

OpenMetal Operator's Manual - Private Cloud Core

version

Nicholas West

February 23, 2022

Contents

OpenStack Operator's Manual - Private Cloud Core	1
Introduction	1
Getting Started with the Operator's Manual	1
Why Days?	1
Brief Summary of each Day	1
Operator's Manual Audience	2
Table of Contents	2
Day 1	2
Introduction to OpenMetal Central and Your Private Cloud Core	2
Introduction	2
A Brief Overview of Your OpenMetal Private Cloud	2
How to View Your Hardware Assets	2
How to Get Support	4
How to Access OpenMetal Documentation	4
How to Submit a Support Ticket	5
Getting Started with OpenStack Horizon	7
Introduction	7
How to Log in to Horizon	7
Step 1: Log in to OpenMetal Central	7
Step 2: Navigate to Cloud's Details Page	7
Step 3: Load Horizon URL	8
Step 4: SSH into a Hardware Node, Obtain Horizon Password	8
Requirements to SSH into a node	8
Step 5: Log in to Horizon	9
Create a User and Project in OpenStack Horizon	9
Introduction	9
Quick Overview: Project, Admin, and Identity Tabs	9
How to Create a Project	10
Create your First Project	10
Project Quotas	11
How to Create a User and Associate with Project	12
Reference	13
Manage and Upload Images in OpenStack Horizon	13
Introduction	13
Managing Images	13
Uploading Images	13
Create Images From Running Instances	15
Reference	15
How to Create an Instance in OpenStack Horizon	15
Introduction	15
Networking	15
Create a Private Network	15
Create a Router	17
Connect Router to Private Network	17

View Network Topology	18
Security Groups	19
Create an SSH Security Group	19
Add Rule to SSH Security Group	19
How to Create your First Instance	20
Prerequisites	20
SSH Public Key	20
Operating System Image	20
Create your First Instance	20
Assign and Attach Floating IP	23
How to Install and Use OpenStack's CLI	24
Introduction	24
How to Install OpenStackClient	25
Prerequisites	25
Install OpenStackClient	25
Initial Preparation	25
Prepare and Install OpenStackClient	25
Command Structure	27
List all Available Subcommands	27
Learn more about a Subcommand	27
List Items and Show Details	27
Enable Bash Autocompletion	27
Reference	28
Create SSH Key Pair for an OpenStack Control Plane Node	28
Introduction	28
How to Create an SSH Key Pair	28
Prerequisites	28
Create the Key Pair	28
Conclusion	29
Day 2	29
How Private Clouds are Deployed	29
Introduction	29
Initial Deployment	29
Containerization of OpenStack	29
Advantages of Containerization Through Docker	29
Disk Storage and Ceph	30
Object Storage	30
Block Storage	30
Advantages of using Ceph	30
Introduction to Ceph	30
Introduction	30
Advantages of Ceph	30
Data Resiliency	30
Ceph Scales Extremely Well	30
Disadvantages of Ceph	30
Ceph Version Used by Private Clouds	30

View Disk Usage of the Ceph Cluster	30
Default Configuration for the Ceph Cluster	31
Default Ceph Pools	31
Pool: images	31
Pool: volumes	31
Pool: vms	31
Pool: backups	31
Swift and Cinder Ceph Configuration	31
Reconfiguring your Ceph Cluster	32
How to Check Ceph's Status and Disk Usage	32
Introduction	32
Prerequisites	32
Check Ceph Status	32
Check Ceph Disk Usage	32
OpenStack Hardware Node Maintenance	33
Introduction	33
Prerequisites	33
How to Perform Operating System Updates	33
Before Performing OpenStack Maintenance	33
Getting Started	33
Disable Docker socket	34
Stop Docker socket	34
Disable Docker service	34
Stop Docker service	34
Update DNF	34
Reboot the node	35
Verify successful reboot	35
How to Obtain Latest OpenStack Images using Kolla Ansible	36
Getting Started	36
Pull latest Kolla Ansible images	36
Deploy Kolla Ansible images	36
Datadog	37
Resources of a Private Cloud	38
View Memory and Compute Usage in Horizon	38
View Instance State Across Cluster	38
How to Access Resource Information from Ceph	38
Adding nodes to your Ceph Cluster	39
Removing nodes from your Ceph Cluster	39
How to Live Migrate Instances Using OpenStack Horizon	39
Introduction	39
Prerequisite	39
Determining an Instance's Parent Host	40
Migrate Instance	40
Day 3	42
Cloud Hardware Selection	42
Types of Private Clouds	42

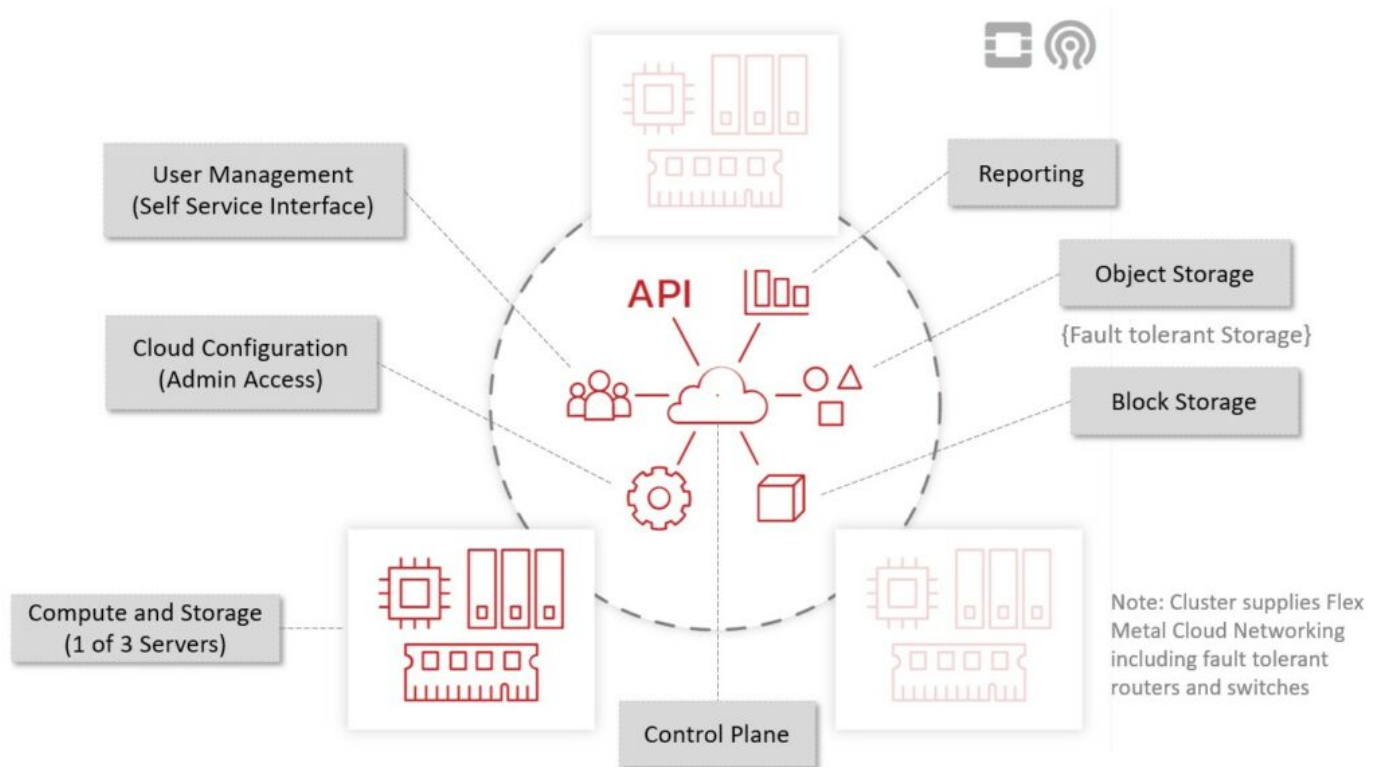
Types of Nodes	43
The Benefit of Homogeneous Clouds	43
Adding Hardware Nodes to a Cloud	43
How to add a Hardware Node	43
Navigate in OpenMetal Central to Cloud Assets Page	43
View Hardware Node Types	44
Confirm Hardware Node Addition	44
Verify Hardware Addition Success	45
Removing Hardware Nodes from a Cloud	45
Consider Before Removing	45
Initial Preparation	46
How to Remove a Hardware Node from a Cloud	46
Getting Started	46
Confirm new Provider Block Addition	46
How are the new Provider Block IPs Used?	46
Create a Volume Backup	47
Test Volume Backups	48
Restore a Volume Backup	48
Ceph, Volumes, and Data Durability	48
Store Data Outside of the Cloud	48
Kolla Ansible and Ceph Ansible	49
Prepare Kolla Ansible and Ceph Ansible Environment	49
Where are my Private Cloud's Configuration Files?	49
Kolla Ansible Configuration Files	49
Ceph Ansible Configuration Files	49
FM-Deploy Configuration File	50
Network Ansible Configuration File	50
Keep a Backup Copy a Private Cloud's Configuration Files	50
How to Restore a Private Cloud's Configuration Files	50
Example: Recover Neutron's Configuration File using Kolla Ansible	50
Prerequisite: Prepare a Kolla Ansible Environment	50
Regenerate an OpenStack Service's Configuration File using Kolla Ansible	50
Create Full and Incremental Copies of a Private Cloud's OpenStack Service Databases	51
Prerequisites	51
How to Create OpenStack Service Database Backups	51
Command Syntax for Full Database Backups	51
Command Syntax for Incremental Database Backups	51
Path to the Kolla Ansible Inventory File	51
Command Usage Example for a Full Database Backup	51
Command Usage Example for an Incremental Database Backup	52
How to Restore a Private Cloud's OpenStack Service Databases	52
Full Database Restoration Steps	52
Full Restoration: Create Temporary Docker Container	53
Full Restoration: Prepare Backup Directory	53
Incremental Database Restoration Steps	53
Incremental Restoration: Create Temporary Docker Container	53

Incremental Restoration: Prepare Backup Directory	54
References	54
Disaster Recovery Strategies	55
Recovery Objectives	55
Off-site Backups	55
RBD Mirroring with Ceph	55
Handling a Hardware Failure	55
Determine Hardware Node Failure	56
Cluster Failure	56
Cloud Monitoring with Datadog	57
Contact Support	57
Additional Reading	57
Day 4	57
Prerequisites	57
Using Kolla Ansible Quick Start	57
Prepare Kolla Ansible for Use	58
Kolla Ansible Configuration Files	58
Before Making Changes	58
Prepare Kolla Ansible Environment	58
References	60
Next Steps	60
Prerequisites	61
Specify an External Fully Qualified Domain Name for Horizon	61
Determine Public IP	61
Configure an FQDN	61
Apply Configuration Change Using Kolla Ansible	61
Enable SSL Externally, Encrypting Horizon Traffic	61
Modify Kolla Ansible Configuration	62
Configure Root CA Bundle	62
Prepare SSL File	62
Specify SSL Certificate	62
Enable External TLS	62
Reconfigure Cloud using Kolla Ansible	62
Reconfigure Ceph Cluster using Ceph Ansible	63
Procedure	63
Reference	63
How to Enable Elasticsearch and Kibana using Kolla Ansible	63
Introduction	63
Prerequisites	63
Prepare Kolla Ansible	63
Root Access to OpenStack Control Plane	63
How to Enable Central Logging	63
Prevent Root Disk from Filling	64
Reference	64
Principle of Least Privilege	64
Users, Groups, Projects, and Roles	65

Users	65
Groups	65
Projects	65
Roles	65
Updating Software	65
Update Individual Instances	65
Update Glance Images	65
Update Kolla Ansible Images	66
Update Control-Plane Nodes	66
Enabling TLS	66
Security Groups	66
SSH Authentication	66
OpenStack Security Advisor and Further Resources	66
How to Prepare and Use Ceph Ansible	66
Introduction	66
Before Proceeding	67
Prerequisites	67
Root Access to OpenStack Control Plane	67
Preparation	67
Use Watcher to Consolidate your Cloud's Workload	68
Cloud State Before Watcher is Applied	68
How to use Watcher's VM Workload Consolidation Strategy	68
Step 1: Obtain List of Goals	68
Step 2: List Strategies Available for a Goal	68
Step 3: Create Audit Template	69
Step 4: Execute Audit	69
Step 5: Retrieve Action Plan	70
Step 6: Review Action Plan	70
Step 7: Execute Action Plan	71
Cloud State After Watcher is Applied	72
Prerequisites	73
Symptoms of a RabbitMQ Problem	73
How Check your RabbitMQ Cluster's Status	73
List RabbitMQ Queues	73
RabbitMQ and Network Partitions	74
Redeploy RabbitMQ Cluster	74
Prerequisites	74
Prepare Kolla Ansible	74
Root Access to OpenStack Control Plane	74
How to Redeploy RabbitMQ	74
Prerequisites	75
Root Access to OpenStack Control Plane	75
Get Ceph's Status	75
Ceph Log Files	75
Common Issues	76
Clock Skew	76

Confirm Ceph's Health	76
Examine Chrony Logs	76
Addressing Clock Skew	76
References	77
Prerequisites	77
Root Access to OpenStack Control Plane	77
Elasticsearch and Kibana	77
Kolla Ansible Log Locations	77
Determining the Correct Log	77
Determining the Correct Host	78
Viewing Logs	78
Overview of Heat Orchestration	78
How to View Heat Stacks in Horizon	78
Architecture	79
Heat Orchestration Template Components	79
Template Version	79
Description	79
Parameters	79
Resources	79
Output	79
Sample Heat Orchestration Template	79
Attributes	80
Deploying a Heat Template in Horizon	80
Viewing Recently Deployed Stacks in Horizon	82
Additional Resources for OpenStack Heat Service	83
Prerequisites	83
How to Create an Instance Using Terraform	83
Step 1: Prepare Terraform Directory	83
Step 2: Specify Terraform Provider	84
Step 3: Initialize Terraform	84
Step 4: Create OpenStack Application Credentials	84
Step 5: Create Main Terraform File	85
Configure OpenStack Provider	85
Configure Compute Resource	85
Step 6: Create Terraform Plan	86
Step 7: Deploy Terraform Plan	86
View Instance Created by Terraform	87

OpenStack Operator's Manual - Private Cloud Core



Introduction

Welcome to the OpenStack Operator's Manual for your OpenMetal Private Cloud! There are several phases of creating an OpenStack cloud and most of the public documentation focuses on the initial creation of a cloud. With OpenMetal, we provide a full private OpenStack cloud where you start as an **Operator**. This manual, with a few noted exceptions, applies to any OpenStack that has been provisioned to OpenStack.org's [RefStack](#) standard. If you are reading this, it is assumed you have followed the OpenMetal Central Process to provision your cloud already. Visit [OpenMetal Central to sign up or sign in](#). For product details please visit the [Private Cloud](#) page.

Getting Started with the Operator's Manual

Guides are grouped into **Days**, of which there are four. A description of each day and their associated guides follows in the Table of Contents.

Why Days?

Days have been selected to break down the information covered in this manual into categories. Loosely speaking, Day 1 is allocated for initial set up and configuration. Day 2 is for operations and maintenance. Day 3 has to do with scaling a cloud, ensuring your data is backed up, and how to restore it. Day 4 is for more advanced administration topics, introducing how to make general changes to your cloud using Kolla Ansible, and other topics.

Brief Summary of each Day

- Day 1: Initial setup of cloud
- Day 2: Cloud administration and maintenance
- Day 3: Scaling the cloud and disaster recovery
- Day 4: Advanced administration, introducing Kolla Ansible

Operator's Manual Audience

The audience for this manual is experienced Linux System Administrators who are new to OpenStack.

Table of Contents

Day 1

Day 1 generally refers to initial setup, which in our case is the process of getting started with a Private Cloud Core OpenStack Cloud. We introduce you to OpenMetal Central, how to view the assets that comprise your cloud, and how to get support from our staff.

Introduction to OpenMetal Central and Your Private Cloud Core

Introduction

We begin by introducing you to OpenMetal Central and explain how to obtain an overview of your Private Cloud. Next, we detail where to view the assets that comprise your cloud, point you to existing documentation, and how to submit and view support requests.

A Brief Overview of Your OpenMetal Private Cloud

OpenMetal Private Clouds are deployed with OpenStack to three bare metal servers. These three servers comprise the Private Cloud Core. To OpenStack, these three servers are considered the control plane. Private Clouds are deployed with Ceph, providing your cloud with shared storage.

How to View Your Hardware Assets

To view your assets, log in to the homepage for [OpenMetal Central](#).

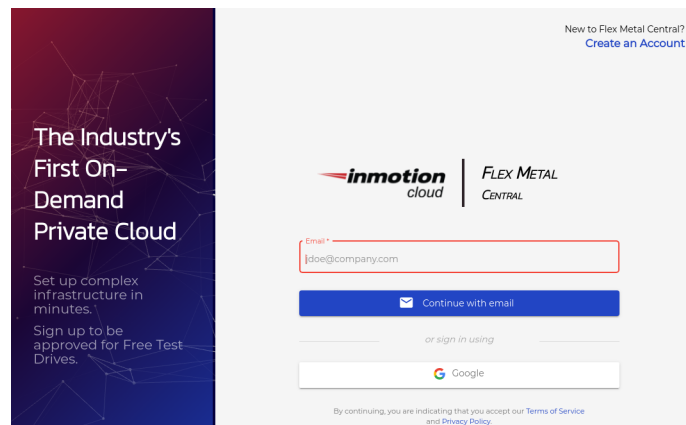


Figure 1: Central Login Page for OpenMetal Central

Figure 2 is the Homepage for OpenMetal Central. The Homepage provides access to your Cloud Management Dashboard. Click on the link **Manage** to access your Cloud Management Dashboard.

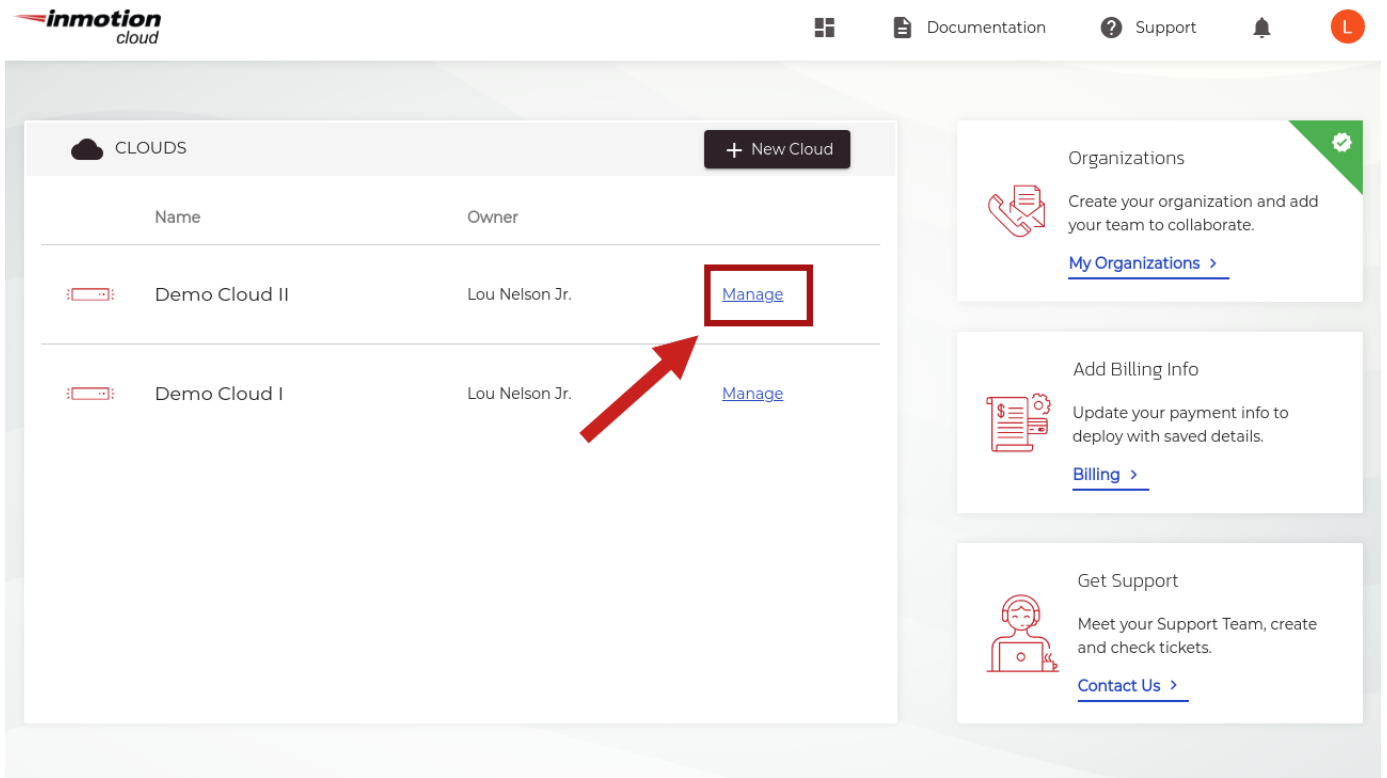


Figure 2: OpenMetal Central Homepage

On the left side of the Cloud Management Dashboard is the Assets Button as shown in Figure 3. Click there to go to the Assets page. This page contains a list of assets included with your Private Cloud Deployment. These include your Hardware Control Plane Nodes and IP blocks for Inventory and Provider IP addresses.

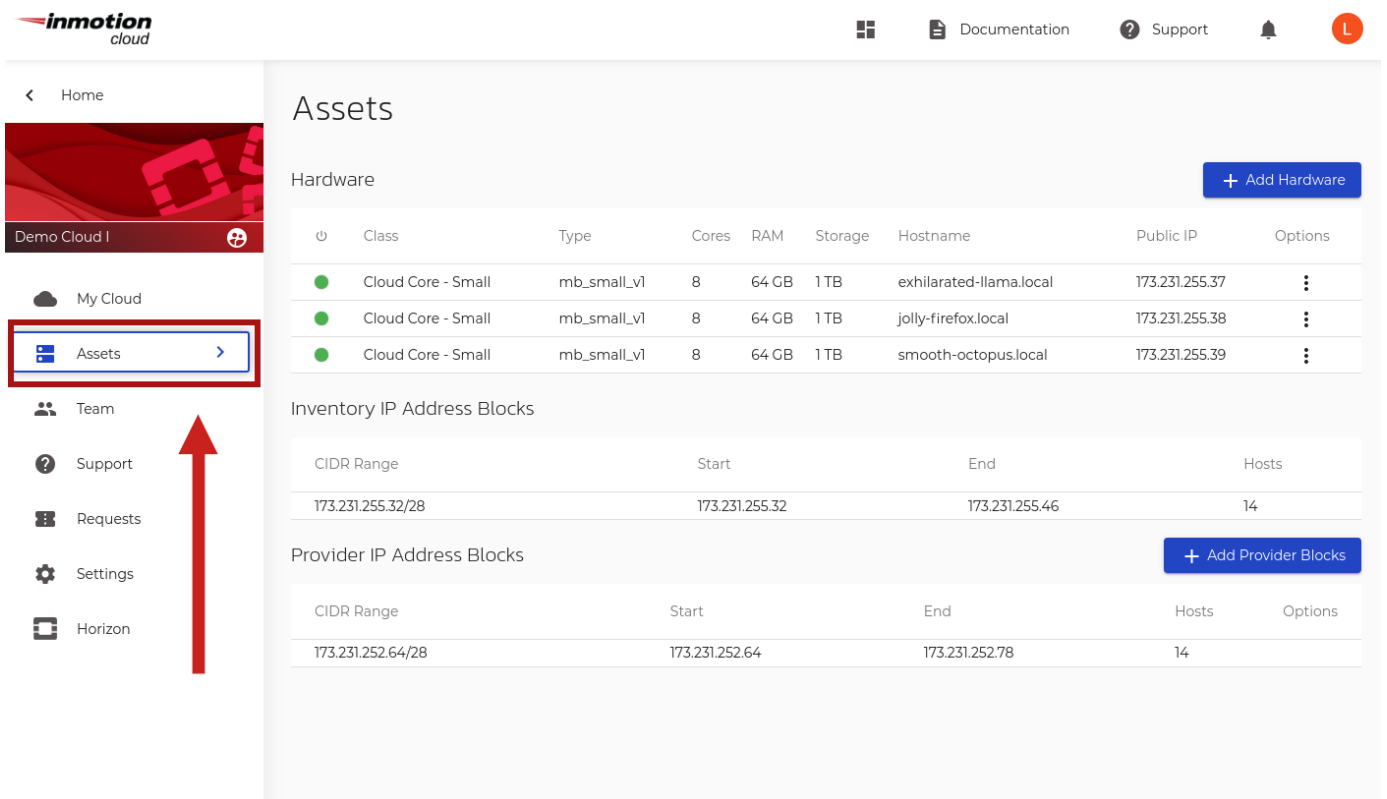


Figure 3: Assets Page of Cloud Management Dashboard for OpenMetal

The following example is a list of assets in a Demo Private Cloud. Your Private Cloud can have different hardware based on the options you have selected in your deployment:

Example list:

- 3 Cloud Core **mb_small_v1** Control Plane Nodes
- Inventory IP Address Blocks
- Provider IP Address Blocks

Note: With our Private Clouds, OpenStack is deployed with three hyper-converged control plane nodes.

You can access your Control Plane Nodes directly through SSH as the root user. This access is done through the SSH keys you provided during your Private Cloud Deployment.

How to Get Support

In OpenMetal Central, the two ways of getting support for your Private Cloud are through support documentation and filing support requests.

How to Access OpenMetal Documentation

Documentation links are found in the OpenMetal Central Homepage as well as the Cloud Management Dashboard. To access documentation, click **Documentation** at the top right-hand side of the page as shown in figure 4.

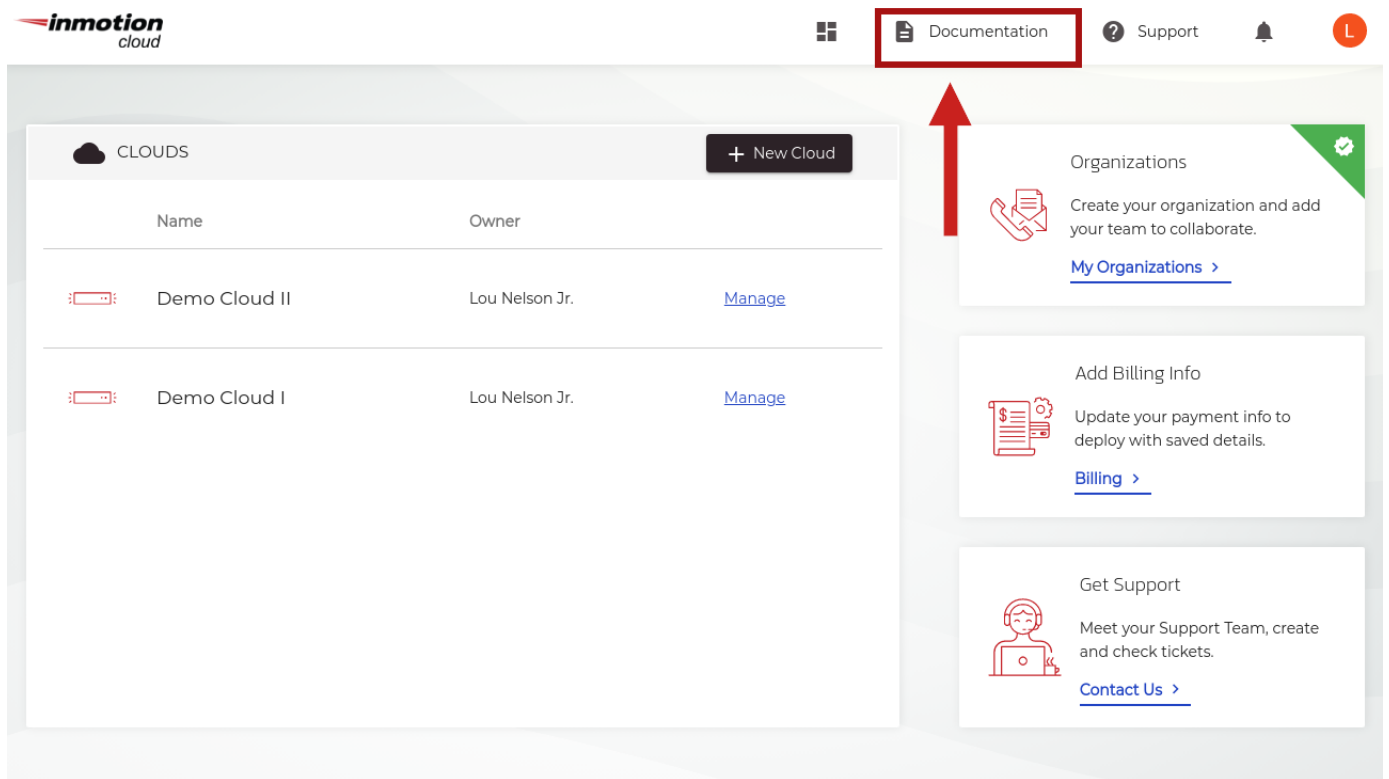


Figure 4: OpenMetal Documentation link found within OpenMetal Central Homepage.

Figure 5 is the index page for all documentation within OpenMetal. All documentation relating to solving common issues, configuration, as well as deployment can be found on this page.

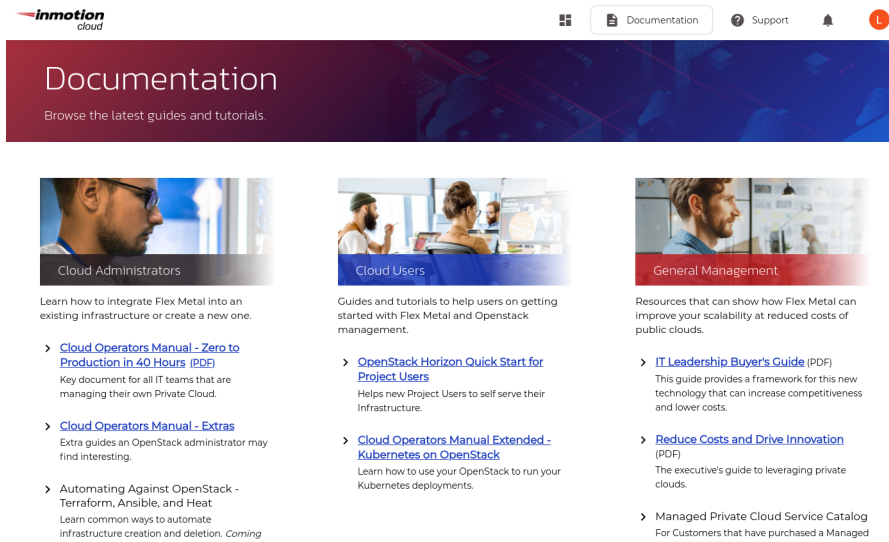


Figure 5: OpenMetal Documentation Index Page

How to Submit a Support Ticket

Support links are found in the top right of both the Cloud Management Dashboard and the OpenMetal Central Homepage. To access the support center click the link marked **Support**.

