OpenShift Container Platform 4.9

Support

Getting support for OpenShift Container Platform

Last Updated: 2022-01-31

OpenShift Container Platform 4.9 Support

Getting support for OpenShift Container Platform

Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

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

Table of Contents

CHAPTER 1. SUPPORT OVERVIEW

Red Hat offers cluster administrators tools for gathering data for your cluster, monitoring, and troubleshooting.

1.1. GET SUPPORT

Get support: Visit the Red Hat Customer Portal to review knowledge base articles, submit a support case, and review additional product documentation and resources.

1.2. REMOTE HEALTH MONITORING ISSUES

Remote health monitoring issues: OpenShift Container Platform collects telemetry and configuration data about your cluster and reports it to Red Hat by using the Telemeter Client and the Insights Operator. Red Hat uses this data to understand and resolve issues in *connected cluster*. Similar to connected clusters, you can Use remote health monitoring in a restricted network . OpenShift Container Platform collects data and monitors health using the following:

- **Telemetry**: The Telemetry Client gathers and uploads the metrics values to Red Hat every four minutes and thirty seconds. Red Hat uses this data to:
    - Monitor the clusters.
    - Roll out OpenShift Container Platform upgrades.
    - Improve the upgrade experience.
- **Insight Operator**: By default, OpenShift Container Platform installs and enables the Insight Operator, which reports configuration and component failure status every two hours. The Insight Operator helps to:
    - Identify potential cluster issues proactively.
    - Provide a solution and preventive action in Red Hat OpenShift Cluster Manager.

You can Review telemetry information.

If you have enabled remote health reporting, Use Insights to identify issues . You can optionally disable remote health reporting.

1.3. GATHER DATA ABOUT YOUR CLUSTER

Gather data about your cluster : Red Hat recommends gathering your debugging information when opening a support case. This helps Red Hat Support to perform a root cause analysis. A cluster administrator can use the following to gather data about your cluster:

- **The must-gather tool**: Use the **must-gather** tool to collect information about your cluster and to debug the issues.
- **sosreport**: Use the **sosreport** tool to collect configuration details, system information, and diagnostic data for debugging purposes.
- **Cluster ID**: Obtain the unique identifier for your cluster, when providing information to Red Hat Support.

- **Bootstrap node journal logs**: Gather **bootkube.service journald** unit logs and container logs from the bootstrap node to troubleshoot bootstrap-related issues.
- **Cluster node journal logs**: Gather **journald** unit logs and logs within **/var/log** on individual cluster nodes to troubleshoot node-related issues.
- **A network trace**: Provide a network packet trace from a specific OpenShift Container Platform cluster node or a container to Red Hat Support to help troubleshoot network-related issues.
- **Diagnostic data**: Use the **redhat-support-tool** command to gather(?) diagnostic data about your cluster.

1.4. TROUBLESHOOTING ISSUES

A cluster administrator can monitor and troubleshoot the following OpenShift Container Platform component issues:

- Installation issues: OpenShift Container Platform installation proceeds through various stages. You can perform the following:
    - Monitor the installation stages.
    - Determine at which stage installation issues occur.
    - Investigate multiple installation issues.
    - Gather logs from a failed installation.
- Node issues: A cluster administrator can verify and troubleshoot node-related issues by reviewing the status, resource usage, and configuration of a node. You can query the following:
    - Kubelet’s status on a node.
    - Cluster node journal logs.
- Crio issues: A cluster administrator can verify CRI-O container runtime engine status on each cluster node. If you experience container runtime issues, perform the following:
    - Gather CRI-O journald unit logs.
    - Cleaning CRI-O storage.
- Operating system issues: OpenShift Container Platform runs on Red Hat Enterprise Linux CoreOS. If you experience operating system issues, you can investigate kernel crash procedures. Ensure the following:
    - Enable kdump.
    - Test the kdump configuration.
    - Analyze a core dump.
- Network issues: To troubleshoot Open vSwitch issues, a cluster administrator can perform the following:
    - Configure the Open vSwitch log level temporarily.
    - Configure the Open vSwitch log level permanently.

  - Display Open vSwitch logs.
- Operator issues: A cluster administrator can do the following to resolve Operator issues:
  - Verify Operator subscription status.
  - Check Operator pod health.
  - Gather Operator logs.
- Pod issues: A cluster administrator can troubleshoot pod-related issues by reviewing the status of a pod and completing the following:
  - Review pod and container logs.
  - Start debug pods with root access.
- Source-to-image issues: A cluster administrator can observe the S2I stages to determine where in the S2I process a failure occurred. Gather the following to resolve Source-to-Image (S2I) issues:
  - Source-to-Image diagnostic data.
  - Application diagnostic data to investigate application failure.
- Storage issues: A multi-attach storage error occurs when the mounting volume on a new node is not possible because the failed node cannot unmount the attached volume. A cluster administrator can do the following to resolve multi-attach storage issues:
  - Enable multiple attachments by using RWX volumes.
  - Recover or delete the failed node when using an RWO volume.
- Monitoring issues: A cluster administrator can follow the procedures on the troubleshooting page for monitoring. If the metrics for your user-defined projects are unavailable or if Prometheus is consuming a lot of disk space, check the following:
  - Investigate why user-defined metrics are unavailable.
  - Determine why Prometheus is consuming a lot of disk space.
- Logging issues: A cluster administrator can follow the procedures on the troubleshooting page for OpenShift Logging issues. Check the following to resolve logging issues:
  - Status of the Logging Operator.
  - Status of the Log store.
  - OpenShift Logging alerts.
  - Information about your OpenShift logging environment using **`oc adm must-gather`** command.
- OpenShift CLI (oc) issues: Investigate OpenShift CLI (oc) issues by increasing the log level.

CHAPTER 2. MANAGING YOUR CLUSTER RESOURCES

You can apply global configuration options in OpenShift Container Platform. Operators apply these configuration settings across the cluster.

2.1. INTERACTING WITH YOUR CLUSTER RESOURCES

You can interact with cluster resources by using the OpenShift CLI (**oc**) tool in OpenShift Container Platform. The cluster resources that you see after running the **oc api-resources** command can be edited.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have access to the web console or you have installed the **oc** CLI tool.

Procedure

1. To see which configuration Operators have been applied, run the following command:

   ```
   $ oc api-resources -o name | grep config.openshift.io
   ```

2. To see what cluster resources you can configure, run the following command:

   ```
   $ oc explain <resource_name>.config.openshift.io
   ```

3. To see the configuration of custom resource definition (CRD) objects in the cluster, run the following command:

   ```
   $ oc get <resource_name>.config -o yaml
   ```

4. To edit the cluster resource configuration, run the following command:

   ```
   $ oc edit <resource_name>.config -o yaml
   ```

CHAPTER 3. GETTING SUPPORT

3.1. 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.

To identify issues with your cluster, you can use Insights in Red Hat OpenShift Cluster Manager. Insights provides details about issues and, if available, information on how to solve a problem.

If you have a suggestion for improving this documentation or have found an error, 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.

3.2. 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.

3.3. 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.

3.4. 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 (**4.9**, 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**.

3.5. ADDITIONAL RESOURCES

- For details about identifying issues with your cluster, see Using Insights to identify issues with your cluster.

CHAPTER 4. REMOTE HEALTH MONITORING WITH CONNECTED CLUSTERS

4.1. ABOUT REMOTE HEALTH MONITORING

OpenShift Container Platform collects telemetry and configuration data about your cluster and reports it to Red Hat by using the Telemeter Client and the Insights Operator. The data that is provided to Red Hat enables the benefits outlined in this document.

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

Telemetry is the term that Red Hat uses to describe the information being sent to Red Hat by the OpenShift Container Platform Telemeter Client. Lightweight attributes are sent from connected clusters to Red Hat to enable subscription management automation, monitor the health of clusters, assist with support, and improve customer experience.

The **Insights Operator** gathers OpenShift Container Platform configuration data and sends it to Red Hat. The data is used to produce insights about potential issues that a cluster might be exposed to. These insights are communicated to cluster administrators on console.redhat.com/openshift.

More information is provided in this document about these two processes.

Telemetry and Insights Operator benefits

Telemetry and the Insights Operator enable the following benefits for end-users:

- **Enhanced identification and resolution of issues** Events that might seem normal to an end-user can be observed by Red Hat from a broader perspective across a fleet of clusters. Some issues can be more rapidly identified from this point of view and resolved without an end-user needing to open a support case or file a Bugzilla.

- **Advanced release management**. OpenShift Container Platform offers the **candidate**, **fast**, and **stable** release channels, which enable you to choose an update strategy. The graduation of a release from **fast** to **stable** is dependent on the success rate of updates and on the events seen during upgrades. With the information provided by connected clusters, Red Hat can improve the quality of releases to **stable** channels and react more rapidly to issues found in the **fast** channels.

- **Targeted prioritization of new features and functionality** The data collected provides insights about which areas of OpenShift Container Platform are used most. With this information, Red Hat can focus on developing the new features and functionality that have the greatest impact for our customers.

- **A streamlined support experience**. You can provide a cluster ID for a connected cluster when creating a support ticket on the Red Hat Customer Portal . This enables Red Hat to deliver a streamlined support experience that is specific to your cluster, by using the connected information. This document provides more information about that enhanced support experience.

- **Predictive analytics**. The insights displayed for your cluster on console.redhat.com/openshift are enabled by the information collected from connected clusters. Red Hat is investing in applying deep learning, machine learning, and artificial intelligence automation to help identify issues that OpenShift Container Platform clusters are exposed to.

4.1.1. About Telemetry

Telemetry sends a carefully chosen subset of the cluster monitoring metrics to Red Hat. The Telemeter Client fetches the metrics values every four minutes and thirty seconds and uploads the data to Red Hat. These metrics are described in this document.

This stream of data is used by Red Hat to monitor the 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 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 through support cases. All connected cluster information is used by Red Hat to help make OpenShift Container Platform better and more intuitive to use.

Additional resources

- See the OpenShift Container Platform update documentation for more information about updating or upgrading a cluster.

4.1.1.1. Information collected by Telemetry

The following information is collected by Telemetry:

- The unique random identifier that is generated during an installation
- Version information, including the OpenShift Container Platform cluster version and installed update details that are used to determine update version availability
- Update information, including the number of updates available per cluster, the channel and image repository used for an update, update progress information, and the number of errors that occur in an update
- The name of the provider platform that OpenShift Container Platform is deployed on and the data center location
- Sizing information about clusters, machine types, and machines, including the number of CPU cores and the amount of RAM used for each
- The number of etcd members and the number of objects stored in the etcd cluster
- The OpenShift Container Platform framework components installed in a cluster and their condition and status
- Usage information about components, features, and extensions
- Usage details about Technology Previews and unsupported configurations
- Information about degraded software
- Information about nodes that are marked as **NotReady**
- Events for all namespaces listed as "related objects" for a degraded Operator
- Configuration details that help Red Hat Support to provide beneficial support for customers, including node configuration at the cloud infrastructure level, hostnames, IP addresses, Kubernetes pod names, namespaces, and services

- Information about the validity of certificates
- Number of application builds by build strategy type

Telemetry does not collect identifying information such as user names or passwords. Red Hat does not intend to collect personal information. If Red Hat discovers that personal information has been inadvertently received, Red Hat will delete such information. To the extent that any telemetry data constitutes personal data, please refer to the Red Hat Privacy Statement for more information about Red Hat's privacy practices.

Additional resources

- See Showing data collected by Telemetry for details about how to list the attributes that Telemetry gathers from Prometheus in OpenShift Container Platform.
- See the upstream cluster-monitoring-operator source code for a list of the attributes that Telemetry gathers from Prometheus.
- Telemetry is installed and enabled by default. If you need to opt out of remote health reporting, see Opting out of remote health reporting .

