4.1 SMT Installation

If you have not installed the SMT server during the initial Cloud Lifecycle Manager server installation as suggested in *Section 3.6.1, "Software Selection"*, run the following command to install it:

```
sudo zypper in -t pattern smt
```

4.2 SMT Configuration

No matter whether the SMT server was installed during the initial installation or in the running system, it needs to be configured with the following steps.

Note: Prerequisites

To configure the SMT server, a SUSE account is required. If you do not have such an account, register at http://www.suse.com/login. All products and extensions for which you want to mirror updates with the SMT server should be registered at the SUSE Customer Center (http://scc.suse.com/).

1. Configuring the SMT server requires you to have your mirroring credentials (user name and password) and your registration e-mail address at hand. To access them, proceed as follows:

   a. Open a Web browser and log in to the SUSE Customer Center at http://scc.suse.com/.

   b. Click your name to see the e-mail address which you have registered.

   c. Click *Organization* › *Organization Credentials* to obtain your mirroring credentials (user name and password).

2. Start *YaST* › *Network Services* › *SMT Configuration Wizard*.

3. Activate *Enable Subscription Management Tool Service (SMT)*.

4. Enter the *Customer Center Configuration* data as follows:

   Use Custom Server: Do *not* activate this option
   User: The user name you retrieved from the SUSE Customer Center
   Password: The password you retrieved from the SUSE Customer Center
   Check your input with *Test*. If the test does not return `success`, check the credentials you entered.

5. Enter the e-mail address you retrieved from the SUSE Customer Center at *SCC E-Mail Used for Registration*.

6. *Your SMT Server URL* shows the HTTP address of your server. Usually it should not be necessary to change it.

7. Select *Next* to proceed to step two of the *SMT Configuration Wizard*.

8. Enter a *Database Password for SMT User* and confirm it by entering it once again.

9. Enter one or more e-mail addresses to which SMT status reports are sent by selecting *Add*.

10. Select *Next* to save your SMT configuration. When setting up the database you will be prompted for the MariaDB root password. If you have not already created one then create it in this step. Note that this is the global MariaDB root password, not the database password for the SMT user you specified before.
    The SMT server requires a server certificate at `/etc/pki/trust/anchors/YaST-CA.pem`. Choose *Run CA Management*, provide a password and choose *Next* to create such a certificate. If your organization already provides a CA certificate, *Skip* this step and import the certificate via *YaST* › *Security and Users* › *CA Management* after the SMT configuration is done. See http://www.suse.com/documentation/sles-12/book_security/data/cha_security_yast_ca.html ↗ for more information.
    After you complete your configuration a synchronization check with the SUSE Customer Center will run, which may take several minutes.

4.3 Setting up Repository Mirroring on the SMT Server

The final step in setting up the SMT server is configuring it to mirror the repositories needed for SUSE OpenStack Cloud. The SMT server mirrors the repositories from the SUSE Customer Center. Make sure to have the appropriate subscriptions registered in SUSE Customer Center with the same e-mail address you specified when configuring SMT.

4.3.1 Adding Mandatory Repositories

Mirroring the SUSE Linux Enterprise Server 12 SP3 and SUSE OpenStack Cloud 8 repositories is mandatory. Run the following commands as user `root` to add them to the list of mirrored repositories:

```
for REPO in SLES12-SP3-{Pool,Updates} SUSE-OpenStack-Cloud-8-{Pool,Updates}; do
  smt-repos $REPO sle-12-x86_64 -e
done
```

4.3.2 Updating the Repositories

New repositories added to SMT must be updated immediately by running the following command as user `root`:

```
smt-mirror -L /var/log/smt/smt-mirror.log
```

This command will download several GB of patches. This process may last up to several hours. A log file is written to `/var/log/smt/smt-mirror.log`. After this first manual update the repositories are updated automatically via cron job. A list of all repositories and their location in the file system on the Cloud Lifecycle Manager server can be found at *Table 5.2, "SMT Repositories Hosted on the Administration Server"*.

4.4 For More Information

For detailed information about SMT refer to the Subscription Management Tool manual at http://www.suse.com/documentation/sles-12/book_smt/data/book_smt.html ↗.

5 Software Repository Setup

Software repositories containing products, extensions, and the respective updates for all software need to be available to all nodes in SUSE OpenStack Cloud in order to complete the deployment. These can be managed manually, or they can be hosted on the Cloud Lifecycle Manager server. In this configuration step, these repositories are made available on the Cloud Lifecycle Manager server. There are two types of repositories:

Product Media Repositories: Product media repositories are copies of the installation media. They need to be directly copied to the Cloud Lifecycle Manager server, “loop-mounted” from an iso image, or mounted from a remote server via NFS. Affected are SUSE Linux Enterprise Server 12 SP3 and SUSE OpenStack Cloud 8. These are static repositories; they do not change or receive updates. See *Section 5.1, “Copying the Product Media Repositories”* for setup instructions.

Update and Pool Repositories: Update and Pool repositories are provided by the SUSE Customer Center. They contain all updates and patches for the products and extensions. To make them available for SUSE OpenStack Cloud they need to be mirrored from the SUSE Customer Center. Since their content is regularly updated, they must be kept in synchronization with SUSE Customer Center. For these purposes, SUSE provides either the Subscription Management Tool (SMT) or the SUSE Manager. See *Section 5.2, “Update and Pool Repositories”* for setup instructions.

5.1 Copying the Product Media Repositories

The files in the product repositories for SUSE OpenStack Cloud do not change, therefore they do not need to be synchronized with a remote source. If you have installed the product media from a downloaded ISO image, the product repositories will automatically be made available to the client nodes and these steps can be skipped. These steps can also be skipped if you prefer to install from the Pool repositories provided by SUSE Customer Center. Otherwise, it is sufficient to either copy the data (from a remote host or the installation media), to mount the product repository from a remote server via `NFS`, or to loop mount a copy of the installation images.

If you choose to install from the product media rather than from the SUSE Customer Center repositories, the following product media needs to reside in the specified directories:

TABLE 5.1: LOCAL PRODUCT REPOSITORIES FOR SUSE OPENSTACK CLOUD

Repository	**Directory**
SUSE OpenStack Cloud 8 DVD #1	`/srv/www/suse-12.3/x86_64/repos/Cloud`

The data can be copied by a variety of methods:

Copying from the Installation Media

We recommend using **`rsync`** for copying. If the installation data is located on a removable device, make sure to mount it first (for example, after inserting the DVD1 in the Administration Server and waiting for the device to become ready):

SUSE OpenStack Cloud 8 DVD#1

```
mkdir -p /srv/www/suse-12.3/x86_64/repos/Cloud
mount /dev/dvd /mnt
rsync -avP /mnt/ /srv/www/suse-12.3/x86_64/repos/Cloud/
umount /mnt
```

Copying from a Remote Host

If the data is provided by a remote machine, log in to that machine and push the data to the Administration Server (which has the IP address `192.168.245.10` in the following example):

SUSE OpenStack Cloud 8 DVD#1

```
mkdir -p /srv/www/suse-12.3/x86_64/repos/Cloud
rsync -avPz /data/SUSE-OPENSTACK-CLOUD//DVD1/ 192.168.245.10:/srv/www/suse-12.3/
x86_64/repos/Cloud/
```

Mounting from an NFS Server

If the installation data is provided via NFS by a remote machine, mount the respective shares as follows. To automatically mount these directories either create entries in `/etc/fstab` or set up the automounter.

SUSE OpenStack Cloud 8 DVD#1

```
mkdir -p /srv/www/suse-12.3/x86_64/repos/Cloud/
mount -t nfs nfs.example.com:/exports/SUSE-OPENSTACK-CLOUD/DVD1/ /srv/www/suse-12.3/
x86_64/repos/Cloud
```

5.2 Update and Pool Repositories

Update and Pool Repositories are required on the Cloud Lifecycle Manager server to set up and maintain the SUSE OpenStack Cloud nodes. They are provided by SUSE Customer Center and contain all software packages needed to install SUSE Linux Enterprise Server 12 SP3 and the extensions (pool repositories). In addition, they contain all updates and patches (update repositories).

The repositories can be made available on the Cloud Lifecycle Manager server using one or more of the following methods:

- *Section 5.2.1, " Repositories Hosted on an SMT Server Installed on the Administration Server "*
- *Section 5.2.2, "Alternative Ways to Make the Repositories Available"*

5.2.1 Repositories Hosted on an SMT Server Installed on the Administration Server

When all update and pool repositories are managed by an SMT server installed on the Cloud Lifecycle Manager server (see *Chapter 4, Installing and Setting Up an SMT Server on the Cloud Lifecycle Manager server (Optional)*), the Cloud Lifecycle Manager automatically detects all available repositories. No further action is required.

5.2.2 Alternative Ways to Make the Repositories Available

If you want to keep your SUSE OpenStack Cloud network as isolated from the company network as possible, or your infrastructure does not allow accessing a SUSE Manager or an SMT server, you can alternatively provide access to the required repositories by one of the following methods:

- Mount the repositories from a remote server.
- Synchronize the repositories from a remote server (for example via **`rsync`** and cron).
- Manually synchronize the update repositories from removable media.

The repositories must be made available at the default locations on the Cloud Lifecycle Manager server as listed in *Table 5.4, "Repository Locations on the Cloud Lifecycle Manager server"*.

5.3 Repository Locations

The following tables show the locations of all repositories that can be used for SUSE OpenStack Cloud.

TABLE 5.2: SMT REPOSITORIES HOSTED ON THE ADMINISTRATION SERVER

Repository	Directory
Mandatory Repositories	
SLES12-SP3-Pool	`/srv/www/htdocs/repo/SUSE/Products/SLE-SERVER/12-SP3/x86_64/product/`
SLES12-SP3-Updates	`/srv/www/htdocs/repo/SUSE/Updates/SLE-SERVER/12-SP3/x86_64/update/`
SUSE-OpenStack-Cloud-8-Pool	`/srv/www/htdocs/repo/SUSE/Products/OpenStack-Cloud/8/x86_64/product/`
SUSE-OpenStack-Cloud-8-Updates	`/srv/www/htdocs/repo/SUSE/Updates/OpenStack-Cloud/8/x86_64/update/`

TABLE 5.3: SUSE MANAGER REPOSITORIES (CHANNELS)

Repository	URL
Mandatory Repositories	
SLES12-SP3-Updates	`http://manager.example.com/ks/dist/child/sles12-sp3-updates-x86_64/sles12-sp3-x86_64/`
SUSE-OpenStack-Cloud-8-Pool	`http://manager.example.com/ks/dist/child/suse-openstack-cloud-8-pool-x86_64/sles12-sp3-x86_64/`
SUSE-OpenStack-Cloud-8-Updates	`http://manager.example.com/ks/dist/child/suse-openstack-cloud-8-updates-x86_64/sles12-sp3-x86_64/`

The following table shows the required repository locations to use when manually copying, synchronizing, or mounting the repositories.

TABLE 5.4: REPOSITORY LOCATIONS ON THE CLOUD LIFECYCLE MANAGER SERVER

Channel	Directory on the Administration Server
Mandatory Repositories	
SLES12-SP3-Pool	`/srv/www/suse-12.3/x86_64/repos/SLES12-SP3-Pool/`
SLES12-SP3-Updates	`/srv/www/suse-12.3/x86_64/repos/SLES12-SP3-Updates/`
SUSE-OpenStack-Cloud-8-Pool	`/srv/www/suse-12.3/x86_64/repos/SUSE-OpenStack-Cloud-8-Pool/`
SUSE-OpenStack-Cloud-8-Updates	`/srv/www/suse-12.3/x86_64/repos/SUSE-OpenStack-Cloud-8-Updates`

6 Boot from SAN and Multipath Configuration

6.1 Introduction

For information about supported hardware for multipathing, see *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 2 "Hardware and Software Support Matrix", Section 2.2 "Supported Hardware Configurations"*.

Important

When exporting a LUN to a node for boot from SAN, you should ensure that *LUN 0* is assigned to the LUN and configure any setup dialog that is necessary in the firmware to consume this LUN 0 for OS boot.

Important

Any hosts that are connected to 3PAR storage must have a `host persona` of `2-generic-alua` set on the 3PAR. Refer to the 3PAR documentation for the steps necessary to check this and change if necessary.

iSCSI boot from SAN is not supported. For more information on the use of Cinder with multipath, see *Section 18.1.3, "Multipath Support"*.

To allow SUSE OpenStack Cloud 8 to use volumes from a SAN, you have to specify configuration options for both the installation and the OS configuration phase. In all cases, the devices that are utilized are devices for which multipath is configured.

6.2 Install Phase Configuration

For FC connected nodes and for FCoE nodes where the network processor used is from the Emulex family such as for the 650FLB, the following changes need to be made.

1. In each stanza of the `servers.yml` insert a line stating `boot-from-san: true`

```
    - id: controller2
      ip-addr: 192.168.10.4
      role: CONTROLLER-ROLE
      server-group: RACK2
      nic-mapping: HP-DL360-4PORT
      mac-addr: "8a:8e:64:55:43:76"
      ilo-ip: 192.168.9.4
      ilo-password: password
      ilo-user: admin
      boot-from-san: true
```

This uses the disk `/dev/mapper/mpatha` as the default device on which to install the OS.

2. In the disk input models, specify the devices that will be used via their multipath names (which will be of the form `/dev/mapper/mpatha`, `/dev/mapper/mpathb` etc.).

```
    volume-groups:
      - name: ardana-vg
        physical-volumes:

          # NOTE: 'sda_root' is a templated value. This value is checked in
          # os-config and replaced by the partition actually used on sda
          #for example sda1 or sda5
          - /dev/mapper/mpatha_root

...
      - name: vg-comp
        physical-volumes:
          - /dev/mapper/mpathb
```

6.3 QLogic FCoE restrictions and additional configurations

If you are using network cards such as Qlogic Flex Fabric 536 and 630 series, there are additional OS configuration steps to support the importation of LUNs as well as some restrictions on supported configurations.

The restrictions are:

- Only one network card can be enabled in the system.
- The FCoE interfaces on this card are dedicated to FCoE traffic. They cannot have IP addresses associated with them.
- NIC mapping cannot be used.

In addition to the configuration options above, you also need to specify the FCoE interfaces for install and for os configuration. There are 3 places where you need to add additional configuration options for fcoe-support:

- In `servers.yml`, which is used for configuration of the system during OS install, FCoE interfaces need to be specified for each server. In particular, the mac addresses of the FCoE interfaces need to be given, *not* the symbolic name (for example, `eth2`).

```
    - id: compute1
      ip-addr: 10.245.224.201
      role: COMPUTE-ROLE
      server-group: RACK2
      mac-addr: 6c:c2:17:33:4c:a0
      ilo-ip: 10.1.66.26
      ilo-user: linuxbox
      ilo-password: linuxbox123
      boot-from-san: True
      fcoe-interfaces:
         - 6c:c2:17:33:4c:a1
         - 6c:c2:17:33:4c:a9
```

Important

NIC mapping cannot be used.

- For the osconfig phase, you will need to specify the `fcoe-interfaces` as a peer of `network-interfaces` in the `net_interfaces.yml` file:

```
    - name: CONTROLLER-INTERFACES
      fcoe-interfaces:
        - name: fcoe
          devices:
             - eth2
             - eth3
      network-interfaces:
        - name: eth0
          device:
              name: eth0
          network-groups:
            - EXTERNAL-API
            - EXTERNAL-VM
            - GUEST
            - MANAGEMENT
```

Important

The MAC addresses specified in the `fcoe-interfaces` stanza in `servers.yml` must correspond to the symbolic names used in the `fcoe-interfaces` stanza in `net_interfaces.yml`.

Also, to satisfy the FCoE restriction outlined in *Section 6.3, "QLogic FCoE restrictions and additional configurations"* above, there can be no overlap between the devices in `fcoe-interfaces` and those in `network-interfaces` in the `net_interfaces.yml` file. In the example, `eth2` and `eth3` are `fcoe-interfaces` while `eth0` is in `network-interfaces`.

- As part of the initial install from an iso, additional parameters need to be supplied on the kernel command line:

```
multipath=true partman-fcoe/interfaces=<mac address1>,<mac address2> disk-detect/
fcoe/enable=true --- quiet
```

  See the sections of installing from an ISO using UEFI (*Section 6.4.2, "UEFI Boot Mode with QLogic FCoE"*) and BIOS (*Section 6.4.4, "Legacy BIOS Boot Mode with QLogic FCoE"*).

Since NIC mapping is not used to guarantee order of the networks across the system the installer will remap the network interfaces in a deterministic fashion as part of the install. As part of the installer dialogue, if DHCP is not configured for the interface, it is necessary to confirm that the appropriate interface is assigned the ip address. The network interfaces may not be at the names expected when installing via an ISO. When you are asked to apply an IP address to an interface, press Alt–F2 and in the console window, run the command **`ip a`** to examine the interfaces and their associated MAC addresses. Make a note of the interface name with the expected MAC address and use this in the subsequent dialog. Press Alt–F1 to return to the installation screen. You should note that the names of the interfaces may have changed after the installation completes. These names are used consistently in any subsequent operations.

Therefore, even if FCoE is not used for boot from SAN (for example for cinder), then it is recommended that `fcoe-interfaces` be specified as part of install (without the multipath or disk detect options). Alternatively, you need to run `osconfig-fcoe-reorder.yml` before `site.yml` or `osconfig-run.yml` is invoked to reorder the networks in a similar manner to the installer. In this case, the nodes will need to be manually rebooted for the network reorder to take effect. Run `osconfig-fcoe-reorder.yml` in the following scenarios:

- If you have used a third-party installer to provision your bare-metal nodes
- If you are booting from a local disk (that is one that is not presented from the SAN) but you want to use FCoE later, for example, for cinder.

To run the command:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts osconfig-fcoe-reorder.yml
```

If you do not run `osconfig-fcoe-reorder.yml`, you will encounter a failure in `osconfig-run.yml`.

If you are booting from a local disk, the LUNs that will be imported over FCoE will not be visible before `site.yml` or `osconfig-run.yml` has been run. However, if you need to import the LUNs before this, for instance, in scenarios where you need to run `wipe_disks.yml`, then you can run the `fcoe-enable` playbook across the nodes in question. This will configure FCoE and import the LUNs presented to the nodes.

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/verb_hosts fcoe-enable.yml
```

6.4 Installing the SUSE OpenStack Cloud 8 ISO for nodes that support Boot from SAN

This section describes how to install the ISO on the Cloud Lifecycle Manager to support the following configurations:

- *Section 6.4.1, "UEFI Boot Mode"*
- *Section 6.4.2, "UEFI Boot Mode with QLogic FCoE"*
- *Section 6.4.3, "Legacy BIOS Boot Mode"*
- *Section 6.4.4, "Legacy BIOS Boot Mode with QLogic FCoE"*

The Cloud Lifecycle Manager will then automatically handle the installation on nodes that support Boot from SAN based on the configuration information in `servers.yml` and the disk models, as described in the preceding section.

6.4.1 UEFI Boot Mode

1. On the installer boot menu, press `E` to enter the `Edit Selection` screen.
2. On the line that starts with `linux`, enter the text `multipath=true` before `--- quiet`.
3. Press `Ctrl`–`X` or `F10` to proceed with the installation.

6.4.2 UEFI Boot Mode with QLogic FCoE

In addition to inserting `multipath=true`, it is necessary to supply details of the FCoE network interfaces.

1. At the installer boot menu, press `E` to enter the `Edit Selection` screen.
2. On the line that starts with `linux`, enter the text `partman-fcoe/interfaces=6c:c2:17:33:4c:a1,6c:c2:17:33:4c:a9 disk-detect/fcoe/enable=true` between `multipath=true` and `--- quiet`.
3. Press `Ctrl`–`X` or `F10` to proceed with the installation.

6.4.3 Legacy BIOS Boot Mode

1. On the installer boot menu, navigate (using `↓`) to the `Advanced options` entry and then press `Enter`.
2. Navigate to the `Multipath install` entry and press `Enter` to start the installation.

6.4.4 Legacy BIOS Boot Mode with QLogic FCoE

1. On the installer boot menu, navigate (using `↓`) to the `Advanced options` entry and then press `Enter`.

2. Navigate to the `Multipath install` entry and press →| to edit the entry details. The kernel command line is now displayed at the bottom of the screen and `multipath=true` is already specified.

3. Edit the kernel command line to add the FCoE network interfaces before `--- quiet`. Add `partman-fcoe/interfaces=6c:c2:17:33:4c:a1,6c:c2:17:33:4c:a9 disk-detect/fcoe/enable=true` before `--- quiet`.

4. Now press Enter to start the installation.

7 Installing the L2 Gateway Agent for the Networking Service

The L2 gateway is a service plug-in to the Neutron networking service that allows two L2 networks to be seamlessly connected to create a single L2 broadcast domain. The initial implementation provides for the ability to connect a virtual Neutron VxLAN network to a physical VLAN using a VTEP-capable HPE 5930 switch. The L2 gateway is to be enabled only for VxLAN deployments.

To begin L2 gateway agent setup, you need to configure your switch. These instructions use an HPE FlexFabric 5930 Switch Series (http://www8.hp.com/us/en/products/networking-switches/product-detail.html?oid=6604154#!tab=models) switch.

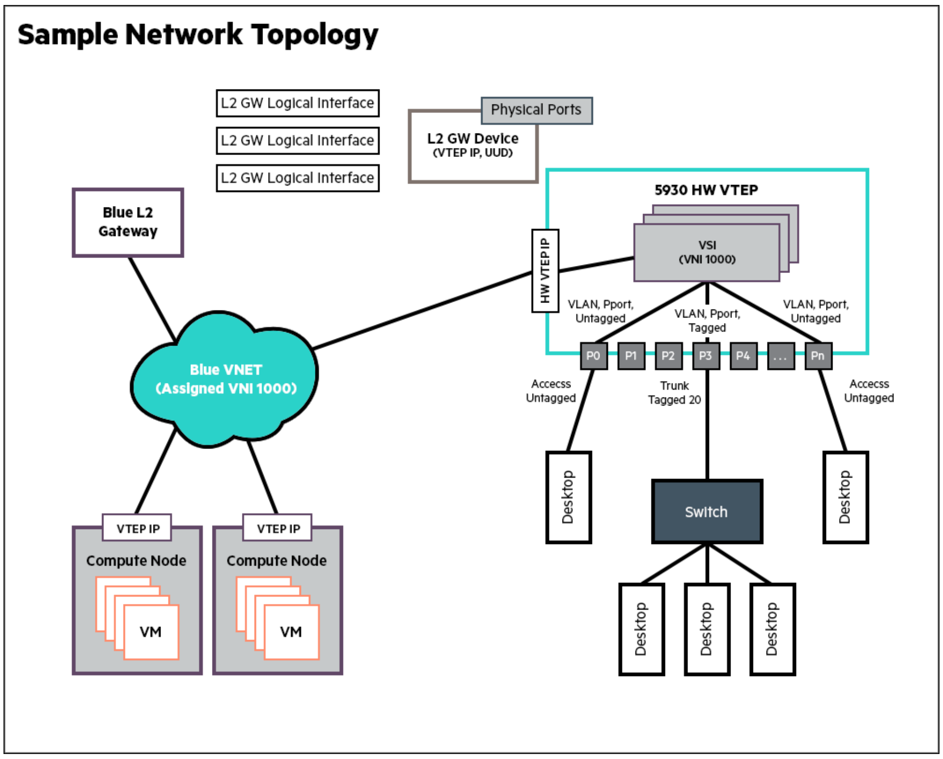
7.1 Sample network topology (for illustration purposes)

When viewing the following network diagram, assume that the blue VNET has been created by the tenant and has been assigned a segmentation ID of 1000 (VNI 1000). The Cloud Admin is now connecting physical servers to this VNET.

Assume also that the blue L2 Gateway is created and points to the HW VTEPS, the physical ports, the VLAN, and if it is an access or trunk port (tagged or untagged)

Note

This example does not apply to distributed route networks, which use Distributed Virtual Routers (DVR).



7.2 Networks

The following diagram illustrates an example network configuration. It does not apply to distributed route networks, which use Distributed Virtual Routers (DVR).

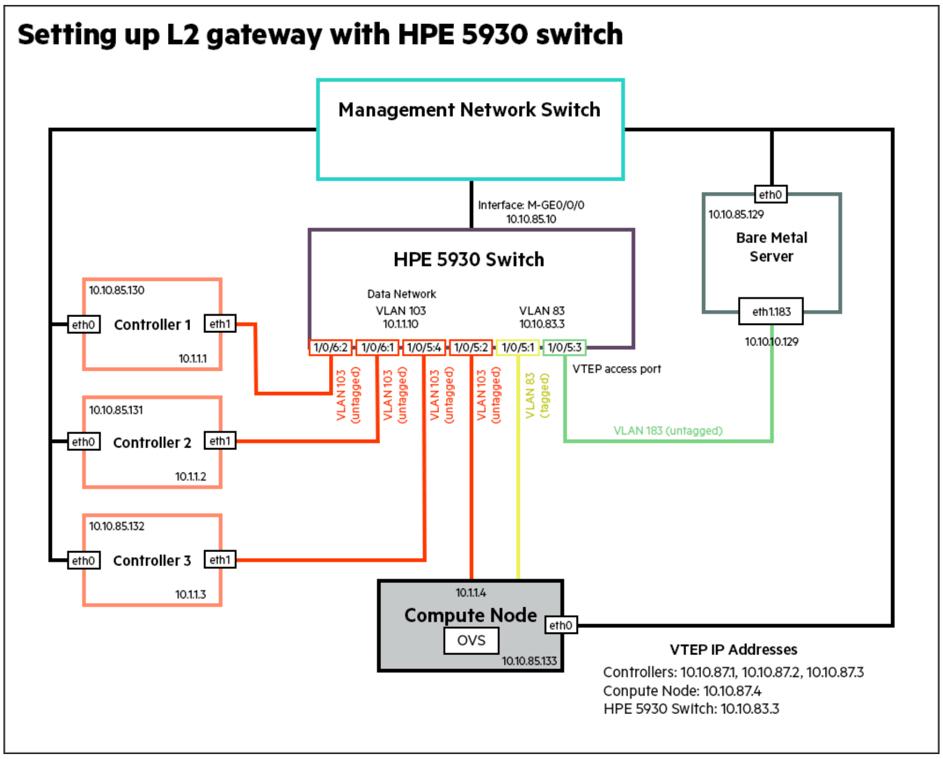
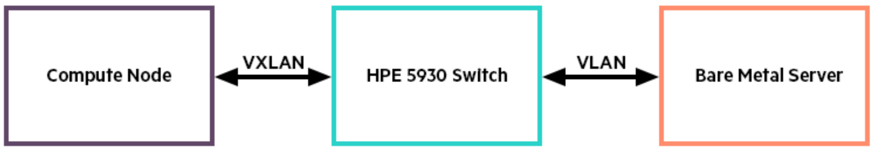


FIGURE 7.1: L2 GATEWAY AND 5930 SWITCH

An L2 gateway is useful in extending virtual networks (VxLAN) in a cloud onto physical VLAN networks. The L2 gateway switch converts VxLAN packets into VLAN packets and back again, as shown in the following diagram. This topic assumes a VxLAN deployment.



- Management Network: 10.10.85.0/24
- Data Network: 10.1.1.0/24
- Tenant VM Network: 10.10.10.0/24

Note

These IP ranges are used in the topology shown in the diagram for illustration only.

7.3 HPE 5930 switch configuration

1. Telnet to the 5930 switch and provide your username and password.

2. Go into system view:

```
system-view
```

3. Create the required VLANs and VLAN ranges:

```
vlan 103
vlan 83
vlan 183
vlan 1261 to 1270
```

4. Assign an IP address to VLAN 103. This is used as a data path network for VxLAN traffic.

```
interface vlan 103
ip address 10.1.1.10 255.255.255.0
```

5. Assign an IP address to VLAN 83. This is used as a hardware VTEP network.

```
interface vlan 83
ip address 10.10.83.3 255.255.255.0
```

The 5930 switch has a fortygigE1/0/5 interface to which a splitter cable is connected that splits the network into four tengigEthernet (tengigEthernet1/0/5:1 to tengigEthernet1/0/5:4) interfaces:

- tengigEthernet1/0/5:1 and tengigEthernet1/0/5:2 are connected to the compute node. This is required just to bring the interface up. In other words, in order to have the HPE 5930 switch work as a router, there should be at least one interface of that particular VLAN up. Alternatively, the interface can be connected to any host or network element.

- tengigEthernet1/0/5:3 is connected to a baremetal server.

- tengigEthernet1/0/5:4 is connected to controller 3, as shown in the figure *L2 Gateway and 5930 Switch*.

The switch's fortygigE1/0/6 interface to which the splitter cable is connected splits it into four tengigEthernet (tengigEthernet1/0/6:1 to tengigEthernet1/0/6:4) interfaces:

- tengigEthernet1/0/6:1 is connected to controller 2
- tengigEthernet1/0/6:2 is connected to controller 1
  Note: 6:3 and 6:4 are not used although they are available.

6. Split the fortygigE 1/0/5 interface into tengig interfaces:

```
interface fortygigE 1/0/5
using tengig
The interface FortyGigE1/0/5 will be deleted. Continue? [Y/N]: y
```

7. Configure the Ten-GigabitEthernet1/0/5:1 interface:

```
interface Ten-GigabitEthernet1/0/5:1
port link-type trunk
port trunk permit vlan 83
```

7.4 Configuring the Provider Data Path Network

1. Configure the Ten-GigabitEthernet1/0/5:2 interface:

```
interface Ten-GigabitEthernet1/0/5:2
port link-type trunk
port trunk permit vlan 103
port trunk permit vlan 1261 to 1270
port trunk pvid vlan 103
```

2. Configure the Ten-GigabitEthernet1/0/5:4 interface:

```
interface Ten-GigabitEthernet1/0/5:4
port link-type trunk
port trunk permit vlan 103
port trunk permit vlan 1261 to 1270
port trunk pvid vlan 103
```

3. Configure the Ten-GigabitEthernet1/0/5:3 interface:

```
interface Ten-GigabitEthernet1/0/5:3
port link-type trunk
port trunk permit vlan 183
vtep access port
```

4. Split the fortygigE 1/0/6 interface into tengig interfaces:

   ```
   interface fortygigE 1/0/6
   using tengig
   The interface FortyGigE1/0/6 will be deleted. Continue? [Y/N]: y
   ```

5. Configure the Ten-GigabitEthernet1/0/6:1 interface:

   ```
   interface Ten-GigabitEthernet1/0/6:1
   port link-type trunk
   port trunk permit vlan 103
   port trunk permit vlan 1261 to 1270
   port trunk pvid vlan 103
   ```

6. Configure the Ten-GigabitEthernet1/0/6:2 interface:

   ```
   interface Ten-GigabitEthernet1/0/6:2
   port link-type trunk
   port trunk permit vlan 103
   port trunk permit vlan 1261 to 1270
   port trunk pvid vlan 103
   ```

7. Enable l2vpn:

   ```
   l2vpn enable
   ```

8. Configure a passive TCP connection for OVSDB on port 6632:

   ```
   ovsdb server ptcp port 6632
   ```

9. Enable OVSDB server:

   ```
   ovsdb server enable
   ```

10. Enable a VTEP process:

    ```
    vtep enable
    ```

11. Configure 10.10.83.3 as the VTEP source IP. This acts as a hardware VTEP IP.

    ```
    tunnel global source-address 10.10.83.3
    ```

12. Configure the VTEP access port:

```
interface Ten-GigabitEthernet1/0/5:3
vtep access port
```

13. Disable VxLAN tunnel mac-learning:

```
vxlan tunnel mac-learning disable
```

14. Display the current configuration of the 5930 switch and verify the configuration:

```
display current-configuration
```

After switch configuration is complete, you can dump OVSDB to see the entries.

- Run the ovsdb-client from any Linux machine reachable by the switch:

```
ovsdb-client dump --pretty tcp:10.10.85.10:6632

sdn@small-linuxbox:~$ ovsdb-client dump --pretty tcp:10.10.85.10:6632
Arp_Sources_Local table
_uuid locator src_mac
----- ------- -------

Arp_Sources_Remote table
_uuid locator src_mac
----- ------- -------

Global table
_uuid                                managers switches
------------------------------------ -------- --------------------------------------
2c891edc-439b-4144-84d9-fa...bf []       [f5f4b43b-40bc-4640-b580-d4...88]

Logical_Binding_Stats table
_uuid bytes_from_local bytes_to_local packets_from_local packets_to_local
----- ---------------- -------------- ------------------ ----------------

Logical_Router table
_uuid description name static_routes switch_binding
----- ----------- ---- ------------- --------------

Logical_Switch table
_uuid description name tunnel_key
----- ----------- ---- ----------

Manager table
_uuid inactivity_probe is_connected max_backoff other_config status target
----- ---------------- ------------ ----------- ------------ ------ ------
```

```
Mcast_Macs_Local table
MAC _uuid ipaddr locator_set logical_switch
--- ----- ------ ----------- --------------

Mcast_Macs_Remote table
MAC _uuid ipaddr locator_set logical_switch
--- ----- ------ ----------- --------------

Physical_Locator table
_uuid dst_ip encapsulation_type
----- ------ ------------------

Physical_Locator_Set table
_uuid locators
----- --------

Physical_Port table
_uuid      description name                         port_flt_status vlan_bindings
 vlan_stats
---------- ----------- --------------------------- --------------- -------------
 ----------
fda9...07e ""          "Ten-GigabitEthernet1/0/5:3" [UP]            {}            {}

Physical_Switch table
_uuid     desc... mgmnt_ips name       ports       sw_flt_status tunnel_ips     tunnels
--------- ------- --------- ---------- ----------- ------------- -------------- -------
f5f...688 ""      []        "L2GTWY02" [fda...07e] []            ["10.10.83.3"] []

Tunnel table
_uuid bfd_config_local bfd_config_remote bfd_params bfd_status local remote
----- ---------------- ----------------- ---------- ---------- ----- ------

Ucast_Macs_Local table
MAC _uuid ipaddr locator logical_switch
--- ----- ------ ------- --------------

Ucast_Macs_Remote table
MAC _uuid ipaddr locator logical_switch
--- ----- ------ ------- --------------
```

7.5 Enabling and Configuring the L2 Gateway Agent

1. Update the input model (in `control_plane.yml`) to specify where you want to run the neutron-l2gateway-agent. For example, see the line in bold in the following yml:

```
---
 product:
```

```
version: 2
control-planes:
- name: cp
  region-name: region1
failure-zones:
- AZ1
  common-service-components:
    - logging-producer
    - openstack-monasca-agent
    - freezer-agent
    - stunnel
    - lifecycle-manager-target
clusters:
- name: cluster1
  cluster-prefix: c1
  server-role: ROLE-CONTROLLER
  member-count: 2
  allocation-policy: strict
service-components:

...
- neutron-l2gateway-agent
... )
```

2. Update `l2gateway_agent.ini.j2`. For example, here the IP address (10.10.85.10) must be the management IP address of your 5930 switch. Open the file in vi:

```
$ vi /home/stack/my_cloud/config/neutron/l2gateway_agent.ini.j2
```

3. Then make the changes:

```
[ovsdb]
# (StrOpt) OVSDB server tuples in the format
# <ovsdb_name>:<ip address>:<port>[,<ovsdb_name>:<ip address>:<port>]
# - ovsdb_name: symbolic name that helps identifies keys and certificate files
# - ip address: the address or dns name for the ovsdb server
# - port: the port (ssl is supported)
ovsdb_hosts = hardware_vtep:10.10.85.10:6632
```

4. By default, the L2 gateway agent initiates a connection to OVSDB servers running on the L2 gateway switches. Set the attribute `enable_manager` to `True` if you want to change this behavior (to make L2 gateway switches initiate a connection to the L2 gateway agent). In this case, it is assumed that the Manager table in the OVSDB hardware_vtep schema on the L2 gateway switch has been populated with the management IP address of the L2 gateway agent and the port.

```
#enable_manager = False
#connection can be initiated by the ovsdb server.
#By default 'enable_manager' value is False, turn on the variable to True
#to initiate the connection from ovsdb server to l2gw agent.
```

5. If the port that is configured with `enable_manager = True` is any port other than 6632, update the `2.0/services/neutron/l2gateway-agent.yml` input model file with that port number:

```
endpoints:
    - port: '6632'
      roles:
      - ovsdb-server
```

6. Note: The following command can be used to set the Manager table on the switch from a remote system:

```
sudo vtep-ctl --db=tcp:10.10.85.10:6632 set-manager tcp:10.10.85.130:6632
```

7. For SSL communication, the command is:

```
sudo vtep-ctl --db=tcp:10.10.85.10:6632 set-manager ssl:10.10.85.130:6632
```

   where **10.10.85.10** is the management IP address of the L2 gateway switch and **10.10.85.130** is the management IP of the host on which the L2 gateway agent runs.
   Therefore, in the above topology, this command has to be repeated for **10.10.85.131** and **10.10.85.132**.

8. If you are not using SSL, comment out the following:

```
#l2_gw_agent_priv_key_base_path={{ neutron_l2gateway_agent_creds_dir }}/keys
#l2_gw_agent_cert_base_path={{ neutron_l2gateway_agent_creds_dir }}/certs
#l2_gw_agent_ca_cert_base_path={{ neutron_l2gateway_agent_creds_dir }}/ca_certs
```

9. If you are using SSL, then rather than commenting out the attributes, specify the directory path of the private key, the certificate, and the CA cert that the agent should use to communicate with the L2 gateway switch which has the OVSDB server enabled for SSL communication.
   Make sure that the directory path of the files is given permissions 755, and the files' owner is root and the group is root with 644 permissions.

Private key: The name should be the same as the symbolic name used above in ovsdb_hosts attribute. The extension of the file should be ".key". With respect to the above example, the filename will be hardware_vtep.key

Certificate The name should be the same as the symbolic name used above in ovsdb_hosts attribute. The extension of the file should be “.cert”. With respect to the above example, the filename will be hardware_vtep.cert

CA certificate The name should be the same as the symbolic name used above in ovsdb_hosts attribute. The extension of the file should be “.ca_cert”. With respect to the above example, the filename will be hardware_vtep.ca_cert

10. To enable the HPE 5930 switch for SSL communication, execute the following commands:

```
undo ovsdb server ptcp
undo ovsdb server enable
ovsdb server ca-certificate flash:/cacert.pem bootstrap
ovsdb server certificate flash:/sc-cert.pem
ovsdb server private-key flash:/sc-privkey.pem
ovsdb server pssl port 6632
ovsdb server enable
```

11. Data from the OVSDB sever with SSL can be viewed using the following command:

```
ovsdb-client -C <ca-cert.pem> -p <client-private-key.pem> -c <client-cert.pem> \
dump ssl:10.10.85.10:6632
```

7.6 Routing Between Software and Hardware - VTEP Networks

In order to allow L2 gateway switches to send VxLAN packets over the correct tunnels destined for the compute node and controller node VTEPs, you must ensure that the cloud VTEP (compute and controller) IP addresses are in different network/subnet from that of the L2 gateway switches. You must also create a route between these two networks. This is explained below.

1. In the following example of the input model file `networks.yml`, the GUEST-NET represents the cloud data VxLAN network. REMOTE-NET is the network that represents the hardware VTEP network.

```
# networks.yml
 networks:
    - name: GUEST-NET
```

```
      vlanid: 103
      tagged-vlan: false
      cidr: 10.1.1.0/24
      gateway-ip: 10.1.1.10
      network-group: GUEST

    - name: REMOTE-NET
      vlanid: 183
      tagged-vlan: false
      cidr: 10.10.83.0/24
      gateway-ip: 10.10.83.3
      network-group: REMOTE
```

2. The route must be configured between the two networks in the `network-groups.yml` input model file:

```
# network_groups.yml
 network-groups:
    - name: REMOTE
      routes:
        - GUEST

    - name: GUEST
      hostname-suffix: guest
      tags:
        - neutron.networks.vxlan:
            tenant-vxlan-id-range: "1:5000"
      routes:
        - REMOTE
```

Note

Note that the IP route is configured on the compute node. Per this route, the HPE 5930 acts as a gateway that routes between the two networks.

3. On the compute node, it looks like this:

```
stack@padawan-cp1-comp0001-mgmt:~$ sudo ip route
10.10.83.0/24 via 10.1.1.10 dev eth4
```

4. Run the following Ansible playbooks to apply the changes.
config-processor-run.yml:

```
cd ~/openstack/ardana/ansible
```

```
ansible-playbook -i hosts/localhost config-processor-run.yml
```

ready-deployment.yml:

```
ansible-playbook -i hosts/localhost ready-deployment.yml
```

ardana-reconfigure.yml:

```
cd ~/scratch/ansible/next/ardana/ansible/
ansible-playbook -i hosts/verb_hosts ardana-reconfigure.yml
```

Notes:

- Make sure that the controller cluster is able to reach the management IP address of the L2 gateway switch. Otherwise, the L2 gateway agents running on the controllers will not be able to reach the gateway switches.
- Make sure that the interface on the baremetal server connected to the 5930 switch is tagged (this is explained shortly).

7.7 Connecting a Bare-Metal Server to the HPE 5930 Switch

As SUSE Linux Enterprise Server (bare-metal server) is not aware of tagged packets, it is not possible for a virtual machine to communicate with a baremetal box.

As the administrator, you must manually perform configuration changes to the interface in the HPE 5930 switch to which the baremetal server is connected so that the switch can send untagged packets. Either one of the following command sets can be used to do so:

```
Interface <interface number>
service-instance <service-instance id>
encapsulation untagged
xconnect vsi <vsi-name>
```

Or:

```
Interface <interface number>
service-instance <service-instance id>
encapsulation s-vid <vlan-id>
xconnect vsi <vsi-name> access-mode ethernet
```

There are two ways of configuring the baremetal server to communicate with virtual machines. If the switch sends tagged traffic, then the baremetal server should be able to receive the tagged traffic.

7.8 Configuration on a Bare-Metal Server

If the configuration changes mentioned previously are not made on the switch to send untagged traffic to the bare-metal server, to make the bare-metal server receive tagged traffic from the switch, perform the following on the bare-metal server:

- Bare-metal management IP is 10.10.85.129 on interface em1
- Switch 5930 must be connected to baremetal on eth1
- IP address must be set into tagged interface of eth1

1. Create a tagged (VLAN 183) interface

   ```
   vconfig add eth1 183
   ```

2. Assign the IP address (10.10.10.129) to eth1.183 tagged interface (IP from the subnet 10.10.10.0/24 as VM (10.10.10.4) spawned in Compute node belongs to this subnet).

   ```
   ifconfig eth1.183 10.10.10.129/24
   ```

7.9 NIC Bonding and IRF Configuration

With L2 gateway in actiondeployment, NIC bonding can be enabled on compute nodes. For more details on NIC bonding, please refer to *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 7 "Configuration Objects", Section 7.11 "Interface Models"*. In order to achieve high availability, HPE 5930 switches can be configured to form a cluster using Intelligent Resilient Framework (IRF). Please refer to the HPE FlexFabric 5930 Switch Series configuration guide (http://h20564.www2.hpe.com/hpsc/doc/public/display?docId=c04567141) ↗ for details.

7.10 Scale Numbers Tested

- Number of neutron port MACs tested on a single switch: 4000
- Number of HPE 5930 switches tested: 2
- Number of baremetal connected to a single HPE 5930 switch: 100
- Number of L2 gateway connections to different networks: 800

7.11 L2 Gateway Commands

These commands are not part of the L2 gateway deployment. They are to be executed after L2 gateway is deployed.

1. Create Network

   ```
   neutron net-create net1
   ```

2. Create Subnet

   ```
   neutron subnet-create net1 10.10.10.0/24
   ```

3. Boot a tenant VM (nova boot –image <Image-id> --flavor 2 –nic net-id=<net_id> <VM name>)

   ```
   nova boot --image 1f3cd49d-9239-49cf-8736-76bac5360489 --flavor 2 \
     --nic net-id=4f6c58b6-0acc-4e93-bb4c-439b38c27a23 VM
   ```

   Assume the VM was assigned the IP address 10.10.10.4

4. Create the L2 gateway filling in your information here:

   ```
   neutron l2-gateway-create \
     --device name="SWITCH_NAME",interface_names="INTERFACE_NAME" \
     GATEWAY-NAME
   ```

   For this example:

   ```
   neutron l2-gateway-create \
     --device name="L2GTWY02",interface_names="Ten-GigabitEthernet1/0/5:3" \
     gw1
   ```

Ping from the VM (10.10.10.4) to baremetal server and from baremetal server (10.10.10.129) to the VM Ping should not work as there is no gateway connection created yet.

5. Create l2 gateway Connection

```
neutron l2-gateway-connection-create gw1 net1 --segmentation-id 183
```

6. Ping from VM (10.10.10.4) to baremetal and from baremetal (10.10.10.129) to VM Ping should work.

7. Delete l2 gateway Connection

```
neutron l2-gateway-connection-delete <gateway id/gateway_name>
```

8. Ping from the VM (10.10.10.4) to baremetal and from baremetal (10.10.10.129) to the VM. Ping should not work as l2 gateway connection was deleted.

II Cloud Installation

8 Overview

Before starting the installation, review the sample configurations offered by SUSE OpenStack Cloud in *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”*. SUSE OpenStack Cloud ships with highly tuned example configurations for each of these cloud models:

- *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.3 “KVM Examples”, Section 10.3.1 “Entry-Scale Cloud”*
- *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.4 “ESX Examples”, Section 10.4.1 “Single-Region Entry-Scale Cloud with a Mix of KVM and ESX Hypervisors”*
- *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.5 “Swift Examples”, Section 10.5.1 “Entry-scale Swift Model”*
- *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.3 “KVM Examples”, Section 10.3.3 “Single-Region Mid-Size Model”*

Name	**Location**
Book “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.3 “KVM Examples”, Section 10.3.1 “Entry-Scale Cloud”*	`~/openstack/examples/entry-scale-kvm`
Book “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.3 “KVM Examples”, Section 10.3.2 “Entry Scale Cloud with Metering and Monitoring Services”*	`~/openstack/examples/entry-scale-kvm-mml`
Book “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.4 “ESX Examples”, Section 10.4.1 “Single-Region Entry-Scale Cloud with a Mix of KVM and ESX Hypervisors”*	`~/openstack/examples/entry-scale-esx-kvm`

Name	Location
Book “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.4 “ESX Examples”, Section 10.4.2 “Single-Region Entry-Scale Cloud with Metering and Monitoring Services, and a Mix of KVM and ESX Hypervisors”*	`~/openstack/examples/entry-scale-esx-kvm-mml`
Book “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.5 “Swift Examples”, Section 10.5.1 “Entry-scale Swift Model”*	`~/openstack/examples/entry-scale-swift`
Book “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.6 “Ironic Examples”, Section 10.6.1 “Entry-Scale Cloud with Ironic Flat Network”*	`~/openstack/examples/entry-scale-ironic-flat-network`
Book “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.6 “Ironic Examples”, Section 10.6.2 “Entry-Scale Cloud with Ironic Multi-Tenancy”*	`~/openstack/examples/entry-scale-ironic-multi-tenancy`
Book “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”, Section 10.3 “KVM Examples”, Section 10.3.3 “Single-Region Mid-Size Model”*	`~/openstack/examples/mid-scale-kvm`

Using the Command-line

You should use the command-line if:

- You are installing a more complex or large-scale cloud.
- You need to use availability zones or the server groups functionality of the cloud model. For more information, see the *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 6 “Input Model”*.

- You want to customize the cloud configuration beyond the tuned defaults that SUSE provides out of the box.
- You need deeper customizations than are possible to express using the GUI.

Instructions for installing via the command-line are here:

- *Chapter 13, Installing Mid-scale and Entry-scale KVM*

9 Installing with the Install UI

SUSE OpenStack Cloud comes with a GUI-based installation wizard for first-time cloud installations. It will guide you through the configuration process and deploy your cloud based on the custom configuration you provide. The Install UI will start with a set of example cloud configurations for you to choose from. Based on your cloud choice, you can refine your configuration to match your needs using Install UI widgets. You can also directly edit your model configuration files.

When you are satisfied with your configuration and the Install UI has validated your configuration successfully, you can then deploy the cloud into your environment. Deploying the cloud will version-control your configuration into a git repository and provide you with live progress of your deployment.

With the Install UI, you have the option of provisioning SLES12-SP3 to IPMI-capable machines described in your configuration files. Provisioning machines with the Install UI will also properly configure them for Ansible access.

The Install UI is designed to make the initial installation process simpler, more accurate, and faster than manual installation.

9.1 Before You Start

1. Review the *Chapter 2, Pre-Installation Checklist* about recommended pre-installation tasks.

2. Prepare the Cloud Lifecycle Manager node. The Cloud Lifecycle Manager must be accessible either directly or via `ssh`, and have SUSE Linux Enterprise Server 12 SP3 installed. All nodes must be accessible to the Cloud Lifecycle Manager. If the nodes do not have direct access to online Cloud subscription channels, the Cloud Lifecycle Manager node will need to host the Cloud repositories.

    a. If you followed the installation instructions for Cloud Lifecycle Manager (see *Chapter 3, Installing the Cloud Lifecycle Manager server* SUSE OpenStack Cloud should already be installed. Double-check whether SUSE Linux Enterprise and SUSE OpenStack Cloud are properly registered at the SUSE Customer Center by starting YaST and running *Software* › *Product Registration*.

If you have not yet installed SUSE OpenStack Cloud, do so by starting YaST and running *Software* › *Product Registration* › *Select Extensions*. Choose *SUSE OpenStack Cloud* and follow the on-screen instructions. Make sure to register SUSE OpenStack Cloud during the installation process and to install the software pattern `patterns-cloud-ardana`.

b. Ensure the SUSE OpenStack Cloud media repositories and updates repositories are made available to all nodes in your deployment. This can be accomplished either by configuring the Cloud Lifecycle Manager server as an SMT mirror as described in *Chapter 4, Installing and Setting Up an SMT Server on the Cloud Lifecycle Manager server (Optional)* or by syncing or mounting the Cloud and updates repositories to the Cloud Lifecycle Manager server as described in *Chapter 5, Software Repository Setup*.

c. Configure passwordless **`sudo`** for the user created when setting up the node (as described in *Section 3.5, "Creating a User"*). Note that this is *not* the user `ardana` that will be used later in this procedure. In the following we assume you named the user `cloud`. Run the command **`visudo`** as user `root` and add the following line to the end of the file:

```
cloud ALL = (root) NOPASSWD:ALL
```

Make sure to replace *`cloud`* with your user name choice.

d. Set the password for the user `ardana`:

```
sudo passwd ardana
```

e. Become the user `ardana`:

```
su - ardana
```

f. Place a copy of the SUSE Linux Enterprise Server 12 SP3 `.iso` in the `ardana` home directory, `var/lib/ardana`, and rename it to `sles12sp3.iso`.

9.2 Preparing to Run the Install UI

Before you launch the Install UI to install your cloud, do the following:

1. Gather the following details from the servers that will make up your cloud:
   - Server names
   - IP addresses
   - Server roles
   - PXE MAC addresses
   - PXE IP addresses
   - PXE interfaces
   - IPMI IP address, username, password
2. Choose an input model from *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations"*. No action other than an understanding of your needs is necessary at this point. In the Install UI you will indicate which input model you wish to deploy.
3. Before you use the Install UI to install your cloud, you may install the operating system, SLES, on your nodes (servers) if you choose. Otherwise, the Install UI will install it for you. If you are installing the operating system on all nodes yourself, you must do so using the SLES image included in the SUSE OpenStack Cloud 8 package.

In SUSE OpenStack Cloud 8, a local git repository is used to track configuration changes; the Configuration Processor (CP) uses this repository. Use of a git workflow means that your configuration history is maintained, making rollbacks easier and keeping a record of previous configuration settings. The git repository also provides a way for you to merge changes that you pull down as "upstream" updates (that is, updates from SUSE). It also allows you to manage your own configuration changes.

The git repository is installed by the Cloud Lifecycle Manager on the Cloud Lifecycle Manager node.

Using the Install UI does not require the use of the git repository. After the installation, it may be useful to know more about *Chapter 12, Using Git for Configuration Management*.

9.3 Optional: Creating a CSV File to Import Server Data

Before beginning the installation, you can create a CSV file with your server information to import directly into the Install UI to avoid entering it manually on the *Assign Servers* page.

The following table shows the fields needed for your CSV file.

Field	Required	Required for OS Provisioning	Aliases
Server ID	Yes	Yes	server_id, id
IP Address	Yes	Yes	ip, ip_address, ip_addr
MAC Address	Yes	Yes	mac, mac_address, mac_ad-dr
IPMI IP Address	No	Yes	ipmi_ip, ipmi_ip_address
IPMI User	No	Yes	ipmi_user, user
IPMI Password	No	Yes	ipmi_password, password
Server Role	No	No	server_role, role
Server Group	No	No	server_group, group
NIC Mapping	No	No	server_nic_map, nic_map, nic_mapping

The aliases are all the valid names that can be used in the CSV file for the column header for a given field. Field names are not case sensitive. You can use either ` ` (space) or `-` (hyphen) in place of underscore for a field name.

An example CSV file could be:

```
id,ip-addr,mac-address,server-group,nic-mapping,server-role,ipmi-ip,ipmi-user
controller1,192.168.110.3,b2:72:8d:ac:7c:6f,RACK1,HP-DL360-4PORT,CONTROLLER-ROLE,192.168.109.3,admin
myserver4,10.2.10.24,00:14:22:01:23:44,AZ1,,,,
```

9.4 Optional: Importing Certificates for SUSE Manager and HPE OneView

If you intend to use SUSE Manager or HPE OneView to add servers, certificates for those services must be accessible to the Install UI.

Use the following steps to import a SUSE Manager certificate.