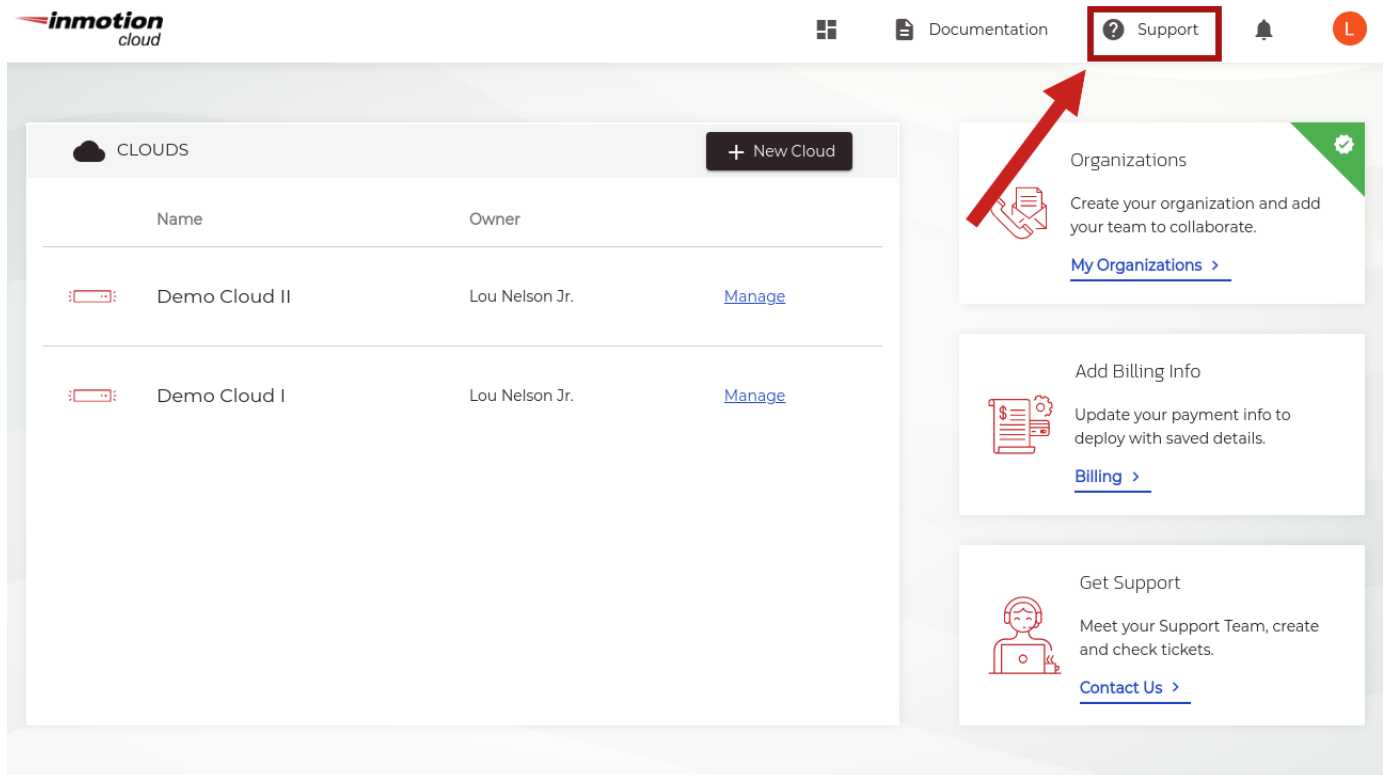


Figure 6: OpenMetal Central Homepage support links.

Support links are found in the top right of both the Cloud Management Dashboard and the OpenMetal Central Homepage. To access the support center click the link marked **Support**.

Figure 7 outlines the support section of the Cloud Management Dashboard. Click **New Request** to start a new support ticket as shown in figure 7. You also have the option to contact individual team members by clicking the profile icon next to their name as shown in figure 8.

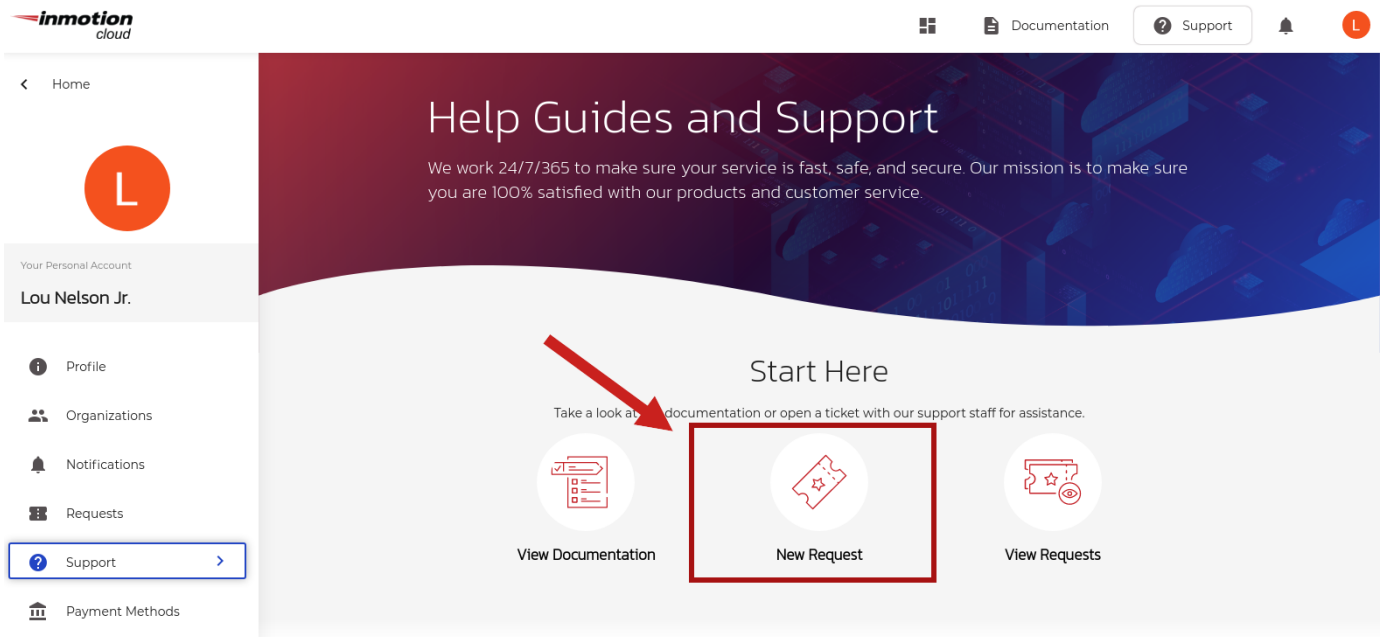


Figure 7: The Support section of your Cloud Management Dashboard.

Figure 8 outlines how you can reach individual members of our support team. Click the profile icon of the individual you wish to contact to submit a support request to that individual.

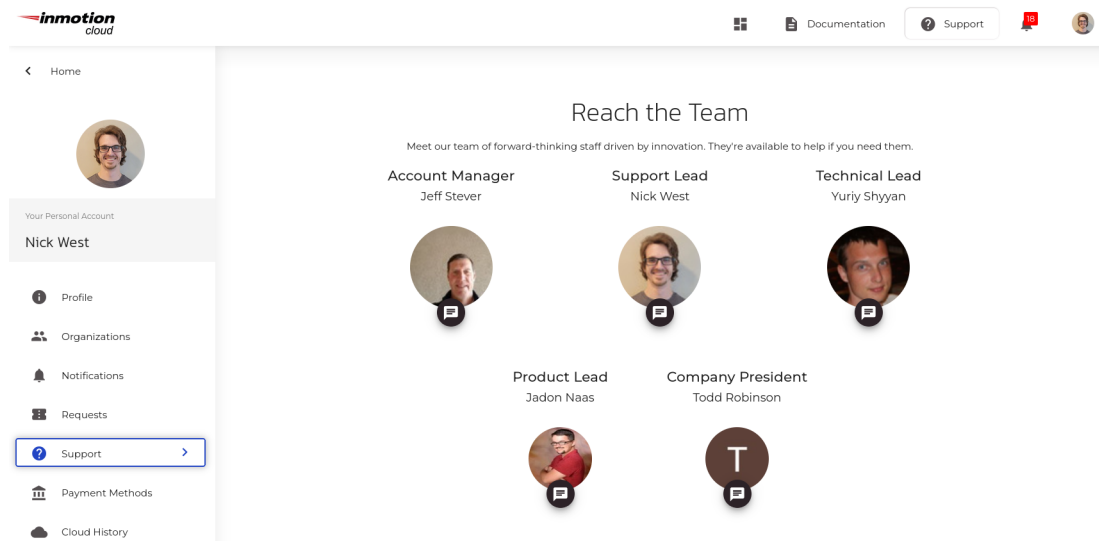


Figure 8: The listed members of the OpenMetal Support Team.

Click **New Request** to bring up a support ticket. After entering the subject, you are presented with options for the type and priority of the ticket you wish to send. Select the appropriate options and fill out the description of your problem. Then you can send the ticket.

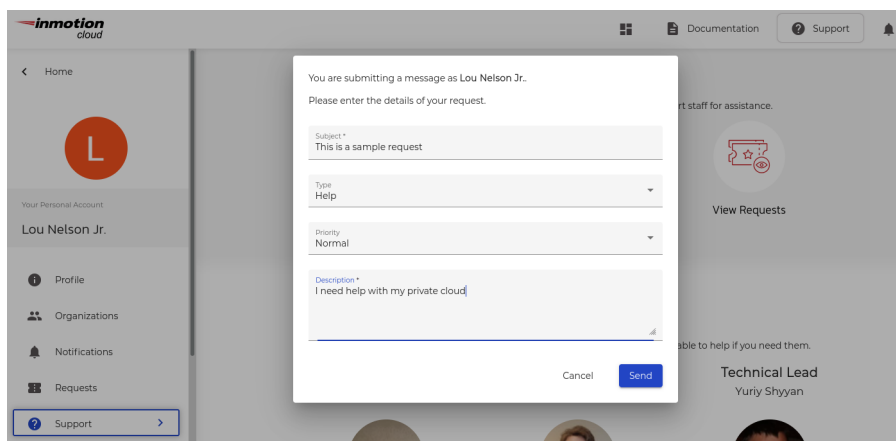


Figure 9: Sample OpenMetal Test Support Request

To view the details of a support request, click the highlighted subject link next to the request you which to view. You will be able to see all responses to your request as shown in figure 10.

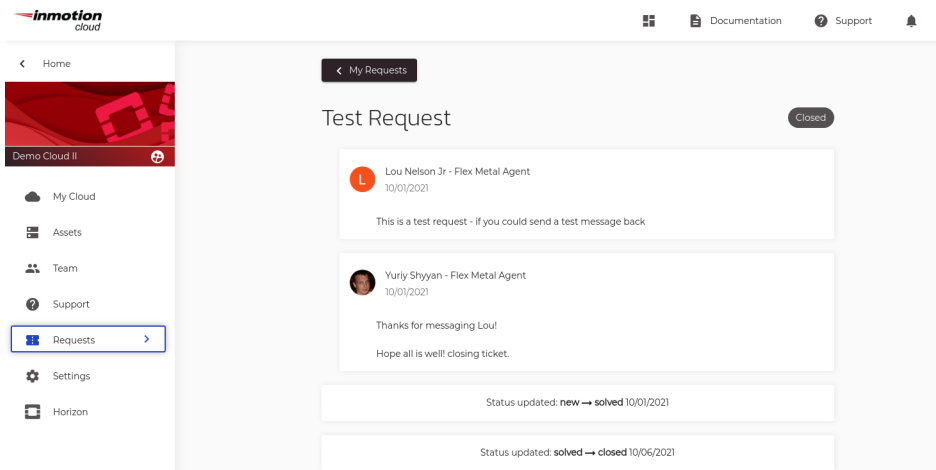


Figure 10: Sample OpenMetal Test Support Request Reply Thread

To view submitted requests, access your Cloud Management Dashboard and click the **Requests** section of your control panel. You are now able to see all request tickets submitted under your user account. If you are the administrator of the Private Cloud, you will be able to see all support requests regarding your Private Cloud.

Getting Started with OpenStack Horizon

Introduction

To get started with using your cloud, we introduce Horizon, OpenStack’s dashboard. It is accessible through a web browser and allows a user to interact with the cloud. As an administrator, most of the cloud can be managed this way.

How to Log in to Horizon

The Horizon administrator password is present in a file within the cloud itself. This section walks you through the steps required to obtain this password.

Step 1: Log in to OpenMetal Central

To get started, navigate to [OpenMetal Central](#) and log in.

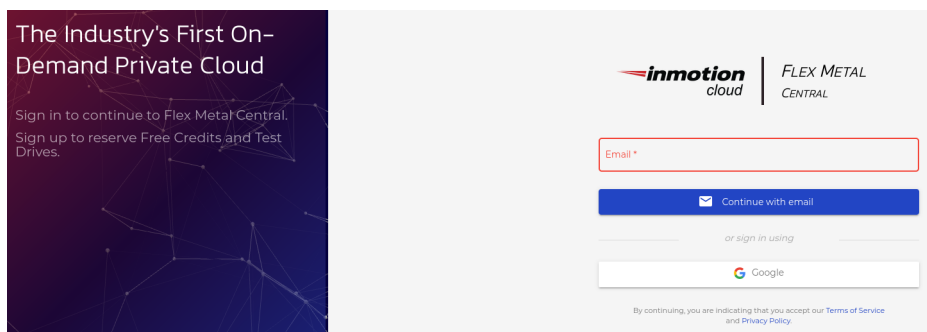


Figure 1: OpenMetal Central Login Page

Step 2: Navigate to Cloud’s Details Page

Click **Manage** to the right for the cloud you’re working with to load this cloud’s details page.

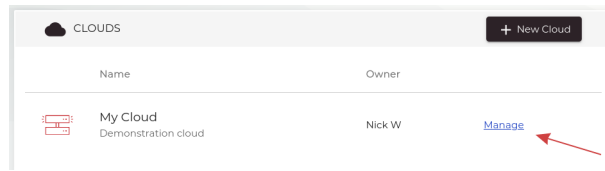


Figure 2: Cloud List

Step 3: Load Horizon URL

Next, navigate to the **Horizon** link within the left sidebar located as the last item in the list.

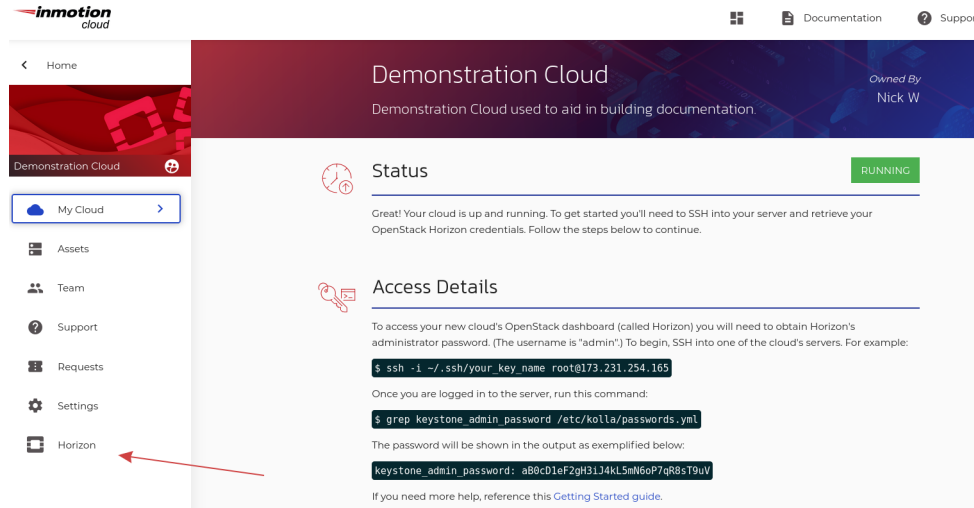


Figure 3: Horizon URL Location

Step 4: SSH into a Hardware Node, Obtain Horizon Password

During cloud creation, your SSH public key was added to each hardware node. With this key, you are able to log in as root using SSH. To SSH in, ensure you have one node's IP address and the private key on the system you use to access your cloud. To find your cloud's hardware node IP addresses, click **Assets** on the cloud's details page.

Hardware									+ Add Hardware
	Class	Type	Cores	RAM	Storage	Hostname	Public IP	Options	
●	Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	relaxed-flamingo.local	173.231.254.165		⋮
●	Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	focused-capbara.local	173.231.254.166		⋮
●	Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	lovely-ladybug.local	173.231.254.167		⋮

Figure 4: Assets Page

The IP address for each node is listed under the **Public IP** column. You can SSH into any of the nodes for this step. Select an IP from this list, then use SSH to log in.

Requirements to SSH into a node

To SSH into a hardware node, ensure these requirements are met:

- Username: root
- Authentication: SSH key pair
- IP address of a hardware node

As an example, we demonstrate using SSH to log in to the first hardware node, located by 173.231.254.165. The SSH key file for this example is `~/.ssh/id_rsa`. Ensure you specify the appropriate key and IP address for this step.

For example:

```
$ ssh -i ~/.ssh/id_rsa root@173.231.254.165
```

Once logged in, search for `keystone_admin_password` inside of `/etc/kolla/passwords.yml`.

