OpenShift Container Platform 4.6

Installing on GCP

Installing OpenShift Container Platform GCP clusters

OpenShift Container Platform 4.6 Installing on GCP

Installing OpenShift Container Platform GCP clusters

Legal Notice

Abstract

This document provides instructions for installing and uninstalling OpenShift Container Platform clusters on Google Cloud Platform.

Table of Contents

CHAPTER 1. INSTALLING ON GCP

1.1. CONFIGURING A GCP PROJECT

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

1.1.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See Creating and Managing Projects in the GCP documentation.

> **IMPORTANT**
>
> Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>.<base_domain>** URL; the Premium Tier is required for internal load balancing.

1.1.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See Enabling services in the GCP documentation.

Table 1.1. Required API services

API service	Console service name
Compute Engine API	**compute.googleapis.com**
Google Cloud APIs	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**

API service	Console service name
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

1.1.3. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the same project that you host the OpenShift Container Platform cluster. This zone must be authoritative for the domain. The DNS service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through GCP or another source.

   > **NOTE**
   >
   > If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see Google Domains.

2. Create a public hosted zone for your domain or subdomain in your GCP project. See Creating public zones in the GCP documentation.
   Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.

3. Extract the new authoritative name servers from the hosted zone records. See Look up your Cloud DNS name servers in the GCP documentation.
   You typically have four name servers.

4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: How to switch to custom name servers .

5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See Migrating to Cloud DNS in the GCP documentation.

6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

1.1.4. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default Quotas do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 1.2. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall rules	Compute	Global	11	1
Forwarding rules	Compute	Global	2	0
In-use global IP addresses	Compute	Global	4	1
Health checks	Compute	Global	3	0
Images	Compute	Global	1	0
Networks	Compute	Global	2	0
Static IP addresses	Compute	Region	4	1
Routers	Compute	Global	1	0
Routes	Compute	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Compute	Global	3	0
CPUs	Compute	Region	28	4
Persistent disk SSD (GB)	Compute	Region	896	128

> **NOTE**
>
> If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**

- **asia-northeast2**

- **asia-south1**

- **australia-southeast1**

- **europe-north1**

- **europe-west2**

- **europe-west3**

- **europe-west6**

- **northamerica-northeast1**

- **southamerica-east1**

- **us-west2**

You can increase resource quotas from the GCP console, but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

1.1.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a service account in the project that you use to host your OpenShift Container Platform cluster. See Creating a service account in the GCP documentation.

2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See Granting roles to a service account for specific resources.

NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. Create the service account key in JSON format. See Creating service account keys in the GCP documentation.
The service account key is required to create a cluster.

1.1.5.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions. If you deploy your cluster into an existing VPC, the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin

- Security Admin

- Service Account Admin

- Service Account User

- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 1.3. GCP service account permissions

Account	Roles
Control Plane	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin

Account	Roles
	roles/iam.serviceAccountUser
Compute	**roles/compute.viewer**
	roles/storage.admin

1.1.6. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- **asia-east1** (Changhua County, Taiwan)

- **asia-east2** (Hong Kong)

- **asia-northeast1** (Tokyo, Japan)

- **asia-northeast2** (Osaka, Japan)

- **asia-northeast3** (Seoul, South Korea)

- **asia-south1** (Mumbai, India)

- **asia-southeast1** (Jurong West, Singapore)

- **asia-southeast2** (Jakarta, Indonesia)

- **australia-southeast1** (Sydney, Australia)

- **europe-north1** (Hamina, Finland)

- **europe-west1** (St. Ghislain, Belgium)

- **europe-west2** (London, England, UK)

- **europe-west3** (Frankfurt, Germany)

- **europe-west4** (Eemshaven, Netherlands)

- **europe-west6** (Zürich, Switzerland)

- **northamerica-northeast1** (Montréal, Québec, Canada)

- **southamerica-east1** (São Paulo, Brazil)

- **us-central1** (Council Bluffs, Iowa, USA)

- **us-east1** (Moncks Corner, South Carolina, USA)

- **us-east4** (Ashburn, Northern Virginia, USA)

- **us-west1** (The Dalles, Oregon, USA)

- **us-west2** (Los Angeles, California, USA)

- **us-west3** (Salt Lake City, Utah, USA)

- **us-west4** (Las Vegas, Nevada, USA)

1.1.7. Next steps

- Install an OpenShift Container Platform cluster on GCP. You can install a customized cluster or quickly install a cluster with default options.

1.2. MANUALLY CREATING IAM FOR GCP

In environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace, you can put the Cloud Credential Operator (CCO) into manual mode before you install the cluster.

1.2.1. Alternatives to storing administrator-level secrets in the kube-system project

The Cloud Credential Operator (CCO) manages cloud provider credentials as Kubernetes custom resource definitions (CRDs). You can configure the CCO to suit the security requirements of your organization by setting different values for the **credentialsMode** parameter in the **install-config.yaml** file.

If you prefer not to store an administrator-level credential secret in the cluster **kube-system** project, you can set the **credentialsMode** parameter for the CCO to **Manual** when installing OpenShift Container Platform and manage your cloud credentials manually.

Using manual mode allows each cluster component to have only the permissions it requires, without storing an administrator-level credential in the cluster. You can also use this mode if your environment does not have connectivity to the cloud provider public IAM endpoint. However, you must manually reconcile permissions with new release images for every upgrade. You must also manually supply credentials for every component that requests them.

Additional resources

- Rotating or removing cloud provider credentials.

For a detailed description of all available CCO credential modes and their supported platforms, see the Cloud Credential Operator reference.

1.2.2. Manually create IAM

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

1. To generate the manifests, run the following command from the directory that contains the installation program:

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

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

2. Insert a config map into the manifests directory so that the Cloud Credential Operator is placed in manual mode:

```
$ cat <<EOF > mycluster/manifests/cco-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cloud-credential-operator-config
  namespace: openshift-cloud-credential-operator
  annotations:
    release.openshift.io/create-only: "true"
data:
  disabled: "true"
EOF
```

3. Remove the **admin** credential secret created using your local cloud credentials. This removal prevents your **admin** credential from being stored in the cluster:

```
$ rm mycluster/openshift/99_cloud-creds-secret.yaml
```

4. From the directory that contains the installation program, obtain details of the OpenShift Container Platform release image that your **openshift-install** binary is built to use:

```
$ openshift-install version
```

Example output

```
release image quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64
```

5. Locate all **CredentialsRequest** objects in this release image that target the cloud you are deploying on:

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.y.z-x86_64 --credentials-requests --cloud=gcp
```

This displays the details for each request.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-gcs
  namespace: openshift-cloud-credential-operator
spec:
```

```
    secretRef:
      name: installer-cloud-credentials
      namespace: openshift-image-registry
    providerSpec:
      apiVersion: cloudcredential.openshift.io/v1
      kind: GCPProviderSpec
      predefinedRoles:
      - roles/storage.admin
      - roles/iam.serviceAccountUser
      skipServiceCheck: true
```

6. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **credentialsRequest**. The format for the secret data varies for each cloud provider.

7. From the directory that contains the installation program, proceed with your cluster creation:

```
$ openshift-install create cluster --dir <installation_directory>
```

> **IMPORTANT**
>
> Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state. For details, see the *Upgrading clusters with manually maintained credentials* section of the installation content for your cloud provider.

1.2.3. Admin credentials root secret format

Each cloud provider uses a credentials root secret in the **kube-system** namespace by convention, which is then used to satisfy all credentials requests and create their respective secrets. This is done either by minting new credentials, with *mint mode*, or by copying the credentials root secret, with *passthrough mode*.

The format for the secret varies by cloud, and is also used for each **CredentialsRequest** secret.

Google Cloud Platform (GCP) secret format

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <ServiceAccount>
```

1.2.4. Upgrading clusters with manually maintained credentials

If credentials are added in a future release, the Cloud Credential Operator (CCO) **upgradable** status for a cluster with manually maintained credentials changes to **false**. For minor release, for example, from 4.5 to 4.6, this status prevents you from upgrading until you have addressed any updated permissions. For z-stream releases, for example, from 4.5.10 to 4.5.11, the upgrade is not blocked, but the credentials must still be updated for the new release.

Use the **Administrator** perspective of the web console to determine if the CCO is upgradeable.

1. Navigate to **Administration → Cluster Settings**.

2. To view the CCO status details, click **cloud-credential** in the **Cluster Operators** list.

3. If the **Upgradeable** status in the **Conditions** section is **False**, examine the **credentialsRequests** for the new release and update the manually maintained credentials on your cluster to match before upgrading.

In addition to creating new credentials for the release image that you are upgrading to, you must review the required permissions for existing credentials and accommodate any new permissions requirements for existing components in the new release. The CCO cannot detect these mismatches and will not set **upgradable** to **false** in this case.

The *Manually creating IAM* section of the installation content for your cloud provider explains how to obtain and use the credentials required for your cloud.

1.2.5. Mint mode

Mint mode is the default and recommended Cloud Credential Operator (CCO) credentials mode for OpenShift Container Platform. In this mode, the CCO uses the provided administrator-level cloud credential to run the cluster. Mint mode is supported for AWS, GCP, and Azure.

In mint mode, the **admin** credential is stored in the **kube-system** namespace and then used by the CCO to process the **CredentialsRequest** objects in the cluster and create users for each with specific permissions.

The benefits of mint mode include:

- Each cluster component has only the permissions it requires

- Automatic, on-going reconciliation for cloud credentials, including additional credentials or permissions that might be required for upgrades

One drawback is that mint mode requires **admin** credential storage in a cluster **kube-system** secret.

1.2.6. Mint Mode with removal or rotation of the admin credential

Currently, this mode is only supported on AWS.

In this mode, a user installs OpenShift Container Platform with an **admin** credential just like the normal mint mode. However, this mode removes the **admin** credential secret from the cluster post-installation.

The administrator can have the Cloud Credential Operator make its own request for a read-only credential that allows it to verify if all **CredentialsRequest** objects have their required permissions, thus the **admin** credential is not required unless something needs to be changed. After the associated credential is removed, it can be destroyed on the underlying cloud, if desired.

Prior to upgrade, the **admin** credential should be restored. In the future, upgrade might be blocked if the credential is not present.

The **admin** credential is not stored in the cluster permanently.

This mode still requires the **admin** credential in the cluster for brief periods of time. It also requires manually re-instating the secret with **admin** credentials for each upgrade.

1.2.7. Next steps

- Install an OpenShift Container Platform cluster:

  - Installing a cluster quickly on GCP with default options on installer-provisioned infrastructure

  - Install a cluster with cloud customizations on installer-provisioned infrastructure

  - Install a cluster with network customizations on installer-provisioned infrastructure

1.3. INSTALLING A CLUSTER QUICKLY ON GCP

In OpenShift Container Platform version 4.6, you can install a cluster on Google Cloud Platform (GCP) that uses the default configuration options.

1.3.1. Prerequisites

- Review details about the OpenShift Container Platform installation and update processes.

- Configure a GCP account to host the cluster.

- If you use a firewall, you must configure it to allow the sites that your cluster requires access to.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can manually create and maintain IAM credentials. Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.3.2. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster.

You must have Internet access to:

- Access OpenShift Cluster Manager 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.

1.3.3. 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 the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

> **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 ed25519 -N '' \
       -f <path>/<file_name> 1
   ```

   1 Specify the path and file name, such as **~/.ssh/id_rsa**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your **~/.ssh** directory.

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

   > **NOTE**
   >
   > If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

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

   ```
   $ eval "$(ssh-agent -s)"
   ```

   Example output

   ```
   Agent pid 31874
   ```

   > **NOTE**
   >
   > If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

```
$ ssh-add <path>/<file_name>
```
1

Example output

```
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**

4. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

5. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

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

1.3.4. Obtaining the installation program

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

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the Infrastructure Provider page on the OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Select your infrastructure provider.

3. 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both files are required to delete the cluster.

> **IMPORTANT**
>
> Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

   ```
   $ tar xvf openshift-install-linux.tar.gz
   ```

5. Download your installation pull secret from the Red Hat OpenShift Cluster Manager . 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. 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

- Configure an account with the cloud platform that hosts your cluster.

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

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:

   - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCLOUD_KEYFILE_JSON** environment variables

   - The **~/.gcp/osServiceAccount.json** file

   - The **gcloud cli** default credentials

2. Change to the directory that contains the installation program and initialize the cluster deployment:

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

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

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.
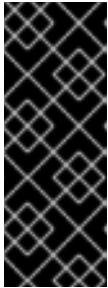
> **IMPORTANT**
>
> Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. 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.

Provide values at the prompts:

a. 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.

b. Select **gcp** as the platform to target.

c. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.

d. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.

e. Select the region to deploy the cluster to.

f. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.

g. Enter a descriptive name for your cluster. If you provide a name that is longer than 6 characters, only the first 6 characters will be used in the infrastructure ID that is generated from the cluster name.

h. Paste the pull secret from the Red Hat OpenShift Cluster Manager .

> **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.

Example output

```
...
```

INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
INFO Time elapsed: 36m22s

> **NOTE**
>
> The cluster access and credential information also outputs to **<installation_directory>/.openshift_install.log** when an installation succeeds.

> **IMPORTANT**
>
> - The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
>
> - It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

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

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.

   - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.

   - If you included the **Service Account Key Admin** role, you can remove it.

1.3.6. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

> **IMPORTANT**
>
> If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.3.6.1. Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Linux Client** entry and save the file.

4. Unpack the archive:

   ```
   $ tar xvzf <file>
   ```

5. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.3.6.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Windows Client** entry and save the file.

4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

   ```
   C:\> path
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

1.3.6.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 MacOSX Client** entry and save the file.

4. Unpack and unzip the archive.

5. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.3.7. Logging in to the cluster by using the CLI

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

- You deployed an OpenShift Container Platform cluster.

- You installed the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

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

   ❶ 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
   ```

   Example output

   ```
   system:admin
   ```

Additional resources

- See Accessing the web console for more details about accessing and understanding the OpenShift Container Platform web console.

1.3.8. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.3.9. Next steps

- Customize your cluster.

- If necessary, you can opt out of remote health reporting .

1.4. INSTALLING A CLUSTER ON GCP WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.6, you can install a customized cluster on infrastructure that the installation program provisions on Google Cloud Platform (GCP). To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

1.4.1. Prerequisites
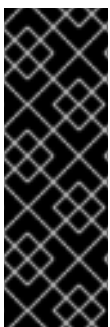
- Review details about the OpenShift Container Platform installation and update processes.

- Configure a GCP account to host the cluster.

- If you use a firewall, you must configure it to allow the sites that your cluster requires access to.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can manually create and maintain IAM credentials. Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.4.2. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster.

You must have Internet access to:

- Access OpenShift Cluster Manager 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.

1.4.3. 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 the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

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.
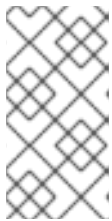
Procedure

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 ed25519 -N '' \
    -f <path>/<file_name> 1
```

1 Specify the path and file name, such as **~/.ssh/id_rsa**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your **~/.ssh** directory.

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

NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

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

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

> **NOTE**
>
> If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

```
$ ssh-add <path>/<file_name> ❶
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

❶ Specify the path and file name for your SSH private key, such as **~/.ssh/id_rsa**

4. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

5. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

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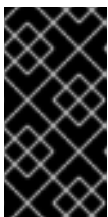
1.4.4. Obtaining the installation program

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

Prerequisites

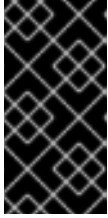- You have a computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the Infrastructure Provider page on the OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Select your infrastructure provider.

3. 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both files are required to delete the cluster.

> **IMPORTANT**
>
> Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Download your installation pull secret from the Red Hat OpenShift Cluster Manager . 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.

1.4.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

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

Procedure

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

   a. Change to the directory that contains the installation program and run the following command:

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

   ❶ 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 **gcp** as the platform to target.

iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.

iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.

v. Select the region to deploy the cluster to.

vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.

vii. Enter a descriptive name for your cluster.

viii. Paste the pull secret from the Red Hat OpenShift Cluster Manager .

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.

1.4.5.1. 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

After installation, you cannot modify these parameters in the **install-config.yaml** file.

IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

1.4.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 1.4. Required parameters

Parameter	Description	Values
apiVersion	The API version for the **install-config.yaml** content. The current version is **v1**. The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain 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**.
metadata	Kubernetes resource **ObjectMeta**, from which only the **name** parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of **{{.metadata.name}}.{{.baseDomain}}**.	String of lowercase letters, hyphens (**-**), and periods (**.**), such as **dev**.

Parameter	Description	Values
platform	The configuration for the specific platform upon which to perform the installation: **aws**, **baremetal**, **azure**, **openstack**, **ovirt**, **vsphere**. For additional information about **platform.<platform>** parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from the Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	```\n{\n "auths":{\n "cloud.openshift.com":{\n "auth":"b3Blb=",\n "email":"you@example.com"\n },\n "quay.io":{\n "auth":"b3Blb=",\n "email":"you@example.com"\n }\n }\n}\n```

1.4.5.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 1.5. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object **NOTE** You cannot modify parameters specified by the **networking** object after installation.

Parameter	Description	Values
networking.network Type	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either **OpenShiftSDN** or **OVNKubernetes**. The default value is **OpenShiftSDN**.

networking.clusterN etwork	The IP address blocks for pods.
	
	The default value is **10.128.0.0/14** with a host prefix of **/23**.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterN etwork.cidr	Required if you use **networking.clusterNetwork**. An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**.
networking.clusterN etwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23** then each node is assigned a /**23** subnet out of the given **cidr**. A **hostPrefix** value of **23** provides 510 (2^(32 – 23) – 2) pod IP addresses.	A subnet prefix. The default value is **23**.

networking.serviceN etwork	The IP address block for services. The default value is **172.30.0.0/16**.
	
	The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.
		An array with an IP address block in CIDR format. For example:
		
		``
		networking:
		  serviceNetwork:
		   - 172.30.0.0/16
		``

networking.machine Network	The IP address blocks for machines.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		``
		networking:
		  machineNetwork:
		  - cidr: 10.0.0.0/16
		``

Parameter	Description	Values
networking.machine Network.cidr	Required if you use **networking.machineNetwork**. An IP address block. The default value is **10.0.0.0/16** for all platforms other than libvirt. For libvirt, the default value is **192.168.126.0/24**.	An IP network block in CIDR notation. For example, **10.0.0.0/16**. **NOTE** Set the **networking.machin eNetwork** to match the CIDR that the preferred NIC resides in.

1.4.5.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 1.6. Optional parameters

Parameter	Description	Values
additionalTrustBund le	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architectur e	Determines the instruction set architecture of the machines in the pool. Currently, heteregeneous clusters are not supported, so all pools must specify the same architecture. Valid values are **amd64** (the default).	String

Parameter	Description	Values
compute.hyperthreading	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. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
compute.name	Required if you use **compute**. The name of the machine pool.	**worker**
compute.platform	Required if you use **compute**. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the **controlPlane.platform** parameter value.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere**, or **{}**
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	The configuration for the machines that comprise the control plane.	Array of **MachinePool** objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are **amd64** (the default).	String

Parameter	Description	Values
controlPlane.hypert hreading	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. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
controlPlane.name	Required if you use **controlPlane**. The name of the machine pool.	**master**
controlPlane.platfor m	Required if you use **controlPlane**. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the **compute.platform** parameter value.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere**, or **{}**
controlPlane.replica s	The number of control plane machines to provision.	The only supported value is **3**, which is the default value.

Parameter	Description	Values
credentialsMode	The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported. **NOTE** Not all CCO modes are supported for all cloud providers. For more information on CCO modes, see the *Cloud Credential Operator* entry in the *Red Hat Operators reference* content.	**Mint**, **Passthrough**, **Manual**, or an empty string (**""**).

Parameter	Description	Values
fips	Enable or disable FIPS mode. The default is **false** (disabled). 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. **IMPORTANT** The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture. **NOTE** If you are using Azure File storage, you cannot enable FIPS mode.	**false** or **true**
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a **source** and, optionally, **mirrors**, as described in the following rows of this table.
imageContentSources.source	Required if you use **imageContentSources**. Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	**Internal** or **External**. To deploy a private cluster, which cannot be accessed from the internet, set **publish** to **Internal**. The default value is **External**.

Parameter	Description	Values
sshKey	The SSH key or keys to authenticate 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.	One or more keys. For example: sshKey: <key1> <key2> <key3>

1.4.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 1.7. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.region	The name of the GCP region that hosts your cluster.	Any valid region name, such as **us-central1**.
platform.gcp.type	The GCP machine type.	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones, such as **us-central1-a**, in a YAML sequence.
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.

Parameter	Description	Values
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.
platform.gcp.licenses	A list of license URLs that must be applied to the compute images. IMPORTANT The **licenses** parameter is a deprecated field and nested virtualization is enabled by default. It is not recommended to use this field.	Any license available with the license API, such as the license to enable nested virtualization. You cannot use this parameter with a mechanism that generates pre-built images. Using a license URL forces the installer to copy the source image before use.
platform.gcp.osDisk.diskSizeGB	The size of the disk in gigabytes (GB).	Any size between 16 GB and 65536 GB.
platform.gcp.osDisk.diskType	The type of disk.	Either the default **pd-ssd** or the **pd-standard** disk type. The control plane nodes must be the **pd-ssd** disk type. The worker nodes can be either type.

1.4.5.2. Sample customized install-config.yaml file for GCP

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

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

```
apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
```

```
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
    replicas: 3
compute: 5 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
    replicas: 3
metadata:
  name: test-cluster 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
pullSecret: '{"auths": ...}' 11
fips: false 12
sshKey: ssh-ed25519 AAAA... 13
```

1 8 9 10 11 Required. The installation program prompts you for this value.

2 5 If you do not provide these parameters and values, the installation program provides the default value.

3 6 The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Only one control plane pool is used.

4 7 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.

> **IMPORTANT**
>
> If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

12 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.

> **IMPORTANT**
>
> The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture.

13 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

> **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.

1.4.5.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

> **NOTE**
>
> The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.
>
> For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace to hold the additional CA certificates. If you provide **additionalTrustBundle** and at least one proxy setting, the **Proxy** object is configured to reference the **user-ca-bundle** config map in the **trustedCA** field. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges the contents specified for the **trustedCA** parameter with the RHCOS trust bundle. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

> **NOTE**
>
> The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

> **NOTE**
>
> Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

1.4.6. 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

- Configure an account with the cloud platform that hosts your cluster.

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

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:

   - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCLOUD_KEYFILE_JSON** environment variables

   - The ~/**.gcp/osServiceAccount.json** file

   - The **gcloud cli** default credentials

2. Change to the directory that contains the installation program and initialize the cluster deployment:

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

   ❶ For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

   ❷ 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.

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
```

> INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-Wt5AL"
> INFO Time elapsed: 36m22s

> **NOTE**
>
> The cluster access and credential information also outputs to
> **<installation_directory>/.openshift_install.log** when an installation succeeds.