4.1.2. About the Insights Operator

The Insights Operator periodically gathers configuration and component failure status and, by default, reports that data every two hours to Red Hat. This information enables Red Hat to assess configuration and deeper failure data than is reported through Telemetry.

Users of OpenShift Container Platform can display the report of each cluster in Red Hat OpenShift Cluster Manager. If any issues have been identified, Insights provides further details and, if available, steps on how to solve a problem.

The Insights Operator does not collect identifying information, such as user names, passwords, or certificates. See Red Hat Insights Data & Application Security for information about Red Hat Insights data collection and controls.

Red Hat uses all connected cluster information to:

- Proactively identify potential cluster issues and provide a solution and preventive actions in Red Hat OpenShift Cluster Manager
- Improve OpenShift Container Platform by providing aggregated and critical information to product and support teams
- Make OpenShift Container Platform more intuitive

Additional resources

- The Insights Operator is installed and enabled by default. If you need to opt out of remote health reporting, see Opting out of remote health reporting .

4.1.2.1. Information collected by the Insights Operator

The following information is collected by the Insights Operator:

- General information about your cluster and its components to identify issues that are specific to your OpenShift Container Platform version and environment

- Configuration files, such as the image registry configuration, of your cluster to determine incorrect settings and issues that are specific to parameters you set
- Errors that occur in the cluster components
- Progress information of running updates, and the status of any component upgrades
- 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
- Cluster workload information transformed into discreet Secure Hash Algorithm (SHA) values, which allows Red Hat to assess workloads for security and version vulnerabilities without disclosing sensitive details
- If an Operator reports an issue, information is collected about core OpenShift Container Platform pods in the **openshift-*** and **kube-*** projects. This includes state, resource, security context, volume information, and more.

Additional resources

- See Showing data collected by the Insights Operator for details about how to review the data that is collected by the Insights Operator.
- The Insights Operator source code is available for review and contribution. See the Insights Operator upstream project for a list of the items collected by the Insights Operator.

4.1.3. Understanding Telemetry and Insights Operator data flow

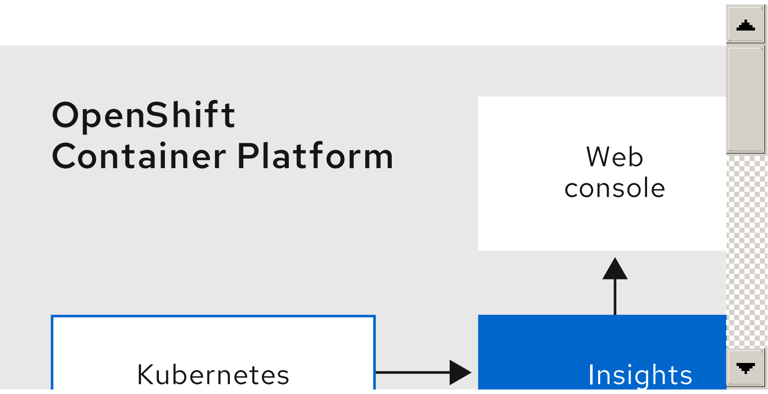
The Telemeter Client collects selected time series data from the Prometheus API. The time series data is uploaded to api.openshift.com every four minutes and thirty seconds for processing.

The Insights Operator gathers selected data from the Kubernetes API and the Prometheus API into an archive. The archive is uploaded to console.redhat.com every two hours for processing. The Insights Operator also downloads the latest Insights analysis from console.redhat.com. This is used to populate the **Insights status** pop-up that is included in the **Overview** page in the OpenShift Container Platform web console.

All of the communication with Red Hat occurs over encrypted channels by using Transport Layer Security (TLS) and mutual certificate authentication. All of the data is encrypted in transit and at rest.

Access to the systems that handle customer data is controlled through multi-factor authentication and strict authorization controls. Access is granted on a need-to-know basis and is limited to required operations.

Telemetry and Insights Operator data flow



Additional resources

- See Understanding the monitoring stack for more information about the OpenShift Container Platform monitoring stack.
- See Configuring your firewall for details about configuring a firewall and enabling endpoints for Telemetry and Insights

4.1.4. Additional details about how remote health monitoring data is used

The information collected to enable remote health monitoring is detailed in Information collected by Telemetry and Information collected by the Insights Operator.

As further described in the preceding sections of this document, Red Hat collects data about your use of the Red Hat Product(s) for purposes such as providing support and upgrades, optimizing performance or configuration, minimizing service impacts, identifying and remediating threats, troubleshooting, improving the offerings and user experience, responding to issues, and for billing purposes if applicable.

Collection safeguards

Red Hat employs technical and organizational measures designed to protect the telemetry and configuration data.

Sharing

Red Hat may share the data collected through Telemetry and the Insights Operator internally within Red Hat to improve your user experience. Red Hat may share telemetry and configuration data with its business partners in an aggregated form that does not identify customers to help the partners better understand their markets and their customers' use of Red Hat offerings or to ensure the successful integration of products jointly supported by those partners.

Third party service providers

Red Hat may engage certain service providers to assist in the collection and storage of the telemetry and configuration data.

User control / enabling and disabling telemetry and configuration data collection

You may disable OpenShift Container Platform Telemetry and the Insights Operator by following the instructions in Opting out of remote health reporting .

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 may choose to opt out of reporting health and usage data for your cluster.

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 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. Connected clusters also help to simplify the subscription and entitlement process and enable the Red Hat OpenShift Cluster Manager service to provide an overview of your clusters and their subscription status.

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 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 console.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 by either replacing the current pull secret or appending a new pull secret.

IMPORTANT

To transfer your cluster to another owner, you must first initiate the transfer in Red Hat OpenShift Cluster Manager, and then update the pull secret on the cluster. Updating a cluster's pull secret without initiating the transfer in OpenShift Cluster Manager causes the cluster to stop reporting Telemetry metrics in OpenShift Cluster Manager.

For more information about transferring cluster ownership, see "Transferring cluster ownership" in the Red Hat OpenShift Cluster Manager documentation.

WARNING

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

Prerequisites

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

Procedure

1. Optional: To append a new pull secret to the existing pull secret, complete the following steps:
   a. Enter the following command to download the pull secret:

```
$ oc get secret/pull-secret -n openshift-config --template='{{index .data
".dockerconfigjson" | base64decode}}' ><pull_secret_location> ①
```

① Provide the path to the pull secret file.

   b. Enter the following command to add the new pull secret:

```
$ oc registry login --registry="<registry>" \ ①
--auth-basic="<username>:<password>" \ ②
--to=<pull_secret_location> ③
```

① Provide the new registry. You can include multiple repositories within the same registry, for example: **--registry="<registry/my-namespace/my-repository>"**.

② Provide the credentials of the new registry.

③ Provide the path to the pull secret file.

Alternatively, you can perform a manual update to the pull secret file.

2. Enter 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.

NOTE

As of OpenShift Container Platform 4.7.4, changes to the global pull secret no longer trigger a node drain or reboot.

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**.

4.4.2. Displaying the Insights status in the web console

Insights repeatedly analyzes your cluster and you can display the status of identified potential issues of your cluster in the OpenShift Container Platform web console. This status shows the number of issues in the different categories and, for further details, links to the reports in the Red Hat OpenShift Cluster Manager.

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 OpenShift Container Platform web console.

Procedure

1. Navigate to **Home** → **Overview** in the OpenShift Container Platform web console.

2. Click **Insights** on the **Status** card.
   The pop-up window lists potential issues grouped by priority. Click the individual categories or **View all in OpenShift Container Platform** to display further details.

4.5. USING REMOTE HEALTH REPORTING IN A RESTRICTED NETWORK

You can manually gather and upload Insights Operator archives to diagnose issues from a restricted network.

To use the Insights Operator in a restricted network, you must:

- Create a copy of your Insights Operator archive.

- Upload the Insights Operator archive to console.redhat.com.

Additionally, you can choose to obfuscate the Insights Operator data before upload.

4.5.1. Running an Insights Operator gather operation

You must run a gather operation to create an Insights Operator archive.

Prerequisites

- You are logged in to OpenShift Container Platform as **cluster-admin**.

Procedure

1. Create a file named **gather-job.yaml** using this template:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: insights-operator-job
  annotations:
    config.openshift.io/inject-proxy: insights-operator
spec:
  backoffLimit: 6
  ttlSecondsAfterFinished: 600
  template:
    spec:
      restartPolicy: OnFailure
      serviceAccountName: operator
      nodeSelector:
        beta.kubernetes.io/os: linux
        node-role.kubernetes.io/master: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
        operator: Exists
      - effect: NoExecute
        key: node.kubernetes.io/unreachable
        operator: Exists
        tolerationSeconds: 900
      - effect: NoExecute
        key: node.kubernetes.io/not-ready
        operator: Exists
        tolerationSeconds: 900
      volumes:
      - name: snapshots
        emptyDir: {}
      - name: service-ca-bundle
        configMap:
          name: service-ca-bundle
          optional: true
      initContainers:
      - name: insights-operator
        image: quay.io/openshift/origin-insights-operator:latest
        terminationMessagePolicy: FallbackToLogsOnError
        volumeMounts:
        - name: snapshots
          mountPath: /var/lib/insights-operator
        - name: service-ca-bundle
          mountPath: /var/run/configmaps/service-ca-bundle
```

```
          readOnly: true
        ports:
        - containerPort: 8443
          name: https
        resources:
          requests:
            cpu: 10m
            memory: 70Mi
        args:
        - gather
        - -v=4
        - --config=/etc/insights-operator/server.yaml
      containers:
        - name: sleepy
          image: quay.io/openshift/origin-base:latest
          args:
            - /bin/sh
            - -c
            - sleep 10m
          volumeMounts: [{name: snapshots, mountPath: /var/lib/insights-operator}]
```

2. Copy your **insights-operator** image version:

```
$ oc get -n openshift-insights deployment insights-operator -o yaml
```

3. Paste your image version in **gather-job.yaml**:

```
initContainers:
      - name: insights-operator
        image: <your_insights_operator_image_version>
        terminationMessagePolicy: FallbackToLogsOnError
        volumeMounts:
```

4. Create the gather job:

```
$ oc apply -n openshift-insights -f gather-job.yaml
```

5. Find the name of the job pod:

```
$ oc describe -n openshift-insights job/insights-operator-job
```

Example output

```
Events:
  Type    Reason            Age    From            Message
  ----    ------            ----   ----            -------
  Normal  SuccessfulCreate  7m18s  job-controller  Created pod: insights-operator-
job-<your_job>
```

where **insights-operator-job-*<your_job>*** is the name of the pod.

6. Verify that the operation has finished:

```
$ oc logs -n openshift-insights insights-operator-job-<your_job> insights-operator
```

Example output

```
I0407 11:55:38.192084       1 diskrecorder.go:34] Wrote 108 records to disk in 33ms
```

7. Save the created archive:

```
$ oc cp openshift-insights/insights-operator-job-<your_job>:/var/lib/insights-operator
./insights-data
```

8. Clean up the job:

```
$ oc delete -n openshift-insights job insights-operator-job
```

4.5.2. Uploading an Insights Operator archive

You can manually upload an Insights Operator archive to console.redhat.com to diagnose potential issues.

Prerequisites

- You are logged in to OpenShift Container Platform as **cluster-admin**.
- You have a workstation with unrestricted internet access.
- You have created a copy of the Insights Operator archive.

Procedure

1. Download the **dockerconfig.json** file:

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

2. Copy your **"cloud.openshift.com" "auth"** token from the **dockerconfig.json** file:

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "<your_token>",
      "email": "asd@redhat.com"
    }
}
```

3. Upload the archive to console.redhat.com:

```
$ curl -v -H "User-Agent: insights-operator/one10time200gather184a34f6a168926d93c330
cluster/<cluster_id>" -H "Authorization: Bearer <your_token>" -F
"upload=@<path_to_archive>; type=application/vnd.redhat.openshift.periodic+tar"
https://console.redhat.com/api/ingress/v1/upload
```

where ***<cluster_id>*** is your cluster ID, ***<your_token>*** is the token from your pull secret, and ***<path_to_archive>*** is the path to the Insights Operator archive.

