OpenShift Container Platform 4.5

Support

Getting support for OpenShift Container Platform

Last Updated: 2020-09-24

OpenShift Container Platform 4.5 Support

Getting support for OpenShift Container Platform

Legal Notice

Abstract

This document provides information on getting support from Red Hat for OpenShift Container Platform. It also contains information on remote health monitoring through Telemetry and the Insights Operator.

Table of Contents

CHAPTER 1. GETTING SUPPORT

1.1. GETTING SUPPORT

If you experience difficulty with a procedure described in this documentation, or with OpenShift Container Platform in general, visit the Red Hat Customer Portal . From the Customer Portal, you can:

- Search or browse through the Red Hat Knowledgebase of articles and solutions relating to Red Hat products.
- Submit a support case to Red Hat Support.
- Access other product documentation.

If you have a suggestion for improving this documentation or have found an error, please submit a Bugzilla report against the **OpenShift Container Platform** product for the **Documentation** component. Please provide specific details, such as the section name and OpenShift Container Platform version.

1.2. ABOUT THE RED HAT KNOWLEDGEBASE

The Red Hat Knowledgebase provides rich content aimed at helping you make the most of Red Hat's products and technologies. The Red Hat Knowledgebase consists of articles, product documentation, and videos outlining best practices on installing, configuring, and using Red Hat products. In addition, you can search for solutions to known issues, each providing concise root cause descriptions and remedial steps.

1.3. SEARCHING THE RED HAT KNOWLEDGEBASE

In the event of an OpenShift Container Platform issue, you can perform an initial search to determine if a solution already exists within the Red Hat Knowledgebase.

Prerequisites

- You have a Red Hat Customer Portal account.

Procedure

1. Log in to the Red Hat Customer Portal .
2. In the main Red Hat Customer Portal search field, input keywords and strings relating to the problem, including:
   - OpenShift Container Platform components (such as **etcd**)
   - Related procedure (such as **installation**)
   - Warnings, error messages, and other outputs related to explicit failures
3. Click **Search**.
4. Select the **OpenShift Container Platform** product filter.
5. Select the **Knowledgebase** content type filter.

1.4. SUBMITTING A SUPPORT CASE

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- You have a Red Hat Customer Portal account.
- You have a Red Hat standard or premium Subscription.

Procedure

1. Log in to the Red Hat Customer Portal and select **SUPPORT CASES → Open a case**.
2. Select the appropriate category for your issue (such as **Defect / Bug**), product (**OpenShift Container Platform**), and product version (**4.5**, if this is not already autofilled).
3. Review the list of suggested Red Hat Knowledgebase solutions for a potential match against the problem that is being reported. If the suggested articles do not address the issue, click **Continue**.
4. Enter a concise but descriptive problem summary and further details about the symptoms being experienced, as well as your expectations.
5. Review the updated list of suggested Red Hat Knowledgebase solutions for a potential match against the problem that is being reported. The list is refined as you provide more information during the case creation process. If the suggested articles do not address the issue, click **Continue**.
6. Ensure that the account information presented is as expected, and if not, amend accordingly.
7. Check that the autofilled OpenShift Container Platform Cluster ID is correct. If it is not, manually obtain your cluster ID.
   - To manually obtain your cluster ID using the OpenShift Container Platform web console:
     a. Navigate to **Home → Dashboards → Overview**.
     b. Find the value in the **Cluster ID** field of the **Details** section.
   - Alternatively, it is possible to open a new support case through the OpenShift Container Platform web console and have your cluster ID autofilled.
     a. From the toolbar, navigate to **(?) Help → Open Support Case**.
     b. The **Cluster ID** value is autofilled.
   - To obtain your cluster ID using the OpenShift CLI (**oc**), run the following command:

     ```
     $ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}{"\n"}'
     ```
8. Complete the following questions where prompted and then click **Continue**:
   - Where are you experiencing the behavior? What environment?

- When does the behavior occur? Frequency? Repeatedly? At certain times?
- What information can you provide around time-frames and the business impact?

9. Upload relevant diagnostic data files and click **Continue**. It is recommended to include data gathered using the **oc adm must-gather** command as a starting point, plus any issue specific data that is not collected by that command.
10. Input relevant case management details and click **Continue**.
11. Preview the case details and click **Submit**.

CHAPTER 2. GATHERING DATA ABOUT YOUR CLUSTER

When opening a support case, it is often helpful to provide debugging information about your cluster to Red Hat Support.

It is recommended to provide:

- Data gathered using the **oc adm must-gather** command
- The unique cluster ID

2.1. ABOUT THE MUST-GATHER TOOL

The **oc adm must-gather** CLI command collects the information from your cluster that is most likely needed for debugging issues, such as:

- Resource definitions
- Audit logs
- Service logs

You can specify one or more images when you run the command by including the **--image** argument. When you specify an image, the tool collects data related to that feature or product.

When you run **oc adm must-gather**, a new Pod is created on the cluster. The data is collected on that Pod and saved in a new directory that starts with **must-gather.local**. This directory is created in the current working directory.

2.2. GATHERING DATA ABOUT YOUR CLUSTER FOR RED HAT SUPPORT

You can gather debugging information about your cluster by using the **oc adm must-gather** CLI command.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- The OpenShift Container Platform CLI (**oc**) installed.

Procedure

1. Navigate to the directory where you want to store the **must-gather** data.
2. Run the **oc adm must-gather** command:

```
$ oc adm must-gather
```

NOTE

If this command fails, for example if you cannot schedule a Pod on your cluster, then use the **oc adm inspect** command to gather information for particular resources. Contact Red Hat Support for the recommended resources to gather.

NOTE

If your cluster is using a restricted network you must import the default must-gather image before running the **oc adm must-gather** command.

```
$ oc import-image is/must-gather -n openshift
```

3. Create a compressed file from the **must-gather** directory that was just created in your working directory. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar cvaf must-gather.tar.gz must-gather.local.5421342344627712289/ 1
```

1 Make sure to replace **must-gather-local.5421342344627712289/** with the actual directory name.

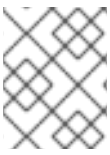
4. Attach the compressed file to your support case on the Red Hat Customer Portal .

2.3. GATHERING DATA ABOUT SPECIFIC FEATURES

You can gather debugging information about specific features by using the **oc adm must-gather** CLI command with the **--image** or **--image-stream** argument. The **must-gather** tool supports multiple images, so you can gather data about more than one feature by running a single command.

Table 2.1. Supported must-gather images

Image	Purpose
registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v2.4.1	Data collection for OpenShift Virtualization.
registry.redhat.io/openshift-serverless-1/svls-must-gather-rhel8	Data collection for OpenShift Serverless.
registry.redhat.io/openshift-service-mesh/istio-must-gather-rhel7	Data collection for Red Hat OpenShift Service Mesh.
registry.redhat.io/rhcam-1-2/openshift-migration-must-gather-rhel8	Data collection for migration-related information.
registry.redhat.io/ocs4/ocs-must-gather-rhel8	Data collection for Red Hat OpenShift Container Storage.



NOTE

To collect the default must-gather data in addition to specific feature data, add the **--image-stream=openshift/must-gather** argument.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- The OpenShift Container Platform CLI (**oc**) installed.

Procedure

1. Navigate to the directory where you want to store the **must-gather** data.
2. Run the **oc adm must-gather** command with one or more **--image** or **--image-stream** arguments. For example, the following command gathers both the default cluster data and information specific to OpenShift Virtualization:

```
$ oc adm must-gather \
 --image-stream=openshift/must-gather \ 1
 --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v2.4.1 2
```

1. The default OpenShift Container Platform must-gather image
2. The must-gather image for OpenShift Virtualization

3. Create a compressed file from the **must-gather** directory that was just created in your working directory. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar cvaf must-gather.tar.gz must-gather.local.5421342344627712289/ 1
```

1. Make sure to replace **must-gather-local.5421342344627712289/** with the actual directory name.

4. Attach the compressed file to your support case on the Red Hat Customer Portal .

2.4. OBTAINING YOUR CLUSTER ID

When providing information to Red Hat Support, it is helpful to provide the unique identifier for your cluster. You can have your cluster ID autofilled by using the OpenShift Container Platform web console. You can also manually obtain your cluster ID by using the web console or the OpenShift CLI (**oc**).

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- Access to the web console or the OpenShift CLI (**oc**) installed.

Procedure

- To open a support case and have your cluster ID autofilled using the web console:
  a. From the toolbar, navigate to **(?) Help → Open Support Case**.
  b. The **Cluster ID** value is autofilled.
- To manually obtain your cluster ID using the web console:

a. Navigate to **Home → Dashboards → Overview**.

b. The value is available in the **Cluster ID** field of the **Details** section.

- To obtain your cluster ID using the OpenShift CLI (**oc**), run the following command:

```
$ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}{"\n"}'
```

2.5. ABOUT SOSREPORT

sosreport is a tool that collects configuration details, system information, and diagnostic data from Red Hat Enterprise Linux (RHEL) and Red Hat Enterprise Linux CoreOS (RHCOS) systems. **sosreport** provides a standardized way to collect diagnostic information relating to a node, which can then be provided to Red Hat Support for issue diagnosis.

In some support interactions, Red Hat Support may ask you to collect a **sosreport** archive for a specific OpenShift Container Platform node. For example, it might sometimes be necessary to review system logs or other node-specific data that is not included within the output of **oc adm must-gather**.

2.6. GENERATING A SOSREPORT ARCHIVE FOR AN OPENSHIFT CONTAINER PLATFORM CLUSTER NODE

The recommended way to generate a **sosreport** for an OpenShift Container Platform 4.5 cluster node is through a debug Pod.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have SSH access to your hosts.
- You have installed the OpenShift CLI (**oc**).
- You have a Red Hat standard or premium Subscription.
- You have a Red Hat Customer Portal account.
- You have an existing Red Hat Support case ID.

Procedure

1. Obtain a list of cluster nodes:

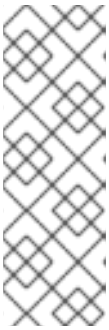
```
$ oc get nodes
```

2. Enter into a debug session on the target node. This step instantiates a debug Pod called **<node_name>-debug**:

```
$ oc debug node/my-cluster-node
```

3. Set **/host** as the root directory within the debug shell. The debug Pod mounts the host's root file system in **/host** within the Pod. By changing the root directory to **/host**, you can run binaries contained in the host's executable paths:

```
# chroot /host
```



NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>** instead.

4. Start a **toolbox** container, which includes the required binaries and plug-ins to run **sosreport**:

```
# toolbox
```

NOTE

If an existing **toolbox** Pod is already running, the **toolbox** command outputs **'toolbox-' already exists. Trying to start...**. Remove the running toolbox container with **podman rm toolbox-** and spawn a new toolbox container, to avoid issues with **sosreport** plugins.

5. Collect a **sosreport** archive.

   a. Run the **sosreport** command and enable the **crio.all** and **crio.logs** CRI-O container engine **sosreport** plug-ins:

```
# sosreport -k crio.all=on -k crio.logs=on (1)
```

   (1) **-k** enables you to define **sosreport** plug-in parameters outside of the defaults.

   b. Press **Enter** when prompted, to continue.

   c. Provide the Red Hat Support case ID. **sosreport** adds the ID to the archive's file name.

   d. The **sosreport** output provides the archive's location and checksum. The following sample output references support case ID **01234567**:

```
Your sosreport has been generated and saved in:
  /host/var/tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz (1)

The checksum is: 382ffc167510fd71b4f12a4f40b97a4e
```

   (1) The **sosreport** archive's file path is outside of the **chroot** environment because the toolbox container mounts the host's root directory at **/host**.