> **IMPORTANT**
>
> - The Ignition config files that the installation program generates contain
>   certificates that expire after 24 hours, which are then renewed at that time. If
>   the cluster is shut down before renewing the certificates and the cluster is
>   later restarted after the 24 hours have elapsed, the cluster automatically
>   recovers the expired certificates. The exception is that you must manually
>   approve the pending **node-bootstrapper** certificate signing requests (CSRs)
>   to recover kubelet certificates. See the documentation for *Recovering from
>   expired control plane certificates* for more information.
>
> - It is recommended that you use Ignition config files within 12 hours after they
>   are generated because the 24-hour certificate rotates from 16 to 22 hours
>   after the cluster is installed. By using the Ignition config files within 12 hours,
>   you can avoid installation failure if the certificate update runs during
>   installation.

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

3. Optional: You can reduce the number of permissions for the service account that you used to
   install the cluster.

   - If you assigned the **Owner** role to your service account, you can remove that role and
     replace it with the **Viewer** role.

   - If you included the **Service Account Key Admin** role, you can remove it.

1.4.7. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a
command-line interface. You can install **oc** on Linux, Windows, or macOS.
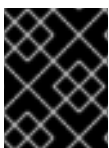
> **IMPORTANT**
>
> If you installed an earlier version of **oc**, you cannot use it to complete all of the commands
> in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.4.7.1. Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Linux Client** entry and save the file.

4. Unpack the archive:

   ```
   $ tar xvzf <file>
   ```

5. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.4.7.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Windows Client** entry and save the file.

4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

   ```
   C:\> path
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

1.4.7.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 MacOSX Client** entry and save the file.

4. Unpack and unzip the archive.

5. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.4.8. Logging in to the cluster by using the CLI

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

- You deployed an OpenShift Container Platform cluster.

- You installed the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

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

   ❶ 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
   ```

   Example output

   ```
   system:admin
   ```

Additional resources

- See Accessing the web console for more details about accessing and understanding the OpenShift Container Platform web console.

1.4.9. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.4.10. Next steps

- Customize your cluster.

- If necessary, you can opt out of remote health reporting .

1.5. INSTALLING A CLUSTER ON GCP WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.6, you can install a cluster with a customized network configuration on infrastructure that the installation program provisions on Google Cloud Platform (GCP). By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.

1.5.1. Prerequisites

- Review details about the OpenShift Container Platform installation and update processes.

- Configure a GCP account to host the cluster.

- If you use a firewall, you must configure it to allow the sites that your cluster requires access to.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can manually create and maintain IAM credentials . Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.5.2. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster.

You must have Internet access to:

- Access OpenShift Cluster Manager 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.

1.5.3. 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 the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

> **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 ed25519 -N '' \
       -f <path>/<file_name> 1
   ```

   1 Specify the path and file name, such as ~/**.ssh**/**id_rsa**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your ~/**.ssh** directory.

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

   > **NOTE**
   >
   > If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

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

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

> **NOTE**
>
> If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

```
$ ssh-add <path>/<file_name>  ❶
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

❶ Specify the path and file name for your SSH private key, such as ~/**.ssh**/**id_rsa**

4. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

5. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

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

1.5.4. Obtaining the installation program

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

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the Infrastructure Provider page on the OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Select your infrastructure provider.

3. 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both files are required to delete the cluster.

> **IMPORTANT**
>
> Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Download your installation pull secret from the Red Hat OpenShift Cluster Manager . 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.

1.5.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

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

Procedure

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

   a. Change to the directory that contains the installation program and run the following command:

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

   ❶ 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 **gcp** as the platform to target.

iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.

iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.

v. Select the region to deploy the cluster to.

vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.

vii. Enter a descriptive name for your cluster.

viii. Paste the pull secret from the Red Hat OpenShift Cluster Manager .

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.

1.5.5.1. 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

After installation, you cannot modify these parameters in the **install-config.yaml** file.

IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

1.5.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 1.8. Required parameters

Parameter	Description	Values
apiVersion	The API version for the **install-config.yaml** content. The current version is **v1**. The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain 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**.
metadata	Kubernetes resource **ObjectMeta**, from which only the **name** parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of **{{.metadata.name}}.{{.baseDomain}}**.	String of lowercase letters, hyphens (**-**), and periods (**.**), such as **dev**.

Parameter	Description	Values
platform	The configuration for the specific platform upon which to perform the installation: **aws**, **baremetal**, **azure**, **openstack**, **ovirt**, **vsphere**. For additional information about **platform.<platform>** parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from the Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	```{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }```

1.5.5.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 1.9. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object NOTE You cannot modify parameters specified by the **networking** object after installation.

Parameter	Description	Values
networking.network Type	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either **OpenShiftSDN** or **OVNKubernetes**. The default value is **OpenShiftSDN**.

networking.clusterN etwork	The IP address blocks for pods.
	
	The default value is **10.128.0.0/14** with a host prefix of **/23**.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterN etwork.cidr	Required if you use **networking.clusterNetwork**. An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**.
networking.clusterN etwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23** then each node is assigned a /**23** subnet out of the given **cidr**. A **hostPrefix** value of **23** provides 510 (2^(32 – 23) – 2) pod IP addresses.	A subnet prefix. The default value is **23**.

networking.serviceN etwork	The IP address block for services. The default value is **172.30.0.0/16**.
	
	The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.
		An array with an IP address block in CIDR format. For example:
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machine Network	The IP address blocks for machines.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

Parameter	Description	Values
networking.machine Network.cidr	Required if you use **networking.machineNetwork**. An IP address block. The default value is **10.0.0.0/16** for all platforms other than libvirt. For libvirt, the default value is **192.168.126.0/24**.	An IP network block in CIDR notation. For example, **10.0.0.0/16**. **NOTE** Set the **networking.machin eNetwork** to match the CIDR that the preferred NIC resides in.

1.5.5.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 1.10. Optional parameters

Parameter	Description	Values
additionalTrustBund le	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architectur e	Determines the instruction set architecture of the machines in the pool. Currently, heteregeneous clusters are not supported, so all pools must specify the same architecture. Valid values are **amd64** (the default).	String

Parameter	Description	Values
compute.hyperthreading	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. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
compute.name	Required if you use **compute**. The name of the machine pool.	**worker**
compute.platform	Required if you use **compute**. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the **controlPlane.platform** parameter value.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere**, or **{}**
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	The configuration for the machines that comprise the control plane.	Array of **MachinePool** objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are **amd64** (the default).	String

Parameter	Description	Values
controlPlane.hypert hreading	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. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
controlPlane.name	Required if you use **controlPlane**. The name of the machine pool.	**master**
controlPlane.platfor m	Required if you use **controlPlane**. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the **compute.platform** parameter value.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere**, or **{}**
controlPlane.replica s	The number of control plane machines to provision.	The only supported value is **3**, which is the default value.

Parameter	Description	Values
credentialsMode	The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported. **NOTE** Not all CCO modes are supported for all cloud providers. For more information on CCO modes, see the *Cloud Credential Operator* entry in the *Red Hat Operators reference* content.	**Mint**, **Passthrough**, **Manual**, or an empty string (**""**).

Parameter	Description	Values
fips	Enable or disable FIPS mode. The default is **false** (disabled). 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. **IMPORTANT** The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture. **NOTE** If you are using Azure File storage, you cannot enable FIPS mode.	**false** or **true**
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a **source** and, optionally, **mirrors**, as described in the following rows of this table.
imageContentSources.source	Required if you use **imageContentSources**. Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	**Internal** or **External**. To deploy a private cluster, which cannot be accessed from the internet, set **publish** to **Internal**. The default value is **External**.

Parameter	Description	Values
sshKey	The SSH key or keys to authenticate 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.	One or more keys. For example: sshKey: <key1> <key2> <key3>

1.5.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 1.11. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.region	The name of the GCP region that hosts your cluster.	Any valid region name, such as **us-central1**.
platform.gcp.type	The [GCP machine type](#).	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid [GCP availability zones](#), such as **us-central1-a**, in a [YAML sequence](#).
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.

Parameter	Description	Values
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.
platform.gcp.licenses	A list of license URLs that must be applied to the compute images. **IMPORTANT** The **licenses** parameter is a deprecated field and nested virtualization is enabled by default. It is not recommended to use this field.	Any license available with the license API, such as the license to enable nested virtualization. You cannot use this parameter with a mechanism that generates pre-built images. Using a license URL forces the installer to copy the source image before use.
platform.gcp.osDisk.diskSizeGB	The size of the disk in gigabytes (GB).	Any size between 16 GB and 65536 GB.
platform.gcp.osDisk.diskType	The type of disk.	Either the default **pd-ssd** or the **pd-standard** disk type. The control plane nodes must be the **pd-ssd** disk type. The worker nodes can be either type.

1.5.5.2. Sample customized install-config.yaml file for GCP

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

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

```
apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
```

```
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
  replicas: 3
compute: 5 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
  replicas: 3
metadata:
  name: test-cluster 8
networking: 9
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 10
    region: us-central1 11
pullSecret: '{"auths": ...}' 12
fips: false 13
sshKey: ssh-ed25519 AAAA... 14
```

1 **8** **10** **11** **12** Required. The installation program prompts you for this value.

2 **5** **9** If you do not provide these parameters and values, the installation program provides the default value.

3 **6** The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Only one control plane pool is used.

4 **7** Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.

> **IMPORTANT**
>
> If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

13 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.

> **IMPORTANT**
>
> The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture.

14 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

> **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.
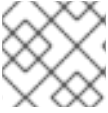
1.5.5.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

> **NOTE**
>
> The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.
>
> For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace to hold the additional CA certificates. If you provide **additionalTrustBundle** and at least one proxy setting, the **Proxy** object is configured to reference the **user-ca-bundle** config map in the **trustedCA** field. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges the contents specified for the **trustedCA** parameter with the RHCOS trust bundle. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

> **NOTE**
>
> The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

> **NOTE**
>
> Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

1.5.6. Network configuration phases

When specifying a cluster configuration prior to installation, there are several phases in the installation procedures when you can modify the network configuration:

Phase 1

After entering the **openshift-install create install-config** command. In the **install-config.yaml** file, you can customize the following network-related fields:

- **networking.networkType**

- **networking.clusterNetwork**

- **networking.serviceNetwork**

- **networking.machineNetwork**
  For more information on these fields, refer to "Installation configuration parameters".

> **NOTE**
>
> Set the **networking.machineNetwork** to match the CIDR that the preferred NIC resides in.

Phase 2

After entering the **openshift-install create manifests** command. If you must specify advanced network configuration, during this phase you can define a customized Cluster Network Operator manifest with only the fields you want to modify.

You cannot override the values specified in phase 1 in the **install-config.yaml** file during phase 2. However, you can further customize the cluster network provider during phase 2.

1.5.7. Specifying advanced network configuration

You can use advanced configuration customization to integrate your cluster into your existing network environment by specifying additional configuration for your cluster network provider. You can specify advanced network configuration only before you install the cluster.
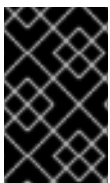
> **IMPORTANT**
>
> Modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- Create the **install-config.yaml** file and complete any modifications to it.

Procedure

1. Change to the directory that contains the installation program and create the manifests:

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

   where:

   <installation_directory>

Specifies the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a stub manifest file for the advanced network configuration that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
EOF
```

where:

<installation_directory>

Specifies the directory name that contains the **manifests/** directory for your cluster.

3. Open the **cluster-network-03-config.yml** file in an editor and specify the advanced network configuration for your cluster, such as in the following example:

Specify a different VXLAN port for the OpenShift SDN network provider

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

4. Save the **cluster-network-03-config.yml** file and quit the text editor.

5. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.

1.5.8. Cluster Network Operator configuration

The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a custom resource (CR) object that is named **cluster**. The CR specifies the fields for the **Network** API in the **operator.openshift.io** API group.

The CNO configuration inherits the following fields during cluster installation from the **Network** API in the **Network.config.openshift.io** API group and these fields cannot be changed:

clusterNetwork

IP address pools from which pod IP addresses are allocated.

serviceNetwork

IP address pool for services.

defaultNetwork.type

Cluster network provider, such as OpenShift SDN or OVN-Kubernetes.

You can specify the cluster network provider configuration for your cluster by setting the fields for the **defaultNetwork** object in the CNO object named **cluster**.

1.5.8.1. Cluster Network Operator configuration object

The fields for the Cluster Network Operator (CNO) are described in the following table:

Table 1.12. Cluster Network Operator configuration object

Field	Type	Description
metadata.name	**string**	The name of the CNO object. This name is always **cluster**.

spec.clusterNetwork	**array**	A list specifying the blocks of IP addresses from which pod IP addresses are allocated and the subnet prefix length assigned to each individual node in the cluster. For example:
		
		``
		spec:
		  clusterNetwork:
		  - cidr: 10.128.0.0/19
		    hostPrefix: 23
		  - cidr: 10.128.32.0/19
		    hostPrefix: 23
		``
		
		This value is ready-only and specified in the **install-config.yaml** file.

spec.serviceNetwork	**array**	A block of IP addresses for services. The OpenShift SDN and OVN-Kubernetes Container Network Interface (CNI) network providers support only a single IP address block for the service network. For example:
		
		``
		spec:
		  serviceNetwork:
		  - 172.30.0.0/14
		``
		
		This value is ready-only and specified in the **install-config.yaml** file.

spec.defaultNetwork	**object**	Configures the Container Network Interface (CNI) cluster network provider for the cluster network.
spec.kubeProxyConfig	**object**	The fields for this object specify the kube-proxy configuration. If you are using the OVN-Kubernetes cluster network provider, the kube-proxy configuration has no effect.

defaultNetwork object configuration
The values for the **defaultNetwork** object are defined in the following table:

Table 1.13. **defaultNetwork** object

Field	Type	Description
type	**string**	Either **OpenShiftSDN** or **OVNKubernetes**. The cluster network provider is selected during installation. This value cannot be changed after cluster installation. **NOTE** OpenShift Container Platform uses the OpenShift SDN Container Network Interface (CNI) cluster network provider by default.
openshiftSDNConfig	**object**	This object is only valid for the OpenShift SDN cluster network provider.
ovnKubernetesConfig	**object**	This object is only valid for the OVN-Kubernetes cluster network provider.

Configuration for the OpenShift SDN CNI cluster network provider

The following table describes the configuration fields for the OpenShift SDN Container Network Interface (CNI) cluster network provider.

Table 1.14. **openshiftSDNConfig** object

Field	Type	Description
mode	**string**	Configures the network isolation mode for OpenShift SDN. The default value is **NetworkPolicy**. The values **Multitenant** and **Subnet** are available for backwards compatibility with OpenShift Container Platform 3.x but are not recommended. This value cannot be changed after cluster installation.

Field	Type	Description
mtu	**integer**	The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU. If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes. If your cluster requires different MTU values for different nodes, you must set this value to **50** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1450**. This value cannot be changed after cluster installation.
vxlanPort	**integer**	The port to use for all VXLAN packets. The default value is **4789**. This value cannot be changed after cluster installation. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network, then you might be required to change this. For example, when running an OpenShift SDN overlay on top of VMware NSX-T, you must select an alternate port for the VXLAN, because both SDNs use the same default VXLAN port number. On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

Example OpenShift SDN configuration

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

Configuration for the OVN-Kubernetes CNI cluster network provider

The following table describes the configuration fields for the OVN-Kubernetes CNI cluster network provider.

Table 1.15. ovnKubernetesConfig object

Field	Type	Description

Field	Type	Description
mtu	**integer**	The maximum transmission unit (MTU) for the Geneve (Generic Network Virtualization Encapsulation) overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU. If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes. If your cluster requires different MTU values for different nodes, you must set this value to **100** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1400**. This value cannot be changed after cluster installation.
genevePort	**integer**	The port to use for all Geneve packets. The default value is **6081**. This value cannot be changed after cluster installation.

Example OVN-Kubernetes configuration

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

kubeProxyConfig object configuration
The values for the **kubeProxyConfig** object are defined in the following table:

Table 1.16. **kubeProxyConfig** object

Field	Type	Description

Field	Type	Description
iptablesSyncPeriod	**string**	The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the Go **time** package documentation. **NOTE** Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

proxyArguments.iptables-min-sync-period	**array**	The minimum duration before refreshing **iptables** rules. This field ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the Go **time** package. The default value is:
		
		kubeProxyConfig:
		  proxyArguments:
		    iptables-min-sync-period:
		    - 0s

1.5.9. 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

- Configure an account with the cloud platform that hosts your cluster.

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

Procedure

1. Change to the directory that contains the installation program and initialize the cluster deployment:

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

1 For **<installation_directory>**, specify the

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.

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

> **NOTE**
>
> The cluster access and credential information also outputs to **<installation_directory>/.openshift_install.log** when an installation succeeds.

> **IMPORTANT**
>
> - The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
>
> - It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

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

1.5.10. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

> **IMPORTANT**
>
> If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.5.10.1. Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Linux Client** entry and save the file.

4. Unpack the archive:

   ```
   $ tar xvzf <file>
   ```

5. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.5.10.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Windows Client** entry and save the file.

4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

1.5.10.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 MacOSX Client** entry and save the file.

4. Unpack and unzip the archive.

5. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.5.11. Logging in to the cluster by using the CLI

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

- You deployed an OpenShift Container Platform cluster.

- You installed 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
```

Example output

```
system:admin
```

Additional resources

- See Accessing the web console for more details about accessing and understanding the OpenShift Container Platform web console.

1.5.12. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.5.13. Next steps

- Customize your cluster.

- If necessary, you can opt out of remote health reporting .

1.6. INSTALLING A CLUSTER ON GCP INTO AN EXISTING VPC

In OpenShift Container Platform version 4.6, you can install a cluster into an existing Virtual Private Cloud (VPC) on Google Cloud Platform (GCP). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

1.6.1. Prerequisites

- Review details about the OpenShift Container Platform installation and update processes.

- Configure a GCP account to host the cluster.

- If you use a firewall, you must configure it to allow the sites that your cluster requires access to.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can manually create and maintain IAM credentials . Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.6.2. About using a custom VPC

In OpenShift Container Platform 4.6, you can deploy a cluster into existing subnets in an existing Virtual Private Cloud (VPC) in Google Cloud Platform (GCP). By deploying OpenShift Container Platform into an existing GCP VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option. You must configure networking for the subnets.

1.6.2.1. Requirements for using your VPC

The union of the VPC CIDR block and the machine network CIDR must be non-empty. The subnets must be within the machine network.

The installation program does not create the following components:

- NAT gateways

- Subnets

- Route tables

- VPC network

> **NOTE**
>
> The installation program requires that you use the cloud-provided DNS server. Using a custom DNS server is not supported and causes the installation to fail.

1.6.2.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.

- You provide one subnet for control-plane machines and one subnet for compute machines.

- The subnet's CIDRs belong to the machine CIDR that you specified.

1.6.2.3. Division of permissions

Some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

1.6.2.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.

- ICMP ingress is allowed to the entire network.

- TCP 22 ingress (SSH) is allowed to the entire network.

- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.

- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

1.6.3. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster.

You must have Internet access to:

- Access OpenShift Cluster Manager 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.

1.6.4. 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 the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

> **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 ed25519 -N '' \
       -f <path>/<file_name> 1
   ```

1 Specify the path and file name, such as ~/**.ssh**/**id_rsa**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your ~/**.ssh** directory.

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

> **NOTE**
>
> If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

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

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

> **NOTE**
>
> If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

```
$ ssh-add <path>/<file_name>  1
```

Example output

```
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**

4. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

5. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

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

1.6.5. Obtaining the installation program

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

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the Infrastructure Provider page on the OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Select your infrastructure provider.

3. 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both files are required to delete the cluster.

   > **IMPORTANT**
   >
   > Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

   ```
   $ tar xvf openshift-install-linux.tar.gz
   ```

5. Download your installation pull secret from the Red Hat OpenShift Cluster Manager . 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.

1.6.6. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

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

Procedure

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

   a. Change to the directory that contains the installation program and 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 **gcp** as the platform to target.

      iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.

      iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.

      v. Select the region to deploy the cluster to.

      vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.

      vii. Enter a descriptive name for your cluster.

      viii. Paste the pull secret from the Red Hat OpenShift Cluster Manager .

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.

1.6.6.1. 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

After installation, you cannot modify these parameters in the **install-config.yaml** file.

IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

1.6.6.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 1.17. Required parameters

Parameter	Description	Values
apiVersion	The API version for the **install-config.yaml** content. The current version is **v1**. The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain 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
metadata	Kubernetes resource **ObjectMeta**, from which only the **name** parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of **{{.metadata.name}}.{{.baseDomain}}**.	String of lowercase letters, hyphens (**-**), and periods (**.**), such as **dev**.
platform	The configuration for the specific platform upon which to perform the installation: **aws**, **baremetal**, **azure**, **openstack**, **ovirt**, **vsphere**. For additional information about **platform.<platform>** parameters, consult the following table for your specific platform.	Object

pullSecret	Get a pull secret from the Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	`{`
		`  "auths":{`
		`    "cloud.openshift.com":{`
		`      "auth":"b3Blb=",`
		`      "email":"you@example.com"`
		`    },`
		`    "quay.io":{`
		`      "auth":"b3Blb=",`
		`      "email":"you@example.com"`
		`    }`
		`  }`
		`}`

1.6.6.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 1.18. Network parameters

Parameter	Description	Values

Parameter	Description	Values
networking	The configuration for the cluster network.	Object **NOTE** You cannot modify parameters specified by the **networking** object after installation.
networking.network Type	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either **OpenShiftSDN** or **OVNKubernetes**. The default value is **OpenShiftSDN**.

networking.clusterN etwork	The IP address blocks for pods.
	
	The default value is **10.128.0.0/14** with a host prefix of **/23**.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23

networking.clusterN etwork.cidr	Required if you use **networking.clusterNetwork**. An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**.
networking.clusterN etwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23** then each node is assigned a /**23** subnet out of the given **cidr**. A **hostPrefix** value of **23** provides 510 (2^(32 – 23) – 2) pod IP addresses.	A subnet prefix. The default value is **23**.

networking.serviceN etwork	The IP address block for services. The default value is **172.30.0.0/16**.
	
