OpenShift Container Platform 4.3

Installing

Installing and configuring OpenShift Container Platform clusters

Last Updated: 2020-01-23

OpenShift Container Platform 4.3 Installing

Installing and configuring OpenShift Container Platform clusters

Legal Notice

Abstract

This document provides information about installing OpenShift Container Platform and details about some configuration processes.

Table of Contents

CHAPTER 1. GATHERING INSTALLATION LOGS

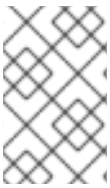
To assist in troubleshooting a failed OpenShift Container Platform installation, you can gather logs from the bootstrap and control plane, or master, machines.

Prerequisites

- You attempted to install an OpenShift Container Platform cluster, and installation failed.
- You provided an SSH key to the installation program, and that key is in your running **ssh-agent** process.

1.1. GATHERING LOGS FROM A FAILED INSTALLATION

If you gave an SSH key to your installation program, you can gather data about your failed installation.



NOTE

You use a different command to gather logs about an unsuccessful installation than to gather logs from a running cluster. If you must gather logs from a running cluster, use the **oc adm must-gather** command.

Prerequisites

- Your OpenShift Container Platform installation failed before the bootstrap process finished. The bootstrap node must be running and accessible through SSH.
- The **ssh-agent** process is active on your computer, and you provided both the **ssh-agent** process and the installation program the same SSH key.
- If you tried to install a cluster on infrastructure that you provisioned, you must have the fully-qualified domain names of the control plane, or master, machines.

Procedure

1. Generate the commands that are required to obtain the installation logs from the bootstrap and control plane machines:
   - If you used installer-provisioned infrastructure, run the following command:

     ```
     $ ./openshift-install gather bootstrap --dir=<directory> 1
     ```

     1 **installation_directory** is the directory you stored the OpenShift Container Platform definition files that the installation program creates.

     For installer-provisioned infrastructure, the installation program stores information about the cluster, so you do not specify the host names or IP addresses

   - If you used infrastructure that you provisioned yourself, run the following command:

     ```
     $ ./openshift-install gather bootstrap --dir=<directory> \ 1
       --bootstrap <bootstrap_address> \ 2
       --master <master_1_address> \ 3
     ```

```
--master <master_2_address> \ 4
--master <master_3_address>" 5
```

1. **installation_directory** is the directory you stored the OpenShift Container Platform definition files that the installation program creates.

2. **<bootstrap_address>** is the fully-qualified domain name or IP address of the cluster's bootstrap machine.

3 4 5 For each control plane, or master, machine in your cluster, replace **<master_*_address>** with its fully-qualified domain name or IP address.

NOTE

A default cluster contains three control plane machines. List all of your control plane machines as shown, no matter how many your cluster uses.

The command output resembles the following example:

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<directory>/log-bundle-<timestamp>.tar.gz"
```

If you open a Red Hat support case about your installation failure, include the compressed logs in the case.

1.2. MANUALLY GATHERING LOGS WITH SSH ACCESS TO YOUR HOST(S)

Manually gather logs in situations where **must-gather** or automated collection methods do not work.

Prerequisites

- You must have SSH access to your host(s).

Procedure

1. Collect the **bootkube.service** service logs from the bootstrap host using the **journalctl** command by running:

```
$ journalctl -b -f -u bootkube.service
```

2. Collect the bootstrap host's container logs using the Podman logs. This is shown as a loop to get all of the container logs from the host:

```
$ for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done
```

3. Alternatively, collect the host's container logs using the **tail** command by running:

```
# tail -f /var/lib/containers/storage/overlay-containers/*/userdata/ctr.log
```

4. Collect the **kubelet.service** and **crio.service** service logs from the master and worker hosts using the **journalctl** command by running:

```
$ journalctl -b -f -u kubelet.service -u crio.service
```

5. Collect the master and worker host container logs using the **tail** command by running:

```
$ sudo tail -f /var/log/containers/*
```

1.3. MANUALLY GATHERING LOGS WITHOUT SSH ACCESS TO YOUR HOST(S)

Manually gather logs in situations where **must-gather** or automated collection methods do not work.

If you do not have SSH access to your node, you can access the systems journal to investigate what is happening on your host.

Prerequisites

- Your OpenShift Container Platform installation must be complete.
- Your API service is still functional.
- You have system administrator privileges.

Procedure

1. Access **journald** unit logs under **/var/log** by running:

```
$ oc adm node-logs --role=master -u kubelet
```

2. Access host file paths under **/var/log** by running:

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

CHAPTER 2. FIPS COMPLIANT OPENSHIFT CONTAINER PLATFORM CLUSTERS

Starting with version 4.3, you can install an OpenShift Container Platform cluster that use FIPS validated cryptographic libraries.

For the Red Hat Enterprise Linux CoreOS (RHCOS) machines in your cluster, this change is applied when the machines are deployed based on the status of an option in the **install-config.yaml** file, which governs the cluster options that a user can change during cluster deployment. With Red Hat Enterprise Linux machines, you must enable FIPS mode when you install the operating system on the machines that you plan to use as worker machines. These configuration methods ensure that your cluster meet the requirements of a FIPS compliance audit: only FIPS validated cryptography packages are enabled before the initial system boot.

Because FIPS must be enabled before the operating system that your cluster uses boots for the first time, you cannot enable FIPS after you deploy a cluster.

2.1. FIPS VALIDATION IN OPENSHIFT CONTAINER PLATFORM

OpenShift Container Platform uses certain FIPS validated modules within Red Hat Enterprise Linux (RHEL) and RHCOS for the operating system components that it uses. See RHEL7 core crypto components. For example, when users SSH into OpenShift Container Platform clusters and containers, those connections are properly encrypted.

OpenShift Container Platform components are written in Go and built with Red Hat's golang compiler. When you enable FIPS mode for your cluster, Red Hat's golang compiler calls RHEL and RHCOS cryptographic libraries for all OpenShift Container Platform components that require cryptographic signing. At the initial release of OpenShift Container Platform version 4.3, only the **ose-sdn** package uses the native golang cryptography, which is not FIPS validated. Red Hat verifies that all other packages use the FIPS validated OpenSSL module.

Table 2.1. FIPS mode attributes and limitations in OpenShift Container Platform 4.3

Attributes	Limitations
FIPS compliant operating systems: RHEL 7 and RHCOS.	The FIPS implementation does not offer a single function that both computes hash functions and validates the keys that are based on that hash. This limitation will continue to be evaluated and improved in future OpenShift Container Platform releases.
FIPS compliant CRI-O runtimes.	
FIPS compliant OpenShift Container Platform services.	
FIPS validated cryptographic module and algorithms that are obtained from RHEL 7 and RHCOS binaries and images.	
Use of FIPS compatible golang compiler.	TLS FIPS compliance is not complete but is planned for future OpenShift Container Platform releases.

2.2. FIPS COMPLIANCE IN COMPONENTS THAT THE CLUSTER USES

Although the OpenShift Container Platform cluster itself uses FIPS validated modules, ensure that the systems that support your OpenShift Container Platform cluster use FIPS validated modules for cryptography.

2.2.1. etcd

To ensure that the secrets that are stored in etcd use FIPS validated encryption, encrypt the etcd datastore by using a FIPS-approved cryptographic algorithm. After you install the cluster, you can encrypt the etcd data by using the **aes cbc** algorithm.

2.2.2. Storage

For local storage, use RHEL-provided disk encryption or Container Native Storage that uses RHEL-provided disk encryption. By storing all data in volumes that use RHEL-provided disk encryption and enabling FIPS mode for your cluster, both data at rest and data in motion, or network data, are protected by FIPS validated encryption.

