OpenShift Container Platform 4.16

Support

Getting support for OpenShift Container Platform

Last Updated: 2026-01-15

OpenShift Container Platform 4.16 Support

Getting support for OpenShift Container Platform

Legal Notice

Copyright © Red Hat.

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, 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 Software Collections 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.
- **Insights Operator**: By default, OpenShift Container Platform installs and enables the Insights Operator, which reports configuration and component failure status every two hours. The Insights 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 with your cluster . 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 in the "Support" and "Troubleshooting logging" sections to resolve logging issues:
  - Viewing the status of the Red Hat OpenShift Logging Operator
  - Viewing the status of logging components
  - Troubleshooting logging alerts
  - Collecting information about your logging environment by using the **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 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 Jira issue for the most relevant 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. Click **Search**.
3. In the 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

4. Click the **Enter** key.
5. Optional: Select the **OpenShift Container Platform** product filter.
6. Optional: Select the **Documentation** 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 **Customer Support** page of the Red Hat Customer Portal.
2. Click **Get support**.
3. On the **Cases** tab of the **Customer Support** page:
   a. Optional: Change the pre-filled account and owner details if needed.
   b. Select the appropriate category for your issue, such as **Bug or Defect**, and click **Continue**.
4. Enter the following information:
   a. In the **Summary** field, enter a concise but descriptive problem summary and further details about the symptoms being experienced, as well as your expectations.
   b. Select **OpenShift Container Platform** from the **Product** drop-down menu.
   c. Select **4.16** from the **Version** drop-down.
5. 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**.
6. 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**.
7. Ensure that the account information presented is as expected, and if not, amend accordingly.
8. 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** → **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"}'
```

9. Complete the following questions where prompted and then click **Continue**:
   - What are you experiencing? What are you expecting to happen?
   - Define the value or impact to you or the business.
   - Where are you experiencing this behavior? What environment?
   - When does this behavior occur? Frequency? Repeatedly? At certain times?
10. 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.
11. Input relevant case management details and click **Continue**.
12. 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 OpenShift Cluster Manager.

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 Jira issue.

- **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 OpenShift Cluster Manager 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:

4.1.1.1.1. System information

- 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 unique random identifier that is generated during an installation
- 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
- The OpenShift Container Platform framework components installed in a cluster and their condition and status
- Events for all namespaces listed as "related objects" for a degraded Operator
- Information about degraded software
- Information about the validity of certificates
- The name of the provider platform that OpenShift Container Platform is deployed on and the data center location

4.1.1.1.2. Sizing Information

- 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

- Number of application builds by build strategy type

4.1.1.1.3. Usage information

- Usage information about components, features, and extensions
- Usage details about Technology Previews and unsupported configurations

Telemetry does not collect identifying information such as usernames 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.
- 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 the Insights Advisor service on Red Hat Hybrid Cloud Console. 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:

- Identify potential cluster issues and provide a solution and preventive actions in the Insights Advisor service on Red Hat Hybrid Cloud Console
- Improve OpenShift Container Platform by providing aggregated and critical information to product and support teams
- Make OpenShift Container Platform more intuitive

Additional resources

- 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 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

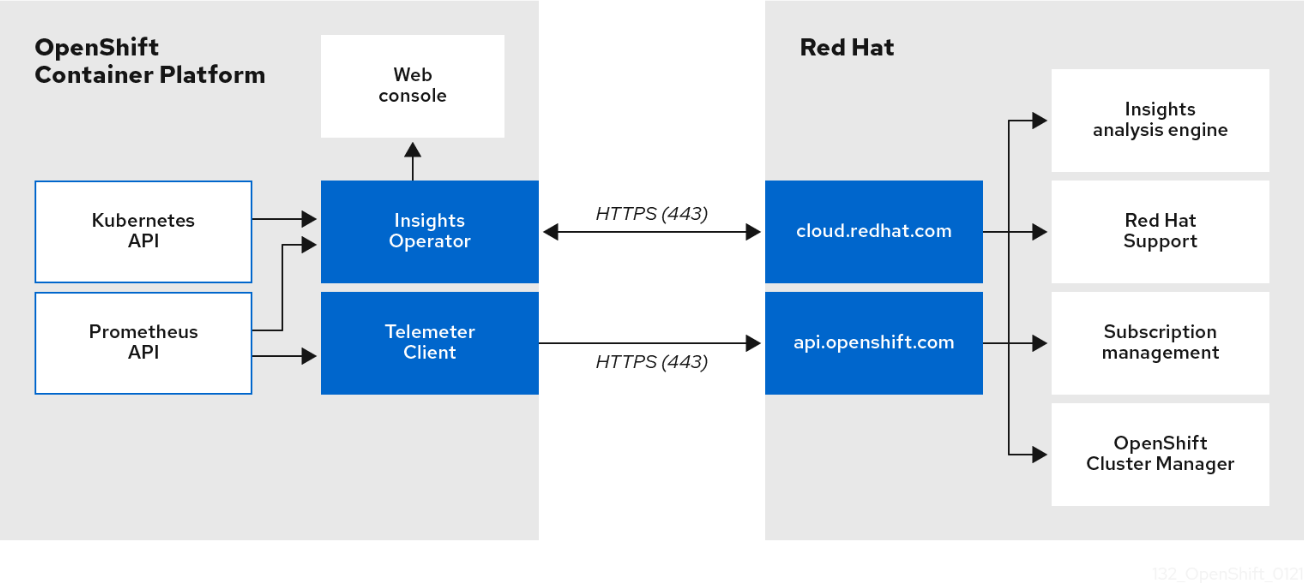
The Telemeter Client collects selected time series data from the Prometheus API. The time series data is uploaded to api.openshift.com every four minutes and thirty seconds for processing.

The Insights Operator gathers selected data from the Kubernetes API and the Prometheus API into an archive. The archive is uploaded to OpenShift Cluster Manager every two hours for processing. The Insights Operator also downloads the latest Insights analysis from OpenShift Cluster Manager. 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 About OpenShift Container Platform monitoring 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 parties

Red Hat may engage certain third parties to assist in the collection, analysis, and storage of the Telemetry and configuration data.

User control / enabling and disabling telemetry and configuration data collection

- 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 view the cluster and components time series data captured by Telemetry.

Prerequisites

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

Procedure

1. Log in to a cluster.
2. Run the following command, which queries a cluster's Prometheus service and returns the full set of time series data captured by Telemetry:

NOTE

The following example contains some values that are specific to OpenShift Container Platform on AWS.

```
$ curl -G -k -H "Authorization: Bearer $(oc whoami -t)" \
https://$(oc get route prometheus-k8s-federate -n \
openshift-monitoring -o jsonpath="{.spec.host}")/federate \
--data-urlencode 'match[]={__name__=~"cluster:usage:.*"}' \
--data-urlencode 'match[]={__name__="count:up0"}' \
--data-urlencode 'match[]={__name__="count:up1"}' \
--data-urlencode 'match[]={__name__="cluster_version"}' \
--data-urlencode 'match[]={__name__="cluster_version_available_updates"}' \
--data-urlencode 'match[]={__name__="cluster_version_capability"}' \
--data-urlencode 'match[]={__name__="cluster_operator_up"}' \
--data-urlencode 'match[]={__name__="cluster_operator_conditions"}' \
--data-urlencode 'match[]={__name__="cluster_version_payload"}' \
--data-urlencode 'match[]={__name__="cluster_installer"}' \
--data-urlencode 'match[]={__name__="cluster_infrastructure_provider"}' \
--data-urlencode 'match[]={__name__="cluster_feature_set"}' \
--data-urlencode 'match[]={__name__="instance:etcd_object_counts:sum"}' \
--data-urlencode 'match[]={__name__="ALERTS",alertstate="firing"}' \
--data-urlencode 'match[]={__name__="code:apiserver_request_total:rate:sum"}' \
--data-urlencode 'match[]={__name__="cluster:capacity_cpu_cores:sum"}' \
--data-urlencode 'match[]={__name__="cluster:capacity_memory_bytes:sum"}' \
--data-urlencode 'match[]={__name__="cluster:cpu_usage_cores:sum"}' \
--data-urlencode 'match[]={__name__="cluster:memory_usage_bytes:sum"}' \
--data-urlencode 'match[]={__name__="openshift:cpu_usage_cores:sum"}' \
--data-urlencode 'match[]={__name__="openshift:memory_usage_bytes:sum"}' \
--data-urlencode 'match[]={__name__="workload:cpu_usage_cores:sum"}' \
```