	The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.
		An array with an IP address block in CIDR format. For example:
		
		networking:
		  serviceNetwork:
		  - 172.30.0.0/16

networking.machine Network	The IP address blocks for machines.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		networking:
		  machineNetwork:
		  - cidr: 10.0.0.0/16

Parameter	Description	Values
networking.machine Network.cidr	Required if you use **networking.machineNetwork**. An IP address block. The default value is **10.0.0.0/16** for all platforms other than libvirt. For libvirt, the default value is **192.168.126.0/24**.	An IP network block in CIDR notation. For example, **10.0.0.0/16**. **NOTE** Set the **networking.machin eNetwork** to match the CIDR that the preferred NIC resides in.

1.6.6.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 1.19. Optional parameters

Parameter	Description	Values
additionalTrustBund le	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architectur e	Determines the instruction set architecture of the machines in the pool. Currently, heteregeneous clusters are not supported, so all pools must specify the same architecture. Valid values are **amd64** (the default).	String

Parameter	Description	Values
compute.hyperthreading	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. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
compute.name	Required if you use **compute**. The name of the machine pool.	**worker**
compute.platform	Required if you use **compute**. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the **controlPlane.platform** parameter value.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere**, or **{}**
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	The configuration for the machines that comprise the control plane.	Array of **MachinePool** objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are **amd64** (the default).	String

Parameter	Description	Values
controlPlane.hyperthreading	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. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
controlPlane.name	Required if you use **controlPlane**. The name of the machine pool.	**master**
controlPlane.platform	Required if you use **controlPlane**. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the **compute.platform** parameter value.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere**, or **{}**
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is **3**, which is the default value.
credentialsMode	The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported. **NOTE** Not all CCO modes are supported for all cloud providers. For more information on CCO modes, see the *Cloud Credential Operator* entry in the *Red Hat Operators reference* content.	**Mint**, **Passthrough**, **Manual**, or an empty string (**""**).

Parameter	Description	Values
fips	Enable or disable FIPS mode. The default is **false** (disabled). 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. **IMPORTANT** The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture. **NOTE** If you are using Azure File storage, you cannot enable FIPS mode.	**false** or **true**
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a **source** and, optionally, **mirrors**, as described in the following rows of this table.
imageContentSources.source	Required if you use **imageContentSources**. Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	**Internal** or **External**. To deploy a private cluster, which cannot be accessed from the internet, set **publish** to **Internal**. The default value is **External**.

Parameter	Description	Values

sshKey	The SSH key or keys to authenticate 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.
		One or more keys. For example:
		
		sshKey:
		  <key1>
		  <key2>
		  <key3>

1.6.6.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 1.20. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.region	The name of the GCP region that hosts your cluster.	Any valid region name, such as **us-central1**.
platform.gcp.type	The [GCP machine type].	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid [GCP availability zones], such as **us-central1-a**, in a [YAML sequence].
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.

Param eter	Description	Values
platfor m.gcp .comp uteSu bnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.
platfor m.gcp .licens es	A list of license URLs that must be applied to the compute images. **IMPORTANT** The **licenses** parameter is a deprecated field and nested virtualization is enabled by default. It is not recommended to use this field.	Any license available with the license API, such as the license to enable nested virtualization. You cannot use this parameter with a mechanism that generates pre-built images. Using a license URL forces the installer to copy the source image before use.
platfor m.gcp .osDis k.disk SizeG B	The size of the disk in gigabytes (GB).	Any size between 16 GB and 65536 GB.
platfor m.gcp .osDis k.disk Type	The type of disk.	Either the default **pd-ssd** or the **pd-standard** disk type. The control plane nodes must be the **pd-ssd** disk type. The worker nodes can be either type.

1.6.6.2. Sample customized install-config.yaml file for GCP

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

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

```
apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
```

```
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
    replicas: 3
compute: 5 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
    replicas: 3
metadata:
  name: test-cluster 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
    network: existing_vpc 11
    controlPlaneSubnet: control_plane_subnet 12
    computeSubnet: compute_subnet 13
pullSecret: '{"auths": ...}' 14
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
```

1 8 9 10 14 Required. The installation program prompts you for this value.

2 5 If you do not provide these parameters and values, the installation program provides the default value.

3 6 The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Only one control plane pool is used.

4 7 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default,

> **IMPORTANT**
>
> If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

11 Specify the name of an existing VPC.

12 Specify the name of the existing subnet to deploy the control plane machines to. The subnet must belong to the VPC that you specified.

13 Specify the name of the existing subnet to deploy the compute machines to. The subnet must belong to the VPC that you specified.

15 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.

> **IMPORTANT**
>
> The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture.

16 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

> **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.

1.6.6.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

   ```
   apiVersion: v1
   baseDomain: my.domain.com
   proxy:
     httpProxy: http://<username>:<pswd>@<ip>:<port> 1
     httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
     noProxy: example.com 3
   additionalTrustBundle: | 4
       -----BEGIN CERTIFICATE-----
       <MY_TRUSTED_CA_CERT>
       -----END CERTIFICATE-----
   ...
   ```

   [1] A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

   [2] A proxy URL to use for creating HTTPS connections outside the cluster.

   [3] A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations.

   [4] If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace to hold the additional CA certificates. If you provide **additionalTrustBundle** and at least one proxy setting, the **Proxy** object is configured to reference the **user-ca-bundle** config map in the **trustedCA** field. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges the contents specified for the **trustedCA** parameter with the RHCOS trust bundle. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

   NOTE

   The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

> **NOTE**
>
> Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

1.6.7. 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

- Configure an account with the cloud platform that hosts your cluster.

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

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:

   - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCLOUD_KEYFILE_JSON** environment variables

   - The **~/.gcp/osServiceAccount.json** file

   - The **gcloud cli** default credentials

2. Change to the directory that contains the installation program and initialize the cluster deployment:

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

   ❶ For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

   ❷ 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.

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

> **NOTE**
>
> The cluster access and credential information also outputs to **<installation_directory>/.openshift_install.log** when an installation succeeds.

> **IMPORTANT**
>
> - The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
>
> - It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

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

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.

   - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.

   - If you included the **Service Account Key Admin** role, you can remove it.

1.6.8. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.
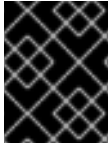
IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.6.8.1. Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Linux Client** entry and save the file.

4. Unpack the archive:

   ```
   $ tar xvzf <file>
   ```

5. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.6.8.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Windows Client** entry and save the file.

4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

   ```
   C:\> path
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

1.6.8.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 MacOSX Client** entry and save the file.

4. Unpack and unzip the archive.

5. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.6.9. Logging in to the cluster by using the CLI

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

- You deployed an OpenShift Container Platform cluster.

- You installed the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

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

   ❶ 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
   ```

Example output

> system:admin

Additional resources

- See Accessing the web console for more details about accessing and understanding the OpenShift Container Platform web console.

1.6.10. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.6.11. Next steps

- Customize your cluster.

- If necessary, you can opt out of remote health reporting .

1.7. INSTALLING A PRIVATE CLUSTER ON GCP

In OpenShift Container Platform version 4.6, you can install a private cluster into an existing VPC on Google Cloud Platform (GCP). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

1.7.1. Prerequisites

- Review details about the OpenShift Container Platform installation and update processes.

- Configure a GCP account to host the cluster.

- If you use a firewall, you must configure it to allow the sites that your cluster requires access to.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can manually create and maintain IAM credentials . Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.7.2. Private clusters

You can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the Internet.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.

To deploy a private cluster, you must use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.

Additionally, you must deploy a private cluster from a machine that has access the API services for the cloud you provision to, the hosts on the network that you provision, and to the internet to obtain installation media. You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network or a machine that has access to the network through a VPN.

1.7.2.1. Private clusters in GCP

To create a private cluster on Google Cloud Platform (GCP), you must provide an existing private VPC and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for only internal traffic.

The cluster still requires access to Internet to access the GCP APIs.

The following items are not required or created when you install a private cluster:

- Public subnets

- Public network load balancers, which support public ingress

- A public DNS zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private DNS zone and the required records for the cluster. The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

Because it is not possible to limit access to external load balancers based on source tags, the private cluster uses only internal load balancers to allow access to internal instances.

The internal load balancer relies on instance groups rather than the target pools that the network load balancers use. The installation program creates instance groups for each zone, even if there is no instance in that group.

- The cluster IP address is internal only.

- One forwarding rule manages both the Kubernetes API and machine config server ports.

- The backend service is comprised of each zone's instance group and, while it exists, the bootstrap instance group.

- The firewall uses a single rule that is based on only internal source ranges.

1.7.2.1.1. Limitations

No health check for the Machine config server, **/healthz**, runs because of a difference in load balancer functionality. Two internal load balancers cannot share a single IP address, but two network load balancers can share a single external IP address. Instead, the health of an instance is determined entirely by the **/readyz** check on port 6443.

1.7.3. About using a custom VPC

In OpenShift Container Platform 4.6, you can deploy a cluster into an existing VPC in Google Cloud Platform (GCP). If you do, you must also use existing subnets within the VPC and routing rules.

By deploying OpenShift Container Platform into an existing GCP VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. This is a good option to use if you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself.

1.7.3.1. Requirements for using your VPC

The installation program will no longer create the following components:

- VPC

- Subnets

- Cloud router

- Cloud NAT

- NAT IP addresses

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. The installation program cannot subdivide network ranges for the cluster to use, set route tables for the subnets, or set VPC options like DHCP, so you must do so before you install the cluster.

Your VPC and subnets must meet the following characteristics:

- The VPC must be in the same GCP project that you deploy the OpenShift Container Platform cluster to.

- To allow access to the Internet from the control plane and compute machines, you must configure cloud NAT on the subnets to allow egress to it. These machines do not have a public address. Even if you do not require access to the Internet, you must allow egress to the VPC network to obtain the installation program and images. Because multiple cloud NATs cannot be configured on the shared subnets, the installation program cannot configure it.

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist and belong to the VPC that you specified.

- The subnet CIDRs belong to the machine CIDR.

- You must provide a subnet to deploy the cluster control plane and compute machines to. You can use the same subnet for both machine types.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted.

1.7.3.2. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different

resources in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or Ingress rules.

The GCP credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as load balancers, security groups, storage, and nodes.

1.7.3.3. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is preserved by firewall rules that reference the machines in your cluster by the cluster's infrastructure ID. Only traffic within the cluster is allowed.

If you deploy multiple clusters to the same VPC, the following components might share access between clusters:

- The API, which is globally available with an external publishing strategy or available throughout the network in an internal publishing strategy

- Debugging tools, such as ports on VM instances that are open to the machine CIDR for SSH and ICMP access

1.7.4. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster.

You must have Internet access to:

- Access OpenShift Cluster Manager 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.

1.7.5. 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 the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

> **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 ed25519 -N '' \
       -f <path>/<file_name> 1
   ```

   1 Specify the path and file name, such as **~/.ssh/id_rsa**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your **~/.ssh** directory.

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

   > **NOTE**
   >
   > If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

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

   ```
   $ eval "$(ssh-agent -s)"
   ```

   Example output

   ```
   Agent pid 31874
   ```

   > **NOTE**
   >
   > If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

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

   ```
   $ ssh-add <path>/<file_name> 1
   ```

Example output

> 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**

4. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

   > $ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"

5. Verify that the credentials were applied.

   > $ gcloud auth list

Next steps

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

1.7.6. Obtaining the installation program

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

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space

Procedure

1. Access the Infrastructure Provider page on the OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Select your infrastructure provider.

3. 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 the installation program and the files that the installation program creates after you finish installing the cluster. Both files are required to delete the cluster.

   > **IMPORTANT**
   >
   > Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar xvf openshift-install-linux.tar.gz
```

5. Download your installation pull secret from the Red Hat OpenShift Cluster Manager . 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.

1.7.7. Manually creating the installation configuration file

For installations of a private OpenShift Container Platform cluster that are only accessible from an internal network and are not visible to the Internet, you must manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

> **IMPORTANT**
>
> You must create a 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.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.

> **NOTE**
>
> You must name this configuration file **install-config.yaml**.

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 next step of the installation process. You must back it up now.

1.7.7.1. 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**
>
> After installation, you cannot modify these parameters in the **install-config.yaml** file.

> **IMPORTANT**
>
> The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

1.7.7.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 1.21. Required parameters

Parameter	Description	Values
apiVersion	The API version for the **install-config.yaml** content. The current version is **v1**. The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain 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**.
metadata	Kubernetes resource **ObjectMeta**, from which only the **name** parameter is consumed.	Object

Parameter	Description	Values
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of **{{.metadata.name}}. {{.baseDomain}}**.	String of lowercase letters, hyphens (**-**), and periods (**.**), such as **dev**.
platform	The configuration for the specific platform upon which to perform the installation: **aws**, **baremetal**, **azure**, **openstack**, **ovirt**, **vsphere**. For additional information about **platform.<platform>** parameters, consult the following table for your specific platform.	Object

pullSecret	Get a pull secret from the Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{
		  "auths":{
		    "cloud.openshift.com":{
		      "auth":"b3Blb=",
		      "email":"you@example.com"
		    },
		    "quay.io":{
		      "auth":"b3Blb=",
		      "email":"you@example.com"
		    }
		  }
		}</pre>

1.7.7.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 1.22. Network parameters

Parameter	Description	Values

Parameter	Description	Values
networking	The configuration for the cluster network.	Object **NOTE** You cannot modify parameters specified by the **networking** object after installation.
networking.network Type	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either **OpenShiftSDN** or **OVNKubernetes**. The default value is **OpenShiftSDN**.

networking.clusterN etwork	The IP address blocks for pods.
	
	The default value is **10.128.0.0/14** with a host prefix of **/23**.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterN etwork.cidr	Required if you use **networking.clusterNetwork**. An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**.
networking.clusterN etwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23** then each node is assigned a **/23** subnet out of the given **cidr**. A **hostPrefix** value of **23** provides 510 (2^(32 – 23) – 2) pod IP addresses.	A subnet prefix. The default value is **23**.

networking.serviceN etwork	The IP address block for services. The default value is **172.30.0.0/16**.
	
	The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.
		An array with an IP address block in CIDR format. For example:
		
		``
		networking:
		  serviceNetwork:
		   - 172.30.0.0/16
		``

Parameter	Description	Values

networking.machine Network	The IP address blocks for machines.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		networking:
		  machineNetwork:
		  - cidr: 10.0.0.0/16

networking.machine Network.cidr	Required if you use **networking.machineNetwork**. An IP address block. The default value is **10.0.0.0/16** for all platforms other than libvirt. For libvirt, the default value is **192.168.126.0/24**.	An IP network block in CIDR notation. For example, **10.0.0.0/16**. **NOTE** Set the **networking.machin eNetwork** to match the CIDR that the preferred NIC resides in.

1.7.7.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 1.23. Optional parameters

Parameter	Description	Values
additionalTrustBund le	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architectur e	Determines the instruction set architecture of the machines in the pool. Currently, heteregeneous clusters are not supported, so all pools must specify the same architecture. Valid values are **amd64** (the default).	String

Parameter	Description	Values
compute.hyperthrea ding	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. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
compute.name	Required if you use **compute**. The name of the machine pool.	**worker**
compute.platform	Required if you use **compute**. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the **controlPlane.platform** parameter value.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere**, or **{}**
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	The configuration for the machines that comprise the control plane.	Array of **MachinePool** objects. For details, see the following "Machine-pool" table.
controlPlane.archite cture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are **amd64** (the default).	String

Parameter	Description	Values
controlPlane.hypert hreading	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. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
controlPlane.name	Required if you use **controlPlane**. The name of the machine pool.	**master**
controlPlane.platfor m	Required if you use **controlPlane**. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the **compute.platform** parameter value.	**aws**, **azure**, **gcp**, **openstack**, **ovirt**, **vsphere**, or **{}**
controlPlane.replica s	The number of control plane machines to provision.	The only supported value is **3**, which is the default value.

Parameter	Description	Values
credentialsMode	The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported. **NOTE** Not all CCO modes are supported for all cloud providers. For more information on CCO modes, see the *Cloud Credential Operator* entry in the *Red Hat Operators reference* content.	**Mint**, **Passthrough**, **Manual**, or an empty string (**""**).

Parameter	Description	Values
fips	Enable or disable FIPS mode. The default is **false** (disabled). 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. **IMPORTANT** The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture. **NOTE** If you are using Azure File storage, you cannot enable FIPS mode.	**false** or **true**
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a **source** and, optionally, **mirrors**, as described in the following rows of this table.
imageContentSources.source	Required if you use **imageContentSources**. Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings

Parameter	Description	Values
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	**Internal** or **External**. To deploy a private cluster, which cannot be accessed from the internet, set **publish** to **Internal**. The default value is **External**.
sshKey	The SSH key or keys to authenticate 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.	One or more keys. For example: sshKey: <key1> <key2> <key3>

1.7.7.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 1.24. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.region	The name of the GCP region that hosts your cluster.	Any valid region name, such as **us-central1**.
platform.gcp.type	The GCP machine type.	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones, such as **us-central1-a**, in a YAML sequence.

Parameter	Description	Values
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.
platform.gcp.licenses	A list of license URLs that must be applied to the compute images. IMPORTANT The **licenses** parameter is a deprecated field and nested virtualization is enabled by default. It is not recommended to use this field.	Any license available with the license API, such as the license to enable nested virtualization. You cannot use this parameter with a mechanism that generates pre-built images. Using a license URL forces the installer to copy the source image before use.
platform.gcp.osDisk.diskSizeGB	The size of the disk in gigabytes (GB).	Any size between 16 GB and 65536 GB.
platform.gcp.osDisk.diskType	The type of disk.	Either the default **pd-ssd** or the **pd-standard** disk type. The control plane nodes must be the **pd-ssd** disk type. The worker nodes can be either type.

1.7.7.2. Sample customized install-config.yaml file for GCP

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

 IMPORTANT

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

```
apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-ssd
        diskSizeGB: 1024
  replicas: 3
compute: 5 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      osDisk:
        diskType: pd-standard
        diskSizeGB: 128
  replicas: 3
metadata:
  name: test-cluster 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
    network: existing_vpc 11
    controlPlaneSubnet: control_plane_subnet 12
    computeSubnet: compute_subnet 13
pullSecret: '{"auths": ...}' 14
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
publish: Internal 17
```

1 **8** **9** **10** **14** Required. The installation program prompts you for this value.

2 5 If you do not provide these parameters and values, the installation program provides the default value.

3 6 The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Only one control plane pool is used.

4 7 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.

> **IMPORTANT**
>
> If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

11 Specify the name of an existing VPC.

12 Specify the name of the existing subnet to deploy the control plane machines to. The subnet must belong to the VPC that you specified.

13 Specify the name of the existing subnet to deploy the compute machines to. The subnet must belong to the VPC that you specified.

15 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.

> **IMPORTANT**
>
> The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture.

16 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

> **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.

17 How to publish the user-facing endpoints of your cluster. Set **publish** to **Internal** to deploy a private cluster, which cannot be accessed from the Internet. The default value is **External**.

1.7.7.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

  > **NOTE**
  >
  > The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.
  >
  > For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

   ```
   apiVersion: v1
   baseDomain: my.domain.com
   proxy:
     httpProxy: http://<username>:<pswd>@<ip>:<port>   1
     httpsProxy: https://<username>:<pswd>@<ip>:<port>   2
     noProxy: example.com   3
   additionalTrustBundle: |   4
       -----BEGIN CERTIFICATE-----
       <MY_TRUSTED_CA_CERT>
       -----END CERTIFICATE-----
   ...
   ```

   1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

   2 A proxy URL to use for creating HTTPS connections outside the cluster.

   3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.

   4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace to hold the additional CA certificates. If you provide **additionalTrustBundle** and at least one proxy setting, the **Proxy** object is configured to reference the **user-ca-bundle** config map in the **trustedCA** field. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges the contents

specified for the **trustedCA** parameter with the RHCOS trust bundle. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

> **NOTE**
>
> The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

> **NOTE**
>
> Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

1.7.8. 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

- Configure an account with the cloud platform that hosts your cluster.

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

Procedure

1. Change to the directory that contains the installation program and initialize the cluster deployment:

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

   ❶ For **<installation_directory>**, specify the

   ❷ 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.

Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "4vYBz-Ee6gm-ymBZj-
Wt5AL"
INFO Time elapsed: 36m22s
```

> **NOTE**
>
> The cluster access and credential information also outputs to **<installation_directory>/.openshift_install.log** when an installation succeeds.

> **IMPORTANT**
>
> - The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
>
> - It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

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

1.7.9. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

> **IMPORTANT**
>
> If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.7.9.1. Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Linux Client** entry and save the file.

4. Unpack the archive:

   ```
   $ tar xvzf <file>
   ```

5. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.7.9.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Windows Client** entry and save the file.

4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

   ```
   C:\> path
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

1.7.9.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 MacOSX Client** entry and save the file.

4. Unpack and unzip the archive.

5. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.7.10. Logging in to the cluster by using the CLI

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

- You deployed an OpenShift Container Platform cluster.

