



OpenShift Container Platform 4.4

Installing on OpenStack

Installing OpenShift Container Platform 4.4 OpenStack clusters

OpenShift Container Platform 4.4 Installing on OpenStack

Installing OpenShift Container Platform 4.4 OpenStack clusters

Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions for installing and uninstalling OpenShift Container Platform 4.4 clusters on OpenStack Platform.

Table of Contents

CHAPTER 1. INSTALLING ON OPENSTACK	5
1.1. INSTALLING A CLUSTER ON OPENSTACK WITH CUSTOMIZATIONS	5
1.1.1. Resource guidelines for installing OpenShift Container Platform on OpenStack	5
1.1.1.1. Control plane and compute machines	6
1.1.1.2. Bootstrap machine	6
1.1.2. Internet and Telemetry access for OpenShift Container Platform	6
1.1.3. Enabling Swift on OpenStack	7
1.1.4. Verifying external network access	8
1.1.5. Defining parameters for the installation program	9
1.1.6. Obtaining the installation program	10
1.1.7. Creating the installation configuration file	11
1.1.8. Installation configuration parameters	12
1.1.8.1. Sample customized install-config.yaml file for OpenStack	16
1.1.9. Generating an SSH private key and adding it to the agent	17
1.1.10. Enabling access to the environment	18
1.1.10.1. Enabling access with floating IP addresses	18
1.1.10.2. Enabling access without floating IP addresses	19
1.1.11. Deploying the cluster	19
1.1.12. Verifying cluster status	20
1.1.13. Logging in to the cluster	21
1.1.14. Configuring application access with floating IP addresses	21
1.2. INSTALLING A CLUSTER ON OPENSTACK WITH KURYR	22
1.2.1. About Kuryr SDN	23
1.2.2. Resource guidelines for installing OpenShift Container Platform on OpenStack with Kuryr	23
1.2.2.1. Increasing quota	25
1.2.2.2. Configuring Neutron	25
1.2.2.3. Configuring Octavia	26
1.2.2.3.1. The Octavia OVN Driver	29
1.2.2.4. Known limitations of installing with Kuryr	30
RHOSP version limitations	30
RHOSP environment limitations	30
1.2.2.5. Control plane and compute machines	30
1.2.2.6. Bootstrap machine	31
1.2.3. Internet and Telemetry access for OpenShift Container Platform	31
1.2.4. Enabling Swift on OpenStack	32
1.2.5. Verifying external network access	32
1.2.6. Defining parameters for the installation program	33
1.2.7. Obtaining the installation program	35
1.2.8. Creating the installation configuration file	35
1.2.9. Installation configuration parameters	37
1.2.9.1. Sample customized install-config.yaml file for OpenStack with Kuryr	40
1.2.10. Generating an SSH private key and adding it to the agent	41
1.2.11. Enabling access to the environment	42
1.2.11.1. Enabling access with floating IP addresses	43
1.2.11.2. Enabling access without floating IP addresses	43
1.2.12. Deploying the cluster	43
1.2.13. Verifying cluster status	44
1.2.14. Logging in to the cluster	45
1.2.15. Configuring application access with floating IP addresses	45
1.3. INSTALLING A CLUSTER ON OPENSTACK ON YOUR OWN INFRASTRUCTURE	46
1.3.1. Internet and Telemetry access for OpenShift Container Platform	47

1.3.2. Resource guidelines for installing OpenShift Container Platform on OpenStack	47
1.3.2.1. Control plane and compute machines	48
1.3.2.2. Bootstrap machine	49
1.3.3. Downloading playbook dependencies	49
1.3.4. Obtaining the installation program	50
1.3.5. Generating an SSH private key and adding it to the agent	50
1.3.6. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image	51
1.3.7. Verifying external network access	52
1.3.8. Enabling access to the environment	53
1.3.8.1. Enabling access with floating IP addresses	53
1.3.9. Defining parameters for the installation program	54
1.3.10. Creating the installation files for RHOSP	55
1.3.11. Creating the installation configuration file	55
1.3.12. Installation configuration parameters	57
1.3.12.1. Sample customized install-config.yaml file for OpenStack	60
1.3.12.2. Setting a custom subnet for machines	60
1.3.12.3. Emptying compute machine pools	61
1.3.13. Creating the Kubernetes manifest and Ignition config files	61
1.3.14. Preparing the bootstrap Ignition files	63
1.3.15. Creating control plane Ignition config files	66
1.3.16. Creating network resources	67
1.3.17. Creating the bootstrap machine	79
1.3.18. Creating the control plane machines	80
1.3.19. Logging in to the cluster	82
1.3.20. Deleting bootstrap resources	82
1.3.21. Creating compute machines	83
1.3.22. Approving the CSRs for your machines	85
1.3.23. Verifying a successful installation	86
1.3.24. Configuring application access with floating IP addresses	87
1.4. INSTALLING A CLUSTER ON OPENSTACK WITH KURYR ON YOUR OWN INFRASTRUCTURE	88
1.4.1. About Kuryr SDN	88
1.4.2. Resource guidelines for installing OpenShift Container Platform on OpenStack with Kuryr	89
1.4.2.1. Increasing quota	91
1.4.2.2. Configuring Neutron	91
1.4.2.3. Configuring Octavia	91
1.4.2.3.1. The Octavia OVN Driver	94
1.4.2.4. Known limitations of installing with Kuryr	95
RHOSP version limitations	95
RHOSP environment limitations	95
1.4.2.5. Control plane and compute machines	96
1.4.2.6. Bootstrap machine	96
1.4.3. Internet and Telemetry access for OpenShift Container Platform	96
1.4.4. Downloading playbook dependencies	97
1.4.5. Obtaining the installation program	98
1.4.6. Generating an SSH private key and adding it to the agent	98
1.4.7. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image	99
1.4.8. Verifying external network access	100
1.4.9. Enabling access to the environment	101
1.4.9.1. Enabling access with floating IP addresses	101
1.4.10. Defining parameters for the installation program	102
1.4.11. Creating the installation files for RHOSP	103
1.4.12. Creating the installation configuration file	103
1.4.13. Installation configuration parameters	105

1.4.13.1. Sample customized install-config.yaml file for OpenStack with Kuryr	107
1.4.13.2. Setting a custom subnet for machines	108
1.4.13.3. Emptying compute machine pools	109
1.4.13.4. Modifying the network type	110
1.4.14. Creating the Kubernetes manifest and Ignition config files	110
1.4.15. Preparing the bootstrap Ignition files	112
1.4.16. Creating control plane Ignition config files	115
1.4.17. Creating network resources	115
1.4.18. Creating the bootstrap machine	128
1.4.19. Creating the control plane machines	129
1.4.20. Logging in to the cluster	131
1.4.21. Deleting bootstrap resources	131
1.4.22. Creating compute machines	132
1.4.23. Approving the CSRs for your machines	134
1.4.24. Verifying a successful installation	135
1.4.25. Configuring application access with floating IP addresses	135
1.5. UNINSTALLING A CLUSTER ON OPENSTACK	136
1.5.1. Removing a cluster that uses installer-provisioned infrastructure	136
1.6. UNINSTALLING A CLUSTER ON OPENSTACK FROM YOUR OWN INFRASTRUCTURE	137
1.6.1. Downloading playbook dependencies	137
1.6.2. Removing a cluster on OpenStack that uses your own infrastructure	138

CHAPTER 1. INSTALLING ON OPENSTACK

1.1. INSTALLING A CLUSTER ON OPENSTACK WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.4, you can install a customized cluster on Red Hat OpenStack Platform (RHOSP). To customize the installation, modify parameters in the **install-config.yaml** before you install the cluster.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
 - Verify that OpenShift Container Platform 4.4 is compatible with your RHOSP version in the *Available platforms* section. You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#).
- Have a storage service installed in RHOSP, like Block Storage (Cinder) or Object Storage (Swift). Object storage is the recommended storage technology for OpenShift Container Platform registry cluster deployment. For more information, see [Optimizing storage](#).
- Have metadata service enabled in RHOSP

1.1.1. Resource guidelines for installing OpenShift Container Platform on OpenStack

To support a OpenShift Container Platform installation, your Red Hat OpenStack Platform (RHOSP) quota must meet the following requirements:

Table 1.1. Recommended resources for a default OpenShift Container Platform cluster on RHOSP

Resource	Value
Floating IP addresses	3
Ports	15
Routers	1
Subnets	1
RAM	112 GB
vCPUs	28
Volume storage	275 GB
Instances	7
Security groups	3
Security group rules	60

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If OpenStack Object Storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



NOTE

By default, your security group and security group rule quotas might be low. If you encounter problems, run **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** as an administrator to increase them.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

1.1.1.1. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

1.1.1.2. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

1.1.2. Internet and Telemetry access for OpenShift Container Platform

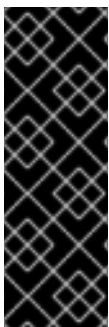
In OpenShift Container Platform 4.4, you require access to the internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of

updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require internet access. Before you update the cluster, you update the content of the mirror registry.

1.1.3. Enabling Swift on OpenStack

Swift is operated by a user account with the **swiftoperator** role. Add the role to an account before you run the installation program.



IMPORTANT

If [OpenStack Object Storage \(Swift\)](#) is available, OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) uses it as the image registry storage. If it is unavailable, the installation program relies on the OpenStack Block Storage service (Cinder).

If Swift is present and you want to use it, you must enable access to it. If it is not present, or if you do not want to use it, skip this section.

Prerequisites

- You have a RHOSP administrator account on the target environment.
- The Swift service is installed.
- On [Ceph RGW](#), the **account in url** option is enabled.

Procedure

To enable Swift on RHOSP:

1. As an administrator in the RHOSP CLI, add the **swiftoperator** role to the account that will access Swift:

```
$ openstack role add --user <user> --project <project> swiftoperator
```

Your RHOSP deployment can now use Swift for the image registry.

1.1.4. Verifying external network access

The OpenShift Container Platform installation process requires external network access. You must provide an external network value to it, or deployment fails. Before you begin the process, verify that a network with the External router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- [Configure OpenStack’s networking service to have DHCP agents forward instances' DNS queries](#)

Procedure

1. Using the RHOSP CLI, verify the name and ID of the 'External' network:

```
$ openstack network list --long -c ID -c Name -c "Router Type"

+-----+-----+-----+
| ID              | Name          | Router Type |
+-----+-----+-----+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+-----+-----+-----+
```

A network with an External router type appears in the network list. If at least one does not, see [Create an external network](#).



IMPORTANT

If the external network’s CIDR range overlaps one of the default network ranges, you must change the matching network ranges in the **install-config.yaml** file before you start the installation process.

The default network ranges are:

Network	Range
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

CAUTION

If the installation program finds multiple networks with the same name, it sets one of them at random. To avoid this behavior, create unique names for resources in RHOSP.

**NOTE**

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

1.1.5. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

1. Create the **clouds.yaml** file:

- If your OpenStack distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.

**IMPORTANT**

Remember to add a password to the **auth** field. You can also keep secrets in [a separate file](#) from **clouds.yaml**.

- If your OpenStack distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:
 - a. Copy the certificate authority file to your machine.
 - b. In the command line, run the following commands to add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

- c. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openshift/clouds.yaml**
 - d. A Unix-specific site configuration directory, for example **/etc/openshift/clouds.yaml**
 The installation program searches for **clouds.yaml** in that order.

1.1.6. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- A computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.1.7. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on OpenStack.

Prerequisites

- Download the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **openstack** as the platform to target.
- iii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.

- iv. Specify the floating IP address to use for external access to the OpenShift API.
 - v. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - vi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - vii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

1.1.8. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

You cannot modify these parameters in the **install-config.yaml** file after installation.


Table 1.2. Required parameters

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. This value is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .

Parameter	Description	Values
controlPlane.platform	The cloud provider to host the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev . The string must be 14 characters or fewer long.
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure. Red Hat OpenStack Platform (RHOSP) does not use this parameter.
pullSecret	The pull secret that you obtained from the Pull Secret page on the Red Hat OpenShift Cluster Manager site. You use this pull secret to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

Table 1.3. Optional parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
sshKey	<p>The SSH key to use to access your cluster machines.</p> <div>  <div> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	A valid, local public SSH key that you added to the ssh-agent process.
fips	Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.	false or true
publish	How to publish the user-facing endpoints of your cluster.	Internal or External . Set publish to Internal to deploy a private cluster, which cannot be accessed from the internet. The default value is External .



Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

Table 1.4. Additional Red Hat OpenStack Platform (RHOSP) parameters

Parameter	Description	Values
compute.platform.openstack.rootVolume.size	For compute machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
compute.platform.openstack.rootVolume.type	For compute machines, the root volume's type.	String, for example performance .
controlPlane.platform.openstack.rootVolume.size	For control plane machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
controlPlane.platform.openstack.rootVolume.type	For control plane machines, the root volume's type.	String, for example performance .
platform.openstack.cloud	The name of the RHOSP cloud to use from the list of clouds in the clouds.yaml file.	String, for example MyCloud .
platform.openstack.externalDNS	<i>Optional.</i> IP addresses for external DNS servers that cluster instances use for DNS resolution.	A list of IP addresses as strings, for example ["8.8.8.8", "192.168.1.12"] .
platform.openstack.externalNetwork	The RHOSP external network name to be used for installation.	String, for example external .
platform.openstack.computeFlavor	The RHOSP flavor to use for control plane and compute machines.	String, for example m1.xlarge .
platform.openstack.lbFloatingIP	An existing floating IP address to associate with the load balancer API.	An IP address, for example 128.0.0.1 .
platform.openstack.defaultMachinePlatform	<i>Optional.</i> The default machine pool platform configuration.	<pre> { "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>

1.1.8.1. Sample customized **install-config.yaml** file for OpenStack

This sample **install-config.yaml** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

This sample file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program.

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
  fips: false
  pullSecret: '{"auths": ...}'
  sshKey: ssh-ed25519 AAAA...
```

1.1.9. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.



IMPORTANT

If you create a new SSH key pair, avoid overwriting existing SSH keys.

1. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
Agent pid 31874
```

2. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

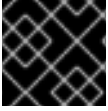
1.1.10. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible with or without floating IP addresses.

1.1.10.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.

**IMPORTANT**

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```

**NOTE**

If you do not control the DNS server you can add the record to your **/etc/hosts** file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

1.1.10.2. Enabling access without floating IP addresses

If you cannot use floating IP addresses, the OpenShift Container Platform installation might still finish. However, the installation program fails after it times out waiting for API access.

After the installation program times out, the cluster might still initialize. After the bootstrapping processing begins, it must complete. You must edit the cluster's networking configuration after it is deployed.

1.1.11. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

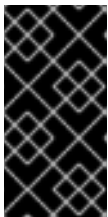
- 1** For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2** To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

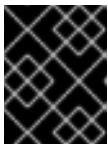
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours. You must keep the cluster running for 24 hours in a non-degraded state to ensure that the first certificate rotation has finished.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

1.1.12. Verifying cluster status

You can verify your OpenShift Container Platform cluster's status during or after installation.

Procedure

1. In the cluster environment, export the administrator's kubeconfig file:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server.