1. Retrieve the `.pem` file from the SUSE Manager.

   ```
   curl -k https://SUSE_MANAGER_IP:PORT/pub/RHN-ORG-TRUSTED-SSL-CERT > PEM_NAME.pem
   ```

2. Copy the `.pem` file to the proper location on the Cloud Lifecycle Manager.

   ```
   cd /etc/pki/trust/anchors
   sudo cp ~/PEM_NAME.pem .
   ```

3. Install the certificate.

   ```
   sudo update-ca-certificates
   ```

4. Add *`SUSE Manager host IP address`* if *`SUSE Manager.test.domain`* is not reachable by DNS.

   ```
   sudo vi /etc/hosts
   ```

   Add *`SUSE Manager host IP address`* *`SUSE Manager.test.domain`*. For example:

   ```
   10.10.10.10 SUSE Manager.test.domain
   ```

Use the following steps to import an HPE OneView certificate.

1. Retrieve the `sessionID`.

   ```
   curl -k -H "X-Api-Version:500" -H "Content-Type: application/json" \
   -d '{"userName":ONEVIEW_USER, "password":ONEVIEW_PASSWORD, \
   "loginMsgAck":"true"}' https://ONEVIEW_MANAGER_URL:PORT/rest/login-sessions
   ```

   The response will be similar to:

   ```
   {"partnerData":{},"sessionID":"LTYxNjA1O1NjkxMHcI1b2ypaGPscErUOHrl7At3-odHPmR"}
   ```

2. Retrieve a Certificate Signing Request (CSR) using the *`sessionID`* from *Step 1*.

```
curl -k -i -H "X-Api-Version:500" -H sessionID \
ONEVIEW_MANAGER_URL/rest/certificates/ca \
> CA_NAME.csr
```

3. Follow instructions in the HPE OneView User Guide to validate the CSR and obtain a signed certificate (`CA_NAME .crt`).

4. Copy the `.crt` file to the proper location on the Cloud Lifecycle Manager.

```
cd /etc/pki/trust/anchors
sudo cp ~/data/CA_NAME.crt .
```

5. Install the certificate.

```
sudo update-ca-certificates
```

6. Follow instructions in your HPE OneView User Guide to import the `CA_NAME`.crt certificate into HPE OneView.

7. Add `HPE OneView host IP address` if `HPE OneView.test.domain` is not reachable by DNS.

```
sudo vi /etc/hosts
```

Add `HPE OneView host IP address` `HPE OneView.test.domain` For example:

```
10.84.84.84  HPE OneView.test.domain
```

9.5 Running the Install UI

Important

The Install UI must run continuously without stopping for authentication at any step. When using the Install UI it is required to launch the Cloud Lifecycle Manager with the following command:

```
ARDANA_INIT_AUTO=1 /usr/bin/ardana-init
```

Deploying the cloud to your servers will reconfigure networking and firewall rules on your cloud servers. To avoid problems with these networking changes when using the Install UI, we recommend you run a browser directly on your Cloud Lifecycle Manager node and point it to `http://localhost:3000`.

If you cannot run a browser on the Cloud Lifecycle Manager node to perform the install, you can run a browser from a Linux-based computer in your **MANAGEMENT** network. However, firewall rules applied during cloud deployment will block access to the Install UI. To avoid blocking the connection, you can use the Install UI via an SSH tunnel to the Cloud Lifecycle Manager server. This will allow SSH connections through the **MANAGEMENT** network when you reach the "Review Configuration Files" step of the install process.

To open an SSH tunnel from your Linux-based computer in your **MANAGEMENT** network to the Cloud Lifecycle Manager:

1. Open a new terminal and enter the following command:

   ```
   ssh -N -L 8080:localhost:3000 ardana@MANAGEMENT IP address of Cloud Lifecycle
    Manager
   ```

   The user name and password should be what was set in *Section 3.3, "Installing the SUSE OpenStack Cloud Extension"*. There will be no prompt after you have logged in.

2. Leave this terminal session open to keep the SSH tunnel open and running. This SSH tunnel will forward connections from your Linux-based computer directly to the Cloud Lifecycle Manager, bypassing firewall restrictions.

3. On your local computer (the one you are tunneling from), point your browser to `http://localhost:8080`.

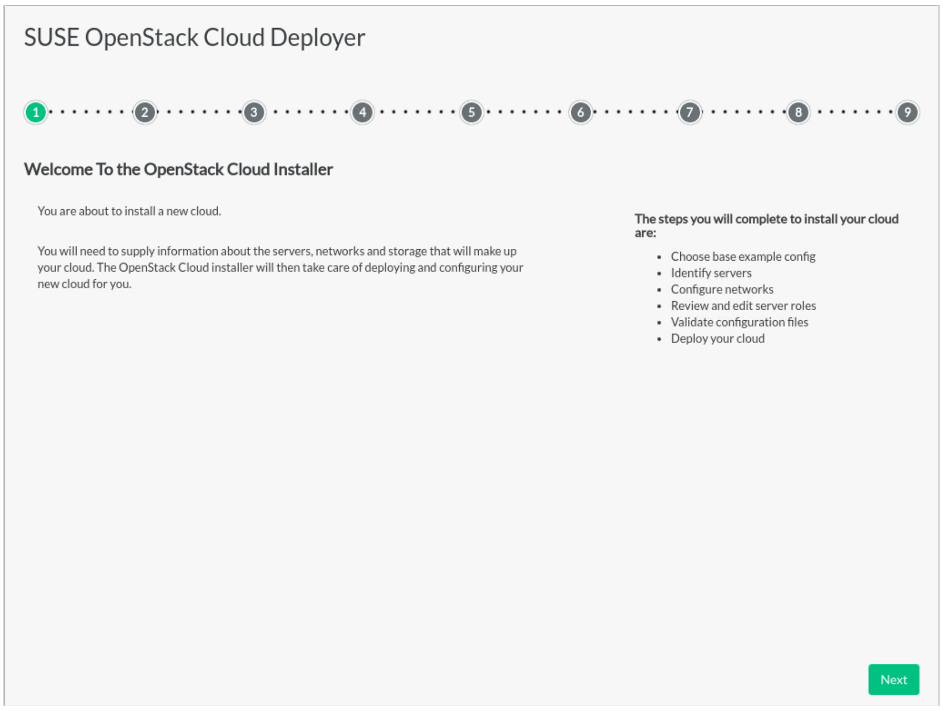
4. If the connection is interrupted, refresh your browser.

Important

If you use an SSH tunnel to connect to the Install UI, there is an important note in the "Review Configuration Files" step about modifying `firewall_rules.yml` to allow SSH connections on the **MANAGEMENT** network.

Overview

The first page of the Install UI shows the general installation process and a reminder to gather some information before beginning. Clicking the *Next* button brings up the *Model Selection* page.



Welcome To the OpenStack Cloud Installer

You are about to install a new cloud.

You will need to supply information about the servers, networks and storage that will make up your cloud. The OpenStack Cloud installer will then take care of deploying and configuring your new cloud for you.

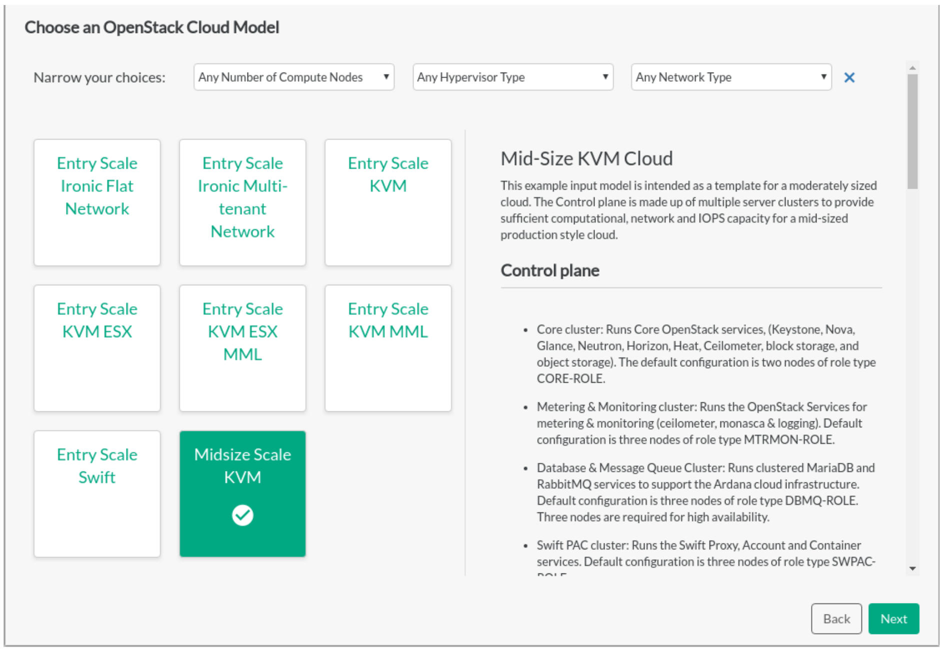
The steps you will complete to install your cloud are:

- Choose base example config
- Identify servers
- Configure networks
- Review and edit server roles
- Validate configuration files
- Deploy your cloud

Choose an OpenStack Cloud Model

The input model choices are displayed on this page. Details of each model can be seen on the right by clicking the model name on the left. If you have already decided some aspects of your cloud environment, models can be filtered using the dropdown selections. Narrowing a parameter affects the range of choices of models and changes other dropdown choices to only those that are compatible.

Selecting a model will determine the base template from which the cloud will be deployed. Models can be adjusted later in the process, though selecting the closest match to your requirements reduces the effort required to deploy your cloud.



Choose an OpenStack Cloud Model

Narrow your choices: Any Number of Compute Nodes · Any Hypervisor Type · Any Network Type

Mid-Size KVM Cloud

This example input model is intended as a template for a moderately sized cloud. The Control plane is made up of multiple server clusters to provide sufficient computational, network and IOPS capacity for a mid-sized production style cloud.

Control plane

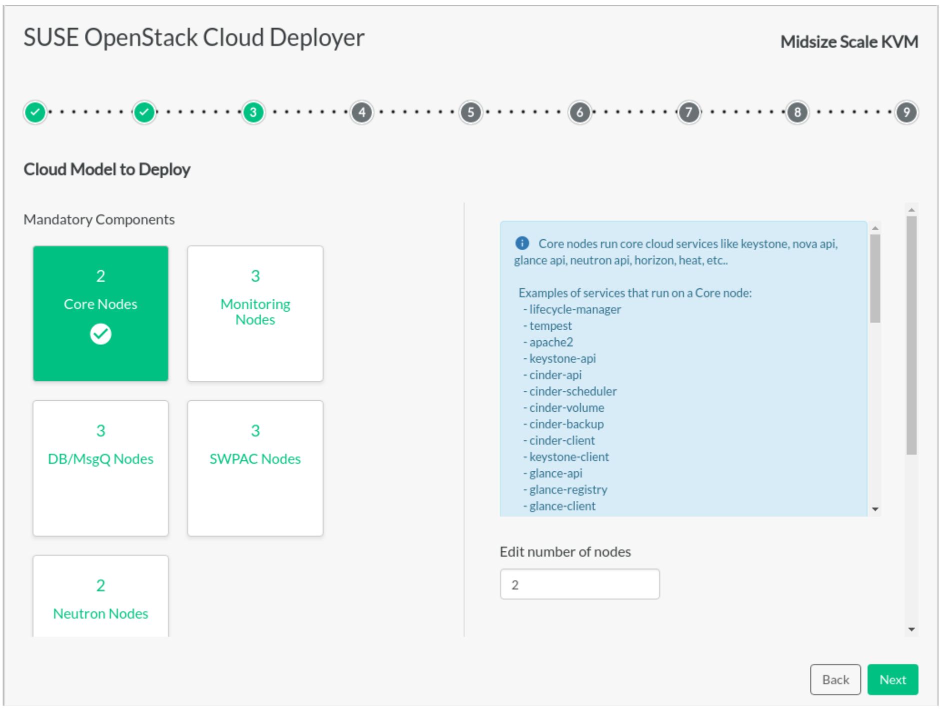
- Core cluster: Runs Core OpenStack services, (Keystone, Nova, Glance, Neutron, Horizon, Heat, Ceilometer, block storage, and object storage). The default configuration is two nodes of role type CORE-ROLE.
- Metering & Monitoring cluster: Runs the OpenStack Services for metering & monitoring (ceilometer, monasca & logging). Default configuration is three nodes of role type MTRMON-ROLE.
- Database & Message Queue Cluster: Runs clustered MariaDB and RabbitMQ services to support the Ardana cloud infrastructure. Default configuration is three nodes of role type DBMQ-ROLE. Three nodes are required for high availability.
- Swift PAC cluster: Runs the Swift Proxy, Account and Container services. Default configuration is three nodes of role type SWPAC-

Cloud Model to Deploy

Based on the cloud example selected on the previous page, more detail is shown about that cloud configuration and the components that will be deployed. If you go back and select a different model, the deployment process restarts from the beginning. Any configuration changes you have made will be deleted.

- *Mandatory components* have assigned quantities. We strongly suggest not changing those quantities to avoid potential problems later in the installation process.
- *Additional components* can be adjusted within the parameters shown.

The number of nodes (servers) dedicated to each server role can be adjusted. Most input models are designed to support High Availability and to distribute OpenStack services appropriately.


SUSE OpenStack Cloud Deployer

Midsize Scale KVM

Cloud Model to Deploy

Mandatory Components

2 Core Nodes ✓

3 Monitoring Nodes

3 DB/MsgQ Nodes

3 SWPAC Nodes

2 Neutron Nodes

Core nodes run core cloud services like keystone, nova api, glance api, neutron api, horizon, heat, etc..

Examples of services that run on a Core node:
- lifecycle-manager
- tempest
- apache2
- keystone-api
- cinder-api
- cinder-scheduler
- cinder-volume
- cinder-backup
- cinder-client
- keystone-client
- glance-api
- glance-registry
- glance-client

Edit number of nodes: 2

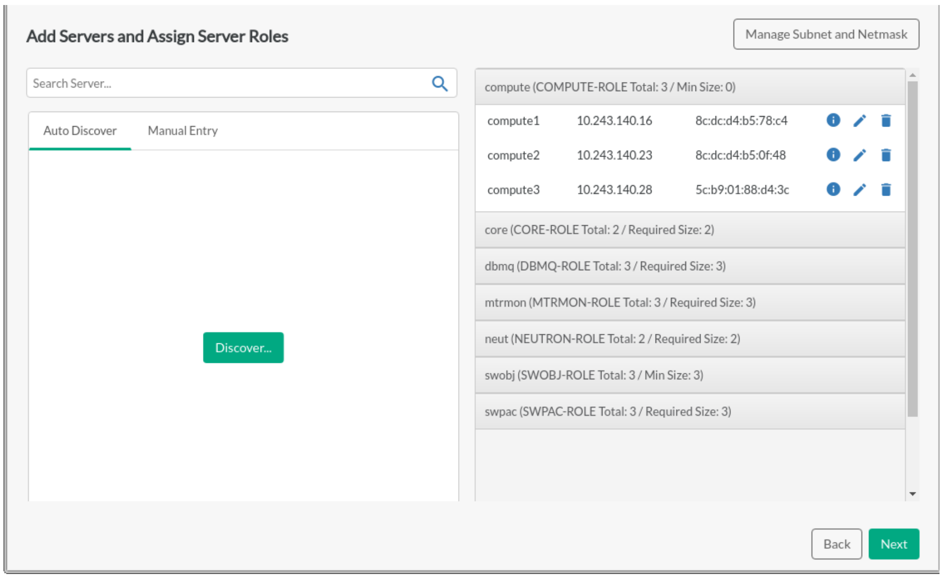
Adding Servers and Assigning Server Roles

This page provides more detail about the number and assignment of each type of node based on the information from the previous page (any changes must be made there).

Components that do not meet the required parameters will be shown in red in the accordion bar. Missing required fields and duplicate server names will also be red, as will the accordion bar. The *Next* button will be disabled.

Servers may be discovered using SUSE Manager, HPE OneView, or both. Ensure that the certificates are accessible, as described in *Section 9.4, "Optional: Importing Certificates for SUSE Manager and HPE OneView"*. Clicking the *Discover* button will prompt for access credentials to the system management software to be used for discovery. Certificates can be verified by checking *Verify SSL certificate*. After validating credentials, Discovery will retrieve a list of known servers from SUSE Manager and/or HPE OneView and allow access to server details on those management platforms.

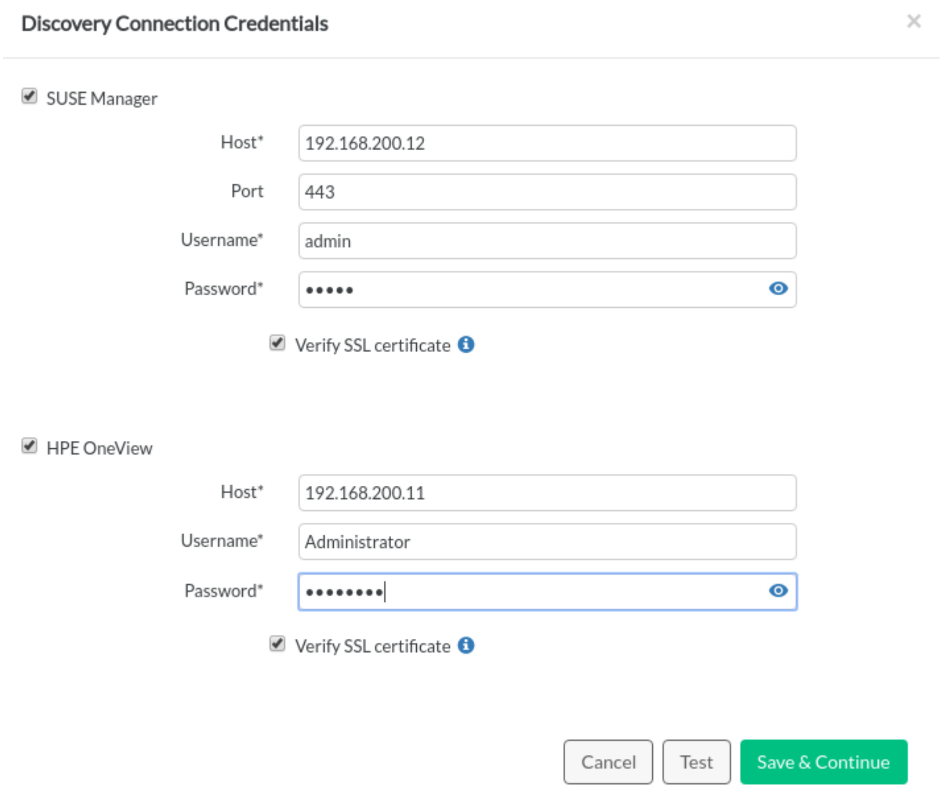
You can drag and drop to move servers from the left to the right in order to assign server roles, from right to left, or up and down in the accordion bar.


Add Servers and Assign Server Roles

compute (COMPUTE-ROLE Total: 3 / Min Size: 0)

compute1	10.243.140.16	8c:dc:d4:b5:78:c4
compute2	10.243.140.23	8c:dc:d4:b5:0f:48
compute3	10.243.140.28	5c:b9:01:88:d4:3c

core (CORE-ROLE Total: 2 / Required Size: 2)
dbmq (DBMQ-ROLE Total: 3 / Required Size: 3)
mtrmon (MTRMON-ROLE Total: 3 / Required Size: 3)
neut (NEUTRON-ROLE Total: 2 / Required Size: 2)
swobj (SWOBJ-ROLE Total: 3 / Min Size: 3)
swpac (SWPAC-ROLE Total: 3 / Required Size: 3)


Discovery Connection Credentials

SUSE Manager [✓]

Host*	192.168.200.12
Port	443
Username*	admin
Password*	•••••

Verify SSL certificate [✓]

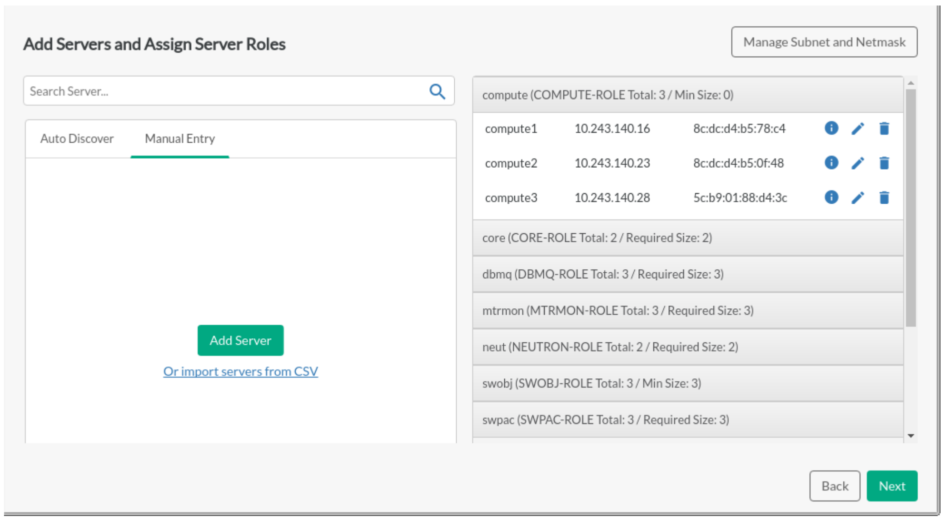
HPE OneView [✓]

Host*	192.168.200.11
Username*	Administrator
Password*	••••••••

Verify SSL certificate [✓]

Server information may also be entered manually or imported via CSV in the *Manual Entry* tab. The format for CSV entry is described in *Section 9.3, “Optional: Creating a CSV File to Import Server Data”*. The server assignment list includes placeholder server details that can be edited to reflect real hardware, or can be removed and replaced with discovered or manually added systems.

For more information about server roles, see *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 6 “Input Model”, Section 6.2 “Concepts”, Section 6.2.4 “Server Roles”*.


Add Servers and Assign Server Roles

Manage Subnet and Netmask

Search Server...

Auto Discover | Manual Entry

Add Server

Or import servers from CSV

compute (COMPUTE-ROLE Total: 3 / Min Size: 0)

compute1	10.243.140.16	8c:dc:d4:b5:78:c4
compute2	10.243.140.23	8c:dc:d4:b5:0f:48
compute3	10.243.140.28	5c:b9:01:88:d4:3c

core (CORE-ROLE Total: 2 / Required Size: 2)

dbmq (DBMQ-ROLE Total: 3 / Required Size: 3)

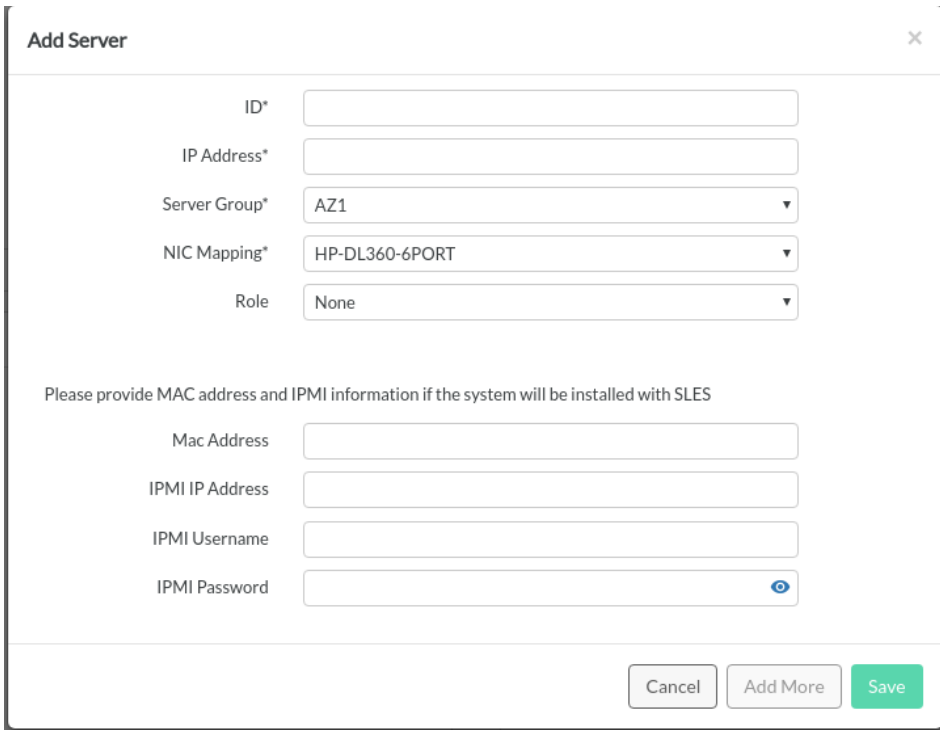
mtrmon (MTRMON-ROLE Total: 3 / Required Size: 3)

neut (NEUTRON-ROLE Total: 2 / Required Size: 2)

swobj (SWOBJ-ROLE Total: 3 / Min Size: 3)

swpac (SWPAC-ROLE Total: 3 / Required Size: 3)

Back | Next


Add Server

ID*	
IP Address*	
Server Group*	AZ1
NIC Mapping*	HP-DL360-6PORT
Role	None

Please provide MAC address and IPMI information if the system will be installed with SLES

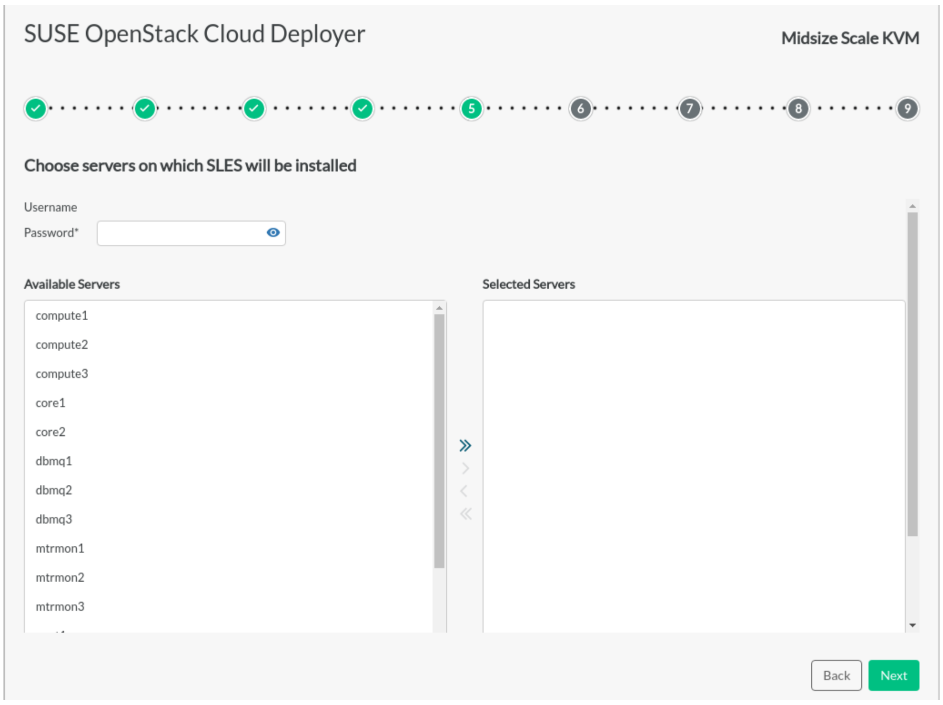
Mac Address	
IPMI IP Address	
IPMI Username	
IPMI Password	

Cancel | Add More | Save

Subnet and netmask values should be set on this page as they may impact the IP addresses being assigned to various servers.

Choose servers on which SLES will be installed

If an OS has not previously been installed on the servers that make up the cloud configuration, the OS installation page allows for Cobbler to deploy SLES on servers in the cloud configuration. Enter password, select servers and click *Install* to deploy SLES to these servers. An installation log and progress indicators will be displayed.



Choose servers on which SLES will be installed

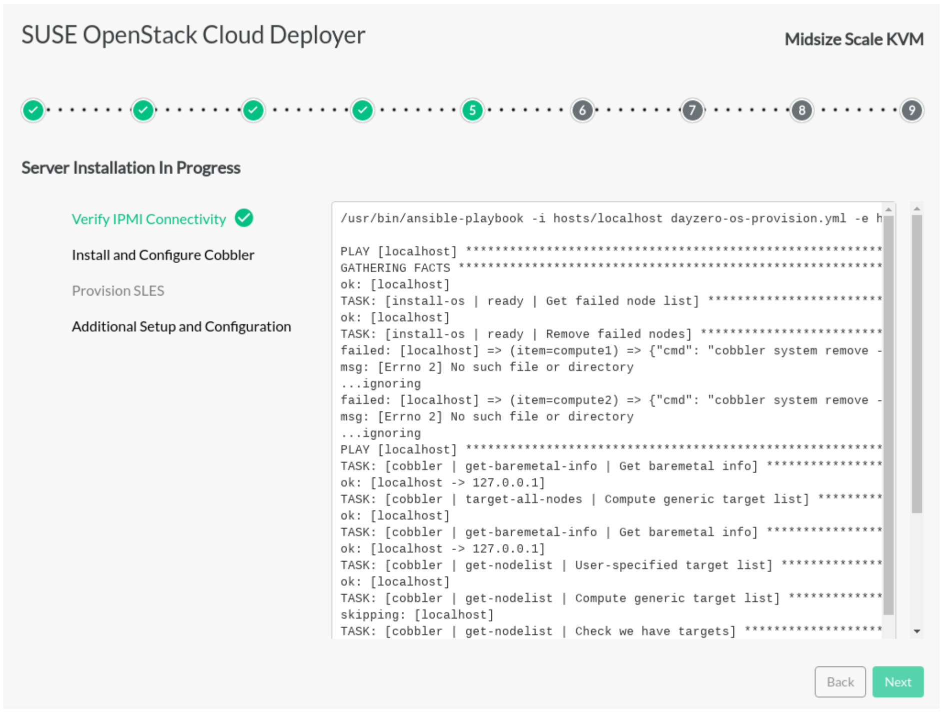
Username

Password*

Available Servers

compute1
compute2
compute3
core1
core2
dbmq1
dbmq2
dbmq3
mtrmon1
mtrmon2
mtrmon3

Selected Servers


SUSE OpenStack Cloud Deployer
Midsize Scale KVM
5
6
7
8
9
Server Installation In Progress
Verify IPMI Connectivity
Install and Configure Cobbler
Provision SLES
Additional Setup and Configuration
Back
Next

```
/usr/bin/ansible-playbook -i hosts/localhost dayzero-os-provision.yml -e h

PLAY [localhost] ********************************************************
GATHERING FACTS *********************************************************
ok: [localhost]
TASK: [install-os | ready | Get failed node list] ***********************
ok: [localhost]
TASK: [install-os | ready | Remove failed nodes] ************************
failed: [localhost] => (item=compute1) => {"cmd": "cobbler system remove -
msg: [Errno 2] No such file or directory
...ignoring
failed: [localhost] => (item=compute2) => {"cmd": "cobbler system remove -
msg: [Errno 2] No such file or directory
...ignoring
PLAY [localhost] ********************************************************
TASK: [cobbler | get-baremetal-info | Get baremetal info] ***************
ok: [localhost -> 127.0.0.1]
TASK: [cobbler | target-all-nodes | Compute generic target list] *********
ok: [localhost]
TASK: [cobbler | get-baremetal-info | Get baremetal info] ***************
ok: [localhost -> 127.0.0.1]
TASK: [cobbler | get-nodelist | User-specified target list] *************
ok: [localhost]
TASK: [cobbler | get-nodelist | Compute generic target list] *************
skipping: [localhost]
TASK: [cobbler | get-nodelist | Check we have targets] ******************
```

Server and Role Summary

When the OS installation is complete, a Server and Server Role Summary page is displayed. It shows which servers have been assigned to each role, and provides an opportunity to edit the server configurations. Various cloud components can be configured by clicking on the *Manage Cloud Settings* button. Incorrect information will be shown in red.

Below is the list of what can be changed within the Install UI, followed by a list of customizations that can only be changed by directly editing the files on the *Review Configuration Files* page. Anything changed directly in the files themselves during the Install UI process will be overwritten by values you have entered with the Install UI.

Changes to the following items can be made:

- servers (including SLES installation configuration)
- networks
- disk models
- interface models
- NIC mappings
- NTP servers

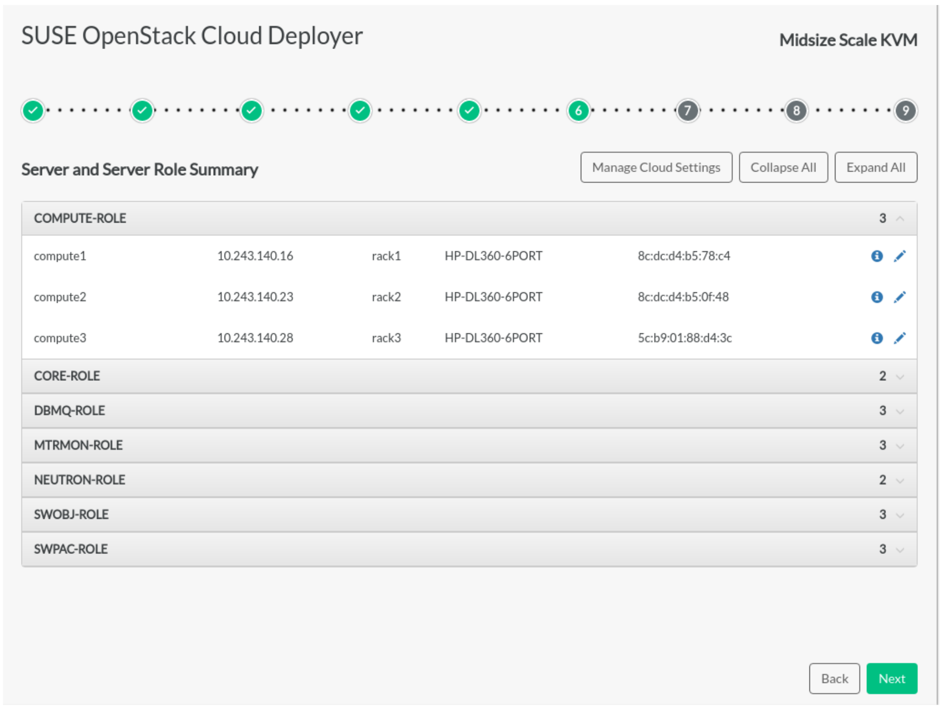
- name servers
- tags in network groups

Changes to the following items can only be made by manually editing the associated `.yml` files on the *Review Configuration* page of the Install UI:

- server groups
- server roles
- network groups
- firewall rules
- DNS, SMTP, firewall settings (`cloudConfig.yml`)
- control planes

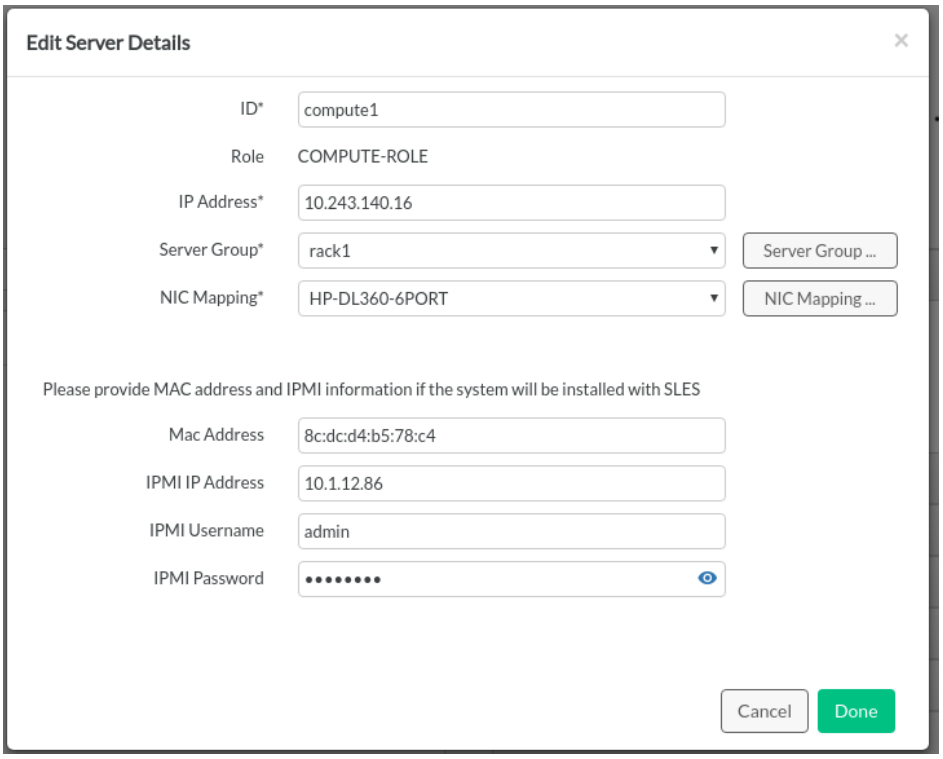
Important

Directly changing files may cause the configuration to fail validation. During the process of installing with the Install UI, any changes should be made with the tools provided within the Install UI.



Server and Server Role Summary

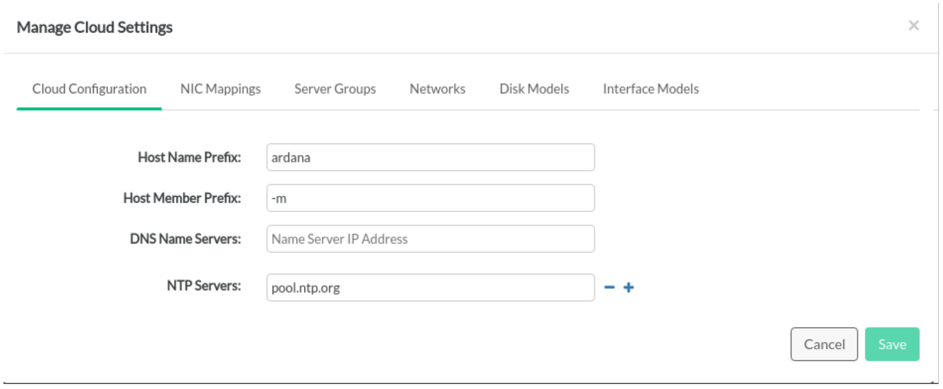
COMPUTE-ROLE				3
compute1	10.243.140.16	rack1	HP-DL360-6PORT	8c:dc:d4:b5:78:c4
compute2	10.243.140.23	rack2	HP-DL360-6PORT	8c:dc:d4:b5:0f:48
compute3	10.243.140.28	rack3	HP-DL360-6PORT	5c:b9:01:88:d4:3c
CORE-ROLE				2
DBMQ-ROLE				3
MTRMON-ROLE				3
NEUTRON-ROLE				2
SWOBJ-ROLE				3
SWPAC-ROLE				3


Edit Server Details

ID*	compute1	
Role	COMPUTE-ROLE	
IP Address*	10.243.140.16	
Server Group*	rack1	Server Group ...
NIC Mapping*	HP-DL360-6PORT	NIC Mapping ...

Please provide MAC address and IPMI information if the system will be installed with SLES

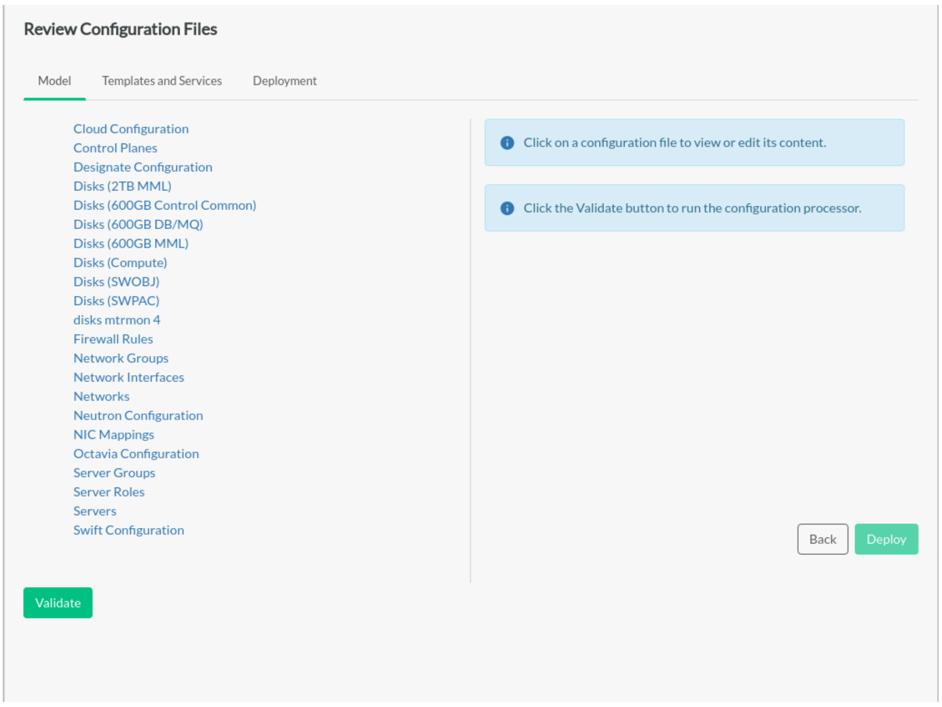
Mac Address	8c:dc:d4:b5:78:c4
IPMI IP Address	10.1.12.86
IPMI Username	admin
IPMI Password	••••••••


Manage Cloud Settings

Host Name Prefix:	ardana
Host Member Prefix:	-m
DNS Name Servers:	Name Server IP Address
NTP Servers:	pool.ntp.org

Review Configuration Files

Advanced editing of the cloud configuration can be done on the `Review Configuration Files` page. Individual `.yml` and `.j2` files can be edited directly with the embedded editor in the *Model* and *Templates and Services* tabs. The *Deployment* tab contains the items *Wipe Data Disks*, *Encryption Key* and *Verbosity Level*.


Review Configuration Files

Cloud Configuration
Control Planes
Designate Configuration
Disks (2TB MML)
Disks (600GB Control Common)
Disks (600GB DB/MQ)
Disks (600GB MML)
Disks (Compute)
Disks (SWOBJ)
Disks (SWPAC)
disks mtrmon 4
Firewall Rules
Network Groups
Network Interfaces
Networks
Neutron Configuration
NIC Mappings
Octavia Configuration
Server Groups
Server Roles
Servers
Swift Configuration

Click on a configuration file to view or edit its content.

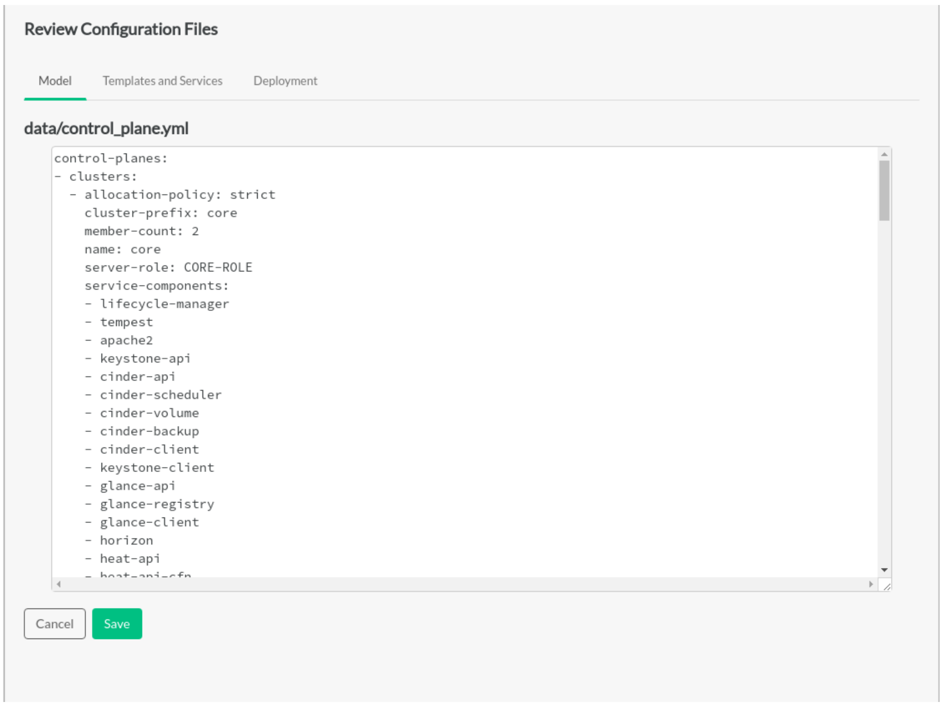
Click the Validate button to run the configuration processor.

Important

If you are using an SSH tunnel to connect to the Install UI, you will need to make an extra modification here to allow SSH connections through the firewall:

1. While on the Review Configuration Files page, click on the *Model* tab.
2. Click on *Firewall Rules*.
3. Uncomment the `SSH` section (remove the `#` at the beginning of the line for the `- name: SSH` section).
4. If you do not have such a `- name: SSH` section, manually add the following under the `firewall-rules:` section:

```
name: SSH
network-groups:
- MANAGEMENT
rules:
- type: allow
  remote-ip-prefix: 0.0.0.0/0
  port-range-min: 22
  port-range-max: 22
  protocol: tcp
```

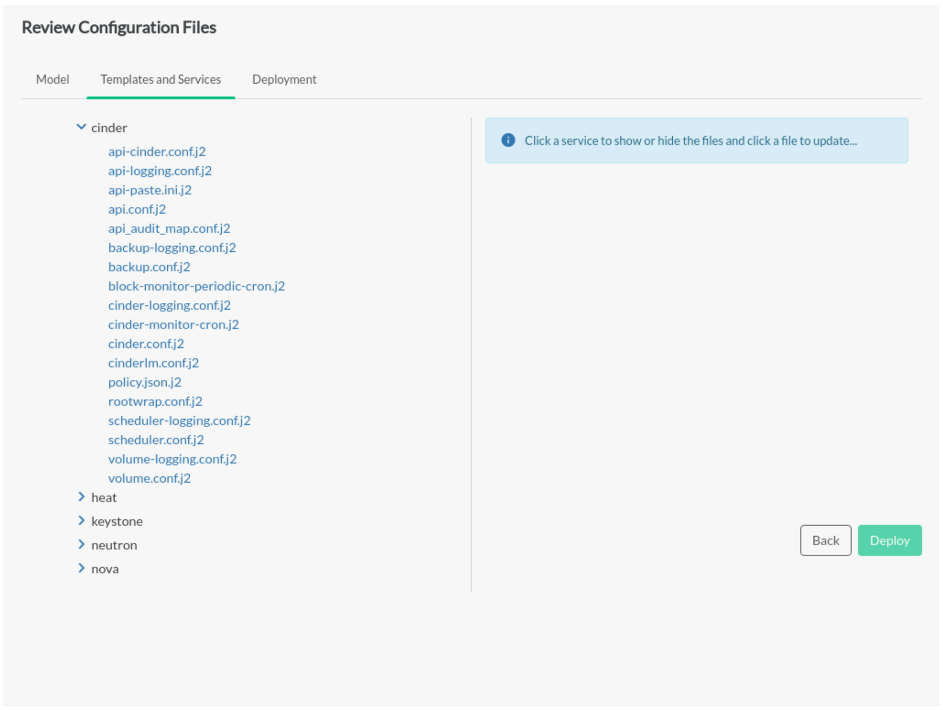


Review Configuration Files

Model | Templates and Services | Deployment

data/control_plane.yml

```
control-planes:
- clusters:
  - allocation-policy: strict
    cluster-prefix: core
    member-count: 2
    name: core
    server-role: CORE-ROLE
    service-components:
    - lifecycle-manager
    - tempest
    - apache2
    - keystone-api
    - cinder-api
    - cinder-scheduler
    - cinder-volume
    - cinder-backup
    - cinder-client
    - keystone-client
    - glance-api
    - glance-registry
    - glance-client
    - horizon
    - heat-api
```

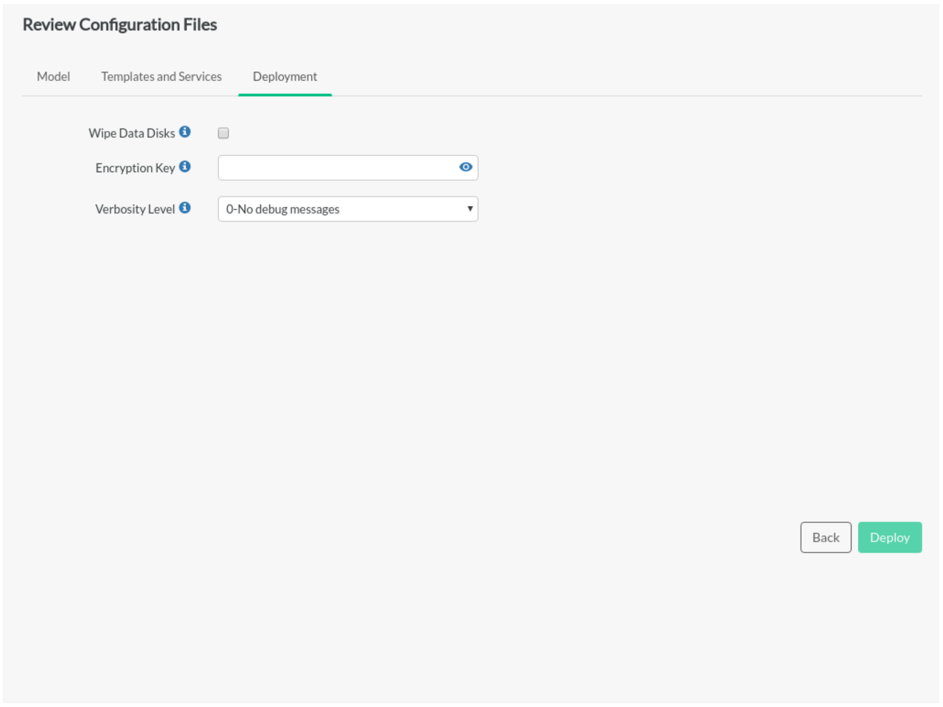


Review Configuration Files

Model | Templates and Services | Deployment

- cinder
  - api-cinder.conf.j2
  - api-logging.conf.j2
  - api-paste.ini.j2
  - api.conf.j2
  - api_audit_map.conf.j2
  - backup-logging.conf.j2
  - backup.conf.j2
  - block-monitor-periodic-cron.j2
  - cinder-logging.conf.j2
  - cinder-monitor-cron.j2
  - cinder.conf.j2
  - cinderlm.conf.j2
  - policy.json.j2
  - rootwrap.conf.j2
  - scheduler-logging.conf.j2
  - scheduler.conf.j2
  - volume-logging.conf.j2
  - volume.conf.j2
- heat
- keystone
- neutron
- nova

Click a service to show or hide the files and click a file to update...



Wipe Data Disks

Encryption Key

Verbosity Level: 0-No debug messages

Important

Manual edits to your configuration files outside of the Install UI may not be reflected in the Install UI. If you make changes to any files directly, refresh the browser to make sure changes are seen by the Install UI.

Before performing the deployment, the configuration must be validated by clicking the *Validate* button below the list of configuration files on the *Model* tab. This ensures the configuration will be successful **before** the actual configuration process runs and possibly fails. The *Validate* button also commits any changes. If there are issues with the validation, the configuration processor will provide detailed information about the causes. When validation completes successfully, a message will be displayed that the model is valid. If either validation or commit fail, the *Next* button is disabled.

Clicking the *Deploy* button starts the actual deployment process.

Cloud Deployment in Progress

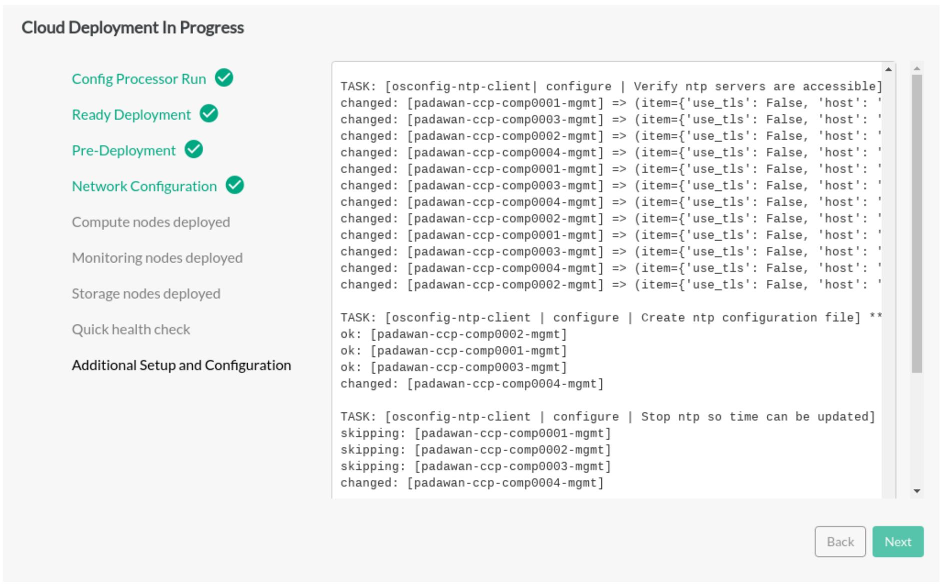
General progress steps are shown on the left. Detailed activity is shown on the right.

To start the deployment process, the Install UI runs scripts and playbooks based on the actual final configuration. Completed operations are green, black means in process, gray items are not started yet.

The log stream on the right shows finished states. If there are any failures, the log stream will show the errors and the *Next* button will be disabled. The *Back* and *Next* buttons are disabled during the deployment process.

The log files in `~/ardana/.ansible/ansible.log` and `/var/cache/ardana_installer/` have debugging information.

- `/var/cache/ardana_installer/log/ardana-service/ardana-service.log` is created and used during the deployment step.
- Each of the time-stamped files in `/var/cache/ardana_installer/log/ardana-service/logs/*.log` shows the output of a single Ansible playbook run invoked during the UI installation process and the log output for each of those runs.
- The `~/ardana/.ansible/ansible.log` file is the output of all Ansible playbook runs. This includes the logs from `/var/cache/ardana_installer/log/ardana-service/logs/*.log`.