- You installed the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

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

   ❶ 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
   ```

Example output

```
system:admin
```

Additional resources

- See Accessing the web console for more details about accessing and understanding the OpenShift Container Platform web console.

1.7.11. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.7.12. Next steps

- Customize your cluster.

- If necessary, you can opt out of remote health reporting .

1.8. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES

In OpenShift Container Platform version 4.6, you can install a cluster on Google Cloud Platform (GCP) that uses infrastructure that you provide.

The steps for performing a user-provided infrastructure install are outlined here. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods.

> **IMPORTANT**
>
> The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

1.8.1. Prerequisites

- Review details about the OpenShift Container Platform installation and update processes.

- If you use a firewall and plan to use telemetry, you must configure the firewall to allow the sites that your cluster requires access to.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can manually create and maintain IAM credentials . Manual mode can also be used in environments where the cloud IAM APIs are not reachable.
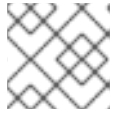
> **NOTE**
>
> Be sure to also review this site list if you are configuring a proxy.

1.8.2. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

1.8.3. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster.

You must have Internet access to:

- Access OpenShift Cluster Manager 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.

1.8.4. Configuring your GCP project

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

1.8.4.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See Creating and Managing Projects in the GCP documentation.
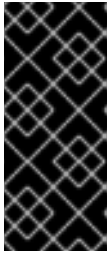
> **IMPORTANT**
>
> Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>.<base_domain>** URL; the Premium Tier is required for internal load balancing.

1.8.4.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See Enabling services in the GCP documentation.

Table 1.25. Required API services

API service	Console service name
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
Compute Engine API	**compute.googleapis.com**
Google Cloud APIs	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**

API service	Console service name
Cloud Storage	**storage-component.googleapis.com**

1.8.4.3. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the same project that you host the OpenShift Container Platform cluster. This zone must be authoritative for the domain. The DNS service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through GCP or another source.

   > **NOTE**
   >
   > If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see Google Domains.

2. Create a public hosted zone for your domain or subdomain in your GCP project. See Creating public zones in the GCP documentation.
   Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.

3. Extract the new authoritative name servers from the hosted zone records. See Look up your Cloud DNS name servers in the GCP documentation.
   You typically have four name servers.

4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: How to switch to custom name servers .

5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See Migrating to Cloud DNS in the GCP documentation.

6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

1.8.4.4. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default Quotas do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 1.26. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall rules	Networking	Global	11	1
Forwarding rules	Compute	Global	2	0
Health checks	Compute	Global	2	0
Images	Compute	Global	1	0
Networks	Networking	Global	1	0
Routers	Networking	Global	1	0
Routes	Networking	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0

NOTE

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**

- **asia-northeast2**

- **asia-south1**

- **australia-southeast1**

- **europe-north1**

- **europe-west2**

- **europe-west3**

- **europe-west6**

- **northamerica-northeast1**

- **southamerica-east1**

- **us-west2**

You can increase resource quotas from the GCP console, but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

1.8.4.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a service account in the project that you use to host your OpenShift Container Platform cluster. See Creating a service account in the GCP documentation.

2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See Granting roles to a service account for specific resources.

   > **NOTE**
   >
   > While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. Create the service account key in JSON format. See Creating service account keys in the GCP documentation.
   The service account key is required to create a cluster.

1.8.4.5.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions. If you deploy your cluster into an existing VPC, the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin

- Security Admin

- Service Account Admin

- Service Account User

- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor

- Service Account Key Admin

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 1.27. GCP service account permissions

Account	Roles
Control Plane	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	**roles/compute.viewer**
	roles/storage.admin

1.8.4.6. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- **asia-east1** (Changhua County, Taiwan)

- **asia-east2** (Hong Kong)

- **asia-northeast1** (Tokyo, Japan)

- **asia-northeast2** (Osaka, Japan)

- **asia-northeast3** (Seoul, South Korea)

- **asia-south1** (Mumbai, India)

- **asia-southeast1** (Jurong West, Singapore)

- **asia-southeast2** (Jakarta, Indonesia)

- **australia-southeast1** (Sydney, Australia)

- **europe-north1** (Hamina, Finland)

- **europe-west1** (St. Ghislain, Belgium)

- **europe-west2** (London, England, UK)

- **europe-west3** (Frankfurt, Germany)

- **europe-west4** (Eemshaven, Netherlands)

- **europe-west6** (Zürich, Switzerland)

- **northamerica-northeast1** (Montréal, Québec, Canada)

- **southamerica-east1** (São Paulo, Brazil)

- **us-central1** (Council Bluffs, Iowa, USA)

- **us-east1** (Moncks Corner, South Carolina, USA)

- **us-east4** (Ashburn, Northern Virginia, USA)

- **us-west1** (The Dalles, Oregon, USA)

- **us-west2** (Los Angeles, California, USA)

- **us-west3** (Salt Lake City, Utah, USA)

- **us-west4** (Las Vegas, Nevada, USA)

1.8.4.7. Installing and configuring CLI tools for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must install and configure the CLI tools for GCP.

Prerequisites

- You created a project to host your cluster.

- You created a service account and granted it the required permissions.

Procedure

1. Install the following binaries in **$PATH**:

    - **gcloud**

- **gsutil**

  See Install the latest Cloud SDK version in the GCP documentation.

2. Authenticate using the **gcloud** tool with your configured service account.
   See Authorizing with a service account in the GCP documentation.

1.8.5. Creating the installation files for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files. You also have the option to first set up a separate **var** partition during the preparation phases of installation.

1.8.5.1. Optional: Creating a separate /var partition

It is recommended that disk partitioning for OpenShift Container Platform be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the /**var** partition or a subdirectory of /**var**. For example:

- /**var**/**lib**/**containers**: Holds container-related content that can grow as more images and containers are added to a system.

- /**var**/**lib**/**etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.

- /**var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a /**var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because /**var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate /**var** partition by creating a machine config that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform installation.

> **IMPORTANT**
>
> If you follow the steps to create a separate /**var** partition in this procedure, it is not necessary to create the Kubernetes manifest and Ignition config files again as described later in this section.

Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

   ```
   $ mkdir $HOME/clusterconfig
   ```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

Example output

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. Optional: Confirm that the installation program created manifests in the **clusterconfig/openshift** directory:

```
$ ls $HOME/clusterconfig/openshift/
```

Example output

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. Create a **MachineConfig** object and add it to a file in the **openshift** directory. For example, name the file **98-var-partition.yaml**, change the disk device name to the name of the storage device on the **worker** systems, and set the storage size as appropriate. This example places the **/var** directory on a separate partition:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      disks:
      - device: /dev/<device_name>     1
        partitions:
        - label: var
          startMiB: <partition_start_offset>     2
          sizeMiB: <partition_size>     3
      filesystems:
        - device: /dev/disk/by-partlabel/var
          path: /var
          format: xfs
    systemd:
      units:
        - name: var.mount     4
          enabled: true
```

```
contents: |
 [Unit]
 Before=local-fs.target
 [Mount]
 What=/dev/disk/by-partlabel/var
 Where=/var
 Options=defaults,prjquota  5
 [Install]
 WantedBy=local-fs.target
```

1 The storage device name of the disk that you want to partition.

2 When adding a data partition to the boot disk, a minimum value of 25000 MiB (Mebibytes) is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.

3 The size of the data partition in mebibytes.

4 The name of the mount unit must match the directory specified in the **Where=** directive. For example, for a filesystem mounted on **/var/lib/containers**, the unit must be named **var-lib-containers.mount**.

5 The **prjquota** mount option must be enabled for filesystems used for container storage.

> **NOTE**
>
> When creating a separate **/var** partition, you cannot use different instance types for worker nodes, if the different instance types do not have the same device name.

5. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Now you can use the Ignition config files as input to the installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

1.8.5.2. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

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

Procedure

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

    a. Change to the directory that contains the installation program and 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 **gcp** as the platform to target.

    iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.

    iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.

    v. Select the region to deploy the cluster to.

    vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.

    vii. Enter a descriptive name for your cluster.

    viii. Paste the pull secret from the Red Hat OpenShift Cluster Manager .

    c. Optional: If you do not want the cluster to provision compute machines, empty the compute pool by editing the resulting **install-config.yaml** file to set **replicas** to **0** for the **compute** pool:

    ```
    compute:
    - hyperthreading: Enabled
    ```

```
        name: worker
        platform: {}
        replicas: 0  1
```

1   Set to **0**.

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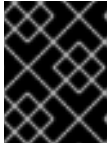
> **IMPORTANT**
>
> The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

1.8.5.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

> **NOTE**
>
> The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.
>
> For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port>  1
  httpsProxy: https://<username>:<pswd>@<ip>:<port>  2
  noProxy: example.com  3
additionalTrustBundle: |  4
```

```
-----BEGIN CERTIFICATE-----
<MY_TRUSTED_CA_CERT>
-----END CERTIFICATE-----
...
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace to hold the additional CA certificates. If you provide **additionalTrustBundle** and at least one proxy setting, the **Proxy** object is configured to reference the **user-ca-bundle** config map in the **trustedCA** field. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges the contents specified for the **trustedCA** parameter with the RHCOS trust bundle. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

1.8.5.4. 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.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to create the cluster.

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

Prerequisites

- You obtained the OpenShift Container Platform installation program.

- You created the **install-config.yaml** installation configuration file.

Procedure

1. Change to the directory that contains the installation program and generate the Kubernetes manifests for the cluster:

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

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

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

   ```
   $ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
   ```

   By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

   ```
   $ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
   ```

   Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:

   a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.

   b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.

c. Save and exit the file.

5. Optional: If you do not want the Ingress Operator to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 **2** Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

6. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

```
$ ./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
```

Additional resources

- Optional: Adding the ingress DNS records

1.8.6. Exporting common variables

1.8.6.1. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The infrastructure name is also used to locate the appropriate GCP resources during an OpenShift Container Platform installation. The provided Deployment Manager

templates contain references to this infrastructure name, so you must extract it.

Prerequisites

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

- You generated the Ignition config files for your cluster.

- You installed the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

      $ jq -r .infraID <installation_directory>/metadata.json **1**

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

  Example output

      openshift-vw9j6 **1**

  1   The output of this command is your cluster name and a random string.

1.8.6.2. Exporting common variables for Deployment Manager templates

You must export a common set of variables that are used with the provided Deployment Manager templates used to assist in completing a user-provided infrastructure install on Google Cloud Platform (GCP).

> **NOTE**
>
> Specific Deployment Manager templates can also require additional exported variables, which are detailed in their related procedures.

Prerequisites

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

- Generate the Ignition config files for your cluster.

- Install the **jq** package.

Procedure

1. Export the following common variables to be used by the provided Deployment Manager templates:

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/19'
$ export WORKER_SUBNET_CIDR='10.0.32.0/19'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig ❶
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

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

1.8.7. Creating a VPC in GCP

You must create a VPC in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements. One way to create the VPC is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

Procedure

1. Copy the template from the **Deployment Manager template for the VPC** section of this topic and save it as **01_vpc.py** on your computer. This template describes the VPC that your cluster requires.

2. Create a **01_vpc.yaml** resource definition file:

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' ❶
    region: '${REGION}' ❷
```

```
        master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
        worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
      EOF
```

1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

2 **region** is the region to deploy the cluster into, for example **us-central1**.

3 **master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/19**.

4 **worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.32.0/19**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

1.8.7.1. Deployment Manager template for the VPC

You can use the following Deployment Manager template to deploy the VPC that you need for your OpenShift Container Platform cluster:

Example 1.1. 01_vpc.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
```

```
        'nats': [{
            'name': context.properties['infra_id'] + '-nat-master',
            'natIpAllocateOption': 'AUTO_ONLY',
            'minPortsPerVm': 7168,
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }, {
            'name': context.properties['infra_id'] + '-nat-worker',
            'natIpAllocateOption': 'AUTO_ONLY',
            'minPortsPerVm': 512,
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }]
        }
    }]

    return {'resources': resources}
```

1.8.8. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 1.28. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	**1936**	Metrics
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101** and the Cluster Version Operator on port**9099**.
	10250-**10259**	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	**4789**	VXLAN and Geneve
	6081	VXLAN and Geneve

Protocol	Port	Description
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101**.
TCP/UDP	**30000**-**32767**	Kubernetes node port

Table 1.29. All machines to control plane

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 1.30. Control plane machines to control plane machines

Protocol	Port	Description
TCP	**2379**-**2380**	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.

> **IMPORTANT**
>
> OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer**: Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.

   - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

     > **IMPORTANT**
     >
     > Do not configure session persistence for an API load balancer.

   Configure the following ports on both the front and back of the load balancers:

Table 1.31. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the **/readyz** endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

> **NOTE**
>
> The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer**: Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

   - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

   Configure the following ports on both the front and back of the load balancers:

Table 1.32. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

> **NOTE**
>
> A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

1.8.9. Creating load balancers in GCP

You must configure load balancers in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.

2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.

3. Export the variables that the deployment template uses:

   a. Export the cluster network location:

   ```
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
   ```

   b. Export the control plane subnet location:

   ```
   $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   c. Export the three zones that the cluster uses:

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[2] | cut -d "/" -f9`)
```

4. Create a **02_infra.yaml** resource definition file:

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py 1
resources:
- name: cluster-lb-ext 2
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' 3
    region: '${REGION}' 4
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' 5
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: 6
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

1 2 Required only when deploying an external cluster.

3 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

4 **region** is the region to deploy the cluster into, for example **us-central1**.

5 **control_subnet** is the URI to the control subnet.

6 **zones** are the zones to deploy the control plane instances into, like **us-east1-b**, **us-east1-c**, and **us-east1-d**.

5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. Export the cluster IP address:

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --
region=${REGION} --format json | jq -r .address`)
```

7. For an external cluster, also export the cluster public IP address:

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-
public-ip --region=${REGION} --format json | jq -r .address`)
```

1.8.9.1. Deployment Manager template for the external load balancer

You can use the following Deployment Manager template to deploy the external load balancer that you need for your OpenShift Container Platform cluster:

Example 1.2. **02_lb_ext.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

1.8.9.2. Deployment Manager template for the internal load balancer

You can use the following Deployment Manager template to deploy the internal load balancer that you need for your OpenShift Container Platform cluster:

Example 1.3. 02_lb_int.py Deployment Manager template

```python
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' + '.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
```

```
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

1.8.10. Creating a private DNS zone in GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.

2. Create a **02_dns.yaml** resource definition file:

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF
```

1   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

2   **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.

3   **cluster_network** is the **selfLink** URL to the cluster network.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. The templates do not create DNS entries due to limitations of Deployment Manager, so you must create them manually:

   a. Add the internal DNS entries:

   ```
   $ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
   $ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
   $ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
   api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
   private-zone
   $ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
   int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
   private-zone
   $ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
   ```

   b. For an external cluster, also add the external DNS entries:

   ```
   $ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
   $ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
   $ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
   api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
   ${BASE_DOMAIN_ZONE_NAME}
   $ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
   ```

1.8.10.1. Deployment Manager template for the private DNS

You can use the following Deployment Manager template to deploy the private DNS that you need for your OpenShift Container Platform cluster:

Example 1.4. 02_dns.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

1.8.11. Creating firewall rules in GCP

You must create firewall rules in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.

2. Create a **03_firewall.yaml** resource definition file:

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' 1
    infra_id: '${INFRA_ID}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
    network_cidr: '${NETWORK_CIDR}' 4
EOF
```

1   **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **${NETWORK_CIDR}**.

2   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3   **cluster_network** is the **selfLink** URL to the cluster network.

4   **network_cidr** is the CIDR of the VPC network, for example **10.0.0.0/16**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

1.8.11.1. Deployment Manager template for firewall rules

You can use the following Deployment Manager template to deploy the firewall rues that you need for your OpenShift Container Platform cluster:

Example 1.5. 03_firewall.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
```

```
      'type': 'compute.v1.firewall',
      'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
          'IPProtocol': 'tcp',
          'ports': ['6443']
        }],
        'sourceRanges': [context.properties['allowed_external_cidr']],
        'targetTags': [context.properties['infra_id'] + '-master']
      }
    }, {
      'name': context.properties['infra_id'] + '-health-checks',
      'type': 'compute.v1.firewall',
      'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
          'IPProtocol': 'tcp',
          'ports': ['6080', '6443', '22624']
        }],
        'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
        'targetTags': [context.properties['infra_id'] + '-master']
      }
    }, {
      'name': context.properties['infra_id'] + '-etcd',
      'type': 'compute.v1.firewall',
      'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
          'IPProtocol': 'tcp',
          'ports': ['2379-2380']
        }],
        'sourceTags': [context.properties['infra_id'] + '-master'],
        'targetTags': [context.properties['infra_id'] + '-master']
      }
    }, {
      'name': context.properties['infra_id'] + '-control-plane',
      'type': 'compute.v1.firewall',
      'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
          'IPProtocol': 'tcp',
          'ports': ['10257']
        },{
          'IPProtocol': 'tcp',
          'ports': ['10259']
        },{
          'IPProtocol': 'tcp',
          'ports': ['22623']
        }],
        'sourceTags': [
          context.properties['infra_id'] + '-master',
          context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [context.properties['infra_id'] + '-master']
      }
    }, {
```

```
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }]

    return {'resources': resources}
```

1.8.12. Creating IAM roles in GCP

You must create IAM roles in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.

2. Create a **03_iam.yaml** resource definition file:

   ```
   $ cat <<EOF >03_iam.yaml
   imports:
   - path: 03_iam.py
   resources:
   - name: cluster-iam
     type: 03_iam.py
     properties:
       infra_id: '${INFRA_ID}' ❶
   EOF
   ```

   ❶ **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3. Create the deployment by using the **gcloud** CLI:

   ```
   $ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
   ```

4. Export the variable for the master service account:

   ```
   $ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

5. Export the variable for the worker service account:

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

6. Export the variable for the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. The templates do not create the policy bindings due to limitations of Deployment Manager, so you must create them manually:

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. Create a service account key and store it locally for later use:

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-account=${MASTER_SERVICE_ACCOUNT}
```

1.8.12.1. Deployment Manager template for IAM roles

You can use the following Deployment Manager template to deploy the IAM roles that you need for your OpenShift Container Platform cluster:

Example 1.6. 03_iam.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
```

```
}]

    return {'resources': resources}
```

1.8.13. Creating the RHCOS cluster image for the GCP infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) image for Google Cloud Platform (GCP) for your OpenShift Container Platform nodes.

Procedure

1. Obtain the RHCOS image from the RHCOS image mirror page.

   > **IMPORTANT**
   >
   > The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

   The file name contains the OpenShift Container Platform version number in the format **rhcos-<version>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

   ```
   $ gsutil mb gs://<bucket_name>
   ```

3. Upload the RHCOS image to the Google storage bucket:

   ```
   $ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
   gs://<bucket_name>
   ```

4. Export the uploaded RHCOS image location as a variable:

   ```
   $ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-
   gcp.x86_64.tar.gz`
   ```

5. Create the cluster image:

   ```
   $ gcloud compute images create "${INFRA_ID}-rhcos-image" \
       --source-uri="${IMAGE_SOURCE}"
   ```

1.8.14. Creating the bootstrap machine in GCP

You must create the bootstrap machine in Google Cloud Platform (GCP) to use during OpenShift Container Platform cluster initialization. One way to create this machine is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your bootstrap machine, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Ensure pyOpenSSL is installed.

Procedure

1. Copy the template from the **Deployment Manager template for the bootstrap machine** section of this topic and save it as **04_bootstrap.py** on your computer. This template describes the bootstrap machine that your cluster requires.

2. Export the location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

   ```
   $ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
   ```

3. Create a bucket and upload the **bootstrap.ign** file:

   ```
   $ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
   $ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
   ```

4. Create a signed URL for the bootstrap instance to use to access the Ignition config. Export the URL from the output as a variable:

   ```
   $ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
   ```

5. Create a **04_bootstrap.yaml** resource definition file:

   ```
   $ cat <<EOF >04_bootstrap.yaml
   imports:
   - path: 04_bootstrap.py

   resources:
   - name: cluster-bootstrap
     type: 04_bootstrap.py
     properties:
       infra_id: '${INFRA_ID}'
   ```

```
        region: '${REGION}' 2
        zone: '${ZONE_0}' 3

        cluster_network: '${CLUSTER_NETWORK}' 4
        control_subnet: '${CONTROL_SUBNET}' 5
        image: '${CLUSTER_IMAGE}' 6
        machine_type: 'n1-standard-4' 7
        root_volume_size: '128' 8

        bootstrap_ign: '${BOOTSTRAP_IGN}' 9
    EOF
```

1   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

2   **region** is the region to deploy the cluster into, for example **us-central1**.

3   **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-b**.

4   **cluster_network** is the **selfLink** URL to the cluster network.

5   **control_subnet** is the **selfLink** URL to the control subnet.

6   **image** is the **selfLink** URL to the RHCOS image.

7   **machine_type** is the machine type of the instance, for example **n1-standard-4**.

8   **root_volume_size** is the boot disk size for the bootstrap machine.

9   **bootstrap_ign** is the URL output when creating a signed URL.

6. Create the deployment by using the **gcloud** CLI:

   ```
   $ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
   04_bootstrap.yaml
   ```

7. The templates do not manage load balancer membership due to limitations of Deployment
   Manager, so you must add the bootstrap machine manually.

   a. Add the bootstrap instance to the internal load balancer instance group:

      ```
      $ gcloud compute instance-groups unmanaged add-instances \
          ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
      bootstrap
      ```

   b. Add the bootstrap instance group to the internal load balancer backend service:

      ```
      $ gcloud compute backend-services add-backend \
          ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
      group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
      ```

1.8.14.1. Deployment Manager template for the bootstrap machine

You can use the following Deployment Manager template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster:

Example 1.7. 04_bootstrap.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
```

```
        }, {
            'name': 'https',
            'port': 6443
        }
    ],
    'network': context.properties['cluster_network'],
    'zone': context.properties['zone']
    }
}]

return {'resources': resources}
```

1.8.15. Creating the control plane machines in GCP

You must create the control plane machines in Google Cloud Platform (GCP) for your cluster to use. One way to create these machines is to modify the provided Deployment Manager template.

NOTE

If you do not use the provided Deployment Manager template to create your control plane machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

Procedure

1. Copy the template from the **Deployment Manager template for control plane machines** section of this topic and save it as **05_control_plane.py** on your computer. This template describes the control plane machines that your cluster requires.

2. Export the following variable required by the resource definition:

   ```
   $ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
   ```

3. Create a **05_control_plane.yaml** resource definition file:

   ```
   $ cat <<EOF >05_control_plane.yaml
   imports:
   - path: 05_control_plane.py
   ```

```
resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}'  1
    zones:  2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}'  3
    image: '${CLUSTER_IMAGE}'  4
    machine_type: 'n1-standard-4'  5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}'  6

    ignition: '${MASTER_IGNITION}'  7
EOF
```

1   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

2   **zones** are the zones to deploy the control plane instances into, for example **us-central1-a**, **us-central1-b**, and **us-central1-c**.

3   **control_subnet** is the **selfLink** URL to the control subnet.

4   **image** is the **selfLink** URL to the RHCOS image.

5   **machine_type** is the machine type of the instance, for example **n1-standard-4**.

6   **service_account_email** is the email address for the master service account that you created.

7   **ignition** is the contents of the **master.ign** file.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually.