2. View the control plane and compute machines created after a deployment:

```
$ oc get nodes
```

3. View your cluster's version:

```
$ oc get clusterversion
```

4. View your operators' status:

```
$ oc get clusteroperator
```

5. View all running Pods in the cluster:

```
$ oc get pods -A
```

1.1.13. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
system:admin
```

1.1.14. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed

- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

- Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

- Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

- Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```

NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oc-auth-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

1.2. INSTALLING A CLUSTER ON OPENSTACK WITH KURYR

In OpenShift Container Platform version 4.4, you can install a customized cluster on Red Hat OpenStack Platform (RHOSP) that uses Kuryr SDN. To customize the installation, modify parameters in the **install-config.yaml** before you install the cluster.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
 - Verify that OpenShift Container Platform 4.4 is compatible with your RHOSP version in the *Available platforms* section. You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#).

- Have a storage service installed in RHOSP, like Block Storage (Cinder) or Object Storage (Swift). Object storage is the recommended storage technology for OpenShift Container Platform registry cluster deployment. For more information, see [Optimizing storage](#).

1.2.1. About Kuryr SDN

Kuryr is a container network interface (CNI) plug-in solution that uses [OpenStack Neutron](#) and [OpenStack Octavia](#) to provide networking for Pods and Services.

Kuryr and OpenShift Container Platform integration is primarily designed for OpenShift Container Platform clusters running on OpenStack VMs. Kuryr improves the network performance by plugging OpenShift Pods into OpenStack SDN. In addition, it provides interconnectivity between OpenShift Pods and OpenStack virtual instances.

Kuryr components are installed as Pods in OpenShift Container Platform using the **openshift-kuryr** namespace:

- **kuryr-controller** – a single Service instance installed on a **master** node. This is modeled in OpenShift Container Platform as a **Deployment**.
- **kuryr-cni** – a container installing and configuring Kuryr as a CNI driver on each OpenShift Container Platform node. This is modeled in OpenShift Container Platform as a **DaemonSet**.

The Kuryr controller watches the OpenShift API server for Pod, Service, and namespace create, update, and delete events. It maps the OpenShift Container Platform API calls to corresponding objects in Neutron and Octavia. This means that every network solution that implements the Neutron trunk port functionality can be used to back OpenShift Container Platform via Kuryr. This includes open source solutions such as Open vSwitch (OVS) and Open Virtual Network (OVN) as well as Neutron-compatible commercial SDNs.

Kuryr is recommended for OpenShift deployments on encapsulated OpenStack tenant networks to avoid double encapsulation, such as running an encapsulated OpenShift SDN over an OpenStack network.

If you use provider networks or tenant VLANs, you do not need to use Kuryr to avoid double encapsulation. The performance benefit is negligible. Depending on your configuration, though, using Kuryr to avoid having two overlays might still be beneficial.

Kuryr is not recommended in deployments where all of the following criteria are true:

- The RHOSP version is less than 16.
- The deployment uses UDP services, or a large number of TCP services on few hypervisors.

or

- The **ovn-octavia** Octavia driver is disabled.
- The deployment uses a large number of TCP services on few hypervisors.

1.2.2. Resource guidelines for installing OpenShift Container Platform on OpenStack with Kuryr

When using Kuryr SDN, the Pods, Services, namespaces, and network policies are using resources from the RHOSP quota; this increases the minimum requirements. Kuryr also has some additional requirements on top of what a default install requires.

Use the following quota to satisfy a default cluster's minimum requirements:

Table 1.5. Recommended resources for a default OpenShift Container Platform cluster on RHOSP with Kuryr

Resource	Value
Floating IP addresses	3 - plus the expected number of Services of LoadBalancer type
Ports	1500 - 1 needed per Pod
Routers	1
Subnets	250 - 1 needed per Namespace/Project
Networks	250 - 1 needed per Namespace/Project
RAM	112 GB
vCPUs	28
Volume storage	275 GB
Instances	7
Security groups	250 - 1 needed per Service and per NetworkPolicy
Security group rules	1000
Load balancers	100 - 1 needed per Service
Load balancer listeners	500 - 1 needed per Service-exposed port
Load balancer pools	500 - 1 needed per Service-exposed port

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If OpenStack Object Storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



IMPORTANT

If you are using Red Hat OpenStack Platform (RHOSP) version 16 with the Amphora driver rather than the OVN Octavia driver, security groups are associated with Service accounts instead of user projects.

Take the following notes into consideration when setting resources:

- The number of ports that are required is larger than the number of Pods. Kuryr uses ports pools to have pre-created ports ready to be used by Pods and speed up the Pods' booting time.
- Each NetworkPolicy is mapped into an RHOSP security group, and depending on the NetworkPolicy spec, one or more rules are added to the security group.
- Each Service is mapped to an RHOSP load balancer. Consider this requirement when estimating the number of security groups required for the quota.
If you are using RHOSP version 15 or earlier, or the **ovn-octavia driver**, each load balancer has a security group with the user project.
- Swift space requirements vary depending on the size of the bootstrap Ignition file and image registry.
- The quota does not account for load balancer resources (such as VM resources), but you must consider these resources when you decide the RHOSP deployment's size. The default installation will have more than 50 load balancers; the clusters must be able to accommodate them.
If you are using RHOSP version 16 with the OVN Octavia driver enabled, only one load balancer VM is generated; Services are load balanced through OVN flows.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

To enable Kuryr SDN, your environment must meet the following requirements:

- Run OpenStack 13+.
- Have Overcloud with Octavia.
- Use Neutron Trunk ports extension.
- Use **openvswitch** firewall driver if ML2/OVS Neutron driver is used instead of **ovs-hybrid**.

1.2.2.1. Increasing quota

When using Kuryr SDN, you must increase quotas to satisfy the OpenStack resources used by Pods, Services, namespaces, and network policies.

Procedure

- Increase the quotas for a project by running the following command:

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --subnets 250 --networks 250 <project>
```

1.2.2.2. Configuring Neutron

Kuryr CNI leverages the Neutron Trunks extension to plug containers into the OpenStack SDN, so you must use the **trunks** extension for Kuryr to properly work.

In addition, if you leverage the default ML2/OVS Neutron driver, the firewall must be set to **openvswitch** instead of **ovs_hybrid** so that security groups are enforced on trunk subports and Kuryr can properly handle network policies.

1.2.2.3. Configuring Octavia

Kuryr SDN uses OpenStack Octavia LBaaS to implement OpenShift Services. Thus, you must install and configure Octavia components in your OpenStack environment to use Kuryr SDN.

To enable Octavia, you must include the Octavia Service during the installation of the OpenStack Overcloud, or upgrade the Octavia Service if the Overcloud already exists. The following steps for enabling Octavia apply to both a clean install of the Overcloud or an Overcloud update.



NOTE

The following steps only capture the key pieces required during the [deployment of OpenStack](#) when dealing with Octavia. It is also important to note that [registry methods](#) vary.

This example uses the local registry method.

Procedure

1. If you are using the local registry, create a template to upload the images to the registry. For example:

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack- \
--tag-from-label {version}-{release} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. Verify that the **local_registry_images.yaml** file contains the Octavia images. For example:

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-manager:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```

**NOTE**

The Octavia container versions vary depending upon the specific RHOSP release installed.

3. Pull the container images from registry.redhat.io to the Undercloud node:

```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file /home/stack/local_registry_images.yaml \
  --verbose
```

This may take some time depending on the speed of your network and Undercloud disk.

4. Since an Octavia load balancer is used to access the OpenShift API, you must increase their listeners' default timeouts for the connections. The default timeout is 50 seconds. Increase the timeout to 20 minutes by passing the following file to the Overcloud deploy command:

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```

**NOTE**

This is not needed for Red Hat OpenStack Platform 14+.

5. Install or update your Overcloud environment with Octavia:

```
openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
  -e octavia_timeouts.yaml
```

**NOTE**

This command only includes the files associated with Octavia; it varies based on your specific installation of OpenStack. See the official OpenStack documentation for further information. For more information on customizing your Octavia installation, see [installation of Octavia using Director](#).

**NOTE**

When leveraging Kuryr SDN, the Overcloud installation requires the Neutron **trunk** extension. This is available by default on Director deployments. Use the **openvswitch** firewall instead of the default **ovs-hybrid** when the Neutron backend is ML2/OVS. There is no need for modifications if the backend is ML2/OVN.

6. In RHOSP versions 13 and 15, add the project ID to the **octavia.conf** configuration file after you create the project.
 - To enforce network policies across Services, like when traffic goes through the Octavia load balancer, you must ensure Octavia creates the Amphora VM security groups on the user project.

This change ensures that required LoadBalancer security groups belong to that project, and that they can be updated to enforce Services isolation.



NOTE

This task is unnecessary in RHOSP version 16 or later.

Octavia implements a new ACL API that restricts access to the Load Balancers VIP.

- a. Get the project ID

```
$ openstack project show <project>
+-----+-----+
| Field | Value |
+-----+-----+
| description |
| domain_id | default |
| enabled | True |
| id | PROJECT_ID |
| is_domain | False |
| name | *<project>* |
| parent_id | default |
| tags | [] |
+-----+-----+
```

- b. Add the project ID to **octavia.conf** for the controllers.

- i. List the Overcloud controllers.

```
$ source stackrc # Undercloud credentials
$ openstack server list
+-----+-----+-----+-----+-----+
| ID | Name | Status | Networks |
+-----+-----+-----+-----+
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE | ctlplane=192.168.24.8 | overcloud-full | controller |
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0 | ACTIVE | ctlplane=192.168.24.6 | overcloud-full | compute |
+-----+-----+-----+-----+-----+
```

- ii. SSH into the controller(s).

```
$ ssh heat-admin@192.168.24.8
```


- iii. Edit the **octavia.conf** to add the project into the list of projects where Amphora security groups are on the user's account.

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

- c. Restart the Octavia worker so the new configuration loads.

```
controller-0$ sudo docker restart octavia_worker
```



NOTE

Depending on your RHOSP environment, Octavia might not support UDP listeners. If you use Kuryr SDN on OpenStack version 15 or earlier, UDP services are not supported. RHOSP version 16 or later support UDP.

1.2.2.3.1. The Octavia OVN Driver

Octavia supports multiple provider drivers through the Octavia API.

To see all available Octavia provider drivers, on a command line, enter:

```
$ openstack loadbalancer provider list
```

The result is a list of drivers:

```
+-----+-----+
| name | description |
+-----+-----+
| amphora | The Octavia Amphora driver. |
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn | Octavia OVN driver. |
+-----+-----+
```

Beginning with RHOSP version 16, the Octavia OVN provider driver (**ovn**) is supported on OpenShift Container Platform on RHOSP deployments.

ovn is an integration driver for the load balancing that Octavia and OVN provide. It supports basic load balancing capabilities, and is based on OpenFlow rules. The driver is automatically enabled in Octavia by Director on deployments that use OVN Neutron ML2.

The Amphora provider driver is the default driver. If **ovn** is enabled, however, Kuryr uses it.

If Kuryr uses **ovn** instead of Amphora, it offers the following benefits:

- Decreased resource requirements. Kuryr does not require a load balancer VM for each Service.
- Reduced network latency.
- Increased service creation speed by using OpenFlow rules instead of a VM for each Service.
- Distributed load balancing actions across all nodes instead of centralized on Amphora VMs.

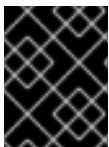
1.2.2.4. Known limitations of installing with Kuryr

Using OpenShift Container Platform with Kuryr SDN has several known limitations.

RHOSP version limitations

Using OpenShift Container Platform with Kuryr SDN has several limitations that depend on the RHOSP version.

- RHOSP versions before 16 use the default Octavia load balancer driver (Amphora). This driver requires that one Amphora load balancer VM is deployed per OpenShift Service. Creating too many Services can cause you to run out of resources.
Deployments of later versions of RHOSP that have the OVN Octavia driver disabled also use the Amphora driver. They are subject to the same resource concerns as earlier versions of RHOSP.
- Octavia RHOSP versions before 16 do not support UDP listeners. Therefore, OpenShift UDP services are not supported.
- Octavia RHOSP versions before 16 cannot listen to multiple protocols on the same port. Services that expose the same port to different protocols, like TCP and UDP, are not supported.



IMPORTANT

The OVN Octavia driver does not support listeners that use different protocols on any RHOSP version.

RHOSP environment limitations

There are limitations when using Kuryr SDN that depend on your deployment environment.

Because of Octavia's lack of support for the UDP protocol and multiple listeners, Kuryr forces Pods to use TCP for DNS resolution if:

- The RHOSP version is earlier than 16
- The OVN Octavia driver is used

In Go versions 1.12 and earlier, applications that are compiled with CGO support disabled use UDP only. In this case, the native Go resolver does not recognize the **use-vc** option in **resolv.conf**, which controls whether TCP is forced for DNS resolution. As a result, UDP is still used for DNS resolution, which fails.

To ensure that TCP forcing is allowed, compile applications either with the environment variable **CGO_ENABLED** set to **1**, i.e. **CGO_ENABLED=1**, or ensure that the variable is absent.

In Go versions 1.13 and later, TCP is used automatically if DNS resolution using UDP fails.



NOTE

musl-based containers, including Alpine-based containers, do not support the **use-vc** option.

1.2.2.5. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

1.2.2.6. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

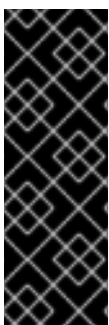
1.2.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.4, you require access to the internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require internet access. Before you update the cluster, you update the content of the mirror registry.

1.2.4. Enabling Swift on OpenStack

Swift is operated by a user account with the **swiftoperator** role. Add the role to an account before you run the installation program.



IMPORTANT

If [OpenStack Object Storage \(Swift\)](#) is available, OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) uses it as the image registry storage. If it is unavailable, the installation program relies on the OpenStack Block Storage service (Cinder).

If Swift is present and you want to use it, you must enable access to it. If it is not present, or if you do not want to use it, skip this section.

Prerequisites

- You have a RHOSP administrator account on the target environment.
- The Swift service is installed.
- On [Ceph RGW](#), the **account in url** option is enabled.

Procedure

To enable Swift on RHOSP:

1. As an administrator in the RHOSP CLI, add the **swiftoperator** role to the account that will access Swift:

```
$ openstack role add --user <user> --project <project> swiftoperator
```

Your RHOSP deployment can now use Swift for the image registry.

1.2.5. Verifying external network access

The OpenShift Container Platform installation process requires external network access. You must provide an external network value to it, or deployment fails. Before you begin the process, verify that a network with the External router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- [Configure OpenStack's networking service to have DHCP agents forward instances' DNS queries](#)

Procedure

1. Using the RHOSP CLI, verify the name and ID of the 'External' network:

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

```
+-----+-----+-----+
| ID           | Name           | Router Type |
+-----+-----+-----+
```

```
+-----+-----+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External |
+-----+-----+
```

A network with an External router type appears in the network list. If at least one does not, see [Create an external network](#).

IMPORTANT

If the external network's CIDR range overlaps one of the default network ranges, you must change the matching network ranges in the **install-config.yaml** file before you start the installation process.

The default network ranges are:

Network	Range
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

CAUTION

If the installation program finds multiple networks with the same name, it sets one of them at random. To avoid this behavior, create unique names for resources in RHOSP.

NOTE

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

1.2.6. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

1. Create the **clouds.yaml** file:
 - If your OpenStack distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.

IMPORTANT

Remember to add a password to the **auth** field. You can also keep secrets in [a separate file](#) from **clouds.yaml**.