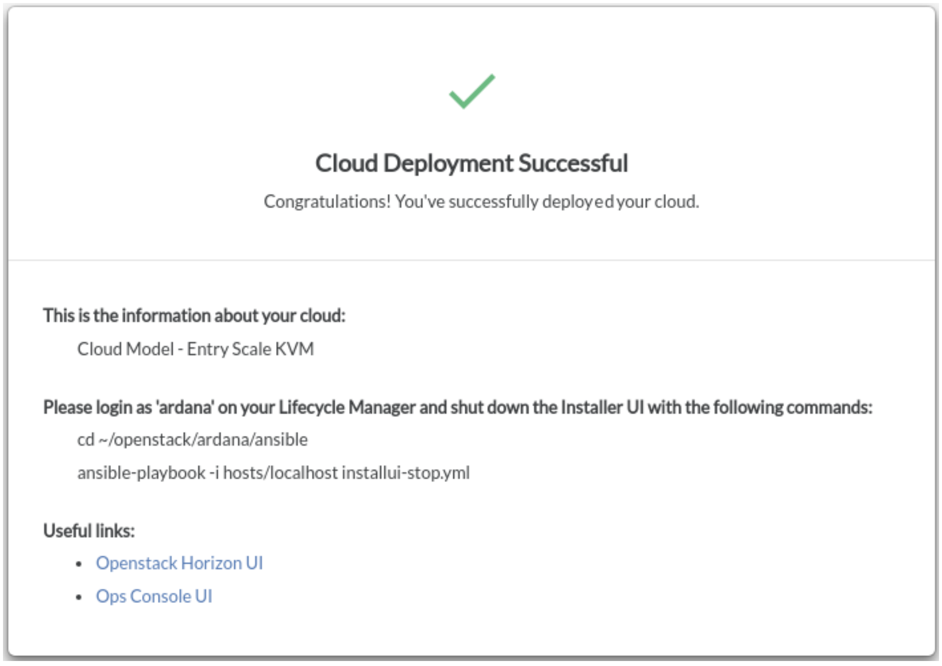


```
TASK: [osconfig-ntp-client| configure | Verify ntp servers are accessible]
changed: [padawan-ccp-comp0001-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0003-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0002-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0004-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0001-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0003-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0004-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0002-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0001-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0003-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0004-mgmt] => (item={'use_tls': False, 'host': '
changed: [padawan-ccp-comp0002-mgmt] => (item={'use_tls': False, 'host': '

TASK: [osconfig-ntp-client | configure | Create ntp configuration file] **
ok: [padawan-ccp-comp0002-mgmt]
ok: [padawan-ccp-comp0001-mgmt]
ok: [padawan-ccp-comp0003-mgmt]
changed: [padawan-ccp-comp0004-mgmt]

TASK: [osconfig-ntp-client | configure | Stop ntp so time can be updated]
skipping: [padawan-ccp-comp0001-mgmt]
skipping: [padawan-ccp-comp0002-mgmt]
skipping: [padawan-ccp-comp0003-mgmt]
changed: [padawan-ccp-comp0004-mgmt]
```

When the deployment process is complete, all items on the left will be green. Some deployments will not include all steps shown if they don't apply to the selected input model. In such a situation, those unneeded steps will remain gray.

The *Next* button will be enabled when deployment is successful.

Clicking *Next* will display the `Cloud Deployment Successful` page with information about the deployment, including the chosen input model and links to cloud management tools.


Cloud Deployment Successful

Congratulations! You've successfully deployed your cloud.

This is the information about your cloud:

Cloud Model - Entry Scale KVM

Please login as 'ardana' on your Lifecycle Manager and shut down the Installer UI with the following commands:

cd ~/openstack/ardana/ansible

ansible-playbook -i hosts/localhost installui-stop.yml

Useful links:

- Openstack Horizon UI
- Ops Console UI

After installation is complete, shutdown the Install UI by logging into the Cloud Lifecycle Manager and running the following commands:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost installui-stop.yml
```

After deployment, continue to *Chapter 22, Cloud Verification* and *Chapter 28, Other Common Post-Installation Tasks*.

To understand cloud configuration more thoroughly and to learn how to make changes later, see:

- *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 6 "Input Model"*
- *Chapter 12, Using Git for Configuration Management*

10 DNS Service Installation Overview

The SUSE OpenStack Cloud DNS Service supports several different backends for domain name service. The choice of backend must be included in the deployment model before the SUSE OpenStack Cloud install is completed.

Warning

By default any user in the project is allowed to manage a DNS domain. This can be changed by updating the Policy.json file for Designate.

The backends that are available within the DNS Service are separated into two categories, self-contained and external.

TABLE 10.1: DNS BACKENDS

Category	Backend	Description	Recommended For
Self-contained	PowerDNS 3.4.1, BIND 9.9.5	All components necessary will be installed and configured as part of the SUSE OpenStack Cloud install.	POCs and customers who wish to keep cloud and traditional DNS separated.
External	InfoBlox	The authoritative DNS server itself is external to SUSE OpenStack Cloud. Management and configuration is out of scope for the Cloud Lifecycle Manager but remains the responsibility of the customer.	Customers who wish to integrate with their existing DNS infrastructure.

10.1 Installing the DNS Service with BIND

SUSE OpenStack Cloud DNS Service defaults to the BIND back-end if another back-end is not configured for domain name service. BIND will be deployed to one or more control planes clusters. The following configuration example shows how the BIND service is installed.

10.1.1 Configuring the Back-end

Ensure the DNS Service components and the BIND component have been placed on a cluster. BIND can be placed on a cluster separate from the other DNS service components.

```
control-planes:
          - name: control-plane-1
          region-name: region1

          clusters:
          - name: cluster1
          service-components:
          - lifecycle-manager
          - mariadb
          - ip-cluster
          - apache2
          - ...
          - designate-api
          - designate-central
          - designate-pool-manager
          - designate-zone-manager
          - designate-mdns
          - designate-client
          - bind
```

Updating the Input Model

When the back-end is configured, add `bind-ext` to the file `network_groups.yml`.

1. Edit `~/openstack/my_cloud/definition/data/network_groups.yml` to add `bind-ext` to component-endpoints.

   ```
   name: EXTERNAL-API
   hostname-suffix: extapi
   component-endpoints:
   - bind-ext
   ```

2. Save the file.

10.2 Install the DNS Service with PowerDNS

10.2.1 Installing DNS Service with PowerDNS

SUSE OpenStack Cloud DNS Service and **PowerDNS** can be installed together instead of the default **BIND** backend. PowerDNS will be deployed to one or more control planes clusters. The following configuration example shows how the PowerDNS service is installed.

10.2.2 Configure the Backend

To configure the backend for PowerDNS, follow these steps.

1. Ensure the DNS Service components and the PowerDNS component have been placed on a cluster. PowerDNS may be placed on a separate cluster to the other DNS Service components. Ensure the default **bind** component has been removed.

```
control-planes:
          - name: control-plane-1
          region-name: region1

          clusters:
          - name: cluster1
          service-components:
          - lifecycle-manager
          - mariadb
          - ip-cluster
          - apache2
          - ...
          - designate-api
          - designate-central
          - designate-pool-manager
          - designate-zone-manager
          - designate-mdns
          - designate-client
          - powerdns
```

2. Edit the `~/openstack/my_cloud/definitions/data/network_groups.yml` file to include the powerdns-ext.

```
- name: EXTERNAL-API
```

```
hostname-suffix: extapi
component-endpoints:
 - powerdns-ext
load-balancers:
 - provider: ip-cluster
```

3. Edit the `~/openstack/my_cloud/definitions/data/firewall_rules.yml` to allow UDP/TCP access.

```
     - name: DNSudp
      # network-groups lists the network group names that the rules apply to
      network-groups:
      - EXTERNAL-API
      rules:
      - type: allow
        # range of remote addresses in CIDR format that this rule applies to
        remote-ip-prefix:  0.0.0.0/0
        port-range-min: 53
        port-range-max: 53
        protocol: udp

    - name: DNStcp
      # network-groups lists the network group names that the rules apply to
      network-groups:
      - EXTERNAL-API
      rules:
      - type: allow
        # range of remote addresses in CIDR format that this rule applies to
        remote-ip-prefix:  0.0.0.0/0
        port-range-min: 53
        port-range-max: 53
        protocol: tcp
```

Please see *Book “Operations Guide”, Chapter 9 “Managing Networking”, Section 9.2 “DNS Service Overview”, Section 9.2.2 “Designate Initial Configuration”* for post-installation DNS Service configuration.

10.3 Installing the DNS Service with InfoBlox

SUSE OpenStack Cloud DNS service can be installed with the InfoBlox back-end instead of the default PowerDNS back-end. To use InfoBlox as the back-end, all prerequisites must be satisfied and the configuration of InfoBlox must be complete.

Note

No DNS server will be deployed onto the SUSE OpenStack Cloud nodes. Instead, zones will be hosted on the InfoBlox servers.

Important

Before you enable and use InfoBlox in SUSE OpenStack Cloud, review the *InfoBlox OpenStack Drive Deployment Guide* at https://www.infoblox.com/wp-content/uploads/infoblox-deployment-guide-infoblox-openstack-driver.pdf. Address any questions with your InfoBlox support contact.

10.3.1 Prerequisites

1. An existing InfoBlox system deployed within your organization.
2. IPs and other credentials required to access the InfoBlox WS API.
3. Network connectivity to and from the cluster containing Designate and the InfoBlox WS API.

This information is available from your InfoBlox system administrator.

10.3.2 Configuring the Back-end

If not already present, DNS service components must be placed on a cluster. Additionally, ensure the default `bind` component has been removed, and `powerdns` has not been added. Replace the `designate-mdns` component with the `designate-mdns-external` component.

```
control-planes:
          - name: control-plane-1
            region-name: region1
              - lifecycle-manager-target

          clusters:
            - name: cluster1
              service-components:
              - lifecycle-manager
              - mariadb
```

```
               - ip-cluster
               - apache2
               - ...
               - designate-api
               - designate-central
               - designate-pool-manager
               - designate-zone-manager
               - designate-mdns-external
               - designate-client
```

You will need to provide DNS service information details on your InfoBlox deployment. Open the Designate pool-manager configuration template:

```
$ cd ~/openstack/my_cloud
$ nano config/designate/pool-manager.conf.j2
```

In the `config/designate/pool-manager.conf.j2`, find the following code block:

```
  nameservers:
    - host: <infoblox-server-ip>
      port: <infoblox-port-number>
   ...
     also_notifies:
    - host: <infoblox-server-ip>
      port: <infoblox-port-number>
```

Make the following changes:

1. Uncomment the block for `infoblox` and `also_notifies`. In Jinja2, this means replacing the `{#` and `#}` markers with `{{` and `}}`.
2. Fill in the API URL, user name, password, and all remaining fields.
3. Save the file.

Once complete, the block should look like this:

```
    - type: infoblox
      description: infoblox Cluster

      masters:
{% if DES_PMG.consumes_DES_MDN_EXT is defined %}
{% for mdn_member in DES_PMG.consumes_DES_MDN_EXT.members.private %}
        - host: {{ mdn_member.ip_address }}
```

```
          port: {{ mdn_member.port }}
{% endfor %}
{% endif %}

      options:
        wapi_url: https://<infoblox-server-ip>/wapi/v2.2.2/
        username: ardana-designate
        password: MySecretPassword
        ns_group: designate
        sslverify: False
        dns_view: default
        network_view: default
```

You will need to inspect and commit the changes before proceeding with the deployment:

```
$git diff ardana/ansible/roles/designate-pool-manager/templates/pools.yaml.j2
index 291c6c9..b7fb39c 100644
--- a/ardana/ansible/roles/designate-pool-manager/templates/pools.yaml.j2
+++ b/ardana/ansible/roles/designate-pool-manager/templates/pools.yaml.j2
@@ -28,6 +28,8 @@
       priority: 2

   nameservers:
+    - host: <infoblox-server-ip>
+      port: <infoblox-port-number>
 {% if DES_PMG.consumes_FND_PDN is defined %}
 {% for pdn_member in DES_PMG.consumes_FND_PDN.members.private %}
     - host: {{ pdn_member.ip_address }}
@@ -40,7 +42,9 @@
       port: {{ bnd_member.port }}
 {% endfor %}
 {% endif %}
-#  also_notifies:
+  also_notifies:
+    - host: <infoblox-server-ip>
+      port: <infoblox-port-number>
   targets:
 {% if DES_PMG.consumes_FND_PDN is defined %}
     - type: powerdns
@@ -89,27 +93,27 @@
 {% endfor %}
 {% endif %}

-#
-#    - type: infoblox
-#      description: infoblox Cluster
```

```
-#
-#      masters:
+
+    - type: infoblox
+      description: infoblox Cluster
+
+      masters:
 {% if DES_PMG.consumes_DES_MDN_EXT is defined %}
 {% for mdn_member in DES_PMG.consumes_DES_MDN_EXT.members.private %}
-#        - host: {{ mdn_member.ip_address }}
-#          port: {{ mdn_member.port }}
+        - host: {{ mdn_member.ip_address }}
+          port: {{ mdn_member.port }}
 {% endfor %}
 {% endif %}
-#
-#      options:
-#        wapi_url: https://127.0.0.1/wapi/v2.2.2/
-#        username: admin
-#        password: infoblox
-#        ns_group: designate
-#        sslverify: False
-#        dns_view: default
-#        network_view: default
-#
+
+      options:
+        wapi_url: https://127.0.0.1/wapi/v2.2.2/
+        username: admin
+        password: infoblox
+        ns_group: designate
+        sslverify: False
+        dns_view: default
```

SSL and CA Certificate

To enable SSL Verify, edit the following file:

```
$ cd ~/openstack/my_cloud
$ nano config/designate/pools.yaml.j2
```

In the `infoblox` section, set `sslverify` to `True`:

```
sslverify: True
```

To generate a CA certificate for InfoBlox, follow these steps:

1. Log in to the InfoBlox user interface.
2. Generate a self-signed certificate by selecting: *System* › *Certificate* › *HTTP Cert* › *Generate Self Signed Certificate*
3. Provide the host name as the InfoBlox server IP.
4. Reload the InfoBlox user interface. The certificate will be loaded and can be verified through the browser.
5. To download the certificate, select: *System* › *Certificate* › *HTTP Cert* › *Download the Certificate*.
6. Copy the certificate file to `~/openstack/my_cloud/config/tls/cacerts/`.

Commit the changes to Git:

```
$ git commit -a -m "Configure Designate InfoBlox back-end"
```

10.4 Configure DNS Domain and NS Records

To configure the default DNS domain and Name Server records for the default pool, follow these steps.

1. Ensure that `designate_config.yml` file is present in the `~/openstack/my_cloud/definition/data/designate` folder. If the file or folder is not present, create the folder and copy `designate_config.yml` file from one of the example input models (for example, `~/openstack/examples/entry-scale-kvm/data/designate/designate_config.yml`).
2. Modify the **dns_domain** and/or **ns_records** entries in the `designate_config.yml` file.

```
data:
dns_domain: example.org.
ns_records:
    hostname: ns1.example.org.
    priority: 1
    hostname: ns2.example.org.
    priority: 2
```

3. Edit your input model's `control_plane.yml` file to include **DESIGNATE-CONFIG-CP1** in **configuration-data** section.

```
control-planes:
   - name: control-plane-1
     region-name: region1
     lifecycle-manager-target
     configuration-data:
        - DESIGNATE-CONFIG-CP1
        - NEUTRON-CONFIG-CP1
```

4. Continue your cloud deployment by reviewing and committing your changes.

```
$ git add ~/openstack/my_cloud/definition/data/designate/designate_config.yml
$ git commit -m "Adding DNS Domain and NS Records"
```

Note

In an entry-scale model (*Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.3 "KVM Examples", Section 10.3.1 "Entry-Scale Cloud"*), you will have 3 ns_records since the DNS service runs on all three control planes.

In a mid-scale model (*Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.3 "KVM Examples", Section 10.3.3 "Single-Region Mid-Size Model"*) or dedicated metering, monitoring and logging model (*Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations", Section 10.3 "KVM Examples", Section 10.3.2 "Entry Scale Cloud with Metering and Monitoring Services"*), the above example would be correct since there are only two controller nodes.

11 Magnum Overview

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.

11.1 Magnum Architecture

As an OpenStack API service, Magnum provides Container as a Service (CaaS) functionality. Magnum is capable of working with container orchestration engines (COE) such as Kubernetes, Docker Swarm, and Apache Mesos. Some operations work with a User CRUD (Create, Read, Update, Delete) filter.

Components

- **Magnum API**: RESTful API for cluster and cluster template operations.
- **Magnum Conductor**: Performs operations on clusters requested by Magnum API in an asynchronous manner.
- **Magnum CLI**: Command-line interface to the Magnum API.
- **Etcd (planned, currently using public service)**: Remote key/value storage for distributed cluster bootstrap and discovery.
- **Kubemaster (in case of Kubernetes COE)**: One or more VM(s) or baremetal server(s), representing a control plane for Kubernetes cluster.
- **Kubeminion (in case of Kubernetes COE)**: One or more VM(s) or baremetal server(s), representing a workload node for Kubernetes cluster.
- **Octavia VM aka Amphora (in case of Kubernetes COE with enabled load balancer functionality)**: One or more VM(s), created by LBaaS v2, performing request load balancing for Kubemasters.

TABLE 11.1: DATA

Data Name	Confiden-tiality	In-tegri-ty	Avail-ability	Back-up?	Description
Session To-kens	Confiden-tial	High	Medium	No	Session tokens not stored.
System Re-quest	Confiden-tial	High	Medium	No	Data in motion or in MQ not stored.
MariaDB Database "Magnum"	Confiden-tial	High	High	Yes	Contains user preferences. Backed up to Swift daily.
etcd data	Confiden-tial	High	Low	No	Kubemaster IPs and cluster info. Only used during cluster boot-strap.

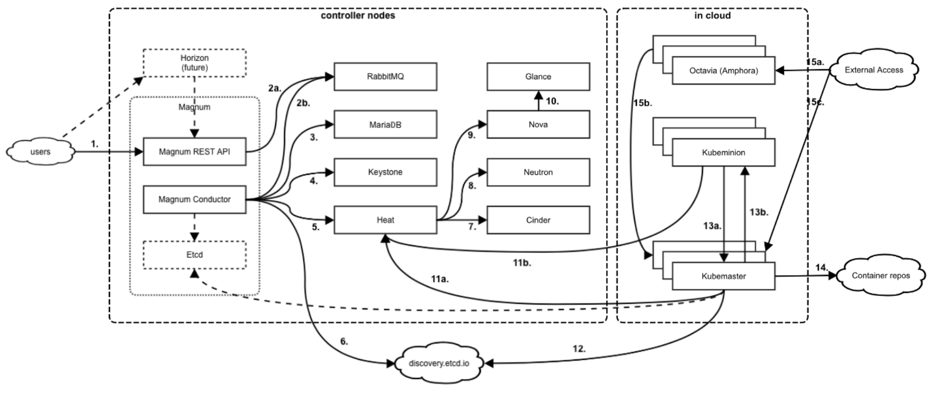


FIGURE 11.1: SERVICE ARCHITECTURE DIAGRAM FOR KUBERNETES

TABLE 11.2: INTERFACES

In-ter-fac	Net-work	Request	Response	Operation Description
1	**Name:** Exter-nal-API	**Request:** Manage clusters **Requester:** User	Operation status with or without data	CRUD operations on cluster templates and clusters

In-ter-fac	Net-work	Request	Response	Operation Description
	Protocol: HTTPS	**Credentials:** Keystone token **Authorization:** Manage objects that belong to current project **Listener:** Magnum API	**Credentials:** TLS certificate	
2a	**Name:** Internal-API **Protocol:** AMQP over HTTPS	**Request:** Enqueue messages **Requester:** Magnum API **Credentials:** RabbitMQ username, password **Authorization:** RabbitMQ queue read/write operations **Listener:** RabbitMQ	Operation status **Credentials:** TLS certificate	Notifications issued when cluster CRUD operations requested
2b	**Name:** Internal-API **Protocol:** AMQP over HTTPS	**Request:** Read queued messages **Requester:** Magnum Conductor **Credentials:** RabbitMQ username, password **Authorization:** RabbitMQ queue read/write operations **Listener:** RabbitMQ	Operation status **Credentials:** TLS certificate	Notifications issued when cluster CRUD operations requested
3	**Name:** Internal-API	**Request:** Persist data in MariaDB	Operation status with or without data	Persist cluster/cluster template data, read persisted data

Interface	Network	Request	Response	Operation Description
	Protocol: MariaDB over HTTPS	**Requester:** Magnum Conductor **Credentials:** MariaDB username, password **Authorization:** Magnum database **Listener:** MariaDB	**Credentials:** TLS certificate	
4	**Name:** Internal-API **Protocol:** HTTPS	**Request:** Create per-cluster user in dedicated domain, no role assignments initially **Requester:** Magnum Conductor **Credentials:** Trustee domain admin username, password **Authorization:** Manage users in dedicated Magnum domain **Listener:** Keystone	Operation status with or without data **Credentials:** TLS certificate	Magnum generates user record in a dedicated Keystone domain for each cluster
5	**Name:** Internal-API **Protocol:** HTTPS	**Request:** Create per-cluster user stack **Requester:** Magnum Conductor **Credentials:** Keystone token **Authorization:** Limited to scope of authorized user	Operation status with or without data **Credentials:** TLS certificate	Magnum creates Heat stack for each cluster

In-ter-fac	Net-work	Request	Response	Operation Description
		Listener: Heat		
6	**Name:** External Network **Protocol:** HTTPS	**Request:** Bootstrap a cluster in public discovery https://discovery.etcd.io/ **Requester:** Magnum Conductor **Credentials:** Unguessable URL over HTTPS. URL is only available to software processes needing it. **Authorization:** Read and update **Listener:** Public discovery service	Cluster discovery URL **Credentials:** TLS certificate	Create key/value registry of specified size in public storage. This is used to stand up a cluster of kubernetes master nodes (refer to interface call #12).
7	**Name:** Internal-API **Protocol:** HTTPS	**Request:** Create Cinder volumes **Requester:** Heat Engine **Credentials:** Keystone token **Authorization:** Limited to scope of authorized user **Listener:** Cinder API	Operation status with or without data **Credentials:** TLS certificate	Heat creates Cinder volumes as part of stack.
8	**Name:** Internal-API **Protocol:** HTTPS	**Request:** Create networks, routers, load balancers **Requester:** Heat Engine **Credentials:** Keystone token	Operation status with or without data **Credentials:** TLS certificate	Heat creates networks, routers, load balancers as part of the stack.

In- ter- fac	Net- work	Request	Response	Operation Description
		Authorization: Limited to scope of authorized user **Listener:** Neutron API		
9	**Name:** Inter- nal-API **Pro- tocol:** HTTPS	**Request:** Create Nova VMs, attach volumes **Requester:** Heat Engine **Credentials:** Keystone to- ken **Authorization:** Limited to scope of authorized user **Listener:** Nova API	Operation status with or without data **Credentials:** TLS certificate	Heat creates Nova VMs as part of the stack.
10	**Name:** Inter- nal-API **Pro- tocol:** HTTPS	**Request:** Read pre-config- ured Glance image **Requester:** Nova **Credentials:** Keystone to- ken **Authorization:** Limited to scope of authorized user **Listener:** Glance API	Operation status with or without data **Credentials:** TLS certificate	Nova uses pre-configured im- age in Glance to bootstrap VMs.
11a	**Name:** Exter- nal-API **Pro- tocol:** HTTPS	**Request:** Heat notification **Requester:** Cluster mem- ber (VM or Ironic node) **Credentials:** Keystone to- ken **Authorization:** Limited to scope of authorized user **Listener:** Heat API	Operation status with or without data **Credentials:** TLS certificate	Heat uses OS::Heat::WaitCon- dition resource. VM is expect- ed to call Heat notification URL upon completion of cer- tain bootstrap operation.

In-ter-fac	Net-work	Request	Response	Operation Description
11b	**Name:** Exter-nal-API **Pro-tocol:** HTTPS	**Request:** Heat notification **Requester:** Cluster mem-ber (VM or Ironic node) **Credentials:** Keystone to-ken **Authorization:** Limited to scope of authorized user **Listener:** Heat API	Operation status with or without data **Credentials:** TLS certificate	Heat uses OS::Heat::WaitCon-dition resource. VM is expect-ed to call Heat notification URL upon completion of cer-tain bootstrap operation.
12	**Name:** Exter-nal-API **Pro-tocol:** HTTPS	**Request:** Update cluster member state in a public registry at https://discov-ery.etcd.io **Requester:** Cluster mem-ber (VM or Ironic node) **Credentials:** Unguess-able URL over HTTPS on-ly available to software processes needing it. **Authorization:** Read and update **Listener:** Public discovery service	Operation status **Credentials:** TLS certificate	Update key/value pair in a registry created by interface call #6.
13a	**Name:** VxLAN en-capsu-lated pri-vate	**Request:** Various com-munications inside Kuber-netes cluster **Requester:** Cluster mem-ber (VM or Ironic node) **Credentials:** Tenant spe-cific	Tenant specific **Credentials:** TLS certificate	Various calls performed to build Kubernetes clusters, deploy applications and put workload

In-ter-fac	Net-work	Request	Response	Operation Description
	net-work on the Guest net-work **Pro-tocol:** HTTPS	**Authorization:** Tenant specific **Listener:** Cluster member (VM or Ironic node)		
13b	**Name:** VxLAN en-capsu-lated pri-vate net-work on the Guest net-work **Pro-tocol:** HTTPS	**Request:** Various com-munications inside Kuber-netes cluster **Requester:** Cluster mem-ber (VM or Ironic node) **Credentials:** Tenant spe-cific **Authorization:** Tenant specific **Listener:** Cluster member (VM or Ironic node)	Tenant specific **Credentials:** TLS certificate	Various calls performed to build Kubernetes clusters, deploy applications and put workload
14	**Name:** Guest/ Exter-nal	**Request:** Download con-tainer images **Requester:** Cluster mem-ber (VM or Ironic node) **Credentials:** None	Container image data **Credentials:** TLS certificate	Kubernetes makes calls to ex-ternal repositories to down-load pre-packed container im-ages

Interface	Network	Request	Response	Operation Description
	Protocol: HTTPS	**Authorization:** None **Listener:** External		
15a	**Name:** External/EXT_VM (Floating IP) **Protocol:** HTTPS	**Request:** Tenant specific **Requester:** Tenant specific **Credentials:** Tenant specific **Authorization:** Tenant specific **Listener:** Octavia load balancer	Tenant specific **Credentials:** Tenant specific	External workload handled by container applications
15b	**Name:** Guest **Protocol:** HTTPS	**Request:** Tenant specific **Requester:** Tenant specific **Credentials:** Tenant specific **Authorization:** Tenant specific **Listener:** Cluster member (VM or Ironic node)	Tenant specific **Credentials:** Tenant specific	External workload handled by container applications
15c	**Name:** External/EXT_VM (Floating IP)	**Request:** Tenant specific **Requester:** Tenant specific **Credentials:** Tenant specific **Authorization:** Tenant specific	Tenant specific **Credentials:** Tenant specific	External workload handled by container applications

In-ter-fac	Net-work	Request	Response	Operation Description
	Pro-tocol: HTTPS	**Listener:** Cluster member (VM or Ironic node)		

Dependencies

- Keystone
- RabbitMQ
- MariaDB
- Heat
- Glance
- Nova
- Cinder
- Neutron
- Barbican
- Swift

Implementation

Magnum API and Magnum Conductor are run on the SUSE OpenStack Cloud controllers (or core nodes in case of mid-scale deployments).

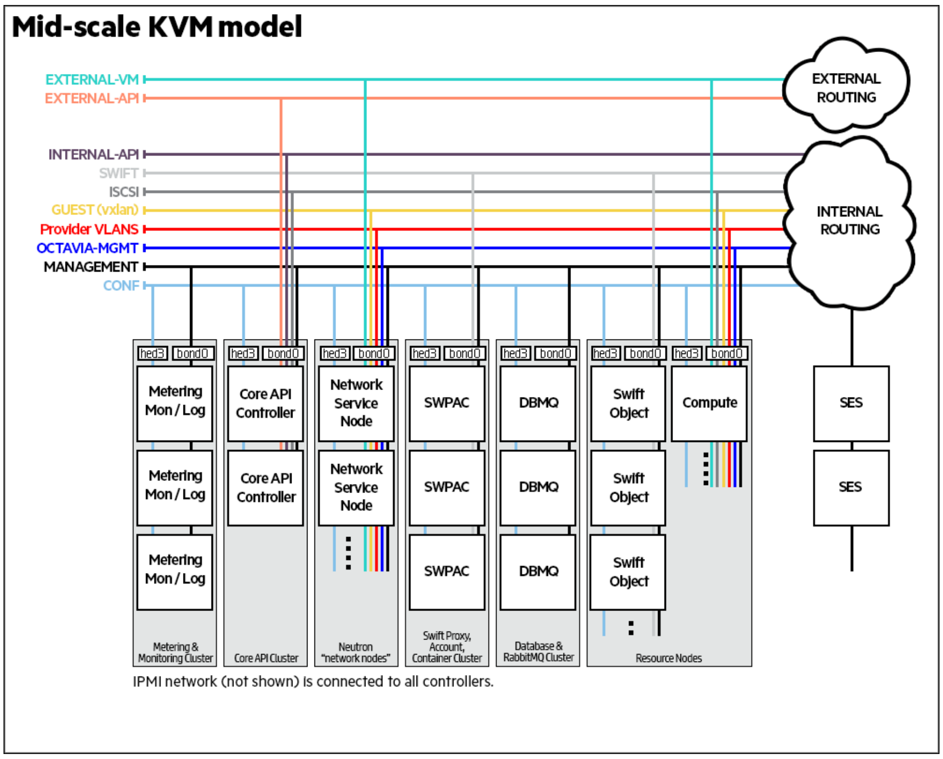


TABLE 11.3: **SECURITY GROUPS**

Source CIDR/Security Group	Port/Range	Protocol	Notes
Any IP	22	SSH	Tenant Admin access
Any IP/Kubernetes Security Group	2379-2380	HTTPS	Etcd Traffic
Any IP/Kubernetes Security Group	6443	HTTPS	kube-apiserver
Any IP/Kubernetes Security Group	7080	HTTPS	kube-apiserver
Any IP/Kubernetes Security Group	8080	HTTPS	kube-apiserver
Any IP/Kubernetes Security Group	30000-32767	HTTPS	kube-apiserver
Any IP/Kubernetes Security Group	any	tenant app specific	tenant app specific

TABLE 11.4: NETWORK PORTS

Port/Range	Protocol	Notes
22	SSH	Admin Access
9511	HTTPS	Magnum API Access
2379-2380	HTTPS	Etcd (planned)

Summary of controls spanning multiple components and interfaces:

- **Audit**: Magnum performs logging. Logs are collected by the centralized logging service.
- **Authentication**: Authentication via Keystone tokens at APIs. Password authentication to MQ and DB using specific users with randomly-generated passwords.
- **Authorization**: OpenStack provides admin and non-admin roles that are indicated in session tokens. Processes run at minimum privilege. Processes run as unique user/group definitions (magnum/magnum). Appropriate filesystem controls prevent other processes from accessing service's files. Magnum config file is mode 600. Logs written using group adm, user magnum, mode 640. IPtables ensure that no unneeded ports are open. Security Groups provide authorization controls between in-cloud components.
- **Availability**: Redundant hosts, clustered DB, and fail-over provide high availability.
- **Confidentiality**: Network connections over TLS. Network separation via VLANs. Data and config files protected via filesystem controls. Unencrypted local traffic is bound to localhost. Separation of customer traffic on the TUL network via Open Flow (VxLANs).
- **Integrity**: Network connections over TLS. Network separation via VLANs. DB API integrity protected by SQL Alchemy. Data and config files are protected by filesystem controls. Unencrypted traffic is bound to localhost.

11.2 Install the Magnum Service

Installing the Magnum Service can be performed as part of a new SUSE OpenStack Cloud 8 environment or can be added to an existing SUSE OpenStack Cloud 8 environment. Both installations require container management services, running in Magnum cluster VMs with access to specific Openstack API endpoints. The following TCP ports need to be open in your firewall to allow access from VMs to external (public) SUSE OpenStack Cloud endpoints.

TCP Port	Service
5000	Identity
8004	Heat
9511	Magnum

Magnum is dependent on the following OpenStack services.

- Keystone
- Heat
- Nova KVM
- Neutron
- Glance
- Cinder
- Swift
- Barbican
- LBaaS v2 (Octavia) - *optional*

Warning

Magnum relies on the public discovery service *https://discovery.etcd.io* during cluster bootstrapping and update. This service does not perform authentication checks. Although running a cluster cannot be harmed by unauthorized changes in the public discovery registry, it can be compromised during a cluster update operation. To avoid this, it is recommended that you keep your cluster discovery URL (that is, `https://discovery.etc.io/SOME_RANDOM_ID`) secret.

11.2.1 Installing Magnum as part of new SUSE OpenStack Cloud 8 environment

Magnum components are already included in example SUSE OpenStack Cloud models based on Nova KVM, such as **entry-scale-kvm**, **entry-scale-kvm-mml** and **mid-scale**. These models contain the Magnum dependencies (see above). You can follow generic installation instruction for Mid-Scale and Entry-Scale KM model by using this guide: *Chapter 13, Installing Mid-scale and Entry-scale KVM*.

Note

1. If you modify the cloud model to utilize a dedicated Cloud Lifecycle Manager, add `magnum-client` item to the list of service components for the Cloud Lifecycle Manager cluster.
2. Magnum needs a properly configured external endpoint. While preparing the cloud model, ensure that `external-name` setting in `data/network_groups.yml` is set to valid hostname, which can be resolved on DNS server, and a valid TLS certificate is installed for your external endpoint. For non-production test installations, you can omit `external-name`. In test installations, the SUSE OpenStack Cloud installer will use an IP address as a public endpoint hostname, and automatically generate a new certificate, signed by the internal CA. Please refer to *Chapter 25, Configuring Transport Layer Security (TLS)* for more details.
3. To use LBaaS v2 (Octavia) for container management and container applications, follow the additional steps to configure LBaaS v2 in the guide.

11.2.2 Adding Magnum to an Existing SUSE OpenStack Cloud Environment

Adding Magnum to an already deployed SUSE OpenStack Cloud 8 installation or during an upgrade can be achieved by performing the following steps.

1. Add items listed below to the list of service components in `~/openstack/my_cloud/definition/data/control_plane.yml`. Add them to clusters which have `server-role` set to `CONTROLLER-ROLE` (entry-scale models) or `CORE_ROLE` (mid-scale model).

```
- magnum-api
- magnum-conductor
```

2. If your environment utilizes a dedicated Cloud Lifecycle Manager, add `magnum-client` to the list of service components for the Cloud Lifecycle Manager.

3. Commit your changes to the local git repository. Run the following playbooks as described in *Chapter 12, Using Git for Configuration Management* for your installation.

   - `config-processor-run.yml`
   - `ready-deployment.yml`
   - `site.yml`

4. Ensure that your external endpoint is configured correctly. The current public endpoint configuration can be verified by running the following commands on the Cloud Lifecycle Manager.

```
$ source service.osrc
$ openstack endpoint list --interface=public --service=identity
+-----------+---------+--------------+----------+---------+-----------+------------------------+
| ID        | Region  | Service Name | Service  | Enabled | Interface | URL                    |
|           |         |              | Type     |         |           |                        |
+-----------+---------+--------------+----------+---------+-----------+------------------------+
| d83...aa3 | region1 | keystone     | identity | True    | public    | https://10.245.41.168: |
|           |         |              |          |         |           |             5000/v2.0  |
+-----------+---------+--------------+----------+---------+-----------+------------------------+
```

   Ensure that the endpoint URL is using either an IP address, or a valid hostname, which can be resolved on the DNS server. If the URL is using an invalid hostname (for example, `myardana.test`), follow the steps in *Chapter 25, Configuring Transport Layer Security (TLS)* to configure a valid external endpoint. You will need to update the `external-name` setting in the `data/network_groups.yml` to a valid hostname, which can be resolved on DNS server, and provide a valid TLS certificate for the external endpoint. For non-production test installations, you can omit the `external-name`. The SUSE OpenStack Cloud installer will use an IP address as public endpoint hostname, and automatically generate a new certificate, signed by the internal CA. For more information, see *Chapter 25, Configuring Transport Layer Security (TLS)*.

5. Ensure that LBaaS v2 (Octavia) is correctly configured. For more information, see *Chapter 27, Configuring Load Balancer as a Service*.

Warning

By default SUSE OpenStack Cloud stores the private key used by Magnum and its passphrase in Barbican which provides a secure place to store such information. You can change this such that this sensitive information is stored on the file system or in the database without encryption. Making such a change exposes you to the risk of this information being exposed to others. If stored in the database then any database backups, or a database breach, could lead to the disclosure of the sensitive information. Similarly, if stored unencrypted on the file system this information is exposed more broadly than if stored in Barbican.

11.3 Integrate Magnum with the DNS Service

Integration with DNSaaS may be needed if:

1. The external endpoint is configured to use `myardana.test` as host name and SUSE OpenStack Cloud front-end certificate is issued for this host name.

2. Minions are registered using Nova VM names as hostnames Kubernetes API server. Most kubectl commands will not work if the VM name (for example, `cl-mu3eevqizh-1-b3vi-fun6qtuh-kube-minion-ff4cqjgsuzhy`) is not getting resolved at the provided DNS server.

Follow these steps to integrate the Magnum Service with the DNS Service.

1. Allow connections from VMs to EXT-API

```
sudo modprobe 8021q
sudo ip link add link virbr5 name vlan108 type vlan id 108
sudo ip link set dev vlan108 up
sudo ip addr add 192.168.14.200/24 dev vlan108
sudo iptables -t nat -A POSTROUTING -o vlan108 -j MASQUERADE
```

2. Run the designate reconfigure playbook.

```
$ cd ~/scratch/ansible/next/ardana/ansible/
$ ansible-playbook -i hosts/verb_hosts designate-reconfigure.yml
```

3. Set up Designate to resolve myardana.test correctly.

```
$ openstack zone create --email hostmaster@myardana.test myardana.test.
# wait for status to become active
$ EXTERNAL_VIP=$(grep HZN-WEB-extapi /etc/hosts | awk '{ print $1 }')
$ openstack recordset create --records $EXTERNAL_VIP --type A myardana.test.
 myardana.test.
# wait for status to become active
$ LOCAL_MGMT_IP=$(grep `hostname` /etc/hosts | awk '{ print $1 }')
$ nslookup myardana.test $LOCAL_MGMT_IP
Server:         192.168.14.2
Address:        192.168.14.2#53
Name:           myardana.test
Address:        192.168.14.5
```

4. If you need to add/override a top level domain record, the following example should be used, substituting proxy.example.org with your own real address:

```
$ openstack tld create --name net
$ openstack zone create --email hostmaster@proxy.example.org proxy.example.org.
$ openstack recordset create --records 16.85.88.10 --type A proxy.example.org.
 proxy.example.org.
$ nslookup proxy.example.org. 192.168.14.2
Server:         192.168.14.2
Address:        192.168.14.2#53
Name:           proxy.example.org
Address:        16.85.88.10
```

5. Enable propagation of dns_assignment and dns_name attributes to neutron ports, as per https://docs.openstack.org/draft/networking-guide/config-dns-int.html

```
# optionally add 'dns_domain = <some domain name>.' to [DEFAULT] section
# of ardana/ansible/roles/neutron-common/templates/neutron.conf.j2
stack@ksperf2-cp1-c1-m1-mgmt:~/openstack$ cat <<-EOF >>ardana/services/designate/api.yml

   provides-data:
   -   to:
       -   name: neutron-ml2-plugin
       data:
       -   option: extension_drivers
           values:
           -   dns
EOF
$ git commit -a -m "Enable DNS support for neutron ports"
$ cd ardana/ansible
$ ansible-playbook -i hosts/localhost config-processor-run.yml
$ ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Enable DNSaaS registration of created VMs by editing the `~/openstack/ardana/ansible/roles/neutron-common/templates/neutron.conf.j2` file. You will need to add `external_dns_driver = designate` to the **[DEFAULT]** section and create a new **[designate]** section for the Designate specific configurations.

```
...
advertise_mtu = False
dns_domain = ksperf.
external_dns_driver = designate
{{ neutron_api_extensions_path|trim }}
{{ neutron_vlan_transparent|trim }}

# Add additional options here

[designate]
url = https://10.240.48.45:9001
admin_auth_url = https://10.240.48.45:35357/v3
admin_username = designate
admin_password = P8lZ9FdHuoW
admin_tenant_name = services
allow_reverse_dns_lookup = True
ipv4_ptr_zone_prefix_size = 24
ipv6_ptr_zone_prefix_size = 116
ca_cert = /etc/ssl/certs/ca-certificates.crt
```

7. Commit your changes.

```
$ git commit -a -m "Enable DNSaaS registration of Nova VMs"
[site f4755c0] Enable DNSaaS registration of Nova VMs
1 file changed, 11 insertions(+)
```

12 Using Git for Configuration Management

In SUSE OpenStack Cloud 8, a local git repository is used to track configuration changes; the Configuration Processor (CP) uses this repository. Use of a git workflow means that your configuration history is maintained, making rollbacks easier and keeping a record of previous configuration settings. The git repository also provides a way for you to merge changes that you pull down as "upstream" updates (that is, updates from SUSE). It also allows you to manage your own configuration changes.

The git repository is installed by the Cloud Lifecycle Manager on the Cloud Lifecycle Manager node.

12.1 Initialization on a new deployment

On a system new to SUSE OpenStack Cloud 8, the Cloud Lifecycle Manager will prepare a git repository under `~/openstack`. The Cloud Lifecycle Manager provisioning runs the `ardana-init-deployer` script automatically. This calls `ansible-playbook -i hosts/localhost git-00-initialise.yml`.

As a result, the `~/openstack` directory is initialized as a git repo (if it is empty). It is initialized with four empty branches:

ardana

This holds the upstream source code corresponding to the contents of the `~/openstack` directory on a pristine installation. Every source code release that is downloaded from SUSE is applied as a fresh commit to this branch. This branch contains no customization by the end user.

site

This branch begins life as a copy of the first `ardana` drop. It is onto this branch that you commit your configuration changes. It is the branch most visible to the end user.

ansible

This branch contains the variable definitions generated by the CP that our main ansible playbooks need. This includes the `verb_hosts` file that describes to ansible what servers are playing what roles. The `ready-deployment` playbook takes this output and assembles a `~/scratch` directory containing the ansible playbooks together with the variable definitions in this branch. The result is a working ansible directory `~/scratch/ansible/next/ardana/ansible` from which the main deployment playbooks may be successfully run.

cp-persistent

This branch contains the persistent state that the CP needs to maintain. That state is mostly the assignment of IP addresses and roles to particular servers. Some operational procedures may involve editing the contents of this branch: for example, retiring a machine from service or repurposing it.

Two temporary branches are created and populated at run time:

staging-ansible

This branch hosts the most recent commit that will be appended to the Ansible branch.

staging-cp-persistent

This branch hosts the most recent commit that will be appended to the cp-persistent branch.

Note

The information above provides insight into the workings of the configuration processor and the git repository. However, in practice you can simply follow the steps below to make configuration changes.

12.2 Updating any configuration, including the default configuration

When you need to make updates to a configuration you must:

1. Check out the `site` branch. You may already be on that branch. If so, git will tell you that and the command will leave you there.

```
git checkout site
```

2. Edit the YAML file or files that contain the configuration you want to change.

3. Commit the changes to the `site` branch.

```
git add -A
git commit -m "your commit message goes here in quotes"
```

If you want to add a single file to your git repository, you can use the command below, as opposed to using **`git add -A`**.

```
git add PATH_TO_FILE
```

For example, if you made a change to your **`servers.yml`** file and wanted to only commit that change, you would use this command:

```
git add ~/openstack/my_cloud/definition/data/servers.yml
```

4. To produce the required configuration processor output from those changes. Review the output files manually if required, run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Ready the deployment area

```
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the deployment playbooks from the resulting scratch directory.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts site.yml
```

13 Installing Mid-scale and Entry-scale KVM

13.1 Important Notes

- If you are looking for information about when to use the GUI installer and when to use the command line (CLI), see the *Installation Overview*.
- Review the *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 2 "Hardware and Software Support Matrix"* that we have listed.
- Review the release notes to make yourself aware of any known issues and limitations.
- The installation process can occur in different phases. For example, you can install the control plane only and then add Compute nodes afterwards if you would like.
- If you run into issues during installation, we have put together a list of *Chapter 19, Troubleshooting the Installation* you can reference.
- Make sure all disks on the system(s) are wiped before you begin the install. (For Swift, refer to *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 12 "Modifying Example Configurations for Object Storage using Swift", Section 12.6 "Swift Requirements for Device Group Drives"*.)
- There is no requirement to have a dedicated network for OS-install and system deployment, this can be shared with the management network. More information can be found in *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 10 "Example Configurations"*.
- You may see the terms deployer and Cloud Lifecycle Manager used interchangeably. These are referring to the same nodes in your environment.
- When running the Ansible playbook in this installation guide, if a runbook fails you will see in the error response to use the `--limit` switch when retrying a playbook. This should be avoided. You can simply re-run any playbook without this switch.
- DVR is not supported with ESX compute.
- When you attach a Cinder volume to the VM running on the ESXi host, the volume will not get detected automatically. Make sure to set the image metadata **vmware_adapter-type=lsiLogicsas** for image before launching the instance. This will help to discover the volume change appropriately.

13.2 Before You Start

1. Review the *Chapter 2, Pre-Installation Checklist* about recommended pre-installation tasks.

2. Prepare the Cloud Lifecycle Manager node. The Cloud Lifecycle Manager must be accessible either directly or via `ssh`, and have SUSE Linux Enterprise Server 12 SP3 installed. All nodes must be accessible to the Cloud Lifecycle Manager. If the nodes do not have direct access to online Cloud subscription channels, the Cloud Lifecycle Manager node will need to host the Cloud repositories.

   a. If you followed the installation instructions for Cloud Lifecycle Manager (see *Chapter 3, Installing the Cloud Lifecycle Manager server* SUSE OpenStack Cloud should already be installed. Double-check whether SUSE Linux Enterprise and SUSE OpenStack Cloud are properly registered at the SUSE Customer Center by starting YaST and running *Software* › *Product Registration*.
   If you have not yet installed SUSE OpenStack Cloud, do so by starting YaST and running *Software* › *Product Registration* › *Select Extensions*. Choose *SUSE OpenStack Cloud* and follow the on-screen instructions. Make sure to register SUSE OpenStack Cloud during the installation process and to install the software pattern `patterns-cloud-ardana`.

   b. Ensure the SUSE OpenStack Cloud media repositories and updates repositories are made available to all nodes in your deployment. This can be accomplished either by configuring the Cloud Lifecycle Manager server as an SMT mirror as described in *Chapter 4, Installing and Setting Up an SMT Server on the Cloud Lifecycle Manager server (Optional)* or by syncing or mounting the Cloud and updates repositories to the Cloud Lifecycle Manager server as described in *Chapter 5, Software Repository Setup*.

   c. Configure passwordless **`sudo`** for the user created when setting up the node (as described in *Section 3.5, "Creating a User"*). Note that this is *not* the user `ardana` that will be used later in this procedure. In the following we assume you named the user `cloud`. Run the command **`visudo`** as user `root` and add the following line to the end of the file:

      ```
      cloud ALL = (root) NOPASSWD:ALL
      ```

      Make sure to replace *`cloud`* with your user name choice.

d. Set the password for the user `ardana`:

```
sudo passwd ardana
```

e. Become the user `ardana`:

```
su - ardana
```

f. Place a copy of the SUSE Linux Enterprise Server 12 SP3 `.iso` in the `ardana` home directory, `var/lib/ardana`, and rename it to `sles12sp3.iso`.

13.3 Configuring Your Environment

During the configuration phase of the installation you will be making modifications to the example configuration input files to match your cloud environment. You should use the *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 10 “Example Configurations”* documentation for detailed information on how to do this. There is also a `README.md` file included in each of the example directories on the Cloud Lifecycle Manager that has useful information about the models.

In the steps below we show how to set up the directory structure with the example input files as well as use the optional encryption methods for your sensitive data.

1. Setup your configuration files, as follows:

   a. Copy the example configuration files into the required setup directory and edit them to contain the details of your environment.
   For example, if you want to use the SUSE OpenStack Cloud Mid-scale KVM model, you can use this command to copy the files to your cloud definition directory:

```
cp -r ~/openstack/examples/mid-scale-kvm/* \
~/openstack/my_cloud/definition/
```

   If you want to use the SUSE OpenStack Cloud Entry-scale KVM model, you can use this command to copy the files to your cloud definition directory:

```
cp -r ~/openstack/examples/entry-scale-kvm/* \
~/openstack/my_cloud/definition/
```

b. Begin inputting your environment information into the configuration files in the `~/openstack/my_cloud/definition` directory.

2. *(Optional)* You can use the `ardanaencrypt.py` script to encrypt your IPMI passwords. This script uses OpenSSL.

   a. Change to the Ansible directory:

   ```
   cd ~/openstack/ardana/ansible
   ```

   b. Put the encryption key into the following environment variable:

   ```
   export ARDANA_USER_PASSWORD_ENCRYPT_KEY=<encryption key>
   ```

   c. Run the python script below and follow the instructions. Enter a password that you want to encrypt.

   ```
   ./ardanaencrypt.py
   ```

   d. Take the string generated and place it in the `ilo-password` field in your `~/openstack/my_cloud/definition/data/servers.yml` file, remembering to enclose it in quotes.

   e. Repeat the above for each server.

   Note

   Before you run any playbooks, remember that you need to export the encryption key in the following environment variable: `export ARDANA_USER_PASSWORD_ENCRYPT_KEY=<encryption key>`

3. Commit your configuration to the local git repo (*Chapter 12, Using Git for Configuration Management*), as follows:

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

Important

This step needs to be repeated any time you make changes to your configuration files before you move onto the following steps. See *Chapter 12, Using Git for Configuration Management* for more information.

13.4 Provisioning Your Baremetal Nodes

To provision the baremetal nodes in your cloud deployment you can either use the automated operating system installation process provided by SUSE OpenStack Cloud or you can use the 3rd party installation tooling of your choice. We will outline both methods below:

13.4.1 Using Third Party Baremetal Installers

If you do not wish to use the automated operating system installation tooling included with SUSE OpenStack Cloud then the requirements that have to be met using the installation tooling of your choice are:

- The operating system must be installed via the SLES ISO provided on the SUSE Customer Center (https://scc.suse.com/).
- Each node must have SSH keys in place that allows the same user from the Cloud Lifecycle Manager node who will be doing the deployment to SSH to each node without a password.
- Passwordless sudo needs to be enabled for the user.
- There should be a LVM logical volume as `/root` on each node.
- If the LVM volume group name for the volume group holding the `root` LVM logical volume is `ardana-vg`, then it will align with the disk input models in the examples.
- Ensure that `openssh-server`, `python`, `python-apt`, and `rsync` are installed.

If you chose this method for installing your baremetal hardware, skip forward to the step *Running the Configuration Processor*.

13.4.2 Using the Automated Operating System Installation Provided by SUSE OpenStack Cloud

If you would like to use the automated operating system installation tools provided by SUSE OpenStack Cloud, complete the steps below.

13.4.2.1 Deploying Cobbler

This phase of the install process takes the baremetal information that was provided in `servers.yml` and installs the Cobbler provisioning tool and loads this information into Cobbler. This sets each node to `netboot-enabled: true` in Cobbler. Each node will be automatically marked as `netboot-enabled: false` when it completes its operating system install successfully. Even if the node tries to PXE boot subsequently, Cobbler will not serve it. This is deliberate so that you can't reimage a live node by accident.

The `cobbler-deploy.yml` playbook prompts for a password - this is the password that will be encrypted and stored in Cobbler, which is associated with the user running the command on the Cloud Lifecycle Manager, that you will use to log in to the nodes via their consoles after install. The username is the same as the user set up in the initial dialogue when installing the Cloud Lifecycle Manager from the ISO, and is the same user that is running the cobbler-deploy play.

1. Run the following playbook which confirms that there is IPMI connectivity for each of your nodes so that they are accessible to be re-imaged in a later step:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost bm-power-status.yml
   ```

2. Run the following playbook to deploy Cobbler:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost cobbler-deploy.yml
   ```

13.4.2.2 Imaging the Nodes

This phase of the install process goes through a number of distinct steps:

1. Powers down the nodes to be installed

2. Sets the nodes hardware boot order so that the first option is a network boot.

3. Powers on the nodes. (The nodes will then boot from the network and be installed using infrastructure set up in the previous phase)
4. Waits for the nodes to power themselves down (this indicates a success install). This can take some time.
5. Sets the boot order to hard disk and powers on the nodes.
6. Waits for the nodes to be ssh-able and verifies that they have the signature expected.