6. Provide the **sosreport** archive to Red Hat Support for analysis, using one of the following methods.

   - Upload the file to an existing Red Hat support case directly from an OpenShift Container Platform cluster.

platform cluster.

a. From within the toolbox container, run **redhat-support-tool** to attach the archive directly to an existing Red Hat support case. This example uses support case ID **01234567**:

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-sosreport.tar.xz
```
(1)

(1) The toolbox container mounts the host's root directory at **/host**. Reference the absolute path from the toolbox container's root directory, including **/host/**, when specifying files to upload through the **redhat-support-tool** command.

- Upload the file to an existing Red Hat support case.

a. Concatenate the **sosreport** archive by running the **oc debug node/<node_name>** command and redirect the output to a file. This command assumes you have exited the previous **oc debug** session:

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz' > /tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz
```
(1)

(1) The debug container mounts the host's root directory at **/host**. Reference the absolute path from the debug container's root directory, including **/host**, when specifying target files for concatenation.

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Transferring a **sosreport** archive from a cluster node by using **scp** is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to copy a **sosreport** archive from a node by running **scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>**.

b. Navigate to an existing support case within https://access.redhat.com/support/cases/.

c. Select **Attach files** and follow the prompts to upload the file.

2.7. QUERYING BOOTSTRAP NODE JOURNAL LOGS

If you experience bootstrap-related issues, you can gather **bootkube.service journald** unit logs and container logs from the bootstrap node.

Prerequisites

- You have SSH access to your bootstrap node.
- You have the fully qualified domain name of the bootstrap node.

Procedure

1. Query **bootkube.service journald** unit logs from a bootstrap node during OpenShift Container Platform installation. Replace **<bootstrap_fqdn>** with the bootstrap node's fully qualified domain name:

```
$ ssh core<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```

NOTE

The **bootkube.service** log on the bootstrap node outputs etcd **connection refused** errors, indicating that the bootstrap server is unable to connect to etcd on master nodes. After etcd has started on each master node and the nodes have joined the cluster, the errors should stop.

2. Collect logs from the bootstrap node containers using **podman** on the bootstrap node. Replace **<bootstrap_fqdn>** with the bootstrap node's fully qualified domain name:

```
$ ssh core@<bootstrap_fqdn> 'for pod in $(sudo podman ps -a -q); do sudo podman logs
$pod; done'
```

2.8. QUERYING CLUSTER NODE JOURNAL LOGS

You can gather **journald** unit logs and other logs within **/var/log** on individual cluster nodes.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).
- You have SSH access to your hosts.

Procedure

1. Query **kubelet journald** unit logs from OpenShift Container Platform cluster nodes. The following example queries master nodes only:

```
$ oc adm node-logs --role=master -u kubelet 1
```

1 Replace **kubelet** as appropriate to query other unit logs.

2. Collect logs from specific subdirectories under **/var/log/** on cluster nodes.

   a. Retrieve a list of logs contained within a **/var/log/** subdirectory. The following example lists files in **/var/log/openshift-apiserver/** on all master nodes:

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

b. Inspect a specific log within a **/var/log/** subdirectory. The following example outputs **/var/log/openshift-apiserver/audit.log** contents from all master nodes:

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

c. If the API is not functional, review the logs on each node using SSH instead. The following example tails **/var/log/openshift-apiserver/audit.log**:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
/var/log/openshift-apiserver/audit.log
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. Before attempting to collect diagnostic data over SSH, review whether the data collected by running **oc adm must gather** and other **oc** commands is sufficient instead. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>**.

2.9. COLLECTING A NETWORK TRACE FROM AN OPENSHIFT CONTAINER PLATFORM NODE OR CONTAINER

When investigating potential network-related OpenShift Container Platform issues, Red Hat Support might request a network packet trace from a specific OpenShift Container Platform cluster node or from a specific container. The recommended method to capture a network trace in OpenShift Container Platform is through a debug Pod.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- You have a Red Hat standard or premium Subscription.
- You have a Red Hat Customer Portal account.
- You have an existing Red Hat Support case ID.
- You have SSH access to your hosts.

Procedure

1. Obtain a list of cluster nodes:

```
$ oc get nodes
```

2. Enter into a debug session on the target node. This step instantiates a debug Pod called **<node_name>-debug**:

```
$ oc debug node/my-cluster-node
```

3. Set **/host** as the root directory within the debug shell. The debug Pod mounts the host's root file system in **/host** within the Pod. By changing the root directory to **/host**, you can run binaries contained in the host's executable paths:

```
# chroot /host
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>** instead.

4. From within the **chroot** environment console, obtain the node's interface names:

```
# ip ad
```

5. Start a **toolbox** container, which includes the required binaries and plug-ins to run **sosreport**:

```
# toolbox
```

NOTE

If an existing **toolbox** Pod is already running, the **toolbox** command outputs **'toolbox-' already exists. Trying to start…**. To avoid **tcpdump** issues, remove the running toolbox container with **podman rm toolbox-** and spawn a new toolbox container.

6. Initiate a **tcpdump** session on the cluster node and redirect output to a capture file. This example uses **ens5** as the interface name:

```
$ tcpdump -nn -s 0 -i ens5 -w /host/var/tmp/my-cluster-node_$(date +%d_%m_%Y-%H_%M_%S-%Z).pcap 1
```

1 The **tcpdump** capture file's path is outside of the **chroot** environment because the toolbox container mounts the host's root directory at **/host**.

7. If a **tcpdump** capture is required for a specific container on the node, follow these steps.

   a. Determine the target container ID. The **chroot host** command precedes the **crictl** command in this step because the toolbox container mounts the host's root directory at **/host**:

```
# chroot /host crictl ps
```

b. Determine the container's process ID. In this example, the container ID is **a7fe32346b120**:

```
# chroot /host crictl inspect --output yaml a7fe32346b120 | grep 'pid' | awk '{print $2}'
```

c. Initiate a **tcpdump** session on the container and redirect output to a capture file. This example uses **49628** as the container's process ID and **ens5** as the interface name. The **nsenter** command enters the namespace of a target process and runs a command in its namespace. because the target process in this example is a container's process ID, the **tcpdump** command is run in the container's namespace from the host:

```
# nsenter -n -t 49628 -- tcpdump -nn -i ens5 -w /host/var/tmp/my-cluster-node-my-container_$(date +%d_%m_%Y-%H_%M_%S-%Z).pcap.pcap (1)
```

(1) The **tcpdump** capture file's path is outside of the **chroot** environment because the toolbox container mounts the host's root directory at **/host**.

8. Provide the **tcpdump** capture file to Red Hat Support for analysis, using one of the following methods.

- Upload the file to an existing Red Hat support case directly from an OpenShift Container Platform cluster.

a. From within the toolbox container, run **redhat-support-tool** to attach the file directly to an existing Red Hat Support case. This example uses support case ID **01234567**:

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-tcpdump-capture-file.pcap (1)
```

(1) The toolbox container mounts the host's root directory at **/host**. Reference the absolute path from the toolbox container's root directory, including **/host/**, when specifying files to upload through the **redhat-support-tool** command.

- Upload the file to an existing Red Hat support case.

a. Concatenate the **sosreport** archive by running the **oc debug node/<node_name>** command and redirect the output to a file. This command assumes you have exited the previous **oc debug** session:

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/my-tcpdump-capture-file.pcap' > /tmp/my-tcpdump-capture-file.pcap (1)
```

(1) The debug container mounts the host's root directory at **/host**. Reference the absolute path from the debug container's root directory, including **/host**, when specifying target files for concatenation.

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Transferring a **tcpdump** capture file from a cluster node by using **scp** is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to copy a **tcpdump** capture file from a node by running **scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>**.

b. Navigate to an existing support case within https://access.redhat.com/support/cases/.

c. Select **Attach files** and follow the prompts to upload the file.

2.10. PROVIDING DIAGNOSTIC DATA TO RED HAT SUPPORT

When investigating OpenShift Container Platform issues, Red Hat Support might ask you to upload diagnostic data to a support case. Files can be uploaded to a support case through the Red Hat Customer Portal, or from an OpenShift Container Platform cluster directly by using the **redhat-support-tool** command.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have SSH access to your hosts.
- You have installed the OpenShift CLI (**oc**).
- You have a Red Hat standard or premium Subscription.
- You have a Red Hat Customer Portal account.
- You have an existing Red Hat Support case ID.

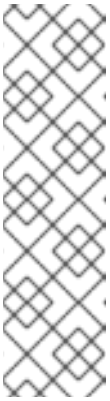
Procedure

- Upload diagnostic data to an existing Red Hat support case through the Red Hat Customer Portal.

1. Concatenate a diagnostic file contained on an OpenShift Container Platform node by using the **oc debug node/<node_name>** command and redirect the output to a file. The following example copies **/host/var/tmp/my-diagnostic-data.tar.gz** from a debug container to **/var/tmp/my-diagnostic-data.tar.gz**:

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/my-diagnostic-data.tar.gz'
> /var/tmp/my-diagnostic-data.tar.gz 1
```

1 The debug container mounts the host's root directory at **/host**. Reference the absolute path from the debug container's root directory, including **/host**, when specifying target files for concatenation.



NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Transferring files from a cluster node by using **scp** is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to copy diagnostic files from a node by running **scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>**.

2. Navigate to an existing support case within https://access.redhat.com/support/cases/.
3. Select **Attach files** and follow the prompts to upload the file.

- Upload diagnostic data to an existing Red Hat support case directly from an OpenShift Container Platform cluster.

1. Obtain a list of cluster nodes:

```
$ oc get nodes
```

2. Enter into a debug session on the target node. This step instantiates a debug Pod called **<node_name>-debug**:

```
$ oc debug node/my-cluster-node
```

3. Set **/host** as the root directory within the debug shell. The debug Pod mounts the host's root file system in **/host** within the Pod. By changing the root directory to **/host**, you can run binaries contained in the host's executable paths:

```
# chroot /host
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>** instead.