If the operation is successful, the command returns a **"request_id"** and **"account_number"**:

Example output

```
* Connection #0 to host console.redhat.com left intact
{"request_id":"393a7cf1093e434ea8dd4ab3eb28884c","upload":
{"account_number":"6274079"}}%
```

Verification steps

1. Log in to https://console.redhat.com/openshift.
2. Click the **Clusters** menu in the left pane.
3. To display the details of the cluster, click the cluster name.
4. Open the **Insights Advisor** tab of the cluster.
   If the upload was successful, the tab displays one of the following:
   - **Your cluster passed all recommendations**, if Insights Advisor did not identify any issues.
   - A list of issues that Insights Advisor has detected, prioritized by risk (low, moderate, important, and critical).

4.5.3. Enabling Insights Operator data obfuscation

You can enable obfuscation to mask sensitive and identifiable IPv4 addresses and cluster base domains that the Insights Operator sends to console.redhat.com.

> **WARNING**
>
> Although this feature is available, Red Hat recommends keeping obfuscation disabled for a more effective support experience.

Obfuscation assigns non-identifying values to cluster IPv4 addresses, and uses a translation table that is retained in memory to change IP addresses to their obfuscated versions throughout the Insights Operator archive before uploading the data to console.redhat.com.

For cluster base domains, obfuscation changes the base domain to a hardcoded substring. For example, **cluster-api.openshift.example.com** becomes **cluster-api.<CLUSTER_BASE_DOMAIN>**.

The following procedure enables obfuscation using the **support** secret in the **openshift-config** namespace.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as **cluster-admin**.

Procedure

1. Navigate to **Workloads** → **Secrets**.
2. Select the **openshift-config** project.
3. Search for the **support** secret using the **Search by name** field. If it does not exist, click **Create** → **Key/value secret** to create it.
4. Click the **Options** menu ⋮, and then click **Edit Secret**.
5. Click **Add Key/Value**.
6. Create a key named **enableGlobalObfuscation** with a value of **true**, and click **Save**.
7. Navigate to **Workloads** → **Pods**
8. Select the **openshift-insights** project.
9. Find the **insights-operator** pod.
10. To restart the **insights-operator** pod, click the **Options** menu ⋮, and then click **Delete Pod**.

Verification

1. Navigate to **Workloads** → **Secrets**.
2. Select the **openshift-insights** project.
3. Search for the **obfuscation-translation-table** secret using the **Search by name** field.

If the **obfuscation-translation-table** secret exists, then obfuscation is enabled and working.

Alternatively, you can inspect **/insights-operator/gathers.json** in your Insights Operator archive for the value **"is_global_obfuscation_enabled": true**.

Additional resources

- For more information on how to download your Insights Operator archive, see Showing data collected by the Insights Operator.

4.6. IMPORTING RHEL SIMPLE CONTENT ACCESS CERTIFICATES WITH INSIGHTS OPERATOR

Insights Operator can import your RHEL Simple Content Access (SCA) certificates from Red Hat OpenShift Cluster Manager. SCA is a capability in Red Hat's subscription tools which simplifies the behavior of the entitlement tooling. It is easier to consume the content provided by your Red Hat subscriptions without the complexity of configuring subscription tooling. After importing the certificates, they are stored in the **etc-pki-entitlement** secret in the **openshift-config-managed** namespace.

Insights Operator imports SCA certificates every 8 hours by default, but can be configured or disabled using the **support** secret in the **openshift-config** namespace.

In OpenShift Container Platform 4.9, this feature is in Technology Preview and must be enabled using the **TechPreviewNoUpgrade** Feature Set. See *Enabling OpenShift Container Platform features using FeatureGates* for more information.

For more information about Simple Content Access certificates see the *Simple Content Access* article in the Red Hat Knowledgebase.

IMPORTANT

InsightsOperatorPullingSCA is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

4.6.1. Configuring Simple Content Access import interval

You can configure how often the Insights Operator imports the RHEL Simple Content Access (SCA) certificates using the **support** secret in the **openshift-config** namespace. The certificate import normally occurs every 8 hours, but you may want to shorten this interval if you update your SCA configuration in Red Hat Subscription Management.

This procedure describes how to update the import interval to one hour.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as **cluster-admin**.

Procedure

1. Navigate to **Workloads** → **Secrets**.
2. Select the **openshift-config** project.
3. Search for the **support** secret using the **Search by name** field. If it does not exist, click **Create** → **Key/value secret** to create it.
4. Click the **Options** menu ⋮, and then click **Edit Secret**.
5. Click **Add Key/Value**.
6. Create a key named **ocmInterval** with a value of **1h**, and click **Save**.

   NOTE

   The interval **1h** can also be entered as **60m** for 60 minutes.

7. Navigate to **Workloads** → **Pods**
8. Select the **openshift-insights** project.

9. Find the **insights-operator** pod.

10. To restart the **insights-operator** pod, click the **Options** menu ⋮, and then click **Delete Pod**.

4.6.2. Disabling Simple Content Access import

You can disable the import of RHEL Simple Content Access certificates using the **support** secret in the **openshift-config** namespace.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as **cluster-admin**.

Procedure

1. Navigate to **Workloads** → **Secrets**.
2. Select the **openshift-config** project.
3. Search for the **support** secret using the **Search by name** field. If it does not exist, click **Create** → **Key/value secret** to create it.
4. Click the **Options** menu ⋮, and then click **Edit Secret**.
5. Click **Add Key/Value**.
6. Create a key named **ocmPullDisabled** with a value of **true**, and click **Save**.
7. Navigate to **Workloads** → **Pods**
8. Select the **openshift-insights** project.
9. Find the **insights-operator** pod.
10. To restart the **insights-operator** pod, click the **Options** menu ⋮, and then click **Delete Pod**.

CHAPTER 5. GATHERING DATA ABOUT YOUR CLUSTER

When opening a support case, it is 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

5.1. 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, including:

- Resource definitions
- Service logs

By default, the **oc adm must-gather** command uses the default plug-in image and writes into **./must-gather.local**.

Alternatively, you can collect specific information by running the command with the appropriate arguments as described in the following sections:

- To collect data related to one or more specific features, use the **--image** argument with an image, as listed in a following section.
  For example:

  ```
  $ oc adm must-gather --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.9.0
  ```

- To collect the audit logs, use the **-- /usr/bin/gather_audit_logs** argument, as described in a following section.
  For example:

  ```
  $ oc adm must-gather -- /usr/bin/gather_audit_logs
  ```

  NOTE

  Audit logs are not collected as part of the default set of information to reduce the size of the files.

When you run **oc adm must-gather**, a new pod with a random name is created in a new project 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.

For example:

```
NAMESPACE                      NAME                 READY   STATUS      RESTARTS      AGE
...
openshift-must-gather-5drcj    must-gather-bklx4    2/2     Running     0             72s
```

```
openshift-must-gather-5drcj   must-gather-s8sdh   2/2    Running    0          72s
...
```

5.1.1. 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 take additional steps. If your mirror registry has a trusted CA, you must first add the trusted CA to the cluster. For all clusters on restricted networks, you must import the default **must-gather** image as an image stream before you use 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 .

5.1.2. 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 5.1. Supported must-gather images

Image	Purpose
registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.9.2	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-rhel8	Data collection for Red Hat OpenShift Service Mesh.
registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v1.6	Data collection for the Migration Toolkit for Containers.
registry.redhat.io/ocs4/ocs-must-gather-rhel8:v4.9	Data collection for Red Hat OpenShift Container Storage.
registry.redhat.io/openshift-logging/cluster-logging-rhel8-operator	Data collection for OpenShift Logging.
registry.redhat.io/openshift4/ose-local-storage-mustgather-rhel8	Data collection for Local Storage Operator.

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:v4.9.2 ②
```

① The default OpenShift Container Platform **must-gather** image

② The must-gather image for OpenShift Virtualization

You can use the **must-gather** tool with additional arguments to gather data that is specifically related to OpenShift Logging and the Red Hat OpenShift Logging Operator in your cluster. For OpenShift Logging, run the following command:

```
$ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator \
 -o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-operator")].image}')
```

Example 5.1. Example must-gather output for OpenShift Logging

```
├── cluster-logging
│   ├── clo
│   │   ├── cluster-logging-operator-74dd5994f-6ttgt
│   │   ├── clusterlogforwarder_cr
│   │   ├── cr
│   │   ├── csv
│   │   ├── deployment
│   │   └── logforwarding_cr
│   ├── collector
│   │   ├── fluentd-2tr64
│   ├── eo
│   │   ├── csv
│   │   ├── deployment
│   │   └── elasticsearch-operator-7dc7d97b9d-jb4r4
│   ├── es
│   │   ├── cluster-elasticsearch
│   │   │   ├── aliases
│   │   │   ├── health
│   │   │   ├── indices
│   │   │   ├── latest_documents.json
│   │   │   ├── nodes
│   │   │   ├── nodes_stats.json
│   │   │   └── thread_pool
│   │   ├── cr
│   │   ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
│   │   └── logs
│   │       ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
│   ├── install
│   │   ├── co_logs
│   │   ├── install_plan
│   │   ├── olmo_logs
│   │   └── subscription
│   └── kibana
│       ├── cr
│       ├── kibana-9d69668d4-2rkvz
├── cluster-scoped-resources
│   └── core
│       ├── nodes
│       │   ├── ip-10-0-146-180.eu-west-1.compute.internal.yaml
│       └── persistentvolumes
```

```
│       ├── pvc-0a8d65d9-54aa-4c44-9ecc-33d9381e41c1.yaml
├── event-filter.html
├── gather-debug.log
└── namespaces
    ├── openshift-logging
    │   ├── apps
    │   │   ├── daemonsets.yaml
    │   │   ├── deployments.yaml
    │   │   ├── replicasets.yaml
    │   │   └── statefulsets.yaml
    │   ├── batch
    │   │   ├── cronjobs.yaml
    │   │   └── jobs.yaml
    │   ├── core
    │   │   ├── configmaps.yaml
    │   │   ├── endpoints.yaml
    │   │   ├── events
    │   │   │   ├── elasticsearch-im-app-1596020400-gm6nl.1626341a296c16a1.yaml
    │   │   │   ├── elasticsearch-im-audit-1596020400-9l9n4.1626341a2af81bbd.yaml
    │   │   │   ├── elasticsearch-im-infra-1596020400-v98tk.1626341a2d821069.yaml
    │   │   │   ├── elasticsearch-im-app-1596020400-cc5vc.1626341a3019b238.yaml
    │   │   │   ├── elasticsearch-im-audit-1596020400-s8d5s.1626341a31f7b315.yaml
    │   │   │   ├── elasticsearch-im-infra-1596020400-7mgv8.1626341a35ea59ed.yaml
    │   │   ├── events.yaml
    │   │   ├── persistentvolumeclaims.yaml
    │   │   ├── pods.yaml
    │   │   ├── replicationcontrollers.yaml
    │   │   ├── secrets.yaml
    │   │   └── services.yaml
    │   ├── openshift-logging.yaml
    │   ├── pods
    │   │   ├── cluster-logging-operator-74dd5994f-6ttgt
    │   │   │   ├── cluster-logging-operator
    │   │   │   │   └── cluster-logging-operator
    │   │   │   │       └── logs
    │   │   │   │           ├── current.log
    │   │   │   │           ├── previous.insecure.log
    │   │   │   │           └── previous.log
    │   │   │   └── cluster-logging-operator-74dd5994f-6ttgt.yaml
    │   │   ├── cluster-logging-operator-registry-6df49d7d4-mxxff
    │   │   │   ├── cluster-logging-operator-registry
    │   │   │   │   └── cluster-logging-operator-registry
    │   │   │   │       └── logs
    │   │   │   │           ├── current.log
    │   │   │   │           ├── previous.insecure.log
    │   │   │   │           └── previous.log
    │   │   │   ├── cluster-logging-operator-registry-6df49d7d4-mxxff.yaml
    │   │   │   └── mutate-csv-and-generate-sqlite-db
    │   │   │       └── mutate-csv-and-generate-sqlite-db
    │   │   │           └── logs
    │   │   │               ├── current.log
    │   │   │               ├── previous.insecure.log
    │   │   │               └── previous.log
    │   │   ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
    │   │   ├── elasticsearch-im-app-1596030300-bpgcx
    │   │   │   ├── elasticsearch-im-app-1596030300-bpgcx.yaml
```