2.2.3. Runtimes

To ensure that containers know that they are running on a host that has is using FIPS validated cryptography modules, use CRI-O to manage your runtimes. CRI-O supports FIPS-Mode, in that it configures the containers to know that they are running in FIPS mode.

2.3. INSTALLING A CLUSTER IN FIPS MODE

To install a cluster in FIPS mode, follow the instructions to install a customized cluster on your preferred infrastructure. Ensure that you set **fips: true** in the **install-config.yaml** file before you deploy your cluster.

- Amazon Web Services
- Microsoft Azure
- Bare metal
- Google Cloud Platform
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere

To apply **AES CBC** encryption to your etcd data store, follow the Encrypting etcd data process after you install your cluster.

If you add RHEL nodes to your cluster, ensure that you enable FIPS mode on the machines before their initial boot. See Adding RHEL compute machines to an OpenShift Container Platform cluster and Enabling FIPS Mode in the RHEL 7 documentation.

CHAPTER 3. INSTALLATION CONFIGURATION

3.1. INSTALLATION METHODS FOR DIFFERENT PLATFORMS

You can perform different types of installations on different platforms.

Table 3.1. Installer-provisioned infrastructure options

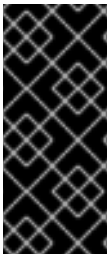
	AWS	Azure	GCP	OpenStack	Bare metal	vSphere	IBM Z
Default	X	X	X				
Custom	X	X	X	X			
Network Operator	X	X	X				
Private clusters	X	X	X				
Existing virtual private networks	X	X	X				

Table 3.2. User-provisioned infrastructure options

	AWS	Azure	GCP	OpenStack	Bare metal	vSphere	IBM Z
Custom	X		X		X	X	X
Network Operator					X	X	
Restricted network	X				X	X	

3.2. CREATING A MIRROR REGISTRY FOR INSTALLATION IN A RESTRICTED NETWORK

Before you install a cluster on infrastructure that you provision in a restricted network, you must create a mirror registry. Installations on a restricted network are supported on only infrastructure that you provision, not infrastructure that the installer provisions.



IMPORTANT

You must have access to the internet to obtain the data that populates the mirror repository. In this procedure, you place the mirror registry on a bastion host that has access to both your network and the internet. If you do not have access to a bastion host, use the method that best fits your restrictions to bring the contents of the mirror registry into your restricted network.

3.2.1. About the mirror registry

You can mirror the contents of the OpenShift Container Platform registry and the images that are required to generate the installation program.

The mirror registry is a key component that is required to complete an installation in a restricted network. You can create this mirror on a bastion host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.

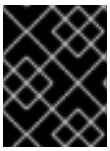
Because of the way that OpenShift Container Platform verifies integrity for the release payload, the image references in your local registry are identical to the ones that are hosted by Red Hat on Quay.io. During the bootstrapping process of installation, the images must have the same digests no matter which repository they are pulled from. To ensure that the release payload is identical, you mirror the images to your local repository.

3.2.2. Preparing the bastion host

Before you create the mirror registry, you must prepare the bastion host.

3.2.2.1. Installing the CLI

You can install the CLI in order to interact with OpenShift Container Platform using a command-line interface.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.3. Download and install the new version of **oc**.

Procedure

1. From the Infrastructure Provider page on the Red Hat OpenShift Cluster Manager site, navigate to the page for your installation type and click **Download Command-line Tools**.
2. Click the folder for your operating system and architecture and click the compressed file.

NOTE

You can install **oc** on Linux, Windows, or macOS.

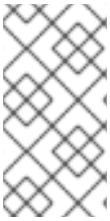
3. Save the file to your file system.
4. Extract the compressed file.
5. Place it in a directory that is on your **PATH**.

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