- If your OpenStack distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
    dev-env:
      region_name: RegionOne
      auth:
        username: 'devuser'
        password: XXX
        project_name: 'devonly'
        auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:
 - a. Copy the certificate authority file to your machine.
 - b. In the command line, run the following commands to add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

- c. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory

- c. A Unix-specific user configuration directory, for example `~/.config/openstack/clouds.yaml`
 - d. A Unix-specific site configuration directory, for example `/etc/openstack/clouds.yaml`
- The installation program searches for **clouds.yaml** in that order.

1.2.7. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- A computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.2.8. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on OpenStack.

Prerequisites

- Download the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
- i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **openstack** as the platform to target.
 - iii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
 - iv. Specify the floating IP address to use for external access to the OpenShift API.
 - v. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - vi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - vii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

1.2.9. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE


You cannot modify these parameters in the **install-config.yaml** file after installation.

Table 1.6. Required parameters

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. This value is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
controlPlane.platform	The cloud provider to host the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev . The string must be 14 characters or fewer long.
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure. Red Hat OpenStack Platform (RHOSP) does not use this parameter.

Parameter	Description	Values
pullSecret	The pull secret that you obtained from the Pull Secret page on the Red Hat OpenShift Cluster Manager site. You use this pull secret to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

Table 1.7. Optional parameters

Parameter	Description	Values
sshKey	<p>The SSH key to use to access your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div>	A valid, local public SSH key that you added to the ssh-agent process.
fips	Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.	false or true
publish	How to publish the user-facing endpoints of your cluster.	Internal or External . Set publish to Internal to deploy a private cluster, which cannot be accessed from the internet. The default value is External .



Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

Table 1.8. Additional Red Hat OpenStack Platform (RHOSP) parameters

Parameter	Description	Values
compute.platform.openstack.rootVolume.size	For compute machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
compute.platform.openstack.rootVolume.type	For compute machines, the root volume's type.	String, for example performance .
controlPlane.platform.openstack.rootVolume.size	For control plane machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
controlPlane.platform.openstack.rootVolume.type	For control plane machines, the root volume's type.	String, for example performance .
platform.openstack.cloud	The name of the RHOSP cloud to use from the list of clouds in the clouds.yaml file.	String, for example MyCloud .
platform.openstack.externalDNS	<i>Optional.</i> IP addresses for external DNS servers that cluster instances use for DNS resolution.	A list of IP addresses as strings, for example ["8.8.8.8", "192.168.1.12"] .
platform.openstack.externalNetwork	The RHOSP external network name to be used for installation.	String, for example external .
platform.openstack.computeFlavor	The RHOSP flavor to use for control plane and compute machines.	String, for example m1.xlarge .
platform.openstack.lbFloatingIP	An existing floating IP address to associate with the load balancer API.	An IP address, for example 128.0.0.1 .
platform.openstack.defaultMachinePlatform	<i>Optional.</i> The default machine pool platform configuration.	<pre> { "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>

1.2.9.1. Sample customized install-config.yaml file for OpenStack with Kuryr

To deploy with Kuryr SDN instead of the default OpenShift SDN, you must modify the **install-config.yaml** file to include **Kuryr** as the desired **networking.networkType** and proceed with the default

OpenShift SDN installation steps. This sample **install-config.yaml** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

This sample file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program.

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
    trunkSupport: true
    octaviaSupport: true
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

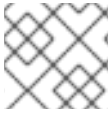


NOTE

Both **trunkSupport** and **octaviaSupport** are automatically discovered by the installer, so there is no need to set them. But if your environment does not meet both requirements, Kuryr SDN will not properly work. Trunks are needed to connect the Pods to the OpenStack network and Octavia is required to create the OpenShift Services.

1.2.10. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.

**NOTE**

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

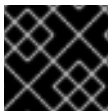
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.

**IMPORTANT**

If you create a new SSH key pair, avoid overwriting existing SSH keys.

1. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
Agent pid 31874
```

2. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

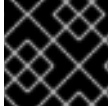
1.2.11. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible with or without floating IP addresses.

1.2.11.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.



IMPORTANT

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```



NOTE

If you do not control the DNS server you can add the record to your **/etc/hosts** file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

1.2.11.2. Enabling access without floating IP addresses

If you cannot use floating IP addresses, the OpenShift Container Platform installation might still finish. However, the installation program fails after it times out waiting for API access.

After the installation program times out, the cluster might still initialize. After the bootstrapping processing begins, it must complete. You must edit the cluster's networking configuration after it is deployed.

1.2.12. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

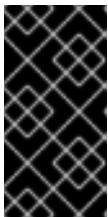
```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1** For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2** To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

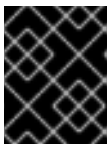
**NOTE**

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours. You must keep the cluster running for 24 hours in a non-degraded state to ensure that the first certificate rotation has finished.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

1.2.13. Verifying cluster status

You can verify your OpenShift Container Platform cluster's status during or after installation.

Procedure

1. In the cluster environment, export the administrator's kubeconfig file:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```


- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server.

2. View the control plane and compute machines created after a deployment:

```
$ oc get nodes
```

3. View your cluster's version:

```
$ oc get clusterversion
```

4. View your operators' status:

```
$ oc get clusteroperator
```

5. View all running Pods in the cluster:

```
$ oc get pods -A
```

1.2.14. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
system:admin
```

1.2.15. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```



NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oidc-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oidc-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

1.3. INSTALLING A CLUSTER ON OPENSTACK ON YOUR OWN INFRASTRUCTURE

In OpenShift Container Platform version 4.4, you can install a cluster on Red Hat OpenStack Platform (RHOSP) that runs on user-provisioned infrastructure.

Using your own infrastructure allows you to integrate your cluster with existing infrastructure and

modifications. The process requires more labor on your part than installer-provisioned installations, because you must create all RHOSP resources, like Nova servers, Neutron ports, and security groups. However, Red Hat provides Ansible playbooks to help you in the deployment process.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
 - Verify that OpenShift Container Platform 4.4 is compatible with your RHOSP version in the *Available platforms* section. You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#).
- Have an RHOSP account where you want to install OpenShift Container Platform
- On the machine from which you run the installation program, have:
 - A single directory in which you can keep the files you create during the installation process
 - Python 3

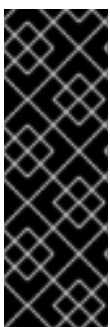
1.3.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.4, you require access to the internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require internet access. Before you update the cluster, you update the content of the mirror registry.

1.3.2. Resource guidelines for installing OpenShift Container Platform on OpenStack

To support a OpenShift Container Platform installation, your Red Hat OpenStack Platform (RHOSP) quota must meet the following requirements:

Table 1.9. Recommended resources for a default OpenShift Container Platform cluster on RHOSP

Resource	Value
Floating IP addresses	3
Ports	15
Routers	1
Subnets	1
RAM	112 GB
vCPUs	28
Volume storage	275 GB
Instances	7
Security groups	3
Security group rules	60

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If OpenStack Object Storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



NOTE

By default, your security group and security group rule quotas might be low. If you encounter problems, run **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** as an administrator to increase them.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

1.3.2.1. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

1.3.2.2. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

1.3.3. Downloading playbook dependencies

The Ansible playbooks that simplify the installation process on user-provisioned infrastructure require several Python modules. On the machine where you will run the installer, add the modules' repositories and then download them.



NOTE

These instructions assume that you are using Red Hat Enterprise Linux 8.

Prerequisites

- Python 3 is installed on your machine

Procedure

1. On a command line, add the repositories:

```
$ sudo subscription-manager register # If not done already
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
$ sudo subscription-manager repos --disable=* # If not done already

$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.8-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. Install the modules:

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. Ensure that the **python** command points to **python3**:

```
$ sudo alternatives --set python /usr/bin/python3
```

1.3.4. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- A computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.3.5. Generating an SSH private key and adding it to the agent

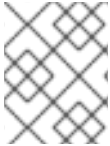
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's **~/.ssh/authorized_keys** list.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.

**IMPORTANT**

If you create a new SSH key pair, avoid overwriting existing SSH keys.

1. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
Agent pid 31874
```

2. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

1.3.6. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image

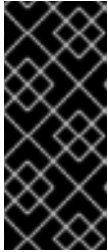
The OpenShift Container Platform installation program requires that a Red Hat Enterprise Linux CoreOS (RHCOS) image be present in the Red Hat OpenStack Platform (RHOSP) cluster. Retrieve the latest RHCOS image, then upload it using the RHOSP CLI.

Prerequisites

- The RHOSP CLI is installed.

Procedure

1. Log in to the Red Hat customer portal's [Product Downloads](#) page.
2. Under **Version**, select the most recent release of OpenShift Container Platform 4.4 for RHEL 8.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

3. Download the *Red Hat Enterprise Linux CoreOS - OpenStack Image (QCOW)*.
4. Decompress the image.



NOTE

You must decompress the OpenStack image before the cluster can use it. The name of the downloaded file might not contain a compression extension, like **.gz** or **.tgz**. To find out if or how the file is compressed, in a command line, enter:

```
$ file <name_of_downloaded_file>
```

5. From the image that you downloaded, create an image that is named **rhcos** in your cluster by using the RHOSP CLI:

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-
${RHCOS_VERSION}-openstack.qcow2 rhcos
```



IMPORTANT

Depending on your RHOSP environment, you might be able to upload the image in either **.raw** or **.qcow2** formats. If you use Ceph, you must use the **.raw** format.

CAUTION

If the installation program finds multiple images with the same name, it chooses one of them at random. To avoid this behavior, create unique names for resources in RHOSP.

After you upload the image to RHOSP, it is usable in the installation process.

1.3.7. Verifying external network access

The OpenShift Container Platform installation process requires external network access. You must provide an external network value to it, or deployment fails. Before you begin the process, verify that a network with the External router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- [Configure OpenStack's networking service to have DHCP agents forward instances' DNS queries](#)

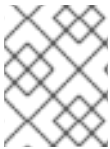
Procedure

1. Using the RHOSP CLI, verify the name and ID of the 'External' network:

```
$ openstack network list --long -c ID -c Name -c "Router Type"

+-----+-----+-----+
| ID                | Name          | Router Type |
+-----+-----+-----+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+-----+-----+-----+
```

A network with an External router type appears in the network list. If at least one does not, see [Create an external network](#).



NOTE

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

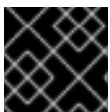
1.3.8. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible by using floating IP addresses.

1.3.8.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.



IMPORTANT

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

- To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```



NOTE

If you do not control the DNS server you can add the record to your `/etc/hosts` file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

1.3.9. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

- Create the **clouds.yaml** file:

- If your OpenStack distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.



IMPORTANT

Remember to add a password to the **auth** field. You can also keep secrets in [a separate file](#) from **clouds.yaml**.

- If your OpenStack distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
```

```
password: XXX
project_name: 'devonly'
auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:
 - a. Copy the certificate authority file to your machine.
 - b. In the command line, run the following commands to add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

- c. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```
clouds:
  shiftstack:
  ...
  cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openstack/clouds.yaml**
 - d. A Unix-specific site configuration directory, for example **/etc/openstack/clouds.yaml**

The installation program searches for **clouds.yaml** in that order.

1.3.10. Creating the installation files for RHOSP

1.3.11. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on OpenStack.

Prerequisites

- Download the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

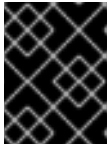
- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **openstack** as the platform to target.
 - iii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
 - iv. Specify the floating IP address to use for external access to the OpenShift API.
 - v. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - vi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - vii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

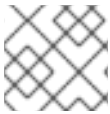
**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

You now have the file **install-config.yaml** in the directory that you specified.

1.3.12. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**


You cannot modify these parameters in the **install-config.yaml** file after installation.



Table 1.10. Required parameters

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. This value is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
controlPlane.platform	The cloud provider to host the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .

Parameter	Description	Values
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure. Red Hat OpenStack Platform (RHOSP) does not use this parameter.
pullSecret	The pull secret that you obtained from the Pull Secret page on the Red Hat OpenShift Cluster Manager site. You use this pull secret to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

Table 1.11. Optional parameters

Parameter	Description	Values
sshKey	<p>The SSH key to use to access your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div>	A valid, local public SSH key that you added to the ssh-agent process.
fips	Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.	false or true

Parameter	Description	Values
publish	How to publish the user-facing endpoints of your cluster.	Internal or External . Set publish to Internal to deploy a private cluster, which cannot be accessed from the internet. The default value is External .
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

1.3.12.1. Sample customized `install-config.yaml` file for OpenStack

This sample **`install-config.yaml`** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

This sample file is provided for reference only. You must obtain your **`install-config.yaml`** file by using the installation program.

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
  fips: false
  pullSecret: '{"auths": ...}'
  sshKey: ssh-ed25519 AAAA...
```

1.3.12.2. Setting a custom subnet for machines

The IP range that the installation program uses by default might not match the Neutron subnet that you create when you install OpenShift Container Platform. If necessary, update the CIDR value for new machines by editing the installation configuration file.

Prerequisites

- You have the **`install-config.yaml`** file that was generated by the OpenShift Container Platform installation program.

Procedure

1. On a command line, browse to the directory that contains **install-config.yaml**.
2. From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```
python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

1 Insert a value that matches your intended Neutron subnet, e.g. **192.0.2.0/24**.

- To set the value manually, open the file and set the value of **networking.machineCIDR** to something that matches your intended Neutron subnet.

1.3.12.3. Emptying compute machine pools

To proceed with an installation that uses your own infrastructure, set the number of compute machines in the installation configuration file to zero. Later, you create these machines manually.

Prerequisites

- You have the **install-config.yaml** file that was generated by the OpenShift Container Platform installation program.

Procedure

1. On a command line, browse to the directory that contains **install-config.yaml**.
2. From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- To set the value manually, open the file and set the value of **compute.<first entry>.replicas** to **0**.

1.3.13. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours. You must complete your cluster installation and keep the cluster running for 24 hours in a non-degraded state to ensure that the first certificate rotation has finished.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines and compute machineSets:

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-
cluster-api_worker-machineset-*.yaml
```

Because you create and manage these resources yourself, you do not have to initialize them.

- You can preserve the MachineSet files to create compute machines by using the machine API, but you must update references to them to match your environment.
3. Modify the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:
 - a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** file.
 - b. Locate the **mastersSchedulable** parameter and set its value to **False**.
 - c. Save and exit the file.



NOTE

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

4. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5. Export the metadata file's **infraID** key as an environment variable:

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

TIP

Extract the **infraID** key from **metadata.json** and use it as a prefix for all of the RHOSP resources that you create. By doing so, you avoid name conflicts when making multiple deployments in the same project.

1.3.14. Preparing the bootstrap Ignition files

The OpenShift Container Platform installation process relies on bootstrap machines that are created from a bootstrap Ignition configuration file.

Edit the file and upload it. Then, create a secondary bootstrap Ignition configuration file that RHOSP uses to download the primary file.

Prerequisites

- You have the bootstrap Ignition file that the installer program generates, **bootstrap.ign**.
- The infrastructure ID from the installer's metadata file is set as an environment variable (**\$INFRA_ID**).
 - If the variable is not set, see **Creating the Kubernetes manifest and Ignition config files**
- You have an HTTP(S)-accessible way to store the bootstrap ignition file.
 - The documented procedure uses the OpenStack Image service (Glance), but you can also use the OpenStack Storage service (Swift), Amazon S3, an internal HTTP server, or an ad hoc Nova server.