The reimaging command is:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-reimage.yml \
  [-e nodelist=node1,node2,node3]
```

If a nodelist is not specified then the set of nodes in cobbler with `netboot-enabled: True` is selected. The playbook pauses at the start to give you a chance to review the set of nodes that it is targeting and to confirm that it is correct.

You can use the command below which will list all of your nodes with the `netboot-enabled: True` flag set:

```
sudo cobbler system find --netboot-enabled=1
```

13.5 Running the Configuration Processor

Once you have your configuration files setup, you need to run the configuration processor to complete your configuration.

When you run the configuration processor, you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the ansible `group_vars` and `host_vars` that it produces for subsequent deploy runs. You will need this password for subsequent Ansible deploy and configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise just press `Enter` to bypass this.

Run the configuration processor with this command:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

For automated installation (for example CI), you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor:

```
ansible-playbook -i hosts/localhost config-processor-run.yml \
  -e encrypt="" -e rekey=""
```

If you receive an error during this step, there is probably an issue with one or more of your configuration files. Verify that all information in each of your configuration files is correct for your environment. Then commit those changes to Git using the instructions in the previous section before re-running the configuration processor again.

For any troubleshooting information regarding these steps, see *Section 19.2, "Issues while Updating Configuration Files"*.

13.6 Configuring TLS

Important

This section is optional, but recommended, for a SUSE OpenStack Cloud installation.

After you run the configuration processor the first time, the IP addresses for your environment will be generated and populated in the `~/openstack/my_cloud/info/address_info.yml` file. At this point, consider whether to configure TLS and set up an SSL certificate for your environment. Please read *Chapter 25, Configuring Transport Layer Security (TLS)* before proceeding for how to achieve this.

13.7 Deploying the Cloud

1. Use the playbook below to create a deployment directory:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

2. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your partitions on your nodes are completely wiped before continuing with the installation. If you are using fresh machines this step may not be necessary.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts wipe_disks.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts wipe_disks.yml --ask-vault-pass
```

3. Run the `site.yml` playbook below:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts site.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts site.yml --ask-vault-pass
```

Note

The step above runs `osconfig` to configure the cloud and `ardana-deploy` to deploy the cloud. Therefore, this step may run for a while, perhaps 45 minutes or more, depending on the number of nodes in your environment.

4. Verify that the network is working correctly. Ping each IP in the `/etc/hosts` file from one of the controller nodes.

For any troubleshooting information regarding these steps, see *Section 19.3, “Issues while Deploying the Cloud”*.

13.8 Configuring a Block Storage Backend (Optional)

SUSE OpenStack Cloud supports multiple block storage backend options. You can use one or more of these for setting up multiple block storage backends. Multiple volume types are also supported.

Whether you have a single or multiple block storage backends defined in your `cinder.conf.j2` file, you can create one or more volume types using the specific attributes associated with the backend. For more information, see *Section 18.1, “Configuring for 3PAR Block Storage Backend”*.

13.9 Post-Installation Verification and Administration

We recommend verifying the installation using the instructions in *Chapter 22, Cloud Verification*.

There are also a list of other common post-installation administrative tasks listed in the *Chapter 28, Other Common Post-Installation Tasks* list.

14 Installing Baremetal (Ironic)

Bare Metal as a Service is enabled in this release for deployment of Nova instances on bare metal nodes using flat networking.

14.1 Installation for SUSE OpenStack Cloud Entry-scale Cloud with Ironic Flat Network

This page describes the installation step requirements for the SUSE OpenStack Cloud Entry-scale Cloud with Ironic Flat Network.

14.1.1 Configure Your Environment

Prior to deploying an operational environment with Ironic, operators need to be aware of the nature of TLS certificate authentication. As pre-built deployment agent ramdisks images are supplied, these ramdisk images will only authenticate known third-party TLS Certificate Authorities in the interest of end-to-end security. As such, uses of self-signed certificates and private certificate authorities will be unable to leverage ironic without modifying the supplied ramdisk images.

1. Set up your configuration files, as follows:

   a. See the sample sets of configuration files in the `~/openstack/examples/` directory. Each set will have an accompanying README.md file that explains the contents of each of the configuration files.

   b. Copy the example configuration files into the required setup directory and edit them to contain the details of your environment:

   ```
   cp -r ~/openstack/examples/entry-scale-ironic-flat-network/* \
     ~/openstack/my_cloud/definition/
   ```

2. *(Optional)* You can use the `ardanaencrypt.py` script to encrypt your IPMI passwords. This script uses OpenSSL.

a. Change to the Ansible directory:

```
cd ~/openstack/ardana/ansible
```

b. Put the encryption key into the following environment variable:

```
export ARDANA_USER_PASSWORD_ENCRYPT_KEY=<encryption key>
```

c. Run the python script below and follow the instructions. Enter a password that you want to encrypt.

```
./ardanaencrypt.py
```

d. Take the string generated and place it in the `ilo-password` field in your `~/openstack/my_cloud/definition/data/servers.yml` file, remembering to enclose it in quotes.

e. Repeat the above for each server.

Note

Before you run any playbooks, remember that you need to export the encryption key in the following environment variable: `export ARDANA_USER_PASSWORD_ENCRYPT_KEY=<encryption key>`

3. Commit your configuration to the local git repo (*Chapter 12, Using Git for Configuration Management*), as follows:

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

Important

This step needs to be repeated any time you make changes to your configuration files before you move onto the following steps. See *Chapter 12, Using Git for Configuration Management* for more information.

14.1.2 Provisioning Your Baremetal Nodes

To provision the baremetal nodes in your cloud deployment you can either use the automated operating system installation process provided by SUSE OpenStack Cloud or you can use the 3rd party installation tooling of your choice. We will outline both methods below:

14.1.2.1 Using Third Party Baremetal Installers

If you do not wish to use the automated operating system installation tooling included with SUSE OpenStack Cloud then the requirements that have to be met using the installation tooling of your choice are:

- The operating system must be installed via the SLES ISO provided on the SUSE Customer Center (https://scc.suse.com/) ↗.
- Each node must have SSH keys in place that allows the same user from the Cloud Lifecycle Manager node who will be doing the deployment to SSH to each node without a password.
- Passwordless sudo needs to be enabled for the user.
- There should be a LVM logical volume as `/root` on each node.
- If the LVM volume group name for the volume group holding the `root` LVM logical volume is `ardana-vg`, then it will align with the disk input models in the examples.
- Ensure that `openssh-server`, `python`, `python-apt`, and `rsync` are installed.

If you chose this method for installing your baremetal hardware, skip forward to the step *Running the Configuration Processor*.

14.1.2.2 Using the Automated Operating System Installation Provided by SUSE OpenStack Cloud

If you would like to use the automated operating system installation tools provided by SUSE OpenStack Cloud, complete the steps below.

14.1.2.2.1 Deploying Cobbler

This phase of the install process takes the baremetal information that was provided in `servers.yml` and installs the Cobbler provisioning tool and loads this information into Cobbler. This sets each node to `netboot-enabled: true` in Cobbler. Each node will be automatically marked as `netboot-enabled: false` when it completes its operating system install successfully. Even if the node tries to PXE boot subsequently, Cobbler will not serve it. This is deliberate so that you can't reimage a live node by accident.

The `cobbler-deploy.yml` playbook prompts for a password - this is the password that will be encrypted and stored in Cobbler, which is associated with the user running the command on the Cloud Lifecycle Manager, that you will use to log in to the nodes via their consoles after install. The username is the same as the user set up in the initial dialogue when installing the Cloud Lifecycle Manager from the ISO, and is the same user that is running the cobbler-deploy play.

1. Run the following playbook which confirms that there is IPMI connectivity for each of your nodes so that they are accessible to be re-imaged in a later step:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost bm-power-status.yml
   ```

2. Run the following playbook to deploy Cobbler:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost cobbler-deploy.yml
   ```

14.1.2.2.2 Imaging the Nodes

This phase of the install process goes through a number of distinct steps:

1. Powers down the nodes to be installed
2. Sets the nodes hardware boot order so that the first option is a network boot.
3. Powers on the nodes. (The nodes will then boot from the network and be installed using infrastructure set up in the previous phase)
4. Waits for the nodes to power themselves down (this indicates a success install). This can take some time.
5. Sets the boot order to hard disk and powers on the nodes.

6. Waits for the nodes to be ssh-able and verifies that they have the signature expected.

The reimaging command is:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-reimage.yml \
  [-e nodelist=node1,node2,node3]
```

If a nodelist is not specified then the set of nodes in cobbler with `netboot-enabled: True` is selected. The playbook pauses at the start to give you a chance to review the set of nodes that it is targeting and to confirm that it is correct.

You can use the command below which will list all of your nodes with the `netboot-enabled: True` flag set:

```
sudo cobbler system find --netboot-enabled=1
```

14.1.3 Running the Configuration Processor

Once you have your configuration files setup, you need to run the configuration processor to complete your configuration.

When you run the configuration processor, you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the ansible `group_vars` and `host_vars` that it produces for subsequent deploy runs. You will need this password for subsequent Ansible deploy and configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise just press `Enter` to bypass this.

Run the configuration processor with this command:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

For automated installation (for example CI), you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor:

```
ansible-playbook -i hosts/localhost config-processor-run.yml \
  -e encrypt="" -e rekey=""
```

If you receive an error during this step, there is probably an issue with one or more of your configuration files. Verify that all information in each of your configuration files is correct for your environment. Then commit those changes to Git using the instructions in the previous section before re-running the configuration processor again.

For any troubleshooting information regarding these steps, see *Section 19.2, "Issues while Updating Configuration Files"*.

14.1.4 Deploying the Cloud

1. Use the playbook below to create a deployment directory:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

2. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your partitions on your nodes are completely wiped before continuing with the installation. If you are using fresh machines this step may not be necessary.

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts wipe_disks.yml
   ```

   If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

   ```
   ansible-playbook -i hosts/verb_hosts wipe_disks.yml --ask-vault-pass
   ```

3. Run the `site.yml` playbook below:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts site.yml
   ```

   If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

   ```
   ansible-playbook -i hosts/verb_hosts site.yml --ask-vault-pass
   ```

Note

The step above runs `osconfig` to configure the cloud and `ardana-deploy` to deploy the cloud. Therefore, this step may run for a while, perhaps 45 minutes or more, depending on the number of nodes in your environment.

4. Verify that the network is working correctly. Ping each IP in the `/etc/hosts` file from one of the controller nodes.

For any troubleshooting information regarding these steps, see *Section 19.3, “Issues while Deploying the Cloud”*.

14.1.5 Ironic configuration

Run the `ironic-cloud-configure.yml` playbook below:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ironic-cloud-configure.yml
```

This step configures ironic flat network, uploads glance images and sets the ironic configuration.

To see the images uploaded to glance, run:

```
$ source ~/service.osrc
$ glance image-list
```

This will produce output like the following example, showing three images that have been added by Ironic:

```
+--------------------------------------+--------------------------+
| ID                                   | Name                     |
+--------------------------------------+--------------------------+
| d4e2a0ff-9575-4bed-ac5e-5130a1553d93 | ir-deploy-iso-HOS3.0     |
| b759a1f0-3b33-4173-a6cb-be5706032124 | ir-deploy-kernel-HOS3.0  |
| ce5f4037-e368-46f2-941f-c01e9072676c | ir-deploy-ramdisk-HOS3.0 |
+--------------------------------------+--------------------------+
```

To see the network created by Ironic, run:

```
$ neutron net-list
```

This returns details of the "flat-net" generated by the Ironic install:

```
+---------------+----------+-------------------------------------------------------+
| id            | name     | subnets                                               |
+---------------+----------+-------------------------------------------------------+
| f9474...11010 | flat-net | ca8f8df8-12c8-4e58-b1eb-76844c4de7e8 192.168.245.0/24 |
+---------------+----------+-------------------------------------------------------+
```

14.1.6 Node Configuration

14.1.6.1 DHCP

Once booted, nodes obtain network configuration via DHCP. If multiple interfaces are to be utilized, you may want to pre-build images with settings to execute DHCP on all interfaces. An easy way to build custom images is with KIWI, the command line utility to build Linux system appliances.

For information about building custom KIWI images, see *Section 14.3.11, “Building Glance Images Using KIWI”*. For more information, see the KIWI documentation at https://suse.github.io/kiwi/ ↗.

14.1.6.2 Configuration Drives

Warning

Configuration Drives are stored unencrypted and should not include any sensitive data.

You can use Configuration Drives to store metadata for initial boot setting customization. Configuration Drives are extremely useful for initial machine configuration. However, as a general security practice, they should not include any sensitive data. Configuration Drives should only be trusted upon the initial boot of an instance. `cloud-init` utilizes a lock file for this purpose. Custom instance images should not rely upon the integrity of a Configuration Drive beyond the initial boot of a host as an administrative user within a deployed instance can potentially modify a configuration drive once written to disk and released for use.

For more information about Configuration Drives, see http://docs.openstack.org/user-guide/cli_config_drive.html ↗.

14.1.7 TLS Certificates with Ironic Python Agent (IPA) Images

As part of SUSE OpenStack Cloud 8, Ironic Python Agent, better known as IPA in the OpenStack community, images are supplied and loaded into Glance. Two types of image exist. One is a traditional boot ramdisk which is used by the `agent_ipmitool`, `pxe_ipmitool`, and `pxe_ilo` drivers. The other is an ISO image that is supplied as virtual media to the host when using the `agent_ilo` driver.

As these images are built in advance, they are unaware of any private certificate authorities. Users attempting to utilize self-signed certificates or a private certificate authority will need to inject their signing certificate(s) into the image in order for IPA to be able to boot on a remote node, and ensure that the TLS endpoints being connected to in SUSE OpenStack Cloud can be trusted. This is not an issue with publicly signed certificates.

As two different types of images exist, below are instructions for disassembling the image ramdisk file or the ISO image. Once this has been done, you will need to re-upload the files to glance, and update any impacted node's `driver_info`, for example, the `deploy_ramdisk` and `ilo_deploy_iso` settings that were set when the node was first defined. Respectively, this can be done with the

```
ironic node-update <node> replace driver_info/deploy_ramdisk=<glance_id>
```

or

```
ironic node-update <node> replace driver_info/ilo_deploy_iso=<glance_id>
```

14.1.7.1 Adding a certificate into a ramdisk image

As root, from a folder where the ramdisk image is present, perform the following steps.

1. Download the ramdisk image to /tmp/ and name the file ironic-deploy.initramfs
2. Change your working directory to /tmp/

   ```
   cd /tmp/
   ```

3. Create a temporary folder that will hold the temporarily extracted image contents.

   ```
   mkdir new_deploy_ramdisk
   ```

4. Change your shell working directory to the temporary folder

```
cd /tmp/new_deploy_ramdisk
```

5. Extract the original deployment archive file.

```
zcat /tmp/ironic-deploy.initramfs | cpio --extract --make-directories
```

6. Append your CA certificate to the file located at `/tmp/new_deploy_ramdisk/usr/local/lib/python2.7/dist-packages/requests/cacert.pem`, example below:

```
cat your_ca_certificate.pem >> /tmp/new_deploy_ramdisk/usr/local/lib/python2.7/dist-
packages/requests/cacert.pem
```

`your_ca_certificate.pem` is `/etc/ssl/certs/ca-certificates.crt` which can be obtained through

```
cat service.osrc | grep CACERT
```

7. Package a new ramdisk file. From within the `/tmp/new_deploy_ramdisk` folder, execute the following command:

```
find . | cpio --create --format='newc' | gzip -c -9 > /tmp/updated-ironic-
deploy.initramfs
```

Once completed, the new image can be found at `/tmp/updated-ironic-deploy.initramfs`, and will need to be uploaded to Glance. New Glance IDs will need to be recorded for any instances requiring this new image, as noted in the parent paragraph.

14.2 Ironic in Multiple Control Plane

SUSE OpenStack Cloud 8 introduces the concept of multiple control planes and multiple regions - see the Input Model documentation for the relevant *Book* "Planning an Installation with Cloud Lifecycle Manager"*, Chapter 6 "Input Model", Section 6.2 "Concepts", Section 6.2.2 "Control Planes", Section 6.2.2.1 "Control Planes and Regions"* and *Book* "Planning an Installation with Cloud Lifecycle Manager"*, Chapter 7 "Configuration Objects", Section 7.2 "Control Plane", Section 7.2.3 "Multiple Control Planes"*. This document covers the use of an Ironic region in a multiple control plane cloud model in SUSE OpenStack Cloud.

14.2.1 Networking for Baremetal in Multiple Control Plane

IRONIC-FLAT-NET is the network configuration for baremetal control plane.

You need to set the environment variable **OS_REGION_NAME** to the Ironic region in baremetal control plane. This will set up the Ironic flat networking in Neutron.

```
export OS_REGION_NAME=<ironic_region>
```

To see details of the `IRONIC-FLAT-NETWORK` created during configuration, use the following command:

```
neutron net-list
```

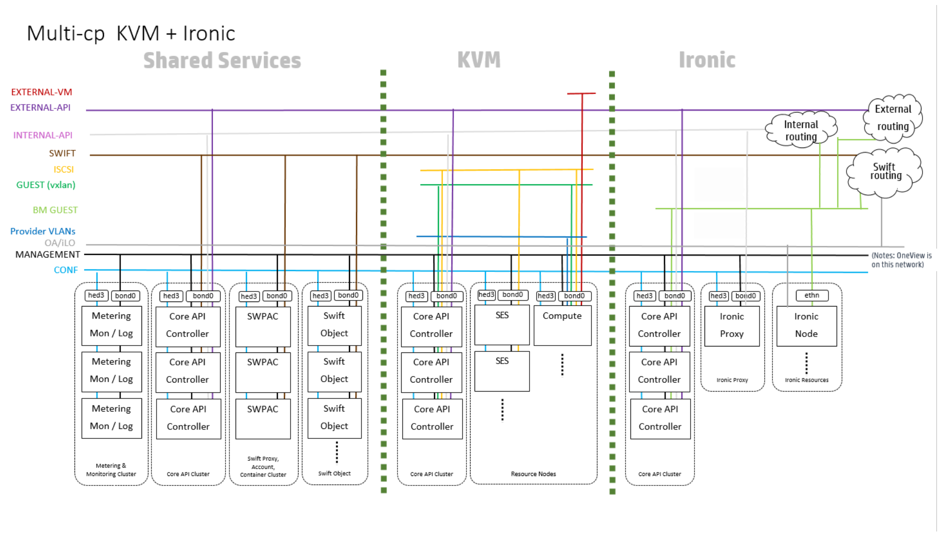


FIGURE 14.1: ARCHITECTURE OF MULTIPLE CONTROL PLANE WITH IRONIC

14.2.2 Handling Optional Swift Service

Swift is very resource-intensive and as a result, it is now optional in the SUSE OpenStack Cloud control plane. A number of services depend on Swift, and if it is not present, they must provide a fallback strategy. For example, Glance can use the filesystem in place of Swift for its backend store.

In Ironic, agent-based drivers require Swift. If it is not present, it is necessary to disable access to this Ironic feature in the control plane. The `enable_agent_driver` flag has been added to the Ironic configuration data and can have values of `true` or `false`. Setting this flag to `false` will disable Swift configurations and the agent based drivers in the Ironic control plane.

14.2.3 Instance Provisioning

In a multiple control plane cloud setup, changes for Glance container name in the Swift namespace of `ironic-conductor.conf` introduces a conflict with the one in `glance-api.conf`. Provisioning with agent-based drivers requires the container name to be the same in Ironic and Glance. Hence, on instance provisioning with agent-based drivers (Swift-enabled), the agent is not able to fetch the images from Glance store and fails at that point.

You can resolve this issue using the following steps:

1. Copy the value of `swift_store_container` from the file `/opt/stack/service/glance-api/etc/glance-api.conf`
2. Log in to the Cloud Lifecycle Manager and use the value for `swift_container` in glance namespace of `~/scratch/ansible/next/ardana/ansible/roles/ironic-common/templates/ironic-conductor.conf.j2`
3. Run the following playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

14.3 Provisioning Bare-Metal Nodes with Flat Network Model

Warning

Providing bare-metal resources to an untrusted third party is not advised as a malicious user can potentially modify hardware firmware.

Important

The steps outlined in *Section 14.1.7, “TLS Certificates with Ironic Python Agent (IPA) Images”* *must* be performed.

A number of drivers are available to provision and manage bare-metal machines. The drivers are named based on the deployment mode and the power management interface. SUSE OpenStack Cloud has been tested with the following drivers:

- agent_ilo
- agent_ipmi
- pxe_ilo
- pxe_ipmi

Before you start, you should be aware that:

1. Node Cleaning is enabled for all the drivers in SUSE OpenStack Cloud 8.
2. Node parameter settings must have matching flavors in terms of `cpus`, `local_gb`, and `memory_mb`, `boot_mode` and `cpu_arch`.
3. It is advisable that nodes enrolled for ipmitool drivers are pre-validated in terms of BIOS settings, in terms of boot mode, prior to setting capabilities.
4. Network cabling and interface layout should also be pre-validated in any given particular boot mode or configuration that is registered.
5. The use of `agent_` drivers is predicated upon Glance images being backed by a Swift image store, specifically the need for the temporary file access features. Using the file system as a Glance back-end image store means that the `agent_` drivers cannot be used.
6. Manual Cleaning (RAID) and Node inspection is supported by ilo drivers (`agent_ilo` and `pxe_ilo)`

14.3.1 Supplied Images

As part of the SUSE OpenStack Cloud Entry-scale Ironic Cloud installation, Ironic Python Agent (IPA) images are supplied and loaded into Glance. To see the images that have been loaded, execute the following commands on the deployer node:

```
$ source ~/service.osrc
glance image-list
```

This will produce output like the following example, showing three images that have been added by Ironic:

```
+------------------+-------------------------+----------------------------+
| ID               | Name                    |                            |
+------------------+-------------------------+----------------------------+
| DEPLOY_UUID      | ir-deploy-iso-ARDANA5.0     |
| KERNEL_UUID      | ir-deploy-kernel-ARDANA5.0  |
| RAMDISK_UUID     | ir-deploy-ramdisk-ARDANA5.0 |
+------------------+-------------------------+----------------------------+
```

The `ir-deploy-ramdisk` image is a traditional boot ramdisk used by the `agent_ipmitool`, `pxe_ipmitool`, and `pxe_ilo` drivers while `ir-deploy-iso` is an ISO image that is supplied as virtual media to the host when using the `agent_ilo` driver.

14.3.2 Provisioning a Node

The information required to provision a node varies slightly depending on the driver used. In general the following details are required.

- Network access information and credentials to connect to the management interface of the node.
- Sufficient properties to allow for Nova flavor matching.
- A deployment image to perform the actual deployment of the guest operating system to the bare-metal node.

A combination of the `ironic node-create` and `ironic node-update` commands are used for registering a node's characteristics with the Ironic service. In particular, `ironic node-update <nodeid> add` and `ironic node-update <nodeid> replace` can be used to modify the properties of a node after it has been created while `ironic node-update <nodeid> remove` will remove a property.

14.3.3 Creating a Node Using **`agent_ilo`**

If you want to use a boot mode of BIOS as opposed to UEFI, then you need to ensure that the boot mode has been set correctly on the IPMI:

While the iLO driver can automatically set a node to boot in UEFI mode via the `boot_mode` defined capability, it cannot set BIOS boot mode once UEFI mode has been set.

Use the `ironic node-create` command to specify the `agent_ilo` driver, network access and credential information for the IPMI, properties of the node and the Glance ID of the supplied ISO IPA image. Note that memory size is specified in megabytes while disk size is specified in gigabytes.

```
ironic node-create -d agent_ilo -i ilo_address=IP_ADDRESS -i \
  ilo_username=Administrator -i ilo_password=PASSWORD \
  -p cpus=2 -p cpu_arch=x86_64 -p memory_mb=64000 -p local_gb=99 \
  -i ilo_deploy_iso=DEPLOY_UUID
```

This will generate output similar to the following:

```
+--------------+---------------------------------------------------------------+
| Property     | Value                                                         |
+--------------+---------------------------------------------------------------+
| uuid         | NODE_UUID                                                     |
| driver_info  | {u'ilo_address': u'IP_ADDRESS', u'ilo_password': u'******',   |
|              | u'ilo_deploy_iso': u'DEPLOY_UUID',                            |
|              | u'ilo_username': u'Administrator'}                            |
| extra        | {}                                                            |
| driver       | agent_ilo                                                     |
| chassis_uuid |                                                               |
| properties   | {u'memory_mb': 64000, u'local_gb': 99, u'cpus': 2,            |
|              | u'cpu_arch': u'x86_64'}                                       |
| name         | None                                                          |
+--------------+---------------------------------------------------------------+
```

Now update the node with `boot_mode` and `boot_option` properties:

```
ironic node-update NODE_UUID add \
  properties/capabilities="boot_mode:bios,boot_option:local"
```

The `ironic node-update` command returns details for all of the node's characteristics.

```
+------------------------+------------------------------------------------------------------+
| Property               | Value                                                            |
+------------------------+------------------------------------------------------------------+
| target_power_state     | None                                                             |
| extra                  | {}                                                               |
| last_error             | None                                                             |
| updated_at             | None                                                             |
| maintenance_reason     | None                                                             |
| provision_state        | available                                                        |
| clean_step             | {}                                                               |
| uuid                   | NODE_UUID                                                        |
| console_enabled        | False                                                            |
| target_provision_state | None                                                          |
| provision_updated_at   | None                                                          |
| maintenance            | False                                                         |
| inspection_started_at  | None                                                          |
| inspection_finished_at | None                                                          |
| power_state            | None                                                          |
| driver                 | agent_ilo                                                     |
| reservation            | None                                                          |
| properties             | {u'memory_mb': 64000, u'cpu_arch': u'x86_64', u'local_gb': 99,   |
|                        | u'cpus': 2, u'capabilities': u'boot_mode:bios,boot_option:local'}|
| instance_uuid          | None                                                          |
| name                   | None                                                          |
| driver_info            | {u'ilo_address': u'10.1.196.117', u'ilo_password': u'******',    |
|                        | u'ilo_deploy_iso': u'DEPLOY_UUID',                            |
|                        | u'ilo_username': u'Administrator'}                            |
| created_at             | 2016-03-11T10:17:10+00:00                                     |
| driver_internal_info   | {}                                                            |
| chassis_uuid           |                                                               |
| instance_info          | {}                                                            |
+------------------------+---------------------------------------------------------------+
```

14.3.4 Creating a Node Using **agent_ipmi**

Use the `ironic node-create` command to specify the `agent_ipmi` driver, network access and credential information for the IPMI, properties of the node and the Glance IDs of the supplied kernel and ramdisk images. Note that memory size is specified in megabytes while disk size is specified in gigabytes.

```
ironic node-create -d agent_ipmitool \
  -i ipmi_address=IP_ADDRESS \
  -i ipmi_username=Administrator -i ipmi_password=PASSWORD \
  -p cpus=2 -p memory_mb=64000 -p local_gb=99 -p cpu_arch=x86_64 \
  -i deploy_kernel=KERNEL_UUID \
  -i deploy_ramdisk=RAMDISK_UUID
```

This will generate output similar to the following:

```
+--------------+-----------------------------------------------------------------------+
| Property     | Value                                                                 |
+--------------+-----------------------------------------------------------------------+
| uuid         | NODE2_UUID                                                            |
| driver_info  | {u'deploy_kernel': u'KERNEL_UUID',                                    |
|              | u'ipmi_address': u'IP_ADDRESS', u'ipmi_username': u'Administrator',   |
|              | u'ipmi_password': u'******',                                          |
|              | u'deploy_ramdisk': u'RAMDISK_UUID'}                                   |
| extra        | {}                                                                    |
| driver       | agent_ipmitool                                                        |
| chassis_uuid |                                                                  |
| properties   | {u'memory_mb': 64000, u'cpu_arch': u'x86_64', u'local_gb': 99,   |
|              | u'cpus': 2}                                                      |
| name         | None                                                             |
+--------------+------------------------------------------------------------------+
```

Now update the node with `boot_mode` and `boot_option` properties:

```
ironic node-update NODE_UUID add \
  properties/capabilities="boot_mode:bios,boot_option:local"
```

The `ironic node-update` command returns details for all of the node's characteristics.

```
+------------------------+------------------------------------------------------------------+
| Property               | Value                                                            |
+------------------------+------------------------------------------------------------------+
| target_power_state     | None                                                             |
| extra                  | {}                                                               |
| last_error             | None                                                             |
| updated_at             | None                                                             |
| maintenance_reason     | None                                                             |
| provision_state        | available                                                        |
| clean_step             | {}                                                               |
| uuid                   | NODE2_UUID                                                       |
| console_enabled        | False                                                            |
| target_provision_state | None                                                             |
| provision_updated_at   | None                                                             |
| maintenance            | False                                                            |
| inspection_started_at  | None                                                             |
| inspection_finished_at | None                                                             |
| power_state            | None                                                             |
| driver                 | agent_ipmitool                                                   |
| reservation            | None                                                             |
| properties             | {u'memory_mb': 64000, u'cpu_arch': u'x86_64',                    |
|                        | u'local_gb': 99, u'cpus': 2,                                     |
|                        | u'capabilities': u'boot_mode:bios,boot_option:local'}            |
| instance_uuid          | None                                                             |
| name                   | None                                                             |
| driver_info            | {u'ipmi_password': u'******', u'ipmi_address': u'IP_ADDRESS',    |
|                        | u'ipmi_username': u'Administrator', u'deploy_kernel':            |
|                        | u'KERNEL_UUID',                                                  |
|                        | u'deploy_ramdisk': u'RAMDISK_UUID'}                              |
| created_at             | 2016-03-11T14:19:18+00:00                                        |
| driver_internal_info   | {}                                                               |
| chassis_uuid           |                                                                  |
| instance_info          | {}                                                               |
+------------------------+------------------------------------------------------------------+
```

For more information on node enrollment, see the OpenStack documentation at http://docs.openstack.org/developer/ironic/deploy/install-guide.html#enrollment.

14.3.5 Creating a Flavor

Nova uses flavors when fulfilling requests for bare-metal nodes. The Nova scheduler attempts to match the requested flavor against the properties of the created Ironic nodes. So an administrator needs to set up flavors that correspond to the available bare-metal nodes using the command **`nova flavor-create`**:

```
nova flavor-create bmtest auto 64000  99 2

+----------------+--------+--------+------+-----------+------+-------+-------------+-----------+
| ID             | Name   | Mem_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----------------+--------+--------+------+-----------+------+-------+-------------+-----------+
| 645de0...b1348 | bmtest | 64000  | 99   | 0         |      | 2     | 1.0         | True      |
+----------------+--------+--------+------+-----------+------+-------+-------------+-----------+
```

To see a list of all the available flavors, run **`nova flavor-list`**:

```
nova flavor-list

+-------------+--------------+--------+------+-----------+------+-------+--------+-----------+
| ID          | Name         | Mem_MB | Disk | Ephemeral | Swap | VCPUs |  RXTX  | Is_Public |
|             |              |        |      |           |      |       | Factor |           |
+-------------+--------------+--------+------+-----------+------+-------+--------+-----------+
| 1           | m1.tiny      | 512    | 1    | 0         |      | 1     | 1.0    | True      |
| 2           | m1.small     | 2048   | 20   | 0         |      | 1     | 1.0    | True      |
| 3           | m1.medium    | 4096   | 40   | 0         |      | 2     | 1.0    | True      |
| 4           | m1.large     | 8192   | 80   | 0         |      | 4     | 1.0    | True      |
| 5           | m1.xlarge    | 16384  | 160  | 0         |      | 8     | 1.0    | True      |
| 6           | m1.baremetal | 4096   | 80   | 0         |      | 2     | 1.0    | True      |
| 645d...1348 | bmtest       | 64000  | 99   | 0         |      | 2     | 1.0    | True      |
+-------------+--------------+--------+------+-----------+------+-------+--------+-----------+
```

Now set the CPU architecture and boot mode and boot option capabilities:

```
nova flavor-key 645de08d-2bc6-43f1-8a5f-2315a75b1348 set cpu_arch=x86_64
nova flavor-key 645de08d-2bc6-43f1-8a5f-2315a75b1348 set capabilities:boot_option="local"
nova flavor-key 645de08d-2bc6-43f1-8a5f-2315a75b1348 set capabilities:boot_mode="bios"
```

For more information on flavor creation, see the OpenStack documentation at http://docs.openstack.org/developer/ironic/deploy/install-guide.html#flavor-creation ↗.

14.3.6 Creating a Network Port

Register the MAC addresses of all connected physical network interfaces intended for use with the bare-metal node.

```
ironic port-create -a 5c:b9:01:88:f0:a4 -n ea7246fd-e1d6-4637-9699-0b7c59c22e67
```

14.3.7 Creating a Glance Image

You can create a complete disk image using the instructions at *Section 14.3.11, “Building Glance Images Using KIWI”*.

The image you create can then be loaded into Glance:

```
glance image-create --name='leap' --disk-format=raw \
  --container-format=bare \
  --file /tmp/myimage/LimeJeOS-Leap-42.3.x86_64-1.42.3.raw

+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 45a4a06997e64f7120795c68beeb0e3c     |
| container_format | bare                                 |
| created_at       | 2018-02-17T10:42:14Z                 |
| disk_format      | raw                                  |
| id               | 17e4915a-ada0-4b95-bacf-ba67133f39a7 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | leap                                 |
| owner            | 821b7bb8148f439191d108764301af64     |
| protected        | False                                |
| size             | 372047872                            |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2018-02-17T10:42:23Z                 |
| virtual_size     | None                                 |
| visibility       | private                              |
+------------------+--------------------------------------+
```

This image will subsequently be used to boot the bare-metal node.

14.3.8 Generating a Key Pair

Create a key pair that you will use when you login to the newly booted node:

```
nova keypair-add ironic_kp > ironic_kp.pem
```

14.3.9 Determining the Neutron Network ID

```
neutron net-list
```

```
+---------------+----------+-------------------------------------------------+
| id            | name     | subnets                                         |
+---------------+----------+-------------------------------------------------+
| c0102...1ca8c | flat-net | 709ee2a1-4110-4b26-ba4d-deb74553adb9 192.3.15.0/24 |
+---------------+----------+-------------------------------------------------+
```

14.3.10 Booting the Node

Before booting, it is advisable to power down the node:

```
ironic node-set-power-state ea7246fd-e1d6-4637-9699-0b7c59c22e67 off
```

You can now boot the bare-metal node with the information compiled in the preceding steps, using the Neutron network ID, the whole disk image ID, the matching flavor and the key name:

```
nova boot --nic net-id=c010267c-9424-45be-8c05-99d68531ca8c \
  --image 17e4915a-ada0-4b95-bacf-ba67133f39a7 \
  --flavor 645de08d-2bc6-43f1-8a5f-2315a75b1348 \
  --key-name ironic_kp leap
```

This command returns information about the state of the node that is booting:

```
+--------------------------------------+------------------------+
| Property                             | Value                  |
+--------------------------------------+------------------------+
| OS-EXT-AZ:availability_zone          |                        |
| OS-EXT-SRV-ATTR:host                 | -                      |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                      |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000001      |
| OS-EXT-STS:power_state               | 0                      |
| OS-EXT-STS:task_state                | scheduling             |
| OS-EXT-STS:vm_state                  | building               |
| OS-SRV-USG:launched_at               | -                      |
| OS-SRV-USG:terminated_at             | -                      |
| accessIPv4                           |                        |
| accessIPv6                           |                        |
| adminPass                            | adpHw3KKTjHk           |
| config_drive                         |                        |
| created                              | 2018-03-11T11:00:28Z   |
| flavor                               | bmtest (645de...b1348) |
| hostId                               |                        |
| id                                   | a9012...3007e          |
| image                                | leap (17e49...f39a7)   |
| key_name                             | ironic_kp              |
| metadata                             | {}                     |
| name                                 | leap                   |
| os-extended-volumes:volumes_attached | []                     |
| progress                             | 0                      |
| security_groups                      | default                |
| status                               | BUILD                  |
| tenant_id                            | d53bcaf...baa60dd      |
| updated                              | 2016-03-11T11:00:28Z   |
| user_id                              | e580c64...4aaf990      |
+--------------------------------------+------------------------+
```

The boot process can take up to 10 minutes. Monitor the progress with the IPMI console or with `nova list`, `nova show <nova_node_id>`, and `ironic node-show <ironic_node_id>` commands.

```
nova list

+---------------+--------+--------+------------+-------------+----------------------+
| ID            | Name   | Status | Task State | Power State | Networks             |
+---------------+--------+--------+------------+-------------+----------------------+
| a9012...3007e | leap   | BUILD  | spawning   | NOSTATE     | flat-net=192.3.15.12 |
+---------------+--------+--------+------------+-------------+----------------------+
```

During the boot procedure, a login prompt will appear for SLES:

Ignore this login screen and wait for the login screen of your target operating system to appear:

If you now run the command **`nova list`**, it should show the node in the running state:

```
nova list
+---------------+--------+--------+------------+-------------+----------------------+
| ID            | Name   | Status | Task State | Power State | Networks             |
+---------------+--------+--------+------------+-------------+----------------------+
| a9012...3007e | leap   | ACTIVE | -          | Running     | flat-net=192.3.15.14 |
+---------------+--------+--------+------------+-------------+----------------------+
```

You can now log in to the booted node using the key you generated earlier. (You may be prompted to change the permissions of your private key files, so that they are not accessible by others).

```
ssh leap@192.3.15.14 -i ironic_kp.pem
```

14.3.11 Building Glance Images Using KIWI

The following sections show you how to create your own images using KIWI, the command line utility to build Linux system appliances. For information on installing KIWI, see https://suse.github.io/kiwi/installation.html.

KIWI creates images in a two-step process:

1. The `prepare` operation generates an unpacked image tree using the information provided in the image description.
2. The `create` operation creates the packed image based on the unpacked image and the information provided in the configuration file (`config.xml`).

Instructions for installing KIWI are available at https://suse.github.io/kiwi/installation.html.

Image creation with KIWI is automated and does not require any user interaction. The information required for the image creation process is provided by the image description.

To use and run KIWI requires:

- A recent Linux distribution such as:
  - openSUSE Leap 42.3
  - SUSE Linux Enterprise 12 SP3
  - openSUSE Tumbleweed
- Enough free disk space to build and store the image (a minimum of 10 GB is recommended).
- Python version 2.7, 3.4 or higher. KIWI supports both Python 2 and 3 versions
- Git (package `git-core`) to clone a repository.
- Virtualization technology to start the image (QEMU is recommended).

14.3.12 Creating an openSUSE Image with KIWI

The following example shows how to build an openSUSE Leap image that is ready to run in QEMU.

1. Retrieve the example image descriptions.

```
git clone https://github.com/SUSE/kiwi-descriptions
```

2. Build the image with KIWI:

```
sudo kiwi-ng --type vmx system build \
  --description kiwi-descriptions/suse/x86_64/suse-leap-42.3-JeOS \
  --target-dir /tmp/myimage
```

A `.raw` image will be built in the `/tmp/myimage` directory.

3. Test the live image with QEMU:

```
qemu \
  -drive file=LimeJeOS-Leap-42.3.x86_64-1.42.3.raw,format=raw,if=virtio \
  -m 4096
```

4. With a successful test, the image is complete.

By default, KIWI generates a file in the `.raw` format. The `.raw` file is a disk image with a structure equivalent to a physical hard disk. `.raw` images are supported by any hypervisor, but are not compressed and do not offer the best performance.

Virtualization systems support their own formats (such as `qcow2` or `vmdk`) with compression and improved I/O performance. To build an image in a format other than `.raw`, add the format attribute to the type definition in in the preferences section of `config.xml`. Using `qcow2` for example:

```
<image ...>
  <preferences>
    <type format="qcow2" .../>
    ...
  </preferences>
  ...
</image
```

More information about KIWI is at https://suse.github.io/kiwi/ ↗.

14.4 Provisioning Baremetal Nodes with Multi-Tenancy

1. Create a network and a subnet:

```
$ neutron net-create guest-net-1
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2017-06-10T02:49:56Z                 |
| description               |                                      |
| id                        | 256d55a6-9430-4f49-8a4c-cc5192f5321e |
| ipv4_address_scope        |                                      |
| ipv6_address_scope        |                                      |
| mtu                       | 1500                                 |
| name                      | guest-net-1                          |
| project_id                | 57b792cdcdd74d16a08fc7a396ee05b6     |
| provider:network_type     | vlan                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  | 1152                                 |
| revision_number           | 2                                    |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | 57b792cdcdd74d16a08fc7a396ee05b6     |
| updated_at                | 2017-06-10T02:49:57Z                 |
+---------------------------+--------------------------------------+

$ neutron  subnet-create guest-net-1 200.0.0.0/24
Created a new subnet:
+-------------------+----------------------------------------------+
| Field             | Value                                        |
+-------------------+----------------------------------------------+
| allocation_pools  | {"start": "200.0.0.2", "end": "200.0.0.254"} |
| cidr              | 200.0.0.0/24                                 |
| created_at        | 2017-06-10T02:53:08Z                         |
| description       |                                              |
| dns_nameservers   |                                              |
| enable_dhcp       | True                                         |
| gateway_ip        | 200.0.0.1                                    |
| host_routes       |                                              |
| id                | 53accf35-ae02-43ae-95d8-7b5efed18ae9         |
| ip_version        | 4                                            |
| ipv6_address_mode |                                              |
| ipv6_ra_mode      |                                              |
| name              |                                              |
| network_id        | 256d55a6-9430-4f49-8a4c-cc5192f5321e         |
| project_id        | 57b792cdcdd74d16a08fc7a396ee05b6             |
| revision_number   | 2                                            |
| service_types     |                                              |
| subnetpool_id     |                                              |
| tenant_id         | 57b792cdcdd74d16a08fc7a396ee05b6             |
| updated_at        | 2017-06-10T02:53:08Z                         |
+-------------------+----------------------------------------------+
```

2. Review glance image list

```
$ glance image-list
+--------------------------------------+--------------------------+
| ID                                   | Name                     |
+--------------------------------------+--------------------------+
| 0526d2d7-c196-4c62-bfe5-a13bce5c7f39 | cirros-0.4.0-x86_64      |
+--------------------------------------+--------------------------+
```

3. Create Ironic node

```
$ ironic --ironic-api-version 1.22 node-create -d agent_ipmitool \
  -n test-node-1 -i ipmi_address=192.168.9.69 -i ipmi_username=ipmi_user \
  -i ipmi_password=XXXXXXXX --network-interface neutron -p  memory_mb=4096 \
  -p cpu_arch=x86_64 -p local_gb=80 -p cpus=2 \
  -p capabilities=boot_mode:bios,boot_option:local \
  -p root_device='{"name":"/dev/sda"}' \
  -i deploy_kernel=db3d131f-2fb0-4189-bb8d-424ee0886e4c \
  -i deploy_ramdisk=304cae15-3fe5-4f1c-8478-c65da5092a2c

+-------------------+-------------------------------------------------------------------+
| Property          | Value                                                             |
+-------------------+-------------------------------------------------------------------+
| chassis_uuid      |                                                                   |
| driver            | agent_ipmitool                                                    |
| driver_info       | {u'deploy_kernel': u'db3d131f-2fb0-4189-bb8d-424ee0886e4c',       |
|                   | u'ipmi_address': u'192.168.9.69',                                 |
|                   | u'ipmi_username': u'gozer', u'ipmi_password': u'******',          |
|                   | u'deploy_ramdisk': u'304cae15-3fe5-4f1c-8478-c65da5092a2c'}       |
| extra             | {}                                                                |
| name              | test-node-1                                                       |
| network_interface | neutron                                                           |
| properties        | {u'cpu_arch': u'x86_64', u'root_device': {u'name': u'/dev/sda'},  |
|                   | u'cpus': 2, u'capabilities': u'boot_mode:bios,boot_option:local', |
|                   | u'memory_mb': 4096, u'local_gb': 80}                              |
| resource_class    | None                                                              |
| uuid              | cb4dda0d-f3b0-48b9-ac90-ba77b8c66162                              |
+-------------------+-------------------------------------------------------------------+
```

ipmi_address, ipmi_username and ipmi_password are IPMI access parameters for baremetal Ironic node. Adjust memory_mb, cpus, local_gb to your node size requirements. They also need to be reflected in flavor setting (see below). Use capabilities boot_mode:bios for baremetal nodes operating in Legacy BIOS mode. For UEFI baremetal nodes, use boot_mode:uefi lookup deploy_kernel and deploy_ramdisk in glance image list output above.

Important

Since we are using Ironic API version 1.22, node is created initial state **enroll**. It needs to be explicitly moved to **available** state. This behavior changed in API version 1.11

4. Create port

```
$ ironic --ironic-api-version 1.22 port-create --address f0:92:1c:05:6c:40 \
  --node cb4dda0d-f3b0-48b9-ac90-ba77b8c66162 -l switch_id=e8:f7:24:bf:07:2e -l \
  switch_info=hp59srv1-a-11b -l port_id="Ten-GigabitEthernet 1/0/34" \
  --pxe-enabled true
+-----------------------+-------------------------------------------+
| Property              | Value                                     |
+-----------------------+-------------------------------------------+
| address               | f0:92:1c:05:6c:40                         |
| extra                 | {}                                        |
| local_link_connection | {u'switch_info': u'hp59srv1-a-11b',       |
|                       | u'port_id': u'Ten-GigabitEthernet 1/0/34', |
|                       | u'switch_id': u'e8:f7:24:bf:07:2e'}       |
| node_uuid             | cb4dda0d-f3b0-48b9-ac90-ba77b8c66162      |
| pxe_enabled           | True                                      |
| uuid                  | a49491f3-5595-413b-b4a7-bb6f9abec212      |
+-----------------------+-------------------------------------------+
```

- for `--address`, use MAC of 1st NIC of ironic baremetal node, which will be used for PXE boot
- for `--node`, use ironic node uuid (see above)
- for `-l switch_id`, use switch management interface MAC address. It can be retrieved by pinging switch management IP and looking up MAC address in 'arp -l -n' command output.
- for `-l switch_info`, use switch_id from `data/ironic/ironic_config.yml` file. If you have several switch config definitions, use the right switch your baremetal node is connected to.
- for -l port_id, use port ID on the switch

5. Move ironic node to manage and then available state

```
$ ironic node-set-provision-state test-node-1 manage
$ ironic node-set-provision-state test-node-1 provide
```

6. Once node is successfully moved to available state, its resources should be included into Nova hypervisor statistics

```
$ nova hypervisor-stats
+----------------------+-------+
| Property             | Value |
+----------------------+-------+
| count                | 1     |
| current_workload     | 0     |
| disk_available_least | 80    |
| free_disk_gb         | 80    |
| free_ram_mb          | 4096  |
| local_gb             | 80    |
| local_gb_used        | 0     |
| memory_mb            | 4096  |
| memory_mb_used       | 0     |
| running_vms          | 0     |
| vcpus                | 2     |
| vcpus_used           | 0     |
+----------------------+-------+
```

7. Prepare a keypair, which will be used for logging into the node

```
$ nova keypair-add ironic_kp > ironic_kp.pem
```

8. Obtain user image and upload it to glance. Please refer to OpenStack documentation on user image creation: https://docs.openstack.org/project-install-guide/baremetal/draft/configure-glance-images.html.

Note

Deployed images are already populated by SUSE OpenStack Cloud installer.

```
$ glance image-create --name='Ubuntu Trusty 14.04' --disk-format=qcow2 \
  --container-format=bare --file ~/ubuntu-trusty.qcow2
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | d586d8d2107f328665760fee4c81caf0     |
| container_format | bare                                 |
| created_at       | 2017-06-13T22:38:45Z                 |
| disk_format      | qcow2                                |
| id               | 9fdd54a3-ccf5-459c-a084-e50071d0aa39 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | Ubuntu Trusty 14.04                  |
| owner            | 57b792cdcdd74d16a08fc7a396ee05b6     |
| protected        | False                                |
| size             | 371508736                            |
| status           | active                               |
| tags             | []                                   |
| updated_at       | 2017-06-13T22:38:55Z                 |
| virtual_size     | None                                 |
| visibility       | private                              |
+------------------+--------------------------------------+

$ glance image-list
+--------------------------------------+--------------------------+
| ID                                   | Name                     |
+--------------------------------------+--------------------------+
| 0526d2d7-c196-4c62-bfe5-a13bce5c7f39 | cirros-0.4.0-x86_64      |
| 83eecf9c-d675-4bf9-a5d5-9cf1fe9ee9c2 | ir-deploy-iso-EXAMPLE    |
| db3d131f-2fb0-4189-bb8d-424ee0886e4c | ir-deploy-kernel-EXAMPLE |
| 304cae15-3fe5-4f1c-8478-c65da5092a2c | ir-deploy-ramdisk- EXAMPLE |
| 9fdd54a3-ccf5-459c-a084-e50071d0aa39 | Ubuntu Trusty 14.04      |
+--------------------------------------+--------------------------+
```

9. Create a baremetal flavor and set flavor keys specifying requested node size, architecture and boot mode. A flavor can be re-used for several nodes having the same size, architecture and boot mode

```
$ nova flavor-create m1.ironic auto 4096 80 2
+-------------+-----------+--------+------+---------+------+-------+-------------
+-----------+
| ID          | Name      | Mem_MB | Disk | Ephemrl | Swap | VCPUs | RXTX_Factor |
 Is_Public |
+-------------+-----------+--------+------+---------+------+-------+-------------
+-----------+
| ab69...87bf | m1.ironic | 4096   | 80   | 0       |      | 2     | 1.0         | True
  |
+-------------+-----------+--------+------+---------+------+-------+-------------
+-----------+

$ nova flavor-key ab6988...e28694c87bf set cpu_arch=x86_64
$ nova flavor-key ab6988...e28694c87bf set capabilities:boot_option="local"
$ nova flavor-key ab6988...e28694c87bf set capabilities:boot_mode="bios"
```

Parameters must match parameters of ironic node above. Use `capabilities:boot_mode="bios"` for Legacy BIOS nodes. For UEFI nodes, use `capabilities:boot_mode="uefi"`

10. Boot the node

```
$ nova boot --flavor m1.ironic --image 9fdd54a3-ccf5-459c-a084-e50071d0aa39 \
  --key-name ironic_kp --nic net-id=256d55a6-9430-4f49-8a4c-cc5192f5321e \
  test-node-1
+--------------------------------------+-------------------------------------------------+
| Property                             | Value                                           |
+--------------------------------------+-------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                          |
| OS-EXT-AZ:availability_zone          |                                                 |
| OS-EXT-SRV-ATTR:host                 | -                                               |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                               |
| OS-EXT-SRV-ATTR:instance_name        |                                                 |
| OS-EXT-STS:power_state               | 0                                               |
| OS-EXT-STS:task_state                | scheduling                                      |
| OS-EXT-STS:vm_state                  | building                                        |
| OS-SRV-USG:launched_at               | -                                               |
| OS-SRV-USG:terminated_at             | -                                               |
| accessIPv4                           |                                                 |
| accessIPv6                           |                                                 |
| adminPass                            | XXXXXXXXXXXX                                    |
| config_drive                         |                                                 |
| created                              | 2017-06-14T21:25:18Z                            |
| flavor                               | m1.ironic (ab69881...5a-497d-93ae-6e28694c87bf) |
| hostId                               |                                                 |
| id                                   | f1a8c63e-da7b-4d9a-8648-b1baa6929682            |
| image                                | Ubuntu Trusty 14.04 (9fdd54a3-ccf5-4a0...0aa39) |
| key_name                             | ironic_kp                                       |
| metadata                             | {}                                              |
| name                                 | test-node-1                                     |
| os-extended-volumes:volumes_attached | []                                              |
| progress                             | 0                                               |
| security_groups                      | default                                         |
| status                               | BUILD                                           |
| tenant_id                            | 57b792cdcdd74d16a08fc7a396ee05b6                |
| updated                              | 2017-06-14T21:25:17Z                            |
| user_id                              | cc76d7469658401fbd4cf772278483d9                |
+--------------------------------------+-------------------------------------------------+
```

- for `--image`, use the ID of user image created at previous step
- for `--nic net-id`, use ID of the tenant network created at the beginning

Note

During the node provisioning, the following is happening in the background:

Neutron connects to switch management interfaces and assigns provisioning VLAN to baremetal node port on the switch. Ironic powers up the node using IPMI interface. Node is booting IPA image via PXE. IPA image is writing provided user image onto specified root device (`/dev/sda`) and powers node down. Neutron connects to switch management interfaces and assigns tenant VLAN to baremetal node port on the switch. A VLAN ID is selected from provided range. Ironic powers up the node using IPMI interface. Node is booting user image from disk.

11. Once provisioned, node will join the private tenant network. Access to private network from other networks is defined by switch configuration.

14.5 View Ironic System Details

14.5.1 View details about the server using **`nova show <nova-node-id>`**

```
nova show a90122ce-bba8-496f-92a0-8a7cb143007e

+--------------------------------------+-----------------------------------------------+
| Property                             | Value                                         |
+--------------------------------------+-----------------------------------------------+
| OS-EXT-AZ:availability_zone          | nova                                          |
| OS-EXT-SRV-ATTR:host                 | ardana-cp1-ir-compute0001-mgmt                |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | ea7246fd-e1d6-4637-9699-0b7c59c22e67          |
| OS-EXT-SRV-ATTR:instance_name        | instance-0000000a                             |
| OS-EXT-STS:power_state               | 1                                             |
| OS-EXT-STS:task_state                | -                                             |
| OS-EXT-STS:vm_state                  | active                                        |
| OS-SRV-USG:launched_at               | 2016-03-11T12:26:25.000000                    |
| OS-SRV-USG:terminated_at             | -                                             |
| accessIPv4                           |                                               |
| accessIPv6                           |                                               |
| config_drive                         |                                               |
| created                              | 2016-03-11T12:17:54Z                          |
| flat-net network                     | 192.3.15.14                                   |
| flavor                               | bmtest (645de08d-2bc6-43f1-8a5f-2315a75b1348) |
| hostId                               | ecafa4f40eb5f72f7298...3bad47cbc01aa0a076114f |
| id                                   | a90122ce-bba8-496f-92a0-8a7cb143007e          |
| image                                | ubuntu (17e4915a-ada0-4b95-bacf-ba67133f39a7) |
| key_name                             | ironic_kp                                     |
| metadata                             | {}                                            |
| name                                 | ubuntu                                        |
| os-extended-volumes:volumes_attached | []                                            |
| progress                             | 0                                             |
| security_groups                      | default                                       |
| status                               | ACTIVE                                        |
| tenant_id                            | d53bcaf15afb4cb5aea3adaedbaa60dd              |
| updated                              | 2016-03-11T12:26:26Z                          |
| user_id                              | e580c645bfec4faeadef7dbd24aaf990              |
+--------------------------------------+-----------------------------------------------+
```

14.5.2 View detailed information about a node using `ironic node-show <ironic-node-id>`