4. Start a **toolbox** container, which includes the required binaries to run **redhat-support-tool**:

```
# toolbox
```

NOTE

If an existing **toolbox** Pod is already running, the **toolbox** command outputs **'toolbox-' already exists. Trying to start…**. Remove the running toolbox container with **podman rm toolbox-** and spawn a new toolbox container, to avoid issues.

a. Run **redhat-support-tool** to attach a file from the debug Pod directly to an existing Red Hat Support case. This example uses support case ID '01234567' and example file path **/host/var/tmp/my-diagnostic-data.tar.gz**:

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-diagnostic-data.tar.gz 1
```

1 The toolbox container mounts the host's root directory at **/host**. Reference the absolute path from the toolbox container's root directory, including **/host/**, when specifying files to upload through the **redhat-support-tool** command.

CHAPTER 3. SUMMARIZING CLUSTER SPECIFICATIONS

3.1. SUMMARIZING CLUSTER SPECIFICATIONS THROUGH CLUSTERVERSION

You can obtain a summary of OpenShift Container Platform cluster specifications by querying the **clusterversion** resource.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Query cluster version, availability, uptime, and general status:

```
$ oc get clusterversion
```

2. Obtain a detailed summary of cluster specifications, update availability, and update history:

```
$ oc describe clusterversion
```

CHAPTER 4. REMOTE HEALTH MONITORING WITH CONNECTED CLUSTERS

4.1. ABOUT REMOTE HEALTH MONITORING

OpenShift Container Platform collects anonymized aggregated information about the health, usage, and size of clusters and reports it to Red Hat via two integrated components: Telemetry and the Insights Operator. This information allows Red Hat to improve OpenShift Container Platform and to react to issues that impact customers more quickly. This also simplifies the subscription and entitlement process for Red Hat customers and enables the Red Hat OpenShift Cluster Manager service to provide an overview of your clusters and their health and subscription status.

A cluster that reports data to Red Hat via Telemetry and the Insights Operator is considered a *connected cluster*.

4.1.1. About Telemetry

Telemetry sends a carefully chosen subset of the cluster monitoring metrics to Red Hat. These metrics are sent continuously and describe:

- The size of an OpenShift Container Platform cluster
- The health and status of OpenShift Container Platform components
- The health and status of any upgrade being performed
- Limited usage information about OpenShift Container Platform components and features
- Summary info about alerts reported by the cluster monitoring component

This continuous stream of data is used by Red Hat to monitor the health of clusters in real time and to react as necessary to problems that impact our customers. It also allows Red Hat to roll out OpenShift Container Platform upgrades to customers so as to minimize service impact and continuously improve the upgrade experience.

This debugging information is available to Red Hat Support and engineering teams with the same restrictions as accessing data reported via support cases. All connected cluster information is used by Red Hat to help make OpenShift Container Platform better and more intuitive to use. None of the information is shared with third parties.

4.1.1.1. Information collected by Telemetry

Primary information collected by Telemetry includes:

- The number of updates available per cluster
- Channel and image repository used for an update
- The number of errors that occurred during an update
- Progress information of running updates
- The number of machines per cluster
- The number of CPU cores and size of RAM of the machines

- The number of members in the etcd cluster and number of objects currently stored in the etcd cluster
- The number of CPU cores and RAM used per machine type - infra or master
- The number of CPU cores and RAM used per cluster
- Use of OpenShift Container Platform framework components per cluster
- The version of the OpenShift Container Platform cluster
- Health, condition, and status for any OpenShift Container Platform framework component that is installed on the cluster, for example Cluster Version Operator, Cluster Monitoring, Image Registry, and Elasticsearch for Logging
- A unique random identifier that is generated during installation
- The name of the platform that OpenShift Container Platform is deployed on, such as Amazon Web Services

Telemetry does not collect identifying information such as user names, passwords, or the names or addresses of user resources.

4.1.2. About the Insights Operator

The Insights Operator periodically gathers anonymized configuration and component failure status and reports that to Red Hat. This is a subset of the information captured by the **must-gather** tool and allows Red Hat to assess important configuration and deeper failure data than is reported via Telemetry. This data is sent several times a day and describes:

- Important configuration information about the environment that the cluster runs in
- Details about the state of the cluster and its major components
- Debugging information about infrastructure Pods or nodes that are reporting failures

This debugging information is available to Red Hat Support and engineering teams with the same restrictions as accessing data reported via support cases. All connected cluster information is used by Red Hat to help make OpenShift Container Platform better and more intuitive to use. None of the information is shared with third parties.

4.1.2.1. Information collected by the Insights Operator

Primary information collected by the Insights Operator includes:

- The version of the cluster and its components, as well as the unique cluster identifier
- Channel and image repository used for an update
- Details about errors that have occurred in the cluster components
- Progress and health information of running updates and the status of any component upgrades
- Anonymized details about the cluster configuration that is relevant to Red Hat Support
- Details about any Technology Preview or unsupported configurations that might impact Red Hat Support

- Details of the platform that OpenShift Container Platform is deployed on, such as Amazon Web Services, and the region that the cluster is located in
- Information about Pods of degraded OpenShift Container Platform cluster Operators
- Information about nodes marked as **NotReady**
- Events for all name spaces listed as "related objects" for Degraded operator
- Anonymized certificate signing requests (CSR) and information about the validity of certificates
- Configuration settings stored in **ConfigMap** objects in the **openshift-config** name space
- The image pruner configuration, as well as how often OpenShift Container Platform prunes images and the status of pruning jobs.
- The image registry configuration, such as where OpenShift Container Platform stores images.

The Insights Operator does not collect identifying information such as user names, passwords, certificates, or the names or addresses of user resources.

4.2. SHOWING DATA COLLECTED BY REMOTE HEALTH MONITORING

As an administrator, you can review the metrics collected by Telemetry and the Insights Operator.

4.2.1. Showing data collected by Telemetry

You can see the cluster and components time series data captured by Telemetry.

Prerequisites

- Install the OpenShift CLI (**oc**).
- You must log in to the cluster with a user that has either the **cluster-admin** role or the **cluster-monitoring-view** role.

Procedure

1. Find the URL for the Prometheus service that runs in the OpenShift Container Platform cluster:

```
$ oc get route prometheus-k8s -n openshift-monitoring -o jsonpath="{.spec.host}"
```

2. Navigate to the URL.
3. Enter this query in the **Expression** input box and press **Execute**:

```
{__name__=~"cluster:usage:.*|count:up0|count:up1|cluster_version|cluster_version_available_u
pdates|cluster_operator_up|cluster_operator_conditions|cluster_version_payload|cluster_install
er|cluster_infrastructure_provider|cluster_feature_set|instance:etcd_object_counts:sum|ALERT
S|code:apiserver_request_total:rate:sum|cluster:capacity_cpu_cores:sum|cluster:capacity_mem
ory_bytes:sum|cluster:cpu_usage_cores:sum|cluster:memory_usage_bytes:sum|openshift:cpu_
usage_cores:sum|openshift:memory_usage_bytes:sum|workload:cpu_usage_cores:sum|worklo
ad:memory_usage_bytes:sum|cluster:virt_platform_nodes:sum|cluster:node_instance_type_cou
nt:sum|cnv:vmi_status_running:count|node_role_os_version_machine:cpu_capacity_cores:sum
node_role_os_version_machine:cpu_capacity_sockets:sum|subscription_sync_total|csv_succee
```

```
ded|csv_abnormal|ceph_cluster_total_bytes|ceph_cluster_total_used_raw_bytes|ceph_health_s
tatus|job:ceph_osd_metadata:count|job:kube_pv:count|job:ceph_pools_iops:total|job:ceph_pool
s_iops_bytes:total|job:ceph_versions_running:count|job:noobaa_total_unhealthy_buckets:sum|j
b:noobaa_bucket_count:sum|job:noobaa_total_object_count:sum|noobaa_accounts_num|noob
aa_total_usage|console_url|cluster:network_attachment_definition_instances:max|cluster:netwo
rk_attachment_definition_enabled_instance_up:max|insightsclient_request_send_total|cam_app
_workload_migrations|cluster:apiserver_current_inflight_requests:sum:max_over_time:2m|clust
er:telemetry_selected_series:count",alertstate=~"firing|"}
```

This query replicates the request that Telemetry makes against a running OpenShift Container Platform cluster's Prometheus service and returns the full set of time series captured by Telemetry.

4.2.2. Showing data collected by the Insights Operator

You can review the data that is collected by the Insights Operator.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.

Procedure

1. Find the name of the currently running Pod for the Insights Operator:

```
$ INSIGHTS_OPERATOR_POD=$(oc get pods --namespace=openshift-insights -o custom-
columns=:metadata.name --no-headers  --field-selector=status.phase=Running)
```

2. Copy the recent data archives collected by the Insights Operator:

```
$ oc cp openshift-insights/$INSIGHTS_OPERATOR_POD:/var/lib/insights-operator ./insights-
data
```

The recent Insights Operator archives are now available in the **insights-data** directory.

4.3. OPTING OUT OF REMOTE HEALTH REPORTING

You might need to opt out of reporting health and usage data for your cluster. For example, you might need to comply with privacy laws or standards governing how your organization reports monitoring data.

To opt out of remote health reporting, you must:

1. Modify the global cluster pull secret to disable remote health reporting.
2. Update the cluster to use this modified pull secret.

4.3.1. Consequences of disabling remote health reporting

In OpenShift Container Platform, customers can opt out of reporting health and usage information. However, connected clusters allow Red Hat to react more quickly to problems and better support our customers, as well as better understand how product upgrades impact clusters.

Red Hat strongly recommends leaving health and usage reporting enabled for pre-production and test clusters even if it is necessary to opt out for production clusters. This allows Red Hat to be a participant

in qualifying OpenShift Container Platform in your environments and react more rapidly to product issues.

Some of the consequences of opting out of having a connected cluster are:

- Red Hat will not be able to monitor the success of product upgrades or the health of your clusters without a support case being opened.
- Red Hat will not be able to use anonymized configuration data to better triage customer support cases and identify which configurations our customers find important.
- The Red Hat OpenShift Cluster Manager will not show data about your clusters including health and usage information.
- Your subscription entitlement information must be manually entered via cloud.redhat.com without the benefit of automatic usage reporting.

In restricted networks, Telemetry and Insights data can still be reported through appropriate configuration of your proxy.

4.3.2. Modifying the global cluster pull secret to disable remote health reporting

You can modify your existing global cluster pull secret to disable remote health reporting. This disables both Telemetry and the Insights Operator.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

1. Download the global cluster pull secret to your local file system.

```
$ oc extract secret/pull-secret -n openshift-config --to=.
```

2. In a text editor, edit the **.dockerconfigjson** file that was downloaded.
3. Remove the **cloud.openshift.com** JSON entry, for example:

```
"cloud.openshift.com":{"auth":"<hash>","email":"<email_address>"}
```

4. Save the file.

You can now update your cluster to use this modified pull secret.

4.3.3. Updating the global cluster pull secret

You can update the global pull secret for your cluster.

WARNING

Cluster resources must adjust to the new pull secret, which can temporarily limit the usability of the cluster.

Prerequisites

- You have a new or modified pull secret file to upload.
- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

- Run the following command to update the global pull secret for your cluster:

```
$ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=<pull-secret-location> 1
```

1 Provide the path to the new pull secret file.

This update is rolled out to all nodes, which can take some time depending on the size of your cluster. During this time, nodes are drained and Pods are rescheduled on the remaining nodes.

4.4. USING INSIGHTS TO IDENTIFY ISSUES WITH YOUR CLUSTER

Insights repeatedly analyzes the data Insights Operator sends. Users of OpenShift Container Platform can display the report on the **Insights** tab of each cluster in Red Hat OpenShift Cluster Manager.

4.4.1. Displaying potential issues with your cluster

This section describes how to display the Insights report in the Red Hat OpenShift Cluster Manager.

Note that Insights repeatedly analyzes your cluster and shows the latest results. These results can change, for example, if you fix an issue or a new issue has been detected.

Prerequisites

- Your cluster is registered in the Red Hat OpenShift Cluster Manager.
- Remote health reporting is enabled, which is the default.
- You are logged in to the Red Hat OpenShift Cluster Manager .

Procedure

1. Click the **Clusters** menu in the left pane.
2. Click the cluster's name to display the details of the cluster.

3. Open the **Insights** tab of the cluster.
   Depending on the result, the tab displays one of the following:

   - **Your cluster passed all health checks**, if Insights did not identify any issues.
   - A list of issues Insights has detected, prioritized by risk (low, moderate, important, and critical).
   - **No health checks to display**, if Insights has not yet analyzed the cluster. The analysis starts shortly after the cluster has been installed and connected to the internet.

4. If any issues are displayed on the tab, click the **>** icon in front of the entry for further details.
   Depending on the issue, the details can also contain a link to an Red Hat Knowledge Base article.
   For details and information on how to solve the problem, click **How to remediate this issue**.

CHAPTER 5. TROUBLESHOOTING

5.1. TROUBLESHOOTING INSTALLATIONS

5.1.1. Determining where installation issues occur

When troubleshooting OpenShift Container Platform installation issues, you can monitor installation logs to determine at which stage issues occur. Then, retrieve diagnostic data relevant to that stage.

OpenShift Container Platform installation proceeds through the following stages:

1. Ignition configuration files are created.
2. The bootstrap machine boots and starts hosting the remote resources required for the master machines to boot.
3. The master machines fetch the remote resources from the bootstrap machine and finish booting.
4. The master machines use the bootstrap machine to form an etcd cluster.
5. The bootstrap machine starts a temporary Kubernetes control plane using the new etcd cluster.
6. The temporary control plane schedules the production control plane to the master machines.
7. The temporary control plane shuts down and passes control to the production control plane.
8. The bootstrap machine adds OpenShift Container Platform components into the production control plane.
9. The installation program shuts down the bootstrap machine.
10. The control plane sets up the worker nodes.
11. The control plane installs additional services in the form of a set of Operators.
12. The cluster downloads and configures remaining components needed for the day-to-day operation, including the creation of worker machines in supported environments.

5.1.2. User-provisioned infrastructure installation considerations

The default installation method uses installer-provisioned infrastructure. With installer-provisioned infrastructure clusters, OpenShift Container Platform manages all aspects of the cluster, including the operating system itself. If possible, use this feature to avoid having to provision and maintain the cluster infrastructure.

You can alternatively install OpenShift Container Platform 4.5 on infrastructure that you provide. If you use this installation method, follow user-provisioned infrastructure installation documentation carefully. Additionally, review the following considerations before the installation:

- Check the Red Hat Enterprise Linux (RHEL) Ecosystem to determine the level of Red Hat Enterprise Linux CoreOS (RHCOS) support provided for your chosen server hardware or virtualization technology.

- Many virtualization and cloud environments require agents to be installed on guest operating systems. Ensure that these agents are installed as a containerized workload deployed through a DaemonSet.

- Install cloud provider integration if you want to enable features such as dynamic storage, on-demand service routing, node host name to Kubernetes host name resolution, and cluster autoscaling.

NOTE

It is not possible to enable cloud provider integration in OpenShift Container Platform environments that mix resources from different cloud providers, or that span multiple physical or virtual platforms. The node life cycle controller will not allow nodes that are external to the existing provider to be added to a cluster, and it is not possible to specify more than one cloud provider integration.

- A provider-specific Machine API implementation is required if you want to use MachineSets or autoscaling to automatically provision OpenShift Container Platform cluster nodes.

- Check whether your chosen cloud provider offers a method to inject Ignition configuration files into hosts as part of their initial deployment. If they do not, you will need to host Ignition configuration files by using an HTTP server. The steps taken to troubleshoot Ignition configuration file issues will differ depending on which of these two methods is deployed.

- Storage needs to be manually provisioned if you want to leverage optional framework components such as the embedded container registry, ElasticSearch, or Prometheus. Default storage classes are not defined in user-provisioned infrastructure installations unless explicitly configured.

- A load balancer is required to distribute API requests across all master nodes in highly available OpenShift Container Platform environments. You can use any TCP-based load balancing solution that meets OpenShift Container Platform DNS routing and port requirements.

5.1.3. Checking a load balancer configuration before OpenShift Container Platform installation

Check your load balancer configuration prior to starting an OpenShift Container Platform installation.

Prerequisites

- You have configured an external load balancer of your choosing, in preparation for an OpenShift Container Platform installation. The following example is based on a Red Hat Enterprise Linux (RHEL) host using HAProxy to provide load balancing services to a cluster.

- You have configured DNS in preparation for an OpenShift Container Platform installation.

- You have SSH access to your load balancer.

Procedure

1. Check that the **haproxy** systemd service is active:

```
$ ssh <user_name>@<load_balancer> systemctl status haproxy
```

2. Verify that the load balancer is listening on the required ports. The following example references ports **80**, **443**, **6443**, and **22623**.

   - For HAProxy instances running on Red Hat Enterprise Linux (RHEL) 6, verify port status by using the **netstat** command:

     ```
     $ ssh <user_name>@<load_balancer> netstat -nltupe | grep -E ':80|:443|:6443|:22623'
     ```

   - For HAProxy instances running on Red Hat Enterprise Linux (RHEL) 7 or 8, verify port status by using the **ss** command:

     ```
     $ ssh <user_name>@<load_balancer> ss -nltupe | grep -E ':80|:443|:6443|:22623'
     ```

     NOTE

     Red Hat recommends the **ss** command instead of **netstat** in Red Hat Enterprise Linux (RHEL) 7 or later. **ss** is provided by the iproute package. For more information on the **ss** command, see the Red Hat Enterprise Linux (RHEL) 7 Performance Tuning Guide.

3. Check that the wildcard DNS record resolves to the load balancer:

   ```
   $ dig <wildcard_fqdn> @<dns_server>
   ```

5.1.4. Specifying OpenShift Container Platform installer log levels

By default, the OpenShift Container Platform installer log level is set to **info**. If more detailed logging is required when diagnosing a failed OpenShift Container Platform installation, you can increase the **openshift-install** log level to **debug** when starting the installation again.

Prerequisites

- You have access to the installation host.

Procedure

- Set the installation log level to **debug** when initiating the installation:

  ```
  $ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete --log-level=debug 1
  ```

  1 Possible log levels include **info**, **warn**, **error,** and **debug**.

5.1.5. Troubleshooting openshift-install command issues

If you experience issues running the **openshift-install** command, check the following:

- The installation has been initiated within 24 hours of Ignition configuration file creation. The Ignition files are created when the following command is run:

  ```
  $ ./openshift-install create ignition-configs --dir=./install_dir
  ```

- The **install-config.yaml** file is in the same directory as the installer. If an alternative installation path is declared by using the **./openshift-install --dir** option, verify that the **install-config.yaml** file exists within that directory.

5.1.6. Monitoring installation progress

You can monitor high-level installation, bootstrap, and control plane logs as an OpenShift Container Platform installation progresses. This provides greater visibility into how an installation progresses and helps identify the stage at which an installation failure occurs.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- You have SSH access to your hosts.
- You have the fully qualified domain names of the bootstrap and master nodes.

NOTE

The initial **kubeadmin** password can be found in **<install_directory>/auth/kubeadmin-password** on the installation host.

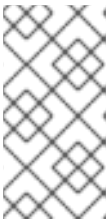
Procedure

1. Watch the installation log as the installation progresses:

```
$ tail -f ~/<installation_directory>/.openshift_install.log
```

2. Monitor the **bootkube.service** journald unit log on the bootstrap node, after it has booted. This provides visibility into the bootstrapping of the first control plane. Replace **<bootstrap_fqdn>** with the bootstrap node's fully qualified domain name:

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```



NOTE

The **bootkube.service** log on the bootstrap node outputs etcd **connection refused** errors, indicating that the bootstrap server is unable to connect to etcd on master nodes. After etcd has started on each master node and the nodes have joined the cluster, the errors should stop.

3. Monitor **kubelet.service** journald unit logs on master nodes, after they have booted. This provides visibility into master node agent activity.

   a. Monitor the logs using **oc**:

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

   b. If the API is not functional, review the logs using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u
kubelet.service
```

4. Monitor **crio.service** journald unit logs on master nodes, after they have booted. This provides visibility into master node CRI-O container runtime activity.

   a. Monitor the logs using **oc**:

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

   b. If the API is not functional, review the logs using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values:

   ```
   $ ssh core@master-N.cluster_name.sub_domain.domain journalctl -b -f -u crio.service
   ```

5.1.7. Gathering bootstrap node diagnostic data

When experiencing bootstrap-related issues, you can gather **bootkube.service journald** unit logs and container logs from the bootstrap node.

Prerequisites

- You have SSH access to your bootstrap node.
- You have the fully qualified domain name of the bootstrap node.
- If you are hosting Ignition configuration files by using an HTTP server, you must have the HTTP server's fully qualified domain name and the port number. You must also have SSH access to the HTTP host.

Procedure

1. If you have access to the bootstrap node's console, monitor the console until the node reaches the login prompt.
2. Verify the Ignition file configuration.
   - If you are hosting Ignition configuration files by using an HTTP server.

     a. Verify the bootstrap node Ignition file URL. Replace **<http_server_fqdn>** with HTTP server's fully qualified domain name:

     ```
     $ curl -I http://<http_server_fqdn>:<port>/bootstrap.ign 1
     ```

     1 The **-I** option returns the header only. If the Ignition file is available on the specified URL, the command returns **200 OK** status. If it is not available, the command returns **404 file not found**.

     b. To verify that the Ignition file was received by the bootstrap node, query the HTTP server logs on the serving host. For example, if you are using an Apache web server to serve Ignition files, enter the following command:

     ```
     $ grep -is 'bootstrap.ign' /var/log/httpd/access_log
     ```

If the bootstrap Ignition file is received, the associated **HTTP GET** log message will include a **200 OK** success status, indicating that the request succeeded.

c. If the Ignition file was not received, check that the Ignition files exist and that they have the appropriate file and web server permissions on the serving host directly.

- If you are using a cloud provider mechanism to inject Ignition configuration files into hosts as part of their initial deployment.

a. Review the bootstrap node's console to determine if the mechanism is injecting the bootstrap node Ignition file correctly.

3. Verify the availability of the bootstrap node's assigned storage device.

4. Verify that the bootstrap node has been assigned an IP address from the DHCP server.

5. Collect **bootkube.service** journald unit logs from the bootstrap node. Replace **<bootstrap_fqdn>** with the bootstrap node's fully qualified domain name:

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```

NOTE

The **bootkube.service** log on the bootstrap node outputs etcd **connection refused** errors, indicating that the bootstrap server is unable to connect to etcd on master nodes. After etcd has started on each master node and the nodes have joined the cluster, the errors should stop.

6. Collect logs from the bootstrap node containers.

a. Collect the logs using **podman** on the bootstrap node. Replace **<bootstrap_fqdn>** with the bootstrap node's fully qualified domain name:

```
$ ssh core@<bootstrap_fqdn> 'for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done'
```

7. If the bootstrap process fails, verify the following.

- You can resolve **api.<cluster_name>.<base_domain>** from the installation host.

- The load balancer proxies port 6443 connections to bootstrap and master nodes. Ensure that the proxy configuration meets OpenShift Container Platform installation requirements.

5.1.8. Investigating master node installation issues

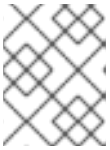
If you experience master node installation issues, determine the master node, OpenShift Container Platform software defined network (SDN), and network Operator status. Collect **kubelet.service**, **crio.service** journald unit logs, and master node container logs for visibility into master node agent, CRI-O container runtime, and Pod activity.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.

- You have installed the OpenShift CLI (**oc**).

- You have SSH access to your hosts.
- You have the fully qualified domain names of the bootstrap and master nodes.
- If you are hosting Ignition configuration files by using an HTTP server, you must have the HTTP server's fully qualified domain name and the port number. You must also have SSH access to the HTTP host.