Procedure

1. Run the following Python script. The script modifies the bootstrap Ignition file to set the host name and, if available, CA certificate file when it runs:

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64,
        'verification': {}
    },
    'filesystem': 'root',
})

ca_cert_path = os.environ.get('OS_CACERT', "")
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64,
            'verification': {}
        },
        'filesystem': 'root',
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

2. Using the OpenStack CLI, create an image that uses the bootstrap Ignition file:

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

3. Get the image's details:

```
$ openstack image show <image_name>
```

Make a note of the **file** value; it follows the pattern **v2/images/<image_ID>/file**.



NOTE

Verify that the image you created is active.

4. Retrieve the Image service's public address:

```
$ openstack catalog show image
```

5. Combine the public address with the image **file** value and save the result as the storage location. The location follows the pattern **<image_service_public_URL>/v2/images/<image_ID>/file**.

6. Generate an auth token and save the token ID:

```
$ openstack token issue -c id -f value
```

7. Insert the following content into a file called **\$INFRA_ID-bootstrap-ignition.json** and edit the placeholders to match your own values:

```
{
  "ignition": {
    "config": {
      "append": [{
        "source": "<storage_url>", ❶
        "verification": {},
        "httpHeaders": [{
          "name": "X-Auth-Token", ❷
          "value": "<token_ID>" ❸
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>", ❹
          "verification": {}
        }]
      }
    },
    "timeouts": {},
    "version": "2.4.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

❶ Replace the value of **ignition.config.append.source** with the bootstrap Ignition file storage URL.

❷ Set **name** in **httpHeaders** to **"X-Auth-Token"**.

- 3 Set **value** in **httpHeaders** to your token's ID.
- 4 If the bootstrap Ignition file server uses a self-signed certificate, include the Base64-encoded certificate.

8. Save the secondary Ignition config file.

The bootstrap Ignition data will be passed to RHOSP during installation.



WARNING

The bootstrap Ignition file contains sensitive information, like **clouds.yaml** credentials. Ensure that you store it in a secure place, and delete it after you complete the installation process.

1.3.15. Creating control plane Ignition config files

Installing OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) on your own infrastructure requires control plane Ignition config files. You must create multiple config files.



NOTE

As with the bootstrap Ignition configuration, you must explicitly define a host name for each control plane machine.

Prerequisites

- The infrastructure ID from the installation program's metadata file is set as an environment variable (**\$INFRA_ID**)
 - If the variable is not set, see **Creating the Kubernetes manifest and Ignition config files**

Procedure

- On a command line, run the following Python script:

```
$ for index in $(seq 0 2); do
  MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
  python -c "import base64, json, sys;
  ignition = json.load(sys.stdin);
  files = ignition['storage'].get('files', []);
  files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
  ignition['storage']['files'] = files;
  json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

You now have three control plane Ignition files: **<INFRA_ID>-master-0-ignition.json**, **<INFRA_ID>-master-1-ignition.json**, and **<INFRA_ID>-master-2-ignition.json**.

1.3.16. Creating network resources

Create the network resources that a OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) installation on your own infrastructure requires. To save time, run supplied Ansible playbooks that generate security groups, networks, subnets, routers, and ports.

Procedure

1. Insert the following content into a local file that is called **common.yaml**:

```
- hosts: localhost
  gather_facts: no

  vars_files:
    - metadata.json

  tasks:
    - name: 'Compute resource names'
      set_fact:
        cluster_id_tag: "openshiftClusterID={{ infraID }}"
        os_network: "{{ infraID }}-network"
        os_subnet: "{{ infraID }}-nodes"
        os_router: "{{ infraID }}-external-router"
        # Port names
        os_port_api: "{{ infraID }}-api-port"
        os_port_ingress: "{{ infraID }}-ingress-port"
        os_port_bootstrap: "{{ infraID }}-bootstrap-port"
        os_port_master: "{{ infraID }}-master-port"
        os_port_worker: "{{ infraID }}-worker-port"
        # Security groups names
        os_sg_master: "{{ infraID }}-master"
        os_sg_worker: "{{ infraID }}-worker"
        # Server names
        os_bootstrap_server_name: "{{ infraID }}-bootstrap"
        os_cp_server_name: "{{ infraID }}-master"
        os_compute_server_name: "{{ infraID }}-worker"
        # Trunk names
        os_cp_trunk_name: "{{ infraID }}-master-trunk"
        os_compute_trunk_name: "{{ infraID }}-worker-trunk"
        # Subnet pool name
        subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
        # Service network name
        os_svc_network: "{{ infraID }}-kuryr-service-network"
        # Service subnet name
        os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
        # Ignition files
        os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"
```

2. Insert the following content into a local file that is called **inventory.yaml**:

```
all:
  hosts:
```

```

localhost:
  ansible_connection: local
  ansible_python_interpreter: "{{ansible_playbook_python}}"

# User-provided values
os_subnet_range: '10.0.0.0/16'
os_flavor_master: 'm1.xlarge'
os_flavor_worker: 'm1.large'
os_image_rhcos: 'rhcos'
os_external_network: 'external'
# OpenShift API floating IP address
os_api_fip: '203.0.113.23'
# OpenShift Ingress floating IP address
os_ingress_fip: '203.0.113.19'
# Service subnet cidr
svc_subnet_range: '172.30.0.0/16'
os_svc_network_range: '172.30.0.0/15'
# Subnet pool prefixes
cluster_network_cidrs: '10.128.0.0/14'
# Subnet pool prefix length
host_prefix: '23'
# Name of the SDN.
# Possible values are OpenshiftSDN or Kuryr.
os_networking_type: 'OpenshiftSDN'

# Number of provisioned Control Plane nodes
# 3 is the minimum number for a fully-functional cluster.
os_cp_nodes_number: 3

# Number of provisioned Compute nodes.
# 3 is the minimum number for a fully-functional cluster.
os_compute_nodes_number: 3

```

3. Insert the following content into a local file that is called **01_security-groups.yaml**

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'Create the master security group'
      os_security_group:
        name: "{{ os_sg_master }}"

    - name: 'Set master security group tag'
      command:
        cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_master }}"

    - name: 'Create the worker security group'

```



```

os_security_group:
  name: "{{ os_sg_worker }}"

- name: 'Set worker security group tag'
  command:
    cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_worker }}"

- name: 'Create master-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: icmp

- name: 'Create master-sg rule "machine config server"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 22623
    port_range_max: 22623

- name: 'Create master-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create master-sg rule "DNS (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: tcp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "DNS (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create master-sg rule "OpenShift API"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    port_range_min: 6443

```

```

    port_range_max: 6443

- name: 'Create master-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create master-sg rule "VXLAN from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create master-sg rule "Geneve"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create master-sg rule "Geneve from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create master-sg rule "ovndb"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 6641
    port_range_max: 6642

- name: 'Create master-sg rule "ovndb from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 6641
    port_range_max: 6642

- name: 'Create master-sg rule "master ingress internal (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 9000

```

```

    port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal from worker (TCP)"
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal (UDP)"
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal from worker (UDP)"
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "kube scheduler"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 10259
    port_range_max: 10259

- name: 'Create master-sg rule "kube scheduler from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 10259
    port_range_max: 10259

- name: 'Create master-sg rule "kube controller manager"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 10257
    port_range_max: 10257

- name: 'Create master-sg rule "kube controller manager from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 10257

```

```
port_range_max: 10257

- name: 'Create master-sg rule "master ingress kubelet secure"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create master-sg rule "master ingress kubelet secure from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create master-sg rule "etcd"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 2379
    port_range_max: 2380

- name: 'Create master-sg rule "master ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (TCP) from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (UDP) from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 30000
```

```

port_range_max: 32767

- name: 'Create master-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"

- name: 'Create worker-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: icmp

- name: 'Create worker-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create worker-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create worker-sg rule "Ingress HTTP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 80
    port_range_max: 80

- name: 'Create worker-sg rule "Ingress HTTPS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 443
    port_range_max: 443

- name: 'Create worker-sg rule "router"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 1936
    port_range_max: 1936

- name: 'Create worker-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"

```

```
port_range_min: 4789
port_range_max: 4789

- name: 'Create worker-sg rule "VXLAN from master"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create worker-sg rule "Geneve"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create worker-sg rule "Geneve from master"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create worker-sg rule "worker ingress internal (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal from master (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal from master (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
```

```

    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress kubelet secure"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create worker-sg rule "worker ingress kubelet secure from master"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create worker-sg rule "worker ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (TCP) from master"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (UDP) from master"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"

```

4. Insert the following content into a local file that is called **02_network.yaml**

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the cluster network'
  os_network:
    name: "{{ os_network }}"

- name: 'Set the cluster network tag'
  command:
    cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_network }}"

- name: 'Create a subnet'
  os_subnet:
    name: "{{ os_subnet }}"
    network_name: "{{ os_network }}"
    cidr: "{{ os_subnet_range }}"
    allocation_pool_start: "{{ os_subnet_range | next_nth_usable(10) }}"
    allocation_pool_end: "{{ os_subnet_range | ipaddr('last_usable') }}"

- name: 'Set the cluster subnet tag'
  command:
    cmd: "openstack subnet set --tag {{ cluster_id_tag }} {{ os_subnet }}"

- name: 'Create the service network'
  os_network:
    name: "{{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

- name: 'Set the service network tag'
  command:
    cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

- name: 'Computing facts for service subnet'
  set_fact:
    first_ip_svc_subnet_range: "{{ svc_subnet_range | ipv4('network') }}"
    last_ip_svc_subnet_range: "{{ svc_subnet_range | ipaddr('last_usable') | ipmath(1) }}"
    first_ip_os_svc_network_range: "{{ os_svc_network_range | ipv4('network') }}"
    last_ip_os_svc_network_range: "{{ os_svc_network_range | ipaddr('last_usable') | ipmath(1) }}"
    allocation_pool: ""
    when: os_networking_type == "Kuryr"

- name: 'Get first part of OpenStack network'

```



```

    set_fact:
      allocation_pool: "{{ allocation_pool + '--allocation-pool start={{
first_ip_os_svc_network_range | ipmath(1) }},end={{ first_ip_svc_subnet_range | ipmath(-1) }}'
}}"
    when:
      - os_networking_type == "Kuryr"
      - first_ip_svc_subnet_range != first_ip_os_svc_network_range

- name: 'Get last part of OpenStack network'
  set_fact:
    allocation_pool: "{{ allocation_pool + '--allocation-pool start={{ last_ip_svc_subnet_range
| ipmath(1) }},end={{ last_ip_os_svc_network_range | ipmath(-1) }}' }}"
  when:
    - os_networking_type == "Kuryr"
    - last_ip_svc_subnet_range != last_ip_os_svc_network_range

- name: 'Get end of allocation'
  set_fact:
    gateway_ip: "{{ allocation_pool.split('=')[-1] }}"
  when: os_networking_type == "Kuryr"

- name: 'replace last IP'
  set_fact:
    allocation_pool: "{{ allocation_pool | replace(gateway_ip, gateway_ip | ipmath(-1)) }}"
  when: os_networking_type == "Kuryr"

- name: 'list service subnet'
  command:
    cmd: "openstack subnet list --name {{ os_svc_subnet }} --tag {{ cluster_id_tag }}"
  when: os_networking_type == "Kuryr"
  register: svc_subnet

- name: 'Create the service subnet'
  command:
    cmd: "openstack subnet create --ip-version 4 --gateway {{ gateway_ip }} --subnet-range {{
os_svc_network_range }} {{ allocation_pool }} --no-dhcp --network {{ os_svc_network }} --tag
{{ cluster_id_tag }} {{ os_svc_subnet }}"
  when:
    - os_networking_type == "Kuryr"
    - svc_subnet.stdout == ""

- name: 'list subnet pool'
  command:
    cmd: "openstack subnet pool list --name {{ subnet_pool }} --tags {{ cluster_id_tag }}"
  when: os_networking_type == "Kuryr"
  register: pods_subnet_pool

- name: 'Create pods subnet pool'
  command:
    cmd: "openstack subnet pool create --default-prefix-length {{ host_prefix }} --pool-prefix {{
cluster_network_cidrs }} --tag {{ cluster_id_tag }} {{ subnet_pool }}"
  when:
    - os_networking_type == "Kuryr"
    - pods_subnet_pool.stdout == ""

- name: 'Create external router'

```

```

os_router:
  name: "{{ os_router }}"
  network: "{{ os_external_network }}"
  interfaces:
    - "{{ os_subnet }}"

- name: 'Set external router tag'
  command:
    cmd: "openstack router set --tag {{ cluster_id_tag }} {{ os_router }}"
  when: os_networking_type == "Kuryr"

- name: 'Create the API port'
  os_port:
    name: "{{ os_port_api }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_master }}"
    fixed_ips:
      - subnet: "{{ os_subnet }}"
        ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"

- name: 'Set API port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_api }}"

- name: 'Create the Ingress port'
  os_port:
    name: "{{ os_port_ingress }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_worker }}"
    fixed_ips:
      - subnet: "{{ os_subnet }}"
        ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"

- name: 'Set the Ingress port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_ingress }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the API floating IP to API port'
  command:
    cmd: "openstack floating ip set --port {{ os_port_api }} {{ os_api_fip }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the Ingress floating IP to Ingress port'
  command:
    cmd: "openstack floating ip set --port {{ os_port_ingress }} {{ os_ingress_fip }}"

```

- On a command line, create security groups by running the first numbered playbook:

```
$ ansible-playbook -i inventory.yaml 01_security-groups.yaml
```

- On a command line, create a network, subnet, and router by running the second numbered playbook:

```
$ ansible-playbook -i inventory.yaml 02_network.yaml
```

- Optional:* If you want to control the default resolvers that Nova servers use, run the OpenStack CLI command:

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

1.3.17. Creating the bootstrap machine

Create a bootstrap machine and give it the network access it needs to run on Red Hat OpenStack Platform (RHOSP). Red Hat provides an Ansible playbook that you run to simplify this process.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The **metadata.yaml** file that the installation program created is in the same directory as the Ansible playbooks

Procedure

- On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
- Insert the following content into a local file that is called **03_bootstrap.yaml**:

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the bootstrap server port'
  os_port:
    name: "{{ os_port_bootstrap }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_master }}"
    allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"
```

```

- name: 'Set bootstrap port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_bootstrap }}"

- name: 'Create the bootstrap server'
  os_server:
    name: "{{ os_bootstrap_server_name }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_master }}"
    userdata: "{{ lookup('file', os_bootstrap_ignition) | string }}"
    auto_ip: no
    nics:
      - port-name: "{{ os_port_bootstrap }}"

- name: 'Create the bootstrap floating IP'
  os_floating_ip:
    state: present
    network: "{{ os_external_network }}"
    server: "{{ os_bootstrap_server_name }}"

```

3. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml 03_bootstrap.yaml
```

4. After the bootstrap server is active, view the logs to verify that the Ignition files were received:

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

1.3.18. Creating the control plane machines

Create three control plane machines by using the Ignition config files that you generated.