   - Run the following commands to add the control plane machines to the appropriate instance groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

1.8.15.1. Deployment Manager template for control plane machines

You can use the following Deployment Manager template to deploy the control plane machines that you need for your OpenShift Container Platform cluster:

Example 1.8. **05_control_plane.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
```

```
      'name': context.properties['infra_id'] + '-master-1',
      'type': 'compute.v1.instance',
      'properties': {
        'disks': [{
          'autoDelete': True,
          'boot': True,
          'initializeParams': {
            'diskSizeGb': context.properties['root_volume_size'],
            'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
            'sourceImage': context.properties['image']
          }
        }],
        'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
          'items': [{
            'key': 'user-data',
            'value': context.properties['ignition']
          }]
        },
        'networkInterfaces': [{
          'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
          'email': context.properties['service_account_email'],
          'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
          'items': [
            context.properties['infra_id'] + '-master',
          ]
        },
        'zone': context.properties['zones'][1]
      }
    }, {
      'name': context.properties['infra_id'] + '-master-2',
      'type': 'compute.v1.instance',
      'properties': {
        'disks': [{
          'autoDelete': True,
          'boot': True,
          'initializeParams': {
            'diskSizeGb': context.properties['root_volume_size'],
            'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
            'sourceImage': context.properties['image']
          }
        }],
        'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
          'items': [{
            'key': 'user-data',
            'value': context.properties['ignition']
          }]
        },
        'networkInterfaces': [{
```

```
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
    }]

    return {'resources': resources}
```

1.8.16. Wait for bootstrap completion and remove bootstrap resources in GCP

After you create all of the required infrastructure in Google Cloud Platform (GCP), wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

    ```
    $ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ ❶
        --log-level info ❷
    ```

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

    ❷ To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

    If the command exits without a **FATAL** warning, your production control plane has initialized.

2. Delete the bootstrap resources:

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

1.8.17. Creating additional worker machines in GCP

You can create worker machines in Google Cloud Platform (GCP) for your cluster to use by launching individual instances discretely or by automated processes outside the cluster, such as auto scaling groups. You can also take advantage of the built-in cluster scaling mechanisms and the machine API in OpenShift Container Platform.

In this example, you manually launch one instance by using the Deployment Manager template. Additional instances can be launched by including additional resources of type **06_worker.py** in the file.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your worker machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

Procedure

1. Copy the template from the **Deployment Manager template for worker machines** section of this topic and save it as **06_worker.py** on your computer. This template describes the worker machines that your cluster requires.

2. Export the variables that the resource definition uses.

   a. Export the subnet that hosts the compute machines:

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

b. Export the email address for your service account:

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

c. Export the location of the compute machine Ignition config file:

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. Create a **06_worker.yaml** resource definition file:

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' 1
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 2
    zone: '${ZONE_0}' 3
    compute_subnet: '${COMPUTE_SUBNET}' 4
    image: '${CLUSTER_IMAGE}' 5
    machine_type: 'n1-standard-4' 6
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
    ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
    compute_subnet: '${COMPUTE_SUBNET}' 11
    image: '${CLUSTER_IMAGE}' 12
    machine_type: 'n1-standard-4' 13
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
    ignition: '${WORKER_IGNITION}' 15
EOF
```

1   **name** is the name of the worker machine, for example **worker-0**.

2 **9** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3 **10** **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.

4 **11** **compute_subnet** is the **selfLink** URL to the compute subnet.

5 **12** **image** is the **selfLink** URL to the RHCOS image.

6 **13** **machine_type** is the machine type of the instance, for example **n1-standard-4**.

7 **14** **service_account_email** is the email address for the worker service account that you created.

created.

8 **15** **ignition** is the contents of the **worker.ign** file.

4. Optional: If you want to launch additional instances, include additional resources of type **06_worker.py** in your **06_worker.yaml** resource definition file.

5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1.8.17.1. Deployment Manager template for worker machines

You can use the following Deployment Manager template to deploy the worker machines that you need for your OpenShift Container Platform cluster:

Example 1.9. 06_worker.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
```

```
    }]

    return {'resources': resources}
```

1.8.18. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

> **IMPORTANT**
>
> If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.8.18.1. Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Linux Client** entry and save the file.

4. Unpack the archive:

   ```
   $ tar xvzf <file>
   ```

5. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.8.18.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Windows Client** entry and save the file.

4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

   ```
   C:\> path
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

1.8.18.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 MacOSX Client** entry and save the file.

4. Unpack and unzip the archive.

5. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.8.19. Logging in to the cluster by using the CLI

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

- You deployed an OpenShift Container Platform cluster.

- You installed the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

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

❶ 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
```

Example output

```
system:admin
```

1.8.20. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES   AGE  VERSION
master-0  Ready    master  63m  v1.19.0
master-1  Ready    master  63m  v1.19.0
master-2  Ready    master  64m  v1.19.0
```

The output lists all of the machines that you created.

> **NOTE**
>
> The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME        AGE    REQUESTOR                                              CONDITION
csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

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. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

> **NOTE**
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

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

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

  ❶ **<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 --no-run-if-empty oc adm certificate approve
  ```

  > **NOTE**
  >
  > Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME        AGE     REQUESTOR                                              CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

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

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

  ❶   **<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
  ```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.20.0
master-1  Ready    master  73m  v1.20.0
master-2  Ready    master  74m  v1.20.0
worker-0  Ready    worker  11m  v1.20.0
worker-1  Ready    worker  11m  v1.20.0
```

> **NOTE**
>
> It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests .

1.8.21. Optional: Adding the ingress DNS records

If you removed the DNS zone configuration when creating Kubernetes manifests and generating Ignition configs, you must manually create DNS records that point at the ingress load balancer. You can create either a wildcard ***.apps.{baseDomain}.** or specific records. You can use A, CNAME, and other records per your requirements.

Prerequisites

- Configure a GCP account.

- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

- Create the worker machines.

Procedure

1. Wait for the Ingress router to create a load balancer and populate the **EXTERNAL-IP** field:

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   Example output

   ```
   NAME            TYPE          CLUSTER-IP      EXTERNAL-IP     PORT(S)                 AGE
   router-default  LoadBalancer  172.30.18.154   35.233.157.184
   80:32288/TCP,443:31215/TCP   98
   ```

2. Add the A record to your zones:

   - To use A records:

     i. Export the variable for the router IP address:

        ```
        $ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-headers | awk '{print $4}'`
        ```

     ii. Add the A record to the private zones:

        ```
        $ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
        $ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
        $ gcloud dns record-sets transaction add ${ROUTER_IP} --name
        \*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
        ${INFRA_ID}-private-zone
        $ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
        ```

iii. For an external cluster, also add the A record to the public zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- To add explicit domains instead of using a wildcard, create entries for each of the cluster's current routes:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

Example output

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

1.8.22. Completing a GCP installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Google Cloud Platform (GCP) user-provisioned infrastructure, you can monitor the cluster events until the cluster is ready.

Prerequisites

- Deploy the bootstrap machine for an OpenShift Container Platform cluster on user-provisioned GCP infrastructure.

- Install the **oc** CLI and log in.

Procedure

1. Complete the cluster installation:

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete ❶
```

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

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

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Observe the running state of your cluster.

   a. Run the following command to view the current cluster version and status:

   ```
   $ oc get clusterversion
   ```

   Example output

   ```
   NAME      VERSION  AVAILABLE  PROGRESSING  SINCE  STATUS
   version            False      True         24m    Working towards 4.5.4: 99% complete
   ```

   b. Run the following command to view the Operators managed on the control plane by the Cluster Version Operator (CVO):

   ```
   $ oc get clusteroperators
   ```

   Example output

   ```
   NAME                             VERSION  AVAILABLE  PROGRESSING  DEGRADED
   SINCE
   authentication                   4.5.4    True       False        False     7m56s
   cloud-credential                 4.5.4    True       False        False     31m
   cluster-autoscaler               4.5.4    True       False        False     16m
   console                          4.5.4    True       False        False     10m
   csi-snapshot-controller          4.5.4    True       False        False     16m
   dns                              4.5.4    True       False        False     22m
   etcd                             4.5.4    False      False        False     25s
   image-registry                   4.5.4    True       False        False     16m
   ingress                          4.5.4    True       False        False     16m
   insights                         4.5.4    True       False        False     17m
   kube-apiserver                   4.5.4    True       False        False     19m
   kube-controller-manager          4.5.4    True       False        False     20m
   kube-scheduler                   4.5.4    True       False        False     20m
   kube-storage-version-migrator    4.5.4    True       False        False     16m
   machine-api                      4.5.4    True       False        False     22m
   machine-config                   4.5.4    True       False        False     22m
   marketplace                      4.5.4    True       False        False     16m
   ```

```
monitoring                          4.5.4    True    False       False     10m
network                             4.5.4    True    False       False     23m
node-tuning                         4.5.4    True    False       False     23m
openshift-apiserver                 4.5.4    True    False       False     17m
openshift-controller-manager        4.5.4    True    False       False     15m
openshift-samples                   4.5.4    True    False       False     16m
operator-lifecycle-manager          4.5.4    True    False       False     22m
operator-lifecycle-manager-catalog  4.5.4    True    False       False     22m
operator-lifecycle-manager-packageserver  4.5.4  True  False     False     18m
service-ca                          4.5.4    True    False       False     23m
service-catalog-apiserver           4.5.4    True    False       False     23m
service-catalog-controller-manager  4.5.4    True    False       False     23m
storage                             4.5.4    True    False       False     17m
```

c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

```
NAMESPACE                             NAME
READY    STATUS     RESTARTS  AGE
kube-system                                 etcd-member-ip-10-0-3-111.us-east-
2.compute.internal          1/1      Running   0       35m
kube-system                                 etcd-member-ip-10-0-3-239.us-east-
2.compute.internal          1/1      Running   0       37m
kube-system                                 etcd-member-ip-10-0-3-24.us-east-
2.compute.internal            1/1      Running   0       35m
openshift-apiserver-operator                openshift-apiserver-operator-6d6674f4f4-
h7t2t               1/1      Running   1       37m
openshift-apiserver                         apiserver-fm48r
1/1      Running   0       30m
openshift-apiserver                         apiserver-fxkvv
1/1      Running   0       29m
openshift-apiserver                         apiserver-q85nm
1/1      Running   0       29m
...
openshift-service-ca-operator               openshift-service-ca-operator-66ff6dc6cd-
9r257               1/1      Running   0       37m
openshift-service-ca                        apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1      Running   0       35m
openshift-service-ca                        configmap-cabundle-injector-8498544d7-
25qn6               1/1      Running   0       35m
openshift-service-ca                        service-serving-cert-signer-6445fc9c6-wqdqn
1/1      Running   0       35m
openshift-service-catalog-apiserver-operator        openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w     1/1      Running   0       32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm    1/1      Running   0       31m
```

When the current cluster version is **AVAILABLE**, the installation is complete.

1.8.23. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.8.24. Next steps

- Customize your cluster.

- If necessary, you can opt out of remote health reporting .

1.9. INSTALLING A CLUSTER INTO A SHARED VPC ON GCP USING DEPLOYMENT MANAGER TEMPLATES

In OpenShift Container Platform version 4.6, you can install a cluster into a shared Virtual Private Cloud (VPC) on Google Cloud Platform (GCP) that uses infrastructure that you provide. In this context, a cluster installed into a shared VPC is a cluster that is configured to use a VPC from a project different from where the cluster is being deployed.

A shared VPC enables an organization to connect resources from multiple projects to a common VPC network. You can communicate within the organization securely and efficiently by using internal IPs from that network. For more information about shared VPC, see Shared VPC overview in the GCP documentation.

The steps for performing a user-provided infrastructure installation into a shared VPC are outlined here. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods.

> **IMPORTANT**
>
> The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

1.9.1. Prerequisites

- Review details about the OpenShift Container Platform installation and update processes.

- If you use a firewall and plan to use telemetry, you must configure the firewall to allow the sites that your cluster requires access to.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can manually create and maintain IAM credentials . Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

NOTE

Be sure to also review this site list if you are configuring a proxy.

1.9.2. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

1.9.3. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster.

You must have Internet access to:

- Access OpenShift Cluster Manager 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.

1.9.4. Configuring the GCP project that hosts your cluster

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

1.9.4.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See Creating and Managing Projects in the GCP documentation.
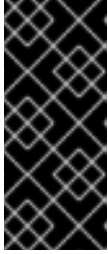
> **IMPORTANT**
>
> Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>. <base_domain>** URL; the Premium Tier is required for internal load balancing.

1.9.4.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See Enabling services in the GCP documentation.

   Table 1.33. Required API services

API service	Console service name
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
Compute Engine API	**compute.googleapis.com**
Google Cloud APIs	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

1.9.4.3. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default Quotas do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 1.34. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall rules	Networking	Global	11	1
Forwarding rules	Compute	Global	2	0
Health checks	Compute	Global	2	0
Images	Compute	Global	1	0
Networks	Networking	Global	1	0
Routers	Networking	Global	1	0
Routes	Networking	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0

> **NOTE**
>
> If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**

- **asia-northeast2**

- **asia-south1**

- **australia-southeast1**

- **europe-north1**

- **europe-west2**

- **europe-west3**

- **europe-west6**

- **northamerica-northeast1**

- **southamerica-east1**

- **us-west2**

You can increase resource quotas from the GCP console, but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

1.9.4.4. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a service account in the project that you use to host your OpenShift Container Platform cluster. See Creating a service account in the GCP documentation.

2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See Granting roles to a service account for specific resources.

> **NOTE**
>
> While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. Create the service account key in JSON format. See Creating service account keys in the GCP documentation.
   The service account key is required to create a cluster.

1.9.4.4.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions. If you

deploy your cluster into an existing VPC, the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin

- Security Admin

- Service Account Admin

- Service Account User

- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor

- Service Account Key Admin

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 1.35. GCP service account permissions

Account	Roles
Control Plane	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	**roles/compute.viewer**
	roles/storage.admin

1.9.4.5. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- **asia-east1** (Changhua County, Taiwan)

- **asia-east2** (Hong Kong)

- **asia-northeast1** (Tokyo, Japan)

- **asia-northeast2** (Osaka, Japan)

- **asia-northeast3** (Seoul, South Korea)

- **asia-south1** (Mumbai, India)

- **asia-southeast1** (Jurong West, Singapore)

- **asia-southeast2** (Jakarta, Indonesia)

- **australia-southeast1** (Sydney, Australia)

- **europe-north1** (Hamina, Finland)

- **europe-west1** (St. Ghislain, Belgium)

- **europe-west2** (London, England, UK)

- **europe-west3** (Frankfurt, Germany)

- **europe-west4** (Eemshaven, Netherlands)

- **europe-west6** (Zürich, Switzerland)

- **northamerica-northeast1** (Montréal, Québec, Canada)

- **southamerica-east1** (São Paulo, Brazil)

- **us-central1** (Council Bluffs, Iowa, USA)

- **us-east1** (Moncks Corner, South Carolina, USA)

- **us-east4** (Ashburn, Northern Virginia, USA)

- **us-west1** (The Dalles, Oregon, USA)

- **us-west2** (Los Angeles, California, USA)

- **us-west3** (Salt Lake City, Utah, USA)

- **us-west4** (Las Vegas, Nevada, USA)

1.9.4.6. Installing and configuring CLI tools for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must install and configure the CLI tools for GCP.

Prerequisites

- You created a project to host your cluster.

- You created a service account and granted it the required permissions.

Procedure

1. Install the following binaries in **$PATH**:

   - **gcloud**

   - **gsutil**

   See Install the latest Cloud SDK version in the GCP documentation.

2. Authenticate using the **gcloud** tool with your configured service account.
   See Authorizing with a service account in the GCP documentation.

1.9.5. Configuring the GCP project that hosts your shared VPC network

If you use a shared Virtual Private Cloud (VPC) to host your OpenShift Container Platform cluster in Google Cloud Platform (GCP), you must configure the project that hosts it.
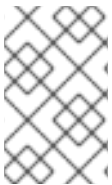
> **NOTE**
>
> If you already have a project that hosts the shared VPC network, review this section to ensure that the project meets all of the requirements to install an OpenShift Container Platform cluster.

Procedure

1. Create a project to host the shared VPC for your OpenShift Container Platform cluster. See Creating and Managing Projects in the GCP documentation.

2. Create a service account in the project that hosts your shared VPC. See Creating a service account in the GCP documentation.

3. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See Granting roles to a service account for specific resources.

NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

The service account for the project that hosts the shared VPC network requires the following roles:

- Compute Network User

- Compute Security Admin

- Deployment Manager Editor

- DNS Administrator

- Security Admin

- Network Management Admin

1.9.5.1. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the project that hosts the shared VPC that you install the cluster into. This zone must be authoritative for the domain. The DNS service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through GCP or another source.

   NOTE

   If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see Google Domains.

2. Create a public hosted zone for your domain or subdomain in your GCP project. See Creating public zones in the GCP documentation.
   Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.

3. Extract the new authoritative name servers from the hosted zone records. See Look up your Cloud DNS name servers in the GCP documentation.
   You typically have four name servers.

4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: How to switch to custom name servers .

5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See Migrating to Cloud DNS in the GCP documentation.

6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

1.9.5.2. Creating a VPC in GCP

You must create a VPC in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements. One way to create the VPC is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

Procedure

1. Copy the template from the **Deployment Manager template for the VPC**section of this topic and save it as **01_vpc.py** on your computer. This template describes the VPC that your cluster requires.

2. Export the following variables required by the resource definition:

   a. Export the control plane CIDR:

   ```
   $ export MASTER_SUBNET_CIDR='10.0.0.0/19'
   ```

   b. Export the compute CIDR:

   ```
   $ export WORKER_SUBNET_CIDR='10.0.32.0/19'
   ```

   c. Export the region to deploy the VPC network and cluster to:

   ```
   $ export REGION='<region>'
   ```

3. Export the variable for the ID of the project that hosts the shared VPC:

   ```
   $ export HOST_PROJECT=<host_project>
   ```

4. Export the variable for the email of the service account that belongs to host project:

   ```
   $ export HOST_PROJECT_ACCOUNT=<host_service_account_email>
   ```

5. Create a **01_vpc.yaml** resource definition file:

   ```
   $ cat <<EOF >01_vpc.yaml
   imports:
   ```

```
- path: 01_vpc.py

  resources:
  - name: cluster-vpc
    type: 01_vpc.py
    properties:
      infra_id: '<prefix>'  1
      region: '${REGION}'  2
      master_subnet_cidr: '${MASTER_SUBNET_CIDR}'  3
      worker_subnet_cidr: '${WORKER_SUBNET_CIDR}'  4
EOF
```

1   **infra_id** is the prefix of the network name.

2   **region** is the region to deploy the cluster into, for example **us-central1**.

3   **master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/19**.

4   **worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.32.0/19**.

6. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create <vpc_deployment_name> --config
01_vpc.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}  1
```

1   For **<vpc_deployment_name>**, specify the name of the VPC to deploy.

7. Export the VPC variable that other components require:

   a. Export the name of the host project network:

   ```
   $ export HOST_PROJECT_NETWORK=<vpc_network>
   ```

   b. Export the name of the host project control plane subnet:

   ```
   $ export HOST_PROJECT_CONTROL_SUBNET=<control_plane_subnet>
   ```

   c. Export the name of the host project compute subnet:

   ```
   $ export HOST_PROJECT_COMPUTE_SUBNET=<compute_subnet>
   ```

8. Set up the shared VPC. See Setting up Shared VPC in the GCP documentation.

1.9.5.2.1. Deployment Manager template for the VPC

You can use the following Deployment Manager template to deploy the VPC that you need for your OpenShift Container Platform cluster:

Example 1.10. 01_vpc.py Deployment Manager template

```
def GenerateConfig(context):
```

```
    resources = [{
      'name': context.properties['infra_id'] + '-network',
      'type': 'compute.v1.network',
      'properties': {
        'region': context.properties['region'],
        'autoCreateSubnetworks': False
      }
    }, {
      'name': context.properties['infra_id'] + '-master-subnet',
      'type': 'compute.v1.subnetwork',
      'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'ipCidrRange': context.properties['master_subnet_cidr']
      }
    }, {
      'name': context.properties['infra_id'] + '-worker-subnet',
      'type': 'compute.v1.subnetwork',
      'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'ipCidrRange': context.properties['worker_subnet_cidr']
      }
    }, {
      'name': context.properties['infra_id'] + '-router',
      'type': 'compute.v1.router',
      'properties': {
        'region': context.properties['region'],
        'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
        'nats': [{
          'name': context.properties['infra_id'] + '-nat-master',
          'natIpAllocateOption': 'AUTO_ONLY',
          'minPortsPerVm': 7168,
          'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
          'subnetworks': [{
            'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
            'sourceIpRangesToNat': ['ALL_IP_RANGES']
          }]
        }, {
          'name': context.properties['infra_id'] + '-nat-worker',
          'natIpAllocateOption': 'AUTO_ONLY',
          'minPortsPerVm': 512,
          'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
          'subnetworks': [{
            'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
            'sourceIpRangesToNat': ['ALL_IP_RANGES']
          }]
        }]
      }
    }]

    return {'resources': resources}
```

1.9.6. Creating the installation files for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files. You also have the option to first set up a separate **var** partition during the preparation phases of installation.

1.9.6.1. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

   ```
   $ mkdir <installation_directory>
   ```

   > **IMPORTANT**
   >
   > You must create a 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.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.

   > **NOTE**
   >
   > You must name this configuration file **install-config.yaml**.

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 next step of the installation process. You must back it up now.

1.9.6.2. Sample customized install-config.yaml file for GCP

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

> **IMPORTANT**
>
> This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```
apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: 5
- hyperthreading: Enabled 6
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 0
metadata:
  name: test-cluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 7
    region: us-central1 8
pullSecret: '{"auths": ...}'
fips: false 9
sshKey: ssh-ed25519 AAAA... 10
publish: Internal 11
```

1 Specify the public DNS on the host project.

2 5 If you do not provide these parameters and values, the installation program provides the default value.

3 6 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both

must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

4   Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.

> **IMPORTANT**
>
> If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

7   Specify the main project where the VM instances reside.

8   Specify the region that your VPC network is in.

9   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.

> **IMPORTANT**
>
> The use of FIPS Validated / Modules in Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64** architecture.

10   You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

> **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.

11   How to publish the user-facing endpoints of your cluster. Set **publish** to **Internal** to deploy a private cluster, which cannot be accessed from the Internet. The default value is **External**. To use a shared VPC in a cluster that uses infrastructure that you provision, you must set **publish** to **Internal**. The installation program will no longer be able to access the public DNS zone for the base domain in the host project.

1.9.6.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

> **NOTE**
>
> The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.
>
> For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

   ```
   apiVersion: v1
   baseDomain: my.domain.com
   proxy:
     httpProxy: http://<username>:<pswd>@<ip>:<port> 1
     httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
     noProxy: example.com 3
   additionalTrustBundle: | 4
       -----BEGIN CERTIFICATE-----
       <MY_TRUSTED_CA_CERT>
       -----END CERTIFICATE-----
   ...
   ```

   1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

   2 A proxy URL to use for creating HTTPS connections outside the cluster.

   3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations.

   4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace to hold the additional CA certificates. If you provide **additionalTrustBundle** and at least one proxy setting, the **Proxy** object is configured to reference the **user-ca-bundle** config map in the **trustedCA** field. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges the contents specified for the **trustedCA** parameter with the RHCOS trust bundle. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

> **NOTE**
>
> The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

> **NOTE**
>
> Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

1.9.6.4. 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.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to create the cluster.