```
ironic node-show  ea7246fd-e1d6-4637-9699-0b7c59c22e67

+------------------------+--------------------------------------------------------------------------+
| Property               | Value                                                                    |
+------------------------+--------------------------------------------------------------------------+
| target_power_state     | None                                                                     |
| extra                  | {}                                                                       |
| last_error             | None                                                                     |
| updated_at             | 2016-03-11T12:26:25+00:00                                                |
| maintenance_reason     | None                                                                     |
| provision_state        | active                                                                   |
| clean_step             | {}                                                                       |
| uuid                   | ea7246fd-e1d6-4637-9699-0b7c59c22e67                                     |
| console_enabled        | False                                                                    |
| target_provision_state | None                                                                     |
| provision_updated_at   | 2016-03-11T12:26:25+00:00                                                |
| maintenance            | False                                                                    |
| inspection_started_at  | None                                                                     |
| inspection_finished_at | None                                                                     |
| power_state            | power on                                                                 |
| driver                 | agent_ilo                                                                |
| reservation            | None                                                                     |
| properties             | {u'memory_mb': 64000, u'cpu_arch': u'x86_64', u'local_gb': 99,           |
|                        | u'cpus': 2, u'capabilities': u'boot_mode:bios,boot_option:local'}        |
| instance_uuid          | a90122ce-bba8-496f-92a0-8a7cb143007e                                     |
| name                   | None                                                                     |
| driver_info            | {u'ilo_address': u'10.1.196.117', u'ilo_password': u'******',            |
|                        | u'ilo_deploy_iso': u'b9499494-7db3-4448-b67f-233b86489c1f',              |
|                        | u'ilo_username': u'Administrator'}                                       |
| created_at             | 2016-03-11T10:17:10+00:00                                                |
| driver_internal_info   | {u'agent_url': u'http://192.3.15.14:9999',                               |
|                        | u'is_whole_disk_image': True, u'agent_last_heartbeat': 1457699159}       |
| chassis_uuid           |                                                                          |
| instance_info          | {u'root_gb': u'99', u'display_name': u'ubuntu', u'image_source': u       |
|                        | '17e4915a-ada0-4b95-bacf-ba67133f39a7', u'capabilities': u'{"boot_mode": |
|                        | "bios", "boot_option": "local"}', u'memory_mb': u'64000', u'vcpus':      |
|                        | u'2', u'image_url': u'http://192.168.12.2:8080/v1/AUTH_ba121db7732f4ac3a |
|                        | 50cc4999a10d58d/glance/17e4915a-ada0-4b95-bacf-ba67133f39a7?temp_url_sig |
|                        | =ada691726337805981ac002c0fbfc905eb9783ea&temp_url_expires=1457699878',  |
|                        | u'image_container_format': u'bare', u'local_gb': u'99',                  |
|                        | u'image_disk_format': u'qcow2', u'image_checksum':                       |
|                        | u'2d7bb1e78b26f32c50bd9da99102150b', u'swap_mb': u'0'}                   |
+------------------------+--------------------------------------------------------------------------+
```

14.5.3 View detailed information about a port using **`ironic port-show <ironic-port-id>`**

```
ironic port-show a17a4ef8-a711-40e2-aa27-2189c43f0b67

+------------+-----------------------------------------------------------+
| Property   | Value                                                     |
+------------+-----------------------------------------------------------+
| node_uuid  | ea7246fd-e1d6-4637-9699-0b7c59c22e67                      |
| uuid       | a17a4ef8-a711-40e2-aa27-2189c43f0b67                      |
| extra      | {u'vif_port_id': u'82a5ab28-76a8-4c9d-bfb4-624aeb9721ea'} |
| created_at | 2016-03-11T10:40:53+00:00                                 |
| updated_at | 2016-03-11T12:17:56+00:00                                 |
| address    | 5c:b9:01:88:f0:a4                                         |
+------------+-----------------------------------------------------------+
```

14.5.4 View detailed information about a hypervisor using **`nova hypervisor-list`** and **`nova hypervisor-show`**

```
nova hypervisor-list

+-----+--------------------------------------+-------+---------+
| ID  | Hypervisor hostname                  | State | Status  |
+-----+--------------------------------------+-------+---------+
| 541 | ea7246fd-e1d6-4637-9699-0b7c59c22e67 | up    | enabled |
+-----+--------------------------------------+-------+---------+
```

```
nova hypervisor-show ea7246fd-e1d6-4637-9699-0b7c59c22e67

+-------------------------+--------------------------------------+
| Property                | Value                                |
+-------------------------+--------------------------------------+
| cpu_info                |                                      |
| current_workload        | 0                                    |
| disk_available_least    | 0                                    |
| free_disk_gb            | 0                                    |
| free_ram_mb             | 0                                    |
| host_ip                 | 192.168.12.6                         |
| hypervisor_hostname     | ea7246fd-e1d6-4637-9699-0b7c59c22e67 |
| hypervisor_type         | ironic                               |
| hypervisor_version      | 1                                    |
| id                      | 541                                  |
| local_gb                | 99                                   |
| local_gb_used           | 99                                   |
| memory_mb               | 64000                                |
| memory_mb_used          | 64000                                |
| running_vms             | 1                                    |
| service_disabled_reason | None                                 |
| service_host            | ardana-cp1-ir-compute0001-mgmt       |
| service_id              | 25                                   |
| state                   | up                                   |
| status                  | enabled                              |
| vcpus                   | 2                                    |
| vcpus_used              | 2                                    |
+-------------------------+--------------------------------------+
```

14.5.5 View a list of all running services using **nova service-list**

```
nova service-list

+----+------------------+----------------------+----------+---------+-------+------------+----------+
| Id | Binary           | Host                 | Zone     | Status  | State | Updated_at | Disabled |
|    |                  |                      |          |         |       |            | Reason   |
+----+------------------+----------------------+----------+---------+-------+------------+----------+
| 1  | nova-conductor   | ardana-cp1-c1-m1-mgmt | internal | enabled | up    | date:time  | -        |
| 7  | nova-conductor   |  " -cp1-c1-m2-mgmt    | internal | enabled | up    | date:time  | -        |
| 10 | nova-conductor   |  " -cp1-c1-m3-mgmt    | internal | enabled | up    | date:time  | -        |
| 13 | nova-scheduler   |  " -cp1-c1-m1-mgmt    | internal | enabled | up    | date:time  | -        |
| 16 | nova-scheduler   |  " -cp1-c1-m3-mgmt    | internal | enabled | up    | date:time  | -        |
| 19 | nova-scheduler   |  " -cp1-c1-m2-mgmt    | internal | enabled | up    | date:time  | -        |
| 22 | nova-consoleauth |  " -cp1-c1-m1-mgmt    | internal | enabled | up    | date:time  | -        |
| 25 | nova-compute     |  " -cp1-ir- | nova    |          | enabled | up    | date:time  | -        |
|    |                  |      compute0001-mgmt |          |         |       |            |          |
+----+------------------+----------------------+----------+---------+-------+------------+----------+
```

14.6 Troubleshooting Ironic Installation

Sometimes the `nova boot` command does not succeed and when you do a `nova list`, you will see output like the following:

```
ardana > nova list

+------------------+--------------+--------+------------+-------------+----------+
| ID               | Name         | Status | Task State | Power State | Networks |
+------------------+--------------+--------+------------+-------------+----------+
| ee08f82...624e5f | OpenSUSE42.3 | ERROR  | -          | NOSTATE     |          |
+------------------+--------------+--------+------------+-------------+----------+
```

You should execute the `nova show <nova-node-id>` and `ironic node-show <ironic-node-id>` commands to get more information about the error.

14.6.1 Error: No valid host was found.

The error `No valid host was found. There are not enough hosts.` is typically seen when performing the `nova boot` where there is a mismatch between the properties set on the node and the flavor used. For example, the output from a `nova show` command may look like this:

```
ardana > nova show ee08f82e-8920-4360-be51-a3f995624e5f

+------------------------+------------------------------------------------------------------------------+
| Property               | Value                                                                        |
+------------------------+------------------------------------------------------------------------------+
| OS-EXT-AZ:             |                                                                              |
|   availability_zone    |                                                                              |
| OS-EXT-SRV-ATTR:host   | -                                                                            |
| OS-EXT-SRV-ATTR:       |                                                                              |
|   hypervisor_hostname  | -                                                                            |
| OS-EXT-SRV-ATTR:       |                                                                              |
|   instance_name        | instance-00000001                                                            |
| OS-EXT-STS:power_state | 0                                                                            |
| OS-EXT-STS:task_state  | -                                                                            |
| OS-EXT-STS:vm_state    | error                                                                        |
| OS-SRV-USG:launched_at | -                                                                            |
| OS-SRV-USG:            |                                                                              |
|    terminated_at       | -                                                                            |
| accessIPv4             |                                                                              |
| accessIPv6             |                                                                              |
| config_drive           |                                                                              |
| created                | 2016-03-11T11:00:28Z                                                         |
| fault                  | {"message": "No valid host was found. There are not enough hosts             |
|                        |  available.", "code": 500, "details": "  File \"/opt/stack/                  |
|                        |  venv/nova-20160308T002421Z/lib/python2.7/site-packages/nova/                |
|                        |  conductor/manager.py\", line 739, in build_instances                   |
|                        |     request_spec, filter_properties)                                    |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/nova/scheduler/utils.py\", line 343, in wrapped         |
|                        |     return func(*args, **kwargs)                                        |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/nova/scheduler/client/__init__.py\", line 52,           |
|                        |     in select_destinations context, request_spec, filter_properties)    |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/nova/scheduler/client/__init__.py\",line 37,in __run_method |
|                        |     return getattr(self.instance, __name)(*args, **kwargs)              |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/nova/scheduler/client/query.py\", line 34,              |
|                        |     in select_destinations context, request_spec, filter_properties)    |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/nova/scheduler/rpcapi.py\", line 120, in select_destinations |
|                        |     request_spec=request_spec, filter_properties=filter_properties)     |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/oslo_messaging/rpc/client.py\", line 158, in call       |
|                        |     retry=self.retry)                                                   |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/oslo_messaging/transport.py\", line 90, in _send        |
|                        |     timeout=timeout, retry=retry)                                       |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/oslo_messaging/_drivers/amqpdriver.py\", line 462, in send  |
|                        |     retry=retry)                                                        |
|                        |   File \"/opt/stack/venv/nova-20160308T002421Z/lib/python2.7/           |
|                        |   site-packages/oslo_messaging/_drivers/amqpdriver.py\", line 453, in _send |
|                        |     raise result                                                        |
|                        | ", "created": "2016-03-11T11:00:29Z"}                                   |
| flavor                 | bmtest (645de08d-2bc6-43f1-8a5f-2315a75b1348)                           |
| hostId                 |                                                                         |
| id                     | ee08f82e-8920-4360-be51-a3f995624e5f                                    |
| image                  | opensuse (17e4915a-ada0-4b95-bacf-ba67133f39a7)                         |
| key_name               | ironic_kp                                                               |
| metadata               | {}                                                                      |
| name                   | opensuse                                                                |
| os-extended-volumes:   |                                                                         |
|    volumes_attached    | []                                                                      |
| status                 | ERROR                                                                   |
| tenant_id              | d53bcaf15afb4cb5aea3adaedbaa60dd                                        |
| updated                | 2016-03-11T11:00:28Z                                                    |
| user_id                | e580c645bfec4faeadef7dbd24aaf990                                        |
+------------------------+-------------------------------------------------------------------------+
```

You can find more information about the error by inspecting the log file at `/var/log/nova/nova-scheduler.log` or alternatively by viewing the error location in the source files listed in the stack-trace (in bold above).

To find the mismatch, compare the properties of the ironic node:

```
+------------------------+---------------------------------------------------------------------+
| Property               | Value                                                               |
+------------------------+---------------------------------------------------------------------+
| target_power_state     | None                                                                |
| extra                  | {}                                                                  |
| last_error             | None                                                                |
| updated_at             | None                                                                |
| maintenance_reason     | None                                                                |
| provision_state        | available                                                           |
| clean_step             | {}                                                                  |
| uuid                   | ea7246fd-e1d6-4637-9699-0b7c59c22e67                                |
| console_enabled        | False                                                               |
| target_provision_state | None                                                                |
| provision_updated_at   | None                                                                |
| maintenance            | False                                                               |
| inspection_started_at  | None                                                                |
| inspection_finished_at | None                                                                |
| power_state            | None                                                                |
| driver                 | agent_ilo                                                           |
| reservation            | None                                                                |
| properties             | {u'memory_mb': 64000, u'local_gb': 99, u'cpus': 2, u'capabilities': |
|                        | u'boot_mode:bios,boot_option:local'}                                |
| instance_uuid          | None                                                                |
| name                   | None                                                                |
| driver_info            | {u'ilo_address': u'10.1.196.117', u'ilo_password': u'******',       |
|                        | u'ilo_deploy_iso': u'b9499494-7db3-4448-b67f-233b86489c1f',         |
|                        | u'ilo_username': u'Administrator'}                                  |
| created_at             | 2016-03-11T10:17:10+00:00                                           |
| driver_internal_info   | {}                                                                  |
| chassis_uuid           |                                                                     |
| instance_info          | {}                                                                  |
+------------------------+---------------------------------------------------------------------+
```

with the flavor characteristics:

```
ardana > nova flavor-show

+----------------------------+-------------------------------------------------------------------
+
| Property                   | Value
 |
+----------------------------+-------------------------------------------------------------------
+
| OS-FLV-DISABLED:disabled   | False
 |
| OS-FLV-EXT-DATA:ephemeral  | 0
 |
| disk                       | 99
 |
| extra_specs                | {"capabilities:boot_option": "local", "cpu_arch": "x86_64",
 |
|                            | "capabilities:boot_mode": "bios"}
 |
| id                         | 645de08d-2bc6-43f1-8a5f-2315a75b1348
 |
| name                       | bmtest
 |
| os-flavor-access:is_public | True
 |
| ram                        | 64000
 |
| rxtx_factor                | 1.0
 |
| swap                       |
 |
| vcpus                      | 2
 |
+----------------------------+-----------------------------------------------------------------
+
```

In this instance, the problem is caused by the absence of the "**cpu_arch**": "**x86_64**" property on the ironic node. This can be resolved by updating the ironic node, adding the missing property:

```
ardana > ironic node-update ea7246fd-e1d6-4637-9699-0b7c59c22e67 \
  add properties/cpu_arch=x86_64
```

and then re-running the `nova boot` command.

14.6.2 Deployment to a node fails and in "ironic node-list" command, the power_state column for the node is shown as "None"

Possible cause: The IPMI commands to the node take longer to change the power state of the server.

Resolution: Check if the node power state can be changed using the following command

```
ardana > ironic node-set-power-state $NODEUUID on
```

If the above command succeeds and the power_state column is updated correctly, then the following steps are required to increase the power sync interval time.

On the first controller, reconfigure Ironic to increase the power sync interval time. In the example below, it is set to 120 seconds. This value may have to be tuned based on the setup.

1. Go to the `~/openstack/my_cloud/config/ironic/` directory and edit `ironic-conductor.conf.j2` to set the `sync_power_state_interval` value:

   ```
   [conductor]
   sync_power_state_interval = 120
   ```

2. Save the file and then run the following playbooks:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

14.6.3 Error Downloading Image

If you encounter the error below during the deployment:

```
"u'message': u'Error downloading image: Download of image id 77700...96551 failed:
Image download failed for all URLs.',
u'code': 500,
u'type': u'ImageDownloadError',
u'details': u'Download of image id 77700b53-9e15-406c-b2d5-13e7d9b96551 failed:
Image download failed for all URLs.'"
```

you should visit the Single Sign-On Settings in the Security page of IPMI and change the Single Sign-On Trust Mode setting from the default of "Trust None (SSO disabled)" to "Trust by Certificate".

14.6.4 Using `node-inspection` can cause temporary claim of IP addresses

Possible cause: Running `node-inspection` on a node discovers all the NIC ports including the NICs that don't have any connectivity. This causes a temporary consumption of the network IPs and increased usage of the allocated quota. As a result, other nodes are deprived of IP addresses and deployments can fail.

Resolution: You can add node properties manually added instead of using the inspection tool.

Note: Upgrade `ipmitool` to a version >= 1.8.15 or it may not return detailed information about the NIC interface for `node-inspection`.

14.6.5 Node permanently stuck in deploying state

Possible causes:

- Ironic conductor service associated with the node could go down.
- There might be a properties mismatch. MAC address registered for the node could be incorrect.

Resolution: To recover from this condition, set the provision state of the node to `Error` and maintenance to `True`. Delete the node and re-register again.

14.6.6 The NICs in the baremetal node should come first in boot order

Possible causes: By default, the boot order of baremetal node is set as NIC1, HDD and NIC2. If NIC1 fails, the nodes starts booting from HDD and the provisioning fails.

Resolution: Set boot order so that all the NICs appear before the hard disk of the baremetal as NIC1, NIC2…, HDD.

14.6.7 Increase in the number of nodes can cause power commands to fail

Possible causes: Ironic periodically performs a power state sync with all the baremetal nodes. When the number of nodes increase, ironic does not get sufficient time to perform power operations.

Resolution: The following procedure gives a way to increase `sync_power_state_interval`:

1. Edit the file `~/openstack/my_cloud/config/ironic/ironic-conductor.conf.j2` and navigate to the section for `[conductor]`
2. Increase the `sync_power_state_interval`. For example, for 100 nodes, set `sync_power_state_interval = 90` and save the file.
3. Execute the following set of commands to reconfigure Ironic:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

14.6.8 DHCP succeeds with PXE but times out with iPXE

If you see DHCP error "No configuration methods succeeded" in iPXE right after successful DHCP performed by embedded NIC firmware, there may be an issue with Spanning Tree Protocol on the switch.

To avoid this error, Rapid Spanning Tree Protocol needs to be enabled on the switch. If this is not an option due to conservative loop detection strategies, use the steps outlined below to install the iPXE binary with increased DHCP timeouts.

1. Clone iPXE source code

```
tux > git clone git://git.ipxe.org/ipxe.git
tux > cd ipxe/src
```

2. Modify lines 22-25 in file `config/dhcp.h`, which declare reduced DHCP timeouts (1-10 secs). Comment out lines with reduced timeouts and uncomment normal PXE timeouts (4-32)

```
//#define DHCP_DISC_START_TIMEOUT_SEC     1
//#define DHCP_DISC_END_TIMEOUT_SEC       10
#define DHCP_DISC_START_TIMEOUT_SEC   4       /* as per PXE spec */
#define DHCP_DISC_END_TIMEOUT_SEC     32      /* as per PXE spec */
```

3. Make `undionly.kpxe` (BIOS) and `ipxe.efi` (UEFI) images

```
tux > make bin/undionly.kpxe
tux > make bin-x86_64-efi/ipxe.efi
```

4. Copy iPXE images to Cloud Lifecycle Manager

```
tux > scp bin/undionly.kpxe bin-x86_64-efi/ipxe.efi stack@10.0.0.4:
stack@10.0.0.4's password:
undionly.kpxe                                    100%    66KB  65.6KB/s   00:00
ipxe.efi                                         100%  918KB 918.2KB/s   00:00
```

5. From deployer, distribute image files onto all 3 controllers

```
stack@ardana-cp1-c1-m1-mgmt:ardana > cd ~/scratch/ansible/next/ardana/ansible/
```

```
stack@ardana-cp1-c1-m1-mgmt:ardana > ~/scratch/ansible/next/ardana/ansible$ ansible -i hosts/
verb_hosts \
IRN-CND -m copy -b -a 'src=/home/stack/ipxe.efi dest=/tftpboot'
...
stack@ardana-cp1-c1-m1-mgmt:ardana > ~/scratch/ansible/next/ardana/ansible$ ansible -i hosts/
verb_hosts \
IRN-CND -m copy -b -a 'src=/home/stack/undionly.kpxe dest=/tftpboot'
...
```

There is no need to restart services. With next PXE boot attempt, iPXE binary with the increased timeout will be downloaded to the target node via TFTP.

14.6.8.1 Ironic Support and Limitations

The following drivers are supported and tested:

- `pxe_ipmitool` (UEFI and Legacy BIOS mode, flat-network)
- `pxe_ipmitool` (UEFI and Legacy BIOS mode, flat-network)
- `pxe_ilo` (UEFI and Legacy BIOS mode, flat-network)
- `agent_ipmitool` (UEFI and Legacy BIOS mode, flat-network)
- `agent_ilo` (UEFI and Legacy BIOS mode, flat-network)

When using the `agent_ilo` driver, provision will fail if the size of the user image exceeds the free space available on the ramdisk partition. This will produce an error in the Ironic Conductor logs that may look like as follows

```
"ERROR root [-] Command failed: prepare_image, error: Error downloading
image: Download of image id 0c4d74e4-58f1-4f8d-8c1d-8a49129a2163 failed: Unable
to write image to /tmp/0c4d74e4-58f1-4f8d-8c1d-8a49129a2163. Error: [Errno 28]
No space left on device: ImageDownloadError: Error downloading image: Download
of image id 0c4d74e4-58f1-4f8d-8c1d-8a49129a2163 failed: Unable to write image
to /tmp/0c4d74e4-58f1-4f8d-8c1d-8a49129a2163. Error: [Errno 28] No space left
on device"
```

By default, the total amount of space allocated to ramdisk is 4GB. To increase the space allocated for the ramdisk, you can update the deploy ISO image using the following workaround.

1. Save the deploy ISO to a file:

   ```
   tux > openstack image save --file deploy.isoIMAGE_ID
   ```

Replace *IMAGE_ID* with the ID of the deploy ISO stored in Glance. The ID can be obtained using the **openstack image list**.

2. Mount the saved ISO:

```
tux > mkdir /tmp/mnt
tux > sudo mount -t iso9660 -o loop deploy.iso /tmp/mnt
```

Since the mount directory is read-only, it is necessary to copy its content to be able to make modifications.

3. Copy the content of the mount directory to a custom directory:

```
tux > mkdir /tmp/custom
tux > cp -aRvf /tmp/mnt/* /tmp/custom/
```

4. Modify the bootloader files to increase the size of the ramdisk:

```
/tmp/custom/boot/x86_64/loader/isolinux.cfg
/tmp/custom/EFI/BOOT/grub.cfg
/tmp/custom/boot/grub2/grub.cfg
```

Find the `openstack-ironic-image` label and modify the `ramdisk_size` parameter in the `append` property. The `ramdisk_size` value must be specified in Kilobytes.

```
label openstack-ironic-image
  kernel linux
  append initrd=initrd ramdisk_size=10485760 ramdisk_blocksize=4096 \
boot_method=vmedia showopts
```

Make sure that your baremetal node has the amount of RAM that equals or exceeds the `ramdisk_size` value.

5. Repackage the ISO using the genisoimage tool:

```
tux > cd /tmp/custom
tux > genisoimage -b boot/x86_64/loader/isolinux.bin -R -J -pad -joliet-long \
-iso-level 4 -A '0xaa2dab53' -no-emul-boot -boot-info-table \
-boot-load-size 4 -c boot/x86_64/boot.catalog -hide boot/x86_64/boot.catalog \
-hide-joliet boot/x86_64/boot.catalog -eltorito-alt-boot -b boot/x86_64/efi \
-no-emul-boot -joliet-long -hide glump -hide-joliet glump -o /tmp/
custom_deploy.iso ./
```

Important

When repackaging the ISO, make sure that you use the same label. You can find the label file in the `/tmp/custom/boot/` directory. The label begins with `0x`. For example, `0x51e568cb`.

6. Delete the existing deploy ISO in Glance:

```
tux > openstack image delete IMAGE_ID
```

7. Create a new image with `custom_deploy.iso`:

```
tux > openstack image create --container-format bare \
--disk-format iso --public --file custom_deploy.iso ir-deploy-iso-ARDANA5.0
```

8. Re-create or update the Ironic node, if needed.

14.7 Node Cleaning

Cleaning is the process by which data is removed after a previous tenant has used the node. Cleaning requires use of ironic's agent_ drivers. It is extremely important to note that if the pxe_ drivers are utilized, no node cleaning operations will occur, and a previous tenant's data could be found on the node. The same risk of a previous tenant's data possibly can occur if cleaning is explicitly disabled as part of the installation.

By default, cleaning attempts to utilize ATA secure erase to wipe the contents of the disk. If secure erase is unavailable, the cleaning functionality built into the Ironic Python Agent falls back to an operation referred to as "shred" where random data is written over the contents of the disk, and then followed up by writing "0"s across the disk. This can be a time-consuming process.

An additional feature of cleaning is the ability to update firmware or potentially assert new hardware configuration, however, this is an advanced feature that must be built into the Ironic Python Agent image. Due to the complex nature of such operations, and the fact that no one size fits all, this requires a custom Ironic Python Agent image to be constructed with an appropriate hardware manager. For more information on hardware managers, see http://docs.openstack.org/developer/ironic-python-agent/#hardware-managers

Ironic's upstream documentation for cleaning may be found here: http://docs.openstack.org/developer/ironic/deploy/cleaning.html

14.7.1 Setup

Cleaning is enabled by default in ironic when installed via the Cloud Lifecycle Manager. You can verify this by examining the ironic-conductor.conf file. Look for:

```
[conductor]
clean_nodes=true
```

14.7.2 In use

When enabled, cleaning will be run automatically when nodes go from active to available state or from manageable to available. To monitor what step of cleaning the node is in, run `ironic node-show`:

```
stack@ardana-cp1-c1-m1-mgmt:~$ ironic node-show 4e6d4273-2535-4830-a826-7f67e71783ed
+------------------------+-----------------------------------------------------------------------
+
| Property               | Value
 |
+------------------------+-----------------------------------------------------------------------
+
| target_power_state     | None
 |
| extra                  | {}
 |
| last_error             | None
 |
| updated_at             | 2016-04-15T09:33:16+00:00
 |
| maintenance_reason     | None
 |
| provision_state        | cleaning
 |
| clean_step             | {}
 |
| uuid                   | 4e6d4273-2535-4830-a826-7f67e71783ed
 |
| console_enabled        | False
 |
| target_provision_state | available
 |
| provision_updated_at   | 2016-04-15T09:33:16+00:00
 |
| maintenance            | False
 |
| inspection_started_at  | None
 |
| inspection_finished_at | None
 |
| power_state            | power off
 |
| driver                 | agent_ilo
 |
| reservation            | ardana-cp1-c1-m1-mgmt
 |
| properties             | {u'memory_mb': 4096, u'cpu_arch': u'amd64', u'local_gb': 80,
 |
|                        | u'cpus': 2, u'capabilities': u'boot_mode:uefi,boot_option:local'}
 |
| instance_uuid          | None
 |
| name                   | None
 |
| driver_info            | {u'ilo_deploy_iso': u'249bf095-e741-441d-bc28-0f44a9b8cd80',
 |
|                        | u'ipmi_username': u'Administrator', u'deploy_kernel':
 |
|                        | u'3a78c0a9-3d8d-4764-9300-3e9c00e167a1', u'ilo_address':
 |
|                        | u'10.1.196.113', u'ipmi_address': u'10.1.196.113', u'deploy_ramdisk':
 |
|                        | u'd02c811c-e521-4926-9f26-0c88bbd2ee6d', u'ipmi_password': u'******',
 |
|                        | u'ilo_password': u'******', u'ilo_username': u'Administrator'}
 |
| created_at             | 2016-04-14T08:30:08+00:00
 |
| driver_internal_info   | {u'clean_steps': None,                         |
|                        | u'hardware_manager_version': {u'generic_hardware_manager': u'1.0'},
 |
|                        | u'is_whole_disk_image': True, u'agent_erase_devices_iterations': 1,
 |
|                        | u'agent_url': u'http://192.168.246.245:9999',
 |
|                        | u'agent_last_heartbeat': 1460633166}
 |
| chassis_uuid           |
 |
| instance_info          | {}
 |
+------------------------+-----------------------------------------------------------------------
+
```

The status will be in the `driver_internal_info` field. You will also be able to see the `clean_steps` list there.

14.7.3 Troubleshooting

If an error occurs during the cleaning process, the node will enter the clean failed state so that it is not deployed. The node remains powered on for debugging purposes. The node can be moved to the manageable state to attempt a fix using the following command:

```
ironic node-set-provision-state <node id> manage
```

Once you have identified and fixed the issue, you can return the node to available state by executing the following commands:

```
ironic node-set-maintenance <node id> false
ironic node-set-provision-state <node id> provide
```

This will retry the cleaning steps and set the node to available state upon their successful completion.

14.7.4 Disabling Node Cleaning

To disable node cleaning, edit `~/openstack/my_cloud/definition/data/ironic/ironic_config.yml` and set `enable_node_cleaning` to `false`.

Commit your changes:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "disable node cleaning"
```

Deploy these changes by re-running the configuration processor and reconfigure the ironic installation:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
ansible-playbook -i hosts/localhost ready-deployment.yml
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

14.8 Ironic and HPE OneView

14.8.1 Enabling Ironic HPE OneView driver in SUSE OpenStack Cloud

Edit the file `~/openstack/my_cloud/definition/data/ironic/ironicconfig.yml` and set the value

```
enable_oneview: true
```

This will enable the HPE OneView driver for Ironic in SUSE OpenStack Cloud.

14.8.2 Adding HPE OneView Appliance Credentials

```
manage_url: https://<Onview appliance URL>

oneview_username: "<Appliance username>"

oneview_encrypted_password: "<Encrypted password>"

oneview_allow_insecure_connections: <true/false>

tls_cacert_file: <CA certificate for connection>
```

14.8.3 Encrypting the HPE OneView Password

Encryption can be applied using `ardanaencrypt.py` or using `openssl`. On the Cloud Lifecycle Manager node, export the key used for encryption as environment variable:

```
export ARDANA_USER_PASSWORD_ENCRYPT_KEY="ENCRYPTION_KEY"
```

And then execute the following commands:

```
cd ~/openstack/ardana/ansible
python ardanaencrypt.py
```

Enter password to be encrypted when prompted. The script uses the key that was exported in the `ARDANA_USER_PASSWORD_ENCRYPT_KEY` to do the encryption.

For more information, see *Book "Security Guide", Chapter 9 "Encryption of Passwords and Sensitive Data"*.

14.8.4 Decrypting the HPE OneView Password

Before running the `site.yml` playbook, export the key used for encryption as environment variable:

```
export ARDANA_USER_PASSWORD_ENCRYPT_KEY="ENCRYPTION_KEY"
```

The decryption of the password is then automatically handled in ironic-ansible playbooks.

14.8.5 Registering Baremetal Node for HPE OneView Driver

```
ironic node-create -d agent_pxe_oneview
```

Update node driver-info:

```
 ironic node-update $NODE_UUID add driver_info/server_hardware_uri=$SH_URI
```

14.8.6 Updating Node Properties

```
ironic node-update $NODE_UUID add \
  properties/capabilities=server_hardware_type_uri:$SHT_URI,\
 enclosure_group_uri:$EG_URI,server_profile_template_uri=$SPT_URI
```

14.8.7 Creating Port for Driver

```
ironic port-create -n $NODE_UUID -a $MAC_ADDRESS
```

14.8.8 Creating a Node

Create Node:

```
ironic node-create -n ovbay7 -d agent_pxe_oneview
```

Update driver info:

```
ironic node-update $ID add driver_info/server_hardware_uri="/rest/
server-hardware/3037...464B" \
driver_info/deploy_kernel="$KERNELDISK" driver_info/
deploy_ramdisk="$RAMDISK"
```

Update node properties:

```
ironic node-update $ID add properties/local_gb=10
ironic node-update $ID add properties/cpus=24 properties/memory_mb=262144 \
properties/cpu_arch=x86_64
```

```
ironic node-update \
$ID add properties/capabilities=server_hardware_type_uri:'/rest/server-hardware-types/B31...F69E',\
enclosure_group_uri:'/rest/enclosure-groups/80efe...b79fa',\
server_profile_template_uri:'/rest/server-profile-templates/faafc3c0-6c81-47ca-a407-f67d11262da5'
```

14.8.9 Getting Data using REST API

GET login session auth id:

```
curl -k https://ONEVIEW_MANAGER_URL/rest/login-sessions \
  -H "content-type:application/json" \
  -X POST \
  -d '{"userName":"USER_NAME", "password":"PASSWORD"}'
```

Get the complete node details in JSON format:

```
curl -k "https://ONEVIEW_MANAGER_URL;/rest/server-
hardware/30373237-3132-4753-4835-32325652464B" -H "content-type:application/json" -H
 "Auth:<auth_session_id>"| python -m json.tool
```

14.8.10 Ironic HPE OneView CLI

`ironic-oneview-cli` is already installed in `ironicclient` venv with a symbolic link to it. To generate an `rc` file for the HPE OneView CLI, follow these steps:

1. Run the following commands:

   ```
   source ~/service.osrc
   glance image-list
   ```

2. Note the `deploy-kernel` and `deploy-ramdisk` UUID and then run the following command to generate the `rc` file:

   ```
   ironic-oneview genrc
   ```

You will be prompted to enter:

- HPE OneView Manager URL
- Username
- deploy-kernel
- deploy-ramdisk
- allow_insecure_connection
- cacert file

The `ironic-oneview.rc` file will be generated in the current directory, by default. It is possible to specify a different location.

3. Source the generated file:

```
source ironic-oneview.rc
```

Now enter the password of the HPE OneView appliance.

4. You can now use the CLI for node and flavor creation as follows:

```
ironic-oneview node-create
ironic-oneview flavor-create
```

14.9 RAID Configuration for Ironic

1. Node Creation:
Check the raid capabilities of the driver:

```
ironic --ironic-api-version 1.15 driver-raid-logical-disk-properties pxe_ilo
```

This will generate output similar to the following:

```
+----------------------+-------------------------------------------------------------------------+
| Property             | Description                                                             |
+----------------------+-------------------------------------------------------------------------+
| controller           | Controller to use for this logical disk. If not specified, the          |
|                      | driver will choose a suitable RAID controller on the bare metal node.   |
|                      | Optional.                                                               |
| disk_type            | The type of disk preferred. Valid values are 'hdd' and 'ssd'. If this   |
|                      | is not specified, disk type will not be a selection criterion for       |
|                      | choosing backing physical disks. Optional.                              |
| interface_type       | The interface type of disk. Valid values are 'sata', 'scsi' and 'sas'.  |
|                      | If this is not specified, interface type will not be a selection        |
|                      | criterion for choosing backing physical disks. Optional.                |
| is_root_volume       | Specifies whether this disk is a root volume. By default, this is False.|
|                      | Optional.                                                               |
| #_of_physical_disks  | Number of physical disks to use for this logical disk. By default, the  |
|                      | driver uses the minimum number of disks required for that RAID level.   |
|                      | Optional.                                                               |
| physical_disks       | The physical disks to use for this logical disk. If not specified, the  |
|                      | driver will choose suitable physical disks to use. Optional.            |
| raid_level           | RAID level for the logical disk. Valid values are '0', '1', '2', '5',   |
|                      | '6', '1+0', '5+0' and '6+0'. Required.                                  |
| share_physical_disks | Specifies whether other logical disks can share physical disks with this|
|                      | logical disk. By default, this is False. Optional.                      |
| size_gb              | Size in GiB (Integer) for the logical disk. Use 'MAX' as size_gb if     |
|                      | this logical disk is supposed to use the rest of                        |
|                      | the space available. Required.                                          |
| volume_name          | Name of the volume to be created. If this is not specified, it will be  |
|                      | auto-generated. Optional.                                               |
+----------------------+-------------------------------------------------------------------------+
```

Node State will be **Available**

```
ironic node-create -d pxe_ilo -i ilo_address=<ip_address> \
  -i ilo_username=<username> -i ilo_password=<password> \
  -i ilo_deploy_iso=<iso_id> -i deploy_kernel=<kernel_id> \
  -i deploy_ramdisk=<ramdisk_id> -p cpus=2 -p memory_mb=4096 \
  -p local_gb=80  -p cpu_arch=amd64 \
  -p capabilities="boot_option:local,boot_mode:bios"
```

```
ironic port-create -a <port> -n <node-uuid>
```

2. Apply the target raid configuration on the node:
See the OpenStack documentation for RAID configuration at http://docs.openstack.org/developer/ironic/deploy/raid.html.
Set the target RAID configuration by editing the file `raid_conf.json` and setting the appropriate values, for example:

```
{ "logical_disks": [ {"size_gb": 5, "raid_level": "0", "is_root_volume": true} ] }
```

and then run the following command:

```
ironic --ironic-api-version 1.15 node-set-target-raid-config <node-uuid>
 raid_conf.json
```

The output produced should be similar to the following:

```
+-----------------------+-----------------------------------------------------------------------
+
| Property              | Value
 |
+-----------------------+-----------------------------------------------------------------------
+
| chassis_uuid          |
 |
| clean_step            | {}
 |
| console_enabled       | False
 |
| created_at            | 2016-06-14T14:58:07+00:00
 |
| driver                | pxe_ilo
 |
| driver_info           | {u'ilo_deploy_iso': u'd43e589a-07db-4fce-a06e-98e2f38340b4',
 |
|                       | u'deploy_kernel': u'915c5c74-1ceb-4f78-bdb4-8547a90ac9c0',
 |
|                       | u'ilo_address': u'10.1.196.116', u'deploy_ramdisk':
 |
|                       | u'154e7024-bf18-4ad2-95b0-726c09ce417a', u'ilo_password': u'******',
 |
|                       | u'ilo_username': u'Administrator'}
 |
| driver_internal_info  | {u'agent_cached_clean_steps_refreshed': u'2016-06-15 07:16:08.264091',
 |
|                       | u'agent_cached_clean_steps': {u'raid': [{u'interface': u'raid',
 |
|                       | u'priority': 0, u'step': u'delete_configuration'}, {u'interface':
 |
|                       | u'raid', u'priority': 0, u'step': u'create_configuration'}], u'deploy':
 |
|                       | [{u'priority': 10, u'interface': u'deploy', u'reboot_requested': False,
 |
|                       | u'abortable': True, u'step': u'erase_devices'}]}, u'clean_steps': None,
 |
|                       | u'hardware_manager_version': {u'generic_hardware_manager': u'3'},
 |
|                       | u'agent_erase_devices_iterations': 1, u'agent_url':
 |
|                       | u'http://192.168.245.143:9999', u'agent_last_heartbeat': 1465974974}
 |
| extra                 | {}
 |
| inspection_finished_at| None
 |
| inspection_started_at | None
 |
| instance_info         | {u'deploy_key': u'XXN2ON0V9ER429MECETJMUG5YHTKOQOZ'}
 |
| instance_uuid          | None
 |
| last_error             | None
 |
| maintenance            | False
 |
| maintenance_reason     | None
 |
| name                   | None
 |
| power_state            | power off
 |
| properties             | {u'cpu_arch': u'amd64', u'root_device': {u'wwn':
 u'0x600508b1001ce286'},|
|                        | u'cpus': 2, u'capabilities':
 |
|                        | u'boot_mode:bios,raid_level:6,boot_option:local', u'memory_mb': 4096,
 |
|                        | u'local_gb': 4}
 |
| provision_state        | available
 |
| provision_updated_at   | 2016-06-15T07:16:27+00:00
 |
| reservation            | padawan-ironic-cp1-c1-m2-mgmt
 |
| target_power_state     | power off
 |
| target_provision_state| None
 |
| target_raid_config     | {u'logical_disks': [{u'size_gb': 5, u'raid_level': u'6',
 |
|                        | u'is_root_volume': True}]}
 |
| updated_at             | 2016-06-15T07:44:22+00:00
 |
| uuid                   | 22ab9f85-71a1-4748-8d6b-f6411558127e
 |
+-----------------------+-----------------------------------------------------------------------
+
```

Now set the state of the node to **manageable**:

```
ironic --ironic-api-version latest node-set-provision-state <node-uuid> manage
```

3. Manual cleaning steps:
   Manual cleaning is enabled by default in production - the following are the steps to enable cleaning if the manual cleaning has been disabled.

   a. Provide `cleaning_network_uuid` in `ironic-conductor.conf`

b. Edit the file `~/openstack/my_cloud/definition/data/ironic/ironic_config.yml` and set `enable_node_cleaning` to `true`.

c. Then run the following set of commands:

```
cd ~/openstack/ardana/ansible
git add -A
git commit -m "enabling node cleaning"
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
ansible-playbook -i hosts/localhost ready-deployment.yml
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

After performing these steps, the state of the node will become **Cleaning**.

Run the following command:

```
ironic --ironic-api-version latest node-set-provision-state 2123254e-8b31...aa6fd \
  clean --clean-steps '[{ "interface": "raid","step": "delete_configuration"}, \
  { "interface": "raid" ,"step": "create_configuration"}]'
```

Node-information after a Manual cleaning:

```
+-----------------------+-------------------------------------------------------------------------
+
| Property              | Value
 |
+-----------------------+-------------------------------------------------------------------------
+
| chassis_uuid          |
 |
| clean_step            | {}
 |
| console_enabled       | False
 |
| created_at            | 2016-06-14T14:58:07+00:00
 |
| driver                | pxe_ilo
 |
| driver_info           | {u'ilo_deploy_iso': u'd43e589a-07db-4fce-a06e-98e2f38340b4',
 |
|                       | u'deploy_kernel': u'915c5c74-1ceb-4f78-bdb4-8547a90ac9c0',
 |
|                       | u'ilo_address': u'10.1.196.116', u'deploy_ramdisk':
 |
|                       | u'154e7024-bf18-4ad2-95b0-726c09ce417a', u'ilo_password': u'******',
 |
|                       | u'ilo_username': u'Administrator'}
 |
| driver_internal_info  | {u'agent_cached_clean_steps_refreshed': u'2016-06-15 07:16:08.264091',
 |
|                       | u'agent_cached_clean_steps': {u'raid': [{u'interface': u'raid',
 |
|                       | u'priority': 0, u'step': u'delete_configuration'}, {u'interface':
 |
|                       | u'raid', u'priority': 0, u'step': u'create_configuration'}], u'deploy':
 |
|                       | [{u'priority': 10, u'interface': u'deploy', u'reboot_requested': False,
 |
|                       | u'abortable': True, u'step': u'erase_devices'}]}, u'clean_steps': None,
 |
|                       | u'hardware_manager_version': {u'generic_hardware_manager': u'3'},
 |
|                       | u'agent_erase_devices_iterations': 1, u'agent_url':
 |
|                       | u'http://192.168.245.143:9999', u'agent_last_heartbeat': 1465974974}
 |
| extra                 | {}
 |
| inspection_finished_at| None
 |
| inspection_started_at | None
 |
| instance_info         | {u'deploy_key': u'XXN2ON0V9ER429MECETJMUG5YHTKOQOZ'}
 |
| instance_uuid         | None
 |
| last_error            | None
 |
| maintenance           | False
 |
| maintenance_reason    | None
 |
| name                  | None
 |
| power_state           | power off
 |
| properties            | {u'cpu_arch': u'amd64', u'root_device': {u'wwn':
 u'0x600508b1001ce286'},|
|                       | u'cpus': 2, u'capabilities':
 |
|                       | u'boot_mode:bios,raid_level:6,boot_option:local', u'memory_mb': 4096,
 |
|                       | u'local_gb': 4}
 |
| provision_state       | manageable
 |
| provision_updated_at  | 2016-06-15T07:16:27+00:00
 |
| raid_config           | {u'last_updated': u'2016-06-15 07:16:14.584014', u'physical_disks':
 |
|                        | [{u'status': u'ready', u'size_gb': 1024, u'interface_type': u'sata',
 |
|                        | u'firmware': u'HPGC', u'controller': u'Smart Array P440ar in Slot 0
 |
|                        | (Embedded)', u'model': u'ATA     MM1000GBKAL', u'disk_type': u'hdd',
 |
|                        | u'id': u'1I:3:3'}, {u'status': u'ready', u'size_gb': 1024,
 |
|                        | u'interface_type': u'sata', u'firmware': u'HPGC', u'controller':
 u'Smart|
|                        | Array P440ar in Slot 0 (Embedded)', u'model': u'ATA     MM1000GBKAL',
 |
|                        | u'disk_type': u'hdd', u'id': u'1I:3:1'}, {u'status': u'active',
 |
|                        | u'size_gb': 1024, u'interface_type': u'sata', u'firmware': u'HPGC',
 |
|                        | u'controller': u'Smart Array P440ar in Slot 0 (Embedded)', u'model':
 |
|                        | u'ATA     MM1000GBKAL', u'disk_type': u'hdd', u'id': u'1I:3:2'},
 |
|                        | {u'status': u'active', u'size_gb': 1024, u'interface_type': u'sata',
 |
|                        | u'firmware': u'HPGC', u'controller': u'Smart Array P440ar in Slot 0
 |
|                        | (Embedded)', u'model': u'ATA     MM1000GBKAL', u'disk_type': u'hdd',
 |
|                        | u'id': u'2I:3:6'}, {u'status': u'active', u'size_gb': 1024,
 |
|                        | u'interface_type': u'sata', u'firmware': u'HPGC', u'controller':
 u'Smart|
|                        | Array P440ar in Slot 0 (Embedded)', u'model': u'ATA     MM1000GBKAL',
 |
|                        | u'disk_type': u'hdd', u'id': u'2I:3:5'}, {u'status': u'active',
 |
|                        | u'size_gb': 1024, u'interface_type': u'sata', u'firmware': u'HPGC',
 |
|                        | u'controller': u'Smart Array P440ar in Slot 0 (Embedded)', u'model':
 |
|                        | u'ATA     MM1000GBKAL', u'disk_type': u'hdd', u'id': u'1I:3:4'}],
 |
|                        | u'logical_disks': [{u'size_gb': 4, u'physical_disks': [u'1I:3:2',
 |
|                        | u'2I:3:6', u'2I:3:5', u'1I:3:4'], u'raid_level': u'6',
 |
|                        | u'is_root_volume': True, u'root_device_hint': {u'wwn':
 |
|                        | u'0x600508b1001ce286'}, u'controller': u'Smart Array P440ar in Slot 0
 |
|                        | (Embedded)', u'volume_name': u'015E795CPDNLH0BRH8N406AAB7'}]}
 |
| reservation            | padawan-ironic-cp1-c1-m2-mgmt
 |
| target_power_state     | power off
 |
| target_provision_state| None
 |
| target_raid_config    | {u'logical_disks': [{u'size_gb': 5, u'raid_level': u'6',
 |
|                       | u'is_root_volume': True}]}
 |
| updated_at            | 2016-06-15T07:44:22+00:00
 |
| uuid                  | 22ab9f85-71a1-4748-8d6b-f6411558127e
 |
+-----------------------+-----------------------------------------------------------------------
+
```

After the manual cleaning, run the following command to change the state of a node to **available**:

```
ironic --ironic-api-version latest node-set-provision-state <node-uuid> \
  provide
```

14.10 Audit Support for Ironic

14.10.1 API Audit Logging

Audit middleware supports delivery of CADF audit events via Oslo messaging notifier capability. Based on `notification_driver` configuration, audit events can be routed to messaging infrastructure (`notification_driver = messagingv2`) or can be routed to a log file (`notification_driver = log`).

Audit middleware creates two events per REST API interaction. The first event has information extracted from request data and the second one contains information on the request outcome (response).

14.10.2 Enabling API Audit Logging

You can enable audit logging for Ironic by changing the configuration in the input model. Edit the file `~/openstack/my_cloud/definition/cloudConfig.yml` and in the `audit-settings` section, change the `default` value to `enabled`. The ironic-ansible playbooks will now enable audit support for Ironic.

API audit events will be logged in the corresponding audit directory, for example, `/var/audit/ironic/ironic-api-audit.log`. An audit event will be logged in the log file for every request and response for an API call.

14.10.3 Sample Audit Event

The following output is an example of an audit event for an `ironic node-list` command:

```
{
   "event_type":"audit.http.request",
   "timestamp":"2016-06-15 06:04:30.904397",
   "payload":{
      "typeURI":"http://schemas.dmtf.org/cloud/audit/1.0/event",
      "eventTime":"2016-06-15T06:04:30.903071+0000",
      "target":{
         "id":"ironic",
         "typeURI":"unknown",
         "addresses":[
            {
               "url":"http://{ironic_admin_host}:6385",
               "name":"admin"
            },
           {
               "url":"http://{ironic_internal_host}:6385",
               "name":"private"
           },
           {
               "url":"http://{ironic_public_host}:6385",
               "name":"public"
           }
         ],
         "name":"ironic"
      },
      "observer":{
         "id":"target"
      },
      "tags":[
         "correlation_id?value=685f1abb-620e-5d5d-b74a-b4135fb32373"
      ],
      "eventType":"activity",
      "initiator":{
         "typeURI":"service/security/account/user",
         "name":"admin",
         "credential":{
            "token":"***",
            "identity_status":"Confirmed"
         },
         "host":{
            "agent":"python-ironicclient",
            "address":"10.1.200.129"
         },
         "project_id":"d8f52dd7d9e1475dbbf3ba47a4a83313",
         "id":"8c1a948bad3948929aa5d5b50627a174"
      },
      "action":"read",
      "outcome":"pending",
      "id":"061b7aa7-5879-5225-a331-c002cf23cb6c",
      "requestPath":"/v1/nodes/?associated=True"
   },
   "priority":"INFO",
   "publisher_id":"ironic-api",
   "message_id":"2f61ebaa-2d3e-4023-afba-f9fca6f21fc2"
}
```

15 Installation for SUSE OpenStack Cloud Entry-scale Cloud with Swift Only

This page describes the installation step requirements for the SUSE OpenStack Cloud Entry-scale Cloud with Swift Only model.

15.1 Important Notes

- If you are looking for information about when to use the GUI installer and when to use the command line (CLI), see the *Installation Overview*.
- Review the *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 2 "Hardware and Software Support Matrix"* that we have listed.
- Review the release notes to make yourself aware of any known issues and limitations.
- The installation process can occur in different phases. For example, you can install the control plane only and then add Compute nodes afterwards if you would like.
- If you run into issues during installation, we have put together a list of *Chapter 19, Troubleshooting the Installation* you can reference.
- Make sure all disks on the system(s) are wiped before you begin the install. (For Swift, refer to *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 12 "Modifying Example Configurations for Object Storage using Swift", Section 12.6 "Swift Requirements for Device Group Drives"*.)
- There is no requirement to have a dedicated network for OS-install and system deployment, this can be shared with the management network. More information can be found in *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 10 "Example Configurations"*.
- You may see the terms deployer and Cloud Lifecycle Manager used interchangeably. These are referring to the same nodes in your environment.
- When running the Ansible playbook in this installation guide, if a runbook fails you will see in the error response to use the `--limit` switch when retrying a playbook. This should be avoided. You can simply re-run any playbook without this switch.

- DVR is not supported with ESX compute.
- When you attach a Cinder volume to the VM running on the ESXi host, the volume will not get detected automatically. Make sure to set the image metadata **vmware_adapter-type=lsiLogicsas** for image before launching the instance. This will help to discover the volume change appropriately.

15.2 Before You Start

1. Review the *Chapter 2, Pre-Installation Checklist* about recommended pre-installation tasks.
2. Prepare the Cloud Lifecycle Manager node. The Cloud Lifecycle Manager must be accessible either directly or via `ssh`, and have SUSE Linux Enterprise Server 12 SP3 installed. All nodes must be accessible to the Cloud Lifecycle Manager. If the nodes do not have direct access to online Cloud subscription channels, the Cloud Lifecycle Manager node will need to host the Cloud repositories.
   a. If you followed the installation instructions for Cloud Lifecycle Manager (see *Chapter 3, Installing the Cloud Lifecycle Manager server* SUSE OpenStack Cloud should already be installed. Double-check whether SUSE Linux Enterprise and SUSE OpenStack Cloud are properly registered at the SUSE Customer Center by starting YaST and running *Software* › *Product Registration*.
      If you have not yet installed SUSE OpenStack Cloud, do so by starting YaST and running *Software* › *Product Registration* › *Select Extensions*. Choose *SUSE OpenStack Cloud* and follow the on-screen instructions. Make sure to register SUSE OpenStack Cloud during the installation process and to install the software pattern `patterns-cloud-ardana`.
   b. Ensure the SUSE OpenStack Cloud media repositories and updates repositories are made available to all nodes in your deployment. This can be accomplished either by configuring the Cloud Lifecycle Manager server as an SMT mirror as described in *Chapter 4, Installing and Setting Up an SMT Server on the Cloud Lifecycle Manager server (Optional)* or by syncing or mounting the Cloud and updates repositories to the Cloud Lifecycle Manager server as described in *Chapter 5, Software Repository Setup*.

c. Configure passwordless **sudo** for the user created when setting up the node (as described in *Section 3.5, "Creating a User"*). Note that this is *not* the user ardana that will be used later in this procedure. In the following we assume you named the user cloud. Run the command **visudo** as user root and add the following line to the end of the file:

```
cloud ALL = (root) NOPASSWD:ALL
```

Make sure to replace *cloud* with your user name choice.

d. Set the password for the user ardana:

```
sudo passwd ardana
```

e. Become the user ardana:

```
su - ardana
```

f. Place a copy of the SUSE Linux Enterprise Server 12 SP3 .iso in the ardana home directory, var/lib/ardana, and rename it to sles12sp3.iso.

15.3 Setting Up the Cloud Lifecycle Manager

15.3.1 Installing the Cloud Lifecycle Manager

Running the **ARDANA_INIT_AUTO=1** command is optional to avoid stopping for authentication at any step. You can also run **ardana-init** to launch the Cloud Lifecycle Manager. You will be prompted to enter an optional SSH passphrase, which is used to protect the key used by Ansible when connecting to its client nodes. If you do not want to use a passphrase, press Enter at the prompt.

If you have protected the SSH key with a passphrase, you can avoid having to enter the passphrase on every attempt by Ansible to connect to its client nodes with the following commands:

```
ardana > eval $(ssh-agent)
ardana > ssh-add ~/.ssh/id_rsa
```

The Cloud Lifecycle Manager will contain the installation scripts and configuration files to deploy your cloud. You can set up the Cloud Lifecycle Manager on a dedicated node or you do so on your first controller node. The default choice is to use the first controller node as the Cloud Lifecycle Manager.

1. Download the product from:
   - SUSE Customer Center (https://scc.suse.com/)
2. Boot your Cloud Lifecycle Manager from the SLES ISO contained in the download.
3. Enter `install` (all lower-case, exactly as spelled out here) to start installation.
4. Select the language. Note that only the English language selection is currently supported.
5. Select the location.
6. Select the keyboard layout.
7. Select the primary network interface, if prompted:
   - Assign IP address, subnet mask, and default gateway
8. Create new account:
   a. Enter a username.
   b. Enter a password.
   c. Enter time zone.

Once the initial installation is finished, complete the Cloud Lifecycle Manager setup with these steps:

1. Ensure your Cloud Lifecycle Manager has a valid DNS nameserver specified in `/etc/resolv.conf`.
2. Set the environment variable LC_ALL:

   ```
   export LC_ALL=C
   ```

   Note

   This can be added to `~/.bashrc` or `/etc/bash.bashrc`.

The node should now have a working SLES setup.

15.4 Configure Your Environment

This part of the install is going to depend on the specific cloud configuration you are going to use.

Setup your configuration files, as follows:

1. See the sample sets of configuration files in the `~/openstack/examples/` directory. Each set will have an accompanying README.md file that explains the contents of each of the configuration files.

2. Copy the example configuration files into the required setup directory and edit them to contain the details of your environment:

```
cp -r ~/openstack/examples/entry-scale-swift/* \
  ~/openstack/my_cloud/definition/
```

3. Begin inputting your environment information into the configuration files in the `~/openstack/my_cloud/definition` directory.
   Full details of how to do this can be found here: *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 12 "Modifying Example Configurations for Object Storage using Swift", Section 12.10 "Understanding Swift Ring Specifications", Section 12.10.1 "Ring Specifications in the Input Model"*.
   In many cases, the example models provide most of the data you need to create a valid input model. However, there are two important aspects you must plan and configure before starting a deploy as follows:

   - Check the disk model used by your nodes. Specifically, check that all disk drives are correctly named and used as described in *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 12 "Modifying Example Configurations for Object Storage using Swift", Section 12.6 "Swift Requirements for Device Group Drives"*.

   - Select an appropriate partition power for your rings. Detailed information about this is provided at *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 12 "Modifying Example Configurations for Object Storage using Swift", Section 12.10 "Understanding Swift Ring Specifications"*.

Optionally, you can use the `ardanaencrypt.py` script to encrypt your IPMI passwords. This script uses OpenSSL.

1. Change to the Ansible directory:

   ```
   cd ~/openstack/ardana/ansible
   ```

2. Put the encryption key into the following environment variable:

   ```
   export ARDANA_USER_PASSWORD_ENCRYPT_KEY=<encryption key>
   ```

3. Run the python script below and follow the instructions. Enter a password that you want to encrypt.

   ```
   ardanaencrypt.py
   ```

4. Take the string generated and place it in the `"ilo_password"` field in your `~/openstack/my_cloud/definition/data/servers.yml` file, remembering to enclose it in quotes.

5. Repeat the above for each server.

Note

Before you run any playbooks, remember that you need to export the encryption key in the following environment variable: `export ARDANA_USER_PASSWORD_ENCRYPT_KEY=<encryption key>`

Commit your configuration to the local git repo (*Chapter 12, Using Git for Configuration Management*), as follows:

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

Important

This step needs to be repeated any time you make changes to your configuration files before you move onto the following steps. See *Chapter 12, Using Git for Configuration Management* for more information.

15.5 Provisioning Your Baremetal Nodes

To provision the baremetal nodes in your cloud deployment you can either use the automated operating system installation process provided by SUSE OpenStack Cloud or you can use the 3rd party installation tooling of your choice. We will outline both methods below:

15.5.1 Using Third Party Baremetal Installers

If you do not wish to use the automated operating system installation tooling included with SUSE OpenStack Cloud then the requirements that have to be met using the installation tooling of your choice are:

- The operating system must be installed via the SLES ISO provided on the SUSE Customer Center (https://scc.suse.com/) ↗.
- Each node must have SSH keys in place that allows the same user from the Cloud Lifecycle Manager node who will be doing the deployment to SSH to each node without a password.
- Passwordless sudo needs to be enabled for the user.
- There should be a LVM logical volume as `/root` on each node.
- If the LVM volume group name for the volume group holding the `root` LVM logical volume is `ardana-vg`, then it will align with the disk input models in the examples.
- Ensure that `openssh-server`, `python`, `python-apt`, and `rsync` are installed.

If you chose this method for installing your baremetal hardware, skip forward to the step *Running the Configuration Processor*.

15.5.2 Using the Automated Operating System Installation Provided by SUSE OpenStack Cloud

If you would like to use the automated operating system installation tools provided by SUSE OpenStack Cloud, complete the steps below.

15.5.2.1 Deploying Cobbler

This phase of the install process takes the baremetal information that was provided in `servers.yml` and installs the Cobbler provisioning tool and loads this information into Cobbler. This sets each node to `netboot-enabled: true` in Cobbler. Each node will be automatically marked as `netboot-enabled: false` when it completes its operating system install successfully. Even if the node tries to PXE boot subsequently, Cobbler will not serve it. This is deliberate so that you can't reimage a live node by accident.

The `cobbler-deploy.yml` playbook prompts for a password - this is the password that will be encrypted and stored in Cobbler, which is associated with the user running the command on the Cloud Lifecycle Manager, that you will use to log in to the nodes via their consoles after install. The username is the same as the user set up in the initial dialogue when installing the Cloud Lifecycle Manager from the ISO, and is the same user that is running the cobbler-deploy play.

1. Run the following playbook which confirms that there is IPMI connectivity for each of your nodes so that they are accessible to be re-imaged in a later step:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost bm-power-status.yml
   ```

2. Run the following playbook to deploy Cobbler:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost cobbler-deploy.yml
   ```

15.5.2.2 Imaging the Nodes

This phase of the install process goes through a number of distinct steps:

1. Powers down the nodes to be installed
2. Sets the nodes hardware boot order so that the first option is a network boot.
3. Powers on the nodes. (The nodes will then boot from the network and be installed using infrastructure set up in the previous phase)
4. Waits for the nodes to power themselves down (this indicates a success install). This can take some time.
5. Sets the boot order to hard disk and powers on the nodes.

6. Waits for the nodes to be ssh-able and verifies that they have the signature expected.

The reimaging command is:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-reimage.yml \
  [-e nodelist=node1,node2,node3]
```

If a nodelist is not specified then the set of nodes in cobbler with `netboot-enabled: True` is selected. The playbook pauses at the start to give you a chance to review the set of nodes that it is targeting and to confirm that it is correct.