```
│   │   │   └── indexmanagement
│   │   │       └── indexmanagement
│   │   │           └── logs
│   │   │               ├── current.log
│   │   │               ├── previous.insecure.log
│   │   │               └── previous.log
│   │   ├── fluentd-2tr64
│   │   │   ├── fluentd
│   │   │   │   └── fluentd
│   │   │   │       └── logs
│   │   │   │           ├── current.log
│   │   │   │           ├── previous.insecure.log
│   │   │   │           └── previous.log
│   │   │   ├── fluentd-2tr64.yaml
│   │   │   └── fluentd-init
│   │   │       └── fluentd-init
│   │   │           └── logs
│   │   │               ├── current.log
│   │   │               ├── previous.insecure.log
│   │   │               └── previous.log
│   │   ├── kibana-9d69668d4-2rkvz
│   │   │   ├── kibana
│   │   │   │   └── kibana
│   │   │   │       └── logs
│   │   │   │           ├── current.log
│   │   │   │           ├── previous.insecure.log
│   │   │   │           └── previous.log
│   │   │   ├── kibana-9d69668d4-2rkvz.yaml
│   │   │   └── kibana-proxy
│   │   │       └── kibana-proxy
│   │   │           └── logs
│   │   │               ├── current.log
│   │   │               ├── previous.insecure.log
│   │   │               └── previous.log
│   └── route.openshift.io
│       └── routes.yaml
└── openshift-operators-redhat
    ├── ...
```

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 .

5.1.3. Gathering audit logs

You can gather audit logs, which are a security-relevant chronological set of records documenting the sequence of activities that have affected the system by individual users, administrators, or other components of the system. You can gather audit logs for:

- etcd server
- Kubernetes API server
- OpenShift OAuth API server
- OpenShift API server

Procedure

1. Run the **oc adm must-gather** command with the **-- /usr/bin/gather_audit_logs** flag:

```
$ oc adm must-gather -- /usr/bin/gather_audit_logs
```

2. 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.472290403699006248 ①
```

① Replace **must-gather-local.472290403699006248** with the actual directory name.

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

5.2. 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"}'
```

5.3. 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**.

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

The recommended way to generate a **sosreport** for an OpenShift Container Platform 4.9 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
   ```

   To enter into a debug session on the target node that is tainted with the **NoExecute** effect, add a toleration to a dummy namespace, and start the debug pod in the dummy namespace:

   ```
   $ oc new-project dummy
   ```

   ```
   $ oc patch namespace dummy --type=merge -p '{"metadata": {"annotations": {
   "scheduler.alpha.kubernetes.io/defaultTolerations": "[{\"operator\": \"Exists\"}]"}}}'
   ```

```
$ 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 4.9 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** plug-ins.

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.

     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 4.9 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.

5.5. 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 control plane nodes. After etcd has started on each control plane 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'
   ```

5.6. 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 control plane 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 control plane 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 control plane 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 4.9 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>**.

5.7. 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 4.9 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 4.9 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.

5.8. 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 4.9 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 4.9 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.

5.9. ABOUT TOOLBOX

toolbox is a tool that starts a container on a Red Hat Enterprise Linux CoreOS (RHCOS) system. The tool is primarily used to start a container that includes the required binaries and plug-ins that are needed to run commands such as **sosreport** and **redhat-support-tool**.

The primary purpose for a **toolbox** container is to gather diagnostic information and to provide it to Red Hat Support. However, if additional diagnostic tools are required, you can add RPM packages or run an image that is an alternative to the standard support tools image.

Installing packages to a toolbox container

By default, running the **toolbox** command starts a container with the **registry.redhat.io/rhel8/support-tools:latest** image. This image contains the most frequently used support tools. If you need to collect node-specific data that requires a support tool that is not part of the image, you can install additional packages.

Prerequisites

- You have accessed a node with the **oc debug node/<node_name>** command.

Procedure

1. 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
```

2. Start the toolbox container:

```
# toolbox
```

3. Install the additional package, such as **wget**:

```
# dnf install -y <package_name>
```

Starting an alternative image with **toolbox**

By default, running the **toolbox** command starts a container with the **registry.redhat.io/rhel8/support-tools:latest** image. You can start an alternative image by creating a **.toolboxrc** file and specifying the image to run.

Prerequisites

- You have accessed a node with the **oc debug node/<node_name>** command.

Procedure

1. 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
   ```

2. Create a **.toolboxrc** file in the home directory for the root user ID:

   ```
   # vi ~/.toolboxrc
   ```

   ```
   REGISTRY=quay.io                <.>
   IMAGE=fedora/fedora:33-x86_64   <.>
   TOOLBOX_NAME=toolbox-fedora-33  <.>
   ```

   <.> Optional: Specify an alternative container registry. <.> Specify an alternative image to start. <.> Optional: Specify an alternative name for the toolbox container.

3. Start a toolbox container with the alternative image:

   ```
   # 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** plug-ins.

CHAPTER 6. SUMMARIZING CLUSTER SPECIFICATIONS

6.1. 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 7. TROUBLESHOOTING

7.1. TROUBLESHOOTING INSTALLATIONS

7.1.1. 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 control plane machines to boot.
3. The control plane machines fetch the remote resources from the bootstrap machine and finish booting.
4. The control plane 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 control plane 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.

7.1.2. 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 4.9 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 daemon set.
- Install cloud provider integration if you want to enable features such as dynamic storage, on-demand service routing, node hostname to Kubernetes hostname 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 machine sets 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 control plane 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.

7.1.3. 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>
    ```

7.1.4. 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**.

7.1.5. 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.

7.1.6. 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 control plane 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 control plane nodes. After etcd has started on each control plane node and the nodes have joined the cluster, the errors should stop.

3. Monitor **kubelet.service** journald unit logs on control plane nodes, after they have booted. This provides visibility into control plane 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 control plane nodes, after they have booted. This provides visibility into control plane 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
    ```

7.1.7. 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 control plane nodes. After etcd has started on each control plane 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 control plane nodes. Ensure that the proxy configuration meets OpenShift Container Platform installation requirements.

7.1.8. Investigating control plane node installation issues

If you experience control plane node installation issues, determine the control plane node OpenShift Container Platform software defined network (SDN), and network Operator status. Collect **kubelet.service**, **crio.service** journald unit logs, and control plane node container logs for visibility into control plane 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 control plane 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 console for the control plane node, 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 control plane 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 control plane 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 console for the control plane node to determine if the mechanism is injecting the control plane node Ignition file correctly.
3. Check the availability of the storage device assigned to the control plane node.
4. Verify that the control plane node has been assigned an IP address from the DHCP server.

5. Determine control plane node status.

    a. Query control plane node status:

    ```
    $ oc get nodes
    ```

    b. If one of the control plane 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** daemon set 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 the pod status in the **openshift-network-operator** namespace to determine whether the Cluster Network Operator (CNO) 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 control plane nodes, after they have booted. This provides visibility into control plane node agent activity.

    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 4.9 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 control plane nodes, after they have booted. This provides visibility into control plane 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 control plane 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 control plane 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 control plane 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 control plane 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 control plane 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
    ```

7.1.9. 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 control plane 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 control plane 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 control plane node by using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values.
   a. List etcd pods on each control plane 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 4.9 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 control plane nodes.

7.1.10. Investigating control plane node kubelet and API server issues

To investigate control plane 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 control plane nodes.

Procedure

1. Verify that the API server's DNS record directs the kubelet on control plane 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 control plane node.

3. Check that unique control plane node hostnames have been provided by DHCP.

4. Inspect the **kubelet.service** journald unit logs on each control plane 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 4.9 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 control plane 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'
   ```

7.1.11. 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 control plane 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 4.9 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
    ```

7.1.12. 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. Verify that all of the required certificate signing requests (CSRs) are approved. Some nodes might not move to a **Ready** status and some cluster Operators might not become available if there are pending CSRs.

   a. Check the status of the CSRs and ensure that you see a client and server request with the **Pending** or **Approved** status for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   Example output

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

   ① A client request CSR.

   ② A server request CSR.

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

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

   NOTE

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

> NOTE
>
> For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

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

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

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

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

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

3. View Operator events:

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

4. Review Operator pod status within the Operator’s namespace:

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

5. Obtain a detailed description for pods that do not have **Running** status:

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

6. Inspect pod logs:

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

7. 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
      ```

7.1.13. 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 control plane 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, change to the directory that contains the installation program and 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 hostnames or IP addresses.

   - If you used infrastructure that you provisioned yourself, change to the directory that contains the installation program and 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.

7.1.14. Additional resources

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

7.2. VERIFYING NODE HEALTH

7.2.1. 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
```

7.2.2. 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
      ```

7.2.3. 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 control plane 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 control plane 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 control plane 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 4.9 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>**.

7.3. TROUBLESHOOTING CRI-O CONTAINER RUNTIME ISSUES

7.3.1. 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.

7.3.2. 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 4.9 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
    ```

7.3.3. 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 control plane machines.

Procedure

1. Gather CRI-O journald unit logs. The following example collects logs from all control plane 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 4.9 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>**.

7.3.4. Cleaning CRI-O storage

You can manually clear the CRI-O ephemeral storage if you experience the following issues:

- A node cannot run on any pods and this error appears:

```
Failed to create pod sandbox: rpc error: code = Unknown desc = failed to mount container
XXX: error recreating the missing symlinks: error reading name of symlink for XXX: open
/var/lib/containers/storage/overlay/XXX/link: no such file or directory
```

- You cannot create a new container on a working node and the “can’t stat lower layer” error appears:

```
can't stat lower layer ...  because it does not exist.  Going through storage to recreate the
missing symlinks.
```

- Your node is in the **NotReady** state after a cluster upgrade or if you attempt to reboot it.
- The container runtime implementation (**crio**) is not working properly.
- You are unable to start a debug shell on the node using **oc debug node/<nodename>** because the container runtime instance (**crio**) is not working.

Follow this process to completely wipe the CRI-O storage and resolve the errors.

Prerequisites:

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

Procedure

1. Use **cordon** on the node. This is to avoid any workload getting scheduled if the node gets into the **Ready** status. You will know that scheduling is disabled when **SchedulingDisabled** is in your Status section:

   ```
   $ oc adm cordon <nodename>
   ```

2. Drain the node as the cluster-admin user:

   ```
   $ oc adm drain <nodename> --ignore-daemonsets --delete-local-data
   ```

3. When the node returns, connect back to the node via SSH or Console. Then connect to the root user:

   ```
   $ ssh core@node1.example.com
   $ sudo -i
   ```

4. Manually stop the kubelet:

   ```
   # systemctl stop kubelet
   ```

5. Stop the containers and pods:

   ```
   # crictl rmp -fa
   ```

6. Manually stop the crio services:

   ```
   # systemctl stop crio
   ```

7. After you run those commands, you can completely wipe the ephemeral storage:

   ```
   # crio wipe -f
   ```

8. Start the crio and kubelet service:

```
# systemctl start crio
# systemctl start kubelet
```

9. You will know if the clean up worked if the crio and kubelet services are started, and the node is in the **Ready** status:

```
$ oc get nodes
```

Example output

```
NAME        STATUS                  ROLES    AGE    VERSION
ci-ln-tkbxyft-f76d1-nvwhr-master-1   Ready, SchedulingDisabled   master   133m   v1.22.0-
rc.0+75ee307
```

10. Mark the node schedulable. You will know that the scheduling is enabled when **SchedulingDisabled** is no longer in status:

```
$ oc adm uncordon <nodename>
```

Example output

```
NAME        STATUS      ROLES    AGE    VERSION
ci-ln-tkbxyft-f76d1-nvwhr-master-1   Ready          master   133m   v1.22.0-rc.0+75ee307
```

= Troubleshooting operating system issues :experimental: :imagesdir: images :prewrap!: :op-system-first: Red Hat Enterprise Linux CoreOS (RHCOS) :op-system: RHCOS :op-system-base: RHEL :op-system-base-full: Red Hat Enterprise Linux (RHEL) :tsb-name: Template Service Broker :kebab: ⋮ :rh-openstack-first: Red Hat OpenStack Platform (RHOSP) :rh-openstack: RHOSP :console-redhat-com: Red Hat OpenShift Cluster Manager :rh-storage-first: Red Hat OpenShift Container Storage :rh-storage: OpenShift Container Storage :rh-rhacm-first: Red Hat Advanced Cluster Management (RHACM) :rh-rhacm: RHACM :sandboxed-containers-first: OpenShift sandboxed containers :sandboxed-containers-operator: OpenShift sandboxed containers Operator :rh-virtualization-first: Red Hat Virtualization (RHV) :rh-virtualization: RHV :rh-virtualization-engine-name: Manager :launch: ▦ :mtc-short: MTC :mtc-full: Migration Toolkit for Containers :mtc-version: 1.6 :mtc-legacy-version: 1.5 :mtc-legacy-version-z: 1.5.3 :context: troubleshooting-operating-system-issues

OpenShift Container Platform runs on RHCOS. You can follow these procedures to troubleshoot problems related to the operating system.

7.3.5. Investigating kernel crashes

7.3.5.1. Enabling kdump

The **kdump** service, included in **kexec-tools**, provides a crash-dumping mechanism. You can use this service to save the contents of the system’s memory for later analysis.

The **kdump** service is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

RHCOS ships with **kexec-tools**, but manual configuration is required to enable **kdump**.

Procedure

Perform the following steps to enable **kdump** on RHCOS.

1. To reserve memory for the crash kernel during the first kernel booting, provide kernel arguments by entering the following command:

```
# rpm-ostree kargs --append='crashkernel=256M'
```

2. Optional: To write the crash dump over the network or to some other location, rather than to the default local **/var/crash** location, edit the **/etc/kdump.conf** configuration file.

   NOTE

   Network dumps are required when using LUKS. **kdump** does not support local crash dumps on LUKS-encrypted devices.

   For details on configuring the **kdump** service, see the comments in **/etc/sysconfig/kdump**, **/etc/kdump.conf**, and the **kdump.conf** manual page. Also refer to the RHEL **kdump** documentation for further information on configuring the dump target.

3. Enable the **kdump** systemd service.

```
# systemctl enable kdump.service
```

4. Reboot your system.

```
# systemctl reboot
```

5. Ensure that **kdump** has loaded a crash kernel by checking that the **kdump.service** has started and exited successfully and that **cat /sys/kernel/kexec_crash_loaded** prints **1**.

7.3.5.2. Enabling kdump on day-1

The **kdump** service is intended to be enabled per node to debug kernel problems. Because there are costs to having **kdump** enabled, and these costs accumulate with each additional **kdump**-enabled node, it is recommended that **kdump** only be enabled on each node as needed. Potential costs of enabling **kdump** on each node include:

- Less available RAM due to memory being reserved for the crash kernel.
- Node unavailability while the kernel is dumping the core.
- Additional storage space being used to store the crash dumps.
- Not being production-ready because the **kdump** service is in Technology Preview.

If you are aware of the downsides and trade-offs of having the **kdump** service enabled, it is possible to enable **kdump** in a cluster-wide fashion. Although machine-specific machine configs are not yet supported, you can perform the previous steps through a **systemd** unit in a **MachineConfig** object on day-1 and have kdump enabled on all nodes in the cluster. You can create a **MachineConfig** object and inject that object into the set of manifest files used by Ignition during cluster setup. See "Customizing nodes" in the *Installing → Installation configuration* section for more information and examples on how to use Ignition configs.

Procedure

Create a **MachineConfig** object for cluster-wide configuration:

1. Create a Butane config file, **99-worker-kdump.bu**, that configures and enables kdump:

```
variant: openshift
version: 4.9.0
metadata:
  name: 99-worker-kdump ①
  labels:
    machineconfiguration.openshift.io/role: worker ②
openshift:
  kernel_arguments: ③
    - crashkernel=256M
storage:
  files:
    - path: /etc/kdump.conf ④
      mode: 0644
      overwrite: true
      contents:
        inline: |
          path /var/crash
          core_collector makedumpfile -l --message-level 7 -d 31

    - path: /etc/sysconfig/kdump ⑤
      mode: 0644
      overwrite: true
      contents:
        inline: |
          KDUMP_COMMANDLINE_REMOVE="hugepages hugepagesz slub_debug quiet log_buf_len swiotlb"
          KDUMP_COMMANDLINE_APPEND="irqpoll nr_cpus=1 reset_devices cgroup_disable=memory mce=off numa=off udev.children-max=2 panic=10 rootflags=nofail acpi_no_memhotplug transparent_hugepage=never nokaslr novmcoredd hest_disable"
          KEXEC_ARGS="-s"
          KDUMP_IMG="vmlinuz"

systemd:
  units:
    - name: kdump.service
      enabled: true
```

① ② Replace **worker** with **master** in both locations when creating a **MachineConfig** object for control plane nodes.

③ Provide kernel arguments to reserve memory for the crash kernel. You can add other kernel arguments if necessary.

④ If you want to change the contents of **/etc/kdump.conf** from the default, include this section and modify the **inline** subsection accordingly.

⑤ If you want to change the contents of **/etc/sysconfig/kdump** from the default, include this section and modify the **inline** subsection accordingly.

2. Use Butane to generate a machine config YAML file, **99-worker-kdump.yaml**, containing the configuration to be delivered to the nodes:

   ```
   $ butane 99-worker-kdump.bu -o 99-worker-kdump.yaml
   ```

3. Put the YAML file into manifests during cluster setup. You can also create this **MachineConfig** object after cluster setup with the YAML file:

   ```
   $ oc create -f ./99-worker-kdump.yaml
   ```

7.3.5.3. Testing the kdump configuration

See the Testing the kdump configuration section in the RHEL documentation for **kdump**.

7.3.5.4. Analyzing a core dump

See the Analyzing a core dump section in the RHEL documentation for **kdump**.

Additional resources

- Setting up kdump in RHEL
- Linux kernel documentation for kdump
- kdump.conf(5) – a manual page for the **/etc/kdump.conf** configuration file containing the full documentation of available options
- kexec(8) – a manual page for **kexec**
- Red Hat Knowledgebase article regarding **kexec** and **kdump**.

7.3.5.4.1. Troubleshooting network issues

7.3.5.4.1.1. How the network interface is selected

For installations on bare metal or with virtual machines that have more than one network interface controller (NIC), the NIC that OpenShift Container Platform uses for communication with the Kubernetes API server is determined by the **nodeip-configuration.service** service unit that is run by systemd when the node boots. The service iterates through the network interfaces on the node and the first network interface that is configured with a subnet than can host the IP address for the API server is selected for OpenShift Container Platform communication.

After the **nodeip-configuration.service** service determines the correct NIC, the service creates the **/etc/systemd/system/kubelet.service.d/20-nodenet.conf** file. The **20-nodenet.conf** file sets the **KUBELET_NODE_IP** environment variable to the IP address that the service selected.

When the kubelet service starts, it reads the value of the environment variable from the **20-nodenet.conf** file and sets the IP address as the value to the **--node-ip** kubelet command-line argument. As a result, the kubelet service uses the selected IP address as the node IP address.

If hardware or networking is reconfigured after installation, it is possible that the **nodeip-configuration.service** service can select a different NIC after a reboot. In some cases, you might be able to detect that a different NIC is selected by reviewing the **INTERNAL-IP** column in the output from the **oc get nodes -o wide** command.

If network communication is disrupted or misconfigured because a different NIC is selected, one strategy for overriding the selection process is to set the correct IP address explicitly. The following list identifies the high-level steps and considerations:

- Create a shell script that determines the IP address to use for OpenShift Container Platform communication. Have the script create a custom unit file such as **/etc/systemd/system/kubelet.service.d/98-nodenet-override.conf**. Use the custom unit file, **98-nodenet-override.conf**, to set the **KUBELET_NODE_IP** environment variable to the IP address.
- Do not overwrite the **/etc/systemd/system/kubelet.service.d/20-nodenet.conf** file. Specify a file name with a numerically higher value such as **98-nodenet-override.conf** in the same directory path. The goal is to have the custom unit file run after **20-nodenet.conf** and override the value of the environment variable.
- Create a machine config object with the shell script as a base64-encoded string and use the Machine Config Operator to deploy the script to the nodes at a file system path such as **/usr/local/bin/override-node-ip.sh**.
- Ensure that **systemctl daemon-reload** runs after the shell script runs. The simplest method is to specify **ExecStart=systemctl daemon-reload** in the machine config, as shown in the following sample.

Sample machine config to override the network interface for kubelet

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-nodenet-override
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,<encoded_script>
        mode: 0755
        overwrite: true
        path: /usr/local/bin/override-node-ip.sh
    systemd:
      units:
      - contents: |
          [Unit]
          Description=Override node IP detection
          Wants=network-online.target
```

```
        Before=kubelet.service
        After=network-online.target
        [Service]
        Type=oneshot
        ExecStart=/usr/local/bin/override-node-ip.sh
        ExecStart=systemctl daemon-reload
        [Install]
        WantedBy=multi-user.target
      enabled: true
      name: nodenet-override.service
```

7.3.5.4.1.2. Troubleshooting Open vSwitch issues

To troubleshoot some Open vSwitch (OVS) issues, you might need to configure the log level to include more information.

If you modify the log level on a node temporarily, be aware that you can receive log messages from the machine config daemon on the node like the following example:

```
E0514 12:47:17.998892    2281 daemon.go:1350] content mismatch for file /etc/systemd/system/ovs-vswitchd.service: [Unit]
```

To avoid the log messages related to the mismatch, revert the log level change after you complete your troubleshooting.

7.3.5.4.1.2.1. Configuring the Open vSwitch log level temporarily

For short-term troubleshooting, you can configure the Open vSwitch (OVS) log level temporarily. The following procedure does not require rebooting the node. In addition, the configuration change does not persist whenever you reboot the node.

After you perform this procedure to change the log level, you can receive log messages from the machine config daemon that indicate a content mismatch for the **ovs-vswitchd.service**. To avoid the log messages, repeat this procedure and set the log level to the original value.