> **IMPORTANT**
>
> - The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
>
> - It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

Prerequisites

- You obtained the OpenShift Container Platform installation program.

- You created the **install-config.yaml** installation configuration file.

Procedure

1. Change to the directory that contains the installation program and generate the Kubernetes manifests for the cluster:

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

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

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

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:

   a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.

   b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.

   c. Save and exit the file.

5. Remove the **privateZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
status: {}
```

1   Remove this section completely.

6. Configure the cloud provider for your VPC.

   a. Open the **<installation_directory>/manifests/cloud-provider-config.yaml** file.

   b. Add the **network-project-id** parameter and set its value to the ID of project that hosts the shared VPC network.

   c. Add the **network-name** parameter and set its value to the name of the shared VPC network that hosts the OpenShift Container Platform cluster.

   d. Replace the value of the **subnetwork-name** parameter with the value of the shared VPC subnet that hosts your compute machines.

The contents of the **<installation_directory>/manifests/cloud-provider-config.yaml** resemble the following example:

```
config: |+
  [global]
  project-id      = example-project
  regional        = true
  multizone       = true
  node-tags       = opensh-ptzzx-master
  node-tags       = opensh-ptzzx-worker
  node-instance-prefix = opensh-ptzzx
  external-instance-groups-prefix = opensh-ptzzx
  network-project-id = example-shared-vpc
  network-name    = example-network
  subnetwork-name = example-worker-subnet
```

7. If you deploy a cluster that is not on a private network, open the **<installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml** file and replace the value of the **scope** parameter with **External**. The contents of the file resemble the following example:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
    type: LoadBalancerService
status:
  availableReplicas: 0
  domain: ''
  selector: ''
```

8. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

```
$ ./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
```

1.9.7. Exporting common variables

1.9.7.1. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The infrastructure name is also used to locate the appropriate GCP resources during an OpenShift Container Platform installation. The provided Deployment Manager templates contain references to this infrastructure name, so you must extract it.

Prerequisites

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

- You generated the Ignition config files for your cluster.

- You installed the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

  ```
  $ jq -r .infraID <installation_directory>/metadata.json ❶
  ```

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

  Example output

  ```
  openshift-vw9j6 ❶
  ```

  ❶   The output of this command is your cluster name and a random string.

1.9.7.2. Exporting common variables for Deployment Manager templates

You must export a common set of variables that are used with the provided Deployment Manager templates used to assist in completing a user-provided infrastructure install on Google Cloud Platform (GCP).

> **NOTE**
>
> Specific Deployment Manager templates can also require additional exported variables, which are detailed in their related procedures.

Prerequisites

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

- Generate the Ignition config files for your cluster.

- Install the **jq** package.

Procedure

1. Export the following common variables to be used by the provided Deployment Manager templates:

```
$ export BASE_DOMAIN='<base_domain>'  1
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'  2
$ export NETWORK_CIDR='10.0.0.0/16'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig  3
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
```

1 **2** Supply the values for the host project.

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

1.9.8. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 1.36. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	**1936**	Metrics
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101** and the Cluster Version Operator on port**9099**.
	10250-**10259**	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	**4789**	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101**.

Protocol	Port	Description
TCP/UDP	**30000**-**32767**	Kubernetes node port

Table 1.37. All machines to control plane

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 1.38. Control plane machines to control plane machines

Protocol	Port	Description
TCP	**2379**-**2380**	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.

> **IMPORTANT**
>
> OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer**: Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.

   - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

   > **IMPORTANT**
   >
   > Do not configure session persistence for an API load balancer.

   Configure the following ports on both the front and back of the load balancers:

   ### Table 1.39. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the **/readyz** endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

> **NOTE**
>
> The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer**: Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

   - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 1.40. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

> **NOTE**
>
> A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

1.9.9. Creating load balancers in GCP

You must configure load balancers in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.

2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.

3. Export the variables that the deployment template uses:

    a. Export the cluster network location:

    ```
    $ export CLUSTER_NETWORK=(`gcloud compute networks describe
    ${HOST_PROJECT_NETWORK} --project ${HOST_PROJECT} --account
    ${HOST_PROJECT_ACCOUNT} --format json | jq -r .selfLink`)
    ```

    b. Export the control plane subnet location:

    ```
    $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe
    ${HOST_PROJECT_CONTROL_SUBNET} --region=${REGION} --project
    ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
    .selfLink`)
    ```

c. Export the three zones that the cluster uses:

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[2] | cut -d "/" -f9`)
```

4. Create a **02_infra.yaml** resource definition file:

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py 1
resources:
- name: cluster-lb-ext 2
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' 3
    region: '${REGION}' 4
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' 5
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: 6
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

1 2 Required only when deploying an external cluster.

3 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

4 **region** is the region to deploy the cluster into, for example **us-central1**.

5 **control_subnet** is the URI to the control subnet.

6 **zones** are the zones to deploy the control plane instances into, like **us-east1-b**, **us-east1-c**, and **us-east1-d**.

5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. Export the cluster IP address:

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --
region=${REGION} --format json | jq -r .address`)
```

7. For an external cluster, also export the cluster public IP address:

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-
public-ip --region=${REGION} --format json | jq -r .address`)
```

1.9.9.1. Deployment Manager template for the external load balancer

You can use the following Deployment Manager template to deploy the external load balancer that you need for your OpenShift Container Platform cluster:

Example 1.11. 02_lb_ext.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

1.9.9.2. Deployment Manager template for the internal load balancer

You can use the following Deployment Manager template to deploy the internal load balancer that you need for your OpenShift Container Platform cluster:

Example 1.12. **02_lb_int.py** Deployment Manager template

```python
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' + '.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
```

```
                'ports': ['6443','22623'],
                'region': context.properties['region'],
                'subnetwork': context.properties['control_subnet']
            }
        }]

        for zone in context.properties['zones']:
            resources.append({
                'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
                'type': 'compute.v1.instanceGroup',
                'properties': {
                    'namedPorts': [
                        {
                            'name': 'ignition',
                            'port': 22623
                        }, {
                            'name': 'https',
                            'port': 6443
                        }
                    ],
                    'network': context.properties['cluster_network'],
                    'zone': zone
                }
            })

        return {'resources': resources}
```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

1.9.10. Creating a private DNS zone in GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.

2. Create a **02_dns.yaml** resource definition file:

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF
```

1   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

2   **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.

3   **cluster_network** is the **selfLink** URL to the cluster network.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

4. The templates do not create DNS entries due to limitations of Deployment Manager, so you must create them manually:

   a. Add the internal DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

   b. For an external cluster, also add the external DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
```

```
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

1.9.10.1. Deployment Manager template for the private DNS

You can use the following Deployment Manager template to deploy the private DNS that you need for your OpenShift Container Platform cluster:

Example 1.13. **02_dns.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

1.9.11. Creating firewall rules in GCP

You must create firewall rules in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.

2. Create a **03_firewall.yaml** resource definition file:

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' 1
    infra_id: '${INFRA_ID}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
    network_cidr: '${NETWORK_CIDR}' 4
EOF
```

1   **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **${NETWORK_CIDR}**.

2   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3   **cluster_network** is the **selfLink** URL to the cluster network.

4   **network_cidr** is the CIDR of the VPC network, for example **10.0.0.0/16**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

1.9.11.1. Deployment Manager template for firewall rules

You can use the following Deployment Manager template to deploy the firewall rues that you need for your OpenShift Container Platform cluster:

Example 1.14. 03_firewall.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
```

```
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
```

```
            context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [context.properties['infra_id'] + '-master']
    }
}, {
    'name': context.properties['infra_id'] + '-internal-network',
    'type': 'compute.v1.firewall',
    'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
            'IPProtocol': 'icmp'
        },{
            'IPProtocol': 'tcp',
            'ports': ['22']
        }],
        'sourceRanges': [context.properties['network_cidr']],
        'targetTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ]
    }
}, {
    'name': context.properties['infra_id'] + '-internal-cluster',
    'type': 'compute.v1.firewall',
    'properties': {
        'network': context.properties['cluster_network'],
        'allowed': [{
            'IPProtocol': 'udp',
            'ports': ['4789', '6081']
        },{
            'IPProtocol': 'tcp',
            'ports': ['9000-9999']
        },{
            'IPProtocol': 'udp',
            'ports': ['9000-9999']
        },{
            'IPProtocol': 'tcp',
            'ports': ['10250']
        },{
            'IPProtocol': 'tcp',
            'ports': ['30000-32767']
        },{
            'IPProtocol': 'udp',
            'ports': ['30000-32767']
        }],
        'sourceTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ]
    }
}
```

```
                    }]

                    return {'resources': resources}
```

1.9.12. Creating IAM roles in GCP

You must create IAM roles in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.

2. Create a **03_iam.yaml** resource definition file:

   ```
   $ cat <<EOF >03_iam.yaml
   imports:
   - path: 03_iam.py
   resources:
   - name: cluster-iam
     type: 03_iam.py
     properties:
       infra_id: '${INFRA_ID}' 1
   EOF
   ```

   1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3. Create the deployment by using the **gcloud** CLI:

   ```
   $ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
   ```

4. Export the variable for the master service account:

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. Export the variable for the worker service account:

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. Assign the permissions that the installation program requires to the service accounts for the subnets that host the control plane and compute subnets:

   a. Grant the **networkViewer** role of the project that hosts your shared VPC to the master service account:

   ```
   $ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
   projects add-iam-policy-binding ${HOST_PROJECT} --member
   "serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role
   "roles/compute.networkViewer"
   ```

   b. Grant the **networkUser** role to the master service account for the control plane subnet:

   ```
   $ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
   compute networks subnets add-iam-policy-binding
   "${HOST_PROJECT_CONTROL_SUBNET}" --member
   "serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
   --region ${REGION}
   ```

   c. Grant the **networkUser** role to the worker service account for the control plane subnet:

   ```
   $ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
   compute networks subnets add-iam-policy-binding
   "${HOST_PROJECT_CONTROL_SUBNET}" --member
   "serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
   "roles/compute.networkUser" --region ${REGION}
   ```

   d. Grant the **networkUser** role to the master service account for the compute subnet:

   ```
   $ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
   compute networks subnets add-iam-policy-binding
   "${HOST_PROJECT_COMPUTE_SUBNET}" --member
   "serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
   --region ${REGION}
   ```

   e. Grant the **networkUser** role to the worker service account for the compute subnet:

   ```
   $ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
   compute networks subnets add-iam-policy-binding
   "${HOST_PROJECT_COMPUTE_SUBNET}" --member
   "serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
   "roles/compute.networkUser" --region ${REGION}
   ```

7. The templates do not create the policy bindings due to limitations of Deployment Manager, so you must create them manually:

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. Create a service account key and store it locally for later use:

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

1.9.12.1. Deployment Manager template for IAM roles

You can use the following Deployment Manager template to deploy the IAM roles that you need for your OpenShift Container Platform cluster:

Example 1.15. **03_iam.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

1.9.13. Creating the RHCOS cluster image for the GCP infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) image for Google Cloud Platform (GCP) for your OpenShift Container Platform nodes.

Procedure

1. Obtain the RHCOS image from the RHCOS image mirror page.

> **IMPORTANT**
>
> The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

The file name contains the OpenShift Container Platform version number in the format **rhcos-<version>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

   ```
   $ gsutil mb gs://<bucket_name>
   ```

3. Upload the RHCOS image to the Google storage bucket:

   ```
   $ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
   gs://<bucket_name>
   ```

4. Export the uploaded RHCOS image location as a variable:

   ```
   $ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz`
   ```

5. Create the cluster image:

   ```
   $ gcloud compute images create "${INFRA_ID}-rhcos-image" \
       --source-uri="${IMAGE_SOURCE}"
   ```

1.9.14. Creating the bootstrap machine in GCP

You must create the bootstrap machine in Google Cloud Platform (GCP) to use during OpenShift Container Platform cluster initialization. One way to create this machine is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your bootstrap machine, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Ensure pyOpenSSL is installed.

Procedure

1. Copy the template from the **Deployment Manager template for the bootstrap machine** section of this topic and save it as **04_bootstrap.py** on your computer. This template describes the bootstrap machine that your cluster requires.

2. Export the location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

   ```
   $ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
   ```

3. Create a bucket and upload the **bootstrap.ign** file:

   ```
   $ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
   $ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
   ```

4. Create a signed URL for the bootstrap instance to use to access the Ignition config. Export the URL from the output as a variable:

   ```
   $ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
   ```

5. Create a **04_bootstrap.yaml** resource definition file:

   ```
   $ cat <<EOF >04_bootstrap.yaml
   imports:
   - path: 04_bootstrap.py

   resources:
   - name: cluster-bootstrap
     type: 04_bootstrap.py
     properties:
       infra_id: '${INFRA_ID}'          1
       region: '${REGION}'              2
       zone: '${ZONE_0}'               3

       cluster_network: '${CLUSTER_NETWORK}'    4
       control_subnet: '${CONTROL_SUBNET}'      5
       image: '${CLUSTER_IMAGE}'        6
       machine_type: 'n1-standard-4'    7
       root_volume_size: '128'          8

       bootstrap_ign: '${BOOTSTRAP_IGN}'    9
   EOF
   ```

1. **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

2. **region** is the region to deploy the cluster into, for example **us-central1**.

3. **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-b**.

4. **cluster_network** is the **selfLink** URL to the cluster network.

5. **control_subnet** is the **selfLink** URL to the control subnet.

6. **image** is the **selfLink** URL to the RHCOS image.

7. **machine_type** is the machine type of the instance, for example **n1-standard-4**.

8. **root_volume_size** is the boot disk size for the bootstrap machine.

9. **bootstrap_ign** is the URL output when creating a signed URL.

6. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Add the bootstrap instance to the internal load balancer instance group:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-bootstrap-
instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

8. Add the bootstrap instance group to the internal load balancer backend service:

```
$ gcloud compute backend-services add-backend ${INFRA_ID}-api-internal-backend-service
--region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --instance-
group-zone=${ZONE_0}
```

1.9.14.1. Deployment Manager template for the bootstrap machine

You can use the following Deployment Manager template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster:

Example 1.16. **04_bootstrap.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
```

```
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

1.9.15. Creating the control plane machines in GCP

You must create the control plane machines in Google Cloud Platform (GCP) for your cluster to use.
One way to create these machines is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your control plane machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

Procedure

1. Copy the template from the **Deployment Manager template for control plane machines** section of this topic and save it as **05_control_plane.py** on your computer. This template describes the control plane machines that your cluster requires.

2. Export the following variable required by the resource definition:

   ```
   $ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
   ```

3. Create a **05_control_plane.yaml** resource definition file:

   ```
   $ cat <<EOF >05_control_plane.yaml
   imports:
   - path: 05_control_plane.py

   resources:
   - name: cluster-control-plane
     type: 05_control_plane.py
     properties:
       infra_id: '${INFRA_ID}' 1
       zones: 2
       - '${ZONE_0}'
       - '${ZONE_1}'
       - '${ZONE_2}'

       control_subnet: '${CONTROL_SUBNET}' 3
       image: '${CLUSTER_IMAGE}' 4
       machine_type: 'n1-standard-4' 5
       root_volume_size: '128'
       service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

       ignition: '${MASTER_IGNITION}' 7
   EOF
   ```

1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

2 **zones** are the zones to deploy the control plane instances into, for example   **us-central1-a**,
us-central1-b, and **us-central1-c**.

3 **control_subnet** is the **selfLink** URL to the control subnet.

4 **image** is the **selfLink** URL to the RHCOS image.

5 **machine_type** is the machine type of the instance, for example **n1-standard-4**.

6 **service_account_email** is the email address for the master service account that you
created.

7 **ignition** is the contents of the **master.ign** file.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. The templates do not manage load balancer membership due to limitations of Deployment
Manager, so you must add the control plane machines manually.

- Run the following commands to add the control plane machines to the appropriate instance
groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- For an external cluster, you must also run the following commands to add the control plane
machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

1.9.15.1. Deployment Manager template for control plane machines

You can use the following Deployment Manager template to deploy the control plane machines that you
need for your OpenShift Container Platform cluster:

Example 1.17. 05_control_plane.py Deployment Manager template

```
def GenerateConfig(context):
```

```
resources = [{
    'name': context.properties['infra_id'] + '-master-0',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][0]
    }
}, {
    'name': context.properties['infra_id'] + '-master-1',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
```

```
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-master-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
    }]

    return {'resources': resources}
```

1.9.16. Wait for bootstrap completion and remove bootstrap resources in GCP

After you create all of the required infrastructure in Google Cloud Platform (GCP), wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

   ```
   $ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ ❶
       --log-level info ❷
   ```

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

   ❷ To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

   If the command exits without a **FATAL** warning, your production control plane has initialized.

2. Delete the bootstrap resources:

   ```
   $ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
   $ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
   $ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
   $ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
   ```

1.9.17. Creating additional worker machines in GCP

You can create worker machines in Google Cloud Platform (GCP) for your cluster to use by launching individual instances discretely or by automated processes outside the cluster, such as auto scaling groups. You can also take advantage of the built-in cluster scaling mechanisms and the machine API in OpenShift Container Platform.

In this example, you manually launch one instance by using the Deployment Manager template. Additional instances can be launched by including additional resources of type **06_worker.py** in the file.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your worker machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

Procedure

1. Copy the template from the **Deployment Manager template for worker machines** section of this topic and save it as **06_worker.py** on your computer. This template describes the worker machines that your cluster requires.

2. Export the variables that the resource definition uses.

   a. Export the subnet that hosts the compute machines:

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${HOST_PROJECT_COMPUTE_SUBNET} --region=${REGION} --project
   ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
   .selfLink`)
   ```

   b. Export the email address for your service account:

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

   c. Export the location of the compute machine Ignition config file:

   ```
   $ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
   ```

3. Create a **06_worker.yaml** resource definition file:

   ```
   $ cat <<EOF >06_worker.yaml
   imports:
   - path: 06_worker.py

   resources:
   - name: 'worker-0'
   ```
   1

```
      type: 06_worker.py
      properties:
        infra_id: '${INFRA_ID}' 2
        zone: '${ZONE_0}' 3
        compute_subnet: '${COMPUTE_SUBNET}' 4
        image: '${CLUSTER_IMAGE}' 5
        machine_type: 'n1-standard-4' 6
        root_volume_size: '128'
        service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
        ignition: '${WORKER_IGNITION}' 8
    - name: 'worker-1'
      type: 06_worker.py
      properties:
        infra_id: '${INFRA_ID}' 9
        zone: '${ZONE_1}' 10
        compute_subnet: '${COMPUTE_SUBNET}' 11
        image: '${CLUSTER_IMAGE}' 12
        machine_type: 'n1-standard-4' 13
        root_volume_size: '128'
        service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
        ignition: '${WORKER_IGNITION}' 15
  EOF
```

[1] **name** is the name of the worker machine, for example **worker-0**.

[2] [9] **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

[3] [10] **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.

[4] [11] **compute_subnet** is the **selfLink** URL to the compute subnet.

[5] [12] **image** is the **selfLink** URL to the RHCOS image.

[6] [13] **machine_type** is the machine type of the instance, for example **n1-standard-4**.

[7] [14] **service_account_email** is the email address for the worker service account that you created.

[8] [15] **ignition** is the contents of the **worker.ign** file.

4. Optional: If you want to launch additional instances, include additional resources of type **06_worker.py** in your **06_worker.yaml** resource definition file.

5. Create the deployment by using the **gcloud** CLI:

   ```
   $ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
   06_worker.yaml
   ```

1.9.17.1. Deployment Manager template for worker machines

You can use the following Deployment Manager template to deploy the worker machines that you need for your OpenShift Container Platform cluster:

Example 1.18. **06_worker.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

1.9.18. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.9.18.1. Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Linux Client** entry and save the file.

4. Unpack the archive:

   ```
   $ tar xvzf <file>
   ```

5. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.9.18.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 Windows Client** entry and save the file.

4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

   ```
   C:\> path
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

1.9.18.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.

2. Select the appropriate version in the **Version** drop-down menu.

3. Click **Download Now** next to the **OpenShift v4.6 MacOSX Client** entry and save the file.

4. Unpack and unzip the archive.

5. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.9.19. Logging in to the cluster by using the CLI

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

- You deployed an OpenShift Container Platform cluster.

- You installed the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

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

   ❶ 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
   ```

   Example output

   ```
   system:admin
   ```

1.9.20. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME      STATUS   ROLES   AGE  VERSION
   master-0  Ready    master  63m  v1.19.0
   master-1  Ready    master  63m  v1.19.0
   master-2  Ready    master  64m  v1.19.0
   ```

   The output lists all of the machines that you created.

   > **NOTE**
   >
   > The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

   ```
   NAME       AGE   REQUESTOR                                            CONDITION
   csr-8b2br  15m   system:serviceaccount:openshift-machine-config-operator:node-
   bootstrapper  Pending
   csr-8vnps  15m   system:serviceaccount:openshift-machine-config-operator:node-
   bootstrapper  Pending
   ...
   ```

   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. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

> **NOTE**
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- 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 --no-run-if-empty oc adm certificate approve
  ```

  > **NOTE**
  >
  > Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

   ```
   NAME        AGE     REQUESTOR                                          CONDITION
   csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
   Pending
   csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
   Pending
   ...
   ```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

   - 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
     ```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME      STATUS   ROLES   AGE  VERSION
   master-0  Ready    master  73m  v1.20.0
   master-1  Ready    master  73m  v1.20.0
   master-2  Ready    master  74m  v1.20.0
   worker-0  Ready    worker  11m  v1.20.0
   worker-1  Ready    worker  11m  v1.20.0
   ```

   > **NOTE**
   >
   > It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests .

1.9.21. Adding the ingress DNS records

DNS zone configuration is removed when creating Kubernetes manifests and generating Ignition configs. You must manually create DNS records that point at the ingress load balancer. You can create either a wildcard ***.apps.{baseDomain}.** or specific records. You can use A, CNAME, and other records per your requirements.

Prerequisites

- Configure a GCP account.

- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

- Create the worker machines.