For example:

```
# grep keystone_admin_password /etc/kolla/passwords.yml  
keystone_admin_password: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Step 5: Log in to Horizon

You can now log in to Horizon using the credentials obtained from the previous section. The username for the Horizon administrator account is **admin**.

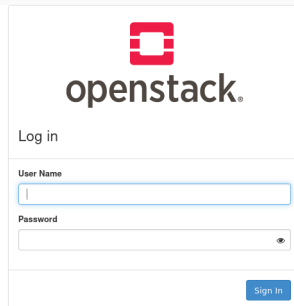


Figure 5: Horizon Login Page

When you log in to Horizon, your dashboard appears similar to the following:

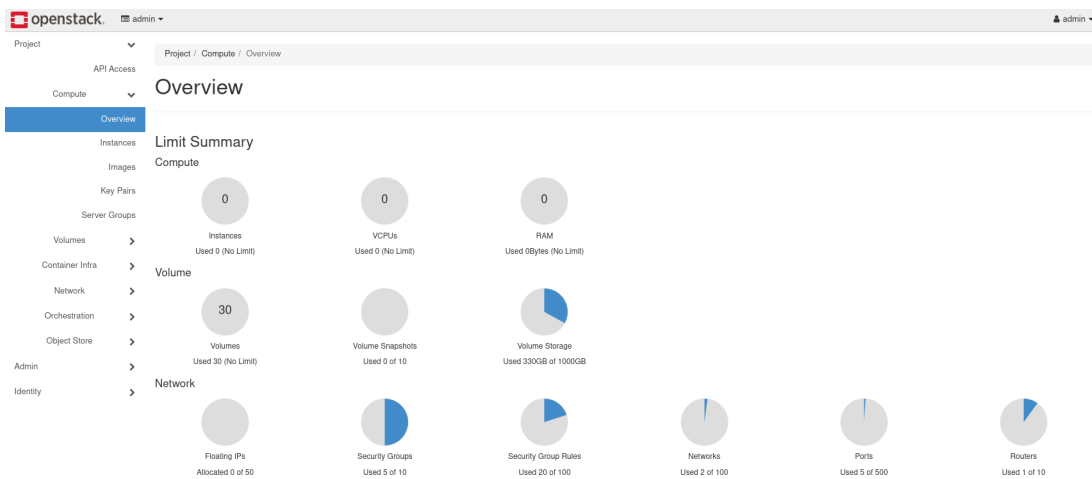


Figure 6: Horizon Dashboard

Create a User and Project in OpenStack Horizon

Introduction

In OpenStack, the cloud is divided through the use of projects. Projects have associated with them users, who have differing levels of access, defined by roles. An administrator defines resource limits per project by modifying quotas. In this guide, we explain how to create a project and associate a user with it. Additionally, we explain how project quotas can be adjusted.

Quick Overview: Project, Admin, and Identity Tabs

Upon initially logging in to Horizon as **admin**, you see on the left tabs that group areas together. The three primary tabs seen are **Project**, **Admin**, and **Identity**. Only a user with the **admin** role sees the **Admin** tab. Administrative functions, such as live migrating an instance, occurs through the **Admin** section. Project users with the **member** role see only the **Project** and **Identity** tabs and can only perform actions within their specific project.

Table of Contents

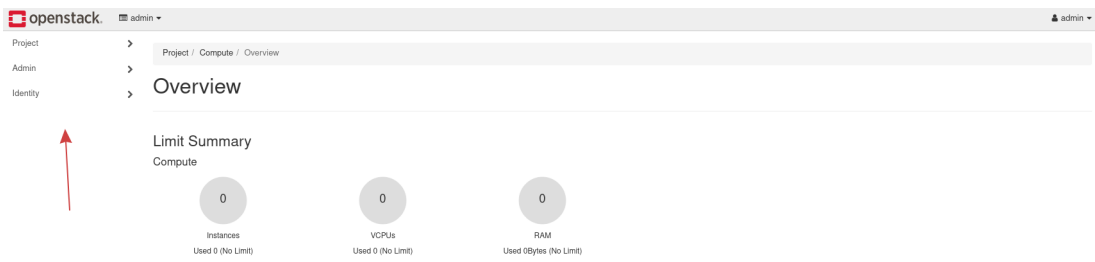


Figure 1: Project, Admin, Identity tabs

How to Create a Project

When you first log in to Horizon as the administrator account, you find yourself in the project called **admin**. You can see this by looking at the very top and near the left of the screen.

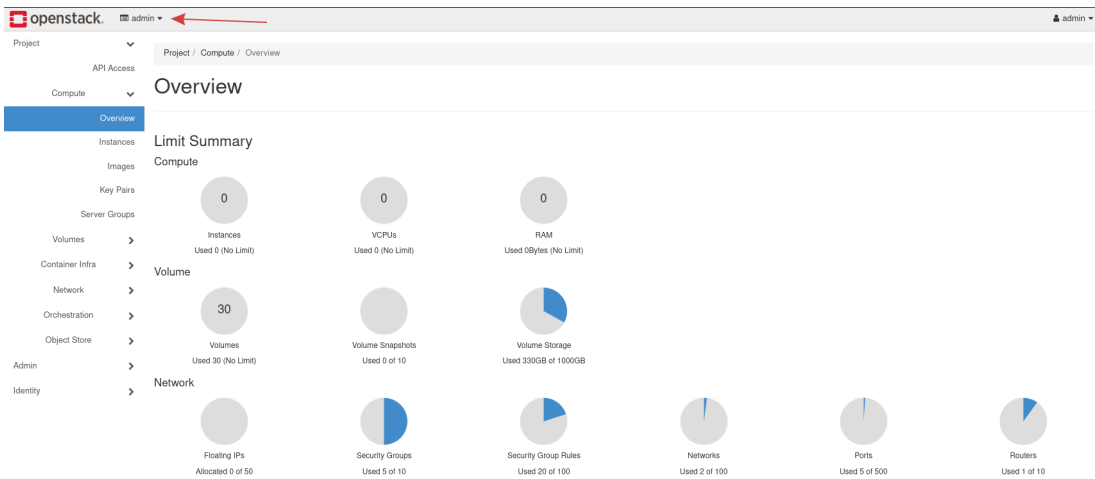


Figure 2: Admin Project Overview

The left most column contains links to different areas in which different tasks can be performed. There are three primary groupings: **Project**, **Admin**, and **Identity**. Only an account with the administrator role can see the **Admin** tab. The **Identity** tab is used to create a project.

Create your First Project

To create your first project, navigate to **Identity -> Projects**.

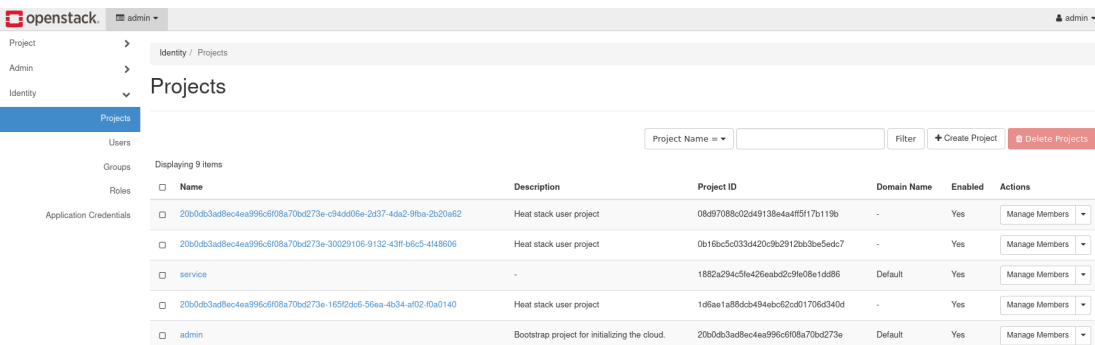


Figure 3: Project Listing

Several projects already exist, including the **admin** project. These projects are deployed by default and generally should not be modified.

Pull up the form to create a project by navigating to the **Create Project** button near the top right.

Figure 4: Create Project Form

Under the **Name** field, specify a name for the project. This example project is called **Development**. There are tabs for adding Project Members and Project Groups, but these are not covered for this demonstration. This guide later explains how to create a user and attach it to this project. Click **Create Project** to finish creating the first project.

Once created, the project appears in the Project Listing page.

<input type="checkbox"/> Development	Development project.	235b08665924b1c83a901994999c2ff	Default	Yes	Manage Members ▾
--------------------------------------	----------------------	---------------------------------	---------	-----	------------------

Figure 5: Development Project Listed

Project Quotas

While in the project listing page, you can view and adjust quotas for this project as the **admin** user. Quotas are limits on resources, like the number of instances for example.

To view the quotas for this project while in **Identity -> Projects** tab, find the drop down to the right with the first option being **Manage Members**. From this drop down, click **Modify Quotas** to view the default quota values.

<input type="checkbox"/> Development	Development project.	235b08665924b1c83a901994999c2ff	Default	Yes	Manage Members ▾
<input type="checkbox"/> 20b0db3ad8ac4ea996c6f08a70bd273a-4069c75a-9eed-476b-bd74-e28bd83	Heat stack user project	4a1d5e7254e46f0a546ed4e9d1e0ac	-	Yes	<ul style="list-style-type: none"> Manage Members Modify Groups Edit Project View Usage Modify Quotas Delete Project
<input type="checkbox"/> 20b0db3ad8ac4ea996c6f08a70bd273a-6057c24f-20b1-42ff-8ff2-5b153cc	Heat stack user project	5c5e8aa287bc46a282996ec0c90c5d3a	-	Yes	
<input type="checkbox"/> 20b0db3ad8ac4ea996c6f08a70bd273a-3b12116a-9bf1-4ca0-96a2-d882e9c	Heat stack user project	88e59bbf1728403c91adb7f967c3d89a6	-	Yes	

Figure 6: Modify Quotas

A form appears with several tabs and you are presented with the quotas for the Compute service. Quotas exist for the Volume and Network services as well.

Figure 7: Edit Quotas

You may want to adjust the parameters in this form depending on your workload.

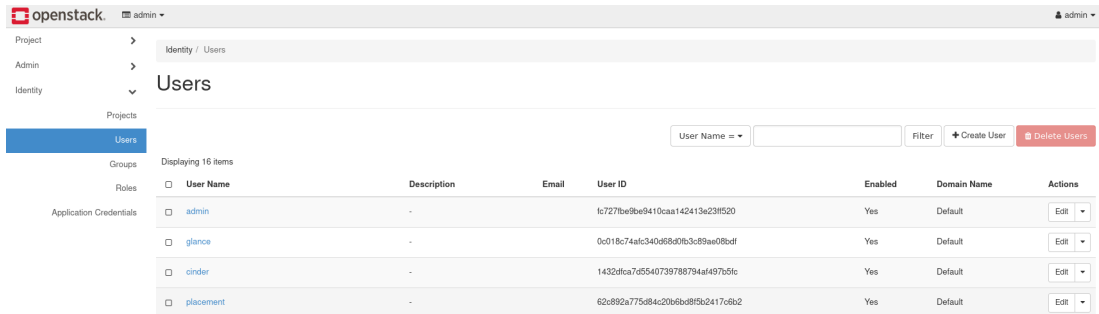
Note! – We have seen issues appears where an item, like an instance, cannot be created. This is often related to a hard limit set for a specific quota. Typically the error message received indicates the quota has been reached.

Note! – Setting a value to -1 means that quota is unlimited.

How to Create a User and Associate with Project

With your first project created, you can now create and associate a user with it. The intention of this guide is to have you create a user, associate it with the project created earlier, and then log out of Horizon as **admin**, and log back in with the new user.

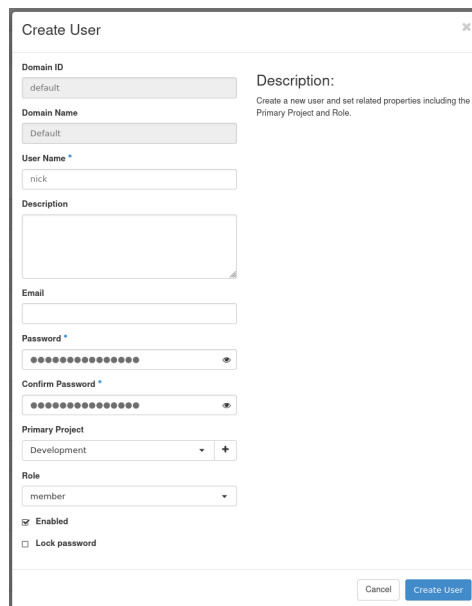
To create a user, first navigate as **admin** to **Identity -> Users**. By default, there are several users already listed, and this is expected. These are created during cloud deployment and should generally not be modified.



User Name	Description	Email	User ID	Enabled	Domain Name	Actions
admin	-		fc727be9be9410caa142413e23ff520	Yes	Default	Edit
glance	-		0c018c74alc340c68d0b3c89ae08bdf	Yes	Default	Edit
cinder	-		1432dfca7d5540739788794af497b5fc	Yes	Default	Edit
placement	-		62c892a775c84c20b6bd8f5b2417c6b2	Yes	Default	Edit

Figure 8: User Listing

Next, load the form to create a user by navigating to **Create User** in the upper right of the screen.



Create User

Domain ID: default

Domain Name: Default

User Name: nick

Description: Create a new user and set related properties including the Primary Project and Role.

Email:

Password: [masked]

Confirm Password: [masked]

Primary Project: Development

Role: member

Enabled

Lock password

Buttons: Cancel, Create User

Figure 9: Create User Form

For this example, we set values for **User Name**, **Password**, **Primary Project**, and **Role**.

- **User Name:** Specify your user here
- **Password:** Set a unique, randomly generated password
- **Email:** Optional, but is useful for password resets
- **Primary Project:** Choose the project created earlier
- **Role:** This example selects the `member` role

For **Role**, there are several options, depending on the level of access required. The default OpenStack roles are **reader**, **member**, and **admin**. Additional roles exist in the drop down, which is expected. This example sets the role **member** for this user. For more about roles in OpenStack, see the latest [Keystone Default Roles documentation](#).

Press **Create User** to create the user.

Next, log out of Horizon as **admin**, and log back in with your new user. Upon logging back in you are by default in the newly created project. You can see the project you are currently in at the top left and your user can be seen at the top right of Horizon.



Figure 10: New User Login

For the rest of this guide, we assume you are working out of the newly created project and using the user associated with it.

Reference

[OpenStack Victoria Horizon Administrator Guide](#)

Manage and Upload Images in OpenStack Horizon

Introduction

This guide provides instructions for how to upload an image into OpenStack as well as create images out of an existing instance. Images contain a bootable operating system that is used to create instances. Within your OpenMetal Cloud, there are several different images that are readily available including CentOS, Debian, Fedora, and Ubuntu. In addition to this, you have the option to upload images from other sources or create your images. Images can either be built from ISO or images can be created out of snapshots of instances. This guide will walk you through how to upload images to Glance through Horizon and how to create an image from an instance snapshot.

Managing Images

The primary tool for managing images is OpenStack's Glance service. Glance uses Ceph to store images instead of the local file system. You can upload images through Glance and uploaded images cannot be changed. For further information on how to create and modify virtual images within OpenStack, view the [Virtual Machine Image Guide](#).

To access images from within your Horizon Dashboard, navigate to the Projects tab. Within the projects tab, select Compute and you are presented with the option to select Images. This tab contains a list of all your images within OpenStack.

Name	Type	Status	Visibility	Protected	Disk Format	Size	Launch
Amphora (x64-haproxy-ubuntu-focal)	Image	Active	Public	No	QCOW2	359.97 MB	Launch
CentOS 7 (el7-x86_64)	Image	Active	Public	No	QCOW2	847.81 MB	Launch
CentOS 8 (el8-x86_64)	Image	Active	Public	No	QCOW2	1.22 GB	Launch

Figure 1: Image List Within Horizon Dashboard

Uploading Images

Images can be uploaded through your Horizon dashboard. With our configuration, the recommended format for images is **QCOW2 - QEMU emulator**. QCOW2 is the most common format for Linux KVM, expands dynamically, and supports [copy on write](#). In this section, we explain how to upload images through Horizon.

In order to upload an image on Horizon, you must first have the image locally on your machine. In this example, we demonstrate uploading the CirrOS image. To download this image visit [CirrOS Latest Download](#).

After downloading the image locally to your machine, navigate in your Horizon dashboard to **Project -> Compute -> Images**, where you are presented with a list of options for managing images. To begin uploading this image, click the **Create Image** button near the top right.

Figure 2: Creating an Image Within Horizon Dashboard

Figure 3: Create Image Form

For this demonstration, we enter in values for the following fields:

- **Image Name:** Name of the image
- **Image Description:** Optional description of the image
- **File:** The source file on your machine
- **Format:** - For this example, we are using QCOW2 - QEMU Emulator

Fill out the details as needed and submit the form. It may take some time to complete uploading the image.

Project / Compute / Images

Images

Search: Cirros + Create Image Delete Images

Displaying 1 item

Owner	Name	Type	Status	Visibility	Protected	Disk Format	Size	
admin	Cirros	Image	Active	Shared	No	QCOW2	15.55 MB	Launch

Figure 3: Cirros Image Listed in Horizon Dashboard

Create Images From Running Instances

To create an image from a running instance, you must first create a snapshot of your instance. Once created, a snapshot is a usable image within Glance that can create instances.

To create a snapshot in Horizon, navigate to **Project -> Compute -> Instances** and locate the **Create Snapshot** option from the listed instance's drop down menu. In the form that appears, enter a name for your snapshot.

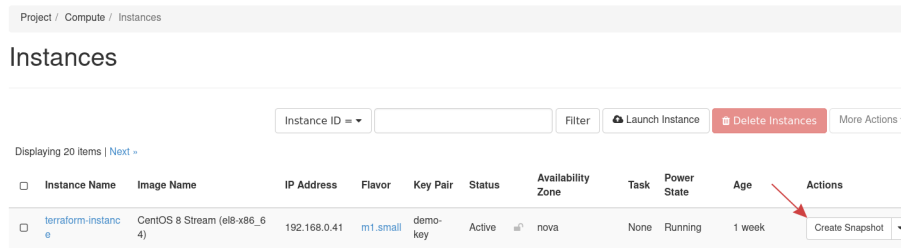


Figure 4: Create Snapshot Button Within Horizon Dashboard

Once your snapshot has been created, the snapshot name is added to your list of images. You can use this image to launch new instances.

Reference

[OpenStack Victoria Glance Documentation](#)

How to Create an Instance in OpenStack Horizon

Introduction

With OpenStack, instances, or virtual machines, play a large role in a cloud's workload. OpenStack provides a way to create and manage instances with its compute service, called **Nova**. In this guide, we cover the preparatory steps required to create an instance, including setting up a private network and router, creating a security group, and how to add an SSH key pair. Then, we explain how to create an instance using Horizon.

Networking

In this section, we explain how to create a private network and router. The instance created later in this guide is created on this private network. The router is created so the private network can be connected to your cloud's public network, allowing you to assign a floating IP address to it, making the instance accessible over the Internet. Next, we cover where security groups can be found and managed. In these next few sections, we explain how to create the components with a screenshot then follow up with an explanation of the details to fill out.

Create a Private Network

To create a private network, begin by navigating to **Project -> Network -> Networks**.

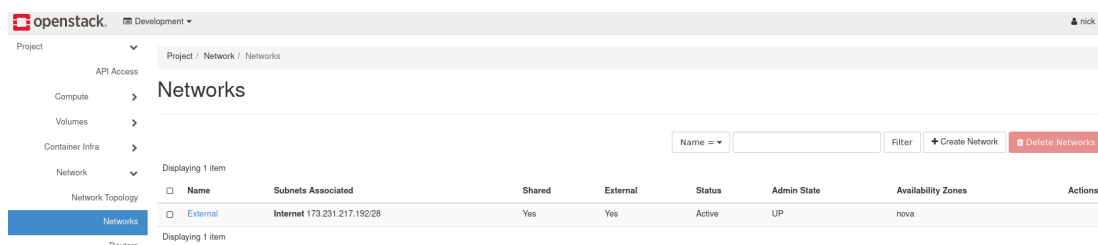


Figure 1: Networks in Horizon

Load the form to create a network, by navigating to **Create Network** near the top right.

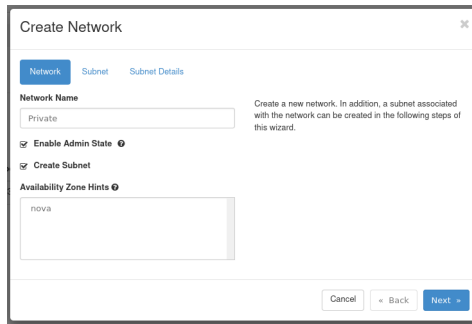


Figure 2: Network Tab

- **Network Name:** Set a name for the network. This example is called **Private**.
- **Enable Admin State:** Leave this checked to enable the network.
- **Create Subnet:** Leave this checked to create a subnet.
- **Availability Zone Hints:** Leave this option as default.

Next, move on to the **Subnet** tab of this form.

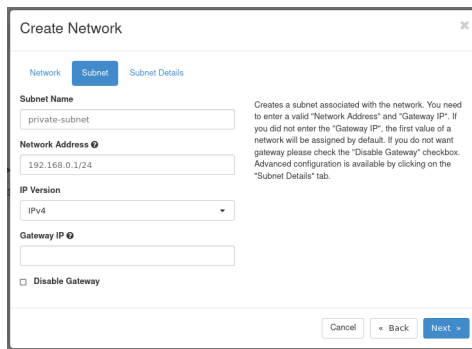


Figure 3: Subnet Tab

- **Subnet Name:** Set a name for the subnet. This example subnet is called **private-subnet**.
- **Network Address:** Select a private network range. For example: 192.168.0.1/24
- **IP Version:** Leave this as IPv4.
- **Gateway IP:** This is optional. If unset, a gateway IP is selected automatically.
- **Disable Gateway:** Leave this unchecked.

Next, move on to the **Subnet Details** tab of this form.

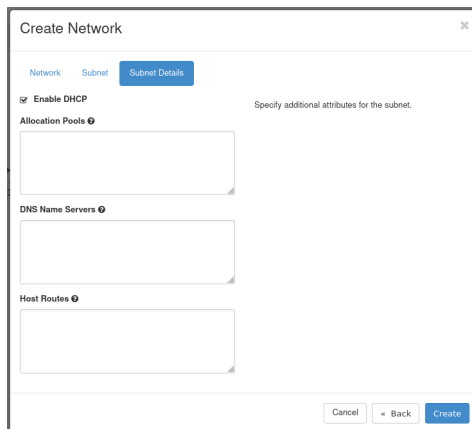


Figure 4: Subnet Details Tab

- **Enable DHCP:** Leave this option checked.

- **Allocation Pools:** Optional, can specify the range from which IPs are selected.
- **DNS Name Servers:** Optional. Specify any DNS name servers here.
- **Host Routes:** Optional. Specify any host routes here.

Click **Create** to create the network. Once created, it appears in the list of networks.

Displaying 2 items

<input type="checkbox"/>	Name	Subnets Associated	Shared	External	Status	Admin State	Availability Zones	Actions
<input type="checkbox"/>	Private	private-subnet 192.168.0.0/24	No	No	Active	UP	nova	Edit Network
<input type="checkbox"/>	External	Internet 173.231.217.192/28	Yes	Yes	Active	UP	nova	

Figure 5: Network Listing

Create a Router

You next need to create a router to bridge the connection between the private network and the public network. The public network is called **External**.

To create a router, begin by navigating to **Project -> Network -> Routers**.

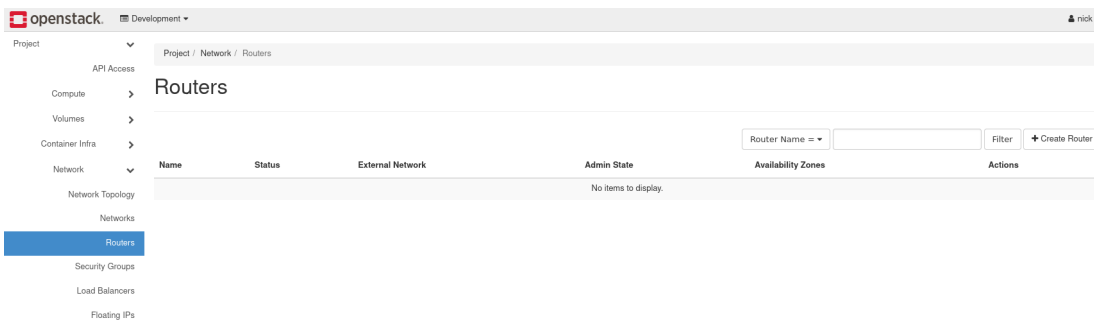


Figure 6: Router Listing

Load the form to create a router by navigating to **Create Router** near the top right.

Create Router

Router Name: Description:

Enable Admin State

External Network:

Availability Zone Hints:

Figure 7: Create a Router

- **Router Name:** Set a name for the router here. This example router is called **Router**.
- **Enable Admin State:** Leave this checked to enable the router.
- **External Network:** Choose the network **External**.
- **Availability Zone Hints:** Leave this as the default.

Once complete, create the router by pressing **Create Router**.

Connect Router to Private Network

Next, connect the router to the private network, by attaching an interface. Performing this step allows network communication between the **Private** and **External** networks.

To attach an interface to the router, first navigate to the list of routers and locate the one previously created.

Displaying 1 item

<input type="checkbox"/>	Name	Status	External Network	Admin State	Availability Zones	Actions
<input type="checkbox"/>	Router	Active	External	UP	nova	Clear Gateway

Figure 8: Router List

Click the name of the router to access its details page. This is where the interface is attached. There are three tabs: **Overview**, **Interfaces**, and **Static Routes**. To attach an interface, navigate to the **Interfaces** tab then load the form to attach an interface by clicking **Add Interface** near the top right.

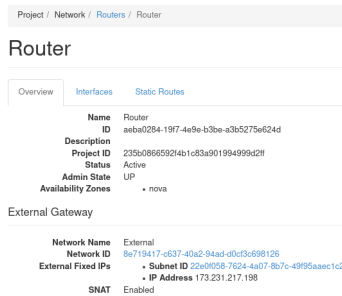


Figure 9: Router Details

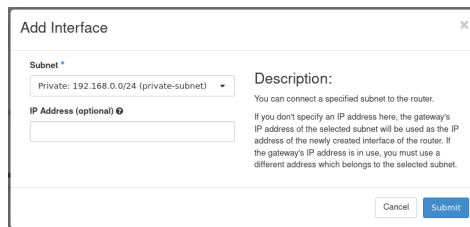


Figure 10: Attach Interface

- **Subnet:** Select the appropriate subnet. In this example, we choose the **private-subnet**.
 - **IP Address:** Optional. Specify an IP if required, otherwise one is selected automatically.
- Press **Submit** to attach the **Private** network to this router. The interface is then attached and now listed.

Displaying 1 item						
<input type="checkbox"/>	Name	Fixed IPs	Status	Type	Admin State	Actions
<input type="checkbox"/>	(1c8951ac-9b7)	• 192.168.0.1	Active		UP	Delete Interface

Figure 11: Interface Listing

View Network Topology

Should you want to visually see the network topology for your cloud, navigate to **Project -> Network -> Network Topology**.

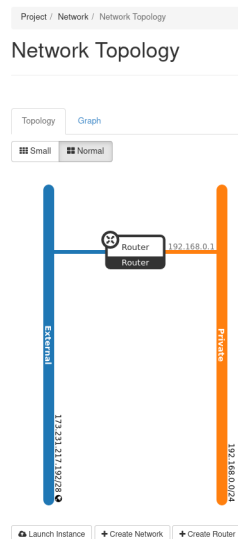


Figure 12: Network Topology

This figure indicates the **External** network is connected to the **Private** network through the router called **Router**.

Security Groups

Security groups allow control of network traffic to and from instances. For example, port 22 can be opened for SSH for a single IP or a range of IPs. In this demonstration, we create a security group for SSH access. This security group is then applied to the instance we later create.

To view and manage security groups, navigate to **Project -> Network -> Security Groups**.

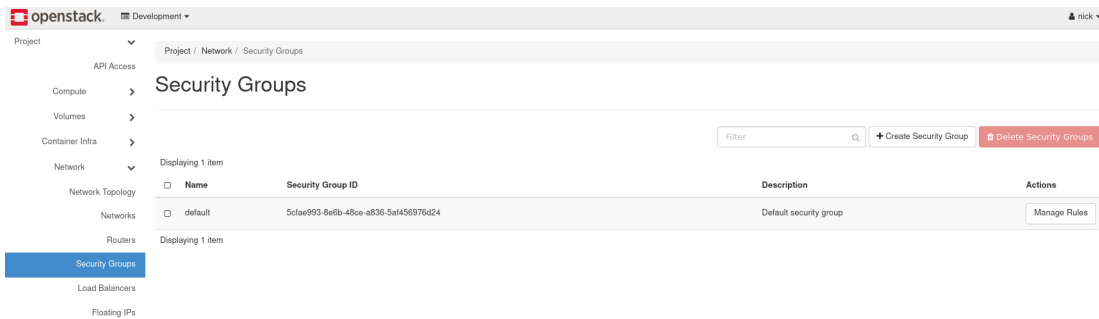


Figure 13: Security Groups

In this cloud, there exists a single security group called **default**. This security group restricts all incoming (ingress) network traffic and allows all outgoing (egress) network traffic. When an instance is created, this security group is applied by default. To allow the network traffic your instance requires, only open ports as required to just the needed IP ranges.

Create an SSH Security Group

To create a security group for SSH, load the form by navigating to **Create Security Group** near the top right.

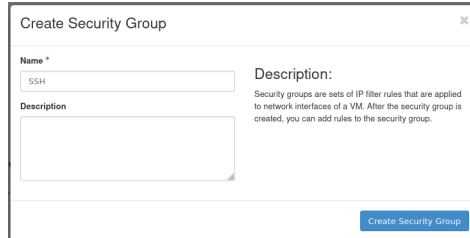


Figure 14: Create Security Group Form

- **Name:** Specify a name for the security group. This example security group is called **SSH**.
- **Description:** Optional. Describe the security group if applicable.

Add Rule to SSH Security Group

After creating the SSH security group, we need to add a rule allowing SSH traffic. This example demonstrates allowing SSH traffic from the first hardware node in this cloud to this instance.

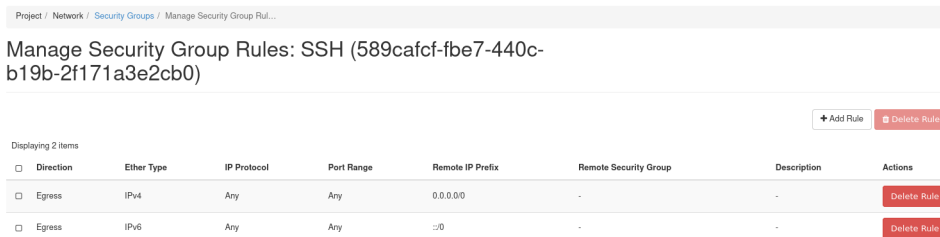


Figure 15: Manage Security Group Rules

To add a rule, load the form by navigating to **Add Rule** near the top right.

To follow this example, obtain the IP address of the first hardware node of your cloud. You can find this using [OpenMetal Central](#) under your cloud's [Assets Page](#). To be consistent, this guide assumes you are working with the first hardware node's IP address and the remaining instruction is created with that understanding.

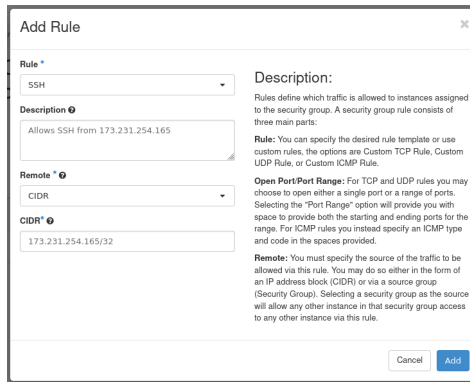


Figure 16: Add SSH Rule

- **Rule:** Select **SSH**. When adding rules you can choose from predefined options. In this case, we choose the **SSH** rule from the first drop down.
- **Description:** Optional. Provide a description of the rule.
- **Remote:** Select **CIDR**.
- **CIDR:** Specify the IP address of your first hardware node.

Press **Add** to add this rule to the security group. This concludes creating the security group.

How to Create your First Instance

With the previous steps complete, almost everything is in place to create your first instance.

Prerequisites

SSH Public Key

An SSH public key is required to access an instance over SSH. This key is injected into the instance when created. An SSH key cannot be added to an already running instance.

For this guide, we are arranging it so this instance can be accessed over SSH from one of the cloud's hardware nodes. Due to this, you must create an SSH key pair in one of the hardware nodes. The public portion of that key pair is associated with the instance created later in this guide. To learn how to create this key pair, see the supplementary guide: [Create SSH Key Pair for an OpenStack Control Plane Node](#).

Operating System Image

Several operating system images are available with which to create instances. To view available options, navigate to **Project -> Compute -> Images**. To upload your own images, see [How to Upload Operating System Images using Horizon](#).

Create your First Instance

To create the first instance, begin by navigating to **Project -> Compute -> Instances**.

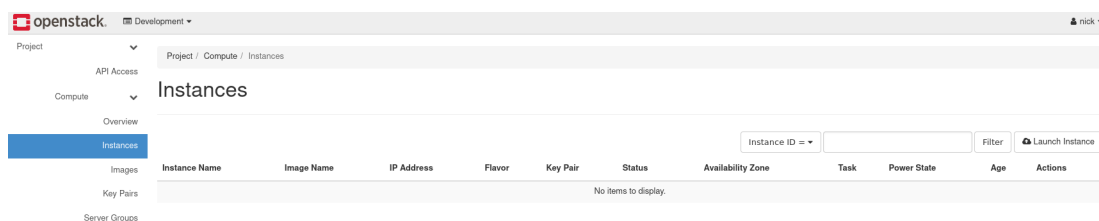


Figure 17: Instances

Pull up the form to create an instance by navigating to **Launch Instance** near the top right.

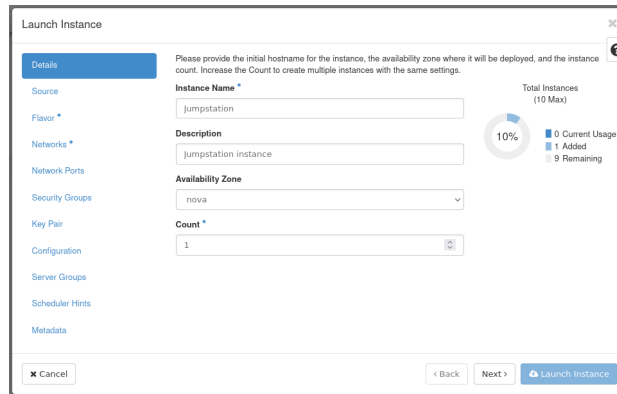


Figure 18: Instance Details

- **Instance Name:** Set a name for the instance. This example instance is called **Jumpstation**.
 - **Description:** Optional. Set a description if this applies.
 - **Availability Zone:** Leave as the default, which is **nova**.
 - **Count:** Controls the number of instances spawned. This example creates a single instance.
- Next, move to the **Source** tab allowing you to specify an operating system image.

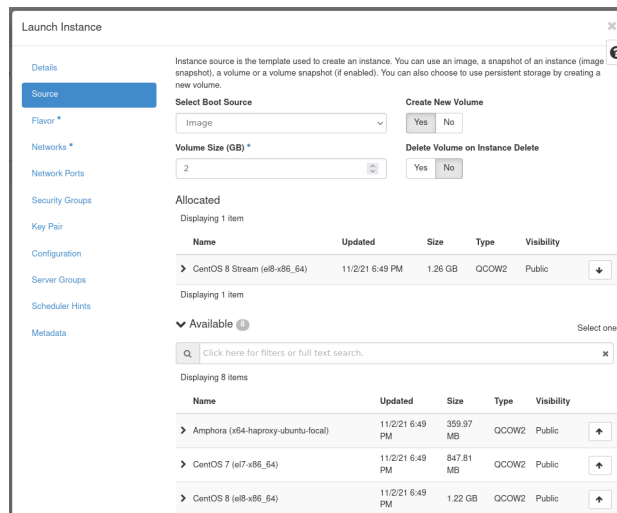


Figure 19: Instance Source

- **Select Boot Source:** In this example, we use **Image** as the boot source.
- **Create New Volume:** Leave this checked as **Yes**. This creates a new Cinder volume where the specified operating system image is copied into it. The volume ultimately exists with the Ceph cluster, in the `vmfs` pool.
- **Volume Size:** Allow the system to determine this for you.
- **Delete Volume on Instance Delete:** Leave this option set as **No**. If checked, when the instance is deleted, the volume is as well.

Under the **Available** section, select the appropriate operating system. This example uses CentOS 8 Stream (e18-x86_64).

This concludes configuring the instance's source. Next, move to the **Flavor** tab.

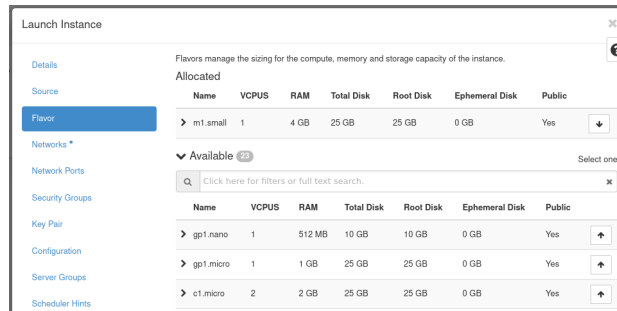


Figure 20: Instance Flavor

Flavors are a way to define the VCPUs, RAM, and Disk space used by an instance. Pre-built flavors are available for you. For this step, select an appropriate flavor from the options under the **Available** heading. This example uses the `m1.small` flavor.

Next, move to the **Networks** tab.

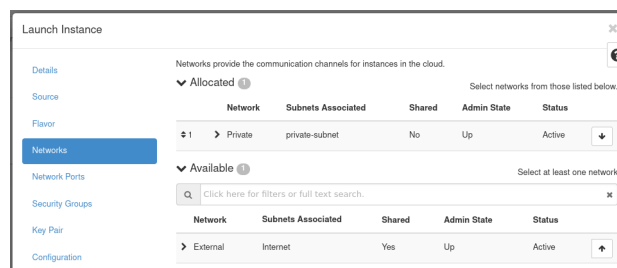


Figure 21: Instance Networks

In this section, you specify the network with which the instance is associated. For this example, select the **Private** network created previously. You can choose the **External** network as well, but this is generally recommended against in favor of using a floating IP should your instance require Internet connectivity.

Note! Only expose portions of your network as necessary. This reduces the attack surface and improves application security. If a private network is not created and an instance is created in a default cloud, it is associated with the **External** network. This means the instance consumes a public IP and it could be reached over the Internet.

Next, skip over the **Network Ports** tab and move to the **Security Groups**.

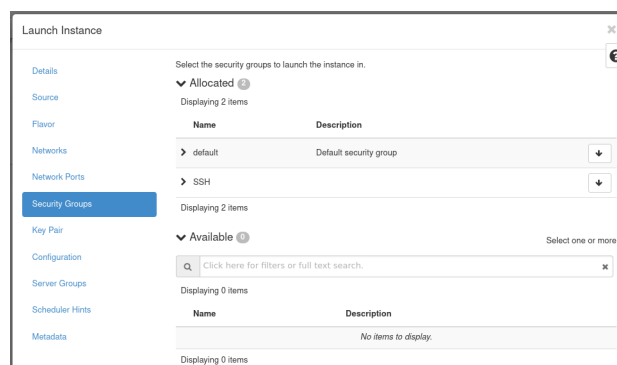


Figure 22: Instance Security Groups

This is where you select security groups for the instance. This example uses the **SSH** security group in the **Available** section.

As the final step, move to the **Key Pair** tab.

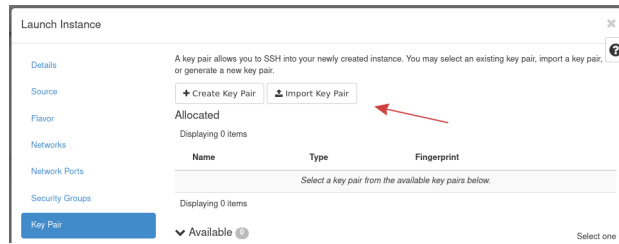


Figure 23: Instance Key Pair

In this section, you specify an SSH public key to inject into the instance. You can upload your key at this stage using this form using the **Import Key Pair** button. For this demonstration, we use **Import Key Pair** to import the existing SSH public key [created previously](#) for one of the control plane nodes.

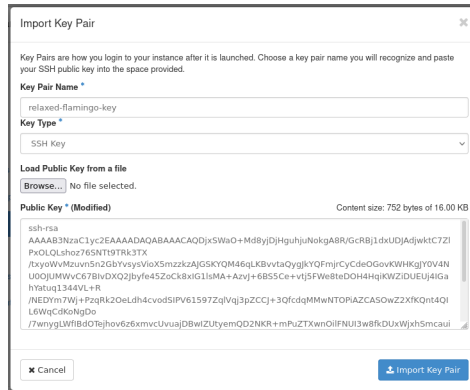


Figure 24: Import Key Pair Form

- **Key Pair Name:** Set a name for the SSH public key. This example public key is called `relaxed-flamingo-key`.
- **Key Type:** This example uses an **SSH Key** key type.
- **Load Public Key from a file:** Specify the location of the public key on your machine.
- **Public Key:** Here you can paste in the public key.

Once the public key is imported, create the instance by pressing **Launch Instance**.

Note! – We skip some sections of the instance creation form as they are not required for this demonstration.

The instance goes through a build process. Allow a few minutes for this to occur. When complete, the instance appears in the **Instances Listing** page.

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
Jumpstation	-	192.168.0.200	m1.small	relaxed-flamingo-key	Active	nova	None	Running	0 minutes	Create Snapshot

Figure 25: Instance Listing

Assign and Attach Floating IP

The instance created previously is associated with a private network. Presently, the only way to access this instance is to connect to it from with the cloud’s hardware nodes. Another option for connecting is to use a floating IP. In this section, we demonstrate how to allocate a floating IP and attach it to this instance.

To allocate a floating IP, first navigate to **Project -> Network -> Floating IPs**.

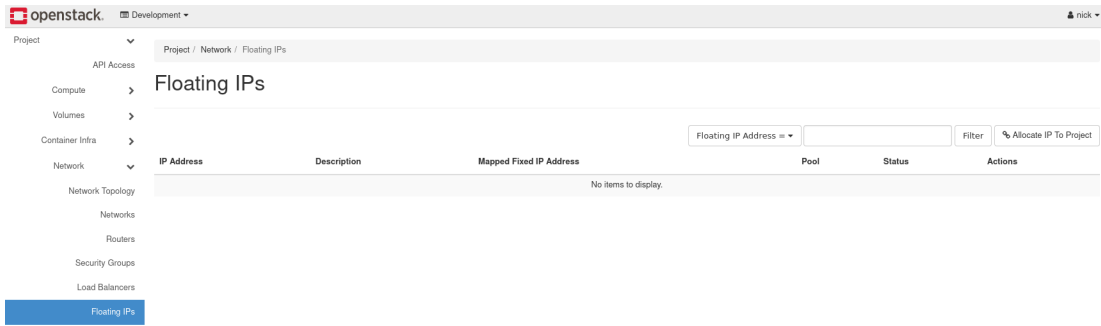
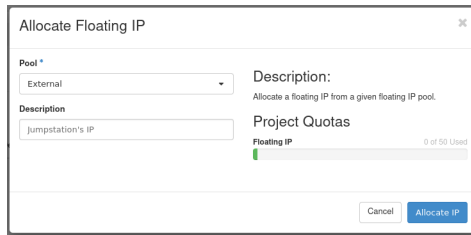


Figure 26: Floating IPs Listing

Next, load the form to allocate a floating IP by pressing **Allocate IP to Project**.



- **Pool:** Select **External** for the allocation pool.
- **Description:** Optional. Set a description for the floating IP.

Press **Allocate IP** to add this floating IP address for use.

Next, in the same section, allocate the IP to the Jumpstation instance by clicking the **Associate** button at the far right.



Figure 27: Associate Floating IP

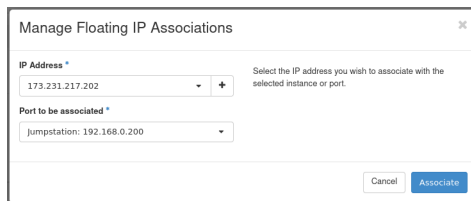


Figure 28: Manage Floating IP Associations

- **IP Address:** This field comes pre-selected with the floating IP so there's no need to change anything here.
- **Port to be associated:** Select the instance created previously. In this case, we use the Jumpstation instance.

This concludes allocating the floating IP to the instance. This instance is now accessible over SSH from the first hardware node of your cloud.

How to Install and Use OpenStack's CLI

Introduction

Your OpenMetal Private Cloud can not only be managed through a web browser, but also through the command line using OpenStack's CLI called OpenStackClient. Using the command line to manage your cloud introduces more flexibility in automation tasks and can generally make an administrator's life simpler. In this guide, we introduce you to the command line method of managing your cloud by explaining how to install and use OpenStackClient.

How to Install OpenStackClient

Prerequisites

- A Linux machine in which you can install OpenStackClient. This can be your own machine, the cloud's hardware nodes, or an instance running in the cloud.
- Python 3.6 or greater
- OpenStack RC file
- `clouds.yaml` file

Install OpenStackClient

In this section, we demonstrate the initial preparation and installation of OpenStackClient to the previously created **Jumpstation** CentOS 8 Stream instance.

Initial Preparation

Before installing OpenStackClient, you must obtain two files from Horizon, which are required to prepare your shell environment. Those two files are `clouds.yaml` and the OpenStack RC file.

- `clouds.yaml`: Used as a source of configuration for how to connect to a cloud
- OpenStack RC file: Used as a source of authentication for your user and project

To collect these files, log in to Horizon as your user. Navigate to **Project -> API Access** to download the OpenStack `clouds.yaml` and the OpenStack RC files to your machine. When you navigate to **Project -> API Access** and collect these files, they are associated with the current user and project that user is in.

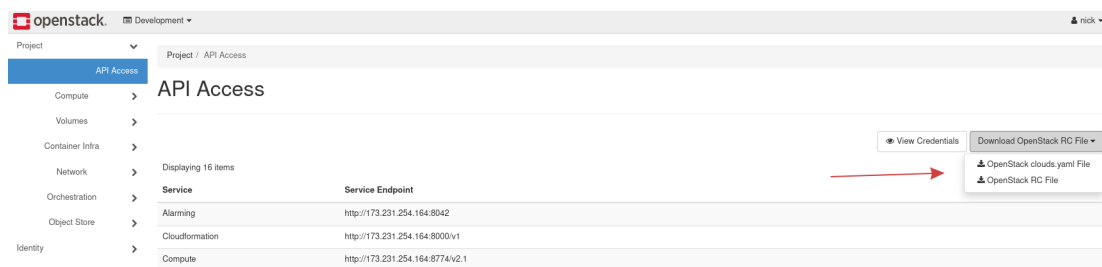


Figure 1: API Access

This example acquires the `clouds.yaml` and OpenStack RC file for the user **nick** and that user's primary project, **Development**.

Prepare and Install OpenStackClient

Next, use SSH to log in to the instance created previously. If you created your instance by following the guide, [How to Create an Instance in OpenStack Horizon](#), then this instance can only be accessed from one of your control plane nodes.

Step 1: Prepare `clouds.yaml` and OpenStack RC files

The `clouds.yaml` file obtained previously must be prepared in this instance. For this demonstration, `clouds.yaml` is located as `~/.config/openstack/clouds.yaml` in this instance. Copy the contents of `clouds.yaml` obtained from Horizon and store it as `~/.config/openstack/clouds.yaml`.

For example:

```
# Create the following directory
$ mkdir -p ~/.config/openstack

# Copy clouds.yaml into this file
$ vi ~/.config/openstack/clouds.yaml
```

Note – The `clouds.yaml` file can be placed in several locations. For more see the [Configuration Files](#) heading of OpenStack Victoria’s documentation.

Next, copy the contents of your OpenStack RC file, in our case called `~/Development-openrc.sh`, into the instance. This file can be placed anywhere and in this example, the file is stored in the `centos` user’s home directory.

For example:

```
$ vi ~/Development-openrc.sh
```

Step 2: Create a Python virtual environment

This environment is created so as to not interfere with the system’s Python version.

In a default CentOS 8 Stream installation, the system’s Python executable is `/usr/libexec/platform-python` and is what will be used to create the virtual environment.

Use `/usr/libexec/platform-python -m venv ~/venv` to create a virtual environment in path `~/venv`.

For example:

```
$ /usr/libexec/platform-python -m venv ~/venv
```

Step 3: Activate the Python virtual environment

Use `source ~/venv/bin/activate` to activate the virtual environment.

For example:

```
$ source ~/venv/bin/activate
```

Step 4: Upgrade pip

Before installing OpenStackClient and to aid in a smooth installation, upgrade `pip`. Upgrade `pip` by using `pip install --upgrade pip`.

For example:

```
$ pip install --upgrade pip
```

Step 5: Install OpenStackClient

With everything prepared, OpenStackClient can be installed.

Note! — There exist two OpenStackClient packages: `python-openstackclient` and `openstackclient`. We recommend using `python-openstackclient` because it is maintained much more frequently than the prior package.

Install OpenStackClient using:

```
$ pip install python-openstackclient
```

Step 6: List servers associated with your project

For an initial command, list the servers associated with your project by running `openstack server list`.

For example:

```
$ openstack server list
```

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



```
+-----+-----+-----+-----+
| 3bb6f079-90d3-4233-9400-94ef49c34a34 | Jumpstation | ACTIVE | Private=173.231.217.202, 192
```

Here, we can see the server created in the previous guide.

Command Structure

When using OpenStackClient, there is typically a common command pattern for what you want to accomplish. All `openstack` commands begin with `openstack`. You can execute `openstack` by itself to enter into a shell, where commands no longer need to be prefixed by `openstack`:

```
(venv) [root@lovely-ladybug ~]# openstack
(openstack)
```

List all Available Subcommands

Use `openstack --help` to list all available subcommands. You initially see all the flags you can pass, but after scrolling a bit, the subcommand list starts:

```
Commands:
  access rule delete  Delete access rule(s)
  access rule list    List access rules
  access rule show    Display access rule details
  access token create Create an access token
  acl delete          Delete ACLs for a secret or container as identified by its href. (py
thon-barbicanclient)
[...output truncated...]
```

Learn more about a Subcommand

After seeing available commands, learn more about a command by using `openstack help <command>`.

For example, to learn more about the `openstack server` command, use `openstack help server`:

```
$ openstack help server
Command "server" matches:
  server add fixed ip
  server add floating ip
  server add network
  server add port
  server add security group
[...output truncated...]
```

List Items and Show Details

It is very common when using OpenStackClient to list items and the command form is typically `openstack <subcommand> list`. For example, `openstack server list`, lists all servers for the currently configured project.

Furthermore, more information about an item can be found by typically running `openstack <subcommand> show <item>`. For example, `openstack server show Jumpstation` shows the details about the instance named **Jumpstation**.

Enable Bash Autocompletion

By default, shell autocompletion is not enabled for the `python-openstackclient` package. To view the autocompletion Bash script, use `openstack complete`. To make the autocompletion persist, store the output of `openstack complete` into `/etc/bash_completion.d/osc.bash_completion` and reload your shell.

For example, we print the autocomplete configuration and redirect its output to `/etc/bash_completion.d/osc.bash_completion` using `tee`:

```
$ openstack complete | sudo tee /etc/bash_completion.d/osc.bash_completion > /dev/null
```

Next, either log out and back in to your shell or use `source` to load the autocompletion script for your current shell.

For example:

```
$ source /etc/bash_completion.d/osc.bash_completion
```

Reference

[OpenStack Victoria OpenStackClient Documentation](#)

Create SSH Key Pair for an OpenStack Control Plane Node

Introduction

In this guide, we explain how to create an SSH key pair within one of your cloud's control plane nodes.

How to Create an SSH Key Pair

Prerequisites

Before beginning, ensure you have SSH access as `root` to your cloud's control plane nodes.

Create the Key Pair

Step 1 – Log in to a control plane node

To begin, connect to one of your cloud's control planes nodes with SSH as `root`. This example connects to the control plane node identified by `173.231.254.165`.

For example:

```
$ ssh root@173.231.254.165
Activate the web console with: systemctl enable --now cockpit.socket
```

```
Last login: Mon Nov  8 16:53:30 2021 from 173.231.218.25
```

Step 2 – Create the key pair

Next, use `ssh-keygen` to generate an SSH key pair. This example demonstrates creating a key pair of size 4096 bits, specified by `ssh-keygen -b 4096`. The private key is saved in the default location of `/root/.ssh/id_rsa` and for additional security, a passphrase is set.

For example:

```
[root@relaxed-flamingo ~]# ssh-keygen -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:iSRlQtRn5VVSRlklmfiWrVJPWtn0oV8HtxLLs6wA7iQ root@relaxed-flamingo.local
The key's randomart image is:
+---[RSA 4096]-----+
|   o=..  .o. **= |
|   . o. oo *oo= |
|   . .  o  + **= |
|   o o .   @ *= |
|   o S   + @ o |
|   E o . . = o |
|   +   . o |
|   .   . |
```

```
|  
+----[SHA256]-----+
```

Step 3 – View contents of public key

To view the contents of the public key, use `cat /root/.ssh/id_rsa.pub`. If the key was saved in another location, be sure to use the appropriate file.

For example:

```
[root@relaxed-flamingo ~]# cat /root/.ssh/id_rsa.pub  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDjxSWaO+Md8yjdJHguhjuNokgA8R/GcRBjldxUDJAdjwktC7ZlPxOI
```

To have access to another machine, this public key needs to exist within the file `~/.ssh/authorized_keys`. For the sake of this series of guides, this public key is injected into the instance created within [How to Create an Instance in OpenStack Horizon](#)

Conclusion

This concludes the steps needed to create an SSH key pair for one of your cloud's control plane nodes.

Day 2

Day 2 initially explains briefly how your Private Cloud was deployed. Next we cover OpenStack administration by explaining how to perform maintenance tasks for your cluster's hardware nodes.

How Private Clouds are Deployed

Introduction

This section details how your Private Cloud was deployed and provides supplemental information regarding your environment. OpenStack can be deployed in several different ways and this section highlights the characteristics of your Private Cloud. We also explain some of the advantages for this type of deployment and areas that are unique to OpenMetal.

Initial Deployment

In OpenMetal, OpenStack is containerized through Docker using Kolla Ansible. This is done through an initial deployment container called FM-Deploy. FM-Deploy provides the initial setup changes during the provisioning process of your Private Cloud. The FM-Deploy Container is a necessary part of the current architecture of your Private Cloud. The FM-Deploy Container should remain running in your Private Cloud as it is used by our systems in the event you want to add or remove nodes from your cloud.

Containerization of OpenStack

OpenMetal uses Kolla Ansible to set up Docker containers for all running services. Should you need to make any configuration changes to your nodes, Kolla Ansible should be used to push these changes. If Kolla Ansible is not used then there is a risk of these changes being reverted during any system updates.

Advantages of Containerization Through Docker

- Containers create an isolated environment reducing software dependencies
- Containers can be scaled and allow for services to balance across your cluster
- Containers provide increased flexibility for test releases, patches, and automation
- Containers have a consistent and repeatable deployment and a shorter initialization time

Disk Storage and Ceph

In OpenMetal, disk storage is provided through Ceph. Ceph is an object storage interface that can provide interfaces for multiple different storage types on a single cluster. In OpenMetal Ceph is comprised of two elements object storage and block storage:

Object Storage

Ceph Object storage utilizes Ceph Object Storage Gateway daemon (RADOSGW). With OpenMetal clouds, Ceph's RGW replaces Swift so there is no Docker container for Swift. Instead, Swift endpoints are connected directly to the RGW. Authentication for RGW is handled through Keystone in `/etc/ceph/ceph.conf`.

Block Storage

Ceph Block storage connects to the Cinder service utilizing Ceph's RADOS Block Device. Within your cloud, those objects are stored in Ceph pools. Ceph provides a layer of abstraction that allows objects to be recognized as blocks.

Advantages of using Ceph

- Data is self-healing and will redistribute data across your cluster in the event of power, hardware, or connectivity issues
- Data is replicated and highly available
- Ceph has the ability to run on commodity hardware and to mix hardware from different vendors

Introduction to Ceph

Introduction

Ceph is an open-source, distributed storage system that provides object, block and file storage interfaces from a single cluster. In this guide we give a high-level overview of Ceph's services and how they are used in your OpenMetal Private Cloud.

Advantages of Ceph

See [Ceph's Benefits](#) page for a more complete understanding of the benefits Ceph provides.

Data Resiliency

With Ceph, your data storage is resilient either through the use of replication or erasure coding. In our configuration, replication is used to create multiple copies of data. Data is replicated at the host level. A Private Cloud is deployed with three hosts. With our Ceph configuration, you could lose two hosts and still have all of your Ceph cluster's data.

Ceph Scales Extremely Well

Ceph is designed to scale horizontally into Petabyte figures.

Disadvantages of Ceph

Data access times are not as quick compared to accessing data directly from a disk. Should your workload require very fast disk reads and writes, consider using a compute only node, where instances can be spun up on local, LVM-backed storage.

For more, see [Spin up an Instance with Ephemeral Storage](#).

Ceph Version Used by Private Clouds

With current Private Cloud deployments, Ceph's [Octopus release](#) is used.

View Disk Usage of the Ceph Cluster

For more, see the guide on [How to Check Ceph's Status and Disk Usage](#).

Default Configuration for the Ceph Cluster

Core Ceph services are deployed to each control plane node upon initial cloud deployment. This means each node comes installed with Ceph's Monitor, Manager, Object Gateway (RADOSGW), and Object Storage Daemon (OSD) services. Each node's secondary NVMe drive backs a Ceph OSD. Our Ceph deployments are configured with host-level replication of data. Your cloud can lose all but one host and still retain all of the Ceph cluster's data.

Default Ceph Pools

By default, your Ceph cluster's data storage is logically divided into pools, a concept associated with Ceph.

From a control plane node as root, we use `ceph osd lspools` to list the default pools associated with a Private Cloud:

```
1 device_health_metrics
2 images
3 volumes
4 vms
5 backups
6 metrics
7 manila_data
8 manila_metadata
9 .rgw.root
10 default.rgw.log
11 default.rgw.control
12 default.rgw.meta
13 default.rgw.buckets.index
```

Next, we explain the purpose of some of the Ceph pools.

Pool: images

Operating System images maintained by Glance are stored in this pool

Pool: volumes

Cinder stores any volumes created by your cloud in this pool.

Pool: vms

When you spin up a volume-backed instance, Nova is configured to create a volume for that instance in this pool.

Pool: backups

Cinder stores volume backups within this pool.

Swift and Cinder Ceph Configuration

With Private Clouds, Swift and Cinder are the services configured to connect to Ceph.

Swift, OpenStack's Object Storage service, connects directly to the Ceph cluster. With our setup you will not see configuration for Swift in a place like `/etc/kolla/swift`. This configuration is instead handled by Ceph directly and can be viewed through `/etc/ceph/ceph.conf`.

Cinder, OpenStack's Block Storage service, is also configured to connect to Ceph. With Cinder, there are several services, of which `cinder-volume` and `cinder-backup` are connected to Ceph. The Ceph configuration for each service can be viewed through `/etc/kolla/cinder-volume/cinder.conf` and `/etc/kolla/cinder-backup/cinder.conf`.

Reconfiguring your Ceph Cluster

WARNING! – Our current deployment system deploys a Private Cloud with a known working state. Should you deviate from this state by adjusting your cloud's Ceph configuration you can no longer safely use the functions in OpenMetal Central to add nodes to your cloud or add IP blocks. Should you use these functions, any custom configurations to Ceph will be reverted. We are working on rolling out a new deployment system allowing custom cloud configurations. We can still add new nodes and IP blocks to your cloud but must do so manually. Please reach out to your Account Manager should this apply to you.

Your Ceph cluster was deployed using Ceph Ansible. Any configuration changes must be made using Ceph Ansible. For more information, see [How to Prepare and Use Ceph Ansible](#).

How to Check Ceph's Status and Disk Usage

Introduction

Ceph was selected as the storage solution for Private Cloud Core OpenStack clouds due to its ability store data in a replicated fashion. The data stored in the Ceph cluster is accessible from any of your cloud's control plane nodes. The storage is considered shared across all nodes, which can make recovering an instance and its data trivial. As an administrator of this cloud, we aim to provide you information about how you can check the status of your Ceph cluster and see available disk usage using the command line.

Prerequisites

- Root access to your cloud's control plane nodes

Check Ceph Status

To check the status of your Ceph cluster, use `ceph status`.

For example:

```
# ceph status
cluster:
  id:          34fa49b3-fff8-4702-8b17-4e8d873c845f
  health: HEALTH_OK

services:
  mon: 3 daemons, quorum relaxed-flamingo,focused-capybara,lovely-ladybug (age 2w)
  mgr: relaxed-flamingo(active, since 2w), standbys: focused-capybara, lovely-ladybug
  osd: 4 osds: 4 up (since 3d), 4 in (since 3d)
  rgw: 3 daemons active (focused-capybara.rgw0, lovely-ladybug.rgw0, relaxed-flamingo.rgw0)

task status:

data:
  pools:   13 pools, 337 pgs
  objects: 69.28k objects, 250 GiB
  usage:   724 GiB used, 11 TiB / 12 TiB avail
  pgs:    337 active+clean

io:
  client: 121 KiB/s rd, 1.2 MiB/s wr, 137 op/s rd, 232 op/s wr
```

Check Ceph Disk Usage

To check the available disk space in your Ceph cluster, use `ceph df`.

For example:

```
# ceph df
--- RAW STORAGE ---
CLASS  SIZE      AVAIL    USED      RAW USED  %RAW USED
```

Table of Contents

ssd	12 TiB	11 TiB	720 GiB	724 GiB	6.08
TOTAL	12 TiB	11 TiB	720 GiB	724 GiB	6.08

--- POOLS ---

POOL	ID	PGS	STORED	OBJECTS	USED	%USED	MAX AVAIL
device_health_metrics	1	1	286 KiB	4	858 KiB	0	3.4 TiB
images	2	32	7.6 GiB	1.02k	23 GiB	0.22	3.4 TiB
volumes	3	32	88 GiB	23.61k	264 GiB	2.45	3.4 TiB
vms	4	32	144 GiB	39.92k	433 GiB	3.96	3.4 TiB
backups	5	32	0 B	0	0 B	0	3.4 TiB
metrics	6	32	25 MiB	4.49k	127 MiB	0	3.4 TiB
manila_data	7	32	0 B	0	0 B	0	3.4 TiB
manila_metadata	8	32	0 B	0	0 B	0	3.4 TiB
.rgw.root	9	32	3.6 KiB	8	96 KiB	0	3.4 TiB
default.rgw.log	10	32	3.4 KiB	207	384 KiB	0	3.4 TiB
default.rgw.control	11	32	0 B	8	0 B	0	3.4 TiB
default.rgw.meta	12	8	954 B	4	36 KiB	0	3.4 TiB
default.rgw.buckets.index	13	8	2.2 MiB	11	6.6 MiB	0	3.4 TiB

OpenStack Hardware Node Maintenance

Introduction

Software in the OpenStack ecosystem evolves over time, either through new feature additions, bug fixes, or when vulnerabilities are patched. Part of operating an OpenStack cloud involves maintaining its software through updates. In this guide, we explain how to perform operating system updates to your cloud's control plane nodes. Additionally we detail how to deploy the latest OpenStack service images using Kolla Ansible.

Prerequisites

Before getting started, ensure the following requirements are met:

- Root access to hardware nodes
- Ansible experience
- Linux command line experience
- How to work with Kolla Ansible

How to Perform Operating System Updates

Private Clouds use CentOS 8 as the operating system running each control plane node. This section explains how to perform operating system updates for any Private Clouds running CentOS 8.

Before Performing OpenStack Maintenance

Operating system updates to a control plane node can be disruptive to any workload you may have running on it. Before performing maintenance on a control plane node you may want to migrate your workload or any instances hosted on it to another compute node. Instances can be migrated using Horizon or through the command line, using OpenStackClient. For information on how to migrate instances, see [How to Live Migrate Instances Using Horizon](#).

When performing updates to control plane nodes, maintenance should occur one node at a time.

Getting Started

To get started, login as root over SSH into a node requiring maintenance. Next, the following commands are used to update the operating system:

```
systemctl disable docker.socket
systemctl stop docker.socket
systemctl disable docker.service
systemctl stop docker service
```

Table of Contents

```
dnf upgrade
reboot
```

Each command needs to be run in order. Once updates are complete through DNF, reboot the hardware node. When the first node restarts successfully and has rejoined the OpenStack and Ceph clusters, move on to the next node and perform the same steps until no nodes remain.

The following demonstrates running the above commands in detail.

Disable Docker socket

Use `systemctl disable docker.socket` to disable the Docker socket.

For example:

```
[root@smiling-pelican ~]# systemctl disable docker.socket
[root@smiling-pelican ~]#
```

Running this command returns no output and has succeeded.

Stop Docker socket

Use `systemctl stop docker.socket` to stop the Docker socket.

For example:

```
[root@smiling-pelican ~]# systemctl stop docker.socket
[root@smiling-pelican ~]#
```

Running this command returns no output and has succeeded.

Disable Docker service

Use `systemctl disable docker.service` to disable the Docker service.

For example:

```
[root@smiling-pelican ~]# systemctl disable docker.service
Removed /etc/systemd/system/multi-user.target.wants/docker.service.
```

Stop Docker service

Use `systemctl stop docker.service` to stop the Docker service.

For example:

```
[root@smiling-pelican ~]# systemctl stop docker.service
[root@smiling-pelican ~]#
```

Running this command returns no output and has succeeded.

Update DNF

Use `dnf upgrade` to update the package manager.

For example:

```
[root@smiling-pelican ~]# dnf update
Last metadata expiration check: 0:11:51 ago on Tue 17 Aug 2021 02:30:48 PM UTC.
Dependencies resolved.
```

```
=====
Package                                Architecture  Version                               Repository  Si
=====
Upgrading:
containerd.io                          x86_64       1.4.9-3.1.el8                        docker      30
docker-ce                                x86_64       3:20.10.8-3.el8                      docker      22
docker-ce-cli                            x86_64       1:20.10.8-3.el8                      docker      29
```


Table of Contents

```
docker-ce-rootless-extras      x86_64      20.10.8-3.el8      docker      4.6
epel-release                    noarch      8-11.el8            epel        23

Transaction Summary
=====
Upgrade 5 Packages

Total download size: 86 M
Is this ok [y/N]:

[...output truncated...]

Upgraded:
  containerd.io-1.4.9-3.1.el8.x86_64      docker-ce-3:20.10.8-3.el8.x86_64
  docker-ce-cli-1:20.10.8-3.el8.x86_64   docker-ce-rootless-extras-20.10.8-3.el8.x86_64
  epel-release-8-11.el8.noarch

Complete!
```

Reboot the node

Use `reboot` to restart the node.

For example:

```
[root@smiling-pelican ~]# reboot
Connection to node3 closed by remote host.
Connection to node3 closed.
```

Use `ping` to watch for when the node comes back online.

For example:

```
$ ping node3
PING node3 (173.231.217.231) 56(84) bytes of data.
64 bytes from node3 (173.231.217.231): icmp_seq=1 ttl=60 time=12.1 ms
64 bytes from node3 (173.231.217.231): icmp_seq=2 ttl=60 time=13.9 ms
64 bytes from node3 (173.231.217.231): icmp_seq=3 ttl=60 time=15.1 ms
```

Note! – If a node fails to come back online, please contact support for assistance.

Verify successful reboot

When the node comes back online, SSH into it to verify the OpenStack Docker containers have started and to check Ceph's cluster status.

To verify the Docker containers have started, use `docker ps`. You should see a number of Docker containers running. Under the **STATUS** column, each container should reflect the status `Up`.

For example:

```
[root@smiling-pelican ~]# docker ps
CONTAINER ID   IMAGE
6f7590bc2191   harbor.imhadmin.net/kolla/centos-binary-telegraf:victoria
67a4d47e8c78   harbor.imhadmin.net/kolla/centos-binary-watcher-api:victoria
af815b1dcb5d   harbor.imhadmin.net/kolla/centos-binary-watcher-engine:victoria
a52ab61933ac   harbor.imhadmin.net/kolla/centos-binary-watcher-applier:victoria
[...output truncated...]
```

Next, if this node is part of a Ceph cluster, check Ceph's status using `ceph status`.

For example:

```
[root@smiling-pelican ~]# ceph status
cluster:
  id:      06bf4555-7c0c-4b96-a3b7-502bf8f6f213
```

```
health: HEALTH_OK
[...output truncated...]
```

The above output shows the status as `HEALTH_OK`, indicating the Ceph cluster is healthy. Ceph is naturally resilient and should recover from a node being rebooted.

How to Obtain Latest OpenStack Images using Kolla Ansible

OpenStack has been deployed using Kolla Ansible. Kolla Ansible deploys OpenStack services as Docker images. These images may need updates periodically. This section explains how to pull the latest images and deploy them.

Getting Started

To get started, first login to a node requiring maintenance as root over SSH. Next, ensure you have [prepared a Kolla Ansible environment](#).

With Kolla Ansible prepared, this section explains how to pull the latest Kolla images.

Pull latest Kolla Ansible images

Using Kolla Ansible, pull down the latest images using `kolla-ansible -i <path-to-inventory> pull`.

The inventory file used is `/etc/fm-deploy/kolla-ansible-inventory`.

For example:

```
$ kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory pull
Pulling Docker images : ansible-playbook -i /etc/fm-deploy/kolla-ansible-inventory -e @/etc/ko
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see d
[WARNING]: Could not match supplied host pattern, ignoring: enable_nova_True

PLAY [Gather facts for all hosts] *****

TASK [Gather facts] *****
ok: [localhost]
ok: [smiling-pelican]
ok: [intelligent-squirrel]
ok: [charming-stoat]

[...output truncated...]