Prerequisites

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

Procedure

1. Start a debug pod for a node:

   ```
   $ oc debug node/<node_name>
   ```

2. Set **/host** as the root directory within the debug shell. The debug pod mounts the root file system from the host in **/host** within the pod. By changing the root directory to **/host**, you can run binaries from the host file system:

   ```
   # chroot /host
   ```

3. View the current syslog level for OVS modules:

```
# ovs-appctl vlog/list
```

The following example output shows the log level for syslog set to **info**.

Example output

```
                 console    syslog    file
                 -------    ------    ------
backtrace          OFF       INFO       INFO
bfd               OFF       INFO       INFO
bond               OFF       INFO       INFO
bridge             OFF       INFO       INFO
bundle             OFF       INFO       INFO
bundles            OFF       INFO       INFO
cfm               OFF       INFO       INFO
collectors         OFF       INFO       INFO
command_line       OFF       INFO       INFO
connmgr            OFF       INFO       INFO
conntrack          OFF       INFO       INFO
conntrack_tp       OFF       INFO       INFO
coverage           OFF       INFO       INFO
ct_dpif            OFF       INFO       INFO
daemon             OFF       INFO       INFO
daemon_unix        OFF       INFO       INFO
dns_resolve        OFF       INFO       INFO
dpdk               OFF       INFO       INFO
...
```

4. Specify the log level in the **/etc/systemd/system/ovs-vswitchd.service.d/10-ovs-vswitchd-restart.conf** file:

```
Restart=always
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :$${OVS_USER_ID##*:} /var/lib/openvswitch'
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :$${OVS_USER_ID##*:} /etc/openvswitch'
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :$${OVS_USER_ID##*:} /run/openvswitch'
ExecStartPost=-/usr/bin/ovs-appctl vlog/set syslog:dbg
ExecReload=-/usr/bin/ovs-appctl vlog/set syslog:dbg
```

In the preceding example, the log level is set to **dbg**. Change the last two lines by setting **syslog:<log_level>** to **off**, **emer**, **err**, **warn**, **info**, or **dbg**. The **off** log level filters out all log messages.

5. Restart the service:

```
# systemctl daemon-reload
```

```
# systemctl restart ovs-vswitchd
```

7.3.5.4.1.2.2. Configuring the Open vSwitch log level permanently

For long-term changes to the Open vSwitch (OVS) log level, you can change the log level permanently.

Prerequisites

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

Procedure

1. Create a file, such as **99-change-ovs-loglevel.yaml**, with a **MachineConfig** object like the following example:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master  <.>
  name: 99-change-ovs-loglevel
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
      - dropins:
        - contents: |
            [Service]
              ExecStartPost=-/usr/bin/ovs-appctl vlog/set syslog:dbg  <.>
              ExecReload=-/usr/bin/ovs-appctl vlog/set syslog:dbg
          name: 20-ovs-vswitchd-restart.conf
        name: ovs-vswitchd.service
```

<.> After you perform this procedure to configure control plane nodes, repeat the procedure and set the role to **worker** to configure worker nodes. <.> Set the **syslog:<log_level>** value. Log levels are **off**, **emer**, **err**, **warn**, **info**, or **dbg**. Setting the value to **off** filters out all log messages.

2. Apply the machine config:

```
$ oc apply -f 99-change-ovs-loglevel.yaml
```

Additional resources

- Understanding the Machine Config Operator
- Checking machine config pool status

7.3.5.4.1.2.3. Displaying Open vSwitch logs

Use the following procedure to display Open vSwitch (OVS) logs.

Prerequisites

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

Procedure

- Run one of the following commands:
    - Display the logs by using the **oc** command from outside the cluster:

      ```
      $ oc adm node-logs -u ovs-vswitchd
      ```

    - Display the logs after logging on to a node in the cluster:

      ```
      # journalctl -b -f -u ovs-vswitchd.service
      ```

      One way to log on to a node is by using the **oc debug node/<node_name>** command.

7.3.5.4.2. 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, such as skipping a software backup process to save time.

OpenShift Container Platform 4.9 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.

7.3.5.4.2.1. Operator subscription condition types

Subscriptions can report the following condition types:

Table 7.1. Subscription condition types

Condition	Description
CatalogSourcesUnhealthy	Some or all of the catalog sources to be used in resolution are unhealthy.
InstallPlanMissing	An install plan for a subscription is missing.
InstallPlanPending	An install plan for a subscription is pending installation.
InstallPlanFailed	An install plan for a subscription 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.

7.3.5.4.2.2. Viewing Operator subscription status by using the CLI

You can view Operator subscription status by 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.

7.3.5.4.2.3. Viewing Operator catalog source status by using the CLI

You can view the status of an Operator catalog source by 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 the catalog sources in a namespace. For example, you can check the **openshift-marketplace** namespace, which is used for cluster-wide catalog sources:

   ```
   $ oc get catalogsources -n openshift-marketplace
   ```

   #### Example output

   ```
   NAME                  DISPLAY               TYPE   PUBLISHER   AGE
   certified-operators   Certified Operators   grpc   Red Hat     55m
   community-operators   Community Operators   grpc   Red Hat     55m
   example-catalog       Example Catalog       grpc   Example Org 2m25s
   redhat-marketplace    Red Hat Marketplace   grpc   Red Hat     55m
   redhat-operators      Red Hat Operators     grpc   Red Hat     55m
   ```

2. Use the **oc describe** command to get more details and status about a catalog source:

   ```
   $ oc describe catalogsource example-catalog -n openshift-marketplace
   ```

   #### Example output

   ```
   Name:         example-catalog
   Namespace:    openshift-marketplace
   ...
   Status:
     Connection State:
       Address:              example-catalog.openshift-marketplace.svc:50051
       Last Connect:         2021-09-09T17:07:35Z
       Last Observed State:  TRANSIENT_FAILURE
     Registry Service:
       Created At:         2021-09-09T17:05:45Z
       Port:               50051
       Protocol:           grpc
       Service Name:       example-catalog
       Service Namespace:  openshift-marketplace
   ```

   In the preceding example output, the last observed state is **TRANSIENT_FAILURE**. This state indicates that there is a problem establishing a connection for the catalog source.

3. List the pods in the namespace where your catalog source was created:

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

   #### Example output

   ```
   NAME                                    READY   STATUS             RESTARTS   AGE
   certified-operators-cv9nn               1/1     Running            0          36m
   community-operators-6v8lp               1/1     Running            0          36m
   ```

```
marketplace-operator-86bfc75f9b-jkgbc   1/1     Running          0          42m
example-catalog-bwt8z                   0/1     ImagePullBackOff   0        3m55s
redhat-marketplace-57p8c                1/1     Running          0          36m
redhat-operators-smxx8                  1/1     Running          0          36m
```

When a catalog source is created in a namespace, a pod for the catalog source is created in that namespace. In the preceding example output, the status for the **example-catalog-bwt8z** pod is **ImagePullBackOff**. This status indicates that there is an issue pulling the catalog source's index image.

4. Use the **oc describe** command to inspect a pod for more detailed information:

```
$ oc describe pod example-catalog-bwt8z -n openshift-marketplace
```

Example output

```
Name:         example-catalog-bwt8z
Namespace:    openshift-marketplace
Priority:     0
Node:         ci-ln-jyryyg2-f76d1-ggdbq-worker-b-vsxjd/10.0.128.2
...
Events:
  Type     Reason          Age                From               Message
  ----     ------          ----               ----               -------
  Normal   Scheduled       48s                default-scheduler  Successfully assigned openshift-
marketplace/example-catalog-bwt8z to ci-ln-jyryyf2-f76d1-fgdbq-worker-b-vsxjd
  Normal   AddedInterface  47s                multus             Add eth0 [10.131.0.40/23] from
openshift-sdn
  Normal   BackOff         20s (x2 over 46s)  kubelet            Back-off pulling image
"quay.io/example-org/example-catalog:v1"
  Warning  Failed          20s (x2 over 46s)  kubelet            Error: ImagePullBackOff
  Normal   Pulling         8s (x3 over 47s)   kubelet            Pulling image "quay.io/example-
org/example-catalog:v1"
  Warning  Failed          8s (x3 over 47s)   kubelet            Failed to pull image
"quay.io/example-org/example-catalog:v1": rpc error: code = Unknown desc = reading
manifest v1 in quay.io/example-org/example-catalog: unauthorized: access to the requested
resource is not authorized
  Warning  Failed          8s (x3 over 47s)   kubelet            Error: ErrImagePull
```

In the preceding example output, the error messages indicate that the catalog source's index image is failing to pull successfully because of an authorization issue. For example, the index image might be stored in a registry that requires login credentials.

Additional resources

- Operator Lifecycle Manager concepts and resources → Catalog source
- gRPC documentation: States of Connectivity
- Accessing images for Operators from private registries

7.3.5.4.2.4. 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 4.9 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.

7.3.5.4.2.5. 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 control plane 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 control plane node by using SSH instead. Replace **<master-node>.<cluster_name>.<base_domain>** with appropriate values.

   a. List pods on each control plane 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 4.9 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>**.

7.3.5.4.2.6. Disabling the Machine Config Operator from automatically rebooting

When configuration changes are made by the Machine Config Operator (MCO), Red Hat Enterprise Linux CoreOS (RHCOS) must reboot for the changes to take effect. Whether the configuration change is automatic or manual, an RHCOS node reboots automatically unless it is paused.

NOTE

The following modifications do not trigger a node reboot:

- When the MCO detects any of the following changes, it applies the update without draining or rebooting the node:
  - Changes to the SSH key in the **spec.config.passwd.users.sshAuthorizedKeys** parameter of a machine config.
  - Changes to the global pull secret or pull secret in the **openshift-config** namespace.
  - Automatic rotation of the **/etc/kubernetes/kubelet-ca.crt** certificate authority (CA) by the Kubernetes API Server Operator.
- When the MCO detects changes to the **/etc/containers/registries.conf** file, such as adding or editing an **ImageContentSourcePolicy** object, it drains the corresponding nodes, applies the changes, and uncordons the nodes.

To avoid unwanted disruptions, you can modify the machine config pool (MCP) to prevent automatic rebooting after the Operator makes changes to the machine config.

NOTE

Pausing an MCP prevents the MCO from applying any configuration changes on the associated nodes. Pausing an MCP also prevents any automatically-rotated certificates from being pushed to the associated nodes, including the automatic rotation of the **kube-apiserver-to-kubelet-signer** CA certificate. If the MCP is paused when the **kube-apiserver-to-kubelet-signer** CA certificate expires, and the MCO attempts to renew the certificate automatically, the new certificate is created but not applied across the nodes in the paused MCP. This causes failure in multiple **oc** commands, including but not limited to **oc debug**, **oc logs**, **oc exec**, and **oc attach**. Pausing an MCP should be done with careful consideration about the **kube-apiserver-to-kubelet-signer** CA certificate expiration and for short periods of time only.

New CA certificates are generated at 292 days from the installation date and removed at 365 days from that date. To determine the next automatic CA certificate rotation, see the Understand CA cert auto renewal in Red Hat OpenShift 4 .

7.3.5.4.2.6.1. Disabling the Machine Config Operator from automatically rebooting by using the console

To avoid unwanted disruptions from changes made by the Machine Config Operator (MCO), you can use the OpenShift Container Platform web console to modify the machine config pool (MCP) to prevent the MCO from making any changes to nodes in that pool. This prevents any reboots that would normally be part of the MCO update process.

NOTE

See second **NOTE** in Disabling the Machine Config Operator from automatically rebooting.

Prerequisites

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

Procedure

To pause or unpause automatic MCO update rebooting:

- Pause the autoreboot process:
    1. Log in to the OpenShift Container Platform web console as a user with the **cluster-admin** role.
    2. Click **Compute** → **MachineConfigPools**.
    3. On the **MachineConfigPools** page, click either **master** or **worker**, depending upon which nodes you want to pause rebooting for.
    4. On the **master** or **worker** page, click **YAML**.
    5. In the YAML, update the **spec.paused** field to **true**.

        Sample MachineConfigPool object

        ```
        apiVersion: machineconfiguration.openshift.io/v1
        kind: MachineConfigPool
         ...
        spec:
         ...
          paused: true ①
        ```

        ① Update the **spec.paused** field to **true** to pause rebooting.

    6. To verify that the MCP is paused, return to the **MachineConfigPools** page.
        On the **MachineConfigPools** page, the **Paused** column reports **True** for the MCP you modified.

        If the MCP has pending changes while paused, the **Updated** column is **False** and **Updating** is **False**. When **Updated** is **True** and **Updating** is **False**, there are no pending changes.

        > **IMPORTANT**
        >
        > If there are pending changes (where both the **Updated** and **Updating** columns are **False**), it is recommended to schedule a maintenance window for a reboot as early as possible. Use the following steps for unpausing the autoreboot process to apply the changes that were queued since the last reboot.

- Unpause the autoreboot process:
    1. Log in to the OpenShift Container Platform web console as a user with the **cluster-admin** role.
    2. Click **Compute** → **MachineConfigPools**.
    3. On the **MachineConfigPools** page, click either **master** or **worker**, depending upon which nodes you want to pause rebooting for.

4. On the **master** or **worker** page, click **YAML**.

5. In the YAML, update the **spec.paused** field to **false**.

   Sample MachineConfigPool object

   ```
   apiVersion: machineconfiguration.openshift.io/v1
   kind: MachineConfigPool
    ...
   spec:
    ...
     paused: false ①
   ```

   ① Update the **spec.paused** field to **false** to allow rebooting.

   NOTE

   By unpausing an MCP, the MCO applies all paused changes reboots Red Hat Enterprise Linux CoreOS (RHCOS) as needed.

6. To verify that the MCP is paused, return to the **MachineConfigPools** page.
   On the **MachineConfigPools** page, the **Paused** column reports **False** for the MCP you modified.

   If the MCP is applying any pending changes, the **Updated** column is **False** and the **Updating** column is **True**. When **Updated** is **True** and **Updating** is **False**, there are no further changes being made.

7.3.5.4.2.6.2. Disabling the Machine Config Operator from automatically rebooting by using the CLI

To avoid unwanted disruptions from changes made by the Machine Config Operator (MCO), you can modify the machine config pool (MCP) using the OpenShift CLI (oc) to prevent the MCO from making any changes to nodes in that pool. This prevents any reboots that would normally be part of the MCO update process.

NOTE

See second **NOTE** in Disabling the Machine Config Operator from automatically rebooting.

Prerequisites

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

Procedure

To pause or unpause automatic MCO update rebooting:

- Pause the autoreboot process:
  1. Update the **MachineConfigPool** custom resource to set the **spec.paused** field to **true**.

Control plane (master) nodes