You can use the command below which will list all of your nodes with the `netboot-enabled: True` flag set:

```
sudo cobbler system find --netboot-enabled=1
```

15.6 Running the Configuration Processor

Once you have your configuration files setup, you need to run the configuration processor to complete your configuration.

When you run the configuration processor, you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the ansible `group_vars` and `host_vars` that it produces for subsequent deploy runs. You will need this password for subsequent Ansible deploy and configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise just press `Enter` to bypass this.

Run the configuration processor with this command:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

For automated installation (for example CI), you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor:

```
ansible-playbook -i hosts/localhost config-processor-run.yml \
  -e encrypt="" -e rekey=""
```

If you receive an error during this step, there is probably an issue with one or more of your configuration files. Verify that all information in each of your configuration files is correct for your environment. Then commit those changes to Git using the instructions in the previous section before re-running the configuration processor again.

For any troubleshooting information regarding these steps, see *Section 19.2, "Issues while Updating Configuration Files"*.

15.7 Deploying the Cloud

1. Use the playbook below to create a deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

2. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your partitions on your nodes are completely wiped before continuing with the installation. If you are using fresh machines this step may not be necessary.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts wipe_disks.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts wipe_disks.yml --ask-vault-pass
```

3. Run the `site.yml` playbook below:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts site.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts site.yml --ask-vault-pass
```

Note

The step above runs `osconfig` to configure the cloud and `ardana-deploy` to deploy the cloud. Therefore, this step may run for a while, perhaps 45 minutes or more, depending on the number of nodes in your environment.

4. Verify that the network is working correctly. Ping each IP in the `/etc/hosts` file from one of the controller nodes.

For any troubleshooting information regarding these steps, see *Section 19.3, “Issues while Deploying the Cloud”*.

15.8 Post-Installation Verification and Administration

We recommend verifying the installation using the instructions in *Chapter 22, Cloud Verification*.

There are also a list of other common post-installation administrative tasks listed in the *Chapter 28, Other Common Post-Installation Tasks* list.

16 Installing SLES Compute

16.1 SLES Compute Node Installation Overview

SUSE OpenStack Cloud 8 supports SLES compute nodes, specifically SUSE Linux Enterprise Server 12 SP3. SUSE does not ship a SLES ISO with SUSE OpenStack Cloud so you will need to download a copy of the SLES ISO (`SLE-12-SP3-Server-DVD-x86_64-GM-DVD1.iso`) and the SLES SDK ISO (`SLE-12-SP3-SDK-DVD-x86_64-GM-DVD1.iso`) from SUSE. You can use the following link to download the ISO. To do so, either log in or create a SUSE account before downloading: https://www.suse.com/products/server/download/ ↗.

There are two approaches for deploying SLES compute nodes in SUSE OpenStack Cloud:

- Using the Cloud Lifecycle Manager to automatically deploy SLES Compute Nodes.
- Provisioning SLES nodes yourself, either manually or using a third-party tool, and then providing the relevant information to the Cloud Lifecycle Manager.

These two approaches can be used whether you are installing a cloud for the first time or adding a compute node to an existing cloud. Regardless of your approach, you should be certain to register your SLES compute nodes in order to get product updates as they come available. For more information, see *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 1 "Registering SLES"*.

16.2 SLES Support

SUSE Linux Enterprise Server (SLES) Host OS KVM and/or supported SLES guests have been tested and qualified by SUSE to run on SUSE OpenStack Cloud.

16.3 Using the Cloud Lifecycle Manager to Deploy SLES Compute Nodes

The method used for deploying SLES compute nodes using Cobbler on the Cloud Lifecycle Manager uses legacy BIOS.

Note

UEFI and Secure boot are not supported on SLES Compute.

16.3.1 Deploying legacy BIOS SLES Compute nodes

The installation process for legacy BIOS SLES Compute nodes is similar to that described in *Chapter 13, Installing Mid-scale and Entry-scale KVM* with some additional requirements:

- The standard SLES ISO (SLE-12-SP3-Server-DVD-x86_64-GM-DVD1.iso) must be accessible via `/home/stack/sles12sp3.iso`. Rename the ISO or create a symbolic link:

  ```
  mv SLE-12-SP3-Server-DVD-x86_64-GM-DVD1.iso /home/stack/sles12sp3.iso
  ```

- You must identify the node(s) on which you want to install SLES, by adding the key/value pair `distro-id: sles12sp3-x86_64` to server details in `servers.yml`. You will also need to update `net_interfaces.yml`, `server_roles.yml`, `disk_compute.yml` and `control_plane.yml`. For more information on configuration of the Input Model for SLES, see *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 11 "Modifying Example Configurations for Compute Nodes", Section 11.1 "SLES Compute Nodes"*.

16.3.2 Deploying UEFI SLES compute nodes

Deplyoing UEFI nodes has been automated in the Cloud Lifecycle Manager and requires the following to be met:

- All of your nodes using SLES must already be installed, either manually or via Cobbler.
- Your input model should be configured for your SLES nodes, per the instructions at *Book* "Planning an Installation with Cloud Lifecycle Manager", *Chapter 11 "Modifying Example Configurations for Compute Nodes", Section 11.1 "SLES Compute Nodes"*.
- You should have run the configuration processor and the `ready-deployment.yml` playbook.

Execute the following steps to re-image one or more nodes after you have run the `ready-deployment.yml` playbook.

1. Run the following playbook, ensuring that you specify only your UEFI SLES nodes using the nodelist. This playbook will reconfigure Cobbler for the nodes listed.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook prepare-sles-grub2.yml -e nodelist=node1[,node2,node3]
```

2. Re-image the node(s), ensuring that you only specify your UEFI SLES nodes using the nodelist.

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost bm-reimage.yml \
-e nodelist=node1[,node2,node3]
```

3. Make backups of the `grub.cfg-*` and `grubx64.efi` files in `/srv/tftp/grub/` as they will be overwritten when running the cobbler-deploy playbook on the next step. You will need these files if you need to reimage the nodes in the future.

4. Run the `cobbler-deploy.yml` playbook, which will reset Cobbler back to the default values:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

16.3.2.1 UEFI Secure Boot

Secure Boot is a method used to restrict binaries execution for booting the system. With this option enabled, system BIOS will only allow boot loaders with trusted cryptographic signatures to be executed, thus enable preventing malware from hiding embedded code in the boot chain. Each boot loader launched during the boot process is digitally signed and that signature is validated against a set of trusted certificates embedded in the UEFI BIOS. Secure Boot is completely implemented in the BIOS and does not require special hardware.

Thus Secure Boot is:

- Intended to prevent boot-sector malware or kernel code injection.
- Hardware-based code signing.
- Extension of the UEFI BIOS architecture.
- Optional with the ability to enable or disable it through the BIOS.

In Boot Options of RBSU, *Boot Mode* needs to be set to `UEFI Mode` and *UEFI Optimized Boot* should be `Enabled` >.

Secure Boot is enabled at *System Configuration* › *BIOS/Platform Configuration (RBSU)* › *Server Security* › *Secure Boot Configuration* › *Secure Boot Enforcement*.

16.4 Provisioning SLES Yourself

This section outlines the steps needed to manually provision a SLES node so that it can be added to a new or existing SUSE OpenStack Cloud 8 cloud.

16.4.1 Configure Cloud Lifecycle Manager to Enable SLES

1. Take note of the IP address of the Cloud Lifecycle Manager node. It will be used below during *Section 16.4.6, “Add zypper repository”*.
2. Mount or copy the contents of `SLE-12-SP3-Server-DVD-x86_64-GM-DVD1.iso` to `/srv/www/suse-12.3/x86_64/repos/ardana/sles12/zypper/OS/`
3. Mount or copy the contents of `SLE-12-SP3-SDK-DVD-x86_64-GM-DVD1.iso` to `/srv/www/suse-12.3/x86_64/repos/ardana/sles12/zypper/SDK/`

Note

If you choose to mount an ISO, we recommend creating an `/etc/fstab` entry to ensure the ISO is mounted after a reboot.

16.4.2 Install SUSE Linux Enterprise Server 12 SP3

Install SUSE Linux Enterprise Server 12 SP3 using the standard iso (`SLE-12-SP3-Server-DVD-x86_64-GM-DVD1.iso`)

16.4.3 Assign a static IP

1. Use the `ip addr` command to find out what network devices are on your system:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
```

```
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether f0:92:1c:05:89:70 brd ff:ff:ff:ff:ff:ff
    inet 10.13.111.178/26 brd 10.13.111.191 scope global eno1
       valid_lft forever preferred_lft forever
    inet6 fe80::f292:1cff:fe05:8970/64 scope link
       valid_lft forever preferred_lft forever
3: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether f0:92:1c:05:89:74 brd ff:ff:ff:ff:ff:ff
```

2. Identify the one that matches the MAC address of your server and edit the corresponding config file in `/etc/sysconfig/network-scripts`.

```
vi /etc/sysconfig/network-scripts/ifcfg-eno1
```

3. Edit the `IPADDR` and `NETMASK` values to match your environment. Note that the `IPADDR` is used in the corresponding stanza in `servers.yml`. You may also need to set `BOOTPROTO` to `none`.

```
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=eno1
UUID=36060f7a-12da-469b-a1da-ee730a3b1d7c
DEVICE=eno1
ONBOOT=yes
NETMASK=255.255.255.192
IPADDR=10.13.111.14
```

4. [OPTIONAL] Reboot your SLES node and ensure that it can be accessed from the Cloud Lifecycle Manager.

16.4.4 Add ardana user and home directory

```
useradd -m ardana
passwd ardana
```

16.4.5 Allow user ardana to sudo without password

Setting up sudo on SLES is covered in the *SLES Administration Guide* at https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_sudo_conf.html.

The recommendation is to create user specific `sudo` config files under `/etc/sudoers.d`, therefore creating an `/etc/sudoers.d/ardana` config file with the following content will allow sudo commands without the requirement of a password.

```
ardana ALL=(ALL) NOPASSWD:ALL
```

16.4.6 Add zypper repository

Using the ISO-based repositories created above, add the zypper repositories.

Follow these steps. Update the value of deployer_ip as necessary.

```
deployer_ip=192.168.10.254
zypper addrepo --no-gpgcheck --refresh http://$deployer_ip:79/ardana/sles12/zypper/OS SLES-OS
zypper addrepo --no-gpgcheck --refresh http://$deployer_ip:79/ardana/sles12/zypper/SDK SLES-SDK
```

To verify that the repositories have been added, run:

```
zypper repos --detail
```

For more information about Zypper, see the *SLES Administration Guide* at https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_zypper.html.

Warning

If you intend on attaching encrypted volumes to any of your SLES Compute nodes, install the cryptographic libraries through cryptsetup on each node. Run the following command to install the necessary cryptographic libraries:

```
sudo zypper in cryptsetup
```

16.4.7 Add Required Packages

As documented in *Section 13.4, "Provisioning Your Baremetal Nodes"*, you need to add extra packages. Ensure that `openssh-server`, `python`, and `rsync` are installed.

16.4.8 Set up passwordless SSH access

Once you have started your installation using the Cloud Lifecycle Manager, or if you are adding a SLES node to an existing cloud, you need to copy the Cloud Lifecycle Manager public key to the SLES node. One way of doing this is to copy the `/home/ardana/.ssh/authorized_keys` from another node in the cloud to the same location on the SLES node. If you are installing a new cloud, this file will be available on the nodes after running the `bm-reimage.yml` playbook.

Important

Ensure that there is global read access to the file `/home/ardana/.ssh/authorized_keys`.

Now test passwordless SSH from the deployer and check your ability to remotely execute sudo commands:

```
ssh ardana@IP_OF_SLES_NODE "sudo tail -5 /var/log/messages"
```

17 Installation of SUSE CaaS Platform Heat Templates

This chapter describes how to install SUSE CaaS Platform Heat template on SUSE OpenStack Cloud.

17.1 SUSE CaaS Platform Heat Installation Procedure

PROCEDURE 17.1: PREPARATION

1. Download the latest SUSE CaaS Platform for OpenStack image (for example, `SUSE-CaaS-Platform-2.0-OpenStack-Cloud.x86_64-1.0.0-GM.qcow2`) from https://download.suse.com.

2. Upload the image to Glance:

```
openstack image create --public --disk-format qcow2 --container-format \
bare --file SUSE-CaaS-Platform-2.0-for-OpenStack-Cloud.x86_64-2.0.0-GM.qcow2 \
CaaSP-2
```

3. Install the `caasp-openstack-heat-templates` package on a machine with SUSE OpenStack Cloud repositories:

```
zypper in caasp-openstack-heat-templates
```

The installed templates are located in `/usr/share/caasp-openstack-heat-templates`.

PROCEDURE 17.2: INSTALLING TEMPLATES VIA HORIZON

1. In Horizon, go to *Project* › *Stacks* › *Launch Stack*.

2. Select *File* from the *Template Source* drop-down box and upload the `caasp-stack.yaml` file.

3. In the *Launch Stack* dialog, provide the required information (stack name, password, flavor size, external network of your environment, etc.).

4. Click *Launch* to launch the stack. This creates all required resources for running SUSE CaaS Platform in an OpenStack environment. The stack creates one Admin Node, one Master Node, and server worker nodes as specified.

PROCEDURE 17.3: INSTALL TEMPLATES FROM THE COMMAND LINE

1. Specify the appropriate flavor and network settings in the `caasp-environment.yaml` file.

2. Create a stack in Heat by passing the template, environment file, and parameters:

```
openstack stack create -t caasp-stack.yaml -e caasp-environment.yaml \
--parameter image=CaaSP-2 caasp-stack
```

PROCEDURE 17.4: ACCESSING VELUM SUSE CAAS PLATFORM DASHBOARD

1. After the stack has been created, the Velum SUSE CaaS Platform dashboard runs on the Admin Node. You can access it using the Admin Node's floating IP address.

2. Create an account and follow the steps in the Velum SUSE CaaS Platform dashboard to complete the SUSE CaaS Platform installation.

When you have successfully accessed the admin node web interface via the floating IP, follow the instructions at https://www.suse.com/documentation + /suse-caasp-2/book_caasp_deployment/data/book_caasp_deployment.html ↗ to continue the set up of SUSE CaaS Platform.

18 Integrations

Once you have completed your cloud installation, these are some of the common integrations you may want to perform.

18.1 Configuring for 3PAR Block Storage Backend

This page describes how to configure your 3PAR backend for the SUSE OpenStack Cloud Entry-scale with KVM cloud model.

18.1.1 Prerequisites

- You must have the license for the following software before you start your 3PAR backend configuration for the SUSE OpenStack Cloud Entry-scale with KVM cloud model:
  - Thin Provisioning
  - Virtual Copy
  - System Reporter
  - Dynamic Optimization
  - Priority Optimization
- Your SUSE OpenStack Cloud Entry-scale KVM Cloud should be up and running. Installation steps can be found in *Chapter 13, Installing Mid-scale and Entry-scale KVM*.
- Your 3PAR Storage Array should be available in the cloud management network or routed to the cloud management network and the 3PAR FC and iSCSI ports configured.
- The 3PAR management IP and iSCSI port IPs must have connectivity from the controller and compute nodes.
- Please refer to the system requirements for 3PAR in the OpenStack configuration guide, which can be found here: 3PAR System Requirements (http://docs.openstack.org/liberty/config-reference/content/hp-3par-sys-reqs.html) ↗.

18.1.2 Notes

Encrypted 3Par Volume: Attaching an encrypted 3Par volume is possible after installation by setting `volume_use_multipath = true` under the libvirt stanza in the `nova/kvm-hypervisor.conf.j2` file and reconfigure nova.

Concerning using multiple backends: If you are using multiple backend options, ensure that you specify each of the backends you are using when configuring your `cinder.conf.j2` file using a comma-delimited list. Also create multiple volume types so you can specify a backend to utilize when creating volumes. Instructions are included below.

Concerning iSCSI and Fiber Channel: You should not configure cinder backends so that multipath volumes are exported over both iSCSI and Fiber Channel from a 3PAR backend to the same Nova compute server.

3PAR driver has updated name: In the OpenStack Mitaka release, the 3PAR driver used for SUSE OpenStack Cloud integration had its name updated from `HP3PARFCDriver` and `HP3PARISCSIDriver` to `HPE3PARFCDriver` and `HPE3PARISCSIDriver`. To prevent issues when upgrading from previous SUSE OpenStack Cloud releases, we left the names as-is in the release and provided a mapping so that the integration would continue to work. This will produce a warning in the `cinder-volume.log` file advising you of the deprecated name. The warning will look similar to this: “Option "hp3par_api_url" from group "< *`YOUR_SECTION`* >" is deprecated. Use option "hpe3par_api_url" from group "< *`YOUR_SECTION`* >"”.

These are just warnings and can be ignored.

18.1.3 Multipath Support

If you want to enable multipath for Cinder volumes carved from 3PAR FC/iSCSI storage system, please go through the `~/openstack/ardana/ansible/roles/multipath/README.md` file on the Cloud Lifecycle Manager. The `README.md` file contains detailed procedures for configuring multipath for 3PAR FC/iSCSI Cinder volumes.

We have also included an additional set of steps needed if you are using 3PAR FC/iSCSI multipath which is included below.

If you are using 3PAR FC/iSCSI multipath, an additional configuration is required:

If you are planning on attaching an encrypted 3PAR volume after installation, ensure that you `volume_use_multipath = true` under the libvirt section in the `nova/kvm-hypervisor.conf.j2` file before configuring Cinder.

1. Log in to the Cloud Lifecycle Manager.
2. Edit the `~/openstack/my_cloud/config/nova/kvm-hypervisor.conf.j2` file add this line under the `[libvirt]` section:

   Example:

   ```
   [libvirt]
   ...
   iscsi_use_multipath=true
   ```
3. Edit the file `~/openstack/my_cloud/config/cinder/cinder.conf.j2` and add this line under the `[DEFAULT]` section:

   Example:

   ```
   [DEFAULT]
   ...
   use_multipath_for_image_xfer=true
   ```
4. Commit your configuration to the local git repo (*Chapter 12, Using Git for Configuration Management*), as follows:

   ```
   cd ~/openstack/ardana/ansible
   git add -A
   git commit -m "My config or other commit message"
   ```
5. Run the configuration processor:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost config-processor-run.yml
   ```
6. Use the playbook below to create a deployment directory:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```
7. Run the Nova reconfigure playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
   ```

18.1.4 Configure 3PAR FC as a Cinder Backend

You must modify the `cinder.conf.j2` to configure the FC details.

Perform the following steps to configure 3PAR FC as Cinder backend:

1. Log in to Cloud Lifecycle Manager.
2. Make the following changes to the `~/openstack/my_cloud/config/cinder/cinder.conf.j2` file:
   a. Add your 3PAR backend to the `enabled_backends` section:

      ```
      # Configure the enabled backends
      enabled_backends=3par_FC
      ```

      Important

      If you are using multiple backend types, you can use a comma-delimited list here.

   b. `[OPTIONAL]` If you want your volumes to use a default volume type, then enter the name of the volume type in the `[DEFAULT]` section with the syntax below. **Remember this value for when you create your volume type in the next section.**

      Important

      If you do not specify a default type then your volumes will default to a non-redundant RAID configuration. It is recommended that you create a volume type and specify it here that meets your environments needs.

      ```
      [DEFAULT]
      # Set the default volume type
      default_volume_type = <your new volume type>
      ```

   c. Uncomment the `StoreServ (3par) iscsi cluster` section and fill the values per your cluster information. Here is an example:

      ```
      [3par_FC]
      san_ip: <3par-san-ipaddr>
      san_login: <3par-san-username>
      san_password: <3par-san-password>
      hp3par_username: <3par-username>
      hp3par_password: <hp3par_password>
      hp3par_api_url: https://<3par-san-ipaddr>:8080/api/v1
      ```

```
hp3par_cpg: <3par-cpg-name-1>[,<3par-cpg-name-2>, ...]
volume_backend_name: <3par-backend-name>
volume_driver: cinder.volume.drivers.san.hp.hp_3par_fc.HP3PARFCDriver
```

Important

Do not use `backend_host` variable in `cinder.conf` file. If `backend_host` is set, it will override the [DEFAULT]/host value which SUSE OpenStack Cloud 8 is dependent on.

3. Commit your configuration to the local git repo (*Chapter 12, Using Git for Configuration Management*), as follows:

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

4. Run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Update your deployment directory:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the following playbook to complete the configuration:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
```

18.1.5 Configure 3PAR iSCSI as Cinder backend

You must modify the `cinder.conf.j2` to configure the iSCSI details.

Perform the following steps to configure 3PAR iSCSI as Cinder backend:

1. Log in to Cloud Lifecycle Manager.

2. Make the following changes to the `~/openstack/my_cloud/config/cinder/cinder.conf.j2` file:

   a. Add your 3PAR backend to the `enabled_backends` section:

   ```
   # Configure the enabled backends
   enabled_backends=3par_iSCSI
   ```

   b. Uncomment the `StoreServ (3par) iscsi cluster` section and fill the values per your cluster information. Here is an example:

   ```
   [3par_iSCSI]
   san_ip: <3par-san-ipaddr>
   san_login: <3par-san-username>
   san_password: <3par-san-password>
   hp3par_username: <3par-username>
   hp3par_password: <hp3par_password>
   hp3par_api_url: https://<3par-san-ipaddr>:8080/api/v1
   hp3par_cpg: <3par-cpg-name-1>[,<3par-cpg-name-2>, ...]
   volume_backend_name: <3par-backend-name>
   volume_driver: cinder.volume.drivers.san.hp.hp_3par_iscsi.HP3PARISCSIDriver
   hp3par_iscsi_ips: <3par-ip-address-1>[,<3par-ip-address-2>,<3par-ip-address-3>, ...]
   hp3par_iscsi_chap_enabled=true
   ```

   ### Important

   Do not use `backend_host` variable in `cinder.conf` file. If `backend_host` is set, it will override the [DEFAULT]/host value which SUSE OpenStack Cloud 8 is dependent on.

3. Commit your configuration your local git repository:

   ```
   cd ~/openstack/ardana/ansible
   git add -A
   git commit -m "<commit message>"
   ```

4. Run the configuration processor:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

   When you run the configuration processor you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the Ansible group_vars

and host_vars that it produces for subsequent deploy runs. You will need this key for subsequent Ansible deploy runs and subsequent configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise press Enter .
For CI purposes you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor

```
ansible-playbook -i hosts/localhost config-processor-run.yml \
  -e encrypt="" -e rekey=""
```

If you receive an error during either of these steps then there is an issue with one or more of your configuration files. We recommend that you verify that all of the information in each of your configuration files is correct for your environment and then commit those changes to git using the instructions above.

5. Run the following command to create a deployment directory.

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the following command to complete the configuration:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
```

18.1.6 Post-Installation Tasks

After configuring 3PAR as your Block Storage backend, perform the following tasks:

- *Book "Operations Guide", Chapter 7 "Managing Block Storage", Section 7.1 "Managing Block Storage using Cinder", Section 7.1.2 "Creating a Volume Type for your Volumes"*
- *Section 23.1, "Verifying Your Block Storage Backend"*

18.2 Ironic HPE OneView Integration

SUSE OpenStack Cloud 8 supports integration of Ironic (Baremetal) service with HPE OneView using *agent_pxe_oneview* driver. Please refer to OpenStack Documentation (https://docs.openstack.org/developer/ironic/drivers/oneview.html) for more information.

18.2.1 Prerequisites

1. Installed SUSE OpenStack Cloud 8 with entry-scale-ironic-flat-network or entry-scale-ironic-multi-tenancy model.
2. HPE OneView 3.0 instance is running and connected to management network.
3. HPE OneView configuration is set into `definition/data/ironic/ironic_config.yml` (and `ironic-reconfigure.yml` playbook ran if needed). This should enable *agent_pxe_oneview* driver in ironic conductor.
4. Managed node(s) should support PXE booting in legacy BIOS mode.
5. Managed node(s) should have PXE boot NIC listed first. That is, embedded 1Gb NIC must be disabled (otherwise it always goes first).

18.2.2 Integrating with HPE OneView

1. On the Cloud Lifecycle Manager, open the file `~/openstack/my_cloud/definition/data/ironic/ironic_config.yml`

```
~$ cd ~/openstack
vi my_cloud/definition/data/ironic/ironic_config.yml
```

2. Modify the settings listed below:

   a. `enable_oneview`: should be set to "true" for HPE OneView integration

   b. `oneview_manager_url`: HTTPS endpoint of HPE OneView management interface, for example: **https://10.0.0.10/**

   c. `oneview_username`: HPE OneView username, for example: **Administrator**

   d. `oneview_encrypted_password`: HPE OneView password in encrypted or clear text form. The encrypted form is distinguished by presence of `@ardana@` at the beginning of the string. The encrypted form can be created by running the **`ardanaencrypt.py`** program. This program is shipped as part of SUSE OpenStack Cloud and can be found in `~/openstack/ardana/ansible` directory on Cloud Lifecycle Manager.

   e. `oneview_allow_insecure_connections`: should be set to "true" if HPE OneView is using self-generated certificate.

3. Once you have saved your changes and exited the editor, add files, commit changes to local git repository, and run `config-processor-run.yml` and `ready-deployment.yml` playbooks, as described in *Chapter 12, Using Git for Configuration Management*.

```
~/openstack$ git add my_cloud/definition/data/ironic/ironic_config.yml
~/openstack$ cd ardana/ansible
~/openstack/ardana/ansible$ ansible-playbook -i hosts/localhost \
  config-processor-run.yml
...
~/openstack/ardana/ansible$ ansible-playbook -i hosts/localhost \
  ready-deployment.yml
```

4. Run ironic-reconfigure.yml playbook.

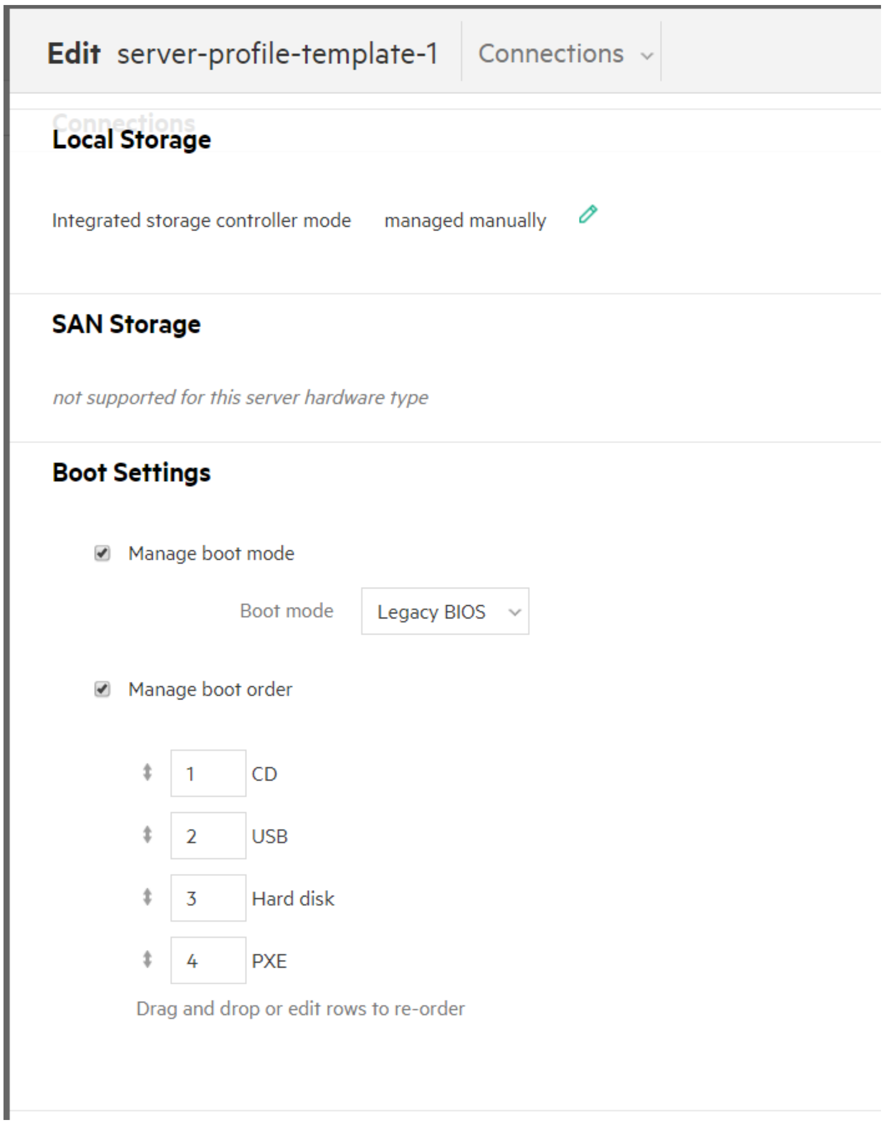
```
$ cd ~/scratch/ansible/next/ardana/ansible/

# This is needed if password was encrypted in ironic_config.yml file
~/scratch/ansible/next/ardana/ansible$ export
 ARDANA_USER_PASSWORD_ENCRYPT_KEY=your_password_encrypt_key
~/scratch/ansible/next/ardana/ansible$ ansible-playbook -i hosts/verb_hosts ironic-
reconfigure.yml
...
```

18.2.3 Registering Node in HPE OneView

In the HPE OneView web interface:

1. Navigate to *Menu* › *Server Hardware*. Add new *Server Hardware* item, using managed node IPMI IP and credentials. If this is the first node of this type being added, corresponding *Server Hardware Type* will be created automatically.

2. Navigate to *Menu* › *Server Profile Template*. Add *Server Profile Template*. Use *Server Hardware Type* corresponding to node being registered. In *BIOS Settings* section, set *Manage Boot Mode* and *Manage Boot Order* options must be turned on:



Edit server-profile-template-1 | Connections

Connections

Local Storage

Integrated storage controller mode — managed manually

SAN Storage

not supported for this server hardware type

Boot Settings

☑ Manage boot mode

Boot mode: Legacy BIOS

☑ Manage boot order

1	CD
2	USB
3	Hard disk
4	PXE

Drag and drop or edit rows to re-order

3. Verify that node is powered off. Power the node off if needed.

18.2.4 Provisioning Ironic Node

1. Login to the Cloud Lifecycle Manager and source respective credentials file (for example `service.osrc` for admin account).
2. Review glance images with `glance image-list`

```
$ glance image-list
+--------------------------------------+--------------------------+
| ID                                   | Name                     |
+--------------------------------------+--------------------------+
| c61da588-622c-4285-878f-7b86d87772da | cirros-0.3.4-x86_64      |
+--------------------------------------+--------------------------+
```

Ironic deploy images (boot image, `ir-deploy-kernel`, `ir-deploy-ramdisk`, `ir-deploy-iso`) are created automatically. The `agent_pxe_oneview` Ironic driver requires `ir-deploy-kernel` and `ir-deploy-ramdisk` images.

3. Create node using `agent_pxe_oneview` driver.

```
$ ironic --ironic-api-version 1.22 node-create -d agent_pxe_oneview --name test-node-1 \
  --network-interface neutron -p memory_mb=131072 -p cpu_arch=x86_64 -p local_gb=80 -p cpus=2 \
  -p 'capabilities=boot_mode:bios,boot_option:local,server_hardware_type_uri:\
     /rest/server-hardware-types/E5366BF8-7CBF-48DF-A752-8670CF780BB2,server_profile_template_uri:
\
     /rest/server-profile-templates/00614918-77f8-4146-a8b8-9fc276cd6ab2' \
  -i 'server_hardware_uri=/rest/server-hardware/32353537-3835-584D-5135-313930373046' \
  -i dynamic_allocation=True \
  -i deploy_kernel=633d379d-e076-47e6-b56d-582b5b977683 \
  -i deploy_ramdisk=d5828785-edf2-49fa-8de2-3ddb7f3270d5

+-------------------+-------------------------------------------------------------------------+
| Property          | Value                                                                   |
+-------------------+-------------------------------------------------------------------------+
| chassis_uuid      |                                                                         |
| driver            | agent_pxe_oneview                                                       |
| driver_info       | {u'server_hardware_uri': u'/rest/server-                                |
|                   | hardware/32353537-3835-584D-5135-313930373046', u'dynamic_allocation':  |
|                   | u'True', u'deploy_ramdisk': u'd5828785-edf2-49fa-8de2-3ddb7f3270d5',    |
|                   | u'deploy_kernel': u'633d379d-e076-47e6-b56d-582b5b977683'}              |
| extra             | {}                                                                      |
| name              | test-node-1                                                             |
| network_interface | neutron                                                                 |
| properties        | {u'memory_mb': 131072, u'cpu_arch': u'x86_64', u'local_gb': 80, u'cpus': |
|                   | 2, u'capabilities':                                                     |
|                   | u'boot_mode:bios,boot_option:local,server_hardware_type_uri:/rest       |
|                   | /server-hardware-types/E5366BF8-7CBF-                                   |
|                   | 48DF-A752-8670CF780BB2,server_profile_template_uri:/rest/server-profile- |
|                   | templates/00614918-77f8-4146-a8b8-9fc276cd6ab2'}                        |
| resource_class    | None                                                                    |
| uuid              | c202309c-97e2-4c90-8ae3-d4c95afdaf06                                    |
+-------------------+-------------------------------------------------------------------------+
```

Note

- For deployments created via Ironic/HPE OneView integration, `memory_mb` property must reflect physical amount of RAM installed in the managed node. That is, for a server with 128 Gb of RAM it works out to 132*1024=13072.
- Boot mode in capabilities property must reflect boot mode used by the server, that is 'bios' for Legacy BIOS and 'uefi' for UEFI.
- Values for `server_hardware_type_uri`, `server_profile_template_uri` and `server_hardware_uri` can be grabbed from browser URL field while navigating to respective objects in HPE OneView UI. URI corresponds to the part of URL which starts form the token `/rest`. That is, the URL `https://oneview.mycorp.net/#/profile-templates/show/overview/r/rest/server-profile-templates/12345678-90ab-cdef-0123-012345678901` corresponds to the URI `/rest/server-profile-templates/12345678-90ab-cdef-0123-012345678901`.
- Grab IDs of `deploy_kernel` and `deploy_ramdisk` from **glance image-list** output above.

4. Create port.

```
$ ironic --ironic-api-version 1.22 port-create \
  --address aa:bb:cc:dd:ee:ff \
  --node c202309c-97e2-4c90-8ae3-d4c95afdaf06 \
  -l switch_id=ff:ee:dd:cc:bb:aa \
  -l switch_info=MY_SWITCH \
  -l port_id="Ten-GigabitEthernet 1/0/1" \
  --pxe-enabled true
+-----------------------+----------------------------------------------------------------+
| Property              | Value                                                          |
+-----------------------+----------------------------------------------------------------+
| address               | 8c:dc:d4:b5:7d:1c                                              |
| extra                 | {}                                                             |
| local_link_connection | {u'switch_info': u'C20DATA', u'port_id': u'Ten-GigabitEthernet |
|                       | 1/0/1',    u'switch_id': u'ff:ee:dd:cc:bb:aa'}                 |
| node_uuid             | c202309c-97e2-4c90-8ae3-d4c95afdaf06                           |
| pxe_enabled           | True                                                           |
| uuid                  | 75b150ef-8220-4e97-ac62-d15548dc8ebe                           |
+-----------------------+----------------------------------------------------------------+
```

Warning

Ironic Multi-Tenancy networking model is used in this example. Therefore, ironic port-create command contains information about the physical switch. HPE OneView integration can also be performed using the Ironic Flat Networking model. For more information, see *Book "Planning an Installation with Cloud Lifecycle Manager", Chapter 10 "Example Configurations", Section 10.6 "Ironic Examples"*.

5. Move node to manageable provisioning state. The connectivity between Ironic and HPE OneView will be verified, Server Hardware Template settings validated, and Server Hardware power status retrieved from HPE OneView and set into the Ironic node.

```
$ ironic node-set-provision-state test-node-1 manage
```

6. Verify that node power status is populated.

```
$ ironic node-show test-node-1
+-----------------------+-------------------------------------------------------------------------+
| Property              | Value                                                                   |
+-----------------------+-------------------------------------------------------------------------+
| chassis_uuid          |                                                                         |
| clean_step            | {}                                                                      |
| console_enabled       | False                                                                   |
| created_at            | 2017-06-30T21:00:26+00:00                                               |
| driver                | agent_pxe_oneview                                                       |
| driver_info           | {u'server_hardware_uri': u'/rest/server-                                |
|                       | hardware/32353537-3835-584D-5135-313930373046', u'dynamic_allocation':|
|                       | u'True', u'deploy_ramdisk': u'd5828785-edf2-49fa-8de2-3ddb7f3270d5',  |
|                       | u'deploy_kernel': u'633d379d-e076-47e6-b56d-582b5b977683'}             |
| driver_internal_info  | {}                                                                      |
| extra                 | {}                                                                      |
| inspection_finished_at| None                                                                    |
| inspection_started_at | None                                                                    |
| instance_info         | {}                                                                      |
| instance_uuid         | None                                                                    |
| last_error            | None                                                                    |
| maintenance           | False                                                                   |
| maintenance_reason    | None                                                                    |
| name                  | test-node-1                                                             |
| network_interface     |                                                                         |
| power_state           | power off                                                               |
| properties            | {u'memory_mb': 131072, u'cpu_arch': u'x86_64', u'local_gb': 80,         |
|                       | u'cpus': 2, u'capabilities':                                            |
|                       | u'boot_mode:bios,boot_option:local,server_hardware_type_uri:/rest      |
|                       | /server-hardware-types/E5366BF8-7CBF-                                   |
|                       | 48DF-A752-86...BB2,server_profile_template_uri:/rest/server-profile-   |
|                       | templates/00614918-77f8-4146-a8b8-9fc276cd6ab2'}                        |
| provision_state       | manageable                                                              |
| provision_updated_at  | 2017-06-30T21:04:43+00:00                                        |
| raid_config           |                                                                  |
| reservation           | None                                                             |
| resource_class        |                                                                  |
| target_power_state    | None                                                             |
| target_provision_state| None                                                             |
| target_raid_config    |                                                                  |
| updated_at            | 2017-06-30T21:04:43+00:00                                        |
| uuid                  | c202309c-97e2-4c90-8ae3-d4c95afdaf06                             |
+-----------------------+------------------------------------------------------------------+
```

7. Move node to available provisioning state. The Ironic node will be reported to Nova as available.

```
$ ironic node-set-provision-state test-node-1 provide
```

8. Verify that node resources were added to Nova hypervisor stats.

```
$ nova hypervisor-stats
+----------------------+--------+
| Property             | Value  |
+----------------------+--------+
| count                | 1      |
| current_workload     | 0      |
| disk_available_least | 80     |
| free_disk_gb         | 80     |
| free_ram_mb          | 131072 |
| local_gb             | 80     |
| local_gb_used        | 0      |
| memory_mb            | 131072 |
| memory_mb_used       | 0      |
| running_vms          | 0      |
| vcpus                | 2      |
| vcpus_used           | 0      |
+----------------------+--------+
```

9. Create Nova flavor.

```
$ nova flavor-create m1.ironic auto 131072 80 2
+-------------+-----------+--------+------+-----------+------+-------+-------------+-----------+
| ID          | Name      | Mem_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+-------------+-----------+--------+------+-----------+------+-------+-------------+-----------+
| 33c8...f8d8 | m1.ironic | 131072 | 80   | 0         |      | 2     | 1.0         | True      |
+-------------+-----------+--------+------+-----------+------+-------+-------------+-----------+
$ nova flavor-key m1.ironic set capabilities:boot_mode="bios"
$ nova flavor-key m1.ironic set capabilities:boot_option="local"
$ nova flavor-key m1.ironic set cpu_arch=x86_64
```

Note

All parameters (specifically, amount of RAM and boot mode) must correspond to ironic node parameters.

10. Create Nova keypair if needed.

```
$ nova keypair-add ironic_kp --pub-key ~/.ssh/id_rsa.pub
```

11. Boot Nova instance.

```
$ nova boot --flavor m1.ironic --image d6b5...e942 --key-name ironic_kp \
  --nic net-id=5f36...dcf3 test-node-1
+-------------------------------+-----------------------------------------------------+
| Property                      | Value                                               |
+-------------------------------+-----------------------------------------------------+
| OS-DCF:diskConfig             | MANUAL                                              |
| OS-EXT-AZ:availability_zone   |                                                     |
| OS-EXT-SRV-ATTR:host          | -                                                   |
| OS-EXT-SRV-ATTR:              |                                                     |
|       hypervisor_hostname     | -                                                   |
| OS-EXT-SRV-ATTR:instance_name |                                                     |
| OS-EXT-STS:power_state        | 0                                                   |
| OS-EXT-STS:task_state         | scheduling                                          |
| OS-EXT-STS:vm_state           | building                                            |
| OS-SRV-USG:launched_at        | -                                                   |
| OS-SRV-USG:terminated_at      | -                                                   |
| accessIPv4                    |                                                     |
| accessIPv6                    |                                                     |
| adminPass                     | pE3m7wRACvYy                                        |
| config_drive                  |                                                     |
| created                       | 2017-06-30T21:08:42Z                                |
| flavor                        | m1.ironic (33c81884-b8aa-46...3b72f8d8)             |
| hostId                        |                                                     |
| id                            | b47c9f2a-e88e-411a-abcd-6172aea45397                |
| image                         | Ubuntu Trusty 14.04 BIOS (d6b5d971-42...5f2d88e942) |
| key_name                      | ironic_kp                                           |
| metadata                      | {}                                                  |
| name                          | test-node-1                                         |
| os-extended-volumes:          |                                                     |
|       volumes_attached        | []                                                  |
| progress                      | 0                                                   |
| security_groups               | default                                             |
| status                        | BUILD                                               |
| tenant_id                     | c8573f7026d24093b40c769ca238fddc                    |
| updated                       | 2017-06-30T21:08:42Z                                |
| user_id                       | 2eae99221545466d8f175eeb566cc1b4                    |
+------------------------------+---------------------------------------------------+
```

During nova instance boot, the following operations will be performed by Ironic via HPE OneView REST API.

- In HPE OneView, new Server Profile is generated for specified Server Hardware, using specified Server Profile Template. Boot order in Server Profile is set to list PXE as the first boot source.
- The managed node is powered on and boots IPA image from PXE.
- IPA image writes user image onto disk and reports success back to Ironic.
- Ironic modifies Server Profile in HPE OneView to list 'Disk' as default boot option.
- Ironic reboots the node (via HPE OneView REST API call).

18.3 SUSE Enterprise Storage Integration

The current version of SUSE OpenStack Cloud supports integration with SUSE Enterprise Storage. Integrating SUSE Enterprise Storage enables Ceph block storage as well as object and image storage services in SUSE OpenStack Cloud.

18.3.1 Enabling SUSE Enterprise Storage Integration

The SUSE Enterprise Storage integration is provided through the `ardana-ses` RPM package. As this package is included in the `patterns-cloud-ardana` pattern, its installation is covered in *Chapter 3, Installing the Cloud Lifecycle Manager server*.

18.3.2 Configuration

After the SUSE Enterprise Storage integration package has been installed, it must be configured in order to work. Files that contain relevant SUSE Enterprise Storage/Ceph deployment information must be placed into a directory on the deployer node. This includes the configuration file that describes various aspects of the Ceph environment as well as keyrings for each user and pool created in the Ceph environment. In addition to that, you need to edit the `settings.yml` file to enable the SUSE Enterprise Storage integration to run and update all of the SUSE OpenStack Cloud service configuration files.

The `settings.yml` file must reside in the `~/openstack/my_config/confg/ses/` directory. Open the file for editing, uncomment the `ses_config_path:` parameter, and specify the location on the deployer host containing the `ses_config.yml` and keyring files as the parameter's value. After you have done that, the `site.yml` and `ardana-reconfigure.yml` playbooks activates the SUSE Enterprise Storage integration and configures the Cinder, Glance, and Nova services.

For Ceph, it is necessary to create pools and users to allow the SUSE OpenStack Cloud services to use the SUSE Enterprise Storage/Ceph cluster. Pools and users must be created for Cinder, Cinder backup, Nova and Glance. After the required pools and users are set up on the SUSE Enterprise Storage/Ceph cluster, you have to create a `ses_config.yml` configuration file (see the example below). This file is used during deployment to configure all of the services. The `ses_config.yml` and the keyring files should be placed in a separate directory. The path to this directory must be specified in the `settings.yml` file.

EXAMPLE 18.1: SES_CONFIG.YML EXAMPLE

```
ses_cluster_configuration:
    ses_cluster_name: ceph
    ses_radosgw_url: "https://192.168.56.8:8080/swift/v1"

    conf_options:
        ses_fsid: d5d7c7cb-5858-3218-a36f-d028df7b1111
        ses_mon_initial_members: ses-osd2, ses-osd3, ses-osd1
        ses_mon_host: 192.168.56.8, 192.168.56.9, 192.168.56.7
        ses_public_network: 192.168.56.0/21
        ses_cluster_network: 192.168.56.0/21

    cinder:
        rbd_store_pool: cinder
        rbd_store_pool_user: cinder
        keyring_file_name: ceph.client.cinder.keyring

    cinder-backup:
        rbd_store_pool: backups
        rbd_store_pool_user: cinder_backup
        keyring_file_name: ceph.client.cinder-backup.keyring

    # Nova uses the cinder user to access the nova pool, cinder pool
    # So all we need here is the nova pool name.
    nova:
        rbd_store_pool: nova

    glance:
        rbd_store_pool: glance
        rbd_store_pool_user: glance
        keyring_file_name: ceph.client.glance.keyring
```

19 Troubleshooting the Installation

We have gathered some of the common issues that occur during installation and organized them by when they occur during the installation. These sections will coincide with the steps labeled in the installation instructions.

- *Section 19.1, “Issues during Cloud Lifecycle Manager Setup”*
- *Section 19.2, “Issues while Updating Configuration Files”*
- *Section 19.3, “Issues while Deploying the Cloud”*

19.1 Issues during Cloud Lifecycle Manager Setup

Issue: Running the ardana-init.bash script when configuring your Cloud Lifecycle Manager does not complete

Part of what the `ardana-init.bash` script does is install Git. So if your DNS server(s) is/are not specified in your `/etc/resolv.conf` file, is not valid, or is not functioning properly on your Cloud Lifecycle Manager, it will not be able to complete.

To resolve this issue, double check your nameserver in your `/etc/resolv.conf` file and then re-run the script.

19.2 Issues while Updating Configuration Files

Configuration Processor Fails Due to Wrong yml Format

If you receive the error below when running the configuration processor then you may have a formatting error:

```
TASK: [fail msg="Configuration processor run failed, see log output above for
details"]
```

First you should check the Ansible log in the location below for more details on which yml file in your input model has the error:

```
~/.ansible/ansible.log
```

Check the configuration file to locate and fix the error, keeping in mind the following tips below.

Check your files to ensure that they don't contain the following:

- Non-ascii characters
- Unneeded spaces

Once you have fixed the formatting error in your files, commit the changes with these steps:

1. Commit your changes to Git:

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

2. Re-run the configuration processor playbook and confirm the error is not received again.

Configuration processor fails with provider network OCTAVIA-MGMT-NET error

If you receive the error below when running the configuration processor then you have not correctly configured your VLAN settings for Octavia.

```
"################################################################################",
"# The configuration processor failed.  ",
"#   config-data-2.0           ERR: Provider network OCTAVIA-MGMT-NET host_routes:
"# destination '192.168.10.0/24' is not defined as a Network in the input model.
""# Add 'external: True' to this host_route if this is for an external network.",
"################################################################################"
```

To resolve the issue, ensure that your settings in `~/openstack/my_cloud/definition/data/neutron/neutron_config.yml` are correct for the VLAN setup for Octavia.

Changes Made to your Configuration Files

If you have made corrections to your configuration files and need to re-run the Configuration Processor, the only thing you need to do is commit your changes to your local Git repository:

```
cd ~/openstack/ardana/ansible
```

```
git add -A
git commit -m "commit message"
```

You can then re-run the configuration processor:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

Configuration Processor Fails Because Encryption Key Does Not Meet Requirements

If you choose to set an encryption password when running the configuration processor, you may receive the following error if the chosen password does not meet the complexity requirements:

```
################################################################################
# The configuration processor failed.
#   encryption-key ERR: The Encryption Key does not meet the following requirement(s):
#       The Encryption Key must be at least 12 characters
#       The Encryption Key must contain at least 3 of following classes of characters:
#                           Uppercase Letters, Lowercase Letters, Digits, Punctuation
################################################################################
```

If you receive the above error, simply run the configuration processor again and select a password that meets the complexity requirements detailed in the error message:

```
cd ~/openstack/ardana/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

19.3 Issues while Deploying the Cloud

Issue: If the site.yml playbook fails, you can query the log for the reason

Ansible is good about outputting the errors into the command line output, however if you'd like to view the full log for any reason the location is:

```
~/.ansible/ansible.log
```

This log is updated real time as you run Ansible playbooks.