Prerequisites

- The infrastructure ID from the installation program's metadata file is set as an environment variable (**\$INFRA_ID**)
- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The three Ignition files created in **Creating control plane Ignition config files**

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. If the control plane Ignition config files aren't already in your working directory, copy them into it.
3. Insert the following content into a local file that is called **04_control-plane.yaml**:

```

# Required Python packages:
#

```

```

# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'Create the Control Plane ports'
      os_port:
        name: "{{ item.1 }}-{{ item.0 }}"
        network: "{{ os_network }}"
        security_groups:
          - "{{ os_sg_master }}"
        allowed_address_pairs:
          - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
          - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"
          - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
        with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"
        register: ports

    - name: 'Set Control Plane ports tag'
      command:
        cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
        with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"

    - name: 'List the Control Plane Trunks'
      command:
        cmd: "openstack network trunk list"
      when: os_networking_type == "Kuryr"
      register: control_plane_trunks

    - name: 'Create the Control Plane trunks'
      command:
        cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{ os_cp_trunk_name
        }}-{{ item.0 }}"
      with_indexed_items: "{{ ports.results }}"
      when:
        - os_networking_type == "Kuryr"
        - "os_cp_trunk_name|string not in control_plane_trunks.stdout"

    - name: 'Create the Control Plane servers'
      os_server:
        name: "{{ item.1 }}-{{ item.0 }}"
        image: "{{ os_image_rhcos }}"
        flavor: "{{ os_flavor_master }}"
        auto_ip: no
        # The ignition filename will be concatenated with the Control Plane node
        # name and its 0-indexed serial number.
        # In this case, the first node will look for this filename:
        #   "{{ infraID }}-master-0-ignition.json"
        userdata: "{{ lookup('file', [item.1, item.0, 'ignition.json'] | join('-')) | string }}"

```

```

    nics:
      - port-name: "{{ os_port_master }}-{{ item.0 }}"
        with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"

```

4. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml 04_control-plane.yaml
```

5. Run the following command to monitor the bootstrapping process:

```
$ openshift-install wait-for bootstrap-complete
```

You will see messages that confirm that the control plane machines are running and have joined the cluster:

```

INFO API v1.14.6+f9b5405 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources

```

1.3.19. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
system:admin
```

1.3.20. Deleting bootstrap resources

Delete the bootstrap resources that you no longer need.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The control plane machines are running
 - If you don't know the machines' status, see **Verifying cluster status**

Procedure

1. Insert the following content into a local file that is called **down-03_bootstrap.yaml**:

```
# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'Remove the bootstrap server'
      os_server:
        name: "{{ os_bootstrap_server_name }}"
        state: absent
        delete_fip: yes

    - name: 'Remove the bootstrap server port'
      os_port:
        name: "{{ os_port_bootstrap }}"
        state: absent
```

2. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml down-03_bootstrap.yaml
```

The bootstrap port, server, and floating IP address are deleted.



WARNING

If you have not disabled the bootstrap Ignition file URL, do so now.

1.3.21. Creating compute machines

After standing up the control plane, create compute machines.

Prerequisites

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The **metadata.yaml** file that the installation program created is in the same directory as the Ansible playbooks
- The control plane is active

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. Insert the following content into a local file that is called **05_compute-nodes.yaml**:

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the Compute ports'
  os_port:
    name: "{{ item.1 }}-{{ item.0 }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_worker }}"
    allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
  with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
  register: ports

- name: 'Set Compute ports tag'
  command:
    cmd: "openstack port set --tag {{ [cluster_id_tag] }} {{ item.1 }}-{{ item.0 }}"
  with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

- name: 'List the Compute Trunks'
  command:
    cmd: "openstack network trunk list"
  when: os_networking_type == "Kuryr"
  register: compute_trunks

- name: 'Create the Compute trunks'
  command:
    cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_compute_trunk_name }}-{{ item.0 }}"
```



```

with_indexed_items: "{{ ports.results }}"
when:
- os_networking_type == "Kuryr"
- "os_compute_trunk_name|string not in compute_trunks.stdout"

- name: 'Create the Compute servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_worker }}"
    auto_ip: no
    userdata: "{{ lookup('file', 'worker.ign') | string }}"
    nics:
      - port-name: "{{ os_port_worker }}-{{ item.0 }}"
  with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"

```

3. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml 05_compute-nodes.yaml
```

Next steps

- Approve the machines' certificate signing requests

1.3.22. Approving the CSRs for your machines

When you add machines to a cluster, two pending certificates signing request (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes

NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready    master   63m   v1.17.1
master-1  Ready    master   63m   v1.17.1
master-2  Ready    master   64m   v1.17.1
worker-0  NotReady worker   76s   v1.17.1
worker-1  NotReady worker   70s   v1.17.1

```

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that the you see a client and server request with **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending 1
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending 2
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

1 A client request CSR.

2 A server request CSR.

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}\n'}}{{end}}{{end}}' | xargs oc adm certificate approve
```

1.3.23. Verifying a successful installation

Verify that the OpenShift Container Platform installation is complete.

Prerequisites

- You have the installation program (**openshift-install**)

Procedure

- On a command line, enter:

```
$ openshift-install --log-level debug wait-for install-complete
```

The program outputs the console URL, as well as the administrator's login information.

1.3.24. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```

NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#).

1.4. INSTALLING A CLUSTER ON OPENSTACK WITH KURYR ON YOUR OWN INFRASTRUCTURE

In OpenShift Container Platform version 4.4, you can install a cluster on Red Hat OpenStack Platform (RHOSP) that runs on user-provisioned infrastructure.

Using your own infrastructure allows you to integrate your cluster with existing infrastructure and modifications. The process requires more labor on your part than installer-provisioned installations, because you must create all RHOSP resources, like Nova servers, Neutron ports, and security groups. However, Red Hat provides Ansible playbooks to help you in the deployment process.

Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
 - Verify that OpenShift Container Platform 4.4 is compatible with your RHOSP version in the *Available platforms* section. You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#).
- Have an RHOSP account where you want to install OpenShift Container Platform
- On the machine from which you run the installation program, have:
 - A single directory in which you can keep the files you create during the installation process
 - Python 3

1.4.1. About Kuryr SDN

Kuryr is a container network interface (CNI) plug-in solution that uses [OpenStack Neutron](#) and [OpenStack Octavia](#) to provide networking for Pods and Services.

Kuryr and OpenShift Container Platform integration is primarily designed for OpenShift Container Platform clusters running on OpenStack VMs. Kuryr improves the network performance by plugging OpenShift Pods into OpenStack SDN. In addition, it provides interconnectivity between OpenShift Pods and OpenStack virtual instances.

Kuryr components are installed as Pods in OpenShift Container Platform using the **openshift-kuryr** namespace:

- **kuryr-controller** - a single Service instance installed on a **master** node. This is modeled in OpenShift Container Platform as a **Deployment**.
- **kuryr-cni** - a container installing and configuring Kuryr as a CNI driver on each OpenShift Container Platform node. This is modeled in OpenShift Container Platform as a **DaemonSet**.

The Kuryr controller watches the OpenShift API server for Pod, Service, and namespace create, update, and delete events. It maps the OpenShift Container Platform API calls to corresponding objects in Neutron and Octavia. This means that every network solution that implements the Neutron trunk port functionality can be used to back OpenShift Container Platform via Kuryr. This includes open source solutions such as Open vSwitch (OVS) and Open Virtual Network (OVN) as well as Neutron-compatible commercial SDNs.

Kuryr is recommended for OpenShift deployments on encapsulated OpenStack tenant networks to avoid double encapsulation, such as running an encapsulated OpenShift SDN over an OpenStack network.

If you use provider networks or tenant VLANs, you do not need to use Kuryr to avoid double encapsulation. The performance benefit is negligible. Depending on your configuration, though, using Kuryr to avoid having two overlays might still be beneficial.

Kuryr is not recommended in deployments where all of the following criteria are true:

- The RHOSP version is less than 16.
- The deployment uses UDP services, or a large number of TCP services on few hypervisors.

or

- The **ovn-octavia** Octavia driver is disabled.
- The deployment uses a large number of TCP services on few hypervisors.

1.4.2. Resource guidelines for installing OpenShift Container Platform on OpenStack with Kuryr

When using Kuryr SDN, the Pods, Services, namespaces, and network policies are using resources from the RHOSP quota; this increases the minimum requirements. Kuryr also has some additional requirements on top of what a default install requires.

Use the following quota to satisfy a default cluster's minimum requirements:

Table 1.12. Recommended resources for a default OpenShift Container Platform cluster on RHOSP with Kuryr

Resource	Value
Floating IP addresses	3 - plus the expected number of Services of LoadBalancer type
Ports	1500 - 1 needed per Pod
Routers	1
Subnets	250 - 1 needed per Namespace/Project
Networks	250 - 1 needed per Namespace/Project
RAM	112 GB
vCPUs	28
Volume storage	275 GB
Instances	7
Security groups	250 - 1 needed per Service and per NetworkPolicy

Resource	Value
Security group rules	1000
Load balancers	100 - 1 needed per Service
Load balancer listeners	500 - 1 needed per Service-exposed port
Load balancer pools	500 - 1 needed per Service-exposed port

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If OpenStack Object Storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



IMPORTANT

If you are using Red Hat OpenStack Platform (RHOSP) version 16 with the Amphora driver rather than the OVN Octavia driver, security groups are associated with Service accounts instead of user projects.

Take the following notes into consideration when setting resources:

- The number of ports that are required is larger than the number of Pods. Kuryr uses ports pools to have pre-created ports ready to be used by Pods and speed up the Pods' booting time.
- Each NetworkPolicy is mapped into an RHOSP security group, and depending on the NetworkPolicy spec, one or more rules are added to the security group.
- Each Service is mapped to an RHOSP load balancer. Consider this requirement when estimating the number of security groups required for the quota.
If you are using RHOSP version 15 or earlier, or the **ovn-octavia driver**, each load balancer has a security group with the user project.
- Swift space requirements vary depending on the size of the bootstrap Ignition file and image registry.
- The quota does not account for load balancer resources (such as VM resources), but you must consider these resources when you decide the RHOSP deployment's size. The default installation will have more than 50 load balancers; the clusters must be able to accommodate them.
If you are using RHOSP version 16 with the OVN Octavia driver enabled, only one load balancer VM is generated; Services are load balanced through OVN flows.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

To enable Kuryr SDN, your environment must meet the following requirements:

- Run OpenStack 13+.
- Have Overcloud with Octavia.
- Use Neutron Trunk ports extension.
- Use **openvswitch** firewall driver if ML2/OVS Neutron driver is used instead of **ovs-hybrid**.

1.4.2.1. Increasing quota

When using Kuryr SDN, you must increase quotas to satisfy the OpenStack resources used by Pods, Services, namespaces, and network policies.

Procedure

- Increase the quotas for a project by running the following command:

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --subnets
250 --networks 250 <project>
```

1.4.2.2. Configuring Neutron

Kuryr CNl leverages the Neutron Trunks extension to plug containers into the OpenStack SDN, so you must use the **trunks** extension for Kuryr to properly work.

In addition, if you leverage the default ML2/OVS Neutron driver, the firewall must be set to **openvswitch** instead of **ovs_hybrid** so that security groups are enforced on trunk subports and Kuryr can properly handle network policies.

1.4.2.3. Configuring Octavia

Kuryr SDN uses OpenStack Octavia LBaaS to implement OpenShift Services. Thus, you must install and configure Octavia components in your OpenStack environment to use Kuryr SDN.

To enable Octavia, you must include the Octavia Service during the installation of the OpenStack Overcloud, or upgrade the Octavia Service if the Overcloud already exists. The following steps for enabling Octavia apply to both a clean install of the Overcloud or an Overcloud update.



NOTE

The following steps only capture the key pieces required during the [deployment of OpenStack](#) when dealing with Octavia. It is also important to note that [registry methods](#) vary.

This example uses the local registry method.

Procedure

1. If you are using the local registry, create a template to upload the images to the registry. For example:

```
(undercloud) $ openstack overcloud container image prepare \
```

```
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack- \
--tag-from-label {version}-{release} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. Verify that the **local_registry_images.yaml** file contains the Octavia images. For example:

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-manager:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```



NOTE

The Octavia container versions vary depending upon the specific RHOSP release installed.

3. Pull the container images from registry.redhat.io to the Undercloud node:

```
(undercloud) $ sudo openstack overcloud container image upload \
--config-file /home/stack/local_registry_images.yaml \
--verbose
```

This may take some time depending on the speed of your network and Undercloud disk.

4. Since an Octavia load balancer is used to access the OpenShift API, you must increase their listeners' default timeouts for the connections. The default timeout is 50 seconds. Increase the timeout to 20 minutes by passing the following file to the Overcloud deploy command:

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```



NOTE

This is not needed for Red Hat OpenStack Platform 14+.

5. Install or update your Overcloud environment with Octavia:

```
openstack overcloud deploy --templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
-e octavia_timeouts.yaml
```


**NOTE**

This command only includes the files associated with Octavia; it varies based on your specific installation of OpenStack. See the official OpenStack documentation for further information. For more information on customizing your Octavia installation, see [installation of Octavia using Director](#).

**NOTE**

When leveraging Kuryr SDN, the Overcloud installation requires the Neutron **trunk** extension. This is available by default on Director deployments. Use the **openvswitch** firewall instead of the default **ovs-hybrid** when the Neutron backend is ML2/OVS. There is no need for modifications if the backend is ML2/OVN.

6. In RHOSP versions 13 and 15, add the project ID to the **octavia.conf** configuration file after you create the project.
 - To enforce network policies across Services, like when traffic goes through the Octavia load balancer, you must ensure Octavia creates the Amphora VM security groups on the user project. This change ensures that required LoadBalancer security groups belong to that project, and that they can be updated to enforce Services isolation.

**NOTE**

This task is unnecessary in RHOSP version 16 or later.

Octavia implements a new ACL API that restricts access to the Load Balancers VIP.

- a. Get the project ID

```
$ openstack project show <project>
+-----+-----+
| Field  | Value                |
+-----+-----+
| description |                    |
| domain_id | default              |
| enabled   | True                 |
| id        | PROJECT_ID           |
| is_domain | False                |
| name      | *<project>*          |
| parent_id | default              |
| tags      | []                   |
+-----+-----+
```

- b. Add the project ID to **octavia.conf** for the controllers.
 - i. List the Overcloud controllers.

```
$ source stackrc # Undercloud credentials
$ openstack server list
```

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

ID	Name	Status	Networks
Image	Flavor		
+-----+-----+-----+-----+			
+-----+-----+			
6bef8e73-2ba5-4860-a0b1-3937f8ca7e01	controller-0	ACTIVE	
ctlplane=192.168.24.8 overcloud-full controller			
dda3173a-ab26-47f8-a2dc-8473b4a67ab9	compute-0	ACTIVE	
ctlplane=192.168.24.6 overcloud-full compute			
+-----+-----+-----+-----+			
+-----+-----+			

- ii. SSH into the controller(s).

```
$ ssh heat-admin@192.168.24.8
```

- iii. Edit the **octavia.conf** to add the project into the list of projects where Amphora security groups are on the user's account.

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

- c. Restart the Octavia worker so the new configuration loads.