```
$ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/master
```

Worker nodes

```
$ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/worker
```

2. Verify that the MCP is paused:

Control plane (master) nodes

```
$ oc get machineconfigpool/master --template='{{.spec.paused}}'
```

Worker nodes

```
$ oc get machineconfigpool/worker --template='{{.spec.paused}}'
```

Example output

```
true
```

The **spec.paused** field is **true** and the MCP is paused.

3. Determine if the MCP has pending changes:

```
# oc get machineconfigpool
```

Example output

```
NAME     CONFIG                                             UPDATED   UPDATING
master   rendered-master-33cf0a1254318755d7b48002c597bf91   True      False
worker   rendered-worker-e405a5bdb0db1295acea08bcca33fa60   False     False
```

If the **UPDATED** column is **False** and **UPDATING** is **False**, there are pending changes. When **UPDATED** is **True** and **UPDATING** is **False**, there are no pending changes. In the previous example, the worker node has pending changes. The control plane node does not have any pending changes.

IMPORTANT

If there are pending changes (where both the **Updated** and **Updating** columns are **False**), it is recommended to schedule a maintenance window for a reboot as early as possible. Use the following steps for unpausing the autoreboot process to apply the changes that were queued since the last reboot.

- Unpause the autoreboot process:

  1. Update the **MachineConfigPool** custom resource to set the **spec.paused** field to **false**.

Control plane (master) nodes

```
$ oc patch --type=merge --patch='{"spec":{"paused":false}}' machineconfigpool/master
```

Worker nodes

```
$ oc patch --type=merge --patch='{"spec":{"paused":false}}' machineconfigpool/worker
```

> **NOTE**
>
> By unpausing an MCP, the MCO applies all paused changes and reboots Red Hat Enterprise Linux CoreOS (RHCOS) as needed.

2. Verify that the MCP is unpaused:

Control plane (master) nodes

```
$ oc get machineconfigpool/master --template='{{.spec.paused}}'
```

Worker nodes

```
$ oc get machineconfigpool/worker --template='{{.spec.paused}}'
```

Example output

```
false
```

The **spec.paused** field is **false** and the MCP is unpaused.

3. Determine if the MCP has pending changes:

```
$ oc get machineconfigpool
```

Example output

```
NAME     CONFIG                                   UPDATED  UPDATING
master   rendered-master-546383f80705bd5aeaba93   True     False
worker   rendered-worker-b4c51bb33ccaae6fc4a6a5   False    True
```

If the MCP is applying any pending changes, the **UPDATED** column is **False** and the **UPDATING** column is **True**. When **UPDATED** is **True** and **UPDATING** is **False**, there are no further changes being made. In the previous example, the MCO is updating the worker node.

7.3.5.4.2.7. Refreshing failing subscriptions

In Operator Lifecycle Manager (OLM), if you subscribe to an Operator that references images that are not accessible on your network, you can find jobs in the **openshift-marketplace** namespace that are failing with the following errors:

Example output

```
ImagePullBackOff for
Back-off pulling image "example.com/openshift4/ose-elasticsearch-operator-
bundle@sha256:6d2587129c846ec28d384540322b40b05833e7e00b25cca584e004af9a1d292e"
```

Example output

```
rpc error: code = Unknown desc = error pinging docker registry example.com: Get
"https://example.com/v2/": dial tcp: lookup example.com on 10.0.0.1:53: no such host
```

As a result, the subscription is stuck in this failing state and the Operator is unable to install or upgrade.

You can refresh a failing subscription by deleting the subscription, cluster service version (CSV), and other related objects. After recreating the subscription, OLM then reinstalls the correct version of the Operator.

Prerequisites

- You have a failing subscription that is unable to pull an inaccessible bundle image.
- You have confirmed that the correct bundle image is accessible.

Procedure

1. Get the names of the **Subscription** and **ClusterServiceVersion** objects from the namespace where the Operator is installed:

   ```
   $ oc get sub,csv -n <namespace>
   ```

   Example output

   ```
   NAME                                                  PACKAGE                  SOURCE             CHANNEL
   subscription.operators.coreos.com/elasticsearch-operator   elasticsearch-operator   redhat-
   operators   5.0

   NAME                                                                        DISPLAY                          VERSION
   REPLACES   PHASE
   clusterserviceversion.operators.coreos.com/elasticsearch-operator.5.0.0-65   OpenShift
   Elasticsearch Operator   5.0.0-65            Succeeded
   ```

2. Delete the subscription:

   ```
   $ oc delete subscription <subscription_name> -n <namespace>
   ```

3. Delete the cluster service version:

   ```
   $ oc delete csv <csv_name> -n <namespace>
   ```

4. Get the names of any failing jobs and related config maps in the **openshift-marketplace** namespace:

   ```
   $ oc get job,configmap -n openshift-marketplace
   ```

Example output

```
NAME                                                                                          COMPLETIONS   DURATION   AGE
job.batch/1de9443b6324e629ddf31fed0a853a121275806170e34c926d69e53a7fcbccb   1/1
26s        9m30s

NAME                                                                                          DATA   AGE
configmap/1de9443b6324e629ddf31fed0a853a121275806170e34c926d69e53a7fcbccb   3
9m30s
```

5. Delete the job:

```
$ oc delete job <job_name> -n openshift-marketplace
```

   This ensures pods that try to pull the inaccessible image are not recreated.

6. Delete the config map:

```
$ oc delete configmap <configmap_name> -n openshift-marketplace
```

7. Reinstall the Operator using OperatorHub in the web console.

Verification

- Check that the Operator has been reinstalled successfully:

```
$ oc get sub,csv,installplan -n <namespace>
```

7.3.5.4.3. 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 4.9.

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.

7.3.5.4.3.1. 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 7.2. 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.
ErrKillPodSandbox	A pod sandbox did not stop successfully.
ErrSetupNetwork	Network initialization failed.

Pod error state	Description
ErrTeardownNetwork	Network termination failed.

7.3.5.4.3.2. 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
```

7.3.5.4.3.3. 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>
      ```

7.3.5.4.3.4. 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.

7.3.5.4.3.5. 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.

7.3.5.4.3.6. 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 The first container in a pod is 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 The first container in a pod is selected if the **-c** option is not specified.

NOTE

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

7.3.5.4.4. Troubleshooting the Source-to-Image process

7.3.5.4.4.1. 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

7.3.5.4.4.2. 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>
     ```

7.3.5.4.4.3. 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 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 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 **DeploymentConfig** object. 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 **DeploymentConfig** object, 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 4.9 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.

7.3.5.4.4.4. Additional resources

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

7.3.5.4.5. Troubleshooting storage issues

7.3.5.4.5.1. 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
```

Procedure

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, force delete the pod on a shutdown or crashed node to avoid data loss in critical workloads, such as when dynamic persistent volumes are attached.

  ```
  $ oc delete pod <old_pod> --force=true --grace-period=0s
  ```

  This command deletes the volumes stuck on shutdown or crashed nodes after six minutes.

7.3.5.4.6. Troubleshooting Windows container workload issues

7.3.5.4.6.1. Windows Machine Config Operator does not install

If you have completed the process of installing the Windows Machine Config Operator (WMCO), but the Operator is stuck in the **InstallWaiting** phase, your issue is likely caused by a networking issue.

The WMCO requires your OpenShift Container Platform cluster to be configured with hybrid networking using OVN-Kubernetes; the WMCO cannot complete the installation process without hybrid networking available. This is necessary to manage nodes on multiple operating systems (OS) and OS variants. This must be completed during the installation of your cluster.

For more information, see Configuring hybrid networking .

7.3.5.4.6.2. Investigating why Windows Machine does not become compute node

There are various reasons why a Windows Machine does not become a compute node. The best way to investigate this problem is to collect the Windows Machine Config Operator (WMCO) logs.

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You have created a Windows machine set.

Procedure

- Run the following command to collect the WMCO logs:

```
$ oc logs -f deployment/windows-machine-config-operator -n openshift-windows-machine-config-operator
```

7.3.5.4.6.3. Accessing a Windows node

Windows nodes cannot be accessed using the **oc debug node** command; the command requires running a privileged pod on the node, which is not yet supported for Windows. Instead, a Windows node can be accessed using a secure shell (SSH) or Remote Desktop Protocol (RDP). An SSH bastion is required for both methods.

7.3.5.4.6.3.1. Accessing a Windows node using SSH

You can access a Windows node by using a secure shell (SSH).

Prerequisites

- You have installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You have created a Windows machine set.
- You have added the key used in the **cloud-private-key** secret and the key used when creating the cluster to the ssh-agent. For security reasons, remember to remove the keys from the ssh-agent after use.
- You have connected to the Windows node using an **ssh-bastion** pod.

Procedure

- Access the Windows node by running the following command:

```
$ ssh -t -o StrictHostKeyChecking=no -o ProxyCommand='ssh -A -o StrictHostKeyChecking=no \
    -o ServerAliveInterval=30 -W %h:%p core@$(oc get service --all-namespaces -l run=ssh-bastion \
    -o go-template="{{ with (index (index .items 0).status.loadBalancer.ingress 0) }}{{ or .hostname .ip }}{{end}}")' <username>@<windows_node_internal_ip> ① ②
```

① Specify the cloud provider username, such as **Administrator** for Amazon Web Services (AWS) or **capi** for Microsoft Azure.

② Specify the internal IP address of the node, which can be discovered by running the following command:

```
$ oc get nodes <node_name> -o jsonpath={.status.addresses[?\
(@.type==\"InternalIP\"\)].address}
```

7.3.5.4.6.3.2. Accessing a Windows node using RDP

You can access a Windows node by using a Remote Desktop Protocol (RDP).

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You have created a Windows machine set.
- You have added the key used in the **cloud-private-key** secret and the key used when creating the cluster to the ssh-agent. For security reasons, remember to remove the keys from the ssh-agent after use.
- You have connected to the Windows node using an **ssh-bastion** pod.

Procedure

1. Run the following command to set up an SSH tunnel:

```
$ ssh -L 2020:<windows_node_internal_ip>:3389 \ ①
    core@$(oc get service --all-namespaces -l run=ssh-bastion -o go-template="{{ with (index
(index .items 0).status.loadBalancer.ingress 0) }}{{ or .hostname .ip }}{{end}}")
```

① Specify the internal IP address of the node, which can be discovered by running the following command:

```
$ oc get nodes <node_name> -o jsonpath={.status.addresses[?\
(@.type==\"InternalIP\"\)].address}
```

2. From within the resulting shell, SSH into the Windows node and run the following command to create a password for the user:

```
C:\> net user <username> * ①
```

① Specify the cloud provider user name, such as **Administrator** for AWS or **capi** for Azure.

You can now remotely access the Windows node at **localhost:2020** using an RDP client.

7.3.5.4.6.4. Collecting Kubernetes node logs for Windows containers

Windows container logging works differently from Linux container logging; the Kubernetes node logs for Windows workloads are streamed to the **C:\var\logs** directory by default. Therefore, you must gather the Windows node logs from that directory.

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You have created a Windows machine set.

Procedure

1. To view the logs under all directories in **C:\var\logs**, run the following command:

```
$ oc adm node-logs -l kubernetes.io/os=windows --path= \
    /ip-10-0-138-252.us-east-2.compute.internal containers \
    /ip-10-0-138-252.us-east-2.compute.internal hybrid-overlay \
    /ip-10-0-138-252.us-east-2.compute.internal kube-proxy \
    /ip-10-0-138-252.us-east-2.compute.internal kubelet \
    /ip-10-0-138-252.us-east-2.compute.internal pods
```

2. You can now list files in the directories using the same command and view the individual log files. For example, to view the kubelet logs, run the following command:

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=/kubelet/kubelet.log
```

7.3.5.4.6.5. Collecting Windows application event logs

The **Get-WinEvent** shim on the kubelet **logs** endpoint can be used to collect application event logs from Windows machines.

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You have created a Windows machine set.

Procedure

- To view logs from all applications logging to the event logs on the Windows machine, run:

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=journal
```

The same command is executed when collecting logs with **oc adm must-gather**.

Other Windows application logs from the event log can also be collected by specifying the respective service with a **-u** flag. For example, you can run the following command to collect logs for the docker runtime service:

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=journal -u docker
```

7.3.5.4.6.6. Collecting Docker logs for Windows containers

The Windows Docker service does not stream its logs to stdout, but instead, logs to the event log for Windows. You can view the Docker event logs to investigate issues you think might be caused by the Windows Docker service.

Prerequisites

- You installed the Windows Machine Config Operator (WMCO) using Operator Lifecycle Manager (OLM).
- You have created a Windows machine set.

Procedure

1. SSH into the Windows node and enter PowerShell:

```
C:\> powershell
```

2. View the Docker logs by running the following command:

```
C:\> Get-EventLog -LogName Application -Source Docker
```

7.3.5.4.6.7. Additional resources

- Containers on Windows troubleshooting
- Troubleshoot host and container image mismatches
- Docker for Windows troubleshooting
- Common Kubernetes problems with Windows

7.3.5.4.7. Investigating monitoring issues

OpenShift Container Platform includes a pre-configured, pre-installed, and self-updating monitoring stack that provides monitoring for core platform components. In OpenShift Container Platform 4.9, cluster administrators can optionally enable monitoring for user-defined projects.

You can follow these procedures if your own metrics are unavailable or if Prometheus is consuming a lot of disk space.

7.3.5.4.7.1. Investigating why user-defined metrics are unavailable

ServiceMonitor resources enable you to determine how to use the metrics exposed by a service in user-defined projects. Follow the steps outlined in this procedure if you have created a **ServiceMonitor** resource but cannot see any corresponding metrics in the Metrics UI.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- You have enabled and configured monitoring for user-defined workloads.
- You have created the **user-workload-monitoring-config** **ConfigMap** object.
- You have created a **ServiceMonitor** resource.

Procedure

1. **Check that the corresponding labels match** in the service and **ServiceMonitor** resource configurations.

    a. Obtain the label defined in the service. The following example queries the **prometheus-example-app** service in the **ns1** project:

    ```
    $ oc -n ns1 get service prometheus-example-app -o yaml
    ```

    Example output

    ```
      labels:
        app: prometheus-example-app
    ```

    b. Check that the **matchLabels app** label in the **ServiceMonitor** resource configuration matches the label output in the preceding step:

    ```
    $ oc -n ns1 get servicemonitor prometheus-example-monitor -o yaml
    ```

    Example output

    ```
    spec:
      endpoints:
      - interval: 30s
        port: web
        scheme: http
      selector:
        matchLabels:
          app: prometheus-example-app
    ```

    NOTE

    You can check service and **ServiceMonitor** resource labels as a developer with view permissions for the project.

2. **Inspect the logs for the Prometheus Operator** in the **openshift-user-workload-monitoring** project.

    a. List the pods in the **openshift-user-workload-monitoring** project:

    ```
    $ oc -n openshift-user-workload-monitoring get pods
    ```

    Example output

    ```
    NAME                                   READY   STATUS    RESTARTS   AGE
    prometheus-operator-776fcbbd56-2nbfm   2/2     Running   0          132m
    prometheus-user-workload-0             5/5     Running   1          132m
    prometheus-user-workload-1             5/5     Running   1          132m
    thanos-ruler-user-workload-0           3/3     Running   0          132m
    thanos-ruler-user-workload-1           3/3     Running   0          132m
    ```

    b. Obtain the logs from the **prometheus-operator** container in the **prometheus-operator** pod. In the following example, the pod is called **prometheus-operator-776fcbbd56-2nbfm**:

```
$ oc -n openshift-user-workload-monitoring logs prometheus-operator-776fcbbd56-2nbfm -c prometheus-operator
```

If there is a issue with the service monitor, the logs might include an error similar to this example:

```
level=warn ts=2020-08-10T11:48:20.906739623Z caller=operator.go:1829
component=prometheusoperator msg="skipping servicemonitor" error="it accesses file
system via bearer token file which Prometheus specification prohibits"
servicemonitor=eagle/eagle namespace=openshift-user-workload-monitoring
prometheus=user-workload
```

3. **Review the target status for your project** in the Prometheus UI directly.

   a. Establish port-forwarding to the Prometheus instance in the **openshift-user-workload-monitoring** project:

   ```
   $ oc port-forward -n openshift-user-workload-monitoring pod/prometheus-user-workload-0 9090
   ```

   b. Open http://localhost:9090/targets in a web browser and review the status of the target for your project directly in the Prometheus UI. Check for error messages relating to the target.

4. **Configure debug level logging for the Prometheus Operator** in the **openshift-user-workload-monitoring** project.

   a. Edit the **user-workload-monitoring-config ConfigMap** object in the **openshift-user-workload-monitoring** project:

   ```
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   ```

   b. Add **logLevel: debug** for **prometheusOperator** under **data/config.yaml** to set the log level to **debug**:

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
       prometheusOperator:
         logLevel: debug
   ```

   c. Save the file to apply the changes.

   NOTE

   The **prometheus-operator** in the **openshift-user-workload-monitoring** project restarts automatically when you apply the log-level change.

d. Confirm that the **debug** log-level has been applied to the **prometheus-operator** deployment in the **openshift-user-workload-monitoring** project:

```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |
grep "log-level"
```

Example output

```
        - --log-level=debug
```

Debug level logging will show all calls made by the Prometheus Operator.

e. Check that the **prometheus-operator** pod is running:

```
$ oc -n openshift-user-workload-monitoring get pods
```

> NOTE
>
> If an unrecognized Prometheus Operator **loglevel** value is included in the config map, the **prometheus-operator** pod might not restart successfully.

f. Review the debug logs to see if the Prometheus Operator is using the **ServiceMonitor** resource. Review the logs for other related errors.

Additional resources

- Creating a user-defined workload monitoring config map
- See Specifying how a service is monitored for details on how to create a service monitor or pod monitor

7.3.5.4.7.2. Determining why Prometheus is consuming a lot of disk space

Developers can create labels to define attributes for metrics in the form of key-value pairs. The number of potential key-value pairs corresponds to the number of possible values for an attribute. An attribute that has an unlimited number of potential values is called an unbound attribute. For example, a **customer_id** attribute is unbound because it has an infinite number of possible values.

Every assigned key-value pair has a unique time series. The use of many unbound attributes in labels can result in an exponential increase in the number of time series created. This can impact Prometheus performance and can consume a lot of disk space.

You can use the following measures when Prometheus consumes a lot of disk:

- **Check the number of scrape samples** that are being collected.
- **Check the time series database (TSDB) status in the Prometheus UI** for more information on which labels are creating the most time series. This requires cluster administrator privileges.
- **Reduce the number of unique time series that are created** by reducing the number of unbound attributes that are assigned to user-defined metrics.

> **NOTE**
>
> Using attributes that are bound to a limited set of possible values reduces the number of potential key-value pair combinations.

- **Enforce limits on the number of samples that can be scraped** across user-defined projects. This requires cluster administrator privileges.

Prerequisites

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

Procedure

1. In the **Administrator** perspective, navigate to **Observe** → **Metrics**.
2. Run the following Prometheus Query Language (PromQL) query in the **Expression** field. This returns the ten metrics that have the highest number of scrape samples:

   ```
   topk(10,count by (job)({__name__=~".+"}))
   ```

3. Investigate the number of unbound label values assigned to metrics with higher than expected scrape sample counts.
   - **If the metrics relate to a user-defined project**, review the metrics key-value pairs assigned to your workload. These are implemented through Prometheus client libraries at the application level. Try to limit the number of unbound attributes referenced in your labels.
   - **If the metrics relate to a core OpenShift Container Platform project**, create a Red Hat support case on the Red Hat Customer Portal.
4. Check the TSDB status in the Prometheus UI.
   a. In the **Administrator** perspective, navigate to **Networking** → **Routes**.
   b. Select the **openshift-monitoring** project in the **Project** list.
   c. Select the URL in the **prometheus-k8s** row to open the login page for the Prometheus UI.
   d. Choose **Log in with OpenShift** to log in using your OpenShift Container Platform credentials.
   e. In the Prometheus UI, navigate to **Status** → **TSDB Status**.

Additional resources

- See Setting a scrape sample limit for user-defined projects for details on how to set a scrape sample limit and create related alerting rules

7.3.5.4.8. Diagnosing OpenShift CLI (oc) issues

7.3.5.4.8.1. 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 7.3. 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.

7.3.5.4.8.2. Specifying OpenShift CLI (oc) log levels

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

Prerequisites

- Install 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
```