Procedure

1. Wait for the Ingress router to create a load balancer and populate the **EXTERNAL-IP** field:

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   Example output

   ```
   NAME          TYPE         CLUSTER-IP     EXTERNAL-IP     PORT(S)              AGE
   router-default   LoadBalancer   172.30.18.154   35.233.157.184
   80:32288/TCP,443:31215/TCP   98
   ```

2. Add the A record to your zones:

   - To use A records:

     i. Export the variable for the router IP address:

        ```
        $ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
        headers | awk '{print $4}'`
        ```

     ii. Add the A record to the private zones:

        ```
        $ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
        $ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
        ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
        $ gcloud dns record-sets transaction add ${ROUTER_IP} --name
        \*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
        ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account
        ${HOST_PROJECT_ACCOUNT}
        $ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --
        project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
        ```

     iii. For an external cluster, also add the A record to the public zones:

        ```
        $ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
        $ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME} -
        -project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
        $ gcloud dns record-sets transaction add ${ROUTER_IP} --name
        \*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
        ${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
        ${HOST_PROJECT_ACCOUNT}
        $ gcloud dns record-sets transaction execute --zone
        ${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
        ${HOST_PROJECT_ACCOUNT}
        ```

- To add explicit domains instead of using a wildcard, create entries for each of the cluster's current routes:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

Example output

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

1.9.22. Adding ingress firewall rules

The cluster requires several firewall rules. If you do not use a shared VPC, these rules are created by the ingress controller via the GCP cloud provider. When you use a shared VPC, you can either create cluster-wide firewall rules for all services now or create each rule based on events, when the cluster requests access. By creating each rule when the cluster requests access, you know exactly which firewall rules are required. By creating cluster-wide firewall rules, you can apply the same rule set across multiple clusters.

If you choose to create each rule based on events, you must create firewall rules after you provision the cluster and during the life of the cluster when the console notifies you that rules are missing. Events that are similar to the following event are displayed, and you must add the firewall rules that are required:

```
$ oc get events -n openshift-ingress --field-selector="reason=LoadBalancerManualChange"
```

Example output

```
Firewall change required by security admin: `gcloud compute firewall-rules create k8s-fw-
a26e631036a3f46cba28f8df67266d55 --network example-network --description "
{\"kubernetes.io/service-name\":\"openshift-ingress/router-default\", \"kubernetes.io/service-
ip\":\"35.237.236.234\"}\" --allow tcp:443,tcp:80 --source-ranges 0.0.0.0/0 --target-tags exampl-fqzq7-
master,exampl-fqzq7-worker --project example-project`
```

If you encounter issues when creating these rule-based events, you can configure the cluster-wide firewall rules while your cluster is running.

1.9.22.1. Creating cluster-wide firewall rules for a shared VPC in GCP

You can create cluster-wide firewall rules to allow the access that the OpenShift Container Platform cluster requires.

> **WARNING**
>
> If you do not choose to create firewall rules based on cluster events, you must create cluster-wide firewall rules.

Prerequisites

- You exported the variables that the Deployment Manager templates require to deploy your cluster.

- You created the networking and load balancing components in GCP that your cluster requires.

Procedure

1. Add a single firewall rule to allow the Google Cloud Engine health checks to access all of the services. This rule enables the ingress load balancers to determine the health status of their instances.

   ```
   $ gcloud compute firewall-rules create --allow='tcp:30000-32767,udp:30000-32767' --
   network="${CLUSTER_NETWORK}" --source-
   ranges='130.211.0.0/22,35.191.0.0/16,209.85.152.0/22,209.85.204.0/22' --target-
   tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress-hc --
   account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
   ```

2. Add a single firewall rule to allow access to all cluster services:

   - For an external cluster:

     ```
     $ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
     network="${CLUSTER_NETWORK}" --source-ranges="0.0.0.0/0" --target-
     tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --
     account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
     ```

   - For a private cluster:

     ```
     $ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
     network="${CLUSTER_NETWORK}" --source-ranges=${NETWORK_CIDR} --target-
     tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --
     account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
     ```

   Because this rule only allows traffic on TCP ports **80** and **443**, ensure that you add all the ports that your services use.

1.9.23. Completing a GCP installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Google Cloud Platform (GCP) user-provisioned infrastructure, you can monitor the cluster events until the cluster is ready.

Prerequisites

- Deploy the bootstrap machine for an OpenShift Container Platform cluster on user-provisioned GCP infrastructure.

- Install the **oc** CLI and log in.

Procedure

1. Complete the cluster installation:

   ```
   $ ./openshift-install --dir <installation_directory> wait-for install-complete ❶
   ```

   Example output

   ```
   INFO Waiting up to 30m0s for the cluster to initialize...
   ```

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

   > **IMPORTANT**
   >
   > - The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
   >
   > - It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Observe the running state of your cluster.

   a. Run the following command to view the current cluster version and status:

   ```
   $ oc get clusterversion
   ```

   Example output

   ```
   NAME      VERSION  AVAILABLE  PROGRESSING  SINCE  STATUS
   version            False      True         24m    Working towards 4.5.4: 99% complete
   ```

   b. Run the following command to view the Operators managed on the control plane by the Cluster Version Operator (CVO):

   ```
   $ oc get clusteroperators
   ```

   Example output

```
NAME                            VERSION  AVAILABLE  PROGRESSING  DEGRADED
SINCE
authentication                  4.5.4    True       False        False     7m56s
cloud-credential                4.5.4    True       False        False     31m
cluster-autoscaler              4.5.4    True       False        False     16m
console                         4.5.4    True       False        False     10m
csi-snapshot-controller         4.5.4    True       False        False     16m
dns                             4.5.4    True       False        False     22m
etcd                            4.5.4    False      False        False     25s
image-registry                  4.5.4    True       False        False     16m
ingress                         4.5.4    True       False        False     16m
insights                        4.5.4    True       False        False     17m
kube-apiserver                  4.5.4    True       False        False     19m
kube-controller-manager         4.5.4    True       False        False     20m
kube-scheduler                  4.5.4    True       False        False     20m
kube-storage-version-migrator   4.5.4    True       False        False     16m
machine-api                     4.5.4    True       False        False     22m
machine-config                  4.5.4    True       False        False     22m
marketplace                     4.5.4    True       False        False     16m
monitoring                      4.5.4    True       False        False     10m
network                         4.5.4    True       False        False     23m
node-tuning                     4.5.4    True       False        False     23m
openshift-apiserver             4.5.4    True       False        False     17m
openshift-controller-manager    4.5.4    True       False        False     15m
openshift-samples               4.5.4    True       False        False     16m
operator-lifecycle-manager      4.5.4    True       False        False     22m
operator-lifecycle-manager-catalog  4.5.4  True     False        False     22m
operator-lifecycle-manager-packageserver 4.5.4 True False        False     18m
service-ca                      4.5.4    True       False        False     23m
service-catalog-apiserver       4.5.4    True       False        False     23m
service-catalog-controller-manager  4.5.4  True     False        False     23m
storage                         4.5.4    True       False        False     17m
```

c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

```
NAMESPACE                         NAME
READY    STATUS    RESTARTS  AGE
kube-system                       etcd-member-ip-10-0-3-111.us-east-
2.compute.internal       1/1     Running   0       35m
kube-system                       etcd-member-ip-10-0-3-239.us-east-
2.compute.internal       1/1     Running   0       37m
kube-system                       etcd-member-ip-10-0-3-24.us-east-
2.compute.internal       1/1     Running   0       35m
openshift-apiserver-operator      openshift-apiserver-operator-6d6674f4f4-
h7t2t            1/1     Running   1       37m
openshift-apiserver               apiserver-fm48r
1/1      Running   0       30m
openshift-apiserver               apiserver-fxkvv
1/1      Running   0       29m
openshift-apiserver               apiserver-q85nm
1/1      Running   0       29m
```

```
...
openshift-service-ca-operator                    openshift-service-ca-operator-66ff6dc6cd-
9r257                  1/1      Running    0      37m
openshift-service-ca                             apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1      Running    0       35m
openshift-service-ca                             configmap-cabundle-injector-8498544d7-
25qn6                  1/1      Running    0      35m
openshift-service-ca                             service-serving-cert-signer-6445fc9c6-wqdqn
1/1      Running    0       35m
openshift-service-catalog-apiserver-operator     openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w      1/1      Running    0       32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm      1/1      Running    0       31m
```

When the current cluster version is **AVAILABLE**, the installation is complete.

1.9.24. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.9.25. Next steps

- Customize your cluster.

- If necessary, you can opt out of remote health reporting .

1.10. INSTALLING A CLUSTER ON GCP IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE

In OpenShift Container Platform version 4.6, you can install a cluster on Google Cloud Platform (GCP) that uses infrastructure that you provide and an internal mirror of the installation release content.
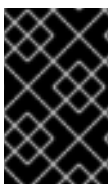
> **IMPORTANT**
>
> While you can install an OpenShift Container Platform cluster by using mirrored installation release content, your cluster still requires internet access to use the GCP APIs.

The steps for performing a user-provided infrastructure install are outlined here. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods.

IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

1.10.1. Prerequisites

- [Create a registry on your mirror host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.

  IMPORTANT

  Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- Review details about the [OpenShift Container Platform installation and update](#) processes.

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to. While you might need to grant access to more sites, you must grant access to ***.googleapis.com** and **accounts.google.com**.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

1.10.2. About installations in restricted networks

In OpenShift Container Platform 4.6, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's Route 53 DNS and IAM services, require internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.
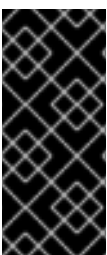
IMPORTANT

Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation using user-provisioned infrastructure. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

1.10.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.

- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

1.10.3. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to obtain the images that are necessary to install your cluster.

You must have Internet access to:

- Access OpenShift Cluster Manager 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.

1.10.4. Configuring your GCP project

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

1.10.4.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See Creating and Managing Projects in the GCP documentation.
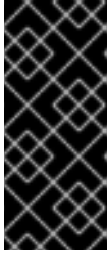
IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>. <base_domain>** URL; the Premium Tier is required for internal load balancing.

1.10.4.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See Enabling services in the GCP documentation.

Table 1.41. Required API services

API service	Console service name
Compute Engine API	**compute.googleapis.com**
Google Cloud APIs	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

1.10.4.3. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the same project that you host the OpenShift Container Platform

cluster. This zone must be authoritative for the domain. The DNS service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through GCP or another source.

   > **NOTE**
   >
   > If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see Google Domains.

2. Create a public hosted zone for your domain or subdomain in your GCP project. See Creating public zones in the GCP documentation.
   Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.

3. Extract the new authoritative name servers from the hosted zone records. See Look up your Cloud DNS name servers in the GCP documentation.
   You typically have four name servers.

4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: How to switch to custom name servers .

5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See Migrating to Cloud DNS in the GCP documentation.

6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

1.10.4.4. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default Quotas do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 1.42. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall rules	Networking	Global	11	1

Service	Component	Location	Total resources required	Resources removed after bootstrap
Forwarding rules	Compute	Global	2	0
Health checks	Compute	Global	2	0
Images	Compute	Global	1	0
Networks	Networking	Global	1	0
Routers	Networking	Global	1	0
Routes	Networking	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0

NOTE

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**

- **asia-northeast2**

- **asia-south1**

- **australia-southeast1**

- **europe-north1**

- **europe-west2**

- **europe-west3**

- **europe-west6**

- **northamerica-northeast1**

- **southamerica-east1**

- **us-west2**

You can increase resource quotas from the GCP console, but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

1.10.4.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a service account in the project that you use to host your OpenShift Container Platform cluster. See Creating a service account in the GCP documentation.

2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See Granting roles to a service account for specific resources.

   > **NOTE**
   >
   > While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. Create the service account key in JSON format. See Creating service account keys in the GCP documentation.
   The service account key is required to create a cluster.

1.10.4.5.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions. If you deploy your cluster into an existing VPC, the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin

- Security Admin

- Service Account Admin

- Service Account User

- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor

- Service Account Key Admin

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 1.43. GCP service account permissions

Account	Roles
Control Plane	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	**roles/compute.viewer**
	roles/storage.admin

1.10.4.6. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- **asia-east1** (Changhua County, Taiwan)

- **asia-east2** (Hong Kong)

- **asia-northeast1** (Tokyo, Japan)

- **asia-northeast2** (Osaka, Japan)

- **asia-northeast3** (Seoul, South Korea)

- **asia-south1** (Mumbai, India)

- **asia-southeast1** (Jurong West, Singapore)

- **asia-southeast2** (Jakarta, Indonesia)

- **australia-southeast1** (Sydney, Australia)

- **europe-north1** (Hamina, Finland)

- **europe-west1** (St. Ghislain, Belgium)

- **europe-west2** (London, England, UK)

- **europe-west3** (Frankfurt, Germany)

- **europe-west4** (Eemshaven, Netherlands)

- **europe-west6** (Zürich, Switzerland)

- **northamerica-northeast1** (Montréal, Québec, Canada)

- **southamerica-east1** (São Paulo, Brazil)

- **us-central1** (Council Bluffs, Iowa, USA)

- **us-east1** (Moncks Corner, South Carolina, USA)

- **us-east4** (Ashburn, Northern Virginia, USA)

- **us-west1** (The Dalles, Oregon, USA)

- **us-west2** (Los Angeles, California, USA)

- **us-west3** (Salt Lake City, Utah, USA)

- **us-west4** (Las Vegas, Nevada, USA)

1.10.4.7. Installing and configuring CLI tools for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must install and configure the CLI tools for GCP.

Prerequisites

- You created a project to host your cluster.

- You created a service account and granted it the required permissions.

Procedure

1. Install the following binaries in **$PATH**:

   - **gcloud**

   - **gsutil**

   See Install the latest Cloud SDK version in the GCP documentation.

2. Authenticate using the **gcloud** tool with your configured service account.
   See Authorizing with a service account in the GCP documentation.

1.10.5. Creating the installation files for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files. You also have the option to first set up a separate **var** partition during the preparation phases of installation.

1.10.5.1. Optional: Creating a separate /var partition

It is recommended that disk partitioning for OpenShift Container Platform be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the /**var** partition or a subdirectory of /**var**. For example:

- /**var**/**lib**/**containers**: Holds container-related content that can grow as more images and containers are added to a system.

- /**var**/**lib**/**etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.

- /**var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a /**var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because /**var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate /**var** partition by creating a machine config that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform installation.

> **IMPORTANT**
>
> If you follow the steps to create a separate /**var** partition in this procedure, it is not necessary to create the Kubernetes manifest and Ignition config files again as described later in this section.

Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

   ```
   $ mkdir $HOME/clusterconfig
   ```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

   ```
   $ openshift-install create manifests --dir $HOME/clusterconfig
   ```

   Example output

   ```
   ? SSH Public Key ...
   ```

```
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. Optional: Confirm that the installation program created manifests in the
   clusterconfig/openshift directory:

```
$ ls $HOME/clusterconfig/openshift/
```

Example output

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. Create a **MachineConfig** object and add it to a file in the **openshift** directory. For example,
   name the file **98-var-partition.yaml**, change the disk device name to the name of the storage
   device on the **worker** systems, and set the storage size as appropriate. This example places the
   /var directory on a separate partition:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      disks:
      - device: /dev/<device_name>      1
        partitions:
        - label: var
          startMiB: <partition_start_offset>      2
          sizeMiB: <partition_size>      3
      filesystems:
        - device: /dev/disk/by-partlabel/var
          path: /var
          format: xfs
    systemd:
      units:
      - name: var.mount      4
        enabled: true
        contents: |
          [Unit]
          Before=local-fs.target
          [Mount]
          What=/dev/disk/by-partlabel/var
          Where=/var
```

```
Options=defaults,prjquota 5
[Install]
WantedBy=local-fs.target
```

1 The storage device name of the disk that you want to partition.

2 When adding a data partition to the boot disk, a minimum value of 25000 MiB (Mebibytes) is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.

3 The size of the data partition in mebibytes.

4 The name of the mount unit must match the directory specified in the **Where=** directive. For example, for a filesystem mounted on **/var/lib/containers**, the unit must be named **var-lib-containers.mount**.

5 The **prjquota** mount option must be enabled for filesystems used for container storage.

> **NOTE**
>
> When creating a separate /**var** partition, you cannot use different instance types for worker nodes, if the different instance types do not have the same device name.

5. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Now you can use the Ignition config files as input to the installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

1.10.5.2. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your mirror host.

- Have the **imageContentSources** values that were generated during mirror registry creation.

- Obtain the contents of the certificate for your mirror registry.

Procedure

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

a. Change to the directory that contains the installation program and 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 **gcp** as the platform to target.

iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.

iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.

v. Select the region to deploy the cluster to.

vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.

vii. Enter a descriptive name for your cluster.

viii. Paste the pull secret from the Red Hat OpenShift Cluster Manager .

2. Edit the **install-config.yaml** file to provide the additional information that is required for an installation in a restricted network.

a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email": "you@example.com"}}}'
```

For **<mirror_host_name>**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

b. Add the **additionalTrustBundle** parameter and value.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

The value must be the contents of the certificate file that you used for your mirror registry, which can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

c. Define the network and subnets for the VPC to install the cluster in under the parent **platform.gcp** field:

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

For **platform.gcp.network**, specify the name for the existing Google VPC. For **platform.gcp.controlPlaneSubnet** and **platform.gcp.computeSubnet**, specify the existing subnets to deploy the control plane machines and compute machines, respectively.

d. Add the image content resources, which look like this excerpt:

```
imageContentSources:
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

To complete these values, use the **imageContentSources** that you recorded during mirror registry creation.

3. Make any other modifications to the **install-config.yaml** file that you require. You can find more information about the available parameters in the **Installation configuration parameters** section.

4. 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.

1.10.5.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

> **NOTE**
>
> The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.
>
> For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace to hold the additional CA certificates. If you provide **additionalTrustBundle** and at least one proxy setting, the **Proxy** object is configured to reference the **user-ca-bundle** config map in the **trustedCA** field. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges the contents

specified for the **trustedCA** parameter with the RHCOS trust bundle. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

> **NOTE**
>
> The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

> **NOTE**
>
> Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

1.10.5.4. 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.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to create the cluster.

> **IMPORTANT**
>
> - The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
>
> - It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

Prerequisites

- You obtained the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.

- You created the **install-config.yaml** installation configuration file.

Procedure

1. Change to the directory that contains the installation program and generate the Kubernetes manifests for the cluster:

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

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

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

   ```
   $ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
   ```

   By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

   ```
   $ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
   ```

   Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:

   a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.

   b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.

   c. Save and exit the file.

5. Optional: If you do not want the Ingress Operator to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS configuration file:

   ```
   apiVersion: config.openshift.io/v1
   kind: DNS
   metadata:
     creationTimestamp: null
     name: cluster
   spec:
     baseDomain: example.openshift.com
     privateZone: ❶
       id: mycluster-100419-private-zone
     publicZone: ❷
       id: example.openshift.com
   status: {}
   ```

   ❶ ❷ Remove this section completely.

   If you do so, you must add ingress DNS records manually in a later step.

6. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

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

❶ 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
```

Additional resources

- Optional: Adding the ingress DNS records

1.10.6. Exporting common variables

1.10.6.1. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The infrastructure name is also used to locate the appropriate GCP resources during an OpenShift Container Platform installation. The provided Deployment Manager templates contain references to this infrastructure name, so you must extract it.

Prerequisites

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

- You generated the Ignition config files for your cluster.

- You installed the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json ❶
```

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

Example output

> openshift-vw9j6 **1**

1 The output of this command is your cluster name and a random string.

1.10.6.2. Exporting common variables for Deployment Manager templates

You must export a common set of variables that are used with the provided Deployment Manager templates used to assist in completing a user-provided infrastructure install on Google Cloud Platform (GCP).

> **NOTE**
>
> Specific Deployment Manager templates can also require additional exported variables, which are detailed in their related procedures.

Prerequisites

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

- Generate the Ignition config files for your cluster.

- Install the **jq** package.

Procedure

1. Export the following common variables to be used by the provided Deployment Manager templates:

    ```
    $ export BASE_DOMAIN='<base_domain>'
    $ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
    $ export NETWORK_CIDR='10.0.0.0/16'
    $ export MASTER_SUBNET_CIDR='10.0.0.0/19'
    $ export WORKER_SUBNET_CIDR='10.0.32.0/19'

    $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
    $ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
    $ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
    $ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
    $ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
    ```

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

1.10.7. Creating a VPC in GCP

You must create a VPC in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements. One way to create the VPC is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

Procedure

1. Copy the template from the **Deployment Manager template for the VPC** section of this topic and save it as **01_vpc.py** on your computer. This template describes the VPC that your cluster requires.

2. Create a **01_vpc.yaml** resource definition file:

   ```
   $ cat <<EOF >01_vpc.yaml
   imports:
   - path: 01_vpc.py

   resources:
   - name: cluster-vpc
     type: 01_vpc.py
     properties:
       infra_id: '${INFRA_ID}' 1
       region: '${REGION}' 2
       master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
       worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
   EOF
   ```

1	**infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
2	**region** is the region to deploy the cluster into, for example **us-central1**.
3	**master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/19**.
4	**worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.32.0/19**.

3. Create the deployment by using the **gcloud** CLI:

   ```
   $ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
   ```

1.10.7.1. Deployment Manager template for the VPC

You can use the following Deployment Manager template to deploy the VPC that you need for your OpenShift Container Platform cluster:

Example 1.19. **01_vpc.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

1.10.8. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 1.44. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	**1936**	Metrics
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101** and the Cluster Version Operator on port**9099**.
	10250-**10259**	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	**4789**	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101**.
TCP/UDP	**30000**-**32767**	Kubernetes node port

Table 1.45. All machines to control plane

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 1.46. Control plane machines to control plane machines

Protocol	Port	Description
TCP	**2379**-**2380**	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer**: Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.

   - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

   > **IMPORTANT**
   >
   > Do not configure session persistence for an API load balancer.

   Configure the following ports on both the front and back of the load balancers:

   Table 1.47. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the **/readyz** endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

   > **NOTE**
   >
   > The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer**: Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 1.48. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

1.10.9. Creating load balancers in GCP

You must configure load balancers in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.

2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.