```
controller-0$ sudo docker restart octavia_worker
```



NOTE

Depending on your RHOSP environment, Octavia might not support UDP listeners. If you use Kuryr SDN on OpenStack version 15 or earlier, UDP services are not supported. RHOSP version 16 or later support UDP.

1.4.2.3.1. The Octavia OVN Driver

Octavia supports multiple provider drivers through the Octavia API.

To see all available Octavia provider drivers, on a command line, enter:

```
$ openstack loadbalancer provider list
```

The result is a list of drivers:

+-----+-----+	
name	description
+-----+-----+	
amphora	The Octavia Amphora driver.

octavia	Deprecated alias of the Octavia Amphora driver.
ovn	Octavia OVN driver.
+-----+	

Beginning with RHOSP version 16, the Octavia OVN provider driver (**ovn**) is supported on OpenShift Container Platform on RHOSP deployments.

ovn is an integration driver for the load balancing that Octavia and OVN provide. It supports basic load balancing capabilities, and is based on OpenFlow rules. The driver is automatically enabled in Octavia by Director on deployments that use OVN Neutron ML2.

The Amphora provider driver is the default driver. If **ovn** is enabled, however, Kuryr uses it.

If Kuryr uses **ovn** instead of Amphora, it offers the following benefits:

- Decreased resource requirements. Kuryr does not require a load balancer VM for each Service.
- Reduced network latency.
- Increased service creation speed by using OpenFlow rules instead of a VM for each Service.
- Distributed load balancing actions across all nodes instead of centralized on Amphora VMs.

1.4.2.4. Known limitations of installing with Kuryr

Using OpenShift Container Platform with Kuryr SDN has several known limitations.

RHOSP version limitations

Using OpenShift Container Platform with Kuryr SDN has several limitations that depend on the RHOSP version.

- RHOSP versions before 16 use the default Octavia load balancer driver (Amphora). This driver requires that one Amphora load balancer VM is deployed per OpenShift Service. Creating too many Services can cause you to run out of resources.
Deployments of later versions of RHOSP that have the OVN Octavia driver disabled also use the Amphora driver. They are subject to the same resource concerns as earlier versions of RHOSP.
- Octavia RHOSP versions before 16 do not support UDP listeners. Therefore, OpenShift UDP services are not supported.
- Octavia RHOSP versions before 16 cannot listen to multiple protocols on the same port. Services that expose the same port to different protocols, like TCP and UDP, are not supported.



IMPORTANT

The OVN Octavia driver does not support listeners that use different protocols on any RHOSP version.

RHOSP environment limitations

There are limitations when using Kuryr SDN that depend on your deployment environment.

Because of Octavia's lack of support for the UDP protocol and multiple listeners, Kuryr forces Pods to use TCP for DNS resolution if:

- The RHOSP version is earlier than 16
- The OVN Octavia driver is used

In Go versions 1.12 and earlier, applications that are compiled with CGO support disabled use UDP only. In this case, the native Go resolver does not recognize the **use-vc** option in **resolv.conf**, which controls whether TCP is forced for DNS resolution. As a result, UDP is still used for DNS resolution, which fails.

To ensure that TCP forcing is allowed, compile applications either with the environment variable **CGO_ENABLED** set to **1**, i.e. **CGO_ENABLED=1**, or ensure that the variable is absent.

In Go versions 1.13 and later, TCP is used automatically if DNS resolution using UDP fails.



NOTE

musl-based containers, including Alpine-based containers, do not support the **use-vc** option.

1.4.2.5. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

1.4.2.6. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

1.4.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.4, you require access to the internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require internet access. Before you update the cluster, you update the content of the mirror registry.

1.4.4. Downloading playbook dependencies

The Ansible playbooks that simplify the installation process on user-provisioned infrastructure require several Python modules. On the machine where you will run the installer, add the modules' repositories and then download them.



NOTE

These instructions assume that you are using Red Hat Enterprise Linux 8.

Prerequisites

- Python 3 is installed on your machine

Procedure

1. On a command line, add the repositories:

```
$ sudo subscription-manager register # If not done already
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
$ sudo subscription-manager repos --disable=* # If not done already

$ sudo subscription-manager repos \
--enable=rhel-8-for-x86_64-baseos-rpms \
--enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
--enable=ansible-2.8-for-rhel-8-x86_64-rpms \
--enable=rhel-8-for-x86_64-appstream-rpms
```

2. Install the modules:

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. Ensure that the **python** command points to **python3**:

```
$ sudo alternatives --set python /usr/bin/python3
```

1.4.5. Obtaining the installation program

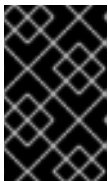
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- A computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

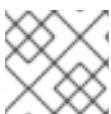
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

1.4.6. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and to the installation program.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's **~/.ssh/authorized_keys** list.

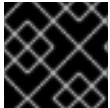
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the SSH key.

Running this command generates an SSH key that does not require a password in the location that you specified.



IMPORTANT

If you create a new SSH key pair, avoid overwriting existing SSH keys.

1. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
Agent pid 31874
```

2. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

1.4.7. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image

The OpenShift Container Platform installation program requires that a Red Hat Enterprise Linux CoreOS (RHCOS) image be present in the Red Hat OpenStack Platform (RHOSP) cluster. Retrieve the latest RHCOS image, then upload it using the RHOSP CLI.

Prerequisites

- The RHOSP CLI is installed.

Procedure

1. Log in to the Red Hat customer portal's [Product Downloads page](#).
2. Under **Version**, select the most recent release of OpenShift Container Platform 4.4 for RHEL 8.

**IMPORTANT**

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

3. Download the *Red Hat Enterprise Linux CoreOS - OpenStack Image (QCOW)* .
4. Decompress the image.

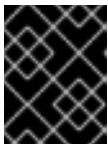
**NOTE**

You must decompress the OpenStack image before the cluster can use it. The name of the downloaded file might not contain a compression extension, like **.gz** or **.tgz**. To find out if or how the file is compressed, in a command line, enter:

```
$ file <name_of_downloaded_file>
```

5. From the image that you downloaded, create an image that is named **rhcos** in your cluster by using the RHOSP CLI:

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-
${RHCOS_VERSION}-openstack.qcow2 rhcos
```

**IMPORTANT**

Depending on your RHOSP environment, you might be able to upload the image in either **.raw** or **.qcow2** formats. If you use Ceph, you must use the **.raw** format.

CAUTION

If the installation program finds multiple images with the same name, it chooses one of them at random. To avoid this behavior, create unique names for resources in RHOSP.

After you upload the image to RHOSP, it is usable in the installation process.

1.4.8. Verifying external network access

The OpenShift Container Platform installation process requires external network access. You must provide an external network value to it, or deployment fails. Before you begin the process, verify that a network with the External router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- [Configure OpenStack's networking service to have DHCP agents forward instances' DNS queries](#)

Procedure

1. Using the RHOSP CLI, verify the name and ID of the 'External' network:


```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

```
+-----+-----+-----+
| ID              | Name          | Router Type |
+-----+-----+-----+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+-----+-----+-----+
```

A network with an External router type appears in the network list. If at least one does not, see [Create an external network](#).



NOTE

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

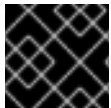
1.4.9. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible by using floating IP addresses.

1.4.9.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.



IMPORTANT

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```

**NOTE**

If you do not control the DNS server you can add the record to your **/etc/hosts** file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

1.4.10. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

1. Create the **clouds.yaml** file:

- If your OpenStack distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.

**IMPORTANT**

Remember to add a password to the **auth** field. You can also keep secrets in [a separate file](#) from **clouds.yaml**.

- If your OpenStack distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:
 - a. Copy the certificate authority file to your machine.

- b. In the command line, run the following commands to add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

- c. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```
clouds:
  shiftstack:
  ...
  cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openstack/clouds.yaml**
 - d. A Unix-specific site configuration directory, for example **/etc/openstack/clouds.yaml**
 The installation program searches for **clouds.yaml** in that order.

1.4.11. Creating the installation files for RHOSP

1.4.12. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on OpenStack.

Prerequisites

- Download the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

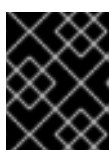
- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **openstack** as the platform to target.
 - iii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
 - iv. Specify the floating IP address to use for external access to the OpenShift API.
 - v. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - vi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - vii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



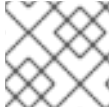
IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

You now have the file **install-config.yaml** in the directory that you specified.

1.4.13. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE


You cannot modify these parameters in the **install-config.yaml** file after installation.



Table 1.13. Required parameters

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. This value is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
controlPlane.platform	The cloud provider to host the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack , or {}
compute.platform	The cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack , or {}
metadata.name	The name of your cluster.	A string that contains uppercase or lowercase letters, such as dev .
platform.<platform>.region	The region to deploy your cluster in.	A valid region for your cloud, such as us-east-1 for AWS, centralus for Azure. Red Hat OpenStack Platform (RHOSP) does not use this parameter.

Parameter	Description	Values
pullSecret	The pull secret that you obtained from the Pull Secret page on the Red Hat OpenShift Cluster Manager site. You use this pull secret to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

Table 1.14. Optional parameters

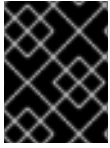
Parameter	Description	Values
sshKey	<p>The SSH key to use to access your cluster machines.</p> <div>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div>	A valid, local public SSH key that you added to the ssh-agent process.
fips	Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.	false or true
publish	How to publish the user-facing endpoints of your cluster.	Internal or External . Set publish to Internal to deploy a private cluster, which cannot be accessed from the internet. The default value is External .

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
controlPlane.replicas	The number of control plane machines to provision.	A positive integer greater than or equal to 3 . The default value is 3 .

1.4.13.1. Sample customized `install-config.yaml` file for OpenStack with Kuryr

To deploy with Kuryr SDN instead of the default OpenShift SDN, you must modify the **install-config.yaml** file to include **Kuryr** as the desired **networking.networkType** and proceed with the default

OpenShift SDN installation steps. This sample **install-config.yaml** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

This sample file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program.

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: m1.large
      replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
    trunkSupport: true
    octaviaSupport: true
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```



NOTE

Both **trunkSupport** and **octaviaSupport** are automatically discovered by the installer, so there is no need to set them. But if your environment does not meet both requirements, Kuryr SDN will not properly work. Trunks are needed to connect the Pods to the OpenStack network and Octavia is required to create the OpenShift Services.

1.4.13.2. Setting a custom subnet for machines

The IP range that the installation program uses by default might not match the Neutron subnet that you create when you install OpenShift Container Platform. If necessary, update the CIDR value for new machines by editing the installation configuration file.

Prerequisites

- You have the **install-config.yaml** file that was generated by the OpenShift Container Platform installation program.

Procedure

1. On a command line, browse to the directory that contains **install-config.yaml**.
2. From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```
python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- 1 Insert a value that matches your intended Neutron subnet, e.g. **192.0.2.0/24**.

- To set the value manually, open the file and set the value of **networking.machineCIDR** to something that matches your intended Neutron subnet.

1.4.13.3. Emptying compute machine pools

To proceed with an installation that uses your own infrastructure, set the number of compute machines in the installation configuration file to zero. Later, you create these machines manually.

Prerequisites

- You have the **install-config.yaml** file that was generated by the OpenShift Container Platform installation program.

Procedure

1. On a command line, browse to the directory that contains **install-config.yaml**.
2. From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```
$ python -c '
import yaml;
path = "install-config.yaml";
```

```
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- To set the value manually, open the file and set the value of **compute.<first entry>.replicas** to **0**.

1.4.13.4. Modifying the network type

By default, the installation program selects the **OpenShiftSDN** network type. To use Kuryr instead, change the value in the installation configuration file that the program generated.

Prerequisites

- You have the file **install-config.yaml** that was generated by the OpenShift Container Platform installation program

Procedure

1. In a command prompt, browse to the directory that contains **install-config.yaml**.
2. From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["networkType"] = "Kuryr";
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- To set the value manually, open the file and set **networking.networkType** to **"Kuryr"**.

1.4.14. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours. You must complete your cluster installation and keep the cluster running for 24 hours in a non-degraded state to ensure that the first certificate rotation has finished.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

INFO Consuming Install Config from target directory

WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines and compute machineSets:

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage these resources yourself, you do not have to initialize them.

- You can preserve the MachineSet files to create compute machines by using the machine API, but you must update references to them to match your environment.
3. Modify the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** Kubernetes manifest file to prevent Pods from being scheduled on the control plane machines:
 - a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** file.
 - b. Locate the **mastersSchedulable** parameter and set its value to **False**.
 - c. Save and exit the file.



NOTE

Currently, due to a [Kubernetes limitation](#), router Pods running on control plane machines will not be reachable by the ingress load balancer. This step might not be required in a future minor version of OpenShift Container Platform.

4. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
└── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

5. Export the metadata file's **infraID** key as an environment variable:

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

TIP

Extract the **infraID** key from **metadata.json** and use it as a prefix for all of the RHOSP resources that you create. By doing so, you avoid name conflicts when making multiple deployments in the same project.

1.4.15. Preparing the bootstrap Ignition files

The OpenShift Container Platform installation process relies on bootstrap machines that are created from a bootstrap Ignition configuration file.

Edit the file and upload it. Then, create a secondary bootstrap Ignition configuration file that RHOSP uses to download the primary file.

Prerequisites

- You have the bootstrap Ignition file that the installer program generates, **bootstrap.ign**.
- The infrastructure ID from the installer's metadata file is set as an environment variable (**\$INFRA_ID**).
 - If the variable is not set, see **Creating the Kubernetes manifest and Ignition config files**
- You have an HTTP(S)-accessible way to store the bootstrap ignition file.
 - The documented procedure uses the OpenStack Image service (Glance), but you can also use the OpenStack Storage service (Swift), Amazon S3, an internal HTTP server, or an ad hoc Nova server.

Procedure

1. Run the following Python script. The script modifies the bootstrap Ignition file to set the host name and, if available, CA certificate file when it runs:

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
```

```

        'path': '/etc/hostname',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64,
            'verification': {}
        },
        'filesystem': 'root',
    })

ca_cert_path = os.environ.get('OS_CACERT', "")
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
        {
            'path': '/opt/openshift/tls/cloud-ca-cert.pem',
            'mode': 420,
            'contents': {
                'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64,
                'verification': {}
            },
            'filesystem': 'root',
        })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)

```

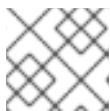
- Using the OpenStack CLI, create an image that uses the bootstrap Ignition file:

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

- Get the image's details:

```
$ openstack image show <image_name>
```

Make a note of the **file** value; it follows the pattern **v2/images/<image_ID>/file**.



NOTE

Verify that the image you created is active.

- Retrieve the Image service's public address:

```
$ openstack catalog show image
```

- Combine the public address with the image **file** value and save the result as the storage location. The location follows the pattern **<image_service_public_URL>/v2/images/<image_ID>/file**.

6. Generate an auth token and save the token ID:

```
$ openstack token issue -c id -f value
```

7. Insert the following content into a file called **\$INFRA_ID-bootstrap-ignition.json** and edit the placeholders to match your own values:

```
{
  "ignition": {
    "config": {
      "append": [{
        "source": "<storage_url>", 1
        "verification": {},
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>", 4
          "verification": {}
        }]
      }
    },
    "timeouts": {},
    "version": "2.4.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

- 1 Replace the value of **ignition.config.append.source** with the bootstrap Ignition file storage URL.
- 2 Set **name** in **httpHeaders** to **"X-Auth-Token"**.
- 3 Set **value** in **httpHeaders** to your token's ID.
- 4 If the bootstrap Ignition file server uses a self-signed certificate, include the Base64-encoded certificate.