NOTE

The initial **kubeadmin** password can be found in **<install_directory>/auth/kubeadmin-password** on the installation host.

Procedure

1. If you have access to the master node's console, monitor the console until the node reaches the login prompt. During the installation, Ignition log messages are output to the console.
2. Verify Ignition file configuration.
   - If you are hosting Ignition configuration files by using an HTTP server.
     a. Verify the master node Ignition file URL. Replace **<http_server_fqdn>** with HTTP server's fully qualified domain name:

     ```
     $ curl -I http://<http_server_fqdn>:<port>/master.ign 1
     ```

     1 The **-I** option returns the header only. If the Ignition file is available on the specified URL, the command returns **200 OK** status. If it is not available, the command returns **404 file not found**.

     b. To verify that the ignition file was received by the master node, query the HTTP server logs on the serving host. For example, if you are using an Apache web server to serve Ignition files:

     ```
     $ grep -is 'master.ign' /var/log/httpd/access_log
     ```

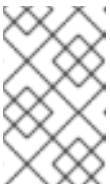
     If the master Ignition file is received, the associated **HTTP GET** log message will include a **200 OK** success status, indicating that the request succeeded.

     c. If the Ignition file was not received, check that it exists on the serving host directly. Ensure that the appropriate file and web server permissions are in place.
   - If you are using a cloud provider mechanism to inject Ignition configuration files into hosts as part of their initial deployment.
     a. Review the master node's console to determine if the mechanism is injecting the master node Ignition file correctly.
3. Check the availability of the master node's assigned storage device.
4. Verify that the master node has been assigned an IP address from the DHCP server.
5. Determine master node status.
   a. Query master node status:

```
$ oc get nodes
```

b. If one of the master nodes does not reach a **Ready** status, retrieve a detailed node description:

```
$ oc describe node <master_node>
```



NOTE

It is not possible to run **oc** commands if an installation issue prevents the OpenShift Container Platform API from running or if the kubelet is not running yet on each node:

6. Determine OpenShift Container Platform SDN status.

a. Review **sdn-controller**, **sdn**, and **ovs** DaemonSet status, in the **openshift-sdn** namespace:

```
$ oc get daemonsets -n openshift-sdn
```

b. If those resources are listed as **Not found**, review Pods in the **openshift-sdn** namespace:

```
$ oc get pods -n openshift-sdn
```

c. Review logs relating to failed OpenShift Container Platform SDN Pods in the **openshift-sdn** namespace:

```
$ oc logs <sdn_pod> -n openshift-sdn
```

7. Determine cluster network configuration status.

a. Review whether the cluster's network configuration exists:

```
$ oc get network.config.openshift.io cluster -o yaml
```

b. If the installer failed to create the network configuration, generate the Kubernetes manifests again and review message output:

```
$ ./openshift-install create manifests
```

c. Review Pod status in the **openshift-network-operator** namespace to determine whether the network Operator is running:

```
$ oc get pods -n openshift-network-operator
```

d. Gather network Operator Pod logs from the **openshift-network-operator** namespace:

```
$ oc logs pod/<network_operator_pod_name> -n openshift-network-operator
```

8. Monitor **kubelet.service** journald unit logs on master nodes, after they have booted. This provides visibility into master node agent activity.

a. Retrieve the logs using **oc**:

a. Retrieve the logs using **oc**:

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

b. If the API is not functional, review the logs using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. Before attempting to collect diagnostic data over SSH, review whether the data collected by running **oc adm must gather** and other **oc** commands is sufficient instead. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>**.

9. Retrieve **crio.service** journald unit logs on master nodes, after they have booted. This provides visibility into master node CRI-O container runtime activity.

   a. Retrieve the logs using **oc**:

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

   b. If the API is not functional, review the logs using SSH instead:

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u crio.service
   ```

10. Collect logs from specific subdirectories under **/var/log/** on master nodes.

    a. Retrieve a list of logs contained within a **/var/log/** subdirectory. The following example lists files in **/var/log/openshift-apiserver/** on all master nodes:

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

    b. Inspect a specific log within a **/var/log/** subdirectory. The following example outputs **/var/log/openshift-apiserver/audit.log** contents from all master nodes:

    ```
    $ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
    ```

    c. If the API is not functional, review the logs on each node using SSH instead. The following example tails **/var/log/openshift-apiserver/audit.log**:

    ```
    $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f /var/log/openshift-apiserver/audit.log
    ```

11. Review master node container logs using SSH.

    a. List the containers:

    ```
    $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps -a
    ```

    b. Retrieve a container's logs using **crictl**:

    ```
    $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
    <container_id>
    ```

12. If you experience master node configuration issues, verify that the MCO, MCO endpoint, and DNS record are functioning. The Machine Config Operator (MCO) manages operating system configuration during the installation procedure. Also verify system clock accuracy and certificate validity.

    a. Test whether the MCO endpoint is available. Replace **<cluster_name>** with appropriate values:

    ```
    $ curl https://api-int.<cluster_name>:22623/config/master
    ```

    b. If the endpoint is unresponsive, verify load balancer configuration. Ensure that the endpoint is configured to run on port 22623.

    c. Verify that the MCO endpoint's DNS record is configured and resolves to the load balancer.

        i. Run a DNS lookup for the defined MCO endpoint name:

        ```
        $ dig api-int.<cluster_name> @<dns_server>
        ```

        ii. Run a reverse lookup to the assigned MCO IP address on the load balancer:

        ```
        $ dig -x <load_balancer_mco_ip_address> @<dns_server>
        ```

    d. Verify that the MCO is functioning from the bootstrap node directly. Replace **<bootstrap_fqdn>** with the bootstrap node's fully qualified domain name:

    ```
    $ ssh core@<bootstrap_fqdn> curl https://api-int.<cluster_name>:22623/config/master
    ```

    e. System clock time must be synchronized between bootstrap, master, and worker nodes. Check each node's system clock reference time and time synchronization statistics:

    ```
    $ ssh core@<node>.<cluster_name>.<base_domain> chronyc tracking
    ```

    f. Review certificate validity:

    ```
    $ openssl s_client -connect api-int.<cluster_name>:22623 | openssl x509 -noout -text
    ```

5.1.9. Investigating etcd installation issues

If you experience etcd issues during installation, you can check etcd Pod status and collect etcd Pod logs. You can also verify etcd DNS records and check DNS availability on master nodes.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- You have SSH access to your hosts.
- You have the fully qualified domain names of the master nodes.

Procedure

1. Check the status of etcd Pods.
   a. Review the status of Pods in the **openshift-etcd** namespace:

   ```
   $ oc get pods -n openshift-etcd
   ```

   b. Review the status of Pods in the **openshift-etcd-operator** namespace:

   ```
   $ oc get pods -n openshift-etcd-operator
   ```

2. If any of the Pods listed by the previous commands are not showing a **Running** or a **Completed** status, gather diagnostic information for the Pod.
   a. Review events for the Pod:

   ```
   $ oc describe pod/<pod_name> -n <namespace>
   ```

   b. Inspect the Pod's logs:

   ```
   $ oc logs pod/<pod_name> -n <namespace>
   ```

   c. If the Pod has more than one container, the preceding command will create an error, and the container names will be provided in the error message. Inspect logs for each container:

   ```
   $ oc logs pod/<pod_name> -c <container_name> -n <namespace>
   ```

3. If the API is not functional, review etcd Pod and container logs on each master node by using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values.
   a. List etcd Pods on each master node:

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl pods --
   name=etcd-
   ```

   b. For any Pods not showing **Ready** status, inspect Pod status in detail. Replace **<pod_id>** with the Pod's ID listed in the output of the preceding command:

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspectp
   <pod_id>
   ```

   c. List containers related to a Pod:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps | grep
'<pod_id>'
```

d. For any containers not showing **Ready** status, inspect container status in detail. Replace **<container_id>** with container IDs listed in the output of the preceding command:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspect
<container_id>
```

e. Review the logs for any containers not showing a **Ready** status. Replace **<container_id>** with the container IDs listed in the output of the preceding command:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. Before attempting to collect diagnostic data over SSH, review whether the data collected by running **oc adm must gather** and other **oc** commands is sufficient instead. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>**.

4. Validate primary and secondary DNS server connectivity from master nodes.

5.1.10. Investigating master node kubelet and API server issues

To investigate master node kubelet and API server issues during installation, check DNS, DHCP, and load balancer functionality. Also, verify that certificates have not expired.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- You have SSH access to your hosts.
- You have the fully qualified domain names of the master nodes.

Procedure

1. Verify that the API server's DNS record directs the kubelet on master nodes to **https://api-int.<cluster_name>.<base_domain>:6443**. Ensure that the record references the load balancer.
2. Ensure that the load balancer's port 6443 definition references each master node.
3. Check that unique master node host names have been provided by DHCP.

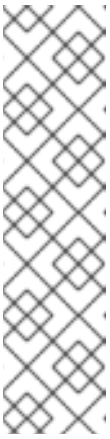
4. Inspect the **kubelet.service** journald unit logs on each master node.

   a. Retrieve the logs using **oc**:

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

   b. If the API is not functional, review the logs using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values:

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
   ```



NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. Before attempting to collect diagnostic data over SSH, review whether the data collected by running **oc adm must gather** and other **oc** commands is sufficient instead. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>**.

5. Check for certificate expiration messages in the master node kubelet logs.

   a. Retrieve the log using **oc**:

   ```
   $ oc adm node-logs --role=master -u kubelet | grep -is 'x509: certificate has expired'
   ```

   b. If the API is not functional, review the logs using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values:

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service  | grep -is 'x509: certificate has expired'
   ```

5.1.11. Investigating worker node installation issues

If you experience worker node installation issues, you can review the worker node status. Collect **kubelet.service**, **crio.service** journald unit logs and the worker node container logs for visibility into the worker node agent, CRI-O container runtime and Pod activity. Additionally, you can check the Ignition file and Machine API Operator functionality. If worker node post-installation configuration fails, check Machine Config Operator (MCO) and DNS functionality. You can also verify system clock synchronization between the bootstrap, master, and worker nodes, and validate certificates.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- You have SSH access to your hosts.

- You have the fully qualified domain names of the bootstrap and worker nodes.
- If you are hosting Ignition configuration files by using an HTTP server, you must have the HTTP server's fully qualified domain name and the port number. You must also have SSH access to the HTTP host.

NOTE

The initial **kubeadmin** password can be found in **<install_directory>/auth/kubeadmin-password** on the installation host.

Procedure

1. If you have access to the worker node's console, monitor the console until the node reaches the login prompt. During the installation, Ignition log messages are output to the console.
2. Verify Ignition file configuration.
   - If you are hosting Ignition configuration files by using an HTTP server.
     a. Verify the worker node Ignition file URL. Replace **<http_server_fqdn>** with HTTP server's fully qualified domain name:

```
$ curl -I http://<http_server_fqdn>:<port>/worker.ign 1
```

     1 The **-I** option returns the header only. If the Ignition file is available on the specified URL, the command returns **200 OK** status. If it is not available, the command returns **404 file not found**.

     b. To verify that the ignition file was received by the worker node, query the HTTP server logs on the HTTP host. For example, if you are using an Apache web server to serve Ignition files:

```
$ grep -is 'worker.ign' /var/log/httpd/access_log
```

     If the worker Ignition file is received, the associated **HTTP GET** log message will include a **200 OK** success status, indicating that the request succeeded.

     c. If the Ignition file was not received, check that it exists on the serving host directly. Ensure that the appropriate file and web server permissions are in place.

   - If you are using a cloud provider mechanism to inject Ignition configuration files into hosts as part of their initial deployment.
     a. Review the worker node's console to determine if the mechanism is injecting the worker node Ignition file correctly.
3. Check the availability of the worker node's assigned storage device.
4. Verify that the worker node has been assigned an IP address from the DHCP server.
5. Determine worker node status.
   a. Query node status:

```
$ oc get nodes
```

b. Retrieve a detailed node description for any worker nodes not showing a **Ready** status:

```
$ oc describe node <worker_node>
```

NOTE

It is not possible to run **oc** commands if an installation issue prevents the OpenShift Container Platform API from running or if the kubelet is not running yet on each node.

6. Unlike master nodes, worker nodes are deployed and scaled using the Machine API Operator. Check the status of the Machine API Operator.

a. Review Machine API Operator Pod status:

```
$ oc get pods -n openshift-machine-api
```

b. If the Machine API Operator Pod does not have a **Ready** status, detail the Pod's events:

```
$ oc describe pod/<machine_api_operator_pod_name> -n openshift-machine-api
```

c. Inspect **machine-api-operator** container logs. The container runs within the **machine-api-operator** Pod:

```
$ oc logs pod/<machine_api_operator_pod_name> -n openshift-machine-api -c machine-api-operator
```

d. Also inspect **kube-rbac-proxy** container logs. The container also runs within the **machine-api-operator** Pod:

```
$ oc logs pod/<machine_api_operator_pod_name> -n openshift-machine-api -c kube-rbac-proxy
```

7. Monitor **kubelet.service** journald unit logs on worker nodes, after they have booted. This provides visibility into worker node agent activity.

a. Retrieve the logs using **oc**:

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

b. If the API is not functional, review the logs using SSH instead. Replace **<worker-node>.<cluster_name>.<base_domain>** with appropriate values:

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. Before attempting to collect diagnostic data over SSH, review whether the data collected by running **oc adm must gather** and other **oc** commands is sufficient instead. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>**.

8. Retrieve **crio.service** journald unit logs on worker nodes, after they have booted. This provides visibility into worker node CRI-O container runtime activity.

   a. Retrieve the logs using **oc**:

   ```
   $ oc adm node-logs --role=worker -u crio
   ```

   b. If the API is not functional, review the logs using SSH instead:

   ```
   $ ssh core@<worker-node>.<cluster_name>.<base_domain> journalctl -b -f -u
   crio.service
   ```

9. Collect logs from specific subdirectories under /**var**/**log**/ on worker nodes.

   a. Retrieve a list of logs contained within a /**var**/**log**/ subdirectory. The following example lists files in /**var**/**log**/**sssd**/ on all worker nodes:

   ```
   $ oc adm node-logs --role=worker --path=sssd
   ```

   b. Inspect a specific log within a /**var**/**log**/ subdirectory. The following example outputs /**var**/**log**/**sssd**/**audit.log** contents from all worker nodes:

   ```
   $ oc adm node-logs --role=worker --path=sssd/sssd.log
   ```

   c. If the API is not functional, review the logs on each node using SSH instead. The following example tails /**var**/**log**/**sssd**/**sssd.log**:

   ```
   $ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo tail -f
   /var/log/sssd/sssd.log
   ```

10. Review worker node container logs using SSH.

    a. List the containers:

    ```
    $ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo crictl ps -a
    ```

    b. Retrieve a container's logs using **crictl**:

    ```
    $ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo crictl logs -f
    <container_id>
    ```

11. If you experience worker node configuration issues, verify that the MCO, MCO endpoint, and DNS record are functioning. The Machine Config Operator (MCO) manages operating system configuration during the installation procedure. Also verify system clock accuracy and certificate validity.

    a. Test whether the MCO endpoint is available. Replace **<cluster_name>** with appropriate values:

    ```
    $ curl https://api-int.<cluster_name>:22623/config/worker
    ```

    b. If the endpoint is unresponsive, verify load balancer configuration. Ensure that the endpoint is configured to run on port 22623.

    c. Verify that the MCO endpoint's DNS record is configured and resolves to the load balancer.

        i. Run a DNS lookup for the defined MCO endpoint name:

        ```
        $ dig api-int.<cluster_name> @<dns_server>
        ```

        ii. Run a reverse lookup to the assigned MCO IP address on the load balancer:

        ```
        $ dig -x <load_balancer_mco_ip_address> @<dns_server>
        ```

    d. Verify that the MCO is functioning from the bootstrap node directly. Replace **<bootstrap_fqdn>** with the bootstrap node's fully qualified domain name:

    ```
    $ ssh core@<bootstrap_fqdn> curl https://api-int.<cluster_name>:22623/config/worker
    ```

    e. System clock time must be synchronized between bootstrap, master, and worker nodes. Check each node's system clock reference time and time synchronization statistics:

    ```
    $ ssh core@<node>.<cluster_name>.<base_domain> chronyc tracking
    ```

    f. Review certificate validity:

    ```
    $ openssl s_client -connect api-int.<cluster_name>:22623 | openssl x509 -noout -text
    ```

5.1.12. Querying Operator status after installation

You can check Operator status at the end of an installation. Retrieve diagnostic data for Operators that do not become available. Review logs for any Operator Pods that are listed as **Pending** or have an error status. Validate base images used by problematic Pods.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Check that cluster Operators are all available at the end of an installation.

```
$ oc get clusteroperators
```

2. If any Operators fail to become available, view Operator events:

```
$ oc describe clusteroperator <operator_name>
```

3. Review Operator Pod status within the Operator's namespace:

```
$ oc get pods -n <operator_namespace>
```

4. Obtain a detailed description for Pods that do not have **Running** status:

```
$ oc describe pod/<operator_pod_name> -n <operator_namespace>
```

5. Inspect Pod logs:

```
$ oc logs pod/<operator_pod_name> -n <operator_namespace>
```

6. When experiencing Pod base image related issues, review base image status.
   a. Obtain details of the base image used by a problematic Pod:

```
$ oc get pod -o "jsonpath={range .status.containerStatuses[*]}{.name}{'\t'}{.state}{'\t'}
{.image}{'\n'}{end}" <operator_pod_name> -n <operator_namespace>
```

   b. List base image release information:

```
$ oc adm release info <image_path>:<tag> --commits
```

5.1.13. 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 is running and accessible through SSH.
- The **ssh-agent** process is active on your computer, and you provided the same SSH key to both the **ssh-agent** process and the installation program.
- If you tried to install a cluster on infrastructure that you provisioned, you must have the fully qualified domain names of the bootstrap and master nodes.

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=<installation_directory> 1
     ```

     1 **installation_directory** is the directory you specified when you ran **./openshift-install create cluster**. This directory contains 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=<installation_directory> \ 1
       --bootstrap <bootstrap_address> \ 2
       --master <master_1_address> \ 3
       --master <master_2_address> \ 4
       --master <master_3_address>" 5
     ```

     1 For **installation_directory**, specify the same directory you specified when you ran **./openshift-install create cluster**. This directory contains 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.

   Example output

   ```
   INFO Pulling debug logs from the bootstrap machine
   INFO Bootstrap gather logs captured here "<installation_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.

5.1.14. Additional resources

- See Installation process for more details on OpenShift Container Platform installation types and process.

5.2. VERIFYING NODE HEALTH

5.2.1. Reviewing node status, resource usage, and configuration

Review cluster node health status, resource consumption statistics, and node logs. Additionally, query **kubelet** status on individual nodes.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. List the name, status, and role for all nodes in the cluster:

```
$ oc get nodes
```

2. Summarize CPU and memory usage for each node within the cluster:

```
$ oc adm top nodes
```

3. Summarize CPU and memory usage for a specific node:

```
$ oc adm top node -l my-node
```

5.2.2. Querying the kubelet's status on a node

You can review cluster node health status, resource consumption statistics, and node logs. Additionally, you can query **kubelet** status on individual nodes.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. The kubelet is managed using a systemd service on each node. Review the kubelet's status by querying the **kubelet** systemd service within a debug Pod.
   a. Start a debug Pod for a node:

```
$ oc debug node/my-node
```

   b. Set **/host** as the root directory within the debug shell. The debug Pod mounts the host's root file system in **/host** within the Pod. By changing the root directory to **/host**, you can run binaries contained in the host's executable paths:

```
# chroot /host
```

NOTE

OpenShift Container Platform cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or **kubelet** is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>** instead.

c. Check whether the **kubelet** systemd service is active on the node:

```
# systemctl is-active kubelet
```

d. Output a more detailed **kubelet.service** status summary:

```
# systemctl status kubelet
```

5.2.3. Querying cluster node journal logs

You can gather **journald** unit logs and other logs within /**var**/**log** on individual cluster nodes.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).
- You have SSH access to your hosts.

Procedure

1. Query **kubelet journald** unit logs from OpenShift Container Platform cluster nodes. The following example queries master nodes only:

```
$ oc adm node-logs --role=master -u kubelet 1
```

1 Replace **kubelet** as appropriate to query other unit logs.

2. Collect logs from specific subdirectories under /**var**/**log**/ on cluster nodes.

a. Retrieve a list of logs contained within a /**var**/**log**/ subdirectory. The following example lists files in /**var**/**log**/**openshift-apiserver**/ on all master nodes:

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

b. Inspect a specific log within a **/var/log/** subdirectory. The following example outputs **/var/log/openshift-apiserver/audit.log** contents from all master nodes:

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

c. If the API is not functional, review the logs on each node using SSH instead. The following example tails **/var/log/openshift-apiserver/audit.log**:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
/var/log/openshift-apiserver/audit.log
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. Before attempting to collect diagnostic data over SSH, review whether the data collected by running **oc adm must gather** and other **oc** commands is sufficient instead. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>**.

5.3. TROUBLESHOOTING CRI-O CONTAINER RUNTIME ISSUES

5.3.1. About CRI-O container runtime engine

CRI-O is a Kubernetes-native container runtime implementation that integrates closely with the operating system to deliver an efficient and optimized Kubernetes experience. CRI-O provides facilities for running, stopping, and restarting containers.

The CRI-O container runtime engine is managed using a systemd service on each OpenShift Container Platform cluster node. When container runtime issues occur, verify the status of the **crio** systemd service on each node. Gather CRI-O journald unit logs from nodes that manifest container runtime issues.

5.3.2. Verifying CRI-O runtime engine status

You can verify CRI-O container runtime engine status on each cluster node.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

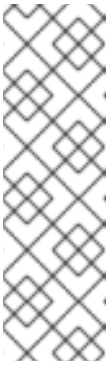
Procedure

1. Review CRI-O status by querying the **crio** systemd service on a node, within a debug Pod.
   a. Start a debug Pod for a node:

```
$ oc debug node/my-node
```

b. Set **/host** as the root directory within the debug shell. The debug Pod mounts the host's root file system in **/host** within the Pod. By changing the root directory to **/host**, you can run binaries contained in the host's executable paths:

```
# chroot /host
```



NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>** instead.

c. Check whether the **crio** systemd service is active on the node:

```
# systemctl is-active crio
```

d. Output a more detailed **kubelet.service** status summary:

```
# systemctl status crio
```

5.3.3. Gathering CRI-O journald unit logs

If you experience CRI-O issues, you can obtain CRI-O journald unit logs from a node.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).
- You have the fully qualified domain names of the control plane, or master machines.