3. Export the variables that the deployment template uses:

   a. Export the cluster network location:

   ```
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
   ```

   b. Export the control plane subnet location:

   ```
   $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   c. Export the three zones that the cluster uses:

   ```
   $ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
   ```

4. Create a **02_infra.yaml** resource definition file:

   ```
   $ cat <<EOF >02_infra.yaml
   imports:
   - path: 02_lb_ext.py
   - path: 02_lb_int.py 1
   resources:
   - name: cluster-lb-ext 2
     type: 02_lb_ext.py
     properties:
       infra_id: '${INFRA_ID}' 3
       region: '${REGION}' 4
   - name: cluster-lb-int
     type: 02_lb_int.py
     properties:
       cluster_network: '${CLUSTER_NETWORK}'
       control_subnet: '${CONTROL_SUBNET}' 5
       infra_id: '${INFRA_ID}'
       region: '${REGION}'
       zones: 6
       - '${ZONE_0}'
       - '${ZONE_1}'
       - '${ZONE_2}'
   EOF
   ```

   1 **2** Required only when deploying an external cluster.

③ **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

④ **region** is the region to deploy the cluster into, for example **us-central1**.

⑤ **control_subnet** is the URI to the control subnet.

⑥ **zones** are the zones to deploy the control plane instances into, like **us-east1-b**, **us-east1-c**, and **us-east1-d**.

5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. Export the cluster IP address:

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. For an external cluster, also export the cluster public IP address:

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

1.10.9.1. Deployment Manager template for the external load balancer

You can use the following Deployment Manager template to deploy the external load balancer that you need for your OpenShift Container Platform cluster:

Example 1.20. 02_lb_ext.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
        probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
```

```
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

1.10.9.2. Deployment Manager template for the internal load balancer

You can use the following Deployment Manager template to deploy the internal load balancer that you need for your OpenShift Container Platform cluster:

Example 1.21. **02_lb_int.py** Deployment Manager template

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
```

```
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
    check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
    service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

1.10.10. Creating a private DNS zone in GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.

2. Create a **02_dns.yaml** resource definition file:

   ```
   $ cat <<EOF >02_dns.yaml
   imports:
   - path: 02_dns.py

   resources:
   - name: cluster-dns
     type: 02_dns.py
     properties:
       infra_id: '${INFRA_ID}' 1
       cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
       cluster_network: '${CLUSTER_NETWORK}' 3
   EOF
   ```

   1   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

   2   **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.

   3   **cluster_network** is the **selfLink** URL to the cluster network.

3. Create the deployment by using the **gcloud** CLI:

   ```
   $ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
   ```

4. The templates do not create DNS entries due to limitations of Deployment Manager, so you must create them manually:

   a. Add the internal DNS entries:

      ```
      $ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
      $ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
      $ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
      ```

```
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

b. For an external cluster, also add the external DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

1.10.10.1. Deployment Manager template for the private DNS

You can use the following Deployment Manager template to deploy the private DNS that you need for your OpenShift Container Platform cluster:

Example 1.22. 02_dns.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

1.10.11. Creating firewall rules in GCP

You must create firewall rules in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.

2. Create a **03_firewall.yaml** resource definition file:

   ```
   $ cat <<EOF >03_firewall.yaml
   imports:
   - path: 03_firewall.py

   resources:
   - name: cluster-firewall
     type: 03_firewall.py
     properties:
       allowed_external_cidr: '0.0.0.0/0' ❶
       infra_id: '${INFRA_ID}' ❷
       cluster_network: '${CLUSTER_NETWORK}' ❸
       network_cidr: '${NETWORK_CIDR}' ❹
   EOF
   ```

   ❶ **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **${NETWORK_CIDR}**.

   ❷ **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

   ❸ **cluster_network** is the **selfLink** URL to the cluster network.

   ❹ **network_cidr** is the CIDR of the VPC network, for example **10.0.0.0/16**.

3. Create the deployment by using the **gcloud** CLI:

   ```
   $ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
   03_firewall.yaml
   ```

1.10.11.1. Deployment Manager template for firewall rules

You can use the following Deployment Manager template to deploy the firewall rues that you need for
your OpenShift Container Platform cluster:

Example 1.23. **03_firewall.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
```

```
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
```

```
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }]

    return {'resources': resources}
```

1.10.12. Creating IAM roles in GCP

You must create IAM roles in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.

2. Create a **03_iam.yaml** resource definition file:

   ```
   $ cat <<EOF >03_iam.yaml
   imports:
   - path: 03_iam.py
   resources:
   - name: cluster-iam
     type: 03_iam.py
   ```

```
  properties:
    infra_id: '${INFRA_ID}' 1
EOF
```

1   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. Export the variable for the master service account:

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. Export the variable for the worker service account:

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. Export the variable for the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. The templates do not create the policy bindings due to limitations of Deployment Manager, so you must create them manually:

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. Create a service account key and store it locally for later use:

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

1.10.12.1. Deployment Manager template for IAM roles

You can use the following Deployment Manager template to deploy the IAM roles that you need for your OpenShift Container Platform cluster:

Example 1.24. 03_iam.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

1.10.13. Creating the RHCOS cluster image for the GCP infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) image for Google Cloud Platform (GCP) for your OpenShift Container Platform nodes.

Procedure

1. Obtain the RHCOS image from the RHCOS image mirror page.

   > **IMPORTANT**
   >
   > The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

   The file name contains the OpenShift Container Platform version number in the format **rhcos-<version>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

   ```
   $ gsutil mb gs://<bucket_name>
   ```

3. Upload the RHCOS image to the Google storage bucket:

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. Export the uploaded RHCOS image location as a variable:

```
$ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz`
```

5. Create the cluster image:

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

1.10.14. Creating the bootstrap machine in GCP

You must create the bootstrap machine in Google Cloud Platform (GCP) to use during OpenShift Container Platform cluster initialization. One way to create this machine is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your bootstrap machine, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Ensure pyOpenSSL is installed.

Procedure

1. Copy the template from the **Deployment Manager template for the bootstrap machine** section of this topic and save it as **04_bootstrap.py** on your computer. This template describes the bootstrap machine that your cluster requires.

2. Export the location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

3. Create a bucket and upload the **bootstrap.ign** file:

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Create a signed URL for the bootstrap instance to use to access the Ignition config. Export the URL from the output as a variable:

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. Create a **04_bootstrap.yaml** resource definition file:

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    zone: '${ZONE_0}' 3

    cluster_network: '${CLUSTER_NETWORK}' 4
    control_subnet: '${CONTROL_SUBNET}' 5
    image: '${CLUSTER_IMAGE}' 6
    machine_type: 'n1-standard-4' 7
    root_volume_size: '128' 8

    bootstrap_ign: '${BOOTSTRAP_IGN}' 9
EOF
```

1   **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

2   **region** is the region to deploy the cluster into, for example **us-central1**.

3   **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-b**.

4   **cluster_network** is the **selfLink** URL to the cluster network.

5   **control_subnet** is the **selfLink** URL to the control subnet.

6   **image** is the **selfLink** URL to the RHCOS image.

7   **machine_type** is the machine type of the instance, for example **n1-standard-4**.

8   **root_volume_size** is the boot disk size for the bootstrap machine.

9   **bootstrap_ign** is the URL output when creating a signed URL.

6. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the bootstrap machine manually.

   a. Add the bootstrap instance to the internal load balancer instance group:

   ```
   $ gcloud compute instance-groups unmanaged add-instances \
       ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
   bootstrap
   ```

   b. Add the bootstrap instance group to the internal load balancer backend service:

   ```
   $ gcloud compute backend-services add-backend \
       ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
   group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
   ```

1.10.14.1. Deployment Manager template for the bootstrap machine

You can use the following Deployment Manager template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster:

Example 1.25. **04_bootstrap.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"}},"version":"3.1.0"}}',
                }]
            },
```

```
                  'networkInterfaces': [{
                      'subnetwork': context.properties['control_subnet'],
                      'accessConfigs': [{
                          'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                      }]
                  }],
                  'tags': {
                      'items': [
                          context.properties['infra_id'] + '-master',
                          context.properties['infra_id'] + '-bootstrap'
                      ]
                  },
                  'zone': context.properties['zone']
              }
          }, {
              'name': context.properties['infra_id'] + '-bootstrap-instance-group',
              'type': 'compute.v1.instanceGroup',
              'properties': {
                  'namedPorts': [
                      {
                          'name': 'ignition',
                          'port': 22623
                      }, {
                          'name': 'https',
                          'port': 6443
                      }
                  ],
                  'network': context.properties['cluster_network'],
                  'zone': context.properties['zone']
              }
          }]

      return {'resources': resources}
```

1.10.15. Creating the control plane machines in GCP

You must create the control plane machines in Google Cloud Platform (GCP) for your cluster to use. One way to create these machines is to modify the provided Deployment Manager template.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your control plane machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

Procedure

1. Copy the template from the **Deployment Manager template for control plane machines** section of this topic and save it as **05_control_plane.py** on your computer. This template describes the control plane machines that your cluster requires.

2. Export the following variable required by the resource definition:

   ```
   $ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
   ```

3. Create a **05_control_plane.yaml** resource definition file:

   ```
   $ cat <<EOF >05_control_plane.yaml
   imports:
   - path: 05_control_plane.py

   resources:
   - name: cluster-control-plane
     type: 05_control_plane.py
     properties:
       infra_id: '${INFRA_ID}' 1
       zones: 2
       - '${ZONE_0}'
       - '${ZONE_1}'
       - '${ZONE_2}'

       control_subnet: '${CONTROL_SUBNET}' 3
       image: '${CLUSTER_IMAGE}' 4
       machine_type: 'n1-standard-4' 5
       root_volume_size: '128'
       service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

       ignition: '${MASTER_IGNITION}' 7
   EOF
   ```

   1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

   2 **zones** are the zones to deploy the control plane instances into, for example **us-central1-a**, **us-central1-b**, and **us-central1-c**.

   3 **control_subnet** is the **selfLink** URL to the control subnet.

   4 **image** is the **selfLink** URL to the RHCOS image.

   5 **machine_type** is the machine type of the instance, for example **n1-standard-4**.

   6 **service_account_email** is the email address for the master service account that you created.

7 **ignition** is the contents of the **master.ign** file.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually.

- Run the following commands to add the control plane machines to the appropriate instance groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-master-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-master-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-master-2
```

- For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-master-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-master-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-master-2
```

1.10.15.1. Deployment Manager template for control plane machines

You can use the following Deployment Manager template to deploy the control plane machines that you need for your OpenShift Container Platform cluster:

Example 1.26. **05_control_plane.py** Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
```

```
context.properties['machine_type'],
      'metadata': {
        'items': [{
          'key': 'user-data',
          'value': context.properties['ignition']
        }]
      },
      'networkInterfaces': [{
        'subnetwork': context.properties['control_subnet']
      }],
      'serviceAccounts': [{
        'email': context.properties['service_account_email'],
        'scopes': ['https://www.googleapis.com/auth/cloud-platform']
      }],
      'tags': {
        'items': [
          context.properties['infra_id'] + '-master',
        ]
      },
      'zone': context.properties['zones'][0]
    }
  }, {
    'name': context.properties['infra_id'] + '-master-1',
    'type': 'compute.v1.instance',
    'properties': {
      'disks': [{
        'autoDelete': True,
        'boot': True,
        'initializeParams': {
          'diskSizeGb': context.properties['root_volume_size'],
          'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
          'sourceImage': context.properties['image']
        }
      }],
      'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
      'metadata': {
        'items': [{
          'key': 'user-data',
          'value': context.properties['ignition']
        }]
      },
      'networkInterfaces': [{
        'subnetwork': context.properties['control_subnet']
      }],
      'serviceAccounts': [{
        'email': context.properties['service_account_email'],
        'scopes': ['https://www.googleapis.com/auth/cloud-platform']
      }],
      'tags': {
        'items': [
          context.properties['infra_id'] + '-master',
        ]
      },
      'zone': context.properties['zones'][1]
    }
```

```
      }, {
        'name': context.properties['infra_id'] + '-master-2',
        'type': 'compute.v1.instance',
        'properties': {
          'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
              'diskSizeGb': context.properties['root_volume_size'],
              'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
              'sourceImage': context.properties['image']
            }
          }],
          'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
          'metadata': {
            'items': [{
              'key': 'user-data',
              'value': context.properties['ignition']
            }]
          },
          'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
          }],
          'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
          }],
          'tags': {
            'items': [
              context.properties['infra_id'] + '-master',
            ]
          },
          'zone': context.properties['zones'][2]
        }
      }]

    return {'resources': resources}
```

1.10.16. Wait for bootstrap completion and remove bootstrap resources in GCP

After you create all of the required infrastructure in Google Cloud Platform (GCP), wait for the
bootstrap process to complete on the machines that you provisioned by using the Ignition config files
that you generated with the installation program.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

   ```
   $ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
       --log-level info 2
   ```

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

   2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

   If the command exits without a **FATAL** warning, your production control plane has initialized.

2. Delete the bootstrap resources:

   ```
   $ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
   service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
   instance-group-zone=${ZONE_0}
   $ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
   $ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
   $ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
   ```

1.10.17. Creating additional worker machines in GCP

You can create worker machines in Google Cloud Platform (GCP) for your cluster to use by launching individual instances discretely or by automated processes outside the cluster, such as auto scaling groups. You can also take advantage of the built-in cluster scaling mechanisms and the machine API in OpenShift Container Platform.

In this example, you manually launch one instance by using the Deployment Manager template. Additional instances can be launched by including additional resources of type **06_worker.py** in the file.

> **NOTE**
>
> If you do not use the provided Deployment Manager template to create your worker machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

Procedure

1. Copy the template from the **Deployment Manager template for worker machines** section of this topic and save it as **06_worker.py** on your computer. This template describes the worker machines that your cluster requires.

2. Export the variables that the resource definition uses.

   a. Export the subnet that hosts the compute machines:

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   b. Export the email address for your service account:

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

   c. Export the location of the compute machine Ignition config file:

   ```
   $ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
   ```

3. Create a **06_worker.yaml** resource definition file:

   ```
   $ cat <<EOF >06_worker.yaml
   imports:
   - path: 06_worker.py

   resources:
   - name: 'worker-0'                                                      1
     type: 06_worker.py
     properties:
       infra_id: '${INFRA_ID}'                                            2
       zone: '${ZONE_0}'                                                  3
       compute_subnet: '${COMPUTE_SUBNET}'                                4
       image: '${CLUSTER_IMAGE}'                                          5
       machine_type: 'n1-standard-4'                                      6
       root_volume_size: '128'
       service_account_email: '${WORKER_SERVICE_ACCOUNT}'                 7
       ignition: '${WORKER_IGNITION}'                                     8
   - name: 'worker-1'
     type: 06_worker.py
     properties:
       infra_id: '${INFRA_ID}'                                            9
       zone: '${ZONE_1}'                                                  10
   ```

```
            compute_subnet: '${COMPUTE_SUBNET}' 11
            image: '${CLUSTER_IMAGE}' 12
            machine_type: 'n1-standard-4' 13
            root_volume_size: '128'
            service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
            ignition: '${WORKER_IGNITION}' 15
        EOF
```

1 **name** is the name of the worker machine, for example **worker-0**.

2 **9** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3 **10** **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.

4 **11** **compute_subnet** is the **selfLink** URL to the compute subnet.

5 **12** **image** is the **selfLink** URL to the RHCOS image.

6 **13** **machine_type** is the machine type of the instance, for example **n1-standard-4**.

7 **14** **service_account_email** is the email address for the worker service account that you created.

8 **15** **ignition** is the contents of the **worker.ign** file.

4. Optional: If you want to launch additional instances, include additional resources of type **06_worker.py** in your **06_worker.yaml** resource definition file.

5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

1.10.17.1. Deployment Manager template for worker machines

You can use the following Deployment Manager template to deploy the worker machines that you need for your OpenShift Container Platform cluster:

Example 1.27. 06_worker.py Deployment Manager template

```python
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
```

```
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
    context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

1.10.18. Logging in to the cluster by using the CLI

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

- You deployed an OpenShift Container Platform cluster.

- You installed the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

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

   ❶ 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
   ```

Example output

```
system:admin
```

1.10.19. Disabling the default OperatorHub sources

Operator catalogs that source content provided by Red Hat and community projects are configured for OperatorHub by default during an OpenShift Container Platform installation. In a restricted network environment, you must disable the default catalogs as a cluster administrator.

Procedure

- Disable the sources for the default catalogs by adding **disableAllDefaultSources: true** to the **OperatorHub** object:

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

TIP

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Global Configuration → OperatorHub** page, click the **Sources** tab, where you can create, delete, disable, and enable individual sources.

1.10.20. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME      STATUS   ROLES   AGE VERSION
   master-0  Ready    master  63m v1.19.0
   master-1  Ready    master  63m v1.19.0
   master-2  Ready    master  64m v1.19.0
   ```

   The output lists all of the machines that you created.

> **NOTE**
>
> The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME       AGE    REQUESTOR                                              CONDITION
csr-8b2br  15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps  15m    system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

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. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

> **NOTE**
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- 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 --no-run-if-empty oc adm certificate approve
```

> **NOTE**
>
> Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME        AGE     REQUESTOR                                          CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- 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
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES   AGE  VERSION
master-0  Ready    master  73m  v1.20.0
master-1  Ready    master  73m  v1.20.0
```

```
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

> **NOTE**
>
> It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests .

1.10.21. Optional: Adding the ingress DNS records

If you removed the DNS zone configuration when creating Kubernetes manifests and generating Ignition configs, you must manually create DNS records that point at the ingress load balancer. You can create either a wildcard ***.apps.{baseDomain}.** or specific records. You can use A, CNAME, and other records per your requirements.

Prerequisites

- Configure a GCP account.

- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.

- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.

- Create the bootstrap machine.

- Create the control plane machines.

- Create the worker machines.

Procedure

1. Wait for the Ingress router to create a load balancer and populate the **EXTERNAL-IP** field:

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   Example output

   ```
   NAME            TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)                  AGE
   router-default  LoadBalancer  172.30.18.154   35.233.157.184
   80:32288/TCP,443:31215/TCP   98
   ```

2. Add the A record to your zones:

   - To use A records:

i. Export the variable for the router IP address:

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-headers | awk '{print $4}'`
```

ii. Add the A record to the private zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

iii. For an external cluster, also add the A record to the public zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- To add explicit domains instead of using a wildcard, create entries for each of the cluster's current routes:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

Example output

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

1.10.22. Completing a GCP installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Google Cloud Platform (GCP) user-provisioned infrastructure, you can monitor the cluster events until the cluster is ready.

Prerequisites

- Deploy the bootstrap machine for an OpenShift Container Platform cluster on user-provisioned GCP infrastructure.

- Install the **oc** CLI and log in.

Procedure

1. Complete the cluster installation:

   ```
   $ ./openshift-install --dir <installation_directory> wait-for install-complete ❶
   ```

   Example output

   ```
   INFO Waiting up to 30m0s for the cluster to initialize...
   ```

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

   > **IMPORTANT**
   >
   > - The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
   >
   > - It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Observe the running state of your cluster.

   a. Run the following command to view the current cluster version and status:

      ```
      $ oc get clusterversion
      ```

      Example output

      ```
      NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
      version             False       True          24m     Working towards 4.5.4: 99% complete
      ```

   b. Run the following command to view the Operators managed on the control plane by the Cluster Version Operator (CVO):

      ```
      $ oc get clusteroperators
      ```

      Example output

      ```
      NAME                        VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
      authentication              4.5.4     True        False         False      7m56s
      cloud-credential            4.5.4     True        False         False      31m
      cluster-autoscaler          4.5.4     True        False         False      16m
      console                     4.5.4     True        False         False      10m
      ```

```
csi-snapshot-controller               4.5.4    True     False     False    16m
dns                          4.5.4    True     False     False    22m
etcd                         4.5.4    False    False     False    25s
image-registry                  4.5.4    True     False     False    16m
ingress                       4.5.4    True     False     False    16m
insights                      4.5.4    True     False     False    17m
kube-apiserver                   4.5.4    True     False     False    19m
kube-controller-manager            4.5.4    True     False     False    20m
kube-scheduler                   4.5.4    True     False     False    20m
kube-storage-version-migrator        4.5.4    True     False     False    16m
machine-api                     4.5.4    True     False     False    22m
machine-config                   4.5.4    True     False     False    22m
marketplace                     4.5.4    True     False     False    16m
monitoring                     4.5.4    True     False     False    10m
network                       4.5.4    True     False     False    23m
node-tuning                     4.5.4    True     False     False    23m
openshift-apiserver               4.5.4    True     False     False    17m
openshift-controller-manager         4.5.4    True     False     False    15m
openshift-samples                 4.5.4    True     False     False    16m
operator-lifecycle-manager          4.5.4    True     False     False    22m
operator-lifecycle-manager-catalog     4.5.4    True     False     False    22m
operator-lifecycle-manager-packageserver 4.5.4    True     False     False    18m
service-ca                     4.5.4    True     False     False    23m
service-catalog-apiserver           4.5.4    True     False     False    23m
service-catalog-controller-manager     4.5.4    True     False     False    23m
storage                       4.5.4    True     False     False    17m
```

c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

```
NAMESPACE                          NAME
READY    STATUS    RESTARTS  AGE
kube-system                         etcd-member-ip-10-0-3-111.us-east-
2.compute.internal          1/1     Running    0      35m
kube-system                         etcd-member-ip-10-0-3-239.us-east-
2.compute.internal          1/1     Running    0      37m
kube-system                         etcd-member-ip-10-0-3-24.us-east-
2.compute.internal              1/1     Running    0      35m
openshift-apiserver-operator              openshift-apiserver-operator-6d6674f4f4-
h7t2t              1/1     Running    1      37m
openshift-apiserver                  apiserver-fm48r
1/1     Running    0      30m
openshift-apiserver                  apiserver-fxkvv
1/1     Running    0      29m
openshift-apiserver                  apiserver-q85nm
1/1     Running    0      29m
...
openshift-service-ca-operator              openshift-service-ca-operator-66ff6dc6cd-
9r257              1/1     Running    0      37m
openshift-service-ca                  apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1     Running    0      35m
openshift-service-ca                  configmap-cabundle-injector-8498544d7-
```

```
25qn6                        1/1      Running    0        35m
openshift-service-ca                          service-serving-cert-signer-6445fc9c6-wqdqn
1/1      Running    0        35m
openshift-service-catalog-apiserver-operator        openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w        1/1      Running    0        32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm    1/1      Running    0        31m
```

When the current cluster version is **AVAILABLE**, the installation is complete.

1.10.23. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.10.24. Next steps

- Customize your cluster.

- Configure image streams for the Cluster Samples Operator and the **must-gather** tool.

- Learn how to use Operator Lifecycle Manager (OLM) on restricted networks .

- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by configuring additional trust stores.

- If necessary, you can opt out of remote health reporting .

1.11. UNINSTALLING A CLUSTER ON GCP

You can remove a cluster that you deployed to Google Cloud Platform (GCP).

1.11.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.

> **NOTE**
>
> After uninstallation, check your cloud provider for any resources not removed properly, especially with User Provisioned Infrastructure (UPI) clusters. There might be resources that the installer did not create or that the installer is unable to access. For example, some Google Cloud resources require IAM permissions in shared VPC host projects, or there might be unused health checks that must be deleted .

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 directory that contains the installation program on 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**.
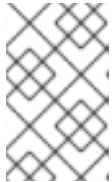
   > **NOTE**
   >
   > You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.