Tip

Use grep to parse through the log. Usage: `grep <text> ~/.ansible/ansible.log`

Issue: How to Wipe the Disks of your Machines

If you have re-run the `site.yml` playbook, you may need to wipe the disks of your nodes

You would generally run the playbook below after re-running the `bm-reimage.yml` playbook but before you re-run the `site.yml` playbook.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts wipe_disks.yml
```

The playbook will show you the disks to be wiped in the output and allow you to confirm that you want to complete this action or abort it if you do not want to proceed. You can optionally use the `--limit <NODE_NAME>` switch on this playbook to restrict it to specific nodes.

If you receive an error stating that `osconfig` has already run on your nodes then you will need to remove the `/etc/ardana/osconfig-ran` file on each of the nodes you want to wipe with this command:

```
sudo rm /etc/ardana/osconfig-ran
```

That will clear this flag and allow the disk to be wiped.

Issue: Freezer installation fails if an independent network is used for the External_API

The Freezer installation fails if an independent network is used for the External_API. If you intend to deploy the External API on an independent network, the following changes need to be made:

In `roles/freezer-agent/defaults/main.yml` add the following line:

```
backup_freezer_api_url: "{{ FRE_API | item('advertises.vips.private[0].url', default=' ') }}"
```

In `roles/freezer-agent/templates/backup.osrc.j2` add the following line:

```
export OS_FREEZER_URL={{ backup_freezer_api_url }}
```

Error Received if Root Logical Volume is Too Small

When running the `site.yml` playbook, you may receive a message that includes the error below if your root logical volume is too small. This error needs to be parsed out and resolved.

```
2015-09-29 15:54:03,022 p=26345 u=stack | stderr: New size given (7128 extents)
not larger than existing size (7629 extents)
```

The error message may also reference the root volume:

```
"name": "root", "size": "10%"
```

The problem here is that the root logical volume, as specified in the `disks_controller.yml` file, is set to `10%` of the overall physical volume and this value is too small.

To resolve this issue you need to ensure that the percentage is set properly for the size of your logical-volume. The default values in the configuration files is based on a 500 GB disk, so if your logical volumes are smaller you may need to increase the percentage so there is enough room.

Multiple Keystone Failures Received during site.yml

If you receive the Keystone error below during your `site.yml` run then follow these steps:

```
TASK: [OPS-MON | _keystone_conf | Create Ops Console service in Keystone] *****
failed:
[...]
msg: An unexpected error prevented the server from fulfilling your request.
(HTTP 500) (Request-ID: req-23a09c72-5991-4685-b09f-df242028d742), failed

FATAL: all hosts have already failed -- aborting
```

The most likely cause of this error is that the virtual IP address is having issues and the Keystone API communication through the virtual IP address is not working properly. You will want to check the Keystone log on the controller where you will likely see authorization failure errors.

Verify that your virtual IP address is active and listening on the proper port on all of your controllers using this command:

```
netstat -tplan | grep 35357
```

Ensure that your Cloud Lifecycle Manager did not pick the wrong (unusable) IP address from the list of IP addresses assigned to your Management network.

The Cloud Lifecycle Manager will take the first available IP address after the `gateway-ip` defined in your `~/openstack/my_cloud/definition/data/networks.yml` file. This IP will be used as the virtual IP address for that particular network. If this IP address is used and reserved for another purpose outside of your SUSE OpenStack Cloud deployment then you will receive the error above.

To resolve this issue we recommend that you utilize the `start-address` and possibly the `end-address` (if needed) options in your `networks.yml` file to further define which IP addresses you want your cloud deployment to use. For more information, see *Book "*Planning an Installation with Cloud Lifecycle Manager*", Chapter 7 "Configuration Objects", Section 7.14 "Networks"*.

After you have made changes to your `networks.yml` file, follow these steps to commit the changes:

1. Ensuring that you stay within the `~/openstack` directory, commit the changes you just made:

   ```
   cd ~/openstack
   git commit -a -m "commit message"
   ```

2. Run the configuration processor:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost config-processor-run.yml
   ```

3. Update your deployment directory:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

4. Re-run the site.yml playbook:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts site.yml
   ```

20 Troubleshooting the ESX

This section contains troubleshooting tasks for your SUSE® OpenStack Cloud 8 for ESX.

20.1 Issue: Ardana-ux-services.service is not running

If you perform any maintenance work or reboot the Cloud Lifecycle Manager/deployer node, make sure to restart the Cloud Lifecycle Manager UX service for standalone deployer node and shared Cloud Lifecycle Manager/controller node based on your environment.

For standalone deployer node, execute `ardana-start.yml` playbook to restart the Cloud Lifecycle Manager UX services on the deployer node after a reboot.

For shared deployer/controller node, execute `ardana-start.yml` playbook on all the controllers to restart Cloud Lifecycle Manager UX services.

For example:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-start.yml --limit HOST_NAME
```

Replace *`HOST_NAME`* with the host name of the Cloud Lifecycle Manager node or the Cloud Lifecycle Manager Node/Shared Controller.

20.2 Issue: ESX Cluster shows UNKNOWN in Operations Console

In the Operations Console Alarms dashboard, if all the alarms for ESX cluster are showing UNKNOWN then restart the `openstack-monasca-agent` running in ESX compute proxy.

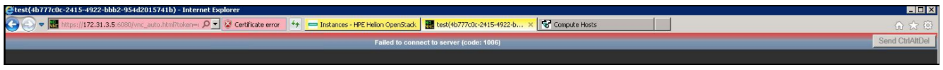
1. SSH to the respective compute proxy. You can find the hostname of the proxy from the dimensions list shown against the respective alarm.

2. Restart the `openstack-monasca-agent` service.

   ```
   sudo systemctl restart openstack-monasca-agent
   ```

20.3 Issue: Unable to view the VM console in Horizon UI

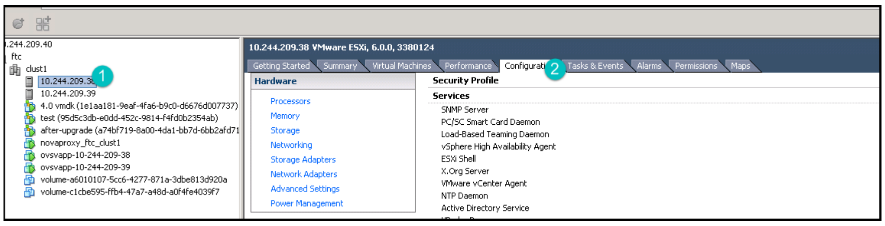
By default the gdbserver firewall is disabled in ESXi host which results in a Handshake error when accessing the VM instance console in the Horizon UI.


test(4b777c0c-2415-4922-bbb2-954d2015741b) - Internet Explorer
https://172.31.3.5:6080/vnc_auto.html?token=
Certificate error
Instances - HPE Helion OpenStack
test(4b777c0c-2415-4922-b...
Compute Hosts
Send CtrlAltDel

Failed to connect to server (code: 1006)

Procedure to enable gdbserver

1. Login to vSphere Client.
2. Select the ESXi Host and click *Configuration* tab in the menu bar. You must perform the following actions on all the ESXi hosts in the compute clusters.



.244.209.40
ftc
clust1
10.244.209.38
10.244.209.39
4.0 vmdk (1e1aa181-9eaf-4fa6-b9c0-d6676d007737)
test (95d5c3db-e0dd-452c-9814-f4fd0b2354ab)
after-upgrade (a74bf719-8a00-4da1-bb7d-6bb2afd71
novaproxy_ftc_clust1
ovsvapp-10-244-209-38
ovsvapp-10-244-209-39
volume-a6010107-5cc6-4277-871a-3dbe813d920a
volume-c1cbe595-ffb4-47a7-a48d-a0f4fe4039f7

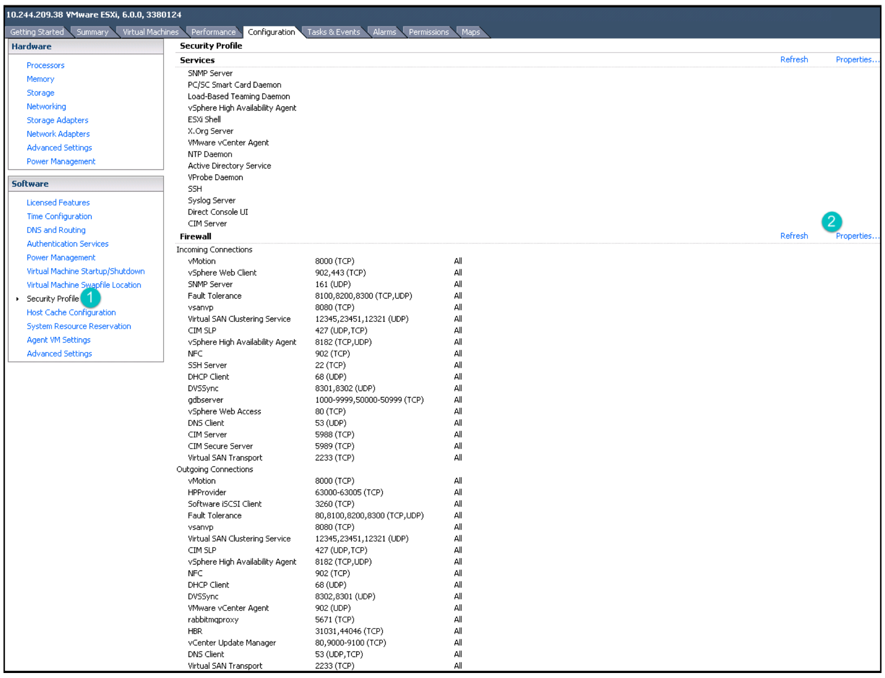
10.244.209.38 VMware ESXi, 6.0.0, 3380124

Security Profile

Services

SNMP Server
PC/SC Smart Card Daemon
Load-Based Teaming Daemon
vSphere High Availability Agent
ESXi Shell
X.Org Server
VMware vCenter Agent
NTP Daemon
Active Directory Service

3. On the left hand side select **Security Profile** from the list of **Software**. Click **Properties** on the right hand side.



10.244.209.38 VMware ESXi, 6.0.0, 3380124

Security Profile

Services

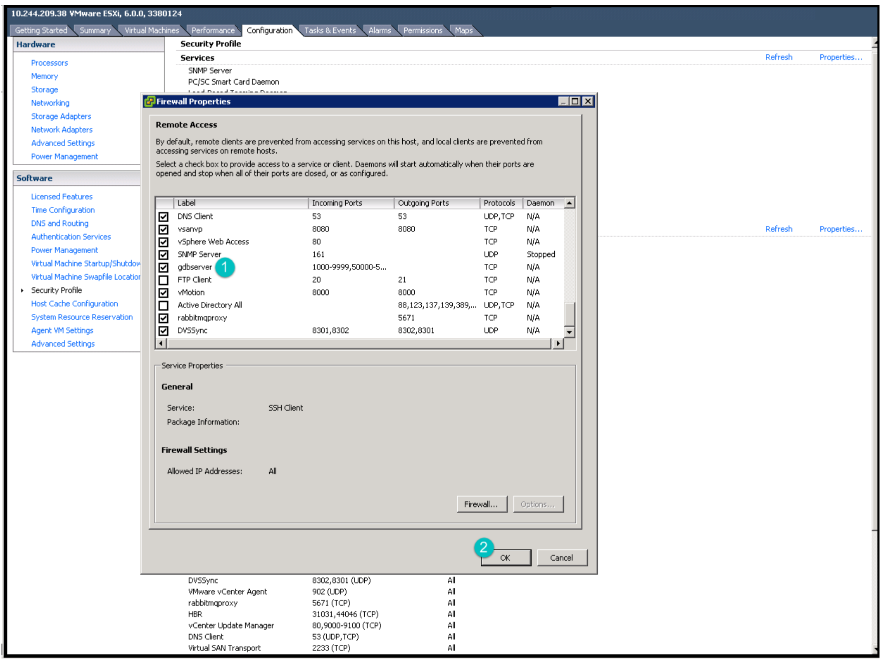
SNMP Server
PC/SC Smart Card Daemon
Load-Based Teaming Daemon
vSphere High Availability Agent
ESXi Shell
X.Org Server
VMware vCenter Agent
NTP Daemon
Active Directory Service
VProbe Daemon
SSH
Syslog Server
Direct Console UI
CIM Server

Firewall

Incoming Connections		
vMotion	8000 (TCP)	All
vSphere Web Client	902,443 (TCP)	All
SNMP Server	161 (UDP)	All
Fault Tolerance	8100,8200,8300 (TCP,UDP)	All
vsanvp	8080 (TCP)	All
Virtual SAN Clustering Service	12345,23451,12321 (UDP)	All
CIM SLP	427 (UDP,TCP)	All
vSphere High Availability Agent	8182 (TCP,UDP)	All
NFC	902 (TCP)	All
SSH Server	22 (TCP)	All
DHCP Client	68 (UDP)	All
DVSSync	8301,8302 (UDP)	All
gdbserver	1000-9999,50000-50999 (TCP)	All
vSphere Web Access	80 (TCP)	All
DNS Client	53 (UDP)	All
CIM Server	5988 (TCP)	All
CIM Secure Server	5989 (TCP)	All
Virtual SAN Transport	2233 (TCP)	All
Outgoing Connections		
vMotion	8000 (TCP)	All
HPProvider	63000-63005 (TCP)	All
Software iSCSI Client	3260 (TCP)	All
Fault Tolerance	80,8100,8200,8300 (TCP,UDP)	All
vsanvp	8080 (TCP)	All
Virtual SAN Clustering Service	12345,23451,12321 (UDP)	All
CIM SLP	427 (UDP,TCP)	All
vSphere High Availability Agent	8182 (TCP,UDP)	All
NFC	902 (TCP)	All
DHCP Client	68 (UDP)	All
DVSSync	8302,8301 (UDP)	All
VMware vCenter Agent	902 (UDP)	All
rabbitmqproxy	5671 (TCP)	All
HBR	31031,44046 (TCP)	All
vCenter Update Manager	80,9000-9100 (TCP)	All
DNS Client	53 (UDP,TCP)	All
Virtual SAN Transport	2233 (TCP)	All

Firewall Properties box displays.

4. Select **gdbserver** checkbox and click **OK**.



Remote Access

By default, remote clients are prevented from accessing services on this host, and local clients are prevented from accessing services on remote hosts.

Select a check box to provide access to a service or client. Daemons will start automatically when their ports are opened and stop when all of their ports are closed, or as configured.

	Label	Incoming Ports	Outgoing Ports	Protocols	Daemon
✓	DNS Client	53	53	UDP,TCP	N/A
✓	vsanvp	8080	8080	TCP	N/A
✓	vSphere Web Access	80		TCP	N/A
✓	SNMP Server	161		UDP	Stopped
✓	gdbserver	1000-9999,50000-5...		TCP	N/A
☐	FTP Client	20	21	TCP	N/A
✓	vMotion	8000	8000	TCP	N/A
☐	Active Directory All		88,123,137,139,389,...	UDP,TCP	N/A
✓	rabbitmqproxy		5671	TCP	N/A
✓	DVSSync	8301,8302	8302,8301	UDP	N/A

Service Properties

General

Service: SSH Client

Package Information:

Firewall Settings

Allowed IP Addresses: All

DVSSync	8302,8301 (UDP)	All
VMware vCenter Agent	902 (UDP)	All
rabbitmqproxy	5671 (TCP)	All
HBR	31031,44046 (TCP)	All
vCenter Update Manager	80,9000-9100 (TCP)	All
DNS Client	53 (UDP,TCP)	All
Virtual SAN Transport	2233 (TCP)	All

III Post-Installation

21 Overview

Once you have completed your cloud deployment, these are some of the common post-installation tasks you may need to perform. Take a look at the descriptions below to determine which of these you need to do.

22 Cloud Verification

Once you have completed your cloud deployment, these are some of the common post-installation tasks you may need to perform to verify your cloud installation.

22.1 API Verification

SUSE OpenStack Cloud 8 provides a tool, Tempest, that you can use to verify that your cloud deployment completed successfully:

- *Section 22.1.1, “Prerequisites”*
- *Section 22.1.2, “Tempest Integration Tests”*
  - *Section 22.1.3, “Running the Tests”*
  - *Section 22.1.4, “Viewing Test Results”*
  - *Section 22.1.5, “Customizing the Test Run”*
- *Section 23.1, “Verifying Your Block Storage Backend”*
- *Section 23.2, “Verify the Object Storage (Swift) Operations”*

22.1.1 Prerequisites

The verification tests rely on you having an external network setup and a cloud image in your image (Glance) repository. Run the following playbook to configure your cloud:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts ardana-cloud-configure.yml
```

Note

In SUSE OpenStack Cloud 8, the EXT_NET_CIDR setting for the external network is now specified in the input model - see *Book* “Planning an Installation with Cloud Lifecycle Manager”, *Chapter 7 “Configuration Objects”, Section 7.16 “Configuration Data”, Section 7.16.2 “Neutron Configuration Data”, Section 7.16.2.2 “neutron-external-networks”*.

22.1.2 Tempest Integration Tests

Tempest is a set of integration tests for OpenStack API validation, scenarios, and other specific tests to be run against a live OpenStack cluster. In SUSE OpenStack Cloud 8, Tempest has been modeled as a service and this gives you the ability to locate Tempest anywhere in the cloud. It is recommended that you install Tempest on your Cloud Lifecycle Manager node - that is where it resides by default in a new installation.

A version of the upstream Tempest (http://docs.openstack.org/developer/tempest/) integration tests is pre-deployed on the Cloud Lifecycle Manager node. For details on what Tempest is testing, you can check the contents of this file on your Cloud Lifecycle Manager:

```
/opt/stack/tempest/run_filters/ci.txt
```

You can use these embedded tests to verify if the deployed cloud is functional.

For more information on running Tempest tests, see Tempest - The OpenStack Integration Test Suite (https://git.openstack.org/cgit/openstack/tempest/tree/README.rst).

Important

Running these tests requires access to the deployed cloud's identity admin credentials

Tempest creates and deletes test accounts and test resources for test purposes.

In certain cases Tempest might fail to clean-up some of test resources after a test is complete, for example in case of failed tests.

22.1.3 Running the Tests

To run the default set of Tempest tests:

1. Log in to the Cloud Lifecycle Manager.
2. Ensure you can access your cloud:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts cloud-client-setup.yml
   source /etc/environment
   ```

3. Run the tests:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ```

```
ansible-playbook -i hosts/verb_hosts tempest-run.yml
```

Optionally, you can *Section 22.1.5, “Customizing the Test Run”*.

22.1.4 Viewing Test Results

Tempest is deployed under `/opt/stack/tempest`. Test results are written in a log file in the following directory:

```
/opt/stack/tempest/logs
```

A detailed log file is written to:

```
/opt/stack/tempest/tempest.log
```

The results are also stored in the `testrepository` database.

To access the results after the run:

1. Log in to the Cloud Lifecycle Manager.
2. Change to the `tempest` directory and list the test results:

   ```
   cd /opt/stack/tempest
   ./venv/bin/testr last
   ```

Important

If you encounter an error saying "local variable 'run_subunit_content' referenced before assignment", you may need to log in as the `tempest` user to run this command. This is due to a known issue reported at https://bugs.launchpad.net/testrepository/+bug/1348970.

See Test Repository Users Manual (https://testrepository.readthedocs.org/en/latest/) for more details on how to manage the test result repository.

22.1.5 Customizing the Test Run

There are several ways available to customize which tests will be executed.

- *Section 22.1.6, “Run Tests for Specific Services and Exclude Specific Features”*
- *Section 22.1.7, “Run Tests Matching a Series of White and Blacklists”*

22.1.6 Run Tests for Specific Services and Exclude Specific Features

Tempest allows you to test specific services and features using the `tempest.conf` configuration file.

A working configuration file with inline documentation is deployed under `/opt/stack/tempest/etc/`.

To use this, follow these steps:

1. Log in to the Cloud Lifecycle Manager.
2. Edit the `/opt/stack/tempest/configs/tempest_region1.conf` file.
3. To test specific service, edit the `[service_available]` section and clear the comment character `#` and set a line to `true` to test that service or `false` to not test that service.

```
cinder = true
neutron = false
```

4. To test specific features, edit any of the `*_feature_enabled` sections to enable or disable tests on specific features of a service.

```
[volume-feature-enabled]
[compute-feature-enabled]
[identity-feature-enabled]
[image-feature-enabled]
[network-feature-enabled]
[object-storage-feature-enabled]
```

```
#Is the v2 identity API enabled (boolean value)
api_v2 = true
#Is the v3 identity API enabled (boolean value)
api_v3 = false
```

5. Then run tests normally

22.1.7 Run Tests Matching a Series of White and Blacklists

You can run tests against specific scenarios by editing or creating a run filter file.

Run filter files are deployed under `/opt/stack/tempest/run_filters`.

Use run filters to whitelist or blacklist specific tests or groups of tests:

- lines starting with # or empty are ignored
- lines starting with `+` are whitelisted
- lines starting with `-` are blacklisted
- lines not matching any of the above conditions are blacklisted

If whitelist is empty, all available tests are fed to blacklist. If blacklist is empty, all tests from whitelist are returned.

Whitelist is applied first. The blacklist is executed against the set of tests returned by the whitelist.

To run whitelist and blacklist tests:

1. Log in to the Cloud Lifecycle Manager.
2. Make sure you can access the cloud:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts cloud-client-setup.yml
   source /etc/environment
   ```

3. Run the tests:

   ```
   cd ~/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts tempest-run.yml  -e run_filter <run_filter_name>
   ```

Note that the run_filter_name is the name of the run_filter file except for the extension. For instance, to run using the filter from the file /opt/stack/tempest/run_filters/ci.txt, use the following:

```
ansible-playbook -i hosts/verb_hosts tempest-run.yml -e run_filter=ci
```

Documentation on the format of white and black-lists is available at:

```
/opt/stack/tempest/tests2skip.py
```

Example:

The following entries run API tests, exclude tests that are less relevant for deployment validation, such as negative, admin, cli and third-party (EC2) tests:

```
+tempest\.api\.*
*[Aa]dmin.*
*[Nn]egative.*
- tempest\.cli.*
- tempest\.thirdparty\.*
```

23 UI Verification

Once you have completed your cloud deployment, these are some of the common post-installation tasks you may need to perform to verify your cloud installation.

23.1 Verifying Your Block Storage Backend

The sections below will show you the steps to verify that your Block Storage backend was setup properly.

23.1.1 Create a Volume

Perform the following steps to create a volume using Horizon dashboard.

1. Log into the Horizon dashboard. For more information, see *Book "User Guide Overview", Chapter 3 "Cloud Admin Actions with the Dashboard"*.
2. Choose *Project* › *Compute* › *Volumes*.
3. On the *Volumes* tabs, click the *Create Volume* button to create a volume.
4. In the *Create Volume* options, enter the required details into the fields and then click the *Create Volume* button:
   a. Volume Name - This is the name you specify for your volume.
   b. Description (optional) - This is an optional description for the volume.
   c. Type - Select the volume type you have created for your volumes from the drop down.
   d. Size (GB) - Enter the size, in GB, you would like the volume to be.
   e. Availability Zone - You can either leave this at the default option of *Any Availability Zone* or select a specific zone from the drop-down box.

The dashboard will then show the volume you have just created.

23.1.2 Attach Volume to an Instance

Perform the following steps to attach a volume to an instance:

1. Log into the Horizon dashboard. For more information, see *Book "User Guide Overview", Chapter 3 "Cloud Admin Actions with the Dashboard"*.
2. Choose *Project* › *Compute* › *Instances*.
3. In the *Action* column, choose the *Edit Attachments* in the drop-down box next to the instance you want to attach the volume to.
4. In the *Attach To Instance* drop-down, select the volume that you want to attach.
5. Edit the *Device Name* if necessary.
6. Click *Attach Volume* to complete the action.
7. On the *Volumes* screen, verify that the volume you attached is displayed in the *Attached To* columns.

23.1.3 Detach Volume from Instance

Perform the following steps to detach the volume from instance:

1. Log into the Horizon dashboard. For more information, see *Book "User Guide Overview", Chapter 3 "Cloud Admin Actions with the Dashboard"*.
2. Choose *Project* › *Compute* › *Instances*.
3. Click the check box next to the name of the volume you want to detach.
4. In the *Action* column, choose the *Edit Attachments* in the drop-down box next to the instance you want to attach the volume to.
5. Click *Detach Attachment*. A confirmation dialog box appears.
6. Click *Detach Attachment* to confirm the detachment of the volume from the associated instance.

23.1.4 Delete Volume

Perform the following steps to delete a volume using Horizon dashboard:

1. Log into the Horizon dashboard. For more information, see *Book "User Guide Overview", Chapter 3 "Cloud Admin Actions with the Dashboard"*.
2. Choose *Project* › *Compute* › *Volumes*.
3. In the *Actions* column, click *Delete Volume* next to the volume you would like to delete.
4. To confirm and delete the volume, click *Delete Volume* again.
5. Verify that the volume was removed from the *Volumes* screen.

23.1.5 Verifying Your Object Storage (Swift)

The following procedure shows how to validate that all servers have been added to the Swift rings:

1. Run the swift-compare-model-rings.yml playbook as follows:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml
```

2. Search for output similar to the following. Specifically, look at the number of drives that are proposed to be added.

```
TASK: [swiftlm-ring-supervisor | validate-input-model | Print report] *********
ok: [ardana-cp1-c1-m1-mgmt] => {
    "var": {
        "report.stdout_lines": [
            "Rings:",
            "  ACCOUNT:",
            "    ring exists",
            "    no device changes",
            "    ring will be rebalanced",
            "  CONTAINER:",
            "    ring exists",
            "    no device changes",
            "    ring will be rebalanced",
            "  OBJECT-0:",
            "    ring exists",
            "    no device changes",
            "    ring will be rebalanced"
        ]
    }
}
```

3. If the text contains "no device changes" then the deploy was successful and no further action is needed.

4. If more drives need to be added, it indicates that the deploy failed on some nodes and that you restarted the deploy to include those nodes. However, the nodes are not in the Swift rings because enough time has not elapsed to allow the rings to be rebuilt. You have two options to continue:

   a. Repeat the deploy. There are two steps:

      i. Delete the ring builder files as described in *Book "Operations Guide", Chapter 15 "Troubleshooting Issues", Section 15.6 "Storage Troubleshooting", Section 15.6.2 "Swift Storage Troubleshooting", Section 15.6.2.8 "Restarting the Object Storage Deployment"*.

      ii. Repeat the installation process starting by running the `site.yml` playbook as described in *Section 13.7, "Deploying the Cloud"*.

   b. Rebalance the rings several times until all drives are incorporated in the rings. This process may take several hours to complete (because you need to wait one hour between each rebalance). The steps are as follows:

      i. Change the min-part-hours to 1 hour. See *Book "Operations Guide", Chapter 8 "Managing Object Storage", Section 8.5 "Managing Swift Rings", Section 8.5.7 "Changing min-part-hours in Swift"*.

      ii. Use the "First phase of ring rebalance" and "Final rebalance phase" as described in *Book "Operations Guide", Chapter 8 "Managing Object Storage", Section 8.5 "Managing Swift Rings", Section 8.5.5 "Applying Input Model Changes to Existing Rings"*. The "Weight change phase of ring rebalance" does not apply because you have not set the weight-step attribute at this stage.

      iii. Set the min-part-hours to the recommended 16 hours as described in *Book "Operations Guide", Chapter 8 "Managing Object Storage", Section 8.5 "Managing Swift Rings", Section 8.5.7 "Changing min-part-hours in Swift"*.

If you receive errors during the validation, see *Book "Operations Guide", Chapter 15 "Troubleshooting Issues", Section 15.6 "Storage Troubleshooting", Section 15.6.2 "Swift Storage Troubleshooting", Section 15.6.2.3 "Interpreting Swift Input Model Validation Errors"*.

23.2 Verify the Object Storage (Swift) Operations

For information about verifying the operations, see *Book "Operations Guide", Chapter 8 "Managing Object Storage", Section 8.1 "Running the Swift Dispersion Report"*.

23.3 Uploading an Image for Use

To create a Compute instance, you need to obtain an image that you can use. The Cloud Lifecycle Manager provides an Ansible playbook that will download a CirrOS Linux image, and then upload it as a public image to your image repository for use across your projects.

23.3.1 Running the Playbook

Use the following command to run this playbook:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts glance-cloud-configure.yml -e proxy=<PROXY>
```

The table below shows the optional switch that you can use as part of this playbook to specify environment-specific information:

Switch	Description
`-e proxy="<proxy_address:port>"`	Optional. If your environment requires a proxy for the internet, use this switch to specify the proxy information.

23.3.2 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 cloud:

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

23.3.3 Using the GlanceClient CLI to Create Images

You can use the GlanceClient on a machine accessible to your cloud or it's also installed automatically on your Cloud Lifecycle Manager.

The GlanceClient allows you to create, update, list, and delete images as well as manage your image member lists, which allows you to share access to images across multiple tenants. As with most of the OpenStack CLI tools, you can use the `glance help` command to get a full list of commands as well as their syntax.

If you would like to use the `--copy-from` option when creating an image, you will need to have your Administrator enable the http store in your environment using the instructions outlined at *Book "Operations Guide", Chapter 5 "Managing Compute", Section 5.6 "Configuring the Image Service", Section 5.6.2 "Allowing the Glance copy-from option in your environment"*.

23.4 Creating an External Network

You must have an external network set up to allow your Compute instances to reach the internet. There are multiple methods you can use to create this external network and we provide two of them here. The SUSE OpenStack Cloud installer provides an Ansible playbook that will create this network for use across your projects. We also show you how to create this network via the command line tool from your Cloud Lifecycle Manager.

23.4.1 Notes

If you have multiple regions in your cloud environment, it is important that when you set up your external networks in each region that you allocate unique IP ranges for each. If you have overlapping CIDRs then you risk having the same floating IP being allocated to two different virtual machines which will cause a conflict.

23.4.2 Using the Ansible Playbook

This playbook will query the Networking service for an existing external network, and then create a new one if you do not already have one. The resulting external network will have the name `ext-net` with a subnet matching the CIDR you specify in the command below.

If you need to specify more granularity, for example specifying an allocation pool for the subnet then you should utilize the *Section 23.4.3, "Using the NeutronClient CLI"*.

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts neutron-cloud-configure.yml -e EXT_NET_CIDR=<CIDR>
```

The table below shows the optional switch that you can use as part of this playbook to specify environment-specific information:

Switch	Description
`-e EXT_NET_CIDR=<CIDR>`	Optional. You can use this switch to specify the external network CIDR. If you choose not to use this switch, or use a wrong value, the VMs will not be accessible over the network. This CIDR will be from the `EXTERNAL VM` network. **Note** If this option is not defined the default value is "172.31.0.0/16"

23.4.3 Using the NeutronClient CLI

For more granularity you can utilize the Neutron command line tool to create your external network.

1. Log in to the Cloud Lifecycle Manager.
2. Source the Admin credentials:

   ```
   source ~/service.osrc
   ```

3. Create the external network and then the subnet using these commands below.
   Creating the network:

   ```
   neutron net-create --router:external <external-network-name>
   ```

   Creating the subnet:

   ```
   neutron subnet-create <external-network-name> <CIDR> --gateway <gateway> \
   ```

```
--allocation-pool start=<IP_start>,end=<IP_end> [--disable-dhcp]
```

Where:

Value	Description
external-network-name	This is the name given to your external network. This is a unique value that you will choose. The value `ext-net` is usually used.
CIDR	You can use this switch to specify the external network CIDR. If you choose not to use this switch, or use a wrong value, the VMs will not be accessible over the network. This CIDR will be from the EXTERNAL VM network.
--gateway	Optional switch to specify the gateway IP for your subnet. If this is not included then it will choose the first available IP.
--allocation-pool start end	Optional switch to specify a start and end IP address to use as the allocation pool for this subnet.
--disable-dhcp	Optional switch if you want to disable DHCP on this subnet. If this is not specified then DHCP will be enabled.

23.4.4 Next Steps

Once the external network is created, users can create a Private Network to complete their networking setup. For instructions, see *Book "User Guide Overview", Chapter 8 "Creating a Private Network"*.

24 Installing OpenStack Clients

If you have a standalone deployer, the OpenStack CLI and other clients will not be installed automatically on that node. If you require access to these clients, you will need to follow the procedure below to add the appropriate software.

1. [OPTIONAL] Connect to your standalone deployer and try to use the OpenStack CLI:

```
source ~/keystone.osrc
openstack project list

-bash: openstack: command not found
```

2. Edit the configuration file containing details of your Control Plane, typically `~/openstack/my_cloud/definition/data/control_plane`.

3. Locate the stanza for the cluster where you want to install the client(s). For a standalone deployer, this will look like the following extract:

```
      clusters:
        - name: cluster0
          cluster-prefix: c0
          server-role: LIFECYCLE-MANAGER-ROLE
          member-count: 1
          allocation-policy: strict
          service-components:
            - ntp-server
            - lifecycle-manager
```

4. Choose the client(s) you wish to install from the following list of available clients:

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

5. Add the client(s) to the list of `service-components` - in this example, we add the `openstack-client` to the standalone deployer:

   ```
         clusters:
           - name: cluster0
             cluster-prefix: c0
             server-role: LIFECYCLE-MANAGER-ROLE
             member-count: 1
             allocation-policy: strict
             service-components:
               - ntp-server
               - lifecycle-manager
               - openstack-client
               - ceilometer-client
               - cinder-client
               - designate-client
               - glance-client
               - heat-client
               - ironic-client
               - keystone-client
               - neutron-client
               - nova-client
               - swift-client
               - monasca-client
               - barbican-client
   ```

6. Commit the configuration changes:

   ```
   cd ~/openstack/ardana/ansible
   git add -A
   git commit -m "Add explicit client service deployment"
   ```

7. Run the configuration processor, followed by the `ready-deployment` playbook:

   ```
   cd ~/openstack/ardana/ansible
   ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" \
     -e rekey=""
   ansible-playbook -i hosts/localhost ready-deployment.yml
   ```

8. Add the software for the clients using the following command:

   ```
   cd /home/stack/scratch/ansible/next/ardana/ansible
   ansible-playbook -i hosts/verb_hosts clients-upgrade.yml
   ```

9. Check that the software has been installed correctly. In this instance, connect to your standalone deployer and try to use the OpenStack CLI:

```
source ~/keystone.osrc
openstack project list
```

You should now see a list of projects returned:

```
stack@ardana-cp1-c0-m1-mgmt:~$ openstack project list

+----------------------------------+------------------+
| ID                               | Name             |
+----------------------------------+------------------+
| 076b6e879f324183bbd28b46a7ee7826 | kronos           |
| 0b81c3a9e59c47cab0e208ea1bb7f827 | backup           |
| 143891c2a6094e2988358afc99043643 | octavia          |
| 1d3972a674434f3c95a1d5ed19e0008f | glance-swift     |
| 2e372dc57cac4915bf06bbee059fc547 | glance-check     |
| 383abda56aa2482b95fb9da0b9dd91f4 | monitor          |
| 606dd3b1fa6146668d468713413fb9a6 | swift-monitor    |
| 87db9d1b30044ea199f0293f63d84652 | admin            |
| 9fbb7494956a483ca731748126f50919 | demo             |
| a59d0c682474434a9ddc240ddfe71871 | services         |
| a69398f0f66a41b2872bcf45d55311a7 | swift-dispersion |
| f5ec48d0328d400992c1c5fb44ec238f | cinderinternal   |
+----------------------------------+------------------+
```

25 Configuring Transport Layer Security (TLS)

TLS is enabled by default during the installation of SUSE OpenStack Cloud 8 and additional configuration options are available to secure your environment, as described below.

In SUSE OpenStack Cloud 8, you can provide your own certificate authority and certificates for internal and public virtual IP addresses (VIPs), and you should do so for any production cloud. The certificates automatically generated by SUSE OpenStack Cloud are useful for testing and setup, but you should always install your own for production use. Certificate installation is discussed below.

Please read the following if you're using the default `cert-name: my-public-cert` in your model.

The bundled test certificate for public endpoints, located at `~/openstack/my_cloud/config/tls/certs/my-public-cert`, is now expired but was left in the product in case you changed the content with your valid certificate. Please verify if the certificate is expired and generate your own, as described in *Section 25.4, "Generate a self-signed CA"*.

You can verify the expiry date by running this command:

```
ardana > openssl x509 -in ~/openstack/my_cloud/config/tls/certs/my-public-cert \
-noout -enddate
notAfter=Oct  8 09:01:58 2016 GMT
```

Before you begin, the following list of terms will be helpful when generating and installing certificates.

SUSE OpenStack Cloud-generated public CA

A SUSE OpenStack Cloud-generated public CA (`openstack_frontend_cacert.crt`) is available for you to use in `/etc/pki/trust/anchors/ca-certificates`.

Fully qualified domain name (FQDN) of the public VIP

The registered domain name. A FQDN is not mandatory. It is perfectly valid to have no FQDN and use IP addresses instead. Note that you can use FQDNs on public endpoints, and you may change them whenever the need arises.

Certificate authority (CA) certificate

Your certificates must be signed by a CA, such as your internal IT department or a public certificate authority. For this example we will use a self-signed certificate.

Server certificate

It is easy to confuse server certificates and CA certificates. Server certificates reside on the server and CA certificates reside on the client. A server certificate affirms that the server that sent it serves a set of IP addresses, domain names, and set of services. A CA certificate is used by the client to authenticate this claim.

SAN (subject-alt-name)

The set of IP addresses and domain names in a server certificate request: A template for a server certificate.

Certificate signing request (CSR)

A blob of data generated from a certificate request and sent to a CA, which would then sign it, produce a server certificate, and send it back.

External VIP

External virtual IP address

Internal VIP

Internal virtual IP address

The major difference between an external VIP certificate and an internal VIP certificate is that the internal VIP has approximately 40 domain names in the SAN. This is because each service has a different domain name in SUSE OpenStack Cloud 8. So it is unlikely that you can create an internal server certificate before running the configuration processor. But after a configuration processor run, a certificate request would be created for each of your cert-names.

25.1 Configuring TLS in the input model

For this example certificate configuration, let's assume there's no FQDN for the external VIP and that you're going to use the default IP address provided by SUSE OpenStack Cloud 8. Let's also assume that for the internal VIP you will use the defaults as well. If you were to call your certificate authority "example-CA," the CA certificate would then be called "example-CA.crt" and the key would be called "example-CA.key." In the following examples, the external VIP certificate will be named "example-public-cert" and the internal VIP certificate will be named "example-internal-cert."

Note

Cautions:

Any time you make a cert change when using your own CA:

- You should use a distinct name from those already existing in `config/tls/cacerts`. This also means that you should not *reuse* your CA names (and use unique and distinguishable names such as MyCompanyXYZ_PrivateRootCA.crt). A new name is what indicates that a file is new or changed, so reusing a name means that the file is not considered changed even its contents have changed.
- You should not remove any existing CA files from `config/tls/cacerts`.
- If you want to remove an existing CA you must
  1. First remove the file.
  2. Then run:

     ```
     ardana > ansible -i hosts/verb_hosts FND-STN -a 'sudo keytool -delete -alias \
     debian:<filename to remove> \
     -keystore /usr/lib/jvm/java-7-openjdk-amd64/jre/lib/security/cacerts \
     -storepass changeit'
     ```

Important

Be sure to install your own certificate for all production clouds after installing and testing your cloud. If you ever want to test or troubleshoot later, you will be able to revert to the sample certificate to get back to a stable state for testing.

Note

Unless this is a new deployment, do not update both the certificate and the CA together. Add the CA first and then run a site deploy. Then update the certificate and run tls-reconfigure, FND-CLU-stop, FND-CLU-start and then ardana-reconfigure. If a playbook has failed, rerun it with -vv to get detailed error information. The configure, HAproxy restart, and reconfigure steps are included below. If this is a new deployment and you are adding your own certs/CA before running site.yml this caveat does not apply.

You can add your own certificate by following the instructions below. All changes must go into the file `~/openstack/my_cloud/definition/data/network_groups.yml`.

Below are the entries for TLS for the internal and admin load balancers:

```
- provider: ip-cluster
        name: lb
        tls-components:
        - default
        components:
        # These services do not currently support TLS so they are not listed
        # under tls-components
        - nova-metadata
        roles:
        - internal
        - admin
        cert-file: openstack-internal-cert
        # The openstack-internal-cert is a reserved name and
        # this certificate will be autogenerated. You
        # can bring in your own certificate with a different name

        # cert-file: customer-provided-internal-cert
        # replace this with name of file in "config/tls/certs/"
```

The configuration processor will also create a request template for each named certificate under `info/cert_reqs/` This will be of the form:

```
info/cert_reqs/customer-provided-internal-cert
```

These request templates contain the subject `Alt-names` that the certificates need. You can add to this template before generating your certificate signing request .

You would then send the CSR to your CA to be signed, and once you receive the certificate, place it in `config/tls/certs`.

When you bring in your own certificate, you may want to bring in the trust chains (or CA certificate) for this certificate. This is usually not required if the CA is a public signer that is typically bundled with the operating system. However, we suggest you include it anyway by copying the file into the directory `config/cacerts/`.

25.2 User-provided certificates and trust chains

SUSE OpenStack Cloud generates its own internal certificates but is designed to allow you to bring in your own certificates for the VIPs. Here is the general process.

1. You must have a server certificate and a CA certificate to go with it (unless the signer is a public CA and it's already bundled with most distributions).
2. You must decide the names of the server certificates and configure the `network_groups.yml` file in the input model such that each load balancer provider has at least one cert-name associated with it.
3. Run the configuration processor. Note that you may or may not have the certificate file at this point. The configuration processor would create certificate request file artifacts under `info/cert_reqs/` for each of the cert-name(s) in the `network_groups.yml` file. While there's no special reason to use the request file created for an external endpoint VIP certificate, it is important to use the request files created for internal certificates since the canonical names for the internal VIP can be many and service specific and each of these need to be in the Subject Alt Names attribute of the certificate.
4. Create a certificate signing request for this request file and send it to your internal CA or a public CA to get it certified and issued with a certificate. You will now have a server certificate and possibly a trust chain or CA certificate.
5. Next, upload it to the Cloud Lifecycle Manager. Server certificates should be added to `config/tls/certs` and CA certificates should be added to `config/tls/cacerts`. The file extension should be CRT file for the CA certificate to be processed by SUSE OpenStack Cloud. Detailed steps are next.

25.3 Edit the input model to include your certificate files

Edit the load balancer configuration in `~/openstack/my_cloud/definition/data/network_groups.yml`:

```
load-balancers:
 - provider: ip-cluster
 name: lb
 tls-components:
 - default
 components:
 - nova-metadata
 roles:
 - internal
 - admin
 cert-file: example-internal-cert #<<<---- Certificate name for the internal VIP
```

```
- provider: ip-cluster
 name: extlb
 external-name: myardana.test #<<<--- Use just IP for the external VIP in this example
 tls-components:
 - default
 roles:
 - public
 cert-file: example-public-cert #<<<---- Certificate name for the external VIP
```

Commit your changes to the local git repository and run the configuration processor:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "changed VIP certificates"
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
```

Verify that certificate requests have been generated by the configuration processor for every certificate file configured in the `networks_groups.yml` file. In this example, there are two files, as shown from the list command:

```
ardana > ls ~/openstack/my_cloud/info/cert_reqs
example-internal-cert
example-public-cert
```

25.4 Generate a self-signed CA

Note

In a production setting you will not perform this step. You will use your company's CA or a valid public CA.

This section demonstrates to how you can create your own self-signed CA and then use this CA to sign server certificates. This CA can be your organization's IT internal CA that is self-signed and whose CA certificates are deployed on your organization's machines. This way the server certificate becomes legitimate.

Note

Please use a unique CN for your example Certificate Authority and do not install multiple CA certificates with the same CN into your cloud.

Copy the commands below to the command line and execute. This will cause the two files, `example-CA.key` and `example-CA.crt` to be created:

```
ardana > export EXAMPLE_CA_KEY_FILE='example-CA.key'
ardana > export EXAMPLE_CA_CERT_FILE='example-CA.crt'
ardana > openssl req -x509 -batch -newkey rsa:2048 -nodes -out "${EXAMPLE_CA_CERT_FILE}" \
-keyout "${EXAMPLE_CA_KEY_FILE}" \
-subj "/C=UK/O=hp/CN=YourOwnUniqueCertAuthorityName" \
-days 365
```

You can tweak the subj and days settings above to meet your needs, or to test. For instance, if you want to test what happens when a CA expires, you can set 'days' to a very low value. Grab the configuration processor-generated request file from `info/cert_reqs/`:

```
ardana > cat ~/openstack/my_cloud/info/cert_reqs/example-internal-cert
```

Now, copy this file to your working directory and append a `.req` extension to it.

```
ardana > cp ~/openstack/my_cloud/info/cert_reqs/example-internal-cert \
example-internal-cert.req
```

EXAMPLE 25.1: CERTIFICATE REQUEST FILE

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no

[ req_distinguished_name ]
CN = "openstack-vip"

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = "deployerincloud-ccp-c0-m1-mgmt"
DNS.2 = "deployerincloud-ccp-vip-CEI-API-mgmt"
DNS.3 = "deployerincloud-ccp-vip-CND-API-mgmt"
DNS.4 = "deployerincloud-ccp-vip-DES-API-mgmt"
DNS.5 = "deployerincloud-ccp-vip-FND-MDB-mgmt"
DNS.6 = "deployerincloud-ccp-vip-FND-RMQ-mgmt"
DNS.7 = "deployerincloud-ccp-vip-FND-VDB-mgmt"
DNS.8 = "deployerincloud-ccp-vip-FRE-API-mgmt"
DNS.9 = "deployerincloud-ccp-vip-GLA-API-mgmt"
DNS.10 = "deployerincloud-ccp-vip-GLA-REG-mgmt"
DNS.11 = "deployerincloud-ccp-vip-HEA-ACF-mgmt"
DNS.12 = "deployerincloud-ccp-vip-HEA-ACW-mgmt"
DNS.13 = "deployerincloud-ccp-vip-HEA-API-mgmt"
DNS.14 = "deployerincloud-ccp-vip-HUX-SVC-mgmt"
DNS.15 = "deployerincloud-ccp-vip-HZN-WEB-mgmt"
DNS.16 = "deployerincloud-ccp-vip-KEY-API-mgmt"
DNS.17 = "deployerincloud-ccp-vip-KEYMGR-API-mgmt"
DNS.18 = "deployerincloud-ccp-vip-LOG-API-mgmt"
DNS.19 = "deployerincloud-ccp-vip-LOG-SVR-mgmt"
DNS.20 = "deployerincloud-ccp-vip-MON-API-mgmt"
DNS.21 = "deployerincloud-ccp-vip-NEU-SVR-mgmt"
DNS.22 = "deployerincloud-ccp-vip-NOV-API-mgmt"
DNS.23 = "deployerincloud-ccp-vip-NOV-MTD-mgmt"
DNS.24 = "deployerincloud-ccp-vip-OCT-API-mgmt"
DNS.25 = "deployerincloud-ccp-vip-OPS-WEB-mgmt"
DNS.26 = "deployerincloud-ccp-vip-SHP-API-mgmt"
DNS.27 = "deployerincloud-ccp-vip-SWF-PRX-mgmt"
DNS.28 = "deployerincloud-ccp-vip-admin-CEI-API-mgmt"
DNS.29 = "deployerincloud-ccp-vip-admin-CND-API-mgmt"
DNS.30 = "deployerincloud-ccp-vip-admin-DES-API-mgmt"
DNS.31 = "deployerincloud-ccp-vip-admin-FND-MDB-mgmt"
DNS.32 = "deployerincloud-ccp-vip-admin-FRE-API-mgmt"
DNS.33 = "deployerincloud-ccp-vip-admin-GLA-API-mgmt"
DNS.34 = "deployerincloud-ccp-vip-admin-HEA-ACF-mgmt"
DNS.35 = "deployerincloud-ccp-vip-admin-HEA-ACW-mgmt"
DNS.36 = "deployerincloud-ccp-vip-admin-HEA-API-mgmt"
DNS.37 = "deployerincloud-ccp-vip-admin-HUX-SVC-mgmt"
DNS.38 = "deployerincloud-ccp-vip-admin-HZN-WEB-mgmt"
DNS.39 = "deployerincloud-ccp-vip-admin-KEY-API-mgmt"
DNS.40 = "deployerincloud-ccp-vip-admin-KEYMGR-API-mgmt"
DNS.41 = "deployerincloud-ccp-vip-admin-MON-API-mgmt"
DNS.42 = "deployerincloud-ccp-vip-admin-NEU-SVR-mgmt"
DNS.43 = "deployerincloud-ccp-vip-admin-NOV-API-mgmt"
DNS.44 = "deployerincloud-ccp-vip-admin-OPS-WEB-mgmt"
DNS.45 = "deployerincloud-ccp-vip-admin-SHP-API-mgmt"
DNS.46 = "deployerincloud-ccp-vip-admin-SWF-PRX-mgmt"
DNS.47 = "192.168.245.5"
IP.1 = "192.168.245.5"

=============end of certificate request file.
```

Note

In the case of a public VIP certificate, please add all the FQDNs you want it to support Currently SUSE OpenStack Cloud does not add the hostname for the external-name specified in `network_groups.yml` to the certificate request file . However, you can add it to the certificate request file manually. Here we assume that `myardana.test` is your external-name. In that case you would add this line (to the certificate request file that is shown above in *Example 25.1, "Certificate request file"*):

```
DNS.48 = "myardana.test"
```

Note

Any attempt to use IP addresses rather than FQDNs in certificates must use subject alternate name entries that list both the IP address (needed for Google) and DNS with an IP (needed for a Python bug workaround). Failure to create the certificates in this manner will cause future installations of Go-based tools (such as Cloud Foundry, Stackato and other PaaS components) to fail.

25.5 Generate a certificate signing request

Now that you have a CA and a certificate request file, it's time to generate a CSR.

```
ardana > export EXAMPLE_SERVER_KEY_FILE='example-internal-cert.key'
ardana > export EXAMPLE_SERVER_CSR_FILE='example-internal-cert.csr'
ardana > export EXAMPLE_SERVER_REQ_FILE=example-internal-cert.req
ardana > openssl req -newkey rsa:2048 -nodes -keyout "$EXAMPLE_SERVER_KEY_FILE" \
-out "$EXAMPLE_SERVER_CSR_FILE" -extensions v3_req -config "$EXAMPLE_SERVER_REQ_FILE"
```

Note that in production you would usually send the generated `example-internal-cert.csr` file to your IT department. But in this example you are your own CA, so sign and generate a server certificate.

25.6 Generate a server certificate

Note

In a production setting you will not perform this step. You will send the CSR created in the previous section to your company CA or a to a valid public CA and have them sign and send you back the certificate.

This section demonstrates how you would use your own self-signed CA that your created earlier to sign and generate a server certificate. A server certificate is essentially a signed public key, the signer being a CA and trusted by a client. When you install this the signing CA's certificate (called CA certificate or trust chain) on the client machine, you are telling the client to trust this CA, and thereby implicitly trusting any server certificates that are signed by this CA, thus creating a trust anchor.

CA configuration file

When the CA signs the certificate, it uses a configuration file that tells it to verify the CSR. Note that in a production scenario the CA takes care of this for you.

Create a file called `openssl.cnf` and add the following contents to it.

```
# Copyright 2010 United States Government as represented by the
# Administrator of the National Aeronautics and Space Administration.
# All Rights Reserved.
#...

# OpenSSL configuration file.
#

# Establish working directory.

dir = .

[ ca ]
default_ca = CA_default

[ CA_default ]
serial = $dir/serial
database = $dir/index.txt
new_certs_dir = $dir/
certificate = $dir/cacert.pem
private_key = $dir/cakey.pem
unique_subject = no
default_crl_days = 365
default_days = 365
default_md = md5
preserve = no
email_in_dn = no
nameopt = default_ca
certopt = default_ca
policy = policy_match
copy_extensions = copy

```

```
[ policy_match ]
countryName = optional
stateOrProvinceName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ req ]
default_bits = 1024 # Size of keys
default_keyfile = key.pem # name of generated keys
default_md = md5 # message digest algorithm
string_mask = nombstr # permitted characters
distinguished_name = req_distinguished_name
req_extensions = v3_req
x509_extensions = v3_ca

[ req_distinguished_name ]
# Variable name Prompt string
#---------------------- ----------------------------------
0.organizationName = Organization Name (company)
organizationalUnitName = Organizational Unit Name (department, division)
emailAddress = Email Address
emailAddress_max = 40
localityName = Locality Name (city, district)
stateOrProvinceName = State or Province Name (full name)
countryName = Country Name (2 letter code)
countryName_min = 2
countryName_max = 2
commonName = Common Name (hostname, IP, or your name)
commonName_max = 64

# Default values for the above, for consistency and less typing.
# Variable name Value
#------------------------------ ------------------------------
0.organizationName_default = Exampleco PLC
localityName_default = Anytown
stateOrProvinceName_default = Anycounty
countryName_default = UK
commonName_default = my-CA

[ v3_ca ]
basicConstraints = CA:TRUE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
subjectAltName = @alt_names
```

```
[ v3_req ]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash

[ alt_names ]

######### end of openssl.cnf #########
```

Sign and create a server certificate

Now you can sign the server certificate with your CA. Copy the commands below to the command line and execute. This will cause the one file, example-internal-cert.crt, to be created:

```
ardana > export EXAMPLE_SERVER_CERT_FILE='example-internal-cert.crt'
ardana > export EXAMPLE_SERVER_CSR_FILE='example-internal-cert.csr'
ardana > export EXAMPLE_CA_KEY_FILE='example-CA.key'
ardana > export EXAMPLE_CA_CERT_FILE='example-CA.crt'
ardana > touch index.txt
ardana > openssl rand -hex -out serial 6
ardana > openssl ca -batch -notext -md sha256 -in "$EXAMPLE_SERVER_CSR_FILE" \
-cert "$EXAMPLE_CA_CERT_FILE" \
-keyfile "$EXAMPLE_CA_KEY_FILE" \
-out "$EXAMPLE_SERVER_CERT_FILE" \
-config openssl.cnf -extensions v3_req
```

Finally, concatenate both the server key and certificate in preparation for uploading to the Cloud Lifecycle Manager.

```
ardana > cat example-internal-cert.key example-internal-cert.crt > example-internal-cert
```

Note that you have only created the internal-cert in this example. Repeat the above sequence for example-public-cert. Make sure you use the appropriate certificate request generated by the configuration processor.