PLAY RECAP *****
charming-stoat      : ok=48   changed=2   unreachable=0   failed=0   skipped=56
intelligent-squirrel : ok=48   changed=2   unreachable=0   failed=0   skipped=57
localhost          : ok=4    changed=2   unreachable=0   failed=0   skipped=0
smiling-pelican    : ok=48   changed=3   unreachable=0   failed=0   skipped=56
```

The above indicates there were no issues with each host when pulling the latest images. You can move on to the next step to deploy the images.

Deploy Kolla Ansible images

Next, use Kolla Ansible to deploy the images with the command `kolla-ansible -i <path-to-inventory> deploy`.

For example:

```
$ kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory deploy
Deploying Playbooks : ansible-playbook -i /etc/fm-deploy/kolla-ansible-inventory -e @/etc/ko
[WARNING]: Invalid characters were found in group names but not replaced, use
-vvvv to see details
[WARNING]: Could not match supplied host pattern, ignoring: enable_nova_True

PLAY [Gather facts for all hosts] *****
```

TASK [Gather facts] *****

```
ok: [localhost]
ok: [smiling-pelican]
ok: [intelligent-squirrel]
ok: [charming-stoat]
```

[...output truncated...]

PLAY RECAP *****

```
charming-stoat      : ok=300  changed=77  unreachable=0    failed=0    skipped=167
intelligent-squirrel : ok=458  changed=94  unreachable=0    failed=0    skipped=188
localhost          : ok=4    changed=2   unreachable=0    failed=0    skipped=0
smiling-pelican    : ok=300  changed=80  unreachable=0    failed=0    skipped=167
```

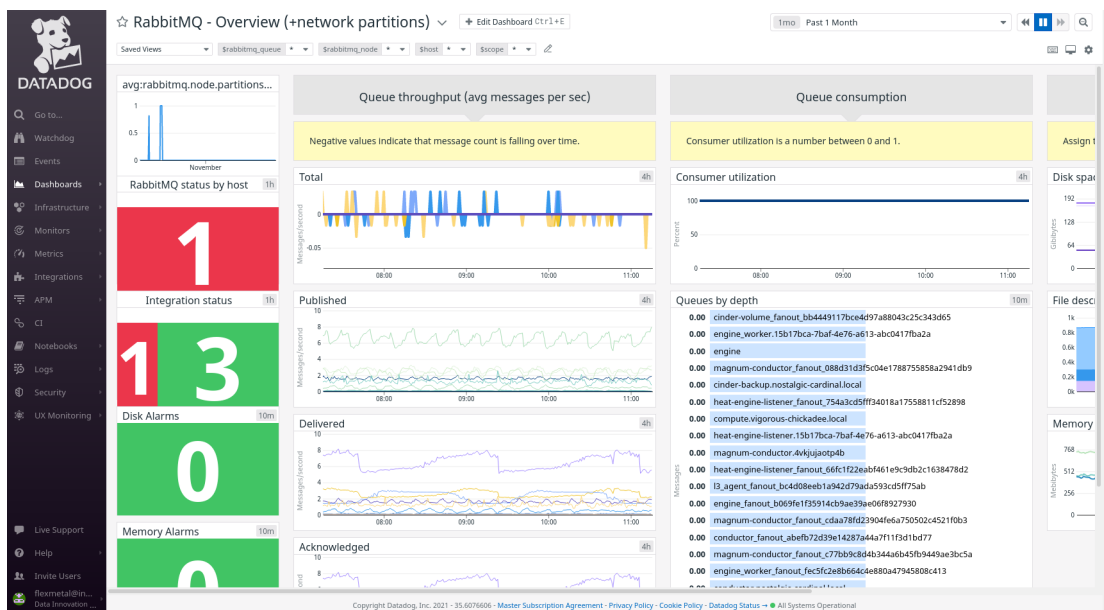
The results of this run indicate a successful deployment and this cloud is now using the latest Kolla Ansible images.

1. [Datadog](#)
2. [Resources of a Private Cloud](#)
 1. [View Memory and Compute Usage in Horizon](#)
 2. [View Instance State Across Cluster](#)
 3. [How to Access Resource Information from Ceph](#)
3. [Adding nodes to your Ceph Cluster](#)
4. [Removing nodes from your Ceph Cluster](#)

Datadog

The primary option for accessing resource data for your private cloud is through the use of Datadog. Datadog allows for the collection of logs from a wide variety of services. Datadog then has the ability to visualize and alert based on this log data. Through Datadog you have the ability to customize logging for your containers as well as any new services you may add. If you are interested in installing and enabling Datadog visit [Monitor OpenStack with Datadog](#).

Figure 1: Sample Datadog dashboard



Resources of a Private Cloud

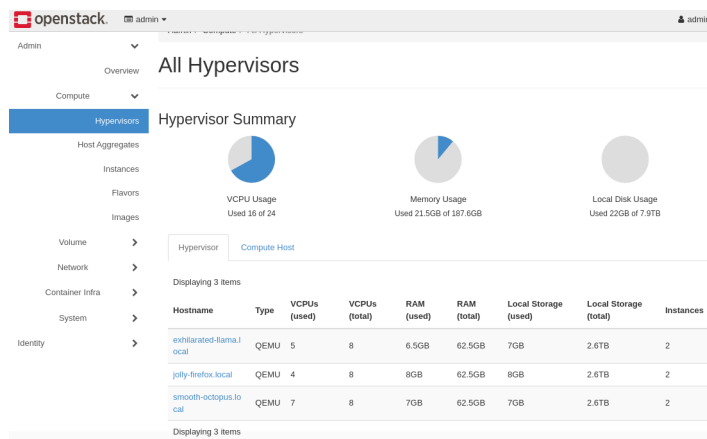
There are currently three variations to private cloud deployments: Small, Standard, and Large. All private cloud deployments have a cluster of three hyper-converged servers but will have different allocations of memory, storage, and CPU processing power depending on the configuration and hardware. In addition, you have the option of adding additional hardware nodes to your cluster.

View Memory and Compute Usage in Horizon

To view the resources used by your cloud, you have to be the admin user and assigned to the admin project. Once you are in the admin project, navigate to **Admin -> Compute -> Hypervisors**. This section, lists the following items:

- VCPU Usage
- Memory Usage
- Local disk usage

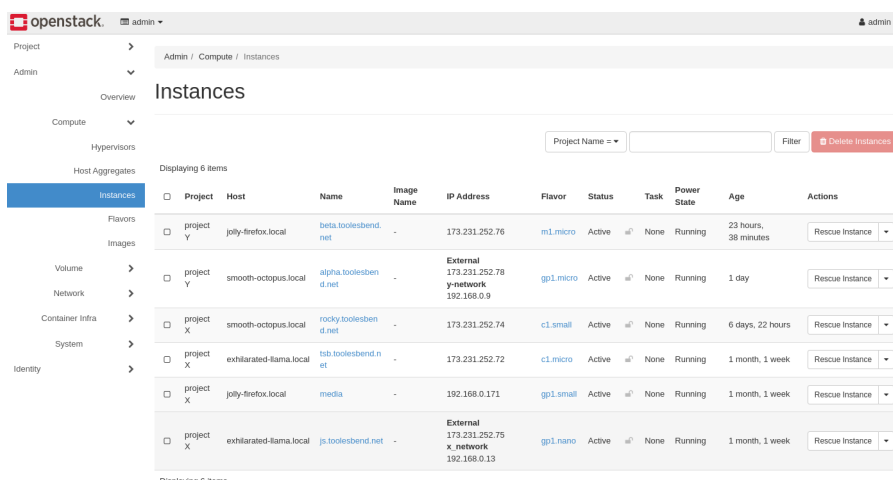
Figure 2: Hypervisor Summary showing disk and resource usage



View Instance State Across Cluster

There is also an option to see the location of your instances within your cluster. To view this information, navigate to **Admin -> Compute -> Instances**. You have the option to see the project, the host, as well as the IP address, and state.

Figure 3: Summary of Instances



How to Access Resource Information from Ceph

To access information regarding your Ceph cluster's resource pools, you will need to use Ceph's CLI. These are a summary of some useful resource monitoring and health commands. For further information, visit Ceph's [Ceph Administration Tool](#) page.

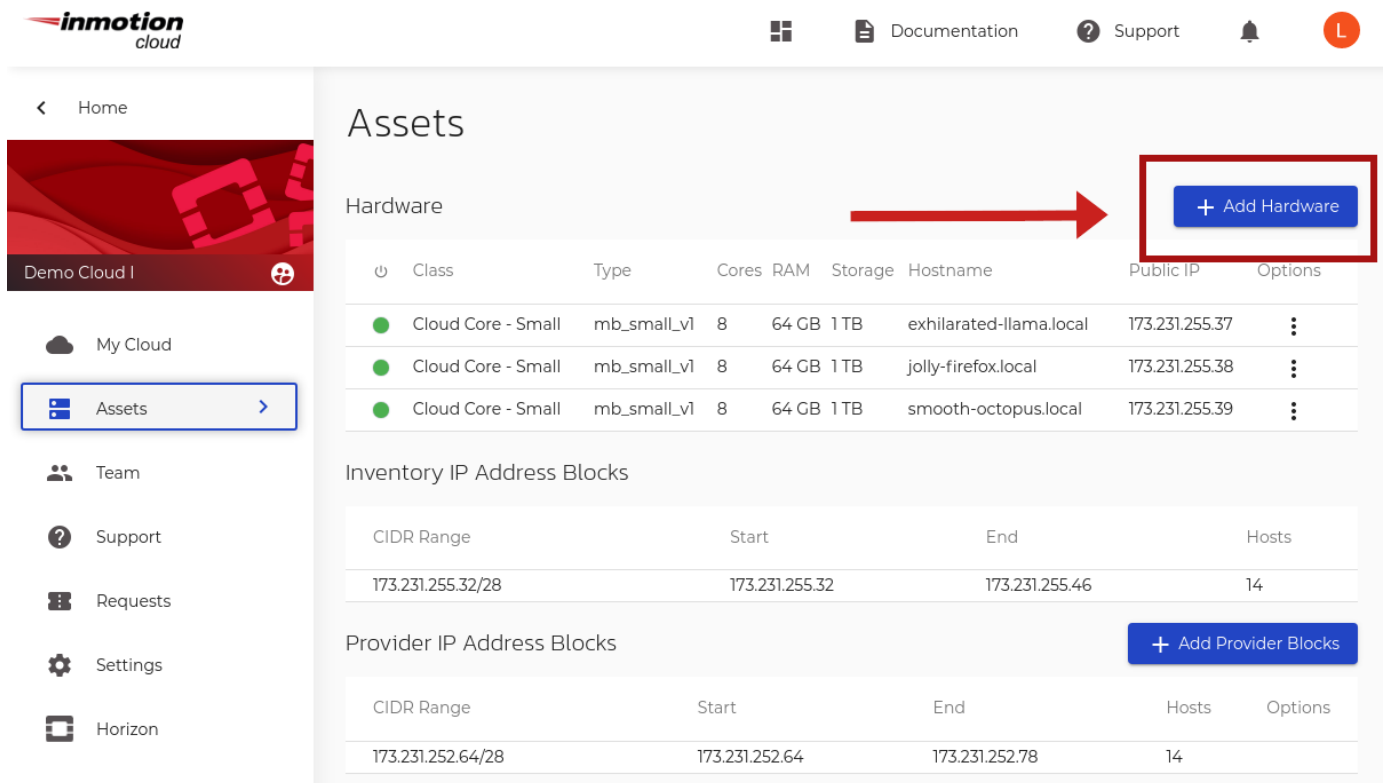
- `ceph -s` to check the status of Ceph

- `ceph df` to list the disk usage overview
- `ceph health detail` provides details about existing health issues
- `ceph osd pool ls` to list pools add `detail` for additional details regarding replication and health metrics

Adding nodes to your Ceph Cluster

In order to add additional hardware for your cluster, first review [Cloud Hardware Best Practices](#). This is to ensure that you have selected the appropriate node for your cloud. Once you have selected the appropriate node, navigate to [OpenMetal Central](#) and click on your **Assets** page. The option to add hardware is located in the top right corner as **Add Hardware**.

Figure 4: Picture of Assets Page where hardware is added to Private Clouds



Removing nodes from your Ceph Cluster

To remove nodes from your Ceph Cluster, you must file a support ticket within [OpenMetal Central](#). A member of our support staff will be able to further assist you with removing hardware from your private cloud. For assistance in submitting support tickets, visit [How to Reach Support](#) in the heading of our [Introduction to OpenMetal Central Guide](#).

How to Live Migrate Instances Using OpenStack Horizon

Introduction

This guide provides instructions for cloud administrators on how to migrate instances through your Horizon dashboard. Migration is the process that a server administrator can move instances to a different host. Live migration keeps instances in an active state during the migration process. This process is useful when applications need to remain running and shutting down an instance is not possible or advantageous.

Prerequisite

Live migrating instances requires having a user account with the administrator role. This is typically the account called **admin**.

Determining an Instance's Parent Host

To determine an instance's parent host, navigate to **Admin -> Compute -> Instances**. You have the option to see the project, the host, as well as the IP address, and the state of your instance.

Project	Host	Name	Image Name	IP Address	Flavor	Status	Task	Power State	Age	Actions	
project X	smooth-octopus.local	rocky.toolsbend.net	-	173.231.252.74	c1.small	Active	ui	None	Running	1 week	Rescue Instance
project X	jolly-firefox.local	media	-	192.168.0.171	gp1.small	Active	ui	None	Running	1 month, 1 week	Rescue Instance
project X	exhilarated-flama.local	js.toolsbend.net	-	External 173.231.252.75 x_network 192.168.0.13	gp1.nano	Active	ui	None	Running	1 month, 1 week	Rescue Instance
project Y	jolly-firefox.local	beta.toolsbend.net	-	173.231.252.76	m1.micro	Active	ui	None	Running	1 day, 22 hours	Rescue Instance
project Y	smooth-octopus.local	alpha.toolsbend.net	-	External 173.231.252.78 y_network 192.168.0.9	gp1.micro	Active	ui	None	Running	1 day, 23 hours	Rescue Instance

Figure 1: Summary of Instances

Migrate Instance

Once you have determined the instance you want to migrate, navigate to **Admin -> Compute -> Instances**. On this page, you are presented with a series of actions. To access these actions click the small triangle next to the button called **Rescue Instance**.

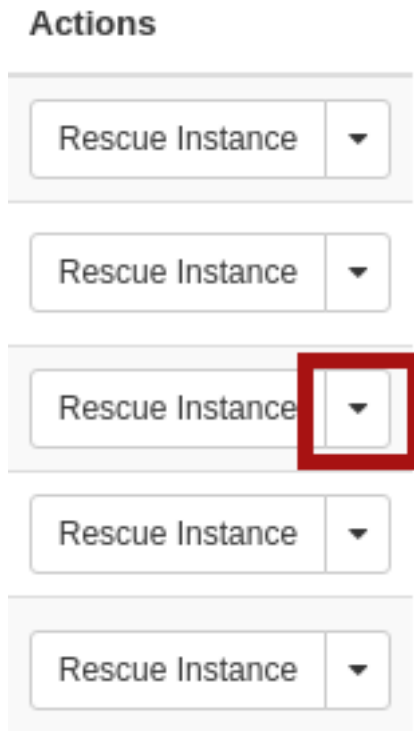


Figure 2: Link for Live Migration Drop Down Menu

From the drop-down menu, select **Live Migrate Instance**.

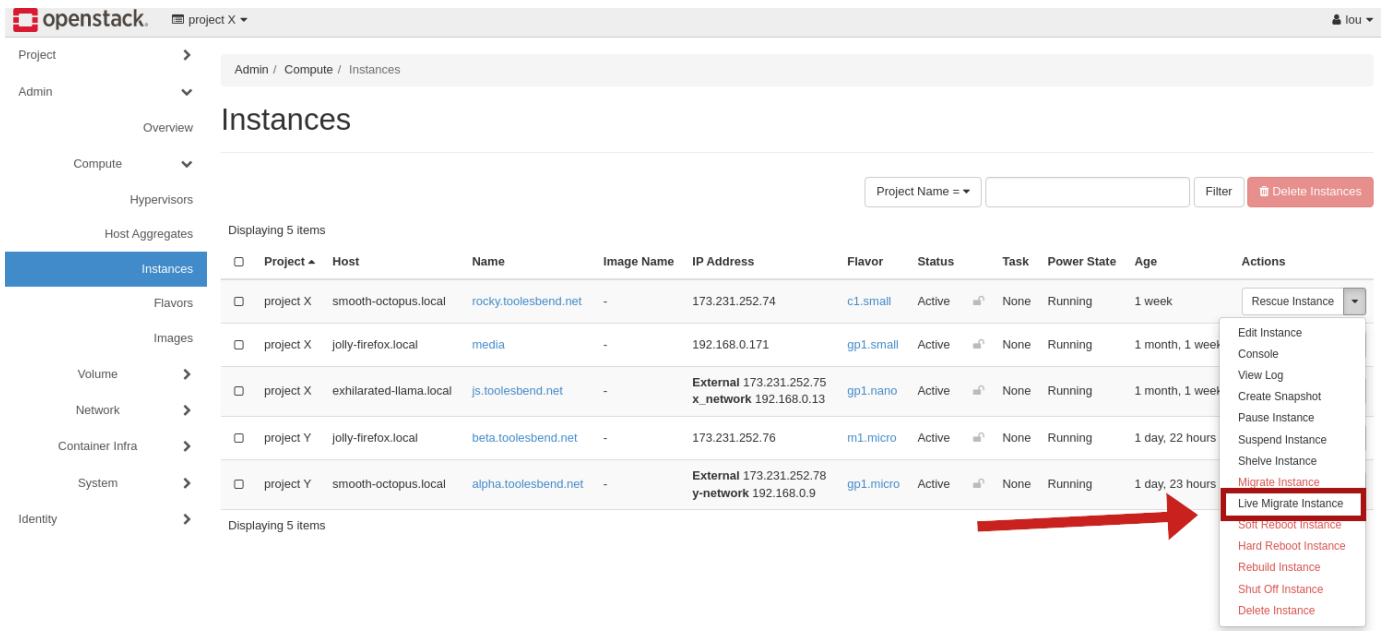


Figure 3: Drop down menu for live migration

Within Live Migrate, the options for **Disk Over Commit** and **Block Migration** are options for local storage. Block live migration uses ephemeral disks on instances. These disks are not shared between source and destination hosts. The storage for OpenMetal uses a shared storage system through Ceph. Because of the shared storage system in OpenMetal, neither option should be selected.

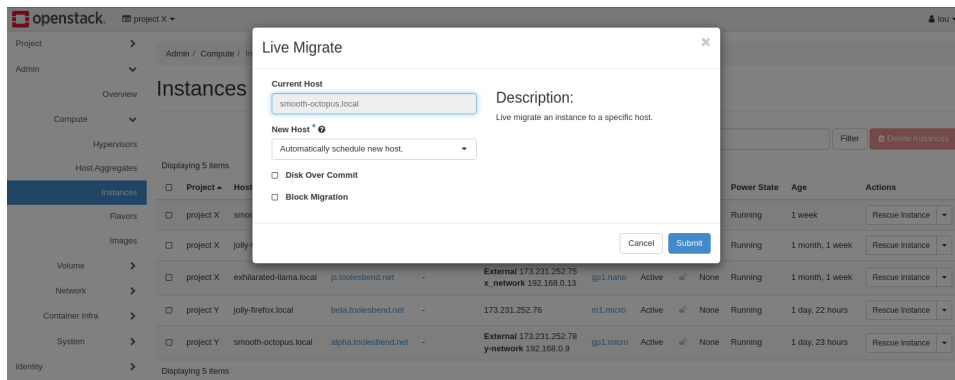


Figure 4: Live Migration Options

Live Migrate does have the option to either auto-select a new host or to choose one manually. If you have a specific node for your instance, select the name of the node under the **New Host** option. Once you have made your selection, click submit. During the live migration process, a status bar appears under tasks, upon completion, the task returns to none and your instance host changes to a new node.

The screenshot shows the OpenStack dashboard for 'project X'. The 'Instances' page displays a table with 5 items. The instance 'project X exhilarated-llama.local' is in a 'Migrating' state, highlighted with a red box and a red arrow pointing to its 'Task' column.

Project	Host	Name	Image Name	IP Address	Flavor	Status	Task	Power State	Age	Actions
project Y	jolly-firefox.local	beta.toolesbend.net	-	173.231.252.76	m1.micro	Active	None	Running	1 day, 23 hours	Rescue Instance
project Y	smooth-octopus.local	alpha.toolesbend.net	-	External 173.231.252.78 y_network 192.168.0.9	gp1.micro	Active	None	Running	2 days	Rescue Instance
project X	exhilarated-llama.local	rocky.toolesbend.net	-	173.231.252.74	c1.small	Migrating	Migrating	Running	1 week	Edit Instance
project X	jolly-firefox.local	media	-	192.168.0.171	gp1.small	Active	None	Running	1 month, 1 week	Rescue Instance
project X	exhilarated-llama.local	js.toolesbend.net	-	External 173.231.252.75 x_network 192.168.0.13	gp1.nano	Active	None	Running	1 month, 1 week	Rescue Instance

Figure 5: Task list for migrating instances

Day 3

Day 3 has to do with how you can scale your cloud by adding more hardware nodes to it. Next we get into preparing for disaster recovery by explaining best practices relating to keeping your data safe. We provide a general outline for how you can recover from a disaster.

1. Cloud Hardware Selection

1. Types of Private Clouds
2. Types of Nodes
3. The Benefit of Homogeneous Clouds

2. Adding Hardware Nodes to a Cloud

1. How to add a Hardware Node
 1. Navigate in OpenMetal Central to Cloud Assets Page
 2. View Hardware Node Types
 3. Confirm Hardware Node Addition
 4. Verify Hardware Addition Success

3. Removing Hardware Nodes from a Cloud

1. Consider Before Removing
2. Initial Preparation
3. How to Remove a Hardware Node from a Cloud

Cloud Hardware Selection

In this section we outline the types of clouds offered, the types of nodes you can add to your cloud, and best practices when choosing hardware for your Private Cloud. Additionally we explain conditions under which your cloud's Ceph cluster is affected when adding hardware.

Types of Private Clouds

We currently offer three server sizes for our Private Clouds as seen in the following image:

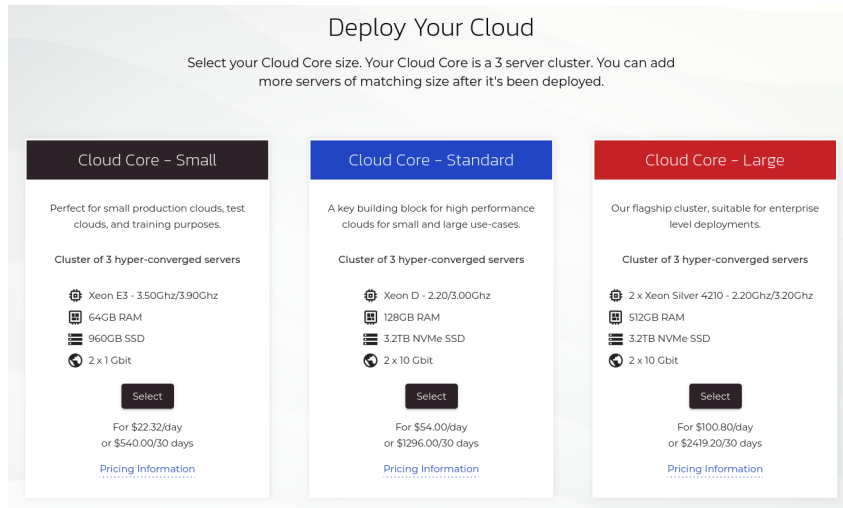


Figure 1: Private Cloud Core Selections

- **Cloud Core - Small:** This cluster type should be considered for testing environments. Each node has a 1Gbit Network Interface Card (NIC). The network throughput across cluster members could be a bottleneck.
- **Cloud Core - Standard:** This cluster type is considered ready for production. Each node has a 10Gbit Network Interface Card. The network throughput across cluster members should not be a bottleneck.
- **Cloud Core - Large:** Production-ready and provides the most resources. Each node also has a 10Gbit Network Interface Card.

Types of Nodes

The following are the types of nodes that can be added to a cloud.

- **Storage and Compute - Small**
- **Storage and Compute - Standard**
- **Storage and Compute - Large**
- **Compute - Large**
- **GPU - A100**

The Benefit of Homogeneous Clouds

By default, Private Clouds are hyper-converged and include a Ceph deployment. Generally speaking, Ceph is only as fast as your slowest node allows, though there are caveats to this. Ceph could be impacted by the type of nodes added to a cluster due to potential differences in hardware, such as performance disparity between storage provided by HDDs, SSDs or NVMe. Having a homogeneous cluster of similarly equipped nodes will have the best results in terms of performance and data density. Additionally, while you can add any hardware node to your cloud, the Network Interface Cards (NICs) for each designation of node may have a different maximum throughput. If a node with a 1Gbit NIC is added to a cluster with nodes having 10Gbit NICs, the internal network traffic is limited by the additional node.

Adding Hardware Nodes to a Cloud

In this section, we explain the steps needed to add hardware nodes to your cloud.

The following demonstrates adding a **Storage and Compute - Standard** node to a **Cloud Core - Standard** cloud.

How to add a Hardware Node

Navigate in OpenMetal Central to Cloud Assets Page

First navigate in OpenMetal Central to your cloud's details page. Next follow the **Assets** link on the left. This page shows you details about your cloud's current assets and also allows you to add a new hardware node. To add a new hardware node, select the **Add Hardware** button located at the top right.

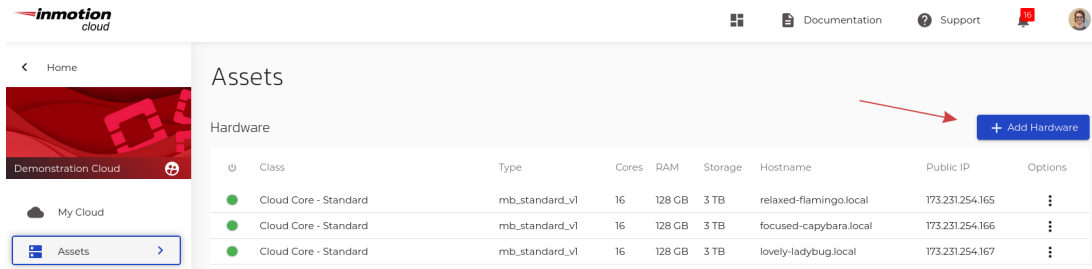


Figure 2: Add Hardware

View Hardware Node Types

The next screen presents you with a list of available hardware nodes as can be seen in the following screenshot.

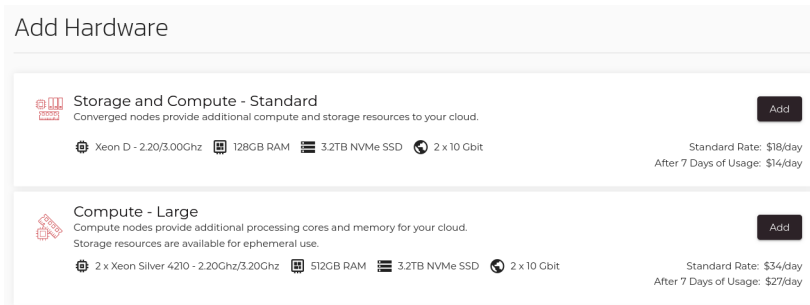


Figure 3: Available Hardware List

Hardware specifications and the cost per day are listed with each available offering.

Select the appropriate hardware node for your cloud. To add a new hardware node, select the **Add** button associated with the node type you would like. Next, specify the amount of nodes to add from the drop down.

Confirm Hardware Node Addition

After choosing the number of nodes to add, confirm the additional hardware.

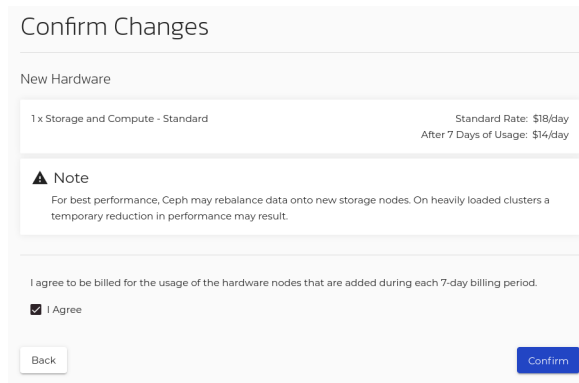
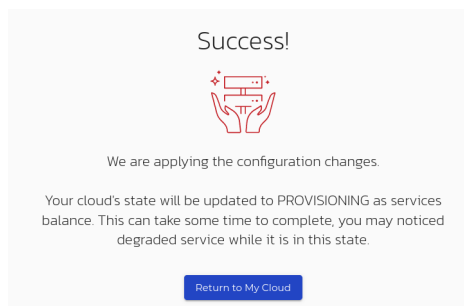


Figure 4: Confirm Addition of new Hardware

After the system processes the request a success message is returned as can be seen below:



Note – Adding a hardware node can take around an hour to complete. An email is sent to the primary account’s address when complete.

Figure 5: Hardware Addition Success

Should you navigate back to your cloud’s assets page, you can visually confirm the additional node by inspecting the hardware associate with your cloud. In this example, there is a red dot next to the newly added node, indicating it is still being added to the cloud.

Assets								
Hardware + Add Hardware ?								
Class	Type	Cores	RAM	Storage	Hostname	Public IP	Options	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	relaxed-flamingo.local	173.231.254.165	⋮	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	focused-capybara.local	173.231.254.166	⋮	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	lovely-ladybug.local	173.231.254.167	⋮	
Storage and Compute - Standard	mb_standard_v1	16	128 GB	3 TB	competent-coypu.local	173.231.254.168	⋮	

Figure 6: Cloud Assets List, Newly Added Node but not Available

Verify Hardware Addition Success

An email is sent to your account upon successfully adding this node. Navigate to your cloud’s assets page to confirm the newly added node.

Assets								
Hardware + Add Hardware								
Class	Type	Cores	RAM	Storage	Hostname	Public IP	Options	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	relaxed-flamingo.local	173.231.254.165	⋮	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	focused-capybara.local	173.231.254.166	⋮	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	lovely-ladybug.local	173.231.254.167	⋮	
Storage and Compute - Standard	mb_standard_v1	16	128 GB	3 TB	competent-coypu.local	173.231.254.168	⋮	

Figure 7: Successful Hardware Node Addition

Removing Hardware Nodes from a Cloud

In this section, we outline the steps required to remove a hardware node from your cloud.

There is not a native feature in OpenMetal Central allowing you to remove hardware nodes from your cloud. Should you need to remove a hardware node from your cloud, consult with your Account Manager first or submit a ticket through OpenMetal Central. It is very important all data required from this node is copied elsewhere prior to making a request to remove a hardware node. You can help facilitate the process of removing the node by [migrating any running instances](#) from it to another node.

Consider Before Removing

At all times, your cloud must have three hyperconverged control plane nodes running to have a fully functioning OpenStack cloud. As such, the node being removed cannot be a control plane node.

Control plane nodes in OpenMetal Central are prefixed with **Cloud Core** and can be distinguished from other nodes by inspecting the **Class** column in your cloud’s assets page in OpenMetal Central.

For example, in the following screenshot, the first three nodes are control plane nodes:

Hardware + Add Hardware ?								
Class	Type	Cores	RAM	Storage	Hostname	Public IP	Options	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	relaxed-flamingo.local	173.231.254.165	⋮	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	focused-capybara.local	173.231.254.166	⋮	
Cloud Core - Standard	mb_standard_v1	16	128 GB	3 TB	lovely-ladybug.local	173.231.254.167	⋮	
Storage and Compute - Standard	mb_standard_v1	16	128 GB	3 TB	competent-coypu.local	173.231.254.168	⋮	

Figure 8: Control Plane nodes

The last node, classified by **Storage and Compute - Standard** is not a control plane node, meaning it does not run all the core OpenStack services.

Our support will review your request to ensure requirements are met prior to removal.

Initial Preparation

To prepare for removal of a hardware node, migrate any instances to another node that has compute services running. All hyperconverged nodes run OpenStack's compute service. For instruction regarding migrating instances to another node, see the [How to Live Migrate Instances](#) guide.

How to Remove a Hardware Node from a Cloud

To remove a hardware node from a cloud a ticket must be submitted with the request through OpenMetal Central. Specify the hostname or IP address of the node you wish to be removed and an agent will review the request.

1. [Getting Started](#)
2. [Confirm new Provider Block Addition](#)
3. [How are the new Provider Block IPs Used?](#)

Getting Started

To add the IP block you need to reach out to your Account Manager who will submit the request. Our support staff will handle the addition and inform you through a OpenMetal Central ticket sent to your primary e-mail address when complete.

Confirm new Provider Block Addition

First to confirm the addition of the IP block, log in to Horizon with a user that has the administrator role. This is typically the account called "admin".

Next, navigate to **Admin -> Network -> Networks**, then click on the "External" network.

Project	Network Name	Subnets Associated	DHCP Agents	Shared	External	Status	Admin State	Availability Zones	Actions
admin	External	Internet 173.231.253.0/28 Internet_ade7d9c0 173.231.217.192/28	2	Yes	Yes	Active	UP	nova	Edit Network
service	lb-mgmt-net	lb-mgmt-subnet 192.168.6.0/24	2	No	No	Active	UP	nova	Edit Network

Figure 1: List of Networks

Choose the Subnets tab and confirm that the new subnet has been added.

Name	CIDR	IP Version	Gateway IP	Used IPs	Free IPs	Actions
Internet	173.231.253.0/28	IPv4	173.231.253.1	2	9	Edit Subnet
Internet_ade7d9c0	173.231.217.192/28	IPv4	173.231.217.193	2	9	Edit Subnet

Figure 2: List of Subnets for the External network

The new subnet will be prefixed with **Internet_** and will have a series of hex values following it.

How are the new Provider Block IPs Used?

When creating a resource that requires an IP from this block, specify the name of the newly added provider block to add IPs from it.

1. **Create a Volume Backup**
 1. Test Volume Backups
 2. Restore a Volume Backup
 3. Ceph, Volumes, and Data Durability
 4. Store Data Outside of the Cloud

Create a Volume Backup

Navigate in Horizon to **Project -> Volume -> Volumes**.

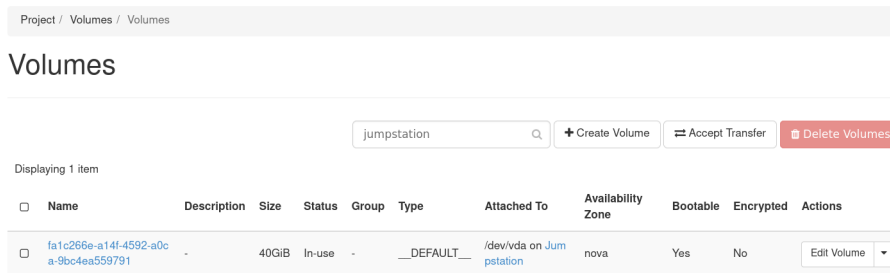


Figure 1: Volumes List

Create a backup of your volume by selecting from the drop down **Create Backup**.

Figure 2: Create Volume Backup Form

- **Backup Name:** Specify a name for the volume backup
- **Description:** Provide a description if necessary
- **Container Name:** Leave this blank otherwise the volume backup cannot be created. Horizon tells you if this field is blank, the backup is stored in a container called `volumebackups`, but this is not the case with our configuration. With Private Clouds, all volume backups created this way are stored in the Ceph pool called `backups`.
- **Backup Snapshot:** If applicable specify a snapshot to create a backup from

After submitting the form, you are navigated to **Project -> Volume -> Volume Backups** where you can see the volume you just created a backup of.

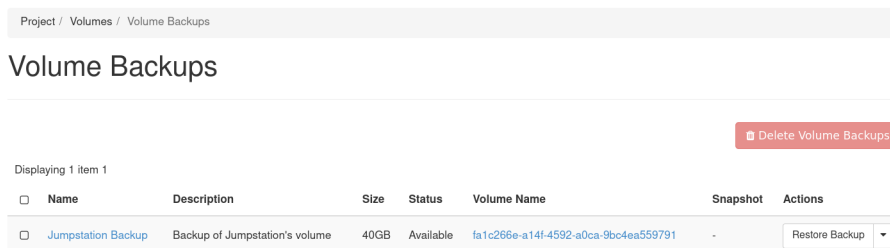


Figure 3: Volume Backup List

Test Volume Backups

Creating backup copies of your important data is only one part of having a solid backup and recovery plan. Additionally, consider testing backed-up data to ensure if something unexpected does happen that restoring your backups will actually be useful. To test volume backups, you can restore a volume backup within OpenStack alongside the original volume and compare the contents.

Restore a Volume Backup

To restore a volume backup, begin by navigating in Horizon to **Project -> Volume -> Volume Backups**.

Next, find the volume backup you wish to restore and from its drop down on the right, select **Restore Backup**.

Name	Description	Size	Status	Volume Name	Snapshot	Actions
<input type="checkbox"/> Jumpstation Backup	Backup of Jumpstation's volume	40GB	Available	fa1c266e-a14f-4592-a0ca-9bc4ea559791	-	Restore Backup

Figure 4: Volume Backup List

Choose the volume to restore to, or have the system restore the backup to a new volume.

Ceph, Volumes, and Data Durability

When volume backups are created, they are stored in your cloud's Ceph cluster in a pool called `backups`. By default, the Ceph cluster is configured with host level replication across each of your cloud's three control plane nodes. With this configuration, your cloud could suffer losing all but one Ceph node and still retain all of the cluster's data. For more about how your Ceph cluster was configured, see the heading **Default Configuration for the Ceph Cluster** in the [Introduction to Ceph](#) guide.

Store Data Outside of the Cloud

Taking data backups a step further, consider storing critical data outside of the cloud. Storing data both in the cloud's Ceph cluster as well as outside of it increases the failure domain.

With Ceph you can use [RBD mirroring](#), which effectively is a way to mirror your Ceph cluster's data to another Ceph cluster.

1. Kolla Ansible and Ceph Ansible

1. [Prepare Kolla Ansible and Ceph Ansible Environment](#)
2. [Where are my Private Cloud's Configuration Files?](#)
 1. [Kolla Ansible Configuration Files](#)
 2. [Ceph Ansible Configuration Files](#)
 3. [FM-Deploy Configuration File](#)
 4. [Network Ansible Configuration File](#)
3. [Keep a Backup Copy of a Private Cloud's Configuration Files](#)

2. How to Restore a Private Cloud's Configuration Files

1. [Example: Recover Neutron's Configuration File using Kolla Ansible](#)
 1. [Prerequisite: Prepare a Kolla Ansible Environment](#)
 2. [Regenerate an OpenStack Service's Configuration File using Kolla Ansible](#)

3. **Create Full and Incremental Copies of a Private Cloud's OpenStack Service Databases**

1. Prerequisites

2. **How to Create OpenStack Service Database Backups**

1. Command Syntax for Full Database Backups

2. Command Syntax for Incremental Database Backups

3. Path to the Kolla Ansible Inventory File

4. Command Usage Example for a Full Database Backup

5. Command Usage Example for an Incremental Database Backup

4. **How to Restore a Private Cloud's OpenStack Service Databases**

1. **Full Database Restoration Steps**

1. Full Restoration: Create Temporary Docker Container

2. Full Restoration: Prepare Backup Directory

2. **Incremental Database Restoration Steps**

1. Incremental Restoration: Create Temporary Docker Container

2. Incremental Restoration: Prepare Backup Directory

5. **References**

Kolla Ansible and Ceph Ansible

Kolla Ansible is responsible for deploying a Private Cloud's OpenStack services. Ceph Ansible then handles deployment of the cloud's Ceph cluster. These two systems deploy all OpenStack and Ceph services required for a Private Cloud, including their configurations.

Prepare Kolla Ansible and Ceph Ansible Environment

Before working with either Kolla Ansible or Ceph Ansible, you must prepare an environment in your shell:

- [How to Prepare and Use Kolla Ansible](#)
- [How to Prepare and Use Ceph Ansible](#)

Where are my Private Cloud's Configuration Files?

When a cloud finishes deploying, the Ansible configurations used to deploy the cloud are exported into each of the control plane nodes within the folders `/etc/fm-deploy` and `/etc/kolla`. This section explains where configuration files are located and their purpose.

Kolla Ansible Configuration Files

The information in these files was used to deploy your Private Cloud's core OpenStack services into Docker containers:

- Kolla Ansible Inventory: `/etc/fm-deploy/kolla-ansible-inventory`
- Kolla Ansible Main Configuration: `/etc/kolla/globals.yml`

Ceph Ansible Configuration Files

The information in these files was used to deploy your Private Cloud's Ceph cluster:

- Ceph Ansible Inventory: `/etc/fm-deploy/ceph-inventory.yml`
- Ceph Ansible Main Configuration: `/etc/fm-deploy/ceph-vars.yml`

FM-Deploy Configuration File

FM-Deploy is a part of the system used to deploy your Private Cloud. Information about this deployment system is provided for the sake of completely explaining the configuration files found within `/etc/fm-deploy`.

- FM-Deploy Main Configuration: `/etc/fm-deploy/config.yml`

Network Ansible Configuration File

The following defines the initial networking configuration for your Private Cloud. This file was used upon initial cloud deployment.

- Inventory: `/etc/fm-deploy/network-inventory.yml`

Keep a Backup Copy a Private Cloud's Configuration Files

In case something unexpected happens to these files, you should keep copies of them off site. To preserve the configuration of OpenStack services and Ceph, copy the directories `/etc/fm-deploy`, `/etc/kolla` and `/etc/ceph` somewhere outside of the cloud, in a secure, private location. These folders contain sensitive information about your cloud so the information within should only be accessible to those you trust or yourself.

How to Restore a Private Cloud's Configuration Files

There are two primary ways a Private Cloud's configuration file can be restored: Copy a known good configuration file from an off site location or use Ansible. This section explains how you can use Kolla Ansible and Ceph Ansible to recover a Private Cloud's configuration files.

Example: Recover Neutron's Configuration File using Kolla Ansible

For example, consider an event where one of your control plane nodes loses its Neutron server configuration file. This section explains how to recover this configuration by using Kolla Ansible.

Prerequisite: Prepare a Kolla Ansible Environment

Before proceeding, a Kolla Ansible environment needs to be prepared. For information about preparing a Kolla Ansible environment, see [How to Prepare and Use Kolla Ansible](#). Once the environment is prepared, navigate back to this section.

Regenerate an OpenStack Service's Configuration File using Kolla Ansible

In this example, we outline restoring the Neutron server configuration file for the control plane node `relaxed-flamingo`.

In the Kolla Ansible inventory file, `/etc/fm-deploy/kolla-ansible-inventory`, the following hosts are defined as control plane nodes under the heading `[control]`:

```
[control]
relaxed-flamingo ansible_host=10.204.28.7
focused-capybara ansible_host=10.204.30.158
lovely-ladybug ansible_host=10.204.25.253
```

To restore the Neutron server configuration file for `relaxed-flamingo`, first ensure you have [prepared a Kolla Ansible environment](#).

Next, use Kolla Ansible's `reconfigure` function, targeting only the Neutron service by using the flag `--tags neutron` and limit the run to the host `relaxed-flamingo` by specifying the flag `--limit control[0]`.

For example:

```
kolla-ansible \
  -i /etc/fm-deploy/kolla-ansible-inventory \
  reconfigure \
  --tags neutron \
  --limit control[0]
```


Create Full and Incremental Copies of a Private Cloud's OpenStack Service Databases

Kolla Ansible provides a utility to create copies of all OpenStack service databases, called `mariadb_backup`. In this section, we explain how to use Kolla Ansible's builtin function to create database backups of a Private Cloud's OpenStack services.

Prerequisites

Before proceeding with this guide, a Kolla Ansible environment needs to be prepared. For information about preparing a Kolla Ansible environment, see [How to Prepare and Use Kolla Ansible](#). Once the environment is prepared, come back to this guide to learn how to create database backups of OpenStack services.

How to Create OpenStack Service Database Backups

The following instruction must be performed from the folder in which you have prepared Kolla Ansible. This section first provides the command syntax, then follows up with an example of the command's execution and output. Note that Kolla Ansible has no way to schedule backups.

Command Syntax for Full Database Backups

The command to perform a full backup of all databases using Kolla Ansible is...:

```
kolla-ansible -i <inventory> mariadb_backup
```

...where `<inventory>` is the path to the Kolla Ansible inventory file.

Command Syntax for Incremental Database Backups

The command to perform an incremental backup of all databases using Kolla Ansible is...:

```
kolla-ansible -i <inventory> mariadb_backup --incremental
```

...where `<inventory>` is the path to the Kolla Ansible inventory file.

Path to the Kolla Ansible Inventory File

The Kolla Ansible inventory file is located across all control plane nodes as:

```
/etc/fm-deploy/kolla-ansible-inventory
```

Command Usage Example for a Full Database Backup

From the host that has Kolla Ansible prepared, the following command is executed:

```
kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory mariadb_backup
```

Truncated output of the above command:

```
# kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory mariadb_backup
Backup MariaDB databases : ansible-playbook -i /etc/fm-deploy/kolla-ansible-inventory -e @/e

[...previous output truncated...]

TASK [mariadb : Taking full database backup via Mariabackup] *****
skipping: [focused-capybara]
skipping: [lovely-ladybug]
[WARNING]: The value False (type bool) in a string field was converted to 'False' (type stri
value to ensure it does not change.
changed: [relaxed-flamingo]

PLAY RECAP *****
focused-capybara      : ok=2    changed=0    unreachable=0    failed=0    skipped=1
lovely-ladybug       : ok=2    changed=0    unreachable=0    failed=0    skipped=1
relaxed-flamingo     : ok=3    changed=1    unreachable=0    failed=0    skipped=0
```

The task [mariadb : Taking full database backup via Mariabackup] is where a backup of all OpenStack service databases is created. Kolla Ansible creates a Docker volume called mariadb_backup to store the database copies. Previous backups made using this method are not overwritten. The host under this task that reports a change (example: changed=1) is where the Docker volume storing the databases is created.

Note! – For this example, since the Docker volume was created on another host, the remaining instruction in this guide must be performed from that host. If Kolla Ansible creates mariadb_backup on another host, you must SSH into that host as root to continue this process.

Command Usage Example for an Incremental Database Backup

Note! – Incremental backups can only be made if a full backup has been made prior, otherwise the following command will result in an error.

From the host that has Kolla Ansible prepared, the following command is executed:

```
kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory mariadb_backup \
  --incremental
```

Truncated output of the above command:

```
# kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory mariadb_backup --incremental
Backup MariaDB databases : ansible-playbook -i /etc/fm-deploy/kolla-ansible-inventory -e @/e

[...previous output truncated...]

TASK [mariadb : include_tasks] *****
included: /opt/kolla-ansible/.venv/share/kolla-ansible/ansible/roles/mariadb/tasks/backup.yml

TASK [mariadb : Taking incremental database backup via Mariabackup] *****
skipping: [focused-capybara]
skipping: [lovely-ladybug]
[WARNING]: The value False (type bool) in a string field was converted to 'False' (type string)
change.
changed: [relaxed-flamingo]

PLAY RECAP *****
focused-capybara      : ok=2    changed=0    unreachable=0    failed=0    skipped=1
lovely-ladybug       : ok=2    changed=0    unreachable=0    failed=0    skipped=1
relaxed-flamingo     : ok=3    changed=1    unreachable=0    failed=0    skipped=0
```

The task [mariadb : Taking incremental database backup via Mariabackup] is where an incremental backup of all OpenStack service databases is created. Kolla Ansible creates a Docker volume called mariadb_backup to store the database copies. Previous backups made using this method are not overwritten. The host under this task that reports a change (example: changed=1) is where the Docker volume storing the databases is created.

Note! – For this example, since the Docker volume was created on another host, the remaining instruction in this guide must be performed from that host. If Kolla Ansible creates mariadb_backup on another host, you must SSH into that host as root to continue this process.

How to Restore a Private Cloud’s OpenStack Service Databases

This section explains how to restore both **full** and **incremental** database backups created using Kolla Ansible's mariadb_backup function.

Full Database Restoration Steps

Follow these steps to learn how to restore full OpenStack service databases created using Kolla Ansible's mariadb_backup function.

Full Restoration: Create Temporary Docker Container

In this section, we create a temporary Docker container called `dbrestore`. This container is created with the same volumes as the `mariadb` Docker container. The `mariadb_backup` Docker volume is mounted as `/backup` in this container. Finally, the container is created using the `kolla/centos-binary-mariadb:victoria` Docker image available from Docker Hub with a Bash shell.

Create the temporary Docker container called `dbrestore` using:

```
docker run --rm -it --volumes-from mariadb --name dbrestore \
  --volume mariadb_backup:/backup \
  kolla/centos-binary-mariadb:victoria \
  /bin/bash
```

Once you run the above Docker command, your terminal should appear this way:

```
() [mysql@06ab93fb83a3 /]$
```

Full Restoration: Prepare Backup Directory

Caution! – Be careful when using commands. The following commands make use of the `rm` command which deletes files.

Next, the backup data must be prepared before it can be copied into place.

This example uses a full MariaDB backup called `mysqlbackup-08-12-2021-1638999340.qp.xbc.xbs.gz`.

To prepare the backup data, in the Docker container, run:

```
cd /backup
rm -rf /backup/restore
mkdir -p /backup/restore/full
gunzip mysqlbackup-08-12-2021-1638999340.qp.xbc.xbs.gz
mbstream -x -C /backup/restore/full/ < mysqlbackup-08-12-2021-1638999340.qp.xbc.xbs
mariabackup --prepare --target-dir /backup/restore/full
```

Load another shell session for the node in which you are working and stop the MariaDB Docker container:

```
docker stop mariadb
```

Navigate back to the Docker container and run:

```
rm -rf /var/lib/mysql/*
rm -rf /var/lib/mysql/\.[^\.]*
mariabackup --copy-back --target-dir /backup/restore/full
```

Next, navigate back to the other shell and start the MariaDB Docker container:

```
docker start mariadb
```

Examine MariaDB's logs to confirm the Galera cluster has synchronized:

```
# tail -1 /var/log/kolla/mariadb/mariadb.log
2021-12-08 22:27:39 2 [Note] WSREP: Synchronized with group, ready for
connections
```

Incremental Database Restoration Steps

Follow these steps to learn how to restore an incremental OpenStack service database backup created using Kolla Ansible's `mariadb_backup` function.

Incremental Restoration: Create Temporary Docker Container

In this section, we create a temporary Docker container called `dbrestore`. This container is created with the same volumes as the `mariadb` Docker container. The `mariadb_backup` Docker volume is mounted as `/backup` in this container. Finally, the container is created using the `kolla/centos-binary-mariadb:victoria` Docker image available from Docker Hub with a Bash shell.

Create the temporary Docker container called `dbrestore` using:

Table of Contents

```
docker run --rm -it \  
  --volumes-from mariadb \  
  --name dbrestore \  
  --volume mariadb_backup:/backup \  
  kolla/centos-binary-mariadb:victoria \  
  /bin/bash
```

Once you run the above Docker command, your terminal should appear this way:

```
() [mysql@06ab93fb83a3 /]$
```

Incremental Restoration: Prepare Backup Directory

Caution! – Be careful when using commands. The following commands make use of the `rm` command which deletes files.

This section assumes a full and incremental backup have been created. Note that your full and incremental backup file names will differ from this example.

Next, we must prepare the backup data before it can be copied into place.

In the Docker container, run:

```
cd /backup/  
rm -rf /backup/restore  
mkdir -p /backup/restore/full  
mkdir -p /backup/restore/incremental  
gunzip mysqlbackup-10-12-2021-1639166052.qp.xbc.xbs.gz  
gunzip incremental-20-mysqlbackup-10-12-2021-1639169695.qp.xbc.xbs.gz  
mbstream -x -C /backup/restore/full/ < mysqlbackup-10-12-2021-1639166052.qp.xbc.xbs  
mbstream -x -C /backup/restore/incremental/ < incremental-20-mysqlbackup-10-12-2021-1639169695.qp.xbc.xbs  
mariabackup --prepare --target-dir=/backup/restore/full/  
mariabackup --prepare --target-dir=/backup/restore/full/ --incremental-dir=/backup/restore/incremental/
```

Load another shell session for the node in which you are working and stop the MariaDB Docker container:

```
docker stop mariadb
```

Navigate back to the Docker container and run:

```
rm -rf /var/lib/mysql/*  
rm -rf /var/lib/mysql/\.[^/.]*  
mariabackup --copy-back --target-dir /backup/restore/full/
```

Next, navigate back to the other shell and start the MariaDB Docker container:

```
docker start mariadb
```

Examine MariaDB's logs to confirm the Galera cluster has synchronized:

```
# tail -1 /var/log/kolla/mariadb/mariadb.log  
2021-12-08 22:27:39 2 [Note] WSREP: Synchronized with group, ready for  
connections
```

References

- Kolla Ansible's [MariaDB database backup and restore](#)
- MariaDB's [Full Backup and Restore with Mariabackup](#)
- MariaDB's [Incremental Backup and Restore with Mariabackup](#)

1. [Disaster Recovery Strategies](#)

1. [Recovery Objectives](#)
2. [Off-site Backups](#)
3. [RBD Mirroring with Ceph](#)
2. **Handling a Hardware Failure**
 1. [Determine Hardware Node Failure](#)
 2. [Cluster Failure](#)
3. [Cloud Monitoring with Datadog](#)
4. [Contact Support](#)
5. [Additional Reading](#)

Disaster Recovery Strategies

Disaster recovery varies greatly depending on the needs of your organization. Your disaster recovery plan should involve analyzing your company's most valuable data. This audit should include an inventory of documents, databases, and systems that are deeply involved in the revenue-generating aspects of your business. As an infrastructure provider, we also have the responsibility to maintain the integrity of our facilities to the best standards and practices to safeguard your data. In addition to our responsibilities, you should take the additional steps necessary to protect the integrity of your data against potential impacts on your private cloud.

Recovery Objectives

The objectives behind a disaster recovery plan fall into two categories:

- **Recovery Point Objectives** - This is a specific point in time that data must be backed up to resume business.
- **Recovery Time Objectives** - Length of time that a system can be offline before the business is negatively impacted.

Consider the above objectives when determining the needs of your disaster recovery strategies. In recovery point objectives, determining the frequency of your backups is critical to your disaster strategy. Recovery time objectives are based on the amount of lost revenue per unit of lost time. This means that every hour or minute that certain systems are offline can greatly impact a business.

Off-site Backups

To have an effective disaster recovery plan, we stress the use of an off-site backup solution. Off-site backups are a critical component to preventing catastrophic loss of data and limiting the blast radius of an outage. Although some services in OpenMetal are highly available, high availability does not protect against unforeseen natural disasters. In the event of a hurricane, flood, or other unforeseen circumstance, it could be possible that you must recover data from a different geographical location. It is highly recommended that off-site backups are used for the most mission-critical data.

RBD Mirroring with Ceph

Private Clouds by default use Ceph as a shared storage backend. As part of your disaster recovery plan, consider using Ceph's [RBD mirroring](#) feature to mirror important data to another Ceph cluster, in a geographically different data center location. **Note** that we currently only provide one data center in which to host your Private Cloud.

Handling a Hardware Failure

Great care is taken to monitor and maintain our infrastructure. However, in the event of hardware failure, it might be necessary to diagnose these issues. If you believe you are experiencing hardware failure, contact our support. Two possible hardware-related issues are failure of a hardware node and failure of an entire cloud. The following sections outline how to diagnose hardware failure.

Determine Hardware Node Failure

To quickly get the status of your cloud’s control plane nodes, check the running system services in Horizon. To do so, log in as an administrator to Horizon and navigate to **Admin -> System -> System Information**. On this page are tabs separating all OpenStack services, including the APIs, Compute, Block Storage, and Network Agents. A service’s state is indicated by the **State** column. If a service is up, the value is reflected as “Up”. If a service is down, the value is reported as “Down”. Click through each tab to see each service’s status. A good indicator that a control plane node is down is if you see “Down” under the **State** column for all services of a particular node.

Name	Host	Zone	Status	State
cinder-scheduler	exhilarated-llama.local	nova	Enabled	Up
cinder-scheduler	jolly-firefox.local	nova	Enabled	Up
cinder-scheduler	smooth-octopus.local	nova	Enabled	Up
cinder-volume	exhilarated-llama.local@rbd-1	nova	Enabled	Up
cinder-volume	jolly-firefox.local@rbd-1	nova	Enabled	Up
cinder-volume	smooth-octopus.local@rbd-1	nova	Enabled	Up
cinder-backup	exhilarated-llama.local	nova	Enabled	Up
cinder-backup	jolly-firefox.local	nova	Enabled	Up
cinder-backup	smooth-octopus.local	nova	Enabled	Up

Figure 1: OpenStack System Information

Cluster Failure

If your entire Private Cloud has failed, the signs will be readily apparent. If you are using monitoring software such as Datadog or Nagios, you will be alerted to your nodes being offline. In addition to being alerted, you will be unable to access your assets.

Note: In some instances, it is also possible to see a failure of your nodes within OpenMetal central. To view the status of these nodes, go to the assets page of OpenMetal central, and under the hardware section, there is a green indicator icon reflecting the current status of your nodes. If the icon is yellow or red, then the issue is likely hardware-related.

The screenshot shows the OpenMetal Central interface. On the left is a navigation menu with 'Assets' selected. The main content area is titled 'Assets' and has a 'Hardware' section. Below 'Hardware' is a table with columns: Class, Type, Cores, RAM, Storage, and Hostname. The first cell of the 'Class' column contains a power icon, and the three rows below it each contain a green status indicator. A red arrow points to the green indicator in the first row. Below the table are sections for 'Inventory IP Address Blocks' and 'Provider IP Address Blocks', each with a table of CIDR ranges, start, and end addresses.

Figure 2: Assets Page of OpenMetal Central

Cloud Monitoring with Datadog

Datadog is an optional cloud monitoring service we provide for OpenMetal Private Clouds. Should you want to add this feature to your cloud, contact your Account Manager or submit a support ticket in OpenMetal Central.

Contact Support

If you are experiencing hardware failure or any other issues with your Open Metal Private Cloud, [Contact Support](#).

Additional Reading

For more regarding OpenStack Disaster Recovery, see:

- RedHat's [Disaster Recovery Enablement in OpenStack](#)
- InMotion Hosting's [What is Disaster Recovery as a Service \(DRaaS\)?](#)
- OpenStack's [Disaster Recovery Wiki Page](#)

Day 4

In Day 4, we cover advanced OpenStack administration through Kolla Ansible, the primary system used to deploy a Private Cloud. Through this system, we explain how configuration changes to your cloud can be made. Next, we detail a few common troubleshooting scenarios and their solutions. Finally we briefly cover automation techniques possible in your cloud through the use of OpenStack's Heat service and through HashiCorp's Terraform application.

1. [Prerequisites](#)
2. [Using Kolla Ansible Quick Start](#)
3. **[Prepare Kolla Ansible for Use](#)**
 1. [Kolla Ansible Configuration Files](#)
 2. [Before Making Changes](#)
 3. [Prepare Kolla Ansible Environment](#)
4. [References](#)
5. [Next Steps](#)

Prerequisites

- **Root Access:** Root access to your cloud's control plane nodes is required.
- **Ansible Experience:** Experience using [Ansible](#).

Using Kolla Ansible Quick Start

This section outlines all steps required to prepare a Kolla Ansible environment within a Private Cloud. For more detailed instruction, see [Prepare Kolla Ansible for Use](#).

Caution! Before using this Quick Start, ensure the node you are working from contains the file `/etc/fm-deploy/kolla-ansible-inventory`. Only a single node should have this file.

To quickly get an idea of what is required to start using Kolla Ansible, here is a high-level overview of the steps:

```
# Copy Kolla Ansible configuration from fm-deploy docker container
$ docker cp fm-deploy:/opt/kolla-ansible /opt/kolla-ansible

# Navigate to /opt/kolla-ansible
$ cd /opt/kolla-ansible
```

Table of Contents

```
# Initialize a Python virtual environment
$ virtualenv .venv

# Activate the virtual environment
$ source .venv/bin/activate

# Update pip
pip install --upgrade pip

# Install Kolla Ansible using requirements.txt
$ pip install -r requirements.txt
```

With these steps complete, you can now use Kolla Ansible. Execute `kolla-ansible --help` for a list of available commands. Alternatively, see [Kolla Ansible CLI](#) for a list of commands.

Prepare Kolla Ansible for Use

To start using Kolla Ansible, an environment needs to be created. This section explains the steps needed to create that environment.

Kolla Ansible Configuration Files

First, we introduce you to the files required for adjusting Kolla Ansible's configuration. These files are used when preparing Kolla Ansible.

- Kolla Ansible Inventory: `/etc/fm-deploy/kolla-ansible-inventory`
- Kolla Ansible Main Configuration: `/etc/kolla/globals.yml`

Note! – The file `/etc/fm-deploy/kolla-ansible-inventory` only exists on one of your cloud's control plane nodes. This is the node from which the deployment process took place. If you do not see this file within a control plane node, check the remaining nodes until you locate it. Execute any Kolla Ansible commands from the node that has the file `/etc/fm-deploy/kolla-ansible-inventory`.

Before Making Changes

Before any changes are made to the Kolla Ansible configuration, the following variables in `/etc/kolla/globals.yml` should not be modified, otherwise they will be overwritten:

```
api_interface
cluster_interface
dns_interface
docker_registry_insecure
kolla_enable_tls_external
kolla_external_fqdn
kolla_external_vip_address
kolla_internal_vip_address
migration_interface
network_interface
neutron_external_interface
openstack_region_name
storage_interface
tunnel_interface
```

Prepare Kolla Ansible Environment

Caution! – Ensure the control plane node from which the following commands are executed has the file `/etc/fm-deploy/kolla-ansible-inventory`.

Step 1 - Prepare environment

From the Docker container called `fm-deploy`, copy `/opt/kolla-ansible` to the local file system:

```
$ docker cp fm-deploy:/opt/kolla-ansible /opt/kolla-ansible
```


Step 2 – Prepare Python virtual environment

From the path `/opt/kolla-ansible`, Create a Python virtual environment then activate it:

```
$ cd /opt/kolla-ansible
$ virtualenv .venv
$ source .venv/bin/activate
```

Step 3 – Update pip

Next, update the virtual environment's `pip` command to the latest version, otherwise packages may not install as expected.

Update `pip` by using:

```
$ pip install --upgrade pip
```

Step 4 – Install Kolla Ansible using `requirements.txt`

Contained within `/opt/kolla-ansible` is a `requirements.txt` file. This file is used to install the appropriate release of Kolla Ansible as well as Ansible.

At the time of writing this guide, Private Clouds run the OpenStack Victoria release. The version of Kolla Ansible used must match the OpenStack version.

By default, this `requirements.txt` contains:

```
ansible>=2.9,<2.10,!2.9.10

# Use one of the following supported OpenStack versions:
#git+https://github.com/openstack/kolla-ansible@stable/train
#git+https://github.com/openstack/kolla-ansible@stable/ussuri
#git+https://github.com/openstack/kolla-ansible@stable/victoria
```

Before proceeding with this section, remove the comment symbol, `#`, from the beginning of appropriate Kolla Ansible GitHub link, which in this case is `stable/victoria`.

For example:

```
ansible>=2.9,<2.10,!2.9.10

# Use one of the following supported OpenStack versions:
git+https://github.com/openstack/kolla-ansible@stable/train
git+https://github.com/openstack/kolla-ansible@stable/ussuri
git+https://github.com/openstack/kolla-ansible@stable/victoria
```

To determine the release your cloud is using, as root from a hardware node, run `docker ps` to get a list of all Docker containers running. In that output is shown each OpenStack service's image, appended with the OpenStack version.

For example:

```
$ docker ps
cf9e23cef540   harbor.imhadmin.net/kolla/centos-binary-mariadb-clustercheck:victoria
c19964a28b4e   harbor.imhadmin.net/kolla/centos-binary-mariadb-server:victoria
```

In this output, the second column represents a Kolla Ansible image and there are two entries. The output has been truncated and you should see many more Docker containers running than two. At the end of the image name the version of OpenStack for which that image is built can be seen following the colon:

```
harbor.imhadmin.net/kolla/centos-binary-mariadb-clustercheck:victoria
```

The above indicates this OpenStack cloud is on the Victoria release.

Now, `requirements.txt` is prepared and can be used to install Kolla Ansible:

```
(.venv) $ pip install -r requirements.txt
```

Step 5 – Kolla Ansible is Ready for Use

At this step, you have everything prepared in order to use Kolla Ansible. Before proceeding, familiarize yourself with the available Kolla Ansible commands by running `kolla-ansible --help`. There are a number of functions possible, including maintenance tasks or making configuration changes to your Private Cloud.

If you were linked to this guide from the guide [`How to Copy and Restore Private Cloud OpenStack Service Databases and Configuration <operators_manual/day-3/create-openstack-service-backups.rst`_](#) you are now ready to use Kolla Ansible.

Caution! – Kolla Ansible’s configuration is set through the file `/etc/kolla/globals.yml` where some variables should not be changed. See the [Before Making Changes](#) section for more information.

An example configuration change made at this step is to enable TLS for Horizon. Enabling TLS falls outside the scope of this guide and for instruction on how to do so, see [How to Enable TLS for OpenStack](#).

For complete documentation regarding available Kolla Ansible commands, see [Operating Kolla](#).

References

- [OpenStack Kolla Ansible documentation](#)
- [Kolla Ansible Quick Start](#)

Next Steps

This concludes the steps needed to have a base understanding of how to use Kolla Ansible.

The following guides go into detail about specific things you can configure using Kolla Ansible, such as enabling TLS for Horizon or enabling Central Logging with an ELK stack:

- [Enable TLS for Horizon](#)
- [Enable Central Logging using ELK](#)

-
1. [Prerequisites](#)
 2. [Specify an External Fully Qualified Domain Name for Horizon](#)
 1. [Determine Public IP](#)
 2. [Configure an FQDN](#)
 3. [Apply Configuration Change Using Kolla Ansible](#)
 3. [Enable SSL Externally, Encrypting Horizon Traffic](#)
 1. [Modify Kolla Ansible Configuration](#)
 2. [Configure Root CA Bundle](#)
 3. [Prepare SSL File](#)
 4. [Specify SSL Certificate](#)
 5. [Enable External TLS](#)
 6. [Reconfigure Cloud using Kolla Ansible](#)
 4. [Reconfigure Ceph Cluster using Ceph Ansible](#)
 1. [Procedure](#)
 5. [Reference](#)

Prerequisites

- **Prepare Kolla Ansible:** This guide explains how to configure your cloud with an SSL using Kolla Ansible. Any time you work with Kolla Ansible, you must prepare a shell environment. For more, see [How to Prepare and Use Kolla Ansible](#). The remaining instruction assume this environment has been prepared. All commands are to be executed from the control plane node where this environment has been prepared.
- **Prepare Ceph Ansible:** This guide makes use of Ceph Ansible to reconfigure your cloud's Ceph cluster. When working with Ceph Ansible, you must first prepare a shell environment. For more, see [How to Prepare and Use Ceph Ansible](#). The portion of this guide that has to do with using Ceph Ansible assumes this environment has been prepared.
- **Root Access:** Root access to your cloud's control plane nodes is required.
- **Provide your own SSL Files:** This guide requires the private key, certificate and intermediate chain. Your certificate and intermediate chain may be combined as a "Full Chain".

Specify an External Fully Qualified Domain Name for Horizon

Before configuring your cloud with an SSL, a Fully Qualified Domain Name (FQDN) must be configured. The FQDN should have its DNS A record adjusted to point to the public IP of your cloud. This section explains how to prepare your cloud with an FQDN and how to determine the public IP of your cloud.

Determine Public IP

Within `/etc/kolla/globals.yml`, your cloud's public IP is defined by the key `kolla_external_vip_address`.

For example:

```
$ grep kolla_external_vip_address /etc/kolla/globals.yml
kolla_external_vip_address: 173.231.254.164
```

For this example, the FQDN selected should have a DNS A record pointed to **173.231.254.164**. Ensure your FQDN's A record is pointed to the value defined by `kolla_external_vip_address` within your cloud's `/etc/kolla/globals.yml`.

Configure an FQDN

To specify a domain name for Horizon, within `/etc/kolla/globals.yml`, set the value for `kolla_external_fqdn` to the domain of your choosing. **Note** your `/etc/kolla/globals.yml` may not have a line containing `kolla_external_fqdn`. If this line is not present, append it the file.

For example, to set the FQDN for a cloud to be **cloud.domain.com**, use:

```
kolla_external_fqdn: cloud.domain.com
```

Apply Configuration Change Using Kolla Ansible

With the FQDN configured, Kolla Ansible must be used to apply that configuration before proceeding with this guide. Before proceeding with this section, ensure you have [prepared a Kolla Ansible environment](#). Also ensure the node from which Kolla Ansible has been prepared contains the file `/etc/fm-deploy/kolla-ansible-inventory`, which is the Ansible inventory file for your cloud.

To configure the cloud to use this FQDN, use the inventory file `/etc/fm-deploy/kolla-ansible-inventory` and Kolla Ansible's `reconfigure` subcommand.

For example:

```
kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory reconfigure
```

Enable SSL Externally, Encrypting Horizon Traffic

This section outlines the steps required to install a signed SSL for your cloud's external, public network using Kolla Ansible.

Modify Kolla Ansible Configuration

Configure Root CA Bundle

OpenMetal private clouds are deployed with CentOS 8 as the operating system. Kolla Ansible needs to be updated to point to the root CA bundle for CentOS 8. To do so, modify `/etc/kolla/globals.yml` to reflect the location of the root CA bundle for CentOS 8:

```
openstack_cacert: '/etc/pki/tls/certs/ca-bundle.crt'
```

Prepare SSL File

Your SSL files need to be prepared into a single file of **PEM** format, including the private key, certificate, and intermediate chain. When preparing the file ensure the order of the information is the private key first, then the certificate, and finally the intermediate chain.

To prepare your SSL, concatenate the contents of the private key, the certificate, and the intermediary chain into single file located `/etc/kolla/certificates/<certificate-name>.pem`, where `<certificate-name>` is the name of the certificate file.

Additionally, ensure 0600 permissions are set for the certificate file:

```
chmod 0600 /etc/kolla/certificates/<certificate-name>.pem
```

As an example, here's truncated output including the headers of a prepared SSL PEM file:

```
-----BEGIN PRIVATE KEY----- <-- Private Key
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE----- <-- Certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE----- <-- Intermediary Certificate #1
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE----- <-- Intermediary Certificate #2
-----END CERTIFICATE-----
```

Ensure your SSL PEM file reflects the above ordering otherwise Kolla Ansible may fail to execute as expected.

Specify SSL Certificate

Next, in `/etc/kolla/globals.yml`, ensure the key `kolla_external_fqdn_cert` is set to the name of the certificate file. `{{ node_config }}` in this case is defined as `/etc/kolla`.

For example:

```
kolla_external_fqdn_cert: '{{ node_config }}/certificates/<certificate-name>.pem'
```

Enable External TLS

Within `/etc/kolla/globals.yml`, ensure `kolla_enable_tls_external` is set to 'yes':

```
kolla_enable_tls_external: 'yes'
```

Reconfigure Cloud using Kolla Ansible

The previous steps conclude the preparation of the SSL file and the Kolla Ansible configuration. Before proceeding with this step, ensure you have [prepared a Kolla Ansible environment](#).

Next, to configure the cloud to use this SSL, use the inventory file `/etc/fm-deploy/kolla-ansible-inventory` and Kolla Ansible's `reconfigure` subcommand.

For example:

```
kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory reconfigure
```

Reconfigure Ceph Cluster using Ceph Ansible

Additionally, the Ceph cluster needs to be reconfigured to update Swift to point to the Keystone HTTPS endpoint instead of the HTTP version. In this section we explain how to configure the new Swift endpoint using Ceph Ansible.

Procedure

First, ensure you have [prepared a Ceph Ansible environment](#).

Next, load `./group_vars/all.yml` in an editor and find the line containing the string `rgw keystone url:`. For this example, this line appears this way:

```
rgw keystone url: http://173.231.254.164:5000
```

This line tells Swift which URL to use to authenticate with Keystone. Since the cloud now has an SSL configured externally, the Keystone URL for Swift needs to include the HTTPS protocol.

In the file `./group_vars/all.yml`, ensure the line with `rgw keystone url:` now specifies HTTPS instead of HTTP. For example:

```
rgw keystone url: https://173.231.254.164:5000
```

Next, run Ceph Ansible, using:

```
ansible-playbook \  
  -i /etc/fm-deploy/ceph-inventory.yml \  
  --private-key /root/.ssh/fm-deploy \  
  /opt/ceph-ansible/site.yml
```

Reference

[TLS Documentation from Kolla Ansible for OpenStack Victoria](#)

How to Enable Elasticsearch and Kibana using Kolla Ansible

Introduction

An OpenStack cloud generates a large quantity of log messages. By default, there's no way to visually see a Private Cloud's log messages. To diagnose issues in logs requires using SSH and `grep`, which can be cumbersome due to the number of hosts and number of OpenStack services. Elasticsearch and Kibana (ELK) could be leveraged to see all of your cloud's log files from a single location in your browser. This feature set is not enabled by default. In this guide we walk you through how to enable the ELK stack for your Private Cloud using Kolla Ansible.

Prerequisites

Prepare Kolla Ansible

This guide explains how to configure your cloud using Kolla Ansible. Any time you work with Kolla Ansible, you must prepare a shell environment. For more, see [How to Prepare and Use Kolla Ansible](#).

All commands are to be executed from the control plane node in which Kolla Ansible has been prepared.

Root Access to OpenStack Control Plane

Root access to your cloud's control plane nodes is required.

How to Enable Central Logging

To enable the ELK stack, in `/etc/kolla/globals.yml` ensure the following is set:

```
enable_central_logging: 'yes'
```

To enforce data retention policies, enable Elasticsearch Curator with:

```
enable_elasticsearch_curator: 'yes'
```

Note! – Enabling Elasticsearch Curator can help prevent your cloud’s local disks from filling up by enforcing retention policies for Elasticsearch data.

Kolla Ansible generates a default configuration for Elasticsearch Curator which can later be found in the `elasticsearch_curator` Docker container as `elasticsearch_curator:/etc/elasticsearch-curator/actions.yml`.

Next, to deploy the configuration changes, use:

```
# kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory reconfigure
```

Prevent Root Disk from Filling

You can enable Elasticsearch Curator to enforce disk retention policies to prevent your cloud’s disks from filling up.

Reference

Kolla Ansible’s [Central Logging](#) guide.

1. Principle of Least Privilege

1. [Roles](#)
2. [Users, Groups, and Projects](#)
 1. [Users](#)
 2. [Groups](#)
 3. [Projects](#)

2. Updating Software

1. [Update Individual Instances](#)
 2. [Update Operating System Images](#)
 3. [Update Kolla Ansible Images](#)
 4. [Update Control Plane Nodes](#)
3. [Enabling TLS](#)
 4. [Security Groups](#)
 5. [SSH Authentication](#)
 6. [Additional Documentation](#)

Principle of Least Privilege

The philosophy around the principle of least privilege is a user account should only have privileges necessary to perform its intended function. For example, a financial analyst does not require admin rights or a software engineer should not need to access financial records.

The benefits of operating a system using the principle of least privilege are:

- **System stability** - Limiting the scope of changes reduces adverse effects of applications running on your system. This means that programs are less likely to perform actions that could crash your machine.
- **System security** - Limiting the system-wide changes can reduce blast radius. The blast radius is the maximum amount of damage an intruder can inflict after gaining access to your system.
- **Ease of deployment** - As a general rule, the fewer privileges an application needs to run, the easier it is to deploy to a larger environment.

When implementing the principles of least privilege, routine audits are important to maintain the security of your system and avoid privilege creep. Privilege creep is when a user or program is given more access or rights beyond

what is necessary to their job. In addition to conducting regular audits, consider starting all user and system accounts with the least privilege and enforcing the separation of privileges.

Users, Groups, Projects, and Roles

Within OpenStack, there is the ability to set individual access controls. These controls have different levels of granularity from setting a project and allocating resources to managing groups of individuals. There is even the ability to manage down to an individual user.

For information about how to create a user and project, see [Create an OpenStack User and Project in Horizon](#).

Users

When working in Openstack, there is only one administrative user. Using the principle of least privilege, you should create additional users when handling various tasks within your private cloud environment.

Groups

Groups are a collection of Users within OpenStack. Groups allow an administrator to manage permissions for several individual users at the same time. This method for managing access helps to avoid privilege creep as individuals can be removed from groups as their roles change within an organization.

Projects

Projects allow you to allocate resources within your private cloud. You can isolate users and individual projects to further control access to cloud resources.

Roles

Keystone roles are rights and privileges given to a user to perform tasks. Each service can require different types of roles to perform different actions. In addition, existing roles can be modified through the services policy.json file. Due to the containerization of OpenStack services these policy files will be found within the docker container of the service in /etc/service_name/policy.json. To find out more information about creating and managing roles visit [Create and manage roles](#).

Updating Software

All software running your Private Cloud is open source. Maintaining up-to-date software is important for the security of your private cloud. Among the major areas that require updating are individual instances, images, and the control plane nodes. Both operating system packages and applications require software updates over time as changes are made.

Update Individual Instances

For any instances your cloud contains, ensure the software within is kept up to date. This includes the operating system's software and the applications running within the instance. This software is typically managed through a package manager like `apt` or `dnf`.

Updating the software in individual instances can be accomplished by connecting to the instance through SSH. Once inside the instance, updates can be run using the package manager of the Linux distribution and rebooting.

Update Glance Images

Glance is the service in OpenStack responsible for managing operating system images. Once your private cloud is deployed, these images are not updated. Over time, these images can become vulnerable as it becomes necessary to maintain security updates and patches. We recommend routinely updating and managing these images within your existing OpenStack cluster. For further information on how to upload images visit [Manage and Upload Images in Horizon](#)

Update Kolla Ansible Images

Kolla Ansible relies on Docker images to deploy the various OpenStack containers. Over time, updates occur, causing existing images to be out of date. As part of regular cloud maintenance, these images can be updated, using Kolla Ansible. To learn more, see [How to Obtain Latest OpenStack Images using Kolla Ansible](#).

Update Control-Plane Nodes

For each hardware node, your cloud has, operating system updates should be performed as part of routine maintenance. These updates are best handled by the operating system's package manager, which for Private Clouds running CentOS 8, is `dnf`. For more information on how to update control plane nodes visit [OpenStack Hardware Node Maintenance](#).

Enabling TLS

TLS stands for Transport Layer Security protocol and is the successor to the SSL. Both TLS and SSL work in much the same way, using encryption to protect the transfer of information and data between two systems. Within OpenMetal, enabling TLS is very important for protecting login credentials for your Horizon Dashboard.

For more information on how to enable TLS within Horizon, visit [How to Enable TLS for OpenStack using Kolla Ansible](#).

Security Groups

In your private cloud, security groups act as a firewall to control inbound and outbound traffic to your instances. OpenStack has many different configurations for security groups that allow you to control the type of traffic to your instance as well as the port. The configuration options are known as rules. Rules define the types of ports that are available on your instance as well as the IP addresses that can connect to these specific ports.

For more information on how to create and manage security groups, see the title heading *Security Groups* <[operators_manual/day-1/horizon/create-first-instance.rst#security-groups](#)> in the [How to Create an Instance in OpenStack Horizon](#) guide.

SSH Authentication

Authentication in your private cloud is handled by SSH keys. These keys are injected into your control plane nodes from the moment of deployment of your cloud. Authentication keys can also be added to instances before being deployed. For additional security, consider restricting access to port 22 and limiting SSH access to control plane nodes.

OpenStack Security Advisor and Further Resources

OpenStack uses two different means for communicating security-related information: Advisories and Notes. OpenStack Security Advisories (OSSA) help to communicate fixes to severe security issues. OpenStack Security Notes (OSSN) provide general notices for potential vulnerabilities in design, deployment, and configuration. OpenStack does have a Vulnerability Management Team (VMT) for further information on how to contact visit [OpenStack Security](#).

A list of current Security Advisories for OpenStack can be found [Here](#).

For a more in-depth look into current best practices with OpenStack see [OpenStack Security Guide](#).

How to Prepare and Use Ceph Ansible

Introduction

Should you want to reconfigure your Private Cloud's Ceph cluster, you can do so using Ceph Ansible. In this guide, we explain how to prepare an environment from which Ceph Ansible can be used. Making specific configuration changes to your Ceph cluster is outside the scope of this guide.

Before Proceeding

WARNING! – Our current deployment system deploys a Private Cloud with a known working state. Should you deviate from this state by adjusting your cloud's Ceph configuration you can no longer safely use the functions in OpenMetal Central to add nodes to your cloud or add IP blocks. Should you use these functions, any custom configurations to Ceph will be reverted. We are working on rolling out a new deployment system allowing custom cloud configurations. We can still add new nodes and IP blocks to your cloud but must do so manually. Please reach out to your Account Manager should this apply to you.

Prerequisites

Root Access to OpenStack Control Plane

Root access to your cloud's control plane nodes is required.

Preparation

To prepare Ceph Ansible:

```
docker cp fm-deploy:/opt/ceph-ansible /opt/ceph-ansible
chmod 700 /opt/ceph-ansible
cd /opt/ceph-ansible
virtualenv .venv
source .venv/bin/activate
pip install -r requirements.txt
pip install six
```

Deploy a Ceph Cluster:

```
ansible-playbook \
  -i /etc/fm-deploy/ceph-inventory.yml \
  --private-key /root/.ssh/fm-deploy \
  /opt/ceph-ansible/site.yml
```

Attempt to repair a broken Ceph cluster:

```
ansible-playbook \
  -i /etc/fm-deploy/ceph-inventory.yml \
  --private-key /root/.ssh/fm-deploy \
  /opt/ceph-ansible/site.yml
```

1. Use Watcher to Consolidate your Cloud's Workload

1. Cloud State Before Watcher is Applied
2. **How to use Watcher's VM Workload Consolidation Strategy**
 1. Obtain List of Goals
 2. List Strategies Available for a Goal
 3. Create Audit Template
 4. Execute Audit
 5. Retrieve Action Plan
 6. Review Action Plan
 7. Execute Action Plan
3. Cloud State After Watcher is Applied

Use Watcher to Consolidate your Cloud's Workload

This section demonstrates use of Watcher's [VM Workload Consolidation Strategy](#). The commands and their output are recorded to provide an example Watcher demonstration. We provide the state of the cloud prior to and after applying Watcher.

Cloud State Before Watcher is Applied

The following screenshot shows the state of the hypervisors and the count of instances associated with each compute node.

Hostname	Type	VCPUs (used)	VCPUs (total)	RAM (used)	RAM (total)	Local Storage (used)	Local Storage (total)	Instances
competent-coypu.local	OEMU	22	16	84GB	125.3GB	544GB	11.6TB	20
focused-capybara.local	OEMU	27	16	125GB	125.3GB	349GB	11.6TB	18
lovely-ladybug.local	OEMU	28	16	105GB	125.3GB	634GB	11.6TB	26
relaxed-flamingo.local	OEMU	18	16	68GB	125.3GB	436GB	11.6TB	16

Figure 1: Hypervisor list and instance count prior to applying Watcher

This is a cloud with four compute nodes and a relatively even distribution of instances spread across them.

How to use Watcher's VM Workload Consolidation Strategy

Step 1: Obtain List of Goals

First, obtain a list of available goals with:

```
openstack optimize goal list
```

For example:

```
# openstack optimize goal list
+-----+-----+-----+
| UUID                                     | Name                               | Display name |
+-----+-----+-----+
| 05260080-9aca-40e3-9ff5-f6d8f398d671   | airflow_optimization              | Airflow Optimization |
| d3618295-7dd2-4174-83ee-8cd466b3381a   | cluster_maintaining               | Cluster Maintaining  |
| 4488913d-b0d6-44eb-b6c6-8f1b4c6c7578   | dummy                              | Dummy goal           |
| cb10ae8f-alf5-41e4-ac6a-2e2dfbe63648   | hardware_maintenance              | Hardware Maintenance |
| 6f676623-71b8-4eaa-a36b-fb133120db42   | noisy_neighbor                    | Noisy Neighbor       |
| 6c0ae27a-d978-46e9-9562-67ef7ee00e65   | saving_energy                     | Saving Energy         |
| 916bb6dc-84df-495e-8b40-1bfa868d93e1   | server_consolidation              | Server Consolidation |
| 5aed3f2b-7d5d-48af-b69c-35afc886bc10   | thermal_optimization              | Thermal Optimization |
| debf193e-50a8-4c90-a0b6-d82fb78b6d98   | unclassified                       | Unclassified         |
| fb940ef6-47bd-4c82-8e3a-caf9de813b03   | workload_balancing                | Workload Balancing   |
+-----+-----+-----+
```

This example makes use of the **Server Consolidation** goal available from the above list. **Note** that the following command line examples reference this goal as `server_consolidation`.

Step 2: List Strategies Available for a Goal

Next, obtain a list of available strategies for the **Server Consolidation** goal, using:

```
openstack optimize strategy list --goal <goal-uuid-or-name>
```

Replace `<goal-uuid-or-name>` with the appropriate goal.

For example:

```
# openstack optimize strategy list --goal server_consolidation
+-----+-----+-----+
| UUID                                     | Name                               | Display name |
+-----+-----+-----+
```

eae7ae0f-1498-45ac-a226-248739a79785	basic	Basic offline consoli
b0f2838e-0eb2-4baf-8976-62744897827b	node_resource_consolidation	Node Resource Consoli
4ad7dbdb-9c90-4208-8dc0-e476c803519c	vm_workload_consolidation	VM Workload Consolida

Here, we can see the **Server Consolidation** has three strategies that can be used to accomplish the goal. For this case, we are using the `vm_workload_consolidation` strategy.

Step 3: Create Audit Template

Next, create an audit template based on the previously selected goal and strategy using:

```
openstack optimize audittemplate create <template-name> <goal> \
  --strategy <strategy>
```

- `<template-name>`: Specify a name for the audit template
- `<goal>`: Specify a goal
- `<strategy>`: Specify a strategy

For example, the following creates an audit template called **server_consolidation-template** based on the goal `server_consolidation` and strategy `vm_workload_consolidation`:

```
# openstack optimize audittemplate create server_consolidation-template server_consolidation
```

Field	Value
UUID	234069e4-d03a-4a63-af54-02763056ac55
Created At	2022-01-21T18:22:09.385581+00:00
Updated At	None
Deleted At	None
Description	None
Name	server_consolidation-template
Goal	server_consolidation
Strategy	vm_workload_consolidation
Audit Scope	[]

Step 4: Execute Audit

Run an audit based on the audit template to generate an action plan:

```
openstack optimize audit create -a server_consolidation-template
```

For example:

```
# openstack optimize audit create -a server_consolidation-template
```

Field	Value
UUID	50b571c6-faff-4b0e-803d-558708bc5ec5
Name	vm_workload_consolidation-2022-01-21T18:23:37.360815
Created At	2022-01-21T18:23:37.387673+00:00
Updated At	None
Deleted At	None
State	PENDING
Audit Type	ONESHOT
Parameters	{'period': 3600, 'granularity': 300}
Interval	None
Goal	server_consolidation
Strategy	vm_workload_consolidation
Audit Scope	[]

Table of Contents

Auto Trigger	False
Next Run Time	None
Hostname	None
Start Time	None
End Time	None
Force	False

This step may take some time to complete. You can use `openstack optimize audit show <audit-uuid>` to get the status of the audit. For example:

```
# openstack optimize audit show 50b571c6-faff-4b0e-803d-558708bc5ec5
```

Field	Value
UUID	50b571c6-faff-4b0e-803d-558708bc5ec5
Name	vm_workload_consolidation-2022-01-21T18:23:37.360815
Created At	2022-01-21T18:23:37+00:00
Updated At	2022-01-21T18:55:15+00:00
Deleted At	None
State	SUCCEEDED
Audit Type	ONESHOT
Parameters	{'period': 3600, 'granularity': 300}
Interval	None
Goal	server_consolidation
Strategy	vm_workload_consolidation
Audit Scope	[]
Auto Trigger	False
Next Run Time	None
Hostname	focused-capybara.local
Start Time	None
End Time	None
Force	False

The audit is complete when the `State` field reflects `SUCCEEDED`.

Step 5: Retrieve Action Plan

Retrieve the action plan generated by the audit:

```
openstack optimize actionplan list --audit <audit-uuid>
```

Replace `<audit-uuid>` with the UUID of the of audit.

For example:

```
# openstack optimize actionplan list --audit 50b571c6-faff-4b0e-803d-558708bc5ec5
```

UUID	Audit	State
2abb3cae-99e2-4339-b952-0f52db59155d	50b571c6-faff-4b0e-803d-558708bc5ec5	RECOMMENDED

Step 6: Review Action Plan

Review the actions recommended by the audit:

```
openstack optimize action list --action-plan <action-plan-uuid>
```

Replace `<action-plan-uuid>` with the UUID of the action plan.

For example:

```
# openstack optimize action list --action-plan 2abb3cae-99e2-4339-b952-0f52db59155d
+-----+-----+
| UUID                                     | Parents |
+-----+-----+
| 7caa5e76-0815-467d-8073-c1b1213e5e70 | []      |
| 29c3a118-3aa7-4445-a5c0-ddc74d58a609 | ['7caa5e76-0815-467d-8073-c1b1213e5e70'] |
| 75cfa548-7679-4271-8d10-d8583beb7f54 | ['7caa5e76-0815-467d-8073-c1b1213e5e70'] |
| 47ae3e44-8bc8-4272-8582-1a24640c5f97 | ['29c3a118-3aa7-4445-a5c0-ddc74d58a609', '75cfa548-7679-4271-8d10-d8583beb7f54'] |
| 29a12a53-cd2d-4d39-8284-4273d07a92ee | ['29c3a118-3aa7-4445-a5c0-ddc74d58a609', '75cfa548-7679-4271-8d10-d8583beb7f54'] |
+-----+-----+
```

This action plan indicates the Nova service's state will be adjusted and that a number of instances are to be migrated. Note that some of the actions listed in the above output have been truncated for brevity.

Step 7: Execute Action Plan

If all actions presented in the action plan are reasonable, you can have Watcher execute the action plan for you.

To execute the actions:

```
openstack optimize actionplan start <action-plan-uuid>
```

Replace <action-plan-uuid> with the UUID of the action plan.

For example:

```
# openstack optimize actionplan start 2abb3cae-99e2-4339-b952-0f52db59155d
+-----+-----+
| Field                                | Value |
+-----+-----+
| UUID                                 | 2abb3cae-99e2-4339-b952-0f52db59155d |
| Created At                           | 2022-01-21T18:55:15+00:00 |
| Updated At                            | 2022-01-21T19:46:00+00:00 |
| Deleted At                            | None |
| Audit                                 | 50b571c6-faff-4b0e-803d-558708bc5ec5 |
| Strategy                              | vm_workload_consolidation |
| State                                 | PENDING |
| Efficacy indicators                   | [{'name': 'compute_nodes_count', 'description': 'The total number of enabled compute nodes.'}] |
| Global efficacy                       | [{'name': 'released_nodes_ratio', 'description': 'Ratio of released compute nodes to be released.'}] |
| Hostname                              | None |
+-----+-----+
```

Get status of the action plan using `openstack optimize actionplan show <action-plan-uuid>`.

For example:

```
# openstack optimize actionplan show 2abb3cae-99e2-4339-b952-0f52db59155d
+-----+-----+
| Field                                | Value |
+-----+-----+
| UUID                                 | 2abb3cae-99e2-4339-b952-0f52db59155d |
| Created At                           | 2022-01-21T18:55:15+00:00 |
| Updated At                            | 2022-01-21T19:46:00+00:00 |
| Deleted At                            | None |
| Audit                                 | 50b571c6-faff-4b0e-803d-558708bc5ec5 |
| Strategy                              | vm_workload_consolidation |
| State                                 | ONGOING |
| Efficacy indicators                   | - Description: The total number of enabled compute nodes.
|                                         Name: compute_nodes_count
|                                         Unit: null
|                                         Value: 4.0
|                                         - Description: The number of compute nodes to be released.
|                                         Name: released_compute_nodes_count
|                                         Unit: null
+-----+-----+
```

	Value: 1.0 - Description: The number of VM migrations to be performed. Name: instance_migrations_count Unit: null Value: 35.0
Global efficacy	Released_nodes_ratio: 25.00 %
Hostname	lovely-ladybug.local

Here is the output of the action plan after it has finished executing:

```
# openstack optimize actionplan show 2abb3cae-99e2-4339-b952-0f52db59155d
```

Field	Value
UUID	2abb3cae-99e2-4339-b952-0f52db59155d
Created At	2022-01-21T18:55:15+00:00
Updated At	2022-01-21T22:09:49+00:00
Deleted At	None
Audit	50b571c6-faff-4b0e-803d-558708bc5ec5
Strategy	vm_workload_consolidation
State	SUCCEEDED
Efficacy indicators	- Description: The total number of enabled compute nodes. Name: compute_nodes_count Unit: null Value: 4.0 - Description: The number of compute nodes to be released. Name: released_compute_nodes_count Unit: null Value: 1.0 - Description: The number of VM migrations to be performed. Name: instance_migrations_count Unit: null Value: 35.0
Global efficacy	Released_nodes_ratio: 25.00 %
Hostname	lovely-ladybug.local

Cloud State After Watcher is Applied

The following screenshot shows the spread of this cloud's instances after Watcher's VM Workload Consolidation Strategy has been applied.

Hostname	Type	VCPUs (used)	VCPUs (total)	RAM (used)	RAM (total)	Local Storage (used)	Local Storage (total)	Instances
competent-coypu.local	OEMU	3	16	8GB	125.3GB	31GB	11.6TB	1
focused-capibara.local	OEMU	42	16	185GB	125.3GB	754GB	11.6TB	33
lovely-ladybug.local	OEMU	48	16	185GB	125.3GB	1.1TB	11.6TB	46
relaxed-flamingo.local	OEMU	2	16	4GB	125.3GB	4GB	11.6TB	0

Figure 2: Hypervisor list and instance count after Watcher has been applied

For this case, Watcher determined it could run the same compute workload with three hypervisors instead of four. Instances were live migrated to free up a single compute host. Then, Watcher disabled the compute service for the freed node.

1. [Prerequisites](#)
2. [Symptoms of a RabbitMQ Problem](#)
3. [How Check your RabbitMQ Cluster's Status](#)
4. [List RabbitMQ Queues](#)
5. [Addressing Network Partitions](#)
6. [Redeploy RabbitMQ Cluster](#)

Prerequisites

This guide requires root access over SSH to your Private Cloud's control plane nodes. You should be comfortable using the command line when troubleshooting RabbitMQ.

Symptoms of a RabbitMQ Problem

Actions hang in Horizon or when using OpenStackClient. For example when an instance is created but never completes, you may have an issue with RabbitMQ. Additionally, say you try to delete something in Horizon, but the item hangs indefinitely and is never actually deleted. This indicates a probable issue with RabbitMQ.

How Check your RabbitMQ Cluster's Status

The command `rabbitmqctl cluster_status` is used to report the status of your cloud's RabbitMQ cluster. This is a quick way to determine the health of the RabbitMQ cluster.

RabbitMQ is deployed into Docker containers which changes the way you interact with the cluster. To check the status of your cloud's RabbitMQ cluster, from a control plane node as root, execute:

```
docker exec -it rabbitmq rabbitmqctl cluster_status
```

List RabbitMQ Queues

List queue name, current count of queued messages, and count of messages consumed for a private cloud's RabbitMQ cluster and put the output into a table:

```
docker exec -it rabbitmq rabbitmqctl list_queues -p / name messages consumers --formatter pr
```

For example, listing the first few queues:

```
# docker exec -it rabbitmq rabbitmqctl list_queues -p / name messages consumers | head -20
Timeout: 60.0 seconds ...
Listing queues for vhost / ...
name      messages      consumers
heat-engine-listener_fanout_7025924940fc4f11b89ed31afe5a642c      0      1
scheduler_fanout_257bb53015e3478db4c7c36236923300      0      1
reply_7c1ef96102e643a8bd8827f7191cf4cc      0      1
reply_742233f6bbbb47e196d53e834dff912      0      0
heat-engine-listener.7b6e3335-0745-4c38-a78c-93eca7f9b336      0      0
watcher.applier.control_fanout_97961841bc9e4304b1dbfe6da039ee0c      0      1
reply_733a0bc0a2e248aea9c307d2dbb6b88b      0      1
neutron-vo-SecurityGroupRule-1.0_fanout_a6dad4171bb94d0aa97ae30b218b779a      0      1
engine_fanout_917e9c14d41d4d69be11122b1bd28485      0      1
reply_b9fa7b24616f40eb9be6e4cb70ebd9d2      0      0
q-l3-plugin_fanout_190b34d25d7444ecb3d901abb971f93f      0      1
reply_849a66b5d7e74d099d89ed842832a7ae      0      1
magnum-conductor.reobzilz72y4      0      0
magnum-conductor_fanout_97e11c7c5ebc46fabfe57adf45a05b11      0      1
reply_faad7408773c4e7ebbaa9abfb9c0534a      0      1
cinder-volume.lovely-ladybug.local@rbd-1_fanout_1d3ef02f49f148618fcb36163687b9cd      0      0
engine_fanout_28b15b87d3ec4541b0d7731b928f6852      0      1
```

RabbitMQ and Network Partitions

Although uncommon with a stable cloud platform, with RabbitMQ, it is possible for the cluster to go into a **Network Partition**. To better understand and resolve these types of issues, see RabbitMQ's [Clustering and Network Partitions](#) guide.

Redeploy RabbitMQ Cluster

In worst case scenarios it is possible to redeploy your cloud's RabbitMQ cluster. This tends to be a last resort effort to get RabbitMQ functioning again. Kolla Ansible is used to redeploy a cloud's RabbitMQ cluster. For more, see [How to Redeploy RabbitMQ Cluster using Kolla Ansible](#).

1. Prerequisites

1. [Prepare Kolla Ansible](#)
2. [Root Access to OpenStack Control Plane](#)
2. [How to Redeploy RabbitMQ](#)

Prerequisites

Prepare Kolla Ansible

This guide explains how to configure your cloud using Kolla Ansible. Any time you work with Kolla Ansible, you must prepare a shell environment. For more, see [How to Prepare and Use Kolla Ansible](#).

All commands are to be executed from the control plane node in which Kolla Ansible has been prepared.

Root Access to OpenStack Control Plane

Root access to your cloud's control plane nodes is required.

How to Redeploy RabbitMQ

For each RabbitMQ cluster member, run:

```
docker stop rabbitmq
cp -Rv /var/lib/docker/volumes/rabbitmq/_data/mnesia{, .bk$(date +%F)}
rm -rfv /var/lib/docker/volumes/rabbitmq/_data/mnesia/
```

Then, use Kolla Ansible's `deploy` function, targeting RabbitMQ:

```
kolla-ansible -i /etc/fm-deploy/kolla-ansible-inventory deploy --tags rabbitmq
```

1. Prerequisites

1. [Root Access to OpenStack Control Plane](#)
2. [Get Ceph's Status](#)
3. [Ceph Log Files](#)
4. **Common Issues**

1. Clock Skew

1. [Confirm Ceph's Health](#)
2. [Examine Chrony Logs](#)
3. [Addressing Clock Skew](#)

5. References

Prerequisites

Root Access to OpenStack Control Plane

Root access to your cloud's control plane nodes is required.

Get Ceph's Status

In most troubleshooting cases, you can get an overview of your Ceph cluster by checking its status. To check your Ceph cluster's status, use `ceph status`.

For example:

```
# ceph status
cluster:
  id:          34fa49b3-fff8-4702-8b17-4e8d873c845f
  health: HEALTH_WARN
             clock skew detected on mon.focused-capybara, mon.lovely-ladybug
             2 daemons have recently crashed

services:
  mon: 3 daemons, quorum relaxed-flamingo,focused-capybara,lovely-ladybug (age 5d)
  mgr: relaxed-flamingo(active, since 5d), standbys: focused-capybara, lovely-ladybug
  osd: 4 osds: 4 up (since 5d), 4 in (since 13d)
  rgw: 3 daemons active (focused-capybara.rgw0, lovely-ladybug.rgw0, relaxed-flamingo.rgw0)

task status:

data:
  pools:   13 pools, 337 pgs
  objects: 110.16k objects, 388 GiB
  usage:   1.1 TiB used, 11 TiB / 12 TiB avail
  pgs:     337 active+clean

io:
  client:  381 KiB/s rd, 1.2 MiB/s wr, 444 op/s rd, 214 op/s wr
```

Ceph Log Files

Ceph's log files are stored in `/var/log/ceph/` within each control plane node.

For example, listed are all log files for host `focused-capybara`:

```
# ls -l /var/log/ceph/*.log
/var/log/ceph/ceph.audit.log
/var/log/ceph/ceph.log
/var/log/ceph/ceph-mgr.focused-capybara.log
/var/log/ceph/ceph-mon.focused-capybara.log
/var/log/ceph/ceph-osd.1.log
/var/log/ceph/ceph-rgw-focused-capybara.rgw0.log
/var/log/ceph/ceph-volume.log
```

An OpenMetal Ceph cluster is comprised of several services: Ceph's Manager, Monitor, OSD, and RADOSGW

Ceph has a primary log file, log files for each service, and additional log files.

For example:

- **Primary Log File:** `/var/log/ceph/ceph.log`
- **Ceph Monitor Log File:** `/var/log/ceph/ceph-mon.focused-capybara.log`
- **Ceph RADOSGW Log File:** `/var/log/ceph/ceph-rgw-focused-capybara.rgw0.log`

If you are unsure which Ceph service's log to look through, consider starting with the primary log file, `/var/log/ceph/ceph.log`.

Common Issues

Clock Skew

Ceph has a number of health checks, including one for clock skew, called `MON_CLOCK_SKEW`. For more, see Ceph's [Health Checks](#) guide and look for the text **MON_CLOCK_SKEW**. Ceph in our configuration uses `chronyd` to sync each node's clock. Kolla Ansible is responsible for installing and configuring `chronyd` into a Docker container for each Ceph Monitor node. To administer `chronyd` you must do so through Docker.

Confirm Ceph's Health

To confirm the status of this health check, execute `ceph status` and examine the output.

For example:

```
cluster:
  id:      34fa49b3-fff8-4702-8b17-4e8d873c845f
  health:  HEALTH_WARN
           clock skew detected on mon.focused-capybara, mon.lovely-ladybug
[...output truncated...]
```

Alternatively, execute `ceph health detail` to only see the status of health checks.

For example:

```
HEALTH_WARN clock skew detected on mon.focused-capybara, mon.lovely-ladybug
[WRN] MON_CLOCK_SKEW: clock skew detected on mon.focused-capybara, mon.lovely-ladybug
      mon.focused-capybara clock skew 0.663159s > max 0.05s (latency 0.000399254s)
      mon.lovely-ladybug clock skew 0.368233s > max 0.05s (latency 0.000385143s)
```

Examine Chrony Logs

From here, you may want to examine the logs for each `chrony` Docker instance running.

For example:

```
docker logs chrony
```

Alternatively, consider viewing logs on the local file system for `chrony` via `/var/log/kolla/chrony/`.

Addressing Clock Skew

There may be a number of methods to addressing clock skew. In this example, we outline addressing this issue by restarting `chrony` for each node.

To address the `MON_CLOCK_SKEW` for the example output in this section, the Docker container `chrony` was restarted for each node. For example:

```
# docker restart chrony
chrony
```

Next, perform the same Ceph health check as before to confirm the status. For example:

```
# ceph health detail
HEALTH_OK
```

If the clock skew issue is no longer present, you should see the status of `HEALTH_OK` assuming there are no other issues with the Ceph cluster.

Note! – Restarting `chrony` may be a heavy handed approach to addressing the issue. Consider alternatively making use of `chronyc`'s `tracking`, `sources`, and `sourcstats` subcommands to diagnose clock skew issues.

References

- Ceph's [Troubleshooting](#)
- Ceph's [Troubleshooting Monitors](#)

1. Prerequisites

1. [Root Access to OpenStack Control Plane](#)
2. [Elasticsearch and Kibana](#)
3. [Kolla Ansible Log Locations](#)
4. [Determining the Correct Log](#)
5. [Determining the Correct Host](#)
6. [Viewing Logs](#)

Prerequisites

Root Access to OpenStack Control Plane

Root access to your cloud's control plane nodes is required.

Elasticsearch and Kibana

This guide focuses on how to manually look through logs, however Elasticsearch and Kibana, commonly referred to as an ELK stack, are often used to aggregate and view logs in a visual manner. With a properly configured ELK stack, you can view all of your cloud's logs from a single location, visually.

For more information about enabling an ELK stack in your cloud, see [How to Enable Elasticsearch and Kibana using Kolla Ansible](#).

Kolla Ansible Log Locations

Private Clouds are deployed using Kolla Ansible, an OpenStack deployment system. This system deploys all OpenStack services into Docker containers. Each service's log file is then stored as `/var/log/kolla/<service-name>`, where `<service-name>` is an OpenStack service, like Neutron for example.

For example, to see all logs associated with the Neutron service, `ls` the directory `/var/log/kolla/neutron`:

```
# ls /var/log/kolla/neutron/*.log
/var/log/kolla/neutron/dnsmasq.log
/var/log/kolla/neutron/neutron-dhcp-agent.log
/var/log/kolla/neutron/neutron-l3-agent.log
/var/log/kolla/neutron/neutron-metadata-agent.log
/var/log/kolla/neutron/neutron-metering-agent.log
/var/log/kolla/neutron/neutron-netns-cleanup.log
/var/log/kolla/neutron/neutron-openvswitch-agent.log
/var/log/kolla/neutron/neutron-server.log
/var/log/kolla/neutron/privsep-helper.log
```

Determining the Correct Log

How do you know which log to look in for an issue? Which host should you be in? It is generally useful to know which OpenStack services are running and what their purposes are before determining which log file to examine to troubleshoot an issue. For a list of OpenStack services and their purpose, see the [OpenStack Components](#) page.

When diagnosing issues, consider the services associated with the action that may be failing. For example, if looking into an issue with creating a volume, consider looking at Cinder's various logs.

Determining the Correct Host

By default, your cloud has three control plane nodes. Each of these nodes has very similar logs, and typically, only one of the nodes is recording log events for a specific service. Due to this, you may need to examine all three host's logs in real time, replicate the issue, then see if any of those logs recorded any events.

To view the logs for all hosts at the same time, consider using a terminal multiplexer, especially one where you can issue the same commands in multiple SSH connections. The application `tmux` is an example of a terminal multiplexer.

Viewing Logs

There are several native ways to view the contents of a log file.

Applications like `less`, `nano`, and `vim` will suffice. For more colorful output, consider using an application like `lnav`, which has proven especially useful for examining unfamiliar logs.

1. **Overview of Heat Orchestration**
 1. Architecture
2. **Heat Orchestration Template Components**
 1. Template Version
 2. Description
 3. Parameters
 4. Resources
 5. Output
3. **Sample Heat Orchestration Template**
 1. Attributes
4. Deploying a Heat Template in Horizon
5. Viewing Recently Deployed Stacks in Horizon

Overview of Heat Orchestration

Orchestration tools utilize automation to deploy, manage, scale, and network containers. Within OpenStack, the primary orchestration component is Heat. Heat uses an orchestration engine for the automation of resources, infrastructure, applications, and services. These created resources are referred to as Stacks. Heat Stacks are deployed using Heat Orchestration Templates also referred to as HOT templates. HOT templates are created in YAML that passes instructions to the Heat Engine specifying the resources to be deployed.

How to View Heat Stacks in Horizon

To view all current Stacks within your OpenMetal Cloud, navigate to **Project -> Orchestration -> Stacks**. Within this section, you can view all your current stacks as well as launch and preview HOT templates.

Table of Contents

Stack Name	Created	Updated	Status	Actions
test_stack2	1 month	Never	Check Complete	Check Stack
glance-images	1 month, 1 week	Never	Create Complete	Check Stack
coe	3 months, 1 week	Never	Create Complete	Check Stack
keystone-swift-endpoints	3 months, 1 week	Never	Create Complete	Check Stack
keystone-swift-user	3 months, 1 week	Never	Create Complete	Check Stack
neutron-provider-network	3 months, 1 week	Never	Create Complete	Check Stack
neutron-security-groups	3 months, 1 week	Never	Create Complete	Check Stack
nova-flavors	3 months, 1 week	Never	Create Complete	Check Stack

Architecture

Heat API receives rest API calls from the following sources: Horizon, HOT Templates, and through the Heat Client. The Heat API is then passed to RabbitMQ or another message broker which activates the Heat Orchestration Engine. The Heat Orchestration Engine then connects to resources including Nova, Glance, Neutron, and Cinder and parses instructions from the HOT template for functions and parameters. These functions and parameters are then used to create resources known as stacks.

Heat Orchestration Template Components

Central to the deployment of Heat and the creation of Stacks is the use of templates. Heat Templates are written in YAML with the following structure:

Template Version

The template version is formatted as date and corresponds to different versions of heat. The template version tells Heat how the template will be structured, what features can be validated, and what resources can be created. Currently, the latest template version that is supported is 2018-08-31 (rocky).

Description

Although optional, a description is highly recommended as it outlines the intended function of the template. This is particularly useful for templates that are not frequently used or are intended to be shared among various members of a team or project.

Parameters

Parameters are optional but allow a user to pass additional values to the template. These values are used to overwrite any default settings that are specified in the resources section.

Resources

Resources are what are ultimately created by the Heat Orchestration Engine. These values can be default values or input from the parameters above.

Output

Output in a Heat Template is optional. However, you can pass values from Heat upon execution of the template. These values are accessed through Heat API or client tools.

Sample Heat Orchestration Template

The following is an example of a Heat Orchestration Template to deploy an instance. In this example, all resources receive their values fed separately as parameters. When formatting parameters, each parameter is grouped in a nested block. The first line contains the name of the parameter and additional attributes are grouped below each element.

Attributes

- Parameter Name: Name of Parameter
- Type: Required value that supports the following types: **string**, **comma_delimited_list**, **json**, **boolean**
- Label: Optional attribute, creates a name for the parameter for additional readability
- Description: Optional attribute, creates a description for parameter
- Default: Optional attribute, default values for parameters if input is not specified by the user

Example Heat Orchestration Template:

```
heat_template_version: 2018-03-02

description: Simple template to deploy a single instance

parameters:
  key_name:
    type: string
    label: Key Name
    description: Name of key-pair to be used for compute instance
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used
  network_name:
    type: string
    description: The network to be used
  security_groups:
    type: comma_delimited_list
    description: Name of the Security Group

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image_id }
      networks:
        - network: { get_param: network_name }
      flavor: { get_param: instance_type }
      security_groups: { get_param: security_groups }
```

Deploying a Heat Template in Horizon

To deploy a heat template in Horizon, first, check to see that you have selected the appropriate project for your deployment and then navigate to **Project -> Orchestration -> Stacks** and locate **Launch Stack** near the top right.

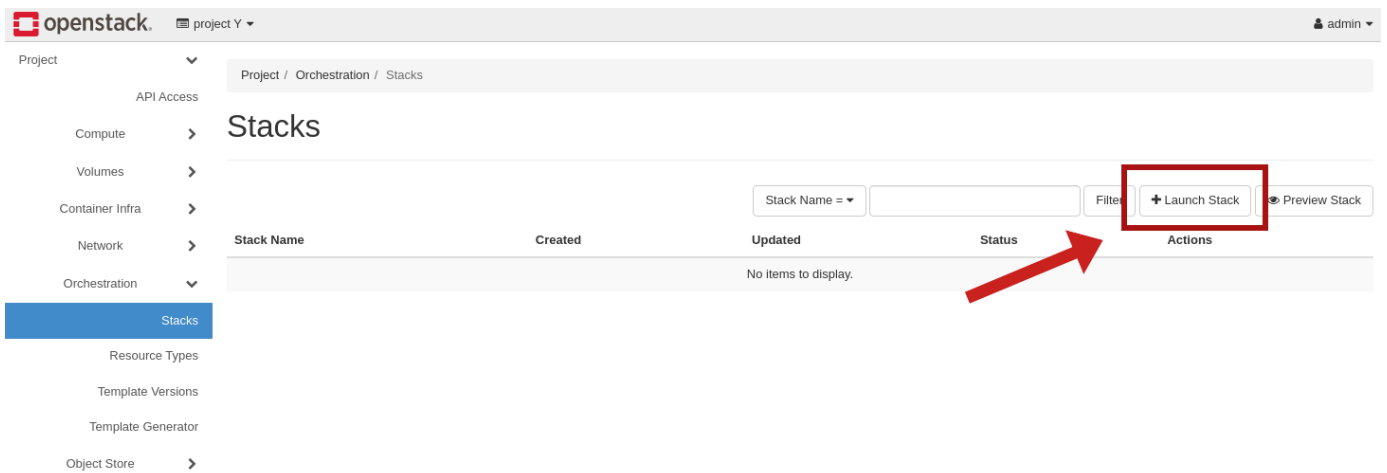


Figure 1: Launch Stack

For this example, we have uploaded the template example as a file called `hot_example.yml` using the **Choose File** button. After uploading your example template, click **Next**.

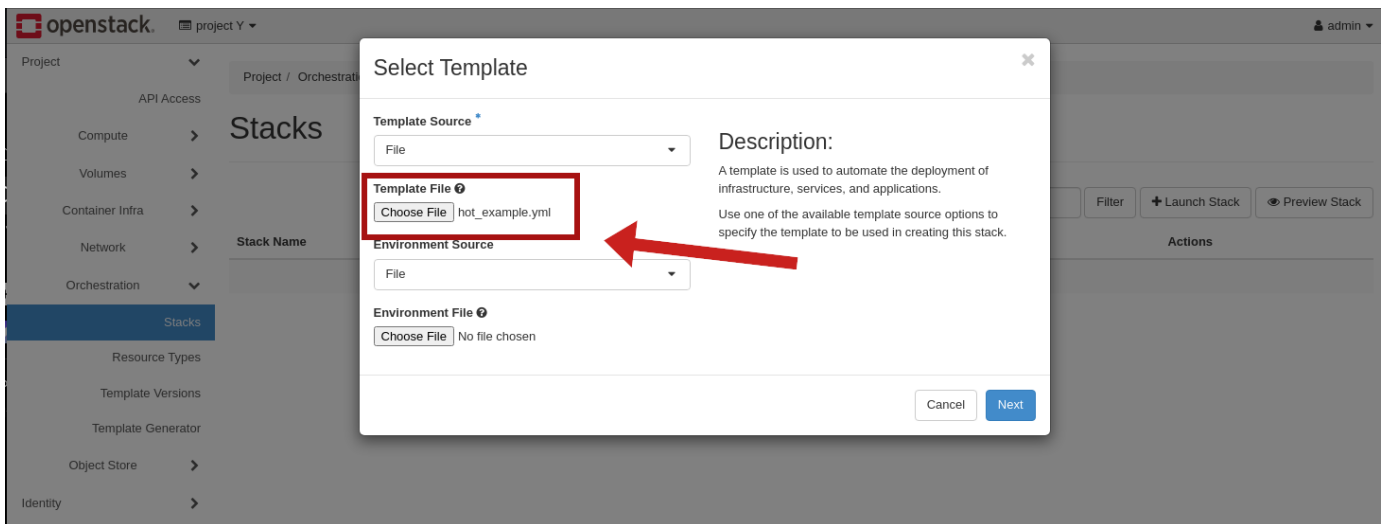


Figure 2: Select Template

For the example template listed above, the following parameters are needed to launch a running instance:

- Stack Name
- Image
- Instance Type
- Key
- Network
- Security groups

In the Launch stack window, fill out the following parameters with the type of stack you wish to deploy.

Note: Instance type refers to the flavor of instance you wish to set up.

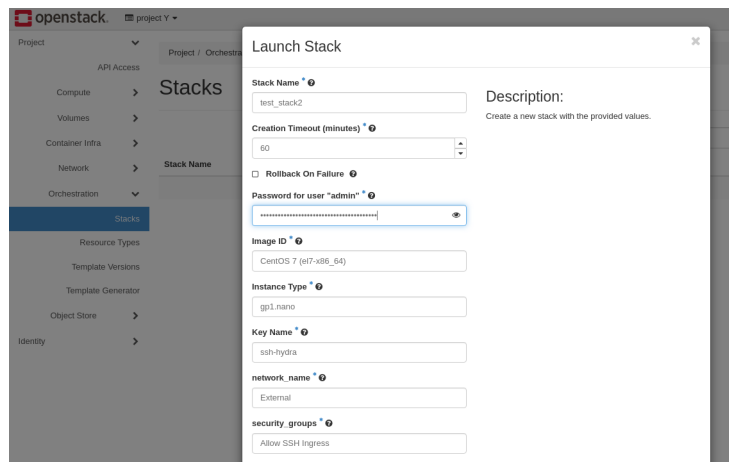


Figure 3: Launch Stack

Viewing Recently Deployed Stacks in Horizon

To view recently deployed stacks in Horizon, switch to the appropriate project where the stack was created. Then navigate to **Project -> Orchestration -> Stacks** to view all created stacks.

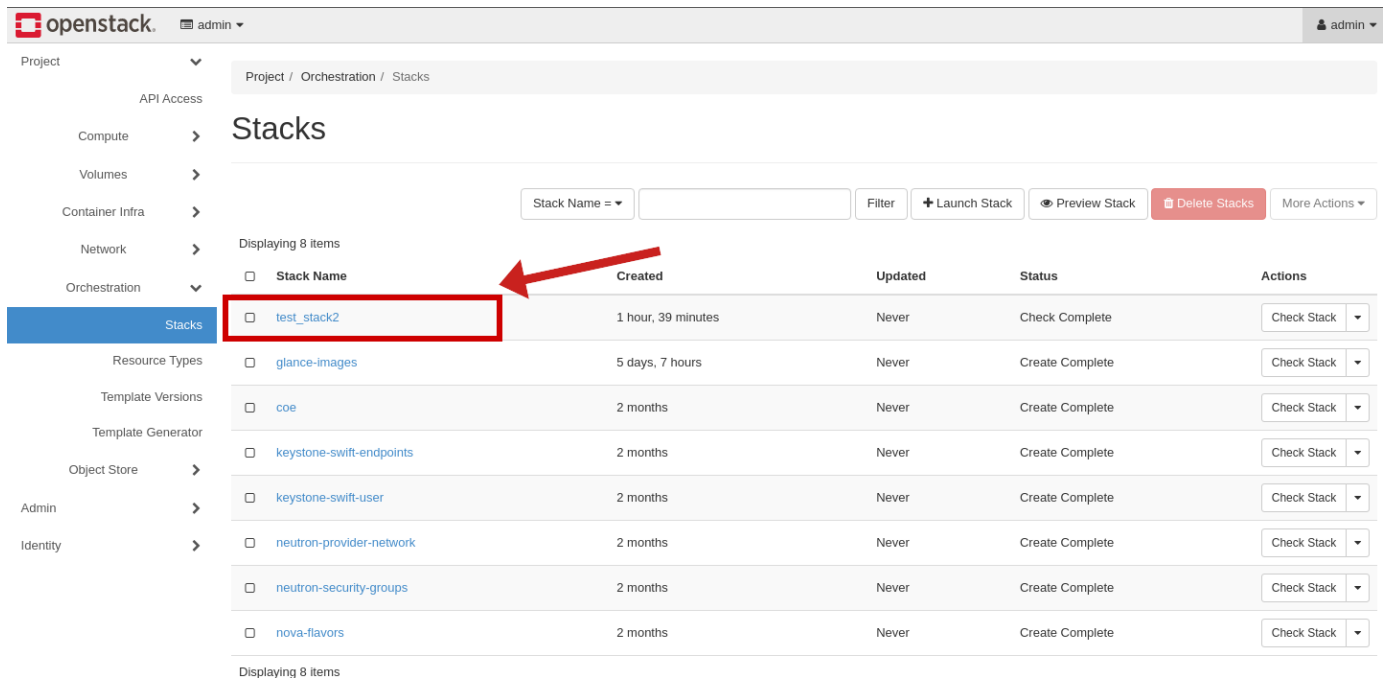


Figure 4: Stacks Created within Admin Project

Figure 4 shows the newly created test stack that was created by the example. You have the option to check the stack as well as view its current status.

If you need additional information, click the link under **Stack Name** and you can view an overview of the stack information, resources, events, and even a copy of the template used to create the stack.

The screenshot shows the OpenStack dashboard interface. The main content area displays the details for a stack named 'test_stack2'. The stack is in a 'Created' state, having been created 1 hour and 22 minutes ago. The description is 'Simple template to deploy a single instance'. The status is 'Check_Complete: Stack CHECK completed successfully'. The stack parameters are listed as follows:

OS::stack_id	75809fce-ac85-42c2-b1bc-fda626026781
OS::project_id	9015c3aa78994e6ab369e2bc99aeb7
OS::stack_name	test_stack2
key_name	ssh-hydra
image_id	CentOS 7 (e17-x86_64)
instance_type	gp1.nano
network_name	External
security_groups	Allow SSH Ingress

The launch parameters are also visible:

Timeout	None Minutes
Rollback	Disabled

Figure 5: Overview of Test Stack

Additional Resources for OpenStack Heat Service

For the latest on OpenStack Heat Documentation visit [Welcome to the Heat documentation](#). Further reading regarding Heat architecture can be found at [Heat architecture](#). Additionally, OpenStack has a guide for creating your first stack through heat that can be found at [Creating your first stack](#).

1. [Prerequisites](#)
2. [How to Create an Instance Using Terraform](#)
 1. [Prepare Terraform Directory](#)
 2. [Specify Terraform Provider](#)
 3. [Initialize Terraform](#)
 4. [Create OpenStack Application Credentials](#)
 5. [Create Main Terraform File](#)
 1. [Configure OpenStack Provider](#)
 2. [Configure Compute Resource](#)
 6. [Create Terraform Plan](#)
 7. [Deploy Terraform Plan](#)
3. [View Instance Created by Terraform](#)

Prerequisites

- An OpenStack user account. The account does not have to have the administrator role.
- Linux command line experience
- A [Terraform installation](#)

How to Create an Instance Using Terraform

Step 1: Prepare Terraform Directory

Terraform should be installed to a machine that has Internet access to your Private Cloud. This could be your own machine or one of your cloud's hardware nodes, for example.

When working with Terraform, we suggest creating a folder to manage your Terraform plans and execution files. For example:

```
mkdir ~/terraform
```

Step 2: Specify Terraform Provider

When working with Terraform, you must specify a provider. There are a number of providers to choose from located in Terraform's [Providers](#) website. For our case, we need to use Terraform's [OpenStack Provider](#) because our clouds are powered by OpenStack.

To specify the OpenStack provider, create a file called `providers.tf` in your Terraform directory containing:

```
terraform {
  required_providers {
    openstack = {
      source = "terraform-provider-openstack/openstack"
      version = "1.46.0"
    }
  }
}
```

Step 3: Initialize Terraform

With a provider defined, Terraform must be initialized.

To initialize Terraform, execute:

```
$ terraform init
```

When Terraform has been successfully initialized, the following message is returned:

```
Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

Step 4: Create OpenStack Application Credentials

To point Terraform to the appropriate cloud and authenticate, you can generate a set of OpenStack Application Credentials. To do so, log in to your cloud and navigate to the section **Identity -> Application Credentials**.

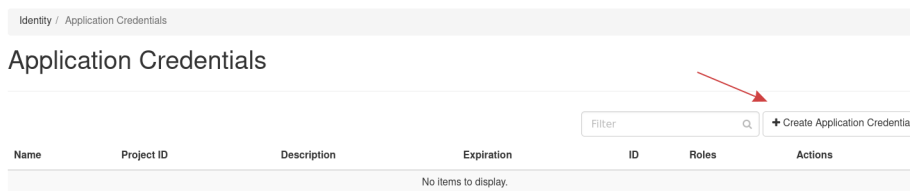


Figure 1: Application Credentials

Click **Create Application Credential** to create a new set of credentials. Fill out the details as needed and submit the form. Another pop up form appears providing you the relevant details, like the **ID**, **Name**, and **Secret**.. Additionally, you can download either an `openrc` file or a `clouds.yaml` with this information pre-defined for you. For this demonstration, we are using `clouds.yaml`. When using `clouds.yaml`, download this file to the same folder in which you have initialized Terraform.

The contents of `clouds.yaml` appears similar to:

```
clouds:
  openstack:
    auth:
```


- **key_pair**: The value for a key pair hosted in your cloud
- **security_groups**: The value in list format for security groups to set

Then within the `resource` block, we also configure a network with which the instance will be associated by creating a `network` block.

A description of the variable used for the `network` block:

- **name**: This is the name of a network available to your project

As an example, here is the above template filled out using details from a Private Cloud:

```
resource "openstack_compute_instance_v2" "terraform-demo-instance" {
  name = "demo-instance"
  image_id = "c786deab-3fc6-4a92-9a1e-54bcab32e2c2"
  flavor_id = "m1.small"
  key_pair = "demo-key"
  security_groups = ["default"]

  network {
    name = "Private"
  }
}
```

The template name can be changed to suit your purposes but must include the `.tf` extension.

Step 6: Create Terraform Plan

Terraform needs to create a plan based on the current configuration. This plan provides the changes Terraform will make to your cloud prior to making them, giving an operator a chance for review.

The command to create a Terraform plan is:

```
$ terraform plan
```

By default `terraform plan` does not save the plan to disk. To have the plan written to disk, use:

```
$ terraform plan -out <path>
```

Replacing `<path>` with where you want to store the Terraform plan.

For this example, we will have Terraform create the plan and write the plan to disk using:

```
$ terraform plan -out ~/terraform/plan
```

This creates a Terraform plan in the location `~/terraform/plan`.

Step 7: Deploy Terraform Plan

After reviewing the plan and ensuring the changes to be made meet your expectations, use Terraform to deploy the plan.

The command to deploy a Terraform plan is:

```
$ terraform apply [PLAN]
```

Where `[PLAN]` is an optional variable. For this example, since we saved the plan to disk, we will use that plan when applying Terraform.

For example:

```
$ terraform apply ~/terraform/plan
```

Results of a successful plan application:

```
$ terraform apply ~/terraform/plan
openstack_compute_instance_v2.terraform-demo-instance: Creating...
openstack_compute_instance_v2.terraform-demo-instance: Still creating... [10s elapsed]
openstack_compute_instance_v2.terraform-demo-instance: Still creating... [20s elapsed]
```

```
openstack_compute_instance_v2.terraform-demo-instance: Creation complete after 24s [id=3a126
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

This concludes using Terraform to spin up an instance in your Private Cloud.

View Instance Created by Terraform

To view your created instance, navigate in Horizon to **Project -> Compute -> Instances**, where you can view the instance created by Terraform.

Project / Compute / Instances

Instances

Instance Name: demo-instance Filter Launch Instance Delete Instances More Actions

Displaying 1 item

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
demo-instance	CentOS 8 Stream (el8-x86_64)	192.168.0.143	m1.small	demo-key	Active	nova	None	Running	3 minutes	Create Snapshot

Figure 2: Newly Created Terraform Instance