```
--data-urlencode 'match[]={__name__="workload:memory_usage_bytes:sum"}' \
--data-urlencode 'match[]={__name__="cluster:virt_platform_nodes:sum"}' \
--data-urlencode 'match[]={__name__="cluster:node_instance_type_count:sum"}' \
--data-urlencode 'match[]={__name__="cnv:vmi_status_running:count"}' \
--data-urlencode 'match[]={__name__="cluster:vmi_request_cpu_cores:sum"}' \
--data-urlencode 'match[]={__name__="node_role_os_version_machine:cpu_capacity_cores:sum"}' \
--data-urlencode 'match[]=
{__name__="node_role_os_version_machine:cpu_capacity_sockets:sum"}' \
--data-urlencode 'match[]={__name__="subscription_sync_total"}' \
--data-urlencode 'match[]={__name__="olm_resolution_duration_seconds"}' \
--data-urlencode 'match[]={__name__="csv_succeeded"}' \
--data-urlencode 'match[]={__name__="csv_abnormal"}' \
--data-urlencode 'match[]=
{__name__="cluster:kube_persistentvolumeclaim_resource_requests_storage_bytes:provisioner:sum"}'
\
--data-urlencode 'match[]={__name__="cluster:kubelet_volume_stats_used_bytes:provisioner:sum"}'
\
--data-urlencode 'match[]={__name__="ceph_cluster_total_bytes"}' \
--data-urlencode 'match[]={__name__="ceph_cluster_total_used_raw_bytes"}' \
--data-urlencode 'match[]={__name__="ceph_health_status"}' \
--data-urlencode 'match[]={__name__="odf_system_raw_capacity_total_bytes"}' \
--data-urlencode 'match[]={__name__="odf_system_raw_capacity_used_bytes"}' \
--data-urlencode 'match[]={__name__="odf_system_health_status"}' \
--data-urlencode 'match[]={__name__="job:ceph_osd_metadata:count"}' \
--data-urlencode 'match[]={__name__="job:kube_pv:count"}' \
--data-urlencode 'match[]={__name__="job:odf_system_pvs:count"}' \
--data-urlencode 'match[]={__name__="job:ceph_pools_iops:total"}' \
--data-urlencode 'match[]={__name__="job:ceph_pools_iops_bytes:total"}' \
--data-urlencode 'match[]={__name__="job:ceph_versions_running:count"}' \
--data-urlencode 'match[]={__name__="job:noobaa_total_unhealthy_buckets:sum"}' \
--data-urlencode 'match[]={__name__="job:noobaa_bucket_count:sum"}' \
--data-urlencode 'match[]={__name__="job:noobaa_total_object_count:sum"}' \
--data-urlencode 'match[]={__name__="odf_system_bucket_count", system_type="OCS",
system_vendor="Red Hat"}' \
--data-urlencode 'match[]={__name__="odf_system_objects_total", system_type="OCS",
system_vendor="Red Hat"}' \
--data-urlencode 'match[]={__name__="noobaa_accounts_num"}' \
--data-urlencode 'match[]={__name__="noobaa_total_usage"}' \
--data-urlencode 'match[]={__name__="console_url"}' \
--data-urlencode 'match[]={__name__="cluster:ovnkube_master_egress_routing_via_host:max"}' \
--data-urlencode 'match[]={__name__="cluster:network_attachment_definition_instances:max"}' \
--data-urlencode 'match[]=
{__name__="cluster:network_attachment_definition_enabled_instance_up:max"}' \
--data-urlencode 'match[]={__name__="cluster:ingress_controller_aws_nlb_active:sum"}' \
--data-urlencode 'match[]={__name__="cluster:route_metrics_controller_routes_per_shard:min"}' \
--data-urlencode 'match[]={__name__="cluster:route_metrics_controller_routes_per_shard:max"}' \
--data-urlencode 'match[]={__name__="cluster:route_metrics_controller_routes_per_shard:avg"}' \
--data-urlencode 'match[]={__name__="cluster:route_metrics_controller_routes_per_shard:median"}'
\
--data-urlencode 'match[]={__name__="cluster:openshift_route_info:tls_termination:sum"}' \
--data-urlencode 'match[]={__name__="insightsclient_request_send_total"}' \
--data-urlencode 'match[]={__name__="cam_app_workload_migrations"}' \
--data-urlencode 'match[]=
{__name__="cluster:apiserver_current_inflight_requests:sum:max_over_time:2m"}' \
--data-urlencode 'match[]={__name__="cluster:alertmanager_integrations:max"}' \
--data-urlencode 'match[]={__name__="cluster:telemetry_selected_series:count"}' \
```

```
--data-urlencode 'match[]={__name__="openshift:prometheus_tsdb_head_series:sum"}' \
--data-urlencode 'match[]=
{__name__="openshift:prometheus_tsdb_head_samples_appended_total:sum"}' \
--data-urlencode 'match[]={__name__="monitoring:container_memory_working_set_bytes:sum"}' \
--data-urlencode 'match[]={__name__="namespace_job:scrape_series_added:topk3_sum1h"}' \
--data-urlencode 'match[]=
{__name__="namespace_job:scrape_samples_post_metric_relabeling:topk3"}' \
--data-urlencode 'match[]={__name__="monitoring:haproxy_server_http_responses_total:sum"}' \
--data-urlencode 'match[]={__name__="rhmi_status"}' \
--data-urlencode 'match[]={__name__="status:upgrading:version:rhoam_state:max"}' \
--data-urlencode 'match[]={__name__="state:rhoam_critical_alerts:max"}' \
--data-urlencode 'match[]={__name__="state:rhoam_warning_alerts:max"}' \
--data-urlencode 'match[]={__name__="rhoam_7d_slo_percentile:max"}' \
--data-urlencode 'match[]={__name__="rhoam_7d_slo_remaining_error_budget:max"}' \
--data-urlencode 'match[]={__name__="cluster_legacy_scheduler_policy"}' \
--data-urlencode 'match[]={__name__="cluster_master_schedulable"}' \
--data-urlencode 'match[]={__name__="che_workspace_status"}' \
--data-urlencode 'match[]={__name__="che_workspace_started_total"}' \
--data-urlencode 'match[]={__name__="che_workspace_failure_total"}' \
--data-urlencode 'match[]={__name__="che_workspace_start_time_seconds_sum"}' \
--data-urlencode 'match[]={__name__="che_workspace_start_time_seconds_count"}' \
--data-urlencode 'match[]={__name__="cco_credentials_mode"}' \
--data-urlencode 'match[]={__name__="cluster:kube_persistentvolume_plugin_type_counts:sum"}' \
--data-urlencode 'match[]={__name__="visual_web_terminal_sessions_total"}' \
--data-urlencode 'match[]={__name__="acm_managed_cluster_info"}' \
--data-urlencode 'match[]={__name__="cluster:vsphere_vcenter_info:sum"}' \
--data-urlencode 'match[]={__name__="cluster:vsphere_esxi_version_total:sum"}' \
--data-urlencode 'match[]={__name__="cluster:vsphere_node_hw_version_total:sum"}' \
--data-urlencode 'match[]={__name__="openshift:build_by_strategy:sum"}' \
--data-urlencode 'match[]={__name__="rhods_aggregate_availability"}' \
--data-urlencode 'match[]={__name__="rhods_total_users"}' \
--data-urlencode 'match[]=
{__name__="instance:etcd_disk_wal_fsync_duration_seconds:histogram_quantile",quantile="0.99"}' \
--data-urlencode 'match[]={__name__="instance:etcd_mvcc_db_total_size_in_bytes:sum"}' \
--data-urlencode 'match[]=
{__name__="instance:etcd_network_peer_round_trip_time_seconds:histogram_quantile",quantile="0.99
"}' \
--data-urlencode 'match[]={__name__="instance:etcd_mvcc_db_total_size_in_use_in_bytes:sum"}' \
--data-urlencode 'match[]=
{__name__="instance:etcd_disk_backend_commit_duration_seconds:histogram_quantile",quantile="0.9
9"}' \
--data-urlencode 'match[]={__name__="appsvcs:cores_by_product:sum"}' \
--data-urlencode 'match[]={__name__="nto_custom_profiles:count"}' \
--data-urlencode 'match[]={__name__="openshift_csi_share_configmap"}' \
--data-urlencode 'match[]={__name__="openshift_csi_share_secret"}' \
--data-urlencode 'match[]={__name__="openshift_csi_share_mount_failures_total"}' \
--data-urlencode 'match[]={__name__="openshift_csi_share_mount_requests_total"}' \
--data-urlencode 'match[]={__name__="cluster:velero_backup_total:max"}' \
--data-urlencode 'match[]={__name__="cluster:velero_restore_total:max"}' \
--data-urlencode 'match[]={__name__="eo_es_storage_info"}' \
--data-urlencode 'match[]={__name__="eo_es_redundancy_policy_info"}' \
--data-urlencode 'match[]={__name__="eo_es_defined_delete_namespaces_total"}' \
--data-urlencode 'match[]={__name__="eo_es_misconfigured_memory_resources_info"}' \
--data-urlencode 'match[]={__name__="cluster:eo_es_data_nodes_total:max"}' \
--data-urlencode 'match[]={__name__="cluster:eo_es_documents_created_total:sum"}' \
--data-urlencode 'match[]={__name__="cluster:eo_es_documents_deleted_total:sum"}' \
```

```
--data-urlencode 'match[]={__name__="pod:eo_es_shards_total:max"}' \
--data-urlencode 'match[]={__name__="eo_es_cluster_management_state_info"}' \
--data-urlencode 'match[]={__name__="imageregistry:imagestreamtags_count:sum"}' \
--data-urlencode 'match[]={__name__="imageregistry:operations_count:sum"}' \
--data-urlencode 'match[]={__name__="log_logging_info"}' \
--data-urlencode 'match[]={__name__="log_collector_error_count_total"}' \
--data-urlencode 'match[]={__name__="log_forwarder_pipeline_info"}' \
--data-urlencode 'match[]={__name__="log_forwarder_input_info"}' \
--data-urlencode 'match[]={__name__="log_forwarder_output_info"}' \
--data-urlencode 'match[]={__name__="cluster:log_collected_bytes_total:sum"}' \
--data-urlencode 'match[]={__name__="cluster:log_logged_bytes_total:sum"}' \
--data-urlencode 'match[]={__name__="cluster:kata_monitor_running_shim_count:sum"}' \
--data-urlencode 'match[]={__name__="platform:hypershift_hostedclusters:max"}' \
--data-urlencode 'match[]={__name__="platform:hypershift_nodepools:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_unhealthy_bucket_claims:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_buckets_claims:max"}' \
--data-urlencode 'match[]=
{__name__="namespace:noobaa_unhealthy_namespace_resources:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_namespace_resources:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_unhealthy_namespace_buckets:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_namespace_buckets:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_accounts:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_usage:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_system_health_status:max"}' \
--data-urlencode 'match[]={__name__="ocs_advanced_feature_usage"}' \
--data-urlencode 'match[]={__name__="os_image_url_override:sum"}' \
--data-urlencode 'match[]={__name__="openshift:openshift_network_operator_ipsec_state:info"}'
```

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. REMOTE HEALTH REPORTING

You can *opt in*, enable, or *opt out*, disable, reporting health and usage data for your cluster.

4.3.1. Enabling remote health reporting

If you or your organization have disabled remote health reporting, you can enable this feature again. You can see that remote health reporting is disabled from the message **Insights not available** in the **Status** tile on the OpenShift Container Platform web console **Overview** page.

To enable remote health reporting, you must change the global cluster pull secret with a new authorization token. Enabling remote health reporting enables both Insights Operator and Telemetry.

4.3.2. Changing your global cluster pull secret to enable remote health reporting

You can change your existing global cluster pull secret to enable remote health reporting. If you have disabled remote health monitoring, you must download a new pull secret with your **console.openshift.com** access token from Red Hat OpenShift Cluster Manager.

Prerequisites

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

Procedure

1. Go to the Downloads page on the Red Hat Hybrid Cloud Console.
2. From **Tokens** → **Pull secret**, click the **Download** button.
   The **pull-secret** file contains your **cloud.openshift.com** access token in JSON format:

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

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

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

4. Make a backup copy of your pull secret.

```
$ cp pull-secret pull-secret-backup
```

5. Open the **pull-secret** file in a text editor.
6. Append the **cloud.openshift.com** JSON entry from the **pull-secret** file that you downloaded earlier into the **auths** file.
7. Save the file.
8. Update the secret in your cluster by running the following command:

```
oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=pull-secret
```

You might need to wait several minutes for the secret to update and your cluster to begin reporting.

Verification

1. For a verification check from the OpenShift Container Platform web console, complete the following steps:
   a. Go to the **Overview** page on the OpenShift Container Platform web console.
   b. View the **Insights** section in the **Status** tile that reports the number of issues found.
2. For a verification check from the OpenShift CLI (**oc**), enter the following command and then check that the value of the **status** parameter states **false**:

```
$ oc get co insights -o jsonpath='{.status.conditions[?(@.type=="Disabled")]}'
```

4.3.3. Consequences of disabling remote health reporting

In OpenShift Container Platform, customers can disable reporting usage information.

Before you disable remote health reporting, read the following benefits of a connected cluster:

- Red Hat can react more quickly to problems and better support our customers.
- Red Hat can better understand how product upgrades impact clusters.
- Connected clusters help to simplify the subscription and entitlement process.
- Connected clusters enable the OpenShift Cluster Manager service to offer an overview of your clusters and their subscription status.

NOTE

Consider leaving health and usage reporting enabled for pre-production, test, and production clusters. This means that Red Hat can participate in qualifying OpenShift Container Platform in your environments and react more rapidly to product issues.

The following lists some consequences of disabling remote health reporting on a connected cluster:

- Red Hat cannot view the success of product upgrades or the health of your clusters without an open support case.
- Red Hat cannot use configuration data to better triage customer support cases and identify which configurations our customers find important.
- The OpenShift Cluster Manager cannot show data about your clusters, which includes health and usage information.
- You must manually enter your subscription information in the **console.redhat.com** web console without the benefit of automatic usage reporting.

In restricted networks, Telemetry and Insights data still gets gathered through the appropriate configuration of your proxy.

4.3.4. Disabling remote health reporting

You can change your existing global cluster pull secret to disable remote health reporting. This configuration 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 you downloaded by removing the **cloud.openshift.com** JSON entry:

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

3. Save the file.

4. Update the secret in your cluster. For more information, see "Updating the global cluster pull secret".
You might need to wait several minutes for the secret to update in your cluster.

4.3.5. Registering your disconnected cluster

Register your disconnected OpenShift Container Platform cluster on the Red Hat Hybrid Cloud Console so that your cluster does not get impacted by disabling remote health reporting. For more information, see "Consequences of disabling remote health reporting".

IMPORTANT

By registering your disconnected cluster, you can continue to report your subscription usage to Red Hat. Red Hat can then return accurate usage and capacity trends associated with your subscription, so that you can use the returned information to better organize subscription allocations across all of your resources.

Prerequisites

- You logged in to the OpenShift Container Platform web console as the **cluster-admin** role.
- You can log in to the Red Hat Hybrid Cloud Console.

Procedure

1. Go to the Register disconnected cluster web page on the Red Hat Hybrid Cloud Console.

2. Optional: To access the **Register disconnected cluster** web page from the home page of the Red Hat Hybrid Cloud Console, go to the **Cluster List** navigation menu item and then select the **Register cluster** button.

3. Enter your cluster's details in the provided fields on the **Register disconnected cluster** page.

4. From the **Subscription settings** section of the page, select the subscription settings that apply to your Red Hat subscription offering.

5. To register your disconnected cluster, select the **Register cluster** button.

- How does the subscriptions service show my subscription data? (Getting Started with the Subscription Service)

4.3.6. Updating the global cluster pull secret

To add new registries or change authentication for your OpenShift Container Platform cluster, you can update the global pull secret by replacing it or appending new credentials. Use the **oc set data secret/pull-secret** command to apply the updated pull secret to all nodes in your cluster.

Use this procedure when you need a separate registry to store images than the registry used during installation.

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:

    a. Download the pull secret by entering the following command:

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

    where:

    <pull_secret_location>

    Specifies the path to the pull secret file.

    b. Add the new pull secret by entering the following command:

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

    where:

    <registry>

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

    <username>:<password>

    Specifies the credentials of the new registry.

<pull_secret_location>

Specifies the path to the pull secret file.

2. Update the global pull secret for your cluster by entering the following command. Note that this update rolls out to all nodes, which can take some time depending on the size of your cluster.

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

where:

<pull_secret_location>

Specifies the path to the new pull secret file.

Additional resources

- Transferring cluster ownership

4.4. USING INSIGHTS TO IDENTIFY ISSUES WITH YOUR CLUSTER

Insights repeatedly analyzes the data Insights Operator sends, which includes workload recommendations from Deployment Validation Operator (DVO). Users of OpenShift Container Platform can display the results in the Insights Advisor service on Red Hat Hybrid Cloud Console.

4.4.1. About Red Hat Insights Advisor for OpenShift Container Platform

You can use the Insights advisor service to assess and monitor the health of your OpenShift Container Platform clusters. Whether you are concerned about individual clusters, or with your whole infrastructure, it is important to be aware of the exposure of your cluster infrastructure to issues that can affect service availability, fault tolerance, performance, or security.

If the cluster has the Deployment Validation Operator (DVO) installed the recommendations also highlight workloads whose configuration might lead to cluster health issues.

The results of the Insights analysis are available in the Insights advisor service on Red Hat Hybrid Cloud Console. In the Red Hat Hybrid Cloud Console, you can perform the following actions:

- View clusters and workloads affected by specific recommendations.
- Use robust filtering capabilities to refine your results to those recommendations.
- Learn more about individual recommendations, details about the risks they present, and get resolutions tailored to your individual clusters.
- Share results with other stakeholders.

Additional resources

- Using the Deployment Validation Operator in your Red Hat Insights for OpenShift workflow

4.4.2. Understanding Insights Advisor recommendations

The Insights advisor service bundles information about various cluster states and component configurations that can negatively affect the service availability, fault tolerance, performance, or security of your clusters and workloads. This information set is called a recommendation in the Insights advisor service. Recommendations for clusters includes the following information:

- **Name:** A concise description of the recommendation
- **Added:** When the recommendation was published to the Insights Advisor service archive
- **Category:** Whether the issue has the potential to negatively affect service availability, fault tolerance, performance, or security
- **Total risk:** A value derived from the *likelihood* that the condition will negatively affect your cluster or workload, and the *impact* on operations if that were to happen
- **Clusters:** A list of clusters on which a recommendation is detected
- **Description:** A brief synopsis of the issue, including how it affects your clusters

4.4.3. Displaying potential issues with your cluster

This section describes how to display the Insights report in **Insights Advisor** on 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 on OpenShift Cluster Manager.
- Remote health reporting is enabled, which is the default.
- You are logged in to OpenShift Cluster Manager.

Procedure

1. Navigate to **Advisor** → **Recommendations** on OpenShift Cluster Manager.
   Depending on the result, the Insights advisor service displays one of the following:
   - **No matching recommendations found**, if Insights did not identify any issues.
   - A list of issues Insights has detected, grouped by risk (low, moderate, important, and critical).
   - **No clusters yet**, if Insights has not yet analyzed the cluster. The analysis starts shortly after the cluster has been installed, registered, and connected to the internet.
2. If any issues are displayed, click the **>** icon in front of the entry for more details.
   Depending on the issue, the details can also contain a link to more information from Red Hat about the issue.

4.4.4. Displaying all Insights advisor service recommendations

The Recommendations view, by default, only displays the recommendations that are detected on your clusters. However, you can view all of the recommendations in the advisor service's archive.

Prerequisites

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

Procedure

1. Navigate to **Advisor → Recommendations** on OpenShift Cluster Manager.
2. Click the **X** icons next to the **Clusters Impacted** and **Status** filters.
   You can now browse through all of the potential recommendations for your cluster.

4.4.5. Advisor recommendation filters

The Insights advisor service can return a large number of recommendations. To focus on your most critical recommendations, you can apply filters to the Advisor recommendations list to remove low-priority recommendations.

By default, filters are set to only show enabled recommendations that are impacting one or more clusters. To view all or disabled recommendations in the Insights library, you can customize the filters.

To apply a filter, select a filter type and then set its value based on the options that are available in the drop-down list. You can apply multiple filters to the list of recommendations.

You can set the following filter types:

- **Name:** Search for a recommendation by name.
- **Total risk:** Select one or more values from **Critical**, **Important**, **Moderate**, and **Low** indicating the likelihood and the severity of a negative impact on a cluster.
- **Impact:** Select one or more values from **Critical**, **High**, **Medium**, and **Low** indicating the potential impact to the continuity of cluster operations.
- **Likelihood:** Select one or more values from **Critical**, **High**, **Medium**, and **Low** indicating the potential for a negative impact to a cluster if the recommendation comes to fruition.
- **Category:** Select one or more categories from **Service Availability**, **Performance**, **Fault Tolerance**, **Security**, and **Best Practice** to focus your attention on.
- **Status:** Click a radio button to show enabled recommendations (default), disabled recommendations, or all recommendations.
- **Clusters impacted:** Set the filter to show recommendations currently impacting one or more clusters, non-impacting recommendations, or all recommendations.
- **Risk of change:** Select one or more values from **High**, **Moderate**, **Low**, and **Very low** indicating the risk that the implementation of the resolution could have on cluster operations.

4.4.5.1. Filtering Insights advisor service recommendations

As an OpenShift Container Platform cluster manager, you can filter the recommendations that are displayed on the recommendations list. By applying filters, you can reduce the number of reported recommendations and concentrate on your highest priority recommendations.

The following procedure demonstrates how to set and remove **Category** filters; however, the procedure is applicable to any of the filter types and respective values.

Prerequisites

You are logged in to the OpenShift Cluster Manager in the Hybrid Cloud Console.

Procedure

1. Go to **OpenShift** > **Advisor** > **Recommendations**.
2. In the main, filter-type drop-down list, select the **Category** filter type.
3. Expand the filter-value drop-down list and select the checkbox next to each category of recommendation you want to view. Leave the checkboxes for unnecessary categories clear.
4. Optional: Add additional filters to further refine the list.

Only recommendations from the selected categories are shown in the list.

Verification

- After applying filters, you can view the updated recommendations list. The applied filters are added next to the default filters.

4.4.5.2. Removing filters from Insights advisor service recommendations

You can apply multiple filters to the list of recommendations. When ready, you can remove them individually or completely reset them.

Removing filters individually

- Click the **X** icon next to each filter, including the default filters, to remove them individually.

Removing all non-default filters

- Click **Reset filters** to remove only the filters that you applied, leaving the default filters in place.

4.4.6. Disabling Insights advisor service recommendations

You can disable specific recommendations that affect your clusters, so that they no longer appear in your reports. It is possible to disable a recommendation for a single cluster or all of your clusters.

NOTE

Disabling a recommendation for all of your clusters also applies to any future clusters.

Prerequisites

- Remote health reporting is enabled, which is the default.

- Your cluster is registered on OpenShift Cluster Manager.
- You are logged in to OpenShift Cluster Manager.

Procedure

1. Navigate to **Advisor** → **Recommendations** on OpenShift Cluster Manager.
2. Optional: Use the **Clusters Impacted** and **Status** filters as needed.
3. Disable an alert by using one of the following methods:
    - To disable an alert:
        a. Click the **Options** menu ⋮ for that alert, and then click **Disable recommendation**.
        b. Enter a justification note and click **Save**.
    - To view the clusters affected by this alert before disabling the alert:
        a. Click the name of the recommendation to disable. You are directed to the single recommendation page.
        b. Review the list of clusters in the **Affected clusters** section.
        c. Click **Actions** → **Disable recommendation** to disable the alert for all of your clusters.
        d. Enter a justification note and click **Save**.

4.4.7. Enabling a previously disabled Insights advisor service recommendation

When a recommendation is disabled for all clusters, you no longer see the recommendation in the Insights advisor service. You can change this behavior.

Prerequisites

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

Procedure

1. Navigate to **Advisor** → **Recommendations** on OpenShift Cluster Manager.
2. Filter the recommendations to display on the disabled recommendations:
    a. From the **Status** drop-down menu, select **Status**.
    b. From the **Filter by status** drop-down menu, select **Disabled**.
    c. Optional: Clear the **Clusters impacted** filter.
3. Locate the recommendation to enable.

4. Click the **Options** menu ⋮ , and then click **Enable recommendation**.

4.4.8. About Insights advisor service recommendations for workloads

You can use the Red Hat Insights for OpenShift advisor service to view and manage information about recommendations that affect not only your clusters, but also your workloads. The advisor service takes advantage of deployment validation and helps OpenShift cluster administrators to see all runtime violations of deployment policies. You can see recommendations for workloads at OpenShift > Advisor > Workloads on the Red Hat Hybrid Cloud Console. For more information, see these additional resources:

- Information about Kubernetes workloads
- Boost your cluster operations with Deployment Validation and Insights Advisor for Workloads
- Identifying workload recommendations for namespaces in your clusters
- Viewing workload recommendations for namespaces in your cluster
- Excluding objects from workload recommendations in your clusters

4.4.9. 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 OpenShift Cluster Manager.

Prerequisites

- Your cluster is registered in 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 risk. Click the individual categories or **View all recommendations in Insights Advisor** to display more details.

4.5. USING 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 in the Insights Advisor service on Red Hat Hybrid Cloud Console.

Additional resources

- Remote health reporting.

- For more information on using the Insights advisor service to identify issues with your cluster, see Using Insights to identify issues with your cluster .

4.5.1. Configuring Insights Operator

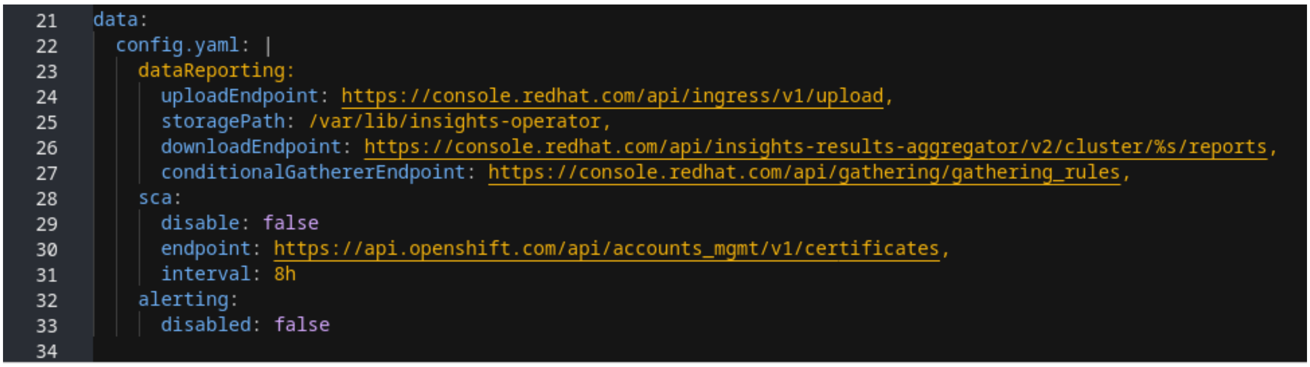
Insights Operator configuration is a combination of the default Operator configuration and the configuration that is stored in either the **insights-config ConfigMap** object in the **openshift-insights** namespace, OR in the support secret in the **openshift-config** namespace.

When a **ConfigMap** object or support secret exists, the contained attribute values override the default Operator configuration values. If both a **ConfigMap** object *and* a support secret exist, the Operator reads the **ConfigMap** object.

The **ConfigMap** object does not exist by default, so an OpenShift Container Platform cluster administrator must create it.

ConfigMap object configuration structure

This example of an **insights-config ConfigMap** object (**config.yaml** configuration) shows configuration options by using standard YAML formatting.



```
data:
  config.yaml: |
    dataReporting:
      uploadEndpoint: https://console.redhat.com/api/ingress/v1/upload,
      storagePath: /var/lib/insights-operator,
      downloadEndpoint: https://console.redhat.com/api/insights-results-aggregator/v2/cluster/%s/reports,
      conditionalGathererEndpoint: https://console.redhat.com/api/gathering/gathering_rules,
    sca:
      disable: false
      endpoint: https://api.openshift.com/api/accounts_mgmt/v1/certificates,
      interval: 8h
    alerting:
      disabled: false
```

Configurable attributes and default values

The following table describes the available configuration attributes:

NOTE

The **insights-config ConfigMap** object follows standard YAML formatting, wherein child values are below the parent attribute and indented two spaces. For the **Obfuscation** attribute, enter values as bulleted children of the parent attribute.

Table 4.1. Insights Operator configurable attributes

Attribute name	Description	Value type	Default value
`alerting:` `  disabled: false`	Disables Insights Operator alerts to the cluster Prometheus instance.	Boolean	**false**

Attribute name	Description	Value type	Default value
`clusterTransfer:` `  endpoint: <url>`	The endpoint for checking and downloading cluster transfer data.	URL	https://api.openshift.com/api/accounts_mgmt/v1/cluster_transfers/
`clusterTransfer:` `  interval: 1h0m0s`	Sets the frequency for checking available cluster transfers.	Time interval	**24h**
`dataReporting:` `  interval: 30m0s`	Sets the data gathering and upload frequency.	Time interval	**2h**
`dataReporting:` `  uploadEndpoint: <url>`	Sets the upload endpoint.	URL	https://console.redhat.com/api/ingress/v1/upload
`dataReporting:` `  storagePath: <path>`	Configures the path where archived data gets stored.	File path	/var/lib/insights-operator
`dataReporting:` `  downloadEndpoint: <url>`	Specifies the endpoint for downloading the latest Insights analysis.	URL	https://console.redhat.com/api/ingress/v1/download
`dataReporting:` `  conditionalGathererEndpoint: <url>`	Sets the endpoint for providing conditional gathering rule definitions.	URL	https://console.redhat.com/api/gathering/gathering_rules

`dataReporting:`
`  obfuscation:`
`  - networking`
	Enables the global obfuscation of IP addresses and the cluster domain name.	String	Not applicable

Attribute name	Description	Value type	Default value

`dataReporting:`
`  obfuscation:`
`  - workload_names`
	Enables the obfuscation of Data Validation Operator data. The cluster resource the resource ID is only visible in the archive file and not the resource name.	String	Not applicable

`proxy:`
`  httpProxy:`
`http://example.com,`
`  httpsProxy:`
`http://example.com,`
`  noProxy: test.org`
	Set custom proxy for Insights Operator.	URL	No default

`sca:` `  interval: 8h0m0s`	Specifies the frequency of the simple content access (SCA) entitlements download.	Time interval	**2h**
`sca:` `  endpoint: <url>`	Specifies the endpoint for downloading the simple content access (SCA) entitlements.	URL	https://api.openshift.com/api/accounts_mgmt/v1/entitlement_certificates
`sca:` `  disabled: false`	Disables the simple content access entitlements download.	Boolean	**false**

4.5.1.1. Creating the insights-config ConfigMap object

This procedure describes how to create the **insights-config ConfigMap** object for the Insights Operator to set custom configurations.

IMPORTANT

Red Hat recommends you consult Red Hat Support before making changes to the default Insights Operator configuration.

Prerequisites

- Remote health reporting is enabled, which is the default.
- You are logged in to the OpenShift Container Platform web console as a user with **cluster-admin** role.

Procedure

1. Go to **Workloads** → **ConfigMaps** and select **Project: openshift-insights**.
2. Click **Create ConfigMap**.
3. Select **Configure via: YAML view** and enter your configuration preferences, for example

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: insights-config
  namespace: openshift-insights
data:
  config.yaml: |
    dataReporting:
      obfuscation:
        - networking
        - workload_names
    sca:
      disabled: false
      interval: 2h
    alerting:
       disabled: false
binaryData: {}
immutable: false
```

4. Optional: Select **Form view** and enter the necessary information that way.
5. In the **ConfigMap Name** field, enter **insights-config**.
6. In the **Key** field, enter **config.yaml**.
7. For the **Value** field, either browse for a file to drag and drop into the field or enter your configuration parameters manually.
8. Click **Create** and you can see the **ConfigMap** object and configuration information.

4.5.2. Understanding Insights Operator alerts

The Insights Operator declares alerts through the Prometheus monitoring system to the Alertmanager. You can view these alerts in the Alerting UI in the OpenShift Container Platform web console by using one of the following methods:

- In the **Administrator** perspective, click **Observe** → **Alerting**.
- In the **Developer** perspective, click **Observe** → <project_name> → **Alerts** tab.

Currently, Insights Operator sends the following alerts when the conditions are met:

Table 4.2. Insights Operator alerts

Alert	Description
InsightsDisabled	Insights Operator is disabled.

Alert	Description
SimpleContentAccessNotAvailable	Simple content access is not enabled in Red Hat Subscription Management.
InsightsRecommendationActive	Insights has an active recommendation for the cluster.

4.5.2.1. Disabling Insights Operator alerts

To prevent the Insights Operator from sending alerts to the cluster Prometheus instance, you create or edit the **insights-config** **ConfigMap** object.

> NOTE
>
> Previously, a cluster administrator would create or edit the Insights Operator configuration using a **support secret** in the **openshift-config** namespace. Red Hat Insights now supports the creation of a **ConfigMap** object to configure the Operator. The Operator gives preference to the config map configuration over the support secret if both exist.

If the **insights-config** **ConfigMap** object does not exist, you must create it when you first add custom configurations. Note that configurations within the **ConfigMap** object take precedence over the default settings defined in the **config/pod.yaml** file.

Prerequisites

- Remote health reporting is enabled, which is the default.
- You are logged in to the OpenShift Container Platform web console as **cluster-admin**.
- The **insights-config** **ConfigMap** object exists in the **openshift-insights** namespace.

Procedure

1. Go to **Workloads** → **ConfigMaps** and select **Project: openshift-insights**.
2. Click on the **insights-config** **ConfigMap** object to open it.
3. Click **Actions** and select **Edit ConfigMap**.
4. Click the **YAML view** radio button.
5. In the file, set the **alerting** attribute to **disabled: true**.

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
```

```
    alerting:
      disabled: true
# ...
```

6. Click **Save**. The **insights-config** config-map details page opens.

7. Verify that the value of the **config.yaml alerting** attribute is set to **disabled: true**.

After you save the changes, Insights Operator no longer sends alerts to the cluster Prometheus instance.

4.5.2.2. Enabling Insights Operator alerts

When alerts are disabled, the Insights Operator no longer sends alerts to the cluster Prometheus instance. You can reenable them.

> **NOTE**
>
> Previously, a cluster administrator would create or edit the Insights Operator configuration using a **support secret** in the **openshift-config** namespace. Red Hat Insights now supports the creation of a **ConfigMap** object to configure the Operator. The Operator gives preference to the config map configuration over the support secret if both exist.

Prerequisites

- Remote health reporting is enabled, which is the default.
- You are logged in to the OpenShift Container Platform web console as **cluster-admin**.
- The **insights-config ConfigMap** object exists in the **openshift-insights** namespace.

Procedure

1. Go to **Workloads** → **ConfigMaps** and select **Project: openshift-insights**.

2. Click on the **insights-config ConfigMap** object to open it.

3. Click **Actions** and select **Edit ConfigMap**.

4. Click the **YAML view** radio button.

5. In the file, set the **alerting** attribute to **disabled: false**.

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    alerting:
      disabled: false
# ...
```

6. Click **Save**. The **insights-config** config-map details page opens.

7. Verify that the value of the **config.yaml alerting** attribute is set to **disabled: false**.

After you save the changes, Insights Operator again sends alerts to the cluster Prometheus instance.

4.5.3. Downloading your Insights Operator archive

Insights Operator stores gathered data in an archive located in the **openshift-insights** namespace of your cluster. You can download and review the data that is gathered by the Insights Operator.

Prerequisites

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

Procedure

1. Find the name of the running pod for the Insights Operator:

```
$ 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_name>:/var/lib/insights-operator ./insights-data ①
```

① ① Replace **<insights_operator_pod_name>** with the pod name output from the preceding command.

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

4.5.4. Running an Insights Operator gather operation

You can run Insights Operator data gather operations on demand. The following procedures describe how to run the default list of gather operations using the OpenShift web console or CLI. You can customize the on demand gather function to exclude any gather operations you choose. Disabling gather operations from the default list degrades Insights Advisor's ability to offer effective recommendations for your cluster. If you have previously disabled Insights Operator gather operations in your cluster, this procedure will override those parameters.

IMPORTANT

The **DataGather** custom resource 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 Technology Preview Features Support Scope .

NOTE

If you enable Technology Preview in your cluster, the Insights Operator runs gather operations in individual pods. This is part of the Technology Preview feature set for the Insights Operator and supports the new data gathering features.

4.5.4.1. Viewing Insights Operator gather durations

You can view the time it takes for the Insights Operator to gather the information contained in the archive. This helps you to understand Insights Operator resource usage and issues with Insights Advisor.

Prerequisites

- A recent copy of your Insights Operator archive.

Procedure

1. From your archive, open **/insights-operator/gathers.json**.
   The file contains a list of Insights Operator gather operations:

```
{
  "name": "clusterconfig/authentication",
  "duration_in_ms": 730, ①
  "records_count": 1,
  "errors": null,
  "panic": null
}
```

① **duration_in_ms** is the amount of time in milliseconds for each gather operation.

2. Inspect each gather operation for abnormalities.

4.5.4.2. Running an Insights Operator gather operation from the web console

To collect data, you can run an Insights Operator gather operation by using the OpenShift Container Platform web console.

Prerequisites

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

Procedure

1. On the console, select **Administration → CustomResourceDefinitions**.
2. On the **CustomResourceDefinitions** page, in the **Search by name** field, find the **DataGather** resource definition, and then click it.
3. On the **CustomResourceDefinition details** page, click the **Instances** tab.
4. Click **Create DataGather**.

5. To create a new **DataGather** operation, edit the following configuration file and then save your changes.

```
apiVersion: insights.openshift.io/v1alpha1
kind: DataGather
metadata:
  name: <your_data_gather> ①
spec:
 gatherers: ②
   - name: workloads
     state: Disabled
```

① Under **metadata**, replace **<your_data_gather>** with a unique name for the gather operation.

② Under **gatherers**, specify any individual gather operations that you intend to disable. In the example provided, **workloads** is the only data gather operation that is disabled and all of the other default operations are set to run. When the **spec** parameter is empty, all of the default gather operations run.

IMPORTANT

Do not add a prefix of **periodic-gathering-** to the name of your gather operation because this string is reserved for other administrative operations and might impact the intended gather operation.

Verification

1. On the console, select to **Workloads** → **Pods**.
2. On the Pods page, go to the **Project** pull-down menu, and then select **Show default projects**.
3. Select the **openshift-insights** project from the **Project** pull-down menu.
4. Check that your new gather operation is prefixed with your chosen name under the list of pods in the **openshift-insights** project. Upon completion, the Insights Operator automatically uploads the data to Red Hat for processing.

4.5.4.3. Running an Insights Operator gather operation from the OpenShift CLI

You can run an Insights Operator gather operation by using the OpenShift Container Platform command-line interface.

Prerequisites

- You are logged in to OpenShift Container Platform as a user with the **cluster-admin** role.

Procedure

- Enter the following command to run the gather operation:

```
$ oc apply -f <your_datagather_definition>.yaml
```

Replace **<your_datagather_definition>.yaml** with a configuration file that contains the following parameters:

```
apiVersion: insights.openshift.io/v1alpha1
kind: DataGather
metadata:
  name: <your_data_gather> ①
spec:
 gatherers: ②
  - name: workloads
    state: Disabled
```

① Under **metadata**, replace **<your_data_gather>** with a unique name for the gather operation.

② Under **gatherers**, specify any individual gather operations that you intend to disable. In the example provided, **workloads** is the only data gather operation that is disabled and all of the other default operations are set to run. When the **spec** parameter is empty, all of the default gather operations run.

IMPORTANT

Do not add a prefix of **periodic-gathering-** to the name of your gather operation because this string is reserved for other administrative operations and might impact the intended gather operation.

Verification

- Check that your new gather operation is prefixed with your chosen name under the list of pods in the **openshift-insights** project. Upon completion, the Insights Operator automatically uploads the data to Red Hat for processing.

Additional resources

- Insights Operator Gathered Data GitHub repository

4.5.4.4. Disabling the Insights Operator gather operations

You can disable the Insights Operator gather operations. Disabling the gather operations gives you the ability to increase privacy for your organization as Insights Operator will no longer gather and send Insights cluster reports to Red Hat. This will disable Insights analysis and recommendations for your cluster without affecting other core functions that require communication with Red Hat such as cluster transfers. You can view a list of attempted gather operations for your cluster from the **/insights-operator/gathers.json** file in your Insights Operator archive. Be aware that some gather operations only occur when certain conditions are met and might not appear in your most recent archive.

IMPORTANT

The **InsightsDataGather** custom resource 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 Technology Preview Features Support Scope .

NOTE

If you enable Technology Preview in your cluster, the Insights Operator runs gather operations in individual pods. This is part of the Technology Preview feature set for the Insights Operator and supports the new data gathering features.

Prerequisites

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

Procedure

1. Navigate to **Administration** → **CustomResourceDefinitions**.
2. On the **CustomResourceDefinitions** page, use the **Search by name** field to find the **InsightsDataGather** resource definition and click it.
3. On the **CustomResourceDefinition details** page, click the **Instances** tab.
4. Click **cluster**, and then click the **YAML** tab.
5. Disable the gather operations by performing one of the following edits to the **InsightsDataGather** configuration file:
    a. To disable all the gather operations, enter **all** under the **disabledGatherers** key:

```
apiVersion: config.openshift.io/v1alpha1
kind: InsightsDataGather
metadata:
....

spec: ①
  gatherConfig:
    disabledGatherers:
      - all ②
```

① The **spec** parameter specifies gather configurations.

② The **all** value disables all gather operations.

b. To disable individual gather operations, enter their values under the **disabledGatherers** key:

```
spec:
  gatherConfig:
    disabledGatherers:
      - clusterconfig/container_images ①
      - clusterconfig/host_subnets
      - workloads/workload_info
```

① Example individual gather operation

6. Click **Save**.
   After you save the changes, the Insights Operator gather configurations are updated and the operations will no longer occur.

NOTE

Disabling gather operations degrades the Insights advisor service's ability to offer effective recommendations for your cluster.

4.5.4.5. Enabling the Insights Operator gather operations

You can enable the Insights Operator gather operations, if the gather operations have been disabled.

IMPORTANT

The **InsightsDataGather** custom resource 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 Technology Preview Features Support Scope .

Prerequisites

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

Procedure

1. Navigate to **Administration** → **CustomResourceDefinitions**.
2. On the **CustomResourceDefinitions** page, use the **Search by name** field to find the **InsightsDataGather** resource definition and click it.
3. On the **CustomResourceDefinition details** page, click the **Instances** tab.
4. Click **cluster**, and then click the **YAML** tab.
5. Enable the gather operations by performing one of the following edits:

- To enable all disabled gather operations, remove the **gatherConfig** stanza:

```
apiVersion: config.openshift.io/v1alpha1
kind: InsightsDataGather
metadata:
....

spec:
  gatherConfig: ①
    disabledGatherers: all
```

① Remove the **gatherConfig** stanza to enable all gather operations.

- To enable individual gather operations, remove their values under the **disabledGatherers** key:

```
spec:
  gatherConfig:
    disabledGatherers:
      - clusterconfig/container_images ①
      - clusterconfig/host_subnets
      - workloads/workload_info
```

① Remove one or more gather operations.

6. Click **Save**.
After you save the changes, the Insights Operator gather configurations are updated and the affected gather operations start.

NOTE

Disabling gather operations degrades Insights Advisor’s ability to offer effective recommendations for your cluster.

4.5.5. Obfuscating Deployment Validation Operator data

By default, when you install the Deployment Validation Operator (DVO), the name and unique identifier (UID) of a resource are included in the data that is captured and processed by the Insights Operator for OpenShift Container Platform. If you are a cluster administrator, you can configure the Insights Operator to obfuscate data from the Deployment Validation Operator (DVO). For example, you can obfuscate workload names in the archive file that is then sent to Red Hat.

To obfuscate the name of resources, you must manually set the **obfuscation** attribute in the **insights-config ConfigMap** object to include the **workload_names** value, as outlined in the following procedure.

Prerequisites

- Remote health reporting is enabled, which is the default.
- You are logged in to the OpenShift Container Platform web console with the "cluster-admin" role.

- The **insights-config ConfigMap** object exists in the **openshift-insights** namespace.
- The cluster is self managed and the Deployment Validation Operator is installed.

Procedure

1. Go to **Workloads** → **ConfigMaps** and select **Project: openshift-insights**.
2. Click the **insights-config ConfigMap** object to open it.
3. Click **Actions** and select **Edit ConfigMap**.
4. Click the **YAML view** radio button.
5. In the file, set the **obfuscation** attribute with the **workload_names** value.

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    dataReporting:
      obfuscation:
        - workload_names
# ...
```

6. Click **Save**. The **insights-config** config-map details page opens.
7. Verify that the value of the **config.yaml obfuscation** attribute is set to **- workload_names**.

4.6. 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 select to obfuscate the Insights Operator data before upload.

4.6.1. 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
```

Example output

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: insights-operator
  namespace: openshift-insights
# ...
spec:
  template:
# ...
    spec:
      containers:
      - args:
# ...
        image: registry.ci.openshift.org/ocp/4.15-2023-10-12-
212500@sha256:a0aa581400805ad0... ①
# ...
```

① Specifies your **insights-operator** image version.

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

```
apiVersion: batch/v1
kind: Job
metadata:
  name: insights-operator-job
# ...
spec:
# ...
  template:
    spec:
    initContainers:
    - name: insights-operator
      image: image: registry.ci.openshift.org/ocp/4.15-2023-10-12-
212500@sha256:a0aa581400805ad0... ①
      terminationMessagePolicy: FallbackToLogsOnError
      volumeMounts:
```

① Replace any existing value with your **insights-operator** image version.

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

```
Name:           insights-operator-job
Namespace:       openshift-insights
# ...
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
```

4.6.2. 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 **Cluster List** 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).

4.6.3. 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.

4.7. IMPORTING SIMPLE CONTENT ACCESS ENTITLEMENTS WITH INSIGHTS OPERATOR

Insights Operator periodically imports your simple content access entitlements from OpenShift Cluster Manager and stores them in the **etc-pki-entitlement** secret in the **openshift-config-managed** namespace. Simple content access is a capability in Red Hat subscription tools which simplifies the behavior of the entitlement tooling. This feature makes it easier to consume the content provided by your Red Hat subscriptions without the complexity of configuring subscription tooling.

NOTE

Previously, a cluster administrator would create or edit the Insights Operator configuration using a **support secret** in the **openshift-config** namespace. Red Hat Insights now supports the creation of a **ConfigMap** object to configure the Operator. The Operator gives preference to the config map configuration over the support secret if both exist.

The Insights Operator imports simple content access entitlements every eight hours, but can be configured or disabled using the **insights-config ConfigMap** object in the **openshift-insights** namespace.

NOTE

Simple content access must be enabled in Red Hat Subscription Management for the importing to function.

Additional resources

- See About simple content access in the Red Hat Subscription Central documentation, for more information about simple content access.
- See Using Red Hat subscriptions in builds for more information about using simple content access entitlements in OpenShift Container Platform builds.

4.7.1. Configuring simple content access import interval

You can configure how often the Insights Operator imports the simple content access (sca) entitlements by using the **insights-config ConfigMap** object in the **openshift-insights** namespace. The entitlement import normally occurs every eight hours, but you can shorten this sca interval if you update your simple content access configuration in the **insights-config ConfigMap** object.

This procedure describes how to update the import interval to two hours (2h). You can specify hours (h) or hours and minutes, for example: 2h30m.

Prerequisites

- Remote health reporting is enabled, which is the default.
- You are logged in to the OpenShift Container Platform web console as a user with the **cluster-admin** role.
- The **insights-config ConfigMap** object exists in the **openshift-insights** namespace.

Procedure

1. Go to **Workloads** → **ConfigMaps** and select **Project: openshift-insights**.
2. Click on the **insights-config ConfigMap** object to open it.
3. Click **Actions** and select **Edit ConfigMap**.
4. Click the **YAML view** radio button.
5. Set the **sca** attribute in the file to **interval: 2h** to import content every two hours.

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    sca:
      interval: 2h
# ...
```

6. Click **Save**. The **insights-config** config-map details page opens.
7. Verify that the value of the **config.yaml sca** attribute is set to **interval: 2h**.

4.7.2. Disabling simple content access import

You can disable the importing of simple content access entitlements by using the **insights-config ConfigMap** object in the **openshift-insights** namespace.

Prerequisites

- Remote health reporting is enabled, which is the default.
- You are logged in to the OpenShift Container Platform web console as **cluster-admin**.
- The **insights-config ConfigMap** object exists in the **openshift-insights** namespace.

Procedure

1. Go to **Workloads** → **ConfigMaps** and select **Project: openshift-insights**.
2. Click on the **insights-config ConfigMap** object to open it.
3. Click **Actions** and select **Edit ConfigMap**.
4. Click the **YAML view** radio button.
5. In the file, set the **sca** attribute to **disabled: true**.

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    sca:
      disabled: true
# ...
```

6. Click **Save**. The **insights-config** config-map details page opens.
7. Verify that the value of the **config.yaml sca** attribute is set to **disabled: true**.

4.7.3. Enabling a previously disabled simple content access import

If the importing of simple content access entitlements is disabled, the Insights Operator does not import simple content access entitlements. You can change this behavior.

Prerequisites

- Remote health reporting is enabled, which is the default.
- You are logged in to the OpenShift Container Platform web console as a user with the **cluster-admin** role.
- The **insights-config ConfigMap** object exists in the **openshift-insights** namespace.

Procedure

1. Go to **Workloads** → **ConfigMaps** and select **Project: openshift-insights**.
2. Click on the **insights-config ConfigMap** object to open it.
3. Click **Actions** and select **Edit ConfigMap**.
4. Click the **YAML view** radio button.
5. In the file, set the **sca** attribute to **disabled: false**.

```
apiVersion: v1
kind: ConfigMap
# ...
data:
```

```
  config.yaml: |
    sca:
      disabled: false
# ...
```

6. Click **Save**. The **insights-config** config-map details page opens.

7. Verify that the value of the **config.yaml sca** attribute is set to **disabled: false**.

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 plugin 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-rhel9:v4.16.28
  ```

  - 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.
    - On a Windows operating system, install the **cwRsync** client and add to the **PATH** variable for use with the **oc rsync** command.

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** 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
...
```

Optionally, you can run the **oc adm must-gather** command in a specific namespace by using the **--run-namespace** option.

For example:

```
$ oc adm must-gather --run-namespace <namespace> \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel9:v4.16.28
```

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

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

Procedure

1. Navigate to the directory where you want to store the **must-gather** data.

   NOTE

   If your cluster is in a disconnected environment, 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 in disconnected environments, you must import the default **must-gather** image as an image stream.

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

2. Run the **oc adm must-gather** command:

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

   IMPORTANT

   If you are in a disconnected environment, use the **--image** flag as part of must-gather and point to the payload image.

NOTE

Because this command picks a random control plane node by default, the pod might be scheduled to a control plane node that is in the **NotReady** and **SchedulingDisabled** state.

a. 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.

NOTE

Contact Red Hat Support for the recommended resources to gather.

3. Create a compressed file from the **must-gather** directory that was just created in your working directory. Make sure you provide the date and cluster ID for the unique must-gather data. For more information about how to find the cluster ID, see How to find the cluster-id or name on OpenShift cluster. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar cvaf must-gather-`date +"%m-%d-%Y-%H-%M-%S"`-<cluster_id>.tar.gz
<must_gather_local_dir> ①
```

① Replace **<must_gather_local_dir>** with the actual directory name.

4. Attach the compressed file to your support case on the the **Customer Support** page of 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-rhel9:v4.16.28	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:<installed_version_service_mesh>	Data collection for Red Hat OpenShift Service Mesh.
registry.redhat.io/multicluster-engine/must-gather-rhel8	Data collection for hosted control planes.

Image	Purpose
registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v<installed_version_migration_toolkit>	Data collection for the Migration Toolkit for Containers.
registry.redhat.io/odf4/odf-must-gather-rhel9:v<installed_version_ODF>	Data collection for Red Hat OpenShift Data Foundation.
registry.redhat.io/openshift-logging/cluster-logging-rhel9-operator:v<installed_version_logging>	Data collection for logging.
quay.io/netobserv/must-gather	Data collection for the Network Observability Operator.
registry.redhat.io/openshift4/ose-csi-driver-shared-resource-mustgather-rhel8	Data collection for OpenShift Shared Resource CSI Driver.
registry.redhat.io/openshift4/ose-local-storage-mustgather-rhel9:v<installed_version_LSO>	Data collection for Local Storage Operator.
registry.redhat.io/openshift-sandboxed-containers/osc-must-gather-rhel8:v<installed_version_sandboxed_containers>	Data collection for OpenShift sandboxed containers.
registry.redhat.io/workload-availability/node-healthcheck-must-gather-rhel8:v<installed_version_NHC>	Data collection for the Red Hat Workload Availability Operators, including the Self Node Remediation (SNR) Operator, the Fence Agents Remediation (FAR) Operator, the Machine Deletion Remediation (MDR) Operator, the Node Health Check (NHC) Operator, and the Node Maintenance Operator (NMO). Use this image if your NHC Operator version is **earlier than 0.9.0**. For more information, see the "Gathering data" section for the specific Operator in Remediation, fencing, and maintenance (Workload Availability for Red Hat OpenShift documentation).

Image	Purpose
registry.redhat.io/workload-availability/node-healthcheck-must-gather-rhel9:v<installed_version_NHC>	Data collection for the Red Hat Workload Availability Operators, including the Self Node Remediation (SNR) Operator, the Fence Agents Remediation (FAR) Operator, the Machine Deletion Remediation (MDR) Operator, the Node Health Check (NHC) Operator, and the Node Maintenance Operator (NMO). Use this image if your NHC Operator version is **0.9.0. or later**. For more information, see the "Gathering data" section for the specific Operator in Remediation, fencing, and maintenance (Workload Availability for Red Hat OpenShift documentation).
registry.redhat.io/numaresources/numaresources-must-gather-rhel9:v<installed-version-nro>	Data collection for the NUMA Resources Operator (NRO).
registry.redhat.io/openshift4/ptp-must-gather-rhel8:v<installed-version-ptp>	Data collection for the PTP Operator.
registry.redhat.io/openshift-gitops-1/must-gather-rhel8:v<installed_version_GitOps>	Data collection for Red Hat OpenShift GitOps.
registry.redhat.io/openshift4/ose-secrets-store-csi-mustgather-rhel9:v<installed_version_secret_store>	Data collection for the Secrets Store CSI Driver Operator.
registry.redhat.io/lvms4/lvms-must-gather-rhel9:v<installed_version_LVMS>	Data collection for the LVM Operator.
registry.redhat.io/compliance/openshift-compliance-must-gather-rhel8:<digest-version>	Data collection for the Compliance Operator.

NOTE

To determine the latest version for an OpenShift Container Platform component's image, see the OpenShift Operator Life Cycles web page on the Red Hat Customer Portal.

Prerequisites

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

NOTE

- To collect the default **must-gather** data in addition to specific feature data, add the **--image-stream=openshift/must-gather** argument.
- For information on gathering data about the Custom Metrics Autoscaler, see the Additional resources section that follows.

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-rhel9:v4.16.28 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. 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 KubeVirt:

```
$ oc adm must-gather \
 --image-stream=openshift/must-gather \ ①
 --image=quay.io/kubevirt/must-gather ②
```

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

② The must-gather image for KubeVirt

4. Create a compressed file from the **must-gather** directory that was just created in your working directory. Make sure you provide the date and cluster ID for the unique must-gather data. For more information about how to find the cluster ID, see How to find the cluster-id or name on OpenShift cluster. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar cvaf must-gather-`date +"%m-%d-%Y-%H-%M-%S"`-<cluster_id>.tar.gz
<must_gather_local_dir> ①
```

① Replace **<must_gather_local_dir>** with the actual directory name.

5. Attach the compressed file to your support case on the the **Customer Support** page of the Red Hat Customer Portal.

5.2. ADDITIONAL RESOURCES

- Gathering debugging data for the Custom Metrics Autoscaler.
- Red Hat OpenShift Container Platform Life Cycle Policy

5.2.1. Gathering network logs

You can gather network logs on all nodes in a cluster.

Procedure

1. Run the **oc adm must-gather** command with **-- gather_network_logs**:

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

NOTE

By default, the **must-gather** tool collects the OVN **nbdb** and **sbdb** databases from all of the nodes in the cluster. Adding the **-- gather_network_logs** option to include additional logs that contain OVN-Kubernetes transactions for OVN **nbdb** database.

2. Create a compressed file from the **must-gather** directory that was just created in your working directory. Make sure you provide the date and cluster ID for the unique must-gather data. For more information about how to find the cluster ID, see How to find the cluster-id or name on OpenShift cluster. For example, on a computer that uses a Linux operating system, run the following command:

   ```
   $ tar cvaf must-gather-`date +"%m-%d-%Y-%H-%M-%S"`-<cluster_id>.tar.gz
   <must_gather_local_dir> ①
   ```

   ① Replace **<must_gather_local_dir>** with the actual directory name.

3. Attach the compressed file to your support case on the the **Customer Support** page of the Red Hat Customer Portal.

5.2.2. Changing the must-gather storage limit

When using the **oc adm must-gather** command to collect data the default maximum storage for the information is 30% of the storage capacity of the container. After the 30% limit is reached the container is killed and the gathering process stops. Information already gathered is downloaded to your local storage. To run the must-gather command again, you need either a container with more storage capacity or to adjust the maximum volume percentage.

If the container reaches the storage limit, an error message similar to the following example is generated.

Example output

```
Disk usage exceeds the volume percentage of 30% for mounted directory. Exiting...
```

Prerequisites

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

Procedure

- Run the **oc adm must-gather** command with the **volume-percentage** flag. The new value cannot exceed 100.

  ```
  $ oc adm must-gather --volume-percentage <storage_percentage>
  ```

5.3. 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

- You have access to the cluster as a user with the **cluster-admin** role.
- You have 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** and select **Share Feedback** from the list.
  b. Click **Open a support case** from the **Tell us about your experience** window.
- To manually obtain your cluster ID using the web console:
  a. Navigate to **Home** → **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"}'
```

5.4. 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**.

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

The recommended way to generate a **sosreport** for an OpenShift Container Platform 4.16 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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 plugins to run **sosreport**:

   ```
   # toolbox
   ```

   NOTE

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

5. Collect a **sosreport** archive.

    a. Run the **sos report** command to collect necessary troubleshooting data on **crio** and **podman**:

    ```
    # sos report -k crio.all=on -k crio.logs=on  -k podman.all=on -k podman.logs=on ①
    ```

    ① **-k** enables you to define **sosreport** plugin parameters outside of the defaults.

    b. Optional: To include information on OVN-Kubernetes networking configurations from a node in your report, run the following command:

    ```
    # sos report --all-logs
    ```

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

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

    e. 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.

        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 4.16 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. 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 the **Customer Support** page of the Red Hat Customer Portal.

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

5.6. 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'
```

5.7. 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.
- You have installed the OpenShift CLI (**oc**).
- Your API service is still functional.
- 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=worker -u kubelet
   ```

   - **kubelet**: Replace 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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>**.

5.8. NETWORK TRACE METHODS

Collecting network traces, in the form of packet capture records, can assist Red Hat Support with troubleshooting network issues.

OpenShift Container Platform supports two ways of performing a network trace. Review the following table and choose the method that meets your needs.

Table 5.2. Supported methods of collecting a network trace

Method	Benefits and capabilities
Collecting a host network trace	You perform a packet capture for a duration that you specify on one or more nodes at the same time. The packet capture files are transferred from nodes to the client machine when the specified duration is met. You can troubleshoot why a specific action triggers network communication issues. Run the packet capture, perform the action that triggers the issue, and use the logs to diagnose the issue.
Collecting a network trace from an OpenShift Container Platform node or container	You perform a packet capture on one node or one container. You run the **tcpdump** command interactively, so you can control the duration of the packet capture. You can start the packet capture manually, trigger the network communication issue, and then stop the packet capture manually. This method uses the **cat** command and shell redirection to copy the packet capture data from the node or container to the client machine.

5.9. COLLECTING A HOST NETWORK TRACE

Sometimes, troubleshooting a network-related issue is simplified by tracing network communication and capturing packets on multiple nodes at the same time.

You can use a combination of the **oc adm must-gather** command and the **registry.redhat.io/openshift4/network-tools-rhel8** container image to gather packet captures from nodes. Analyzing packet captures can help you troubleshoot network communication issues.

The **oc adm must-gather** command is used to run the **tcpdump** command in pods on specific nodes. The **tcpdump** command records the packet captures in the pods. When the **tcpdump** command exits, the **oc adm must-gather** command transfers the files with the packet captures from the pods to your client machine.

TIP

The sample command in the following procedure demonstrates performing a packet capture with the **tcpdump** command. However, you can run any command in the container image that is specified in the **--image** argument to gather troubleshooting information from multiple nodes at the same time.

Prerequisites

- You are logged in to OpenShift Container Platform as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Run a packet capture from the host network on some nodes by running the following command:

```
$ oc adm must-gather \
  --dest-dir /tmp/captures \  <.>
  --source-dir '/tmp/tcpdump/' \  <.>
  --image registry.redhat.io/openshift4/network-tools-rhel8:latest \  <.>
  --node-selector 'node-role.kubernetes.io/worker' \  <.>
  --host-network=true \  <.>
  --timeout 30s \  <.>
  -- \
  tcpdump -i any \  <.>
  -w /tmp/tcpdump/%Y-%m-%dT%H:%M:%S.pcap -W 1 -G 300
```

<.> The **--dest-dir** argument specifies that **oc adm must-gather** stores the packet captures in directories that are relative to **/tmp/captures** on the client machine. You can specify any writable directory. <.> When **tcpdump** is run in the debug pod that **oc adm must-gather** starts, the **--source-dir** argument specifies that the packet captures are temporarily stored in the **/tmp/tcpdump** directory on the pod. <.> The **--image** argument specifies a container image that includes the **tcpdump** command. <.> The **--node-selector** argument and example value specifies to perform the packet captures on the worker nodes. As an alternative, you can specify the **--node-name** argument instead to run the packet capture on a single node. If you omit both the **--node-selector** and the **--node-name** argument, the packet captures are performed on all nodes. <.> The **--host-network=true** argument is required so that the packet captures are performed on the network interfaces of the node. <.> The **--timeout** argument and value specify to run the debug pod for 30 seconds. If you do not specify the **--timeout** argument and a duration, the debug pod runs for 10 minutes. <.> The **-i any** argument for the **tcpdump** command specifies to capture packets on all network interfaces. As an alternative, you can specify a network interface name.

2. Perform the action, such as accessing a web application, that triggers the network communication issue while the network trace captures packets.

3. Review the packet capture files that **oc adm must-gather** transferred from the pods to your client machine:

```
tmp/captures
├── event-filter.html
├── ip-10-0-192-217-ec2-internal ①
│   └── registry-redhat-io-openshift4-network-tools-rhel8-sha256-bca...
│       └── 2022-01-13T19:31:31.pcap
├── ip-10-0-201-178-ec2-internal ②
│   └── registry-redhat-io-openshift4-network-tools-rhel8-sha256-bca...
│       └── 2022-01-13T19:31:30.pcap
├── ip-...
└── timestamp
```

① ② The packet captures are stored in directories that identify the hostname, container, and file name. If you did not specify the **--node-selector** argument, then the directory level for the hostname is not present.

5.10. 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 an existing Red Hat Support case ID.
- You have a Red Hat standard or premium Subscription.
- You have a Red Hat Customer Portal account.
- 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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 plugins 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 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.
     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 4.16 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. 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 the **Customer Support** page of the Red Hat Customer Portal.

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

5.11. 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.

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 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 4.16 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. 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 the **Customer Support** page of the Red Hat Customer Portal.
3. Select **Attach files** and follow the prompts to upload the file.

5.12. 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 plugins that are needed to run commands such as **sosreport**.

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.

5.12.1. Installing packages to a toolbox container

By default, running the **toolbox** command starts a container with the **registry.redhat.io/rhel9/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.
- You can access your system as a user with root privileges.

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>
```

5.12.2. Starting an alternative image with toolbox

By default, running the **toolbox** command starts a container with the **registry.redhat.io/rhel9/support-tools:latest** image.

NOTE

You can start an alternative image by creating a **.toolboxrc** file and specifying the image to run. However, running an older version of the **support-tools** image, such as **registry.redhat.io/rhel8/support-tools:latest**, is not supported on OpenShift Container Platform 4.16.

Prerequisites

- You have accessed a node with the **oc debug node/<node_name>** command.
- You can access your system as a user with root privileges.

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. Optional: If you need to use an alternative image instead of the default image, create a **.toolboxrc** file in the home directory for the root user ID, and specify the image metadata:

```
REGISTRY=quay.io              ①
IMAGE=fedora/fedora:latest    ②
TOOLBOX_NAME=toolbox-fedora-latest ③
```

① Optional: Specify an alternative container registry.

② Specify an alternative image to start.

③ Optional: Specify an alternative name for the toolbox container.

3. Start a toolbox container by entering the following command:

```
# toolbox
```

NOTE

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

CHAPTER 6. SUMMARIZING CLUSTER SPECIFICATIONS

6.1. SUMMARIZING CLUSTER SPECIFICATIONS BY USING A CLUSTER VERSION OBJECT

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
```

Example output

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.13.8    True        False         8h      Cluster version is 4.13.8
```

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

```
$ oc describe clusterversion
```

Example output

```
Name:         version
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  config.openshift.io/v1
Kind:         ClusterVersion
# ...
    Image:    quay.io/openshift-release-dev/ocp-
release@sha256:a956488d295fe5a59c8663a4d9992b9b5d0950f510a7387dbbfb8d20fc5970ce

    URL:      https://access.redhat.com/errata/RHSA-2023:4456
    Version:  4.13.8
  History:
    Completion Time:    2023-08-17T13:20:21Z
    Image:              quay.io/openshift-release-dev/ocp-
release@sha256:a956488d295fe5a59c8663a4d9992b9b5d0950f510a7387dbbfb8d20fc5970ce

    Started Time:       2023-08-17T12:59:45Z
    State:              Completed
```

```
    Verified:           false
    Version:             4.13.8
# ...
```

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 4.16 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** cluster 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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 postinstallation 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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/sssd.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

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

```
$ oc get nodes
```

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

```
$ oc adm top nodes
```

- Summarize CPU and memory usage for a specific node:

```
$ oc adm top node 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
    ```

    > NOTE
    >
    > If you are running **oc debug** on a control plane node, you can find administrative **kubeconfig** files in the **/etc/kubernetes/static-pod-resources/kube-apiserver-certs/secrets/node-kubeconfigs** directory.

    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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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.
- You have installed the OpenShift CLI (**oc**).
- Your API service is still functional.
- 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=worker -u kubelet
   ```

   - **kubelet**: Replace 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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 engine implementation that integrates closely with the operating system to deliver an efficient and optimized Kubernetes experience. The CRI-O container engine runs as 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 have 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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 **crio.service** status summary:

    ```
    # systemctl status crio.service
    ```

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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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 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/<node_name>** 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 <node_name>
   ```

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

   ```
   $ oc adm drain <node_name> --ignore-daemonsets --delete-emptydir-data
   ```

   NOTE

   The **terminationGracePeriodSeconds** attribute of a pod or pod template controls the graceful termination period. This attribute defaults at 30 seconds, but can be customized for each application as necessary. If set to more than 90 seconds, the pod might be marked as **SIGKILLed** and fail to terminate successfully.

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:

   a. Use the following command to stop the pods that are not in the **HostNetwork**. They must be removed first because their removal relies on the networking plugin pods, which are in the **HostNetwork**.

      ```
      .. for pod in $(crictl pods -q); do if [[ "$(crictl inspectp $pod | jq -r
      .status.linux.namespaces.options.network)" != "NODE" ]]; then crictl rmp -f $pod; fi; done
      ```

   b. Stop all other 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   v1.29.4
   ```

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

    ```
    $ oc adm uncordon <node_name>
    ```

    Example output

    ```
    NAME        STATUS      ROLES    AGE    VERSION
    ci-ln-tkbxyft-f76d1-nvwhr-master-1   Ready           master   133m   v1.29.4
    ```

7.4. TROUBLESHOOTING OPERATING SYSTEM ISSUES

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

7.4.1. Investigating kernel crashes

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

The **x86_64** architecture supports kdump in General Availability (GA) status, whereas other architectures support kdump in Technology Preview (TP) status.

The following table provides details about the support level of kdump for different architectures.

Table 7.1. Kdump support in RHCOS

Architecture	Support level
x86_64	GA
aarch64	TP
s390x	TP
ppc64le	TP

IMPORTANT

Kdump support, for the preceding three architectures in the table, 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 Technology Preview Features Support Scope .

7.4.1.1. Enabling kdump

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

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'
```

NOTE

For the **ppc64le** platform, the recommended value for **crashkernel** is **crashkernel=2G-4G:384M,4G-16G:512M,16G-64G:1G,64G-128G:2G,128G-:4G**.

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

   If your node uses LUKS-encrypted devices, you must use network dumps as kdump does not support saving crash dumps to 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.

   IMPORTANT

   If you have multipathing enabled on your primary disk, the dump target must be either an NFS or SSH server and you must exclude the multipath module from your **/etc/kdump.conf** configuration file.

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** systemd service has started and exited successfully and that the command, **cat /sys/kernel/kexec_crash_loaded**, prints the value **1**.

7.4.1.2. 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 the **kdump** service only be enabled on each node as needed. Potential costs of enabling the **kdump** service 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.

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 use a **systemd** unit in a **MachineConfig** object as a day-1 customization 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.

NOTE

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:

NOTE

The Butane version you specify in the config file should match the OpenShift Container Platform version and always ends in **0**. For example, **4.16.0**. See "Creating machine configs with Butane" for information about Butane.

```
variant: openshift
version: 4.16.0
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" ⑥
          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. For the **ppc64le** platform, the recommended value for **crashkernel** is **crashkernel=2G-4G:384M,4G-16G:512M,16G-64G:1G,64G-128G:2G,128G-:4G**.

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.

⑥ For the **ppc64le** platform, replace **nr_cpus=1** with **maxcpus=1**, which is not supported on this platform.

> NOTE
>
> To export the dumps to NFS targets, some kernel modules must be explicitly added to the configuration file:
>
> **Example /etc/kdump.conf file**
>
> ```
> nfs server.example.com:/export/cores
> core_collector makedumpfile -l --message-level 7 -d 31
> extra_bins /sbin/mount.nfs
> extra_modules nfs nfsv3 nfs_layout_nfsv41_files blocklayoutdriver nfs_layout_flexfiles
> nfs_layout_nfsv41_files
> ```

1. 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
   ```

2. Put the YAML file into the **<installation_directory>/manifests/** directory during cluster setup. You can also create this **MachineConfig** object after cluster setup with the YAML file:

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

7.4.1.3. Testing the kdump configuration

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

7.4.1.4. Analyzing a core dump

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

NOTE

It is recommended to perform vmcore analysis on a separate RHEL system.

7.4.1.5. 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 the **kexec** package
- Red Hat Knowledgebase article regarding kexec and kdump

7.4.2. Debugging Ignition failures

If a machine cannot be provisioned, Ignition fails and RHCOS will boot into the