8. Save the secondary Ignition config file.

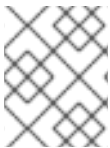
The bootstrap Ignition data will be passed to RHOSP during installation.

**WARNING**

The bootstrap Ignition file contains sensitive information, like **clouds.yaml** credentials. Ensure that you store it in a secure place, and delete it after you complete the installation process.

1.4.16. Creating control plane Ignition config files

Installing OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) on your own infrastructure requires control plane Ignition config files. You must create multiple config files.

**NOTE**

As with the bootstrap Ignition configuration, you must explicitly define a host name for each control plane machine.

Prerequisites

- The infrastructure ID from the installation program's metadata file is set as an environment variable (**\$INFRA_ID**)
 - If the variable is not set, see **Creating the Kubernetes manifest and Ignition config files**

Procedure

- On a command line, run the following Python script:

```
$ for index in $(seq 0 2); do
  MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
  python -c "import base64, json, sys;
  ignition = json.load(sys.stdin);
  files = ignition['storage'].get('files', []);
  files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
  ignition['storage']['files'] = files;
  json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

You now have three control plane Ignition files: **<INFRA_ID>-master-0-ignition.json**, **<INFRA_ID>-master-1-ignition.json**, and **<INFRA_ID>-master-2-ignition.json**.

1.4.17. Creating network resources

Create the network resources that a OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) installation on your own infrastructure requires. To save time, run supplied Ansible playbooks that generate security groups, networks, subnets, routers, and ports.

Procedure

1. Insert the following content into a local file that is called **common.yaml**:

```
- hosts: localhost
  gather_facts: no

  vars_files:
  - metadata.json

  tasks:
  - name: 'Compute resource names'
    set_fact:
      cluster_id_tag: "openshiftClusterID={{ infraID }}"
      os_network: "{{ infraID }}-network"
      os_subnet: "{{ infraID }}-nodes"
      os_router: "{{ infraID }}-external-router"
      # Port names
      os_port_api: "{{ infraID }}-api-port"
      os_port_ingress: "{{ infraID }}-ingress-port"
      os_port_bootstrap: "{{ infraID }}-bootstrap-port"
      os_port_master: "{{ infraID }}-master-port"
      os_port_worker: "{{ infraID }}-worker-port"
      # Security groups names
      os_sg_master: "{{ infraID }}-master"
      os_sg_worker: "{{ infraID }}-worker"
      # Server names
      os_bootstrap_server_name: "{{ infraID }}-bootstrap"
      os_cp_server_name: "{{ infraID }}-master"
      os_compute_server_name: "{{ infraID }}-worker"
      # Trunk names
      os_cp_trunk_name: "{{ infraID }}-master-trunk"
      os_compute_trunk_name: "{{ infraID }}-worker-trunk"
      # Subnet pool name
      subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
      # Service network name
      os_svc_network: "{{ infraID }}-kuryr-service-network"
      # Service subnet name
      os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
      # Ignition files
      os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"
```

2. Insert the following content into a local file that is called **inventory.yaml**:

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'm1.xlarge'
      os_flavor_worker: 'm1.large'
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
      # OpenShift API floating IP address
      os_api_fip: '203.0.113.23'
```



```

# OpenShift Ingress floating IP address
os_ingress_fip: '203.0.113.19'
# Service subnet cidr
svc_subnet_range: '172.30.0.0/16'
os_svc_network_range: '172.30.0.0/15'
# Subnet pool prefixes
cluster_network_cidrs: '10.128.0.0/14'
# Subnet pool prefix length
host_prefix: '23'
# Name of the SDN.
# Possible values are OpenshiftSDN or Kuryr.
os_networking_type: 'OpenshiftSDN'

# Number of provisioned Control Plane nodes
# 3 is the minimum number for a fully-functional cluster.
os_cp_nodes_number: 3

# Number of provisioned Compute nodes.
# 3 is the minimum number for a fully-functional cluster.
os_compute_nodes_number: 3

```

3. Insert the following content into a local file that is called **01_security-groups.yaml**

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the master security group'
  os_security_group:
    name: "{{ os_sg_master }}"

- name: 'Set master security group tag'
  command:
    cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_master }}"

- name: 'Create the worker security group'
  os_security_group:
    name: "{{ os_sg_worker }}"

- name: 'Set worker security group tag'
  command:
    cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_worker }}"

- name: 'Create master-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: icmp

```

```
- name: 'Create master-sg rule "machine config server"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 22623
    port_range_max: 22623

- name: 'Create master-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create master-sg rule "DNS (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: tcp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "DNS (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create master-sg rule "OpenShift API"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    port_range_min: 6443
    port_range_max: 6443

- name: 'Create master-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create master-sg rule "VXLAN from worker"'
  os_security_group_rule:
```

```

    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create master-sg rule "Geneve"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create master-sg rule "Geneve from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create master-sg rule "ovndb"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 6641
    port_range_max: 6642

- name: 'Create master-sg rule "ovndb from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 6641
    port_range_max: 6642

- name: 'Create master-sg rule "master ingress internal (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal from worker (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal (UDP)"'
  os_security_group_rule:

```

```

    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal from worker (UDP)"
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "kube scheduler"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 10259
    port_range_max: 10259

- name: 'Create master-sg rule "kube scheduler from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 10259
    port_range_max: 10259

- name: 'Create master-sg rule "kube controller manager"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 10257
    port_range_max: 10257

- name: 'Create master-sg rule "kube controller manager from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 10257
    port_range_max: 10257

- name: 'Create master-sg rule "master ingress kubelet secure"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create master-sg rule "master ingress kubelet secure from worker"'
  os_security_group_rule:

```

```

    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create master-sg rule "etcd"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 2379
    port_range_max: 2380

- name: 'Create master-sg rule "master ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (TCP) from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (UDP) from worker"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"

- name: 'Create worker-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"

```

```
protocol: icmp

- name: 'Create worker-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create worker-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create worker-sg rule "Ingress HTTP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 80
    port_range_max: 80

- name: 'Create worker-sg rule "Ingress HTTPS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 443
    port_range_max: 443

- name: 'Create worker-sg rule "router"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 1936
    port_range_max: 1936

- name: 'Create worker-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create worker-sg rule "VXLAN from master"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create worker-sg rule "Geneve"'
```

```

os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: udp
  remote_group: "{{ os_sg_worker }}"
  port_range_min: 6081
  port_range_max: 6081

- name: 'Create worker-sg rule "Geneve from master"'
os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: udp
  remote_group: "{{ os_sg_master }}"
  port_range_min: 6081
  port_range_max: 6081

- name: 'Create worker-sg rule "worker ingress internal (TCP)"'
os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: tcp
  remote_group: "{{ os_sg_worker }}"
  port_range_min: 9000
  port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal from master (TCP)"'
os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: tcp
  remote_group: "{{ os_sg_master }}"
  port_range_min: 9000
  port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal (UDP)"'
os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: udp
  remote_group: "{{ os_sg_worker }}"
  port_range_min: 9000
  port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal from master (UDP)"'
os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: udp
  remote_group: "{{ os_sg_master }}"
  port_range_min: 9000
  port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress kubelet secure"'
os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: tcp
  remote_group: "{{ os_sg_worker }}"
  port_range_min: 10250
  port_range_max: 10250

- name: 'Create worker-sg rule "worker ingress kubelet secure from master"'

```

```

os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: tcp
  remote_group: "{{ os_sg_master }}"
  port_range_min: 10250
  port_range_max: 10250

- name: 'Create worker-sg rule "worker ingress services (TCP)'"
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (TCP) from master"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (UDP)'"
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_worker }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (UDP) from master"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_group: "{{ os_sg_master }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"

```

4. Insert the following content into a local file that is called **02_network.yaml**

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

```



```

- hosts: all
gather_facts: no

tasks:
- name: 'Create the cluster network'
  os_network:
    name: "{{ os_network }}"

- name: 'Set the cluster network tag'
  command:
    cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_network }}"

- name: 'Create a subnet'
  os_subnet:
    name: "{{ os_subnet }}"
    network_name: "{{ os_network }}"
    cidr: "{{ os_subnet_range }}"
    allocation_pool_start: "{{ os_subnet_range | next_nth_usable(10) }}"
    allocation_pool_end: "{{ os_subnet_range | ipaddr('last_usable') }}"

- name: 'Set the cluster subnet tag'
  command:
    cmd: "openstack subnet set --tag {{ cluster_id_tag }} {{ os_subnet }}"

- name: 'Create the service network'
  os_network:
    name: "{{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

- name: 'Set the service network tag'
  command:
    cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

- name: 'Computing facts for service subnet'
  set_fact:
    first_ip_svc_subnet_range: "{{ svc_subnet_range | ipv4('network') }}"
    last_ip_svc_subnet_range: "{{ svc_subnet_range | ipaddr('last_usable') | ipmath(1) }}"
    first_ip_os_svc_network_range: "{{ os_svc_network_range | ipv4('network') }}"
    last_ip_os_svc_network_range: "{{ os_svc_network_range | ipaddr('last_usable') | ipmath(1) }}"
    allocation_pool: ""
    when: os_networking_type == "Kuryr"

- name: 'Get first part of OpenStack network'
  set_fact:
    allocation_pool: "{{ allocation_pool + '--allocation-pool start={{ first_ip_os_svc_network_range | ipmath(1) }},end={{ first_ip_svc_subnet_range | ipmath(-1) }}}"
    when:
      - os_networking_type == "Kuryr"
      - first_ip_svc_subnet_range != first_ip_os_svc_network_range

- name: 'Get last part of OpenStack network'
  set_fact:
    allocation_pool: "{{ allocation_pool + '--allocation-pool start={{ last_ip_svc_subnet_range"

```

```

| ipmath(1) }},end={{ last_ip_os_svc_network_range | ipmath(-1) }}' }}"
  when:
    - os_networking_type == "Kuryr"
    - last_ip_svc_subnet_range != last_ip_os_svc_network_range

- name: 'Get end of allocation'
  set_fact:
    gateway_ip: "{{ allocation_pool.split('=')[-1] }}"
  when: os_networking_type == "Kuryr"

- name: 'replace last IP'
  set_fact:
    allocation_pool: "{{ allocation_pool | replace(gateway_ip, gateway_ip | ipmath(-1)) }}"
  when: os_networking_type == "Kuryr"

- name: 'list service subnet'
  command:
    cmd: "openstack subnet list --name {{ os_svc_subnet }} --tag {{ cluster_id_tag }}"
  when: os_networking_type == "Kuryr"
  register: svc_subnet

- name: 'Create the service subnet'
  command:
    cmd: "openstack subnet create --ip-version 4 --gateway {{ gateway_ip }} --subnet-range {{
os_svc_network_range }} {{ allocation_pool }} --no-dhcp --network {{ os_svc_network }} --tag
{{ cluster_id_tag }} {{ os_svc_subnet }}"
  when:
    - os_networking_type == "Kuryr"
    - svc_subnet.stdout == ""

- name: 'list subnet pool'
  command:
    cmd: "openstack subnet pool list --name {{ subnet_pool }} --tags {{ cluster_id_tag }}"
  when: os_networking_type == "Kuryr"
  register: pods_subnet_pool

- name: 'Create pods subnet pool'
  command:
    cmd: "openstack subnet pool create --default-prefix-length {{ host_prefix }} --pool-prefix {{
cluster_network_cidrs }} --tag {{ cluster_id_tag }} {{ subnet_pool }}"
  when:
    - os_networking_type == "Kuryr"
    - pods_subnet_pool.stdout == ""

- name: 'Create external router'
  os_router:
    name: "{{ os_router }}"
    network: "{{ os_external_network }}"
    interfaces:
      - "{{ os_subnet }}"

- name: 'Set external router tag'
  command:
    cmd: "openstack router set --tag {{ cluster_id_tag }} {{ os_router }}"
  when: os_networking_type == "Kuryr"

```

```

- name: 'Create the API port'
  os_port:
    name: "{{ os_port_api }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_master }}"
    fixed_ips:
      - subnet: "{{ os_subnet }}"
        ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"

- name: 'Set API port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_api }}"

- name: 'Create the Ingress port'
  os_port:
    name: "{{ os_port_ingress }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_worker }}"
    fixed_ips:
      - subnet: "{{ os_subnet }}"
        ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"

- name: 'Set the Ingress port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_ingress }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the API floating IP to API port'
  command:
    cmd: "openstack floating ip set --port {{ os_port_api }} {{ os_api_fip }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the Ingress floating IP to Ingress port'
  command:
    cmd: "openstack floating ip set --port {{ os_port_ingress }} {{ os_ingress_fip }}"

```

5. On a command line, create security groups by running the first numbered playbook:

```
$ ansible-playbook -i inventory.yaml 01_security-groups.yaml
```

6. On a command line, create a network, subnet, and router by running the second numbered playbook:

```
$ ansible-playbook -i inventory.yaml 02_network.yaml
```

7. *Optional:* If you want to control the default resolvers that Nova servers use, run the OpenStack CLI command:

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

1.4.18. Creating the bootstrap machine

Create a bootstrap machine and give it the network access it needs to run on Red Hat OpenStack Platform (RHOSP). Red Hat provides an Ansible playbook that you run to simplify this process.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The **metadata.yaml** file that the installation program created is in the same directory as the Ansible playbooks

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. Insert the following content into a local file that is called **03_bootstrap.yaml**:

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the bootstrap server port'
  os_port:
    name: "{{ os_port_bootstrap }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_master }}"
    allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"

- name: 'Set bootstrap port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_bootstrap }}"

- name: 'Create the bootstrap server'
  os_server:
    name: "{{ os_bootstrap_server_name }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_master }}"
    userdata: "{{ lookup('file', os_bootstrap_ignition) | string }}"
    auto_ip: no
    nics:
```

```
- port-name: "{{ os_port_bootstrap }}"

- name: 'Create the bootstrap floating IP'
  os_floating_ip:
    state: present
    network: "{{ os_external_network }}"
    server: "{{ os_bootstrap_server_name }}"
```

3. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml 03_bootstrap.yaml
```

4. After the bootstrap server is active, view the logs to verify that the Ignition files were received:

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

1.4.19. Creating the control plane machines

Create three control plane machines by using the Ignition config files that you generated.