Procedure

1. Gather CRI-O journald unit logs. The following example collects logs from all master nodes within the cluster:

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

2. Gather CRI-O journald unit logs from a specific node:

```
$ oc adm node-logs <node_name> -u crio
```

3. If the API is not functional, review the logs using SSH instead. Replace **<node>.<cluster_name>.<base_domain>** with appropriate values:

```
$ ssh core@<node>.<cluster_name>.<base_domain> journalctl -b -f -u crio.service
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. Before attempting to collect diagnostic data over SSH, review whether the data collected by running **oc adm must gather** and other **oc** commands is sufficient instead. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>**.

5.4. TROUBLESHOOTING OPERATOR ISSUES

Operators are a method of packaging, deploying, and managing an OpenShift Container Platform application. They act like an extension of the software vendor's engineering team, watching over an OpenShift Container Platform environment and using its current state to make decisions in real time. Operators are designed to handle upgrades seamlessly, react to failures automatically, and not take shortcuts, like skipping a software backup process to save time.

OpenShift Container Platform 4.5 includes a default set of Operators that are required for proper functioning of the cluster. These default Operators are managed by the Cluster Version Operator (CVO).

As a cluster administrator, you can install application Operators from the OperatorHub using the OpenShift Container Platform web console or the CLI. You can then subscribe the Operator to one or more namespaces to make it available for developers on your cluster. Application Operators are managed by Operator Lifecycle Manager (OLM).

If you experience Operator issues, verify Operator Subscription status. Check Operator Pod health across the cluster and gather Operator logs for diagnosis.

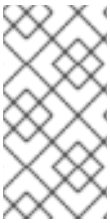
5.4.1. Operator Subscription condition types

Subscriptions can report the following condition types:

Table 5.1. Subscription condition types

Condition	Description
CatalogSourcesUnhealthy	Some or all of the Catalog Sources to be used in resolution are unhealthy.
InstallPlanMissing	A Subscription's InstallPlan is missing.
InstallPlanPending	A Subscription's InstallPlan is pending installation.

Condition	Description
InstallPlanFailed	A Subscription's InstallPlan has failed.



NOTE

Default OpenShift Container Platform cluster Operators are managed by the Cluster Version Operator (CVO) and they do not have a Subscription object. Application Operators are managed by Operator Lifecycle Manager (OLM) and they have a Subscription object.

5.4.2. Viewing Operator Subscription status using the CLI

You can view Operator Subscription status using the CLI.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. List Operator Subscriptions:

```
$ oc get subs -n <operator_namespace>
```

2. Use the **oc describe** command to inspect a Subscription resource:

```
$ oc describe sub <subscription_name> -n <operator_namespace>
```

3. In the command output, find the **Conditions** section for the status of Operator subscription condition types. In the following example, the **CatalogSourcesUnhealthy** condition type has a status of **false** because all available catalog sources are healthy:

   Example output

```
Conditions:
  Last Transition Time:  2019-07-29T13:42:57Z
  Message:               all available catalogsources are healthy
  Reason:                AllCatalogSourcesHealthy
  Status:                False
  Type:                  CatalogSourcesUnhealthy
```

NOTE

Default OpenShift Container Platform cluster Operators are managed by the Cluster Version Operator (CVO) and they do not have a Subscription object. Application Operators are managed by Operator Lifecycle Manager (OLM) and they have a Subscription object.

5.4.3. Querying Operator Pod status

You can list Operator Pods within a cluster and their status. You can also collect a detailed Operator Pod summary.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. List Operators running in the cluster. The output includes Operator version, availability, and up-time information:

```
$ oc get clusteroperators
```

2. List Operator Pods running in the Operator's namespace, plus Pod status, restarts, and age:

```
$ oc get pod -n <operator_namespace>
```

3. Output a detailed Operator Pod summary:

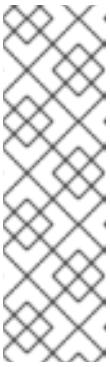
```
$ oc describe pod <operator_pod_name> -n <operator_namespace>
```

4. If an Operator issue is node-specific, query Operator container status on that node.
   a. Start a debug Pod for the node:

```
$ oc debug node/my-node
```

   b. Set **/host** as the root directory within the debug shell. The debug Pod mounts the host's root file system in **/host** within the Pod. By changing the root directory to **/host**, you can run binaries contained in the host's executable paths:

```
# chroot /host
```



NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>** instead.

   c. List details about the node's containers, including state and associated Pod IDs:

```
# crictl ps
```

d. List information about a specific Operator container on the node. The following example lists information about the **network-operator** container:

```
# crictl ps --name network-operator
```

e. Exit from the debug shell.

5.4.4. Gathering Operator logs

If you experience Operator issues, you can gather detailed diagnostic information from Operator Pod logs.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).
- You have the fully qualified domain names of the control plane, or master machines.

Procedure

1. List the Operator Pods that are running in the Operator's namespace, plus the Pod status, restarts, and age:

```
$ oc get pods -n <operator_namespace>
```

2. Review logs for an Operator Pod:

```
$ oc logs pod/<pod_name> -n <operator_namespace>
```

If an Operator Pod has multiple containers, the preceding command will produce an error that includes the name of each container. Query logs from an individual container:

```
$ oc logs pod/<operator_pod_name> -c <container_name> -n <operator_namespace>
```

3. If the API is not functional, review Operator Pod and container logs on each master node by using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values.

a. List Pods on each master node:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl pods
```

b. For any Operator Pods not showing a **Ready** status, inspect the Pod's status in detail. Replace **<operator_pod_id>** with the Operator Pod's ID listed in the output of the preceding command:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspectp
<operator_pod_id>
```

c. List containers related to an Operator Pod:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps --pod=
<operator_pod_id>
```

d. For any Operator container not showing a **Ready** status, inspect the container's status in detail. Replace **<container_id>** with a container ID listed in the output of the preceding command:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspect
<container_id>
```

e. Review the logs for any Operator containers not showing a **Ready** status. Replace **<container_id>** with a container ID listed in the output of the preceding command:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```

NOTE

OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. Before attempting to collect diagnostic data over SSH, review whether the data collected by running **oc adm must gather** and other **oc** commands is sufficient instead. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>**.

5.5. INVESTIGATING POD ISSUES

OpenShift Container Platform leverages the Kubernetes concept of a Pod, which is one or more containers deployed together on one host. A Pod is the smallest compute unit that can be defined, deployed, and managed on OpenShift Container Platform 4.5.

After a Pod is defined, it is assigned to run on a node until its containers exit, or until it is removed. Depending on policy and exit code, Pods are either removed after exiting or retained so that their logs can be accessed.

The first thing to check when Pod issues arise is the Pod's status. If an explicit Pod failure has occurred, observe the Pod's error state to identify specific image, container, or Pod network issues. Focus diagnostic data collection according to the error state. Review Pod event messages, as well as Pod and container log information. Diagnose issues dynamically by accessing running Pods on the command line, or start a debug Pod with root access based on a problematic Pod's deployment configuration.

5.5.1. Understanding Pod error states

Pod failures return explicit error states that can be observed in the **status** field in the output of **oc get Pods**. Pod error states cover image, container, and container network related failures.

The following table provides a list of Pod error states along with their descriptions.

Table 5.2. Pod error states

Pod error state	Description
ErrImagePull	Generic image retrieval error.
ErrImagePullBackOff	Image retrieval failed and is backed off.
ErrInvalidImageName	The specified image name was invalid.
ErrImageInspect	Image inspection did not succeed.
ErrImageNeverPull	**PullPolicy** is set to **NeverPullImage** and the target image is not present locally on the host.
ErrRegistryUnavailable	When attempting to retrieve an image from a registry, an HTTP error was encountered.
ErrContainerNotFound	The specified container is either not present or not managed by the kubelet, within the declared Pod.
ErrRunInitContainer	Container initialization failed.
ErrRunContainer	None of the Pod's containers started successfully.
ErrKillContainer	None of the Pod's containers were killed successfully.
ErrCrashLoopBackOff	A container has terminated. The kubelet will not attempt to restart it.
ErrVerifyNonRoot	A container or image attempted to run with root privileges.
ErrCreatePodSandbox	Pod sandbox creation did not succeed.
ErrConfigPodSandbox	Pod sandbox configuration was not obtained.

Pod error state	Description
ErrKillPodSandbox	A Pod's sandbox did not stop successfully.
ErrSetupNetwork	Network initialization failed.
ErrTeardownNetwork	Network termination failed.

5.5.2. Reviewing Pod status

You can query Pod status and error states. You can also query a Pod's associated deployment configuration and review base image availability.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- **skopeo** is installed.

Procedure

1. Switch into a project:

```
$ oc project <project_name>
```

2. List Pods running within the namespace, as well as Pod status, error states, restarts, and age:

```
$ oc get pods
```

3. Determine whether the namespace is managed by a deployment configuration:

```
$ oc status
```

   If the namespace is managed by a deployment configuration, the output includes the deployment configuration name and a base image reference.

4. Inspect the base image referenced in the preceding command's output:

```
$ skopeo inspect docker://<image_reference>
```

5. If the base image reference is not correct, update the reference in the deployment configuration:

```
$ oc edit deployment/my-deployment
```

6. When deployment configuration changes on exit, the configuration will automatically redeploy. Watch Pod status as the deployment progresses, to determine whether the issue has been resolved:

```
$ oc get pods -w
```

7. Review events within the namespace for diagnostic information relating to Pod failures:

```
$ oc get events
```

5.5.3. Inspecting Pod and container logs

You can inspect Pod and container logs for warnings and error messages related to explicit Pod failures. Depending on policy and exit code, Pod and container logs remain available after Pods have been terminated.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Query logs for a specific Pod:

```
$ oc logs <pod_name>
```

2. Query logs for a specific container within a Pod:

```
$ oc logs <pod_name> -c <container_name>
```

   Logs retrieved using the preceding **oc logs** commands are composed of messages sent to stdout within Pods or containers.

3. Inspect logs contained in **/var/log/** within a Pod.
   a. List log files and subdirectories contained in **/var/log** within a Pod:

```
$ oc exec <pod_name> ls -alh /var/log
```

   b. Query a specific log file contained in **/var/log** within a Pod:

```
$ oc exec <pod_name> cat /var/log/<path_to_log>
```

   c. List log files and subdirectories contained in **/var/log** within a specific container:

```
$ oc exec <pod_name> -c <container_name> ls /var/log
```

   d. Query a specific log file contained in **/var/log** within a specific container:

```
$ oc exec <pod_name> -c <container_name> cat /var/log/<path_to_log>
```

5.5.4. Accessing running Pods

You can review running Pods dynamically by opening a shell inside a Pod or by gaining network access through port forwarding.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Switch into the project that contains the Pod you would like to access. This is necessary because the **oc rsh** command does not accept the **-n** namespace option:

```
$ oc project <namespace>
```

2. Start a remote shell into a Pod:

```
$ oc rsh <pod_name> 1
```

1 If a Pod has multiple containers, **oc rsh** defaults to the first container unless **-c <container_name>** is specified.

3. Start a remote shell into a specific container within a Pod:

```
$ oc rsh -c <container_name> pod/<pod_name>
```

4. Create a port forwarding session to a port on a Pod:

```
$ oc port-forward <pod_name> <host_port>:<pod_port> 1
```

1 Enter **Ctrl+C** to cancel the port forwarding session.

5.5.5. Starting debug Pods with root access

You can start a debug Pod with root access, based on a problematic Pod's deployment or deployment configuration. Pod users typically run with non-root privileges, but running troubleshooting Pods with temporary root privileges can be useful during issue investigation.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.

- You have installed the OpenShift CLI (**oc**).

Procedure

1. Start a debug Pod with root access, based on a deployment.
   a. Obtain a project's deployment name:

```
$ oc get deployment -n <project_name>
```

   b. Start a debug Pod with root privileges, based on the deployment:

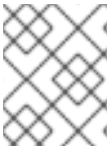
```
$ oc debug deployment/my-deployment --as-root -n <project_name>
```

2. Start a debug Pod with root access, based on a deployment configuration.
   a. Obtain a project's deployment configuration name:

```
$ oc get deploymentconfigs -n <project_name>
```

   b. Start a debug Pod with root privileges, based on the deployment configuration:

```
$ oc debug deploymentconfig/my-deployment-configuration --as-root -n <project_name>
```



NOTE

You can append **-- <command>** to the preceding **oc debug** commands to run individual commands within a debug Pod, instead of running an interactive shell.

5.5.6. Copying files to and from Pods and containers

You can copy files to and from a Pod to test configuration changes or gather diagnostic information.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Copy a file to a Pod:

```
$ oc cp <local_path> <pod_name>:/<path> -c <container_name>  1
```

1 Note that a Pod's first container will be selected if the **-c** option is not specified.

2. Copy a file from a Pod:

```
$ oc cp <pod_name>:/<path>  -c <container_name><local_path>  1
```

1 Note that a Pod's first container will be selected if the **-c** option is not specified.

NOTE

For **oc cp** to function, the **tar** binary must be available within the container.

5.6. TROUBLESHOOTING THE SOURCE-TO-IMAGE PROCESS

5.6.1. Strategies for Source-to-Image troubleshooting

Use Source-to-Image (S2I) to build reproducible, Docker-formatted container images. You can create ready-to-run images by injecting application source code into a container image and assembling a new image. The new image incorporates the base image (the builder) and built source.

To determine where in the S2I process a failure occurs, you can observe the state of the Pods relating to each of the following S2I stages:

1. **During the build configuration stage**, a build Pod is used to create an application container image from a base image and application source code.

2. **During the deployment configuration stage**, a deployment Pod is used to deploy application Pods from the application container image that was built in the build configuration stage. The deployment Pod also deploys other resources such as services and routes. The deployment configuration begins after the build configuration succeeds.

3. **After the deployment Pod has started the application Pods**, application failures can occur within the running application Pods. For instance, an application might not behave as expected even though the application Pods are in a **Running** state. In this scenario, you can access running application Pods to investigate application failures within a Pod.

When troubleshooting S2I issues, follow this strategy:

1. Monitor build, deployment, and application Pod status

2. Determine the stage of the S2I process where the problem occurred

3. Review logs corresponding to the failed stage

5.6.2. Gathering Source-to-Image diagnostic data

The S2I tool runs a build Pod and a deployment Pod in sequence. The deployment Pod is responsible for deploying the application Pods based on the application container image created in the build stage. Watch build, deployment and application Pod status to determine where in the S2I process a failure occurs. Then, focus diagnostic data collection accordingly.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- Your API service is still functional.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Watch the Pod status throughout the S2I process to determine at which stage a failure occurs:

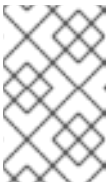
```
$ oc get pods -w 1
```

1 Use **-w** to monitor Pods for changes until you quit the command using **Ctrl+C**.

2. Review a failed Pod's logs for errors.

- **If the build Pod fails**, review the build Pod's logs:

```
$ oc logs -f pod/<application_name>-<build_number>-build
```



NOTE

Alternatively, you can review the build configuration's logs using **oc logs -f bc/<application_name>**. The build configuration's logs include the logs from the build Pod.

- **If the deployment Pod fails**, review the deployment Pod's logs:

```
$ oc logs -f pod/<application_name>-<build_number>-deploy
```

NOTE

Alternatively, you can review the deployment configuration's logs using **oc logs -f dc/<application_name>**. This outputs logs from the deployment Pod until the deployment Pod completes successfully. The command outputs logs from the application Pods if you run it after the deployment Pod has completed. After a deployment Pod completes, its logs can still be accessed by running **oc logs -f pod/<application_name>-<build_number>-deploy**.

- **If an application Pod fails, or if an application is not behaving as expected within a running application Pod**, review the application Pod's logs:

```
$ oc logs -f pod/<application_name>-<build_number>-<random_string>
```

5.6.3. Gathering application diagnostic data to investigate application failures

Application failures can occur within running application Pods. In these situations, you can retrieve diagnostic information with these strategies:

- Review events relating to the application Pods.
- Review the logs from the application Pods, including application-specific log files that are not collected by the OpenShift Container Platform logging framework.
- Test application functionality interactively and run diagnostic tools in an application container.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. List events relating to a specific application Pod. The following example retrieves events for an application Pod named **my-app-1-akdlg**:

```
$ oc describe pod/my-app-1-akdlg
```

2. Review logs from an application Pod:

```
$ oc logs -f pod/my-app-1-akdlg
```

3. Query specific logs within a running application Pod. Logs that are sent to stdout are collected by the OpenShift Container Platform logging framework and are included in the output of the preceding command. The following query is only required for logs that are not sent to stdout.

   a. If an application log can be accessed without root privileges within a Pod, concatenate the log file as follows:

```
$ oc exec my-app-1-akdlg -- cat /var/log/my-application.log
```

   b. If root access is required to view an application log, you can start a debug container with root privileges and then view the log file from within the container. Start the debug container from the project's deployment configuration. Pod users typically run with non-root privileges, but running troubleshooting Pods with temporary root privileges can be useful during issue investigation:

```
$ oc debug dc/my-deployment-configuration --as-root -- cat /var/log/my-application.log
```

NOTE

You can access an interactive shell with root access within the debug Pod if you run **oc debug dc/<deployment_configuration> --as-root** without appending **-- <command>**.

4. Test application functionality interactively and run diagnostic tools, in an application container with an interactive shell.

   a. Start an interactive shell on the application container:

```
$ oc exec -it my-app-1-akdlg /bin/bash
```

   b. Test application functionality interactively from within the shell. For example, you can run the container's entry point command and observe the results. Then, test changes from the command line directly, before updating the source code and rebuilding the application container through the S2I process.

   c. Run diagnostic binaries available within the container.

NOTE

Root privileges are required to run some diagnostic binaries. In these situations you can start a debug Pod with root access, based on a problematic Pod's deployment configuration, by running **oc debug dc/<deployment_configuration> --as-root**. Then, you can run diagnostic binaries as root from within the debug Pod.

5. If diagnostic binaries are not available within a container, you can run a host's diagnostic binaries within a container's namespace by using **nsenter**. The following example runs **ip ad** within a container's namespace, using the host`s **ip** binary.

   a. Enter into a debug session on the target node. This step instantiates a debug Pod called **<node_name>-debug**:

   ```
   $ oc debug node/my-cluster-node
   ```

   b. Set **/host** as the root directory within the debug shell. The debug Pod mounts the host's root file system in **/host** within the Pod. By changing the root directory to **/host**, you can run binaries contained in the host's executable paths:

   ```
   # chroot /host
   ```

   NOTE

   OpenShift Container Platform 4.5 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes using SSH is not recommended and nodes will be tainted as *accessed*. However, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on the target node, **oc** operations will be impacted. In such situations, it is possible to access nodes using **ssh core@<node>.<cluster_name>.<base_domain>** instead.

   c. Determine the target container ID:

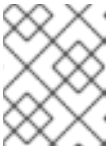
   ```
   # crictl ps
   ```

   d. Determine the container's process ID. In this example, the target container ID is **a7fe32346b120**:

   ```
   # crictl inspect a7fe32346b120 --output yaml | grep 'pid:' | awk '{print $2}'
   ```

   e. Run **ip ad** within the container's namespace, using the host's **ip** binary. This example uses **31150** as the container's process ID. The **nsenter** command enters the namespace of a target process and runs a command in its namespace. Because the target process in this example is a container's process ID, the **ip ad** command is run in the container's namespace from the host:

   ```
   # nsenter -n -t 31150 -- ip ad
   ```



NOTE

Running a host's diagnostic binaries within a container's namespace is only possible if you are using a privileged container such as a debug node.

5.6.4. Additional resources

- See Source-to-Image (S2I) build for more details about the S2I build strategy.

5.7. TROUBLESHOOTING STORAGE ISSUES

5.7.1. Resolving multi-attach errors

When a node crashes or shuts down abruptly, the attached ReadWriteOnce (RWO) volume is expected to be unmounted from the node so that it can be used by a Pod scheduled on another node.

However, mounting on a new node is not possible because the failed node is unable to unmount the attached volume.

A multi-attach error is reported:

Example output

```
Unable to attach or mount volumes: unmounted volumes=[sso-mysql-pvol], unattached volumes=
[sso-mysql-pvol default-token-x4rzc]: timed out waiting for the condition
Multi-Attach error for volume "pvc-8837384d-69d7-40b2-b2e6-5df86943eef9" Volume is already used
by pod(s) sso-mysql-1-ns6b4
```

To resolve the multi-attach issue, use one of the following solutions:

- Enable multiple attachments by using RWX volumes.
  For most storage solutions, you can use ReadWriteMany (RWX) volumes to prevent multi-attach errors.

- Recover or delete the failed node when using an RWO volume.
  For storage that does not support RWX, such as VMware vSphere, RWO volumes must be used instead. However, RWO volumes cannot be mounted on multiple nodes.

  If you encounter a multi-attach error message with an RWO volume, you can either recover or delete the failed node to make the volume available to other nodes.

5.8. DIAGNOSING OPENSHIFT CLI (oc) ISSUES

5.8.1. Understanding OpenShift CLI (oc) log levels

With the OpenShift CLI (**oc**), you can create applications and manage OpenShift Container Platform projects from a terminal.

If **oc** command-specific issues arise, increase the **oc** log level to output API request, API response, and **curl** request details generated by the command. This provides a granular view of a particular **oc** command's underlying operation, which in turn might provide insight into the nature of a failure.

oc log levels range from 1 to 10. The following table provides a list of **oc** log levels, along with their descriptions.

Table 5.3. OpenShift CLI (oc) log levels

Log level	Description
1 to 5	No additional logging to stderr.
6	Log API requests to stderr.
7	Log API requests and headers to stderr.
8	Log API requests, headers, and body, plus API response headers and body to stderr.
9	Log API requests, headers, and body, API response headers and body, plus **curl** requests to stderr.
10	Log API requests, headers, and body, API response headers and body, plus **curl** requests to stderr, in verbose detail.

5.8.2. Specifying OpenShift CLI (oc) log levels

You can investigate OpenShift CLI (**oc**) issues by increasing the command's log level.

Prerequisites

- You have installed the OpenShift CLI (**oc**).

Procedure

1. Specify the **oc** log level when running an **oc** command:

```
$ oc <options> --loglevel <log_level>
```

2. The OpenShift Container Platform user's current session token is typically included in logged **curl** requests where required. You can also obtain the current user's session token manually, for use when testing aspects of an **oc** command's underlying process step by step:

```
$ oc whoami -t
```