25.7 Upload to the Cloud Lifecycle Manager

The following two files created from the example run above will need to be uploaded to the Cloud Lifecycle Manager and copied into `config/tls`.

- example-internal-cert
- example-CA.crt

Once on the Cloud Lifecycle Manager, execute the following two copy commands to copy to their respective directories. Note if you had created an external cert, you can copy that in a similar manner, specifying its name using the copy command as well.

```
ardana > cp example-internal-cert ~/openstack/my_cloud/config/tls/certs/
ardana > cp example-CA.crt ~/openstack/my_cloud/config/tls/cacerts/
```

Continue with the deployment

Next, log into the Cloud Lifecycle Manager node, and save and commit the changes to the local git repository:

```
ardana > cd ~/openstack/ardana/ansible
ardana > git add -A
ardana > git commit -m "updated certificate and CA"
```

Next, rerun the `config-processor-run` playbook, and run `ready-deployment.yml`:

```
ardana > cd ~/openstack/ardana/ansible
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

If you receive any prompts, enter the required information.

Note

For automated installation (for example CI) you can specify the required passwords on the Ansible command line. For example, the command below will disable encryption by the configuration processor:

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e
 rekey=""
```

Run this series of runbooks to complete the deployment:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts tls-reconfigure.yml
ardana > ansible-playbook -i hosts/verb_hosts FND-CLU-stop.yml
ardana > ansible-playbook -i hosts/verb_hosts FND-CLU-start.yml
ardana > ansible-playbook -i hosts/verb_hosts monasca-stop.yml
ardana > ansible-playbook -i hosts/verb_hosts monasca-start.yml
ardana > ansible-playbook -i hosts/verb_hosts ardana-reconfigure.yml
```

25.8 Configuring the cipher suite

By default, the cipher suite is set to: `HIGH:!aNULL:!eNULL:!DES:!3DES`. This setting is recommended in the OpenStack documentation site (http://docs.openstack.org/security-guide/secure-communication/introduction-to-ssl-and-tls.html). You may override this. To do so, open `config/haproxy/defaults.yml` and edit it. The parameters can be found under the `haproxy_globals` list.

```
- "ssl-default-bind-ciphers HIGH:!aNULL:!eNULL:!DES:!3DES"
- "ssl-default-server-ciphers HIGH:!aNULL:!eNULL:!DES:!3DES"
```

Make the changes as needed. It's best to keep the two options identical.

25.9 Testing

You can easily determine if an endpoint is behind TLS. To do so, run the following command, which probes a Keystone identity service endpoint that's behind TLS:

```
ardana > echo | openssl s_client -connect 192.168.245.5:5000 | openssl x509 -fingerprint -noout
depth=0 CN = openstack-vip
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 CN = openstack-vip
verify error:num=27:certificate not trusted
verify return:1
depth=0 CN = openstack-vip
verify error:num=21:unable to verify the first certificate
verify return:1
DONE
SHA1 Fingerprint=C6:46:1E:59:C6:11:BF:72:5E:DD:FC:FF:B0:66:A7:A2:CC:32:1C:B8
```

The next command probes a MariaDB endpoint that is not behind TLS:

```
ardana > echo | openssl s_client -connect 192.168.245.5:3306 | openssl x509 -fingerprint -noout
140448358213264:error:140770FC:SSL routines:SSL23_GET_SERVER_HELLO:unknown protocol:s23_clnt.c:795:
unable to load certificate
140454148159120:error:0906D06C:PEM routines:PEM_read_bio:no start line:pem_lib.c:703
:Expecting: TRUSTED CERTIFICATE
```

25.10 Verifying that the trust chain is correctly deployed

You can determine if the trust chain is correctly deployed by running the following commands:

```
ardana > echo | openssl s_client -connect 192.168.245.9:5000 2>/dev/null | grep code
Verify return code: 21 (unable to verify the first certificate)
echo | openssl s_client -connect 192.168.245.9:5000 \
-CAfile /etc/pki/trust/anchors/ca-certificates/openstack_frontend_cacert.crt 2>/dev/null | grep
 code
Verify return code: 0 (ok)
```

Here, the first command produces error 21, which is then fixed by providing the CA certificate file. This verifies that the CA certificate matches the server certificate.

25.11 Turning TLS on or off

You should leave TLS enabled in production. However, if you need to disable it for any reason, you must change "tls-components" to "components" in `network_groups.yml` (as shown earlier) and comment out the cert-file. Additionally, if you have a `network_groups.yml` file from a previous installation, you won't have TLS enabled unless you change "components" to "tls-components" in that file. By default, Horizon is configured with TLS in the input model. Note that you should not disable TLS in the input model for Horizon as that is a public endpoint and is required. Additionally, you should keep all services behind TLS, but using the input model file `network_groups.yml` you may turn TLS off for a service for troubleshooting or debugging. TLS should always be enabled for production environments.

If you are using an example input model on a clean install, all supported TLS services will be enabled before deployment of your cloud. If you want to change this setting later, for example, when upgrading, you can change the input model and reconfigure the system. The process is as follows:

Edit the input model `network_groups.yml` file appropriately, as described above. Then, commit the changes to the git repository:

```
ardana > cd ~/openstack/ardana/ansible/
ardana > git add -A
ardana > git commit -m "TLS change"
```

Change directories again and run the configuration processor and ready deployment playbooks:

```
ardana > ansible-playbook -i hosts/localhost config-processor-run.yml
ardana > ansible-playbook -i hosts/localhost ready-deployment.yml
```

Change directories again and run the reconfigure playbook:

```
ardana > cd ~/scratch/ansible/next/ardana/ansible
ardana > ansible-playbook -i hosts/verb_hosts ardana-reconfigure.yml
```

26 Configuring Availability Zones

The Cloud Lifecycle Manager only creates a default availability zone during installation. If your system has multiple failure/availability zones defined in your input model, these zones will not get created automatically.

Once the installation has finished, you can run the `nova-cloud-configure.yml` playbook to configure availability zones and assign compute nodes to those zones based on the configuration specified in the model.

You can run the playbook `nova-cloud-configure.yml` any time you make changes to the configuration of availability zones in your input model. Alternatively, you can use Horizon or the command line to perform the configuration.

For more details, see the OpenStack Availability Zone documentation at https://docs.openstack.org/nova/pike/user/aggregates.html.

27 Configuring Load Balancer as a Service

The SUSE OpenStack Cloud Neutron LBaaS service supports several load balancing providers. By default, both Octavia and the namespace HAProxy driver are configured to be used. We describe this in more detail here.

The SUSE OpenStack Cloud Neutron LBaaS service supports several load balancing providers. By default, both Octavia and the namespace HAProxy driver are configured to be used. A user can specify which provider to use with the `--provider` flag upon load balancer creation.

Example:

```
tux > neutron lbaas-loadbalancer-create --name NAME --provider \
  [octavia|haproxy] SUBNET
```

If you do not specify the `--provider` option it will default to Octavia. The Octavia driver provides more functionality than the HAProxy namespace driver which is deprecated. The HAProxy namespace driver will be retired in a future version of SUSE OpenStack Cloud.

There are additional drivers for third-party hardware load balancers. Please refer to the vendor directly. The `neutron service-provider-list` command displays not only the currently installed load balancer drivers but also other installed services such as VPN. You can see a list of available services as follows:

```
tux > neutron service-provider-list
+----------------+----------+---------+
| service_type   | name     | default |
+----------------+----------+---------+
| LOADBALANCERV2 | octavia  | True    |
| VPN            | openswan | True    |
| LOADBALANCERV2 | haproxy  | False   |
| LOADBALANCERV2 | octavia  | True    |
| VPN            | openswan | True    |
| LOADBALANCERV2 | haproxy  | False   |
+----------------+----------+---------+
```

Note

The Octavia load balancer provider is listed as the default.

27.1 Prerequisites

You will need to create an external network and create an image to test LBaaS functionality. If you have already created an external network and registered an image, this step can be skipped.

Creating an external network: *Section 23.4, "Creating an External Network"*.

Creating and uploading a Glance image: *Book "User Guide Overview", Chapter 10 "Creating and Uploading a Glance Image"*.

27.2 Octavia Load Balancing Provider

The Octavia Load balancing provider bundled with SUSE OpenStack Cloud 8 is an operator grade load balancer for OpenStack. It is based on the OpenStack Pike version of Octavia. It differs from the namespace driver by starting a new Nova virtual machine to house the HAProxy load balancer software, called an *amphora*, that provides the load balancer function. A virtual machine for each load balancer requested provides a better separation of load balancers between tenants and makes it easier to grow load balancing capacity alongside compute node growth. Additionally, if the virtual machine fails for any reason Octavia will replace it with a replacement VM from a pool of spare VMs, assuming that the feature is configured.

Note

The Health Monitor will not create or replace failed amphorae. If the pool of spare VMs is exhausted there will be no additional virtual machines to handle load balancing requests.

Octavia uses two-way SSL encryption to communicate with the amphora. There are demo Certificate Authority (CA) certificates included with SUSE OpenStack Cloud 8 in `~/scratch/ansible/next/ardana/ansible/roles/octavia-common/files` on the Cloud Lifecycle Manager. For additional security in production deployments, all certificate authorities should be replaced with ones you generated yourself by running the following commands:

```
ardana > openssl genrsa -passout pass:foobar -des3 -out cakey.pem 2048
ardana > openssl req -x509 -passin pass:foobar -new -nodes -key cakey.pem -out ca_01.pem
ardana > openssl genrsa -passout pass:foobar -des3 -out servercakey.pem 2048
ardana > openssl req -x509 -passin pass:foobar -new -nodes -key cakey.pem -out serverca_01.pem
```

For more details refer to the openssl man page (https://www.openssl.org/docs/manmaster/apps/openssl.html).

Note

If you change the certificate authority and have amphora running with an old CA you won't be able to control the amphora. The amphora's will need to be failed over so they can utilize the new certificate. If you change the CA password for the server certificate you need to change that in the Octavia configuration files as well. For more information, see *Book "Operations Guide", Chapter 9 "Managing Networking", Section 9.3 "Networking Service Overview", Section 9.3.9 "Load Balancer: Octavia Driver Administration", Section 9.3.9.2 "Tuning Octavia Installation"*.

27.3 Setup of prerequisites

Octavia Network and Management Network Ports

The Octavia management network and Management network must have access to each other. If you have a configured firewall between the Octavia management network and Management network, you must open up the following ports to allow network traffic between the networks.

- From Management network to Octavia network
  - TCP 9443 (amphora API)
- From Octavia network to Management network
  - TCP 9876 (Octavia API)
  - UDP 5555 (Octavia Health Manager)

Installing the Amphora Image

Octavia uses Nova VMs for its load balancing function and SUSE provides images used to boot those VMs called `octavia-amphora`.

Warning

Without these images the Octavia load balancer will not work.

Register the image. The OpenStack load balancing service (Octavia) does not automatically register the Amphora guest image.

1. The full path and name for the Amphora image is `/srv/www/suse-12.3/x86_64/repos/Cloud/suse/noarch/openstack-octavia-amphora-image.rpm`
   Switch to the ansible directory and register the image by giving the full path and name for the Amphora image as an argument to service_package:

   ```
   ardana > cd ~/scratch/ansible/next/ardana/ansible/
   ardana > ansible-playbook -i hosts/verb_hosts service-guest-image.yml \
   -e service_package=\
   /srv/www/suse-12.3/x86_64/repos/Cloud/suse/noarch/openstack-octavia-amphora-image.rpm
   ```

2. Source the service user (this can be done on a different computer)

   ```
   tux > source service.osrc
   ```

3. Verify that the image was registered (this can be done on a computer with access to the Glance CLI client)

   ```
   tux > openstack image-list
   +--------------------------------------+------------------------+---------
   | ID                                   | Name                   | Status |
   +--------------------------------------+------------------------+--------+
   ...
   | 1d4dd309-8670-46b6-801d-3d6af849b6a9 | octavia-amphora-x86_64 | active |
   ...
   ```

Important

In the example above, the status of the `octavia-amphora-x86_64` image is *active* which means the image was successfully registered. If a status of the images is *queued*, you need to run the image registration again.

If you run the registration by accident, the system will only upload a new image if the underlying image has been changed.

Please be aware that if you have already created load balancers they will not receive the new image. Only load balancers created after the image has been successfully installed will use the new image. If existing load balancers need to be switched to the new image please follow the instructions in *Book "Operations Guide", Chapter 9 "Managing Networking", Section 9.3 "Networking Service Overview", Section 9.3.9 "Load Balancer: Octavia Driver Administration", Section 9.3.9.2 "Tuning Octavia Installation"*.

Setup network, subnet, router, security and IP's

If you have already created a network, subnet, router, security settings and IPs you can skip the following steps and go directly to creating the load balancers.

1. Create a network.

```
tux > neutron network create lb_net1
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 71a1ac88-30a3-48a3-a18b-d98509fbef5c |
| mtu                       | 0                                    |
| name                      | lb_net1                              |
| provider:network_type     | vxlan                                |
| provider:physical_network |                                      |
| provider:segmentation_id  | 1061                                 |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 4b31d0508f83437e83d8f4d520cda22f     |
+---------------------------+--------------------------------------+
```

2. Create a subnet.

```
tux > neutron subnet create --name lb_subnet1 lb_net1 10.247.94.128/26 \
  --gateway 10.247.94.129
+-------------------+----------------------------------------------------+
| Field             | Value                                              |
+-------------------+----------------------------------------------------+
| allocation_pools  | {"start": "10.247.94.130", "end": "10.247.94.190"} |
| cidr              | 10.247.94.128/26                                   |
| dns_nameservers   |                                                    |
| enable_dhcp       | True                                               |
| gateway_ip        | 10.247.94.129                                      |
| host_routes       |                                                    |
| id                | 6fc2572c-53b3-41d0-ab63-342d9515f514               |
| ip_version        | 4                                                  |
| ipv6_address_mode |                                                    |
| ipv6_ra_mode      |                                                    |
| name              | lb_subnet1                                         |
| network_id        | 71a1ac88-30a3-48a3-a18b-d98509fbef5c               |
| subnetpool_id     |                                                    |
| tenant_id         | 4b31d0508f83437e83d8f4d520cda22f                   |
+-------------------+----------------------------------------------------+
```

3. Create a router.

```
tux > neutron router create --distributed False lb_router1
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| distributed           | False                                |
| external_gateway_info |                                      |
| ha                    | False                                |
| id                    | 6aafc9a9-93f6-4d7e-94f2-3068b034b823 |
| name                  | lb_router1                           |
| routes                |                                      |
| status                | ACTIVE                               |
| tenant_id             | 4b31d0508f83437e83d8f4d520cda22f     |
+-----------------------+--------------------------------------+
```

4. Add interface to router. In the following example, the interface `426c5898-f851-4f49-b01f-7a6fe490410c` will be added to the router `lb_router1`.

```
tux > neutron router add subnet lb_router1 lb_subnet1
```

5. Set gateway for router.

```
tux > neutron router set lb_router1 ext-net
```

6. Check networks.

```
tux > neutron network list
+----------------------------+-----------------+----------------------------+
| id                         | name            | subnets                    |
+----------------------------+-----------------+----------------------------+
| d3cb12a6-a000-4e3e-        | ext-net         | f4152001-2500-4ebe-ba9d-   |
| 82c4-ee04aa169291          |                 | a8d6149a50df 10.247.96.0/23 |
| 8306282a-3627-445a-a588-c18 | OCTAVIA-MGMT-NET | f00299f8-3403-45ae-ac4b-   |
| 8b6a13163                  |                 | 58af41d57bdc               |
|                            |                 | 10.247.94.128/26           |
| 71a1ac88-30a3-48a3-a18b-   | lb_net1         | 6fc2572c-                  |
| d98509fbef5c               |                 | 53b3-41d0-ab63-342d9515f514 |
|                            |                 | 10.247.94.128/26           |
+----------------------------+-----------------+----------------------------+
```

7. Create security group.

```
tux > neutron security group create lb_secgroup1
+----------------+---------------------------------------------------------------------------+
| Field          | Value                                                                     |
+----------------+---------------------------------------------------------------------------+
| description    |                                                                            |
| id             | 75343a54-83c3-464c-8773-802598afaee9                                       |
| name           | lb_secgroup1                                                               |
| security group | {"remote_group_id": null, "direction": "egress", "remote_ip_prefix": null,|
|    rules       | "protocol": null, "tenant_id": "4b31d...da22f", "port_range_max": null,   |
|                | "security_group_id": "75343a54-83c3-464c-8773-802598afaee9",              |
|                | "port_range_min": null, "ethertype": "IPv4", "id": "20ae3...97a7a"}       |
|                | {"remote_group_id": null, "direction": "egress",                          |
|                | "remote_ip_prefix": null, "protocol": null, "tenant_id": "4b31...a22f",   |
|                | "port_range_max": null, "security_group_id": "7534...98afaee9",          |
|                | "port_range_min": null, "ethertype": "IPv6", "id": "563c5c...aaef9"}      |
| tenant_id      | 4b31d0508f83437e83d8f4d520cda22f                                           |
+----------------+----------------------------------------------------------------------------+
```

8. Create icmp security group rule.

```
tux > neutron security group rule create lb_secgroup1 --protocol icmp
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 16d74150-a5b2-4cf6-82eb-a6c49a972d93 |
| port_range_max    |                                      |
| port_range_min    |                                      |
| protocol          | icmp                                 |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | 75343a54-83c3-464c-8773-802598afaee9 |
| tenant_id         | 4b31d0508f83437e83d8f4d520cda22f     |
+-------------------+--------------------------------------+
```

9. Create TCP port 22 rule.

```
tux > neutron security group rule create lb_secgroup1 --protocol tcp \
  --port-range-min 22 --port-range-max 22
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 472d3c8f-c50f-4ad2-97a1-148778e73af5 |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| protocol          | tcp                                  |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | 75343a54-83c3-464c-8773-802598afaee9 |
| tenant_id         | 4b31d0508f83437e83d8f4d520cda22f     |
+-------------------+--------------------------------------+
```

10. Create TCP port 80 rule.

```
tux > neutron security group rule create lb_secgroup1 --protocol tcp \
  --port-range-min 80 --port-range-max 80
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 10a76cad-8b1c-46f6-90e8-5dddd279e5f7 |
| port_range_max    | 80                                   |
| port_range_min    | 80                                   |
| protocol          | tcp                                  |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | 75343a54-83c3-464c-8773-802598afaee9 |
| tenant_id         | 4b31d0508f83437e83d8f4d520cda22f     |
+-------------------+--------------------------------------+
```

11. If you have not already created a keypair, create one now with `nova keypair create`. You will use the keypair to boot images.

```
tux > nova keypair create lb_kp1 > lb_kp1.pem

chmod 400 lb_kp1.pem

cat lb_kp1.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEqAIBAAKCAQEAkbW5W/XWGRGC0LAJI7lttR7EdDfiTDeFJ7A9b9Cff+OMXjhx
WL26eKIr+jp8DR64YjV2mNnQLsDyCxekFpkyjnGRId3KVAeV5sRQqXgtaCXI+Rvd
IyUtd8p1cp3DRgTd1dxO0oL6bBmwrZatNrrRn4HgKc2c7ErekeXrwLHyE0Pia/pz
C6qs0coRdfIeXxsmS3kXExP0YfsswRS/OyDl8QhRAF0ZW/zV+DQIi8+HpLZT+RW1
8sTTYZ6b0kXoH9wLER4IUBj1I1IyrYdxlAhe2VIn+tF0Ec4nDBn1py9iwEfGmn0+
N2jHCJAkrK/QhWdXO4O8zeXfL4mCZ9FybW4nzQIDAQABAoIBACe0PvgB+v8FuIGp
FjR32J8b7ShF+hIOpufzrCoFzRCKLruV4bzuphstBZK/0QG6Nz/7lX99Cq9SwCGp
pXrK7+3EoGl8CB/xmTUylVA4gRb6BNNsdkuXW9ZigrJirs0rkk8uIwRV0GsYbP5A
Kp7ZNTmjqDN75aC1ngRfhGgTlQUOdxBH+4xSb7GukekD13V8V5MF1Qft089asdWp
l/TpvhYeW9O92xEnZ3qXQYpXYQgEFQoM2PKa3VW7FGLgfw9gdS/MSqpHuHGyKmjl
uT6upUX+Lofbe7V+9kfxuV32sLL/S5YFvkBy2q8VpuEV1sXI7O7Sc411WX4cqmlb
YoFwhrkCggCBALkYE7OMTtdCAGcMotJhTiiS5l8d4U/fn1x0zus43XV5Y7wCnMuU
r5vCoK+a+TR9Ekzc/GjccAx7Wz/YYKp6G8FXW114dLcADXZjqjIlX7ifUud4sLCS
y+x3KAJa7LqyzH53I6FOts9RaB5xx4gZ2WjcJquCTbATZWj7j1yGeNgvAoIAgQDJ
h0r0Te5IliYbCRg+ES9YRZzH/PSLuIn00bbLvpOPNEoKe2Pxs+KI8Fqp6ZIDAB3c
4EPOK5QrJvAny9Z58ZArrNZi15t84KEVAkWUATl+c4SmHc8sW/atgmUoqIzgDQXe
AlwadHLY7JCdg7EYDuUxuTKLLOdqfpf6fKkhNxtEwwKCAIAMxi+d5aIPUxvKAOI/
2L1XKYRCrkI9i/ZooBsjusH1+JG8iQWfOzy/aDhExlJKoBMiQOIerpABHIZYqqtJ
```

```
OLIvrsK8ebK8aoGDWS+G1HN9v2kuVnMDTK5MPJEDUJkj7XEVjU1lNZSCTGD+MOYP
a5FInmEA1zZbX4tRKoNjZFh0uwKCAIEAiLs7drAdOLBu4C72fL4KIljwu5t7jATD
zRAwduIxmZq/lYcMU2RaEdEJonivsUt193NNbeeRWwnLLSUWupvT1l4pAt0ISNzb
TbbB4F5IVOwpls9ozc8DecubuM9K7YTIc02kkepqNZWjtMsx74HDrU3a5iSsSkvj
73Z/BeMupCMCggCAS48BsrcsDsHSHE3tO4D8pAIr1r+6WPaQn49pT3GIrdQNc7aO
d9PfXmPoe/PxUlqaXoNAvT99+nNEadp+GTId21VM0Y28pn3EkIGE1Cqoeyl3BEO8
f9SUiRNruDnH4F4OclsDBlmqWXImuXRfeiDHxM8X03UDZoqyHmGD3RqA53I=
-----END RSA PRIVATE KEY-----
```

12. Check and boot images.

```
tux > nova image list
+--------------+------------------------+--------+--------+
| ID           | Name                   | Status | Server |
+--------------+------------------------+--------+--------+
| 0526d...7f39 | cirros-0.4.0-x86_64    | ACTIVE |        |
| 8aa51...8f2f | octavia-amphora-x86_64 | ACTIVE |        |
+--------------+------------------------+--------+--------+
```

Boot first VM.

```
tux > nova server create --flavor 1 --image 04b1528b-b1e2-45d4-96d1-fbe04c6b2efd --key-name
 lb_kp1 \
  --security-groups lb_secgroup1 --nic net-id=71a1ac88-30a3-48a3-a18b-d98509fbef5c \
  lb_vm1 --poll
+---------------------------------------+--------------------------------------+
| Property                              | Value                                |
+---------------------------------------+--------------------------------------+
| OS-DCF:diskConfig                     | MANUAL                               |
| OS-EXT-AZ:availability_zone           |                                      |
| OS-EXT-SRV-ATTR:host                  | -                                    |
| OS-EXT-SRV-ATTR:hypervisor_hostname   | -                                    |
| OS-EXT-SRV-ATTR:instance_name         | instance-00000031                    |
| OS-EXT-STS:power_state                | 0                                    |
| OS-EXT-STS:task_state                 | scheduling                           |
| OS-EXT-STS:vm_state                   | building                             |
| OS-SRV-USG:launched_at                | -                                    |
| OS-SRV-USG:terminated_at              | -                                    |
| accessIPv4                            |                                      |
| accessIPv6                            |                                      |
| adminPass                             | NeVvhP5E8iCy                         |
| config_drive                          |                                      |
| created                               | 2016-06-15T16:53:00Z                 |
| flavor                                | m1.tiny (1)                          |
| hostId                                |                                      |
| id                                    | dfdfe15b-ce8d-469c-a9d8-2cea0e7ca287 |
| image                                 | cirros-0.4.0-x86_64 (0526d...7f39)   |
| key_name                              | lb_kp1                               |
| metadata                              | {}                                   |
| name                                  | lb_vm1                               |
| os-extended-volumes:volumes_attached | []                                   |
| progress                             | 0                                    |
| security_groups                      | lb_secgroup1                         |
| status                               | BUILD                                |
| tenant_id                            | 4b31d0508f83437e83d8f4d520cda22f     |
| updated                              | 2016-06-15T16:53:00Z                 |
| user_id                              | fd471475faa84680b97f18e55847ec0a     |
+--------------------------------------+--------------------------------------+


            Server building... 100% complete
            Finished
```

Boot second VM.

```
tux > nova server create --flavor 1 --image 04b1528b-b1e2-45d4-96d1-fbe04c6b2efd --key-name
 lb_kp1 \
  --security-groups lb_secgroup1 --nic net-id=71a1ac88-30a3-48a3-a18b-d98509fbef5c \
  lb_vm2 --poll
+--------------------------------------+---------------------------------------+
| Property                             | Value                                 |
+--------------------------------------+---------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                |
| OS-EXT-AZ:availability_zone          |                                       |
| OS-EXT-SRV-ATTR:host                 | -                                     |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                     |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000034                     |
| OS-EXT-STS:power_state               | 0                                     |
| OS-EXT-STS:task_state                | scheduling                            |
| OS-EXT-STS:vm_state                  | building                              |
| OS-SRV-USG:launched_at               | -                                     |
| OS-SRV-USG:terminated_at             | -                                     |
| accessIPv4                           |                                       |
| accessIPv6                           |                                       |
| adminPass                            | 3nFXjNrTrmNm                          |
| config_drive                         |                                       |
| created                              | 2016-06-15T16:55:10Z                  |
| flavor                               | m1.tiny (1)                           |
| hostId                               |                                       |
| id                                   | 3844bb10-2c61-4327-a0d4-0c043c674344  |
| image                                | cirros-0.4.0-x86_64 (0526d...7f39)    |
| key_name                             | lb_kp1                                |
| metadata                             | {}                                    |
| name                                 | lb_vm2                                |
| os-extended-volumes:volumes_attached | []                                    |
| progress                             | 0                                     |
| security_groups                      | lb_secgroup1                          |
| status                               | BUILD                                 |
| tenant_id                            | 4b31d0508f83437e83d8f4d520cda22f      |
| updated                              | 2016-06-15T16:55:09Z                  |
| user_id                              | fd471475faa84680b97f18e55847ec0a      |
+--------------------------------------+---------------------------------------+
            Server building... 100% complete
            Finished
```

13. List the running VM with `nova list`

```
tux > nova server list
+----------------+--------+--------+------------+-------------+-----------------------+
| ID             | Name   | Status | Task State | Power State | Networks              |
+----------------+--------+--------+------------+-------------+-----------------------+
| dfdfe...7ca287 | lb_vm1 | ACTIVE | -          | Running     | lb_net1=10.247.94.132 |
| 3844b...674344 | lb_vm2 | ACTIVE | -          | Running     | lb_net1=10.247.94.133 |
+----------------+--------+--------+------------+-------------+-----------------------+
```

14. Check ports.

```
tux > neutron port list
+----------------+------+-------------------+-------------------------------+
| id             | name | mac_address       | fixed_ips                     |
+----------------+------+-------------------+-------------------------------+
...
| 7e5e0...36450e |      | fa:16:3e:66:fd:2e | {"subnet_id": "6fc25...5f514", |
|                |      |                   | "ip_address": "10.247.94.132"} |
| ca95c...b36854 |      | fa:16:3e:e0:37:c4 | {"subnet_id": "6fc25...5f514", |
|                |      |                   | "ip_address": "10.247.94.133"} |
+----------------+------+-------------------+-------------------------------+
```

15. Create the first floating IP.

```
tux > neutron floating ip create ext-net --port-id 7e5e0038-88cf-4f97-a366-b58cd836450e
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.247.94.132                        |
| floating_ip_address | 10.247.96.26                         |
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291 |
| id                  | 3ce608bf-8835-4638-871d-0efe8ebf55ef |
| port_id             | 7e5e0038-88cf-4f97-a366-b58cd836450e |
| router_id           | 6aafc9a9-93f6-4d7e-94f2-3068b034b823 |
| status              | DOWN                                 |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f     |
+---------------------+--------------------------------------+
```

16. Create the second floating IP.

```
tux > neutron floating ip create ext-net --port-id ca95cc24-4e8f-4415-9156-7b519eb36854
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.247.94.133                        |
| floating_ip_address | 10.247.96.27                         |
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291 |
| id                  | 680c0375-a179-47cb-a8c5-02b836247444 |
| port_id             | ca95cc24-4e8f-4415-9156-7b519eb36854 |
| router_id           | 6aafc9a9-93f6-4d7e-94f2-3068b034b823 |
| status              | DOWN                                 |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f     |
+---------------------+--------------------------------------+
```

17. List the floating IP's.

```
tux > neutron floating ip list
+----------------+------------------+---------------------+---------------+
| id             | fixed_ip_address | floating_ip_address | port_id       |
+----------------+------------------+---------------------+---------------+
| 3ce60...bf55ef | 10.247.94.132    | 10.247.96.26        | 7e5e0...6450e |
| 680c0...247444 | 10.247.94.133    | 10.247.96.27        | ca95c...36854 |
+----------------+------------------+---------------------+---------------+
```

18. Show first Floating IP.

```
tux > neutron floating ip show 3ce608bf-8835-4638-871d-0efe8ebf55ef
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.247.94.132                        |
| floating_ip_address | 10.247.96.26                         |
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291 |
| id                  | 3ce608bf-8835-4638-871d-0efe8ebf55ef |
| port_id             | 7e5e0038-88cf-4f97-a366-b58cd836450e |
| router_id           | 6aafc9a9-93f6-4d7e-94f2-3068b034b823 |
| status              | ACTIVE                               |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f     |
+---------------------+--------------------------------------+
```

19. Show second Floating IP.

```
tux > neutron floating ip show 680c0375-a179-47cb-a8c5-02b836247444
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.247.94.133                        |
| floating_ip_address | 10.247.96.27                         |
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291 |
| id                  | 680c0375-a179-47cb-a8c5-02b836247444 |
| port_id             | ca95cc24-4e8f-4415-9156-7b519eb36854 |
| router_id           | 6aafc9a9-93f6-4d7e-94f2-3068b034b823 |
| status              | ACTIVE                               |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f     |
+---------------------+--------------------------------------+
```

20. Ping first Floating IP.

```
tux > ping -c 1 10.247.96.26
PING 10.247.96.26 (10.247.96.26) 56(84) bytes of data.
64 bytes from 10.247.96.26: icmp_seq=1 ttl=62 time=3.50 ms

--- 10.247.96.26 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.505/3.505/3.505/0.000 ms
```

21. Ping second Floating IP.

```
tux > ping -c 1 10.247.96.27
PING 10.247.96.27 (10.247.96.27) 56(84) bytes of data.
64 bytes from 10.247.96.27: icmp_seq=1 ttl=62 time=3.47 ms

--- 10.247.96.27 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.473/3.473/3.473/0.000 ms
```

22. Listing the VMs will give you both the fixed and floating IP's for each virtual machine.

```
tux > nova list
+---------------+--------+--------+-------+---------+-----------------------------------+
| ID            | Name   | Status | Task  | Power   | Networks                          |
|               |        |        | State | State   |                                   |
+---------------+--------+--------+-------+---------+-----------------------------------+
| dfdfe...ca287 | lb_vm1 | ACTIVE | -     | Running | lb_net1=10.247.94.132, 10.247.96.26 |
| 3844b...74344 | lb_vm2 | ACTIVE | -     | Running | lb_net1=10.247.94.133, 10.247.96.27 |
+---------------+--------+--------+-------+---------+-----------------------------------+
```

23. List Floating IP's.

```
tux > neutron floating ip list
+---------------+-----------------+--------------------+----------------+
| id            | fixed_ip_address | floating_ip_address | port_id        |
+---------------+-----------------+--------------------+----------------+
| 3ce60...f55ef | 10.247.94.132    | 10.247.96.26        | 7e5e00...36450e |
| 680c0...47444 | 10.247.94.133    | 10.247.96.27        | ca95cc...b36854 |
+---------------+-----------------+--------------------+----------------+
```

27.4 Create Load Balancers

The following steps will setup new Octavia Load Balancers.

Note

The following examples assume names and values from the previous section.

1. Create load balancer for subnet

```
tux > neutron lbaas-loadbalancer-create --provider octavia \
  --name lb1 6fc2572c-53b3-41d0-ab63-342d9515f514
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| description         |                                      |
| id                  | 3d9170a1-8605-43e6-9255-e14a8b4aae53 |
| listeners           |                                      |
| name                | lb1                                  |
| operating_status    | OFFLINE                              |
| provider            | octavia                              |
| provisioning_status | PENDING_CREATE                       |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f     |
| vip_address         | 10.247.94.134                        |
| vip_port_id         | da28aed3-0eb4-4139-afcf-2d8fd3fc3c51 |
| vip_subnet_id       | 6fc2572c-53b3-41d0-ab63-342d9515f514 |
+---------------------+--------------------------------------+
```

2. List load balancers. You will need to wait until the load balancer `provisioning_status` is `ACTIVE` before proceeding to the next step.

```
tux > neutron lbaas-loadbalancer-list
+----------------+------+---------------+---------------------+----------+
| id             | name | vip_address   | provisioning_status | provider |
+----------------+------+---------------+---------------------+----------+
| 3d917...aae53 | lb1  | 10.247.94.134 | ACTIVE              | octavia  |
+----------------+------+---------------+---------------------+----------+
```

3. Once the load balancer is created, create the listener. This may take some time.

```
tux > neutron lbaas-listener-create --loadbalancer lb1 \
  --protocol HTTP --protocol-port=80 --name lb1_listener
+---------------------------+------------------------------------------------+
| Field                     | Value                                          |
+---------------------------+------------------------------------------------+
| admin_state_up            | True                                           |
| connection_limit          | -1                                             |
| default_pool_id           |                                                |
| default_tls_container_ref |                                                |
| description               |                                                |
| id                        | c723b5c8-e2df-48d5-a54c-fc240ac7b539           |
| loadbalancers             | {"id": "3d9170a1-8605-43e6-9255-e14a8b4aae53"} |
| name                      | lb1_listener                                   |
| protocol                  | HTTP                                           |
| protocol_port             | 80                                             |
| sni_container_refs        |                                                |
| tenant_id                 | 4b31d0508f83437e83d8f4d520cda22f               |
+---------------------------+------------------------------------------------+
```

4. Create the load balancing pool. During the creation of the load balancing pool, the status for the load balancer goes to `PENDING_UPDATE`. Use `neutron lbaas-loadbalancer-list` to watch for the change to `ACTIVE`. Once the load balancer returns to `ACTIVE`, proceed with the next step.

```
tux > neutron lbaas-pool-create --lb-algorithm ROUND_ROBIN \
  --listener lb1_listener --protocol HTTP --name lb1_pool
+---------------------+------------------------------------------------+
| Field               | Value                                          |
+---------------------+------------------------------------------------+
| admin_state_up      | True                                           |
| description         |                                                |
| healthmonitor_id    |                                                |
| id                  | 0f5951ee-c2a0-4e62-ae44-e1491a8988e1           |
| lb_algorithm        | ROUND_ROBIN                                    |
| listeners           | {"id": "c723b5c8-e2df-48d5-a54c-fc240ac7b539"} |
| members             |                                                |
| name                | lb1_pool                                       |
| protocol            | HTTP                                           |
| session_persistence |                                                |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f               |
+---------------------+------------------------------------------------+
```

5. Create first member of the load balancing pool.

```
tux > neutron lbaas-member-create --subnet 6fc2572c-53b3-41d0-ab63-342d9515f514 \
  --address 10.247.94.132 --protocol-port 80 lb1_pool
+----------------+---------------------------------------+
| Field          | Value                                 |
+----------------+---------------------------------------+
| address        | 10.247.94.132                         |
| admin_state_up | True                                  |
| id             | 61da1e21-e0ae-4158-935a-c909a81470e1 |
| protocol_port  | 80                                   |
| subnet_id      | 6fc2572c-53b3-41d0-ab63-342d9515f514 |
| tenant_id      | 4b31d0508f83437e83d8f4d520cda22f     |
| weight         | 1                                    |
+----------------+--------------------------------------+
```

6. Create the second member.

```
tux > neutron lbaas-member-create --subnet 6fc2572c-53b3-41d0-ab63-342d9515f514 \
  --address 10.247.94.133 --protocol-port 80 lb1_pool
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| address        | 10.247.94.133                        |
| admin_state_up | True                                 |
| id             | 459c7f21-46f7-49e8-9d10-dc7da09f8d5a |
| protocol_port  | 80                                   |
| subnet_id      | 6fc2572c-53b3-41d0-ab63-342d9515f514 |
| tenant_id      | 4b31d0508f83437e83d8f4d520cda22f     |
| weight         | 1                                    |
+----------------+--------------------------------------+
```

7. You should check to make sure the load balancer is active and check the pool members.

```
tux > neutron lbaas-loadbalancer-list
+----------------+------+---------------+---------------------+----------+
| id             | name | vip_address   | provisioning_status | provider |
+----------------+------+---------------+---------------------+----------+
| 3d9170...aae53 | lb1  | 10.247.94.134 | ACTIVE              | octavia  |
+----------------+------+---------------+---------------------+----------+

neutron lbaas-member-list lb1_pool
+---------------+---------------+---------------+--------+---------------+----------------+
| id            | address       | protocol_port | weight | subnet_id     | admin_state_up |
+---------------+---------------+---------------+--------+---------------+----------------+
| 61da1...470e1 | 10.247.94.132 |            80 |      1 | 6fc25...5f514 | True           |
| 459c7...f8d5a | 10.247.94.133 |            80 |      1 | 6fc25...5f514 | True           |
+---------------+---------------+---------------+--------+---------------+----------------+
```

8. You can view the details of the load balancer, listener and pool.

```
tux > neutron lbaas-loadbalancer-show 3d9170a1-8605-43e6-9255-e14a8b4aae53
+---------------------+--------------------------------------------------+
| Field               | Value                                            |
+---------------------+--------------------------------------------------+
| admin_state_up      | True                                             |
| description         |                                                  |
| id                  | 3d9170a1-8605-43e6-9255-e14a8b4aae53             |
| listeners           | {"id": "c723b5c8-e2df-48d5-a54c-fc240ac7b539"} |
| name                | lb1                                            |
| operating_status    | ONLINE                                         |
| provider            | octavia                                        |
| provisioning_status | ACTIVE                                         |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f               |
| vip_address         | 10.247.94.134                                  |
| vip_port_id         | da28aed3-0eb4-4139-afcf-2d8fd3fc3c51           |
| vip_subnet_id       | 6fc2572c-53b3-41d0-ab63-342d9515f514           |
+---------------------+------------------------------------------------+

neutron lbaas-listener-list
+-------------+-----------------+-------------+----------+--------------+----------------
+
| id          | default_pool_id | name        | protocol | protocol_port | admin_state_up
 |
+-------------+-----------------+-------------+----------+--------------+----------------
+
| c723...b539 | 0f595...8988e1  | lb1_listener | HTTP     |            80 | True
 |
+-------------+-----------------+-------------+----------+--------------+----------------
+

neutron lbaas-pool-list
+--------------------------------------+----------+----------+----------------+
| id                                   | name     | protocol | admin_state_up |
+--------------------------------------+----------+----------+----------------+
| 0f5951ee-c2a0-4e62-ae44-e1491a8988e1 | lb1_pool | HTTP     | True           |
+--------------------------------------+----------+----------+----------------+
```

27.5 Create Floating IPs for Load Balancer

To create the floating IP's for the load balancer, you will need to list the current ports to get the load balancer id. Once you have the id, you can then create the floating IP.

1. List the current ports.

```
tux > neutron port list
+--------------+--------------+-------------------+---------------------------------------+
| id           | name         | mac_address       | fixed_ips                             |
+--------------+--------------+-------------------+---------------------------------------+
...
| 7e5e...6450e |              | fa:16:3e:66:fd:2e | {"subnet_id": "6fc2572c-              |
|              |              |                   | 53b3-41d0-ab63-342d9515f514",         |
|              |              |                   | "ip_address": "10.247.94.132"}        |
| a3d0...55efe |              | fa:16:3e:91:a2:5b | {"subnet_id": "f00299f8-3403-45ae-ac4b- |
|              |              |                   | 58af41d57bdc", "ip_address":          |
|              |              |                   | "10.247.94.142"}                      |
| ca95...36854 |              | fa:16:3e:e0:37:c4 | {"subnet_id": "6fc2572c-              |
|              |              |                  | 53b3-41d0-ab63-342d9515f514",           |
|              |              |                  | "ip_address": "10.247.94.133"}          |
| da28...c3c51 | loadbalancer- | fa:16:3e:1d:a2:1c | {"subnet_id": "6fc2572c-                |
|              | 3d917...aae53 |                  | 53b3-41d0-ab63-342d9515f514",           |
|              |              |                  | "ip_address": "10.247.94.134"}          |
+--------------+--------------+------------------+-----------------------------------------+
```

2. Create the floating IP for the load balancer.

```
tux > neutron floating ip create ext-net --port-id da28aed3-0eb4-4139-afcf-2d8fd3fc3c51
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.247.94.134                        |
| floating_ip_address | 10.247.96.28                         |
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291 |
| id                  | 9a3629bd-b0a6-474c-abe9-89c6ecb2b22c |
| port_id             | da28aed3-0eb4-4139-afcf-2d8fd3fc3c51 |
| router_id           | 6aafc9a9-93f6-4d7e-94f2-3068b034b823 |
| status              | DOWN                                 |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f     |
+---------------------+--------------------------------------+
```

27.6 Testing the Octavia Load Balancer

To test the load balancers, create the following web server script so you can run it on each virtual machine. You will use `curl <ip address>` to test if the load balance services are responding properly.

1. Start running web servers on both of the virtual machines. Create the webserver.sh script with below contents. In this example, the port is 80.

```
tux > vi webserver.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
```

2. Deploy the web server and run it on the first virtual machine.

```
ardana > ssh-keygen -R 10.247.96.26
```

```
/home/ardana/.ssh/known_hosts updated.
Original contents retained as /home/ardana/.ssh/known_hosts.old

scp -o StrictHostKeyChecking=no -i lb_kp1.pem webserver.sh cirros@10.247.96.26:
webserver.sh                                       100%  263     0.3KB/s   00:00

ssh -o StrictHostKeyChecking=no -i lb_kp1.pem cirros@10.247.96.26 'chmod +x ./webserver.sh'
ssh -o StrictHostKeyChecking=no -i lb_kp1.pem cirros@10.247.96.26 ./webserver.sh
```

3. Test the first web server.

```
tux > curl 10.247.96.26
 Welcome to 10.247.94.132
```

4. Deploy and start the web server on the second virtual machine like you did in the previous steps. Once the second web server is running, list the floating IPs.

```
tux > neutron floating ip list
+-----------------+------------------+---------------------+----------------+
| id              | fixed_ip_address | floating_ip_address | port_id        |
+-----------------+------------------+---------------------+----------------+
| 3ce60...bf55ef | 10.247.94.132    | 10.247.96.26        | 7e5e0...6450e |
| 680c0...247444 | 10.247.94.133    | 10.247.96.27        | ca95c...36854 |
| 9a362...b2b22c | 10.247.94.134    | 10.247.96.28        | da28a...c3c51 |
+-----------------+------------------+---------------------+----------------+
```

5. Display the floating IP for the load balancer.

```
tux > neutron floating ip show 9a3629bd-b0a6-474c-abe9-89c6ecb2b22c
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.247.94.134                        |
| floating_ip_address | 10.247.96.28                         |
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291 |
| id                  | 9a3629bd-b0a6-474c-abe9-89c6ecb2b22c |
| port_id             | da28aed3-0eb4-4139-afcf-2d8fd3fc3c51 |
| router_id           | 6aafc9a9-93f6-4d7e-94f2-3068b034b823 |
| status              | ACTIVE                               |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f     |
+---------------------+--------------------------------------+
```

6. Finally, test the load balancing.

```
tux > curl 10.247.96.28
Welcome to 10.247.94.132
```

```
tux > curl 10.247.96.28
Welcome to 10.247.94.133

tux > curl 10.247.96.28
Welcome to 10.247.94.132

tux > curl 10.247.96.28
Welcome to 10.247.94.133

tux > curl 10.247.96.28
Welcome to 10.247.94.132

tux > curl 10.247.96.28
Welcome to 10.247.94.133
```

28 Other Common Post-Installation Tasks

28.1 Determining Your User Credentials

On your Cloud Lifecycle Manager, in the `~/scratch/ansible/next/ardana/ansible/group_vars/` directory you will find several files. In the one labeled as first control plane node you can locate the user credentials for both the Administrator user (`admin`) and your Demo user (`demo`) which you will use to perform many other actions on your cloud.

For example, if you are using the Entry-scale KVM model and used the default naming scheme given in the example configuration files, you can use these commands on your Cloud Lifecycle Manager to **`grep`** for your user credentials:

Administrator

```
grep keystone_admin_pwd entry-scale-kvm-control-plane-1
```

Demo

```
grep keystone_demo_pwd entry-scale-kvm-control-plane-1
```

28.2 Configure your Cloud Lifecycle Manager to use the command-line tools

This playbook will do a series of steps to update your environment variables for your cloud so you can use command-line clients.

Run the following command, which will replace `/etc/hosts` on the Cloud Lifecycle Manager:

```
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts cloud-client-setup.yml
```

As the `/etc/hosts` file no longer has entries for Cloud Lifecycle Manager, sudo commands may become a bit slower. To fix this issue, once this step is complete, add "ardana" after "127.0.0.1 localhost". The result will look like this:

```
...
# Localhost Information
```

```
127.0.0.1 localhost ardana
```

28.3 Protect home directory

The home directory of the user that owns the SUSE OpenStack Cloud 8 scripts should not be world readable. Change the permissions so that they are only readable by the owner:

```
chmod 0700 ~
```

28.4 Back up Your SSH Keys

As part of the cloud deployment setup process, SSH keys to access the systems are generated and stored in `~/.ssh` on your Cloud Lifecycle Manager.

These SSH keys allow access to the subsequently deployed systems and should be included in the list of content to be archived in any backup strategy.

28.5 Retrieving Service Endpoints

1. Log in to your Cloud Lifecycle Manager.
2. Source the keystone admin credentials:

   ```
   unset OS_TENANT_NAME
   source ~/keystone.osrc
   ```

3. Using the OpenStack command-line tool you can then query the Keystone service for your endpoints:

   ```
   openstack endpoint list
   ```

Tip

You can use `openstack -h` to access the client help file and a full list of commands.

To learn more about Keystone, see *Book "Operations Guide", Chapter 4 "Managing Identity", Section 4.1 "The Identity Service"*.

28.6 Other Common Post-Installation Tasks

Here are the links to other common post-installation tasks that either the Administrator or Demo users can perform:

- *Book "Operations Guide", Chapter 5 "Managing Compute", Section 5.4 "Enabling the Nova Resize and Migrate Features"*
- *Section 23.4, "Creating an External Network"*
- *Section 23.3, "Uploading an Image for Use"*
- *Book "User Guide Overview", Chapter 8 "Creating a Private Network"*
- *Book "Operations Guide", Chapter 8 "Managing Object Storage", Section 8.1 "Running the Swift Dispersion Report"*

29 Support

Find solutions for the most common pitfalls and technical details on how to create a support request for SUSE OpenStack Cloud here.

29.1 FAQ

29.1.1 Node Deplyoment

Q: *How to Disable the YaST Installer Self-Update when deplyoying nodes?*

A: Prior to starting an installation, the YaST installer can update itself if respective updates are available. By default this feature is enabled. In case of problems with this feature, disable it as follows:

1. Open `~/openstack/ardana/ansible/roles/cobbler/templates/sles.grub.j2` with an editor and add `self_update=0` to the line starting with `linuxefi`. The results needs to look like the following:

```
linuxefi images/{{ sles_profile_name }}-x86_64/linux ifcfg={{ item[0] }}=dhcp
 install=http://{{ cobbler_server_ip_addr }}:79/cblr/ks_mirror/
{{ sles_profile_name }} self_update=0 AutoYaST2=http://
{{ cobbler_server_ip_addr }}:79/cblr/svc/op/ks/system/{{ item[1] }}
```

2. Commit your changes:

```
git commit -m "Disable Yast Self Update feature" \
~/openstack/ardana/ansible/roles/cobbler/templates/sles.grub.j2
```

3. If you need to reenable the installer self-update, remove `self_update=0` and commit the changes.

29.2 Support

Before contacting support to help you with a problem on SUSE OpenStack Cloud, it is strongly recommended that you gather as much information about your system and the problem as possible. For this purpose, SUSE OpenStack Cloud ships with a tool called **`supportconfig`**. It gathers system information such as the current kernel version being used, the hardware, RPM database, partitions, and other items. **`supportconfig`** also collects the most important log files, making it easier for the supporters to identify and solve your problem.

It is recommended to always run **`supportconfig`** on the CLM Server and on the Control Node(s). If a Compute Node or a Storage Node is part of the problem, run **`supportconfig`** on the affected node as well. For details on how to run **`supportconfig`**, see http://www.suse.com/documentation/sles-12/book_sle_admin/data/cha_adm_support.html.

29.2.1 Applying PTFs (Program Temporary Fixes) Provided by the SUSE L3 Support

Under certain circumstances, the SUSE support may provide temporary fixes, the so-called PTFs, to customers with an L3 support contract. These PTFs are provided as RPM packages. To make them available on all nodes in SUSE OpenStack Cloud, proceed as follows. If you prefer to test them first on a single node see FIXME

1. Download the packages from the location provided by the SUSE L3 Support to a temporary location on the CLM Server.

2. Move the packages from the temporary download location to the following directories on the CLM Server:

   "noarch" packages (`*.noarch.rpm`):
   : `/srv/www/suse-12.2/x86_64/repos/PTF/rpm/noarch/`

   "x86_64" packages (`*.x86_64.rpm`)
   : `/srv/www/suse-12.2/x86_64/repos/PTF/rpm/x86_64/`

3. Create or update the repository metadata:

```
createrepo-cloud-ptf
```

4. To deploy the updates, proceed as described in *Book "Operations Guide", Chapter 13 "System Maintenance", Section 13.3 "Cloud Lifecycle Manager Maintenance Update Procedure"* and refresh the PTF repository before installing package updates on a node:

```
zypper refresh -fr PTF
```

29.2.2 Testing PTFs (Program Temporary Fixes) on a Single Node

If you want to test a PTF (Program Temporary Fixes) before deploying it on all nodes, if you want to verify that it fixes a certain issue, you can manually install the PTF on a single node.

EXAMPLE 29.1: TESTING A FIX FOR NOVA

In the following example, a PTF named `venv-openstack-nova-x86_64-ptf.rpm`, containing a fix for Nova, is installed on the Compute Node 01.

1. Check the version number of the package(s) that will be upgraded with the PTF. Run the following command on the deployer node:

```
rpm -q venv-openstack-nova-x86_64
```

2. Install the PTF on the deployer node:

```
sudo zypper up ./venv-openstack-nova-x86_64-ptf.rpm
```

This will install a new TAR archive in `/opt/ardana_packager/ardana-8/sles_venv/x86_64/`.

3. Register the TAR archive with the indexer:

```
sudo create_index --dir
      /opt/ardana_packager/ardana-8/sles_venv/x86_64
```

This will update the indexer `/opt/ardana_packager/ardana-8/sles_venv/x86_64/packages`.

4. Deploy the fix on Compute Node 01:

   a. Check whether the fix can be deployed on a single Compute Node without updating the Control Nodes:

```
cd ~/scratch/ansible/next/ardana/ansible
```

```
ansible-playbook -i hosts/verb_hosts nova-upgrade.yml \
--limit=inputmodel-ccp-compute0001-mgmt --list-hosts
```

b. If the previous test passes, install the fix:

```
ansible-playbook -i hosts/verb_hosts nova-upgrade.yml \
--limit=inputmodel-ccp-compute0001-mgmt
```

5. Validate the fix, for example by logging in to the Compute Node to check the log files:

```
ssh ardana@inputmodel-ccp-compute0001-mgmt
```

6. In case your tests are positive, install the PTF on all nodes as described in *Section 29.2.1, "Applying PTFs (Program Temporary Fixes) Provided by the SUSE L3 Support"*.
In case the test are negative uninstall the fix and restore the previous state of the Compute Node by running the following commands on the deployer node;

```
sudo zypper install --force venv-openstack-nova-x86_64-OLD-VERSION
cd ~/scratch/ansible/next/ardana/ansible
ansible-playbook -i hosts/verb_hosts nova-upgrade.yml \
--limit=inputmodel-ccp-compute0001-mgmt
```

Make sure to replace *`OLD-VERSION`* with the version number you checked in the first step.