```
$ oc <command>
```

3.2.3. Creating a mirror registry

Create a registry to host the mirrored content that you require for installing OpenShift Container Platform. For installation in a restricted network, you must place the mirror on your bastion host.



NOTE

The following procedure creates a simple registry that stores data in the **/opt/registry** folder and runs in a **podman** container. You can use a different registry solution, such as Red Hat Quay. Review the following procedure to ensure that your registry functions correctly.

Prerequisites

- You have a Red Hat Enterprise Linux (RHEL) server on your network to use as the registry host.
- The registry host can access the internet.

Procedure

On the bastion host, take the following actions:

1. Install the required packages:

   ```
   # yum -y install podman httpd-tools
   ```

   The **podman** package provides the container package that you run the registry in. The **httpd-tools** package provides the **htpasswd** utility, which you use to create users.

2. Create folders for the registry:

   ```
   # mkdir -p /opt/registry/{auth,certs,data}
   ```

   These folders are mounted inside the registry container.

3. Provide a certificate for the registry. If you do not have an existing, trusted certificate authority, you can generate a self-signed certificate:

   ```
   $ cd /opt/registry/certs
   # openssl req -newkey rsa:4096 -nodes -sha256 -keyout domain.key -x509 -days 365 -out
   domain.crt
   ```

   At the prompts, provide the required values for the certificate:

Country Name (2 letter code)	Specify the two-letter ISO country code for your location. See the ISO 3166 country codes standard.

State or Province Name (full name)	Enter the full name of your state or province.
Locality Name (eg, city)	Enter the name of your city.
Organization Name (eg, company)	Enter your company name.
Organizational Unit Name (eg, section)	Enter your department name.
Common Name (eg, your name or your server's hostname)	Enter the host name for the registry host. Ensure that your hostname is in DNS and that it resolves to the expected IP address.
Email Address	Enter your email address. For more information, see the req description in the OpenSSL documentation.

4. Generate a user name and a password for your registry that uses the **bcrpt** format:

```
# htpasswd -bBc /opt/registry/auth/htpasswd <user_name> <password> 1
```

1 Replace **<user_name>** and **<password>** with a user name and a password.

5. Create the **mirror-registry** container to host your registry:

```
# podman run --name mirror-registry -p <local_registry_host_port>:5000 \ 1
    -v /opt/registry/data:/var/lib/registry:z \
    -v /opt/registry/auth:/auth:z \
    -e "REGISTRY_AUTH=htpasswd" \
    -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
    -e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
    -v /opt/registry/certs:/certs:z \
    -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt \
    -e REGISTRY_HTTP_TLS_KEY=/certs/domain.key \
    -d docker.io/library/registry:2
```

1 For **<local_registry_host_port>**, specify the port that your mirror registry uses to serve content.

6. Open the required ports for your registry:

```
# firewall-cmd --add-port=<local_registry_host_port>/tcp --zone=internal --permanent (1)
# firewall-cmd --add-port=<local_registry_host_port>/tcp --zone=public   --permanent (2)
# firewall-cmd --reload
```

1 2 For **<local_registry_host_port>**, specify the port that your mirror registry uses to serve content.

7. Add the self-signed certificate to your list of trusted certificates:

```
# cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
# update-ca-trust
```

You must trust your certificate to log in to your registry during the mirror process.

8. Confirm that the registry is available:

```
$ curl -u <user_name>:<password> -k https://<local_registry_host_name>:
<local_registry_host_port>/v2/_catalog (1)

{"repositories":[]}
```

1 For **<user_name>** and **<password>**, specify the user name and password for your registry. For **<local_registry_host_name>**, specify the registry domain name that you specified in your certificate, such as **registry.example.com**. For **<local_registry_host_port>**, specify the port that your mirror registry uses to serve content.

If the command output displays an empty repository, your registry is available.

3.2.4. Adding the registry to your pull secret

Modify your the pull secret for your OpenShift Container Platform cluster to describe your local registry before you install an OpenShift Container Platform cluster in a restricted network.

Prerequisites

- You configured a mirror registry to use in your restricted network.

Procedure

Complete the following steps on the bastion host:

1. Download your **registry.redhat.io** pull secret from the Pull Secret page on the Red Hat OpenShift Cluster Manager site.

2. Generate the base64-encoded user name and password or token for your mirror registry:

```
$ echo -n '<user_name>:<password>' | base64 -w0 (1)

BGVtbYk3ZHAtqXs=
```

(1) For **<user_name>** and **<password>**, specify the user name and password that you configured for your registry.

3. Make a copy of your pull secret in JSON format:

```
$ cat ./pull-secret.text | jq .  > <path>/<pull-secret-file> (1)
```

(1) Specify the path to the folder to store the pull secret in and a name for the JSON file that you create.

The contents of the file resemble the following example:

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

4. Edit the new file and add a section that describes your registry to it:

```
  "auths": {
...
    "<local_registry_host_name>:<local_registry_host_port>": { (1)
      "auth": "<credentials>", (2)
      "email": "you@example.com"
  },
...
```

(1) For **bastion_host_name**, specify the registry domain name that you specified in your certificate, and for **<local_registry_host_port>**, specify the port that your mirror registry uses to serve content.

(2) For **<credentials>**, specify the base64-encoded user name and password for the mirror registry that you generated.

The file resembles the following example:

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "<local_registry_host_name>:<local_registry_host_port>": {
      "auth": "<credentials>",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

3.2.5. Mirroring the OpenShift Container Platform image repository

Mirror the OpenShift Container Platform image repository to use during cluster installation or upgrade.

Prerequisites

- You configured a mirror registry to use in your restricted network and can access the certificate and credentials that you configured.
- You downloaded the pull secret from the Pull Secret page on the Red Hat OpenShift Cluster Manager site and modified it to include authentication to your mirror repository.

Procedure

Complete the following steps on the bastion host:

1. Review the OpenShift Container Platform downloads page to determine the version of OpenShift Container Platform that you want to install.
2. Set the required environment variables:

```
$ export OCP_RELEASE=<release_version> 1
$ export LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>' 2
$ export LOCAL_REPOSITORY='<repository_name>' 3
```

```
$ export PRODUCT_REPO='openshift-release-dev' 4
$ export LOCAL_SECRET_JSON='<path_to_pull_secret>' 5
$ export RELEASE_NAME="ocp-release" 6
```

1. For **<release_version>**, specify the version number of OpenShift Container Platform to install, such as **4.3.0**.
2. For **<local_registry_host_name>**, specify the registry domain name for your mirror repository, and for **<local_registry_host_port>**, specify the port that it serves content on.
3. For **<repository_name>**, specify the name of the repository to create in your registry, such as **ocp4/openshift4**.
4. The repository to mirror. For a production release, you must specify **openshift-release-dev**.
5. For **<path_to_pull_secret>**, specify the absolute path to and file name of the pull secret for your mirror registry that you created.
6. The release mirror. For a production release, you must specify **ocp-release**.

3. Mirror the repository:

```
$ oc adm -a ${LOCAL_SECRET_JSON} release mirror \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}
```

This command pulls the release information as a digest, and its output includes the **imageContentSources** data that you require when you install your cluster.

4. Record the entire **imageContentSources** section from the output of the previous command. The information about your mirrors is unique to your mirrored repository, and you must add the **imageContentSources** section to the **install-config.yaml** file during installation.

5. To create the installation program that is based on the content that you mirrored, extract it and pin it to the release:

```
$ oc adm -a ${LOCAL_SECRET_JSON} release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}"
```

IMPORTANT

To ensure that you use the correct images for the version of OpenShift Container Platform that you selected, you must extract the installation program from the mirrored content.

You must perform this step on a machine with an active internet connection.

3.2.6. Using sample imagestreams in a restricted network installation

Most imagestreams in the OpenShift namespace managed by the Samples Operator point to images located in the Red Hat registry at registry.redhat.io. Mirroring will not apply to these imagestreams.

The **jenkins**, **jenkins-agent-maven**, and **jenkins-agent-nodejs** imagestreams come from the install payload and are managed by the Samples Operator, so no further mirroring procedures are needed for those imagestreams.

NOTE

The **cli**, **installer**, **must-gather**, and **tests** imagestreams, while part of the install payload, are not managed by the Samples Operator. These are not addressed in this procedure.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- Create a pull secret for your mirror registry.

Procedure

1. Mirror images from registry.redhat.io associated with any imagestreams you need in the restricted network environment into one of the defined mirrors:

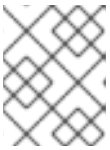
```
$ oc image mirror myregistry.com/myimage:latest myregistry.com/myimage:stable
```

2. Add the required trusted CAs for the mirror in the cluster's image configuration object:

```
$ oc create configmap registry-config --from-file=$path/ca.crt -n openshift-config
$ oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA":
{"name":"registry-config"}}}' --type=merge
```

3. Update the **samplesRegistry** field in the Samples Operator configuration object to contain the **hostname** portion of the mirror location defined in the mirror configuration:

```
$ oc get configs.samples.operator.openshift.io -n openshift-cluster-samples-operator
```



NOTE

This is required because the imagestream import process does not use the mirror or search mechanism at this time.

4. Add any imagestreams that are not mirrored into the **skippedImagestreams** field of the Samples Operator configuration object. Or if you do not want to support any of the sample imagestreams, set the Samples Operator to **Removed** in the Samples Operator configuration object.

NOTE

Any unmirrored imagestreams that are not skipped, or if the Samples Operator is not changed to **Removed**, will result in the Samples Operator reporting a **Degraded** status two hours after the imagestream imports start failing.

Many of the templates in the OpenShift namespace reference the imagestreams. So using **Removed** to purge both the imagestreams and templates will eliminate the possibility of attempts to use them if they are not functional because of any missing imagestreams.

Next steps

- Install a cluster on infrastructure that you provision in your restricted nework, such as on VMware vSphere, bare metal, or Amazon Web Services.

3.3. AVAILABLE CLUSTER CUSTOMIZATIONS

You complete most of the cluster configuration and customization after you deploy your OpenShift Container Platform cluster. A number of *configuration resources* are available.

You modify the configuration resources to configure the major features of the cluster, such as the image registry, networking configuration, image build behavior, and the identity provider.

For current documentation of the settings that you control by using these resources, use the **oc explain** command, for example **oc explain builds --api-version=config.openshift.io/v1**

3.3.1. Cluster configuration resources

All cluster configuration resources are globally scoped (not namespaced) and named **cluster**.

Resource name	Description
apiserver.config.openshift.io	Provides api-server configuration such as certificates and certificate authorities.
authentication.config.openshift.io	Controls the identity providerand authentication configuration for the cluster.
build.config.openshift.io	Controls default and enforced configuration for all builds on the cluster.
console.config.openshift.io	Configures the behavior of the web console interface, including the logout behavior.
featuregate.config.openshift.io	Enables FeatureGates so that you can use Tech Preview features.
image.config.openshift.io	Configures how specific image registries should be treated (allowed, disallowed, insecure, CA details).
ingress.config.openshift.io	Configuration details related to routing such as the default domain for routes.
oauth.config.openshift.io	Configures identity providers and other behavior related to internal OAuth server flows.
project.config.openshift.io	Configures how projects are createdincluding the project template.
proxy.config.openshift.io	Defines proxies to be used by components needing external network access. Note: not all components currently consume this value.

Resource name	Description
scheduler.config.openshift.io	Configures scheduler behavior such as policies and default nodeselectors.

3.3.2. Operator configuration resources

These configuration resources are cluster-scoped instances, named **cluster**, which control the behavior of a specific component as owned by a particular operator.

Resource name	Description
console.operator.openshift.io	Controls console appearance such as branding customizations
config.imageregistry.operator.openshift.io	Configures internal image registry settings such as public routing, log levels, proxy settings, resource constraints, replica counts, and storage type.
config.samples.operator.openshift.io	Configures the Samples Operator to control which example imagestreams and templates are installed on the cluster.

3.3.3. Additional configuration resources

These configuration resources represent a single instance of a particular component. In some cases, you can request multiple instances by creating multiple instances of the resource. In other cases, the Operator can use only a specific resource instance name in a specific namespace. Reference the component-specific documentation for details on how and when you can create additional resource instances.

Resource name	Instance name	Namespace	Description
alertmanager.monitoring.coreos.com	main	openshift-monitoring	Controls the alertmanager deployment parameters.
ingresscontroller.operator.openshift.io	default	openshift-ingress-operator	Configures Ingress Operator behavior such as domain, number of replicas, certificates, and controller placement.

3.3.4. Informational Resources

You use these resources to retrieve information about the cluster. Do not edit these resources directly.

Resource name	Instance name	Description
clusterversion.config.openshift.io	version	In OpenShift Container Platform 4.3, you must not customize the ClusterVersion resource for production clusters. Instead, follow the process to update a cluster.
dns.config.openshift.io	cluster	You cannot modify the DNS settings for your cluster. You can view the DNS Operator status.
infrastructure.config.openshift.io	cluster	Configuration details allowing the cluster to interact with its cloud provider.
network.config.openshift.io	cluster	You cannot modify your cluster networking after installation. To customize your network, follow the process to customize networking during installation.

3.4. CONFIGURING YOUR FIREWALL

If you use a firewall, you must configure it so that OpenShift Container Platform can access the sites that it requires to function. You must always grant access to some sites, and you grant access to more if you use Red Hat Insights, the Telemetry service, a cloud to host your cluster, and certain build strategies.

3.4.1. Configuring your firewall for OpenShift Container Platform

Before you install OpenShift Container Platform, you must configure your firewall to grant access to the sites that OpenShift Container Platform requires.

Procedure

1. Whitelist the following registry URLs:

URL	Function
registry.redhat.io	Provides core container images
***.quay.io**	Provides core container images
sso.redhat.com	The **https://cloud.redhat.com/openshift** site uses authentication from **sso.redhat.com**

2. Whitelist any site that provides resources for a language or framework that your builds require.

3. If you do not disable Telemetry, you must grant access to the following URLs to access Red Hat Insights:

URL	Function
cert-api.access.redhat.com	Required for Telemetry
api.access.redhat.com	Required for Telemetry
infogw.api.openshift.com	Required for Telemetry

4. If you use Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP) to host your cluster, you must grant access to the URLs that provide the cloud provider API and DNS for that cloud:

Cloud	URL	Function
AWS	***.amazonaws.com**	Required to access AWS services and resources. Review the AWS Service Endpoints in the AWS documentation to determine the exact endpoints to allow for the regions that you use.
GCP	***.googleapis.com**	Required to access GCP services and resources. Review Cloud Endpoints in the GCP documentation to determine the endpoints to allow for your APIs.
	accounts.google.com	Required to access your GCP account.
Azure	**management.azure.com**	Required to access Azure services and resources. Review the Azure REST API Reference in the Azure documentation to determine the endpoints to allow for your APIs.

5. Whitelist the following URLs:

URL	Function
mirror.openshift.com	Required to access mirrored installation content and images
***.cloudfront.net**	Required by the Quay CDN to deliver the Quay.io images that the cluster requires
***.apps.<cluster_name>.<base_domain>**	Required to access the default cluster routes unless you set an ingress wildcard during installation
quay-registry.s3.amazonaws.com	Required to access Quay image content in AWS

URL	Function
api.openshift.com	Required to check if updates are available for the cluster
art-rhcos-ci.s3.amazonaws.com	Required to download Red Hat Enterprise Linux CoreOS (RHCOS) images
api.openshift.com	Required for your cluster token
cloud.redhat.com/openshift	Required for your cluster token