Prerequisites

- The infrastructure ID from the installation program's metadata file is set as an environment variable (**\$INFRA_ID**)
- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The three Ignition files created in **Creating control plane Ignition config files**

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. If the control plane Ignition config files aren't already in your working directory, copy them into it.
3. Insert the following content into a local file that is called **04_control-plane.yaml**:

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the Control Plane ports'
```

```

os_port:
  name: "{{ item.1 }}-{{ item.0 }}"
  network: "{{ os_network }}"
  security_groups:
    - "{{ os_sg_master }}"
  allowed_address_pairs:
    - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
    - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"
    - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
  with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"
  register: ports

- name: 'Set Control Plane ports tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
  with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"

- name: 'List the Control Plane Trunks'
  command:
    cmd: "openstack network trunk list"
  when: os_networking_type == "Kuryr"
  register: control_plane_trunks

- name: 'Create the Control Plane trunks'
  command:
    cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{ os_cp_trunk_name }}-{{ item.0 }}"
  with_indexed_items: "{{ ports.results }}"
  when:
    - os_networking_type == "Kuryr"
    - "os_cp_trunk_name|string not in control_plane_trunks.stdout"

- name: 'Create the Control Plane servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_master }}"
    auto_ip: no
    # The ignition filename will be concatenated with the Control Plane node
    # name and its 0-indexed serial number.
    # In this case, the first node will look for this filename:
    #   "{{ infraID }}-master-0-ignition.json"
    userdata: "{{ lookup('file', [item.1, item.0, 'ignition.json'] | join('-')) | string }}"
    nics:
      - port-name: "{{ os_port_master }}-{{ item.0 }}"
  with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"

```

4. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml 04_control-plane.yaml
```

5. Run the following command to monitor the bootstrapping process:

```
$ openshift-install wait-for bootstrap-complete
```

You will see messages that confirm that the control plane machines are running and have joined the cluster:

```
INFO API v1.14.6+f9b5405 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

1.4.20. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
system:admin
```

1.4.21. Deleting bootstrap resources

Delete the bootstrap resources that you no longer need.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The control plane machines are running
 - If you don't know the machines' status, see **Verifying cluster status**

Procedure

1. Insert the following content into a local file that is called **down-03_bootstrap.yaml**:

```

# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'Remove the bootstrap server'
      os_server:
        name: "{{ os_bootstrap_server_name }}"
        state: absent
        delete_fip: yes

    - name: 'Remove the bootstrap server port'
      os_port:
        name: "{{ os_port_bootstrap }}"
        state: absent

```

2. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml down-03_bootstrap.yaml
```

The bootstrap port, server, and floating IP address are deleted.



WARNING

If you have not disabled the bootstrap Ignition file URL, do so now.

1.4.22. Creating compute machines

After standing up the control plane, create compute machines.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The **metadata.yaml** file that the installation program created is in the same directory as the Ansible playbooks
- The control plane is active

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. Insert the following content into a local file that is called **05_compute-nodes.yaml**:

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the Compute ports'
  os_port:
    name: "{{ item.1 }}-{{ item.0 }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_worker }}"
    allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
  with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
  register: ports

- name: 'Set Compute ports tag'
  command:
    cmd: "openstack port set --tag {{ [cluster_id_tag] }} {{ item.1 }}-{{ item.0 }}"
  with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

- name: 'List the Compute Trunks'
  command:
    cmd: "openstack network trunk list"
  when: os_networking_type == "Kuryr"
  register: compute_trunks

- name: 'Create the Compute trunks'
  command:
    cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_compute_trunk_name }}-{{ item.0 }}"
  with_indexed_items: "{{ ports.results }}"
  when:
    - os_networking_type == "Kuryr"
    - "os_compute_trunk_name|string not in compute_trunks.stdout"

- name: 'Create the Compute servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_worker }}"
    auto_ip: no
    userdata: "{{ lookup('file', 'worker.ign') | string }}"
```

```

    nics:
    - port-name: "{{ os_port_worker }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"

```

3. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml 05_compute-nodes.yaml
```

Next steps

- Approve the machines' certificate signing requests

1.4.23. Approving the CSRs for your machines

When you add machines to a cluster, two pending certificates signing request (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.17.1
master-1	Ready	master	63m	v1.17.1
master-2	Ready	master	64m	v1.17.1
worker-0	NotReady	worker	76s	v1.17.1
worker-1	NotReady	worker	70s	v1.17.1

The output lists all of the machines that you created.

2. Review the pending certificate signing requests (CSRs) and ensure that the you see a client and server request with **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstraptrapper	Pending 1
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstraptrapper	Pending
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending 2
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

- 1 A client request CSR.
- 2 A server request CSR.

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**. You must implement a method of automatically approving the kubelet serving certificate requests.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

1.4.24. Verifying a successful installation

Verify that the OpenShift Container Platform installation is complete.

Prerequisites

- You have the installation program (**openshift-install**)

Procedure

- On a command line, enter:

```
$ openshift-install --log-level debug wait-for install-complete
```

The program outputs the console URL, as well as the administrator's login information.

1.4.25. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```

NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

1.5. UNINSTALLING A CLUSTER ON OPENSTACK

You can remove a cluster that you deployed to Red Hat OpenStack Platform (RHOSP).

1.5.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.

Prerequisites

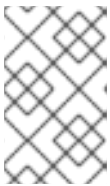
- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

1. From the computer that you used to install the cluster, run the following command:

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2** To view different details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

1.6. UNINSTALLING A CLUSTER ON OPENSTACK FROM YOUR OWN INFRASTRUCTURE

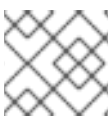
You can remove a cluster that you deployed to Red Hat OpenStack Platform (RHOSP) on user-provisioned infrastructure.

Prerequisites

- Have on your machine
 - A single directory in which you can create files to help you with the removal process
 - Python 3

1.6.1. Downloading playbook dependencies

The Ansible playbooks that simplify the removal process on user-provisioned infrastructure require several Python modules. On the machine where you will run the process, add the modules' repositories and then download them.



NOTE

These instructions assume that you are using Red Hat Enterprise Linux 8.

Prerequisites

- Python 3 is installed on your machine

Procedure

1. On a command line, add the repositories:

```
$ sudo subscription-manager register # If not done already
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
$ sudo subscription-manager repos --disable=* # If not done already

$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.8-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. Install the modules:

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk
```

3. Ensure that the **python** command points to **python3**:

```
$ sudo alternatives --set python /usr/bin/python3
```

1.6.2. Removing a cluster on OpenStack that uses your own infrastructure

You can remove an OpenShift Container Platform cluster on Red Hat OpenStack Platform (RHOSP) that uses your own infrastructure. To complete the removal process quickly, create and run several Ansible playbooks.

Prerequisites

- Python 3 is installed on your machine
- You downloaded the modules in "Downloading playbook dependencies"



PROCEDURE

You may have the **common.yaml** and **inventory.yaml** playbooks left over from when you installed OpenShift Container Platform. If you do, you can skip the first two steps of the procedure.

1. Insert the following content into a local file called **common.yaml**:

```
- hosts: localhost
  gather_facts: no

vars_files:
- metadata.json

tasks:
- name: 'Compute resource names'
  set_fact:
    cluster_id_tag: "openshiftClusterID={{ infraID }}"
    os_network: "{{ infraID }}-network"
    os_subnet: "{{ infraID }}-nodes"
```

```

os_router: "{{ infraID }}-external-router"
# Port names
os_port_api: "{{ infraID }}-api-port"
os_port_ingress: "{{ infraID }}-ingress-port"
os_port_bootstrap: "{{ infraID }}-bootstrap-port"
os_port_master: "{{ infraID }}-master-port"
os_port_worker: "{{ infraID }}-worker-port"
# Security groups names
os_sg_master: "{{ infraID }}-master"
os_sg_worker: "{{ infraID }}-worker"
# Server names
os_bootstrap_server_name: "{{ infraID }}-bootstrap"
os_cp_server_name: "{{ infraID }}-master"
os_compute_server_name: "{{ infraID }}-worker"
# Trunk names
os_cp_trunk_name: "{{ infraID }}-master-trunk"
os_compute_trunk_name: "{{ infraID }}-worker-trunk"
# Subnet pool name
subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
# Service network name
os_svc_network: "{{ infraID }}-kuryr-service-network"
# Service subnet name
os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
# Ignition files
os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"

```

2. Insert the following content into a local file called **inventory.yaml**, and edit the values to match your own:

```

all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

# User-provided values
os_subnet_range: '10.0.0.0/16'
os_flavor_master: 'm1.xlarge'
os_flavor_worker: 'm1.large'
os_image_rhcos: 'rhcos'
os_external_network: 'external'
# OpenShift API floating IP address
os_api_fip: '203.0.113.23'
# OpenShift Ingress floating IP address
os_ingress_fip: '203.0.113.19'
# Service subnet cidr
svc_subnet_range: '172.30.0.0/16'
os_svc_network_range: '172.30.0.0/15'
# Subnet pool prefixes
cluster_network_cidrs: '10.128.0.0/14'
# Subnet pool prefix length
host_prefix: '23'
# Name of the SDN.
# Possible values are OpenshiftSDN or Kuryr.
os_networking_type: 'OpenshiftSDN'

```

```

# Number of provisioned Control Plane nodes
# 3 is the minimum number for a fully-functional cluster.
os_cp_nodes_number: 3

# Number of provisioned Compute nodes.
# 3 is the minimum number for a fully-functional cluster.
os_compute_nodes_number: 3

```

3. *Optional:* If your cluster uses Kuryr, insert the following content into a local file called **down-06_load-balancers.yaml**:

```

# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'Get a token for creating the server group'
      os_auth:
        register: cloud
        when: os_networking_type == "Kuryr"

    - name: 'List octavia versions'
      uri:
        method: GET
        headers:
          X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
          Content-Type: 'application/json'
        url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url') }}"
        register: octavia_versions
        when: os_networking_type == "Kuryr"

    - set_fact:
        versions: "{{ octavia_versions.json.versions | selectattr('id', 'match', 'v2.5') |
map(attribute='id') | list }}"
        when: os_networking_type == "Kuryr"

    - name: 'List tagged loadbalancers'
      uri:
        method: GET
        headers:
          X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
        url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url')
}}/v2.0/lbaas/loadbalancers?tags={{cluster_id_tag}}"
        when:
          - os_networking_type == "Kuryr"
          - versions | length > 0
        register: lbs_tagged

```



```

# NOTE: Kuryr creates an Octavia load balancer
# for each service present on the cluster. Let's make
# sure to remove the resources generated.
- name: 'Remove the cluster load balancers'
  os_loadbalancer:
    name: "{{ item.name }}"
    state: absent
    wait: no
  with_items: "{{ lbs_tagged.json.loadbalancers }}"
  when:
    - os_networking_type == "Kuryr"
    - versions | length > 0

- name: 'List loadbalancers tagged on description'
  uri:
    method: GET
    headers:
      X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
    url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url')
}}/v2.0/lbaas/loadbalancers?description={{cluster_id_tag}}"
  when:
    - os_networking_type == "Kuryr"
    - versions | length == 0
  register: lbs_description

# NOTE: Kuryr creates an Octavia load balancer
# for each service present on the cluster. Let's make
# sure to remove the resources generated.
- name: 'Remove the cluster load balancers'
  os_loadbalancer:
    name: "{{ item.name }}"
    state: absent
  with_items: "{{ lbs_description.json.loadbalancers }}"
  when:
    - os_networking_type == "Kuryr"
    - versions | length == 0

```

4. Insert the following content into a local file called **down-05_compute-nodes.yaml**:

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Remove the Compute servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    state: absent

```

```

    with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"

- name: 'List the Compute trunks'
  command:
    cmd: "openstack network trunk list -c Name -f value"
  when: os_networking_type == "Kuryr"
  register: trunks

- name: 'Remove the Compute trunks'
  command:
    cmd: "openstack network trunk delete {{ item.1 }}-{{ item.0 }}"
  when:
    - os_networking_type == "Kuryr"
    - (item.1|string + '-' + item.0|string) in trunks.stdout_lines|list
  with_indexed_items: "{{ [os_compute_trunk_name] * os_compute_nodes_number }}"

- name: 'Remove the Compute ports'
  os_port:
    name: "{{ item.1 }}-{{ item.0 }}"
    state: absent
  with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

```

5. Insert the following content into a local file called **down-04_control-plane.yaml**:

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Remove the Control Plane servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    state: absent
  with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"

- name: 'List the Compute trunks'
  command:
    cmd: "openstack network trunk list -c Name -f value"
  when: os_networking_type == "Kuryr"
  register: trunks

- name: 'Remove the Control Plane trunks'
  command:
    cmd: "openstack network trunk delete {{ item.1 }}-{{ item.0 }}"
  when:
    - os_networking_type == "Kuryr"
    - (item.1|string + '-' + item.0|string) in trunks.stdout_lines|list
  with_indexed_items: "{{ [os_cp_trunk_name] * os_cp_nodes_number }}"

```

```

- name: 'Remove the Control Plane ports'
  os_port:
    name: "{{ item.1 }}-{{ item.0 }}"
    state: absent
  with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"

```

6. Insert the following content into a local file called **down-03_bootstrap.yaml**:

```

# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'Remove the bootstrap server'
      os_server:
        name: "{{ os_bootstrap_server_name }}"
        state: absent
        delete_fip: yes

    - name: 'Remove the bootstrap server port'
      os_port:
        name: "{{ os_port_bootstrap }}"
        state: absent

```

7. Insert the following content into a local file called **down-02_network.yaml**:

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'List ports attached to router'
      command:
        cmd: "openstack port list --device-owner=network:router_interface --tags {{ cluster_id_tag }}"
        register: router_ports

    - name: 'Remove the ports from router'
      command:
        cmd: "openstack router remove port {{ os_router }} {{ item.1 }}"
        with_indexed_items: "{{ router_ports.stdout_lines }}"

```

```

- name: 'List ha ports attached to router'
  command:
    cmd: "openstack port list --device-owner=network:ha_router_replicated_interface --tags {{
cluster_id_tag }} -f value -c id"
  register: ha_router_ports

- name: 'Remove the ha ports from router'
  command:
    cmd: "openstack router remove port {{ os_router }} {{ item.1 }}"
    with_indexed_items: "{{ ha_router_ports.stdout_lines }}"

- name: 'List ports'
  command:
    cmd: "openstack port list --tags {{ cluster_id_tag }} -f value -c id "
  register: ports

- name: 'Remove the cluster ports'
  command:
    cmd: "openstack port delete {{ item.1 }}"
    with_indexed_items: "{{ ports.stdout_lines }}"

- name: 'Remove the cluster router'
  os_router:
    name: "{{ os_router }}"
    state: absent

- name: 'List cluster networks'
  command:
    cmd: "openstack network list --tags {{ cluster_id_tag }} -f value -c Name"
  register: networks

- name: 'Remove the cluster networks'
  os_network:
    name: "{{ item.1 }}"
    state: absent
  with_indexed_items: "{{ networks.stdout_lines }}"

- name: 'List the cluster subnet pool'
  command:
    cmd: "openstack subnet pool list --name {{ subnet_pool }}"
  when: os_networking_type == "Kuryr"
  register: pods_subnet_pool

- name: 'Remove the cluster subnet pool'
  command:
    cmd: "openstack subnet pool delete {{ subnet_pool }}"
  when:
    - os_networking_type == "Kuryr"
    - pods_subnet_pool.stdout != ""

```

8. Insert the following content into a local file called **down-01_security-groups.yaml**:

```

# Required Python packages:
#
# ansible
# openstackclient

```

```
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'List security groups'
      command:
        cmd: "openstack security group list --tags {{ cluster_id_tag }} -f value -c Name"
      register: security_groups

    - name: 'Remove the cluster security groups'
      command:
        cmd: "openstack security group delete {{ item.1 }}"
      with_indexed_items: "{{ security_groups.stdout_lines }}"
```

9. On a command line, run the playbooks you created:

```
$ ansible-playbook -i inventory.yaml \
  down-03_bootstrap.yaml \
  down-04_control-plane.yaml \
  down-05_compute-nodes.yaml \
  down-06_load-balancers.yaml \
  down-02_network.yaml \
  down-01_security-groups.yaml
```

10. Remove any DNS record changes you made for the OpenShift Container Platform installation.

OpenShift Container Platform is removed from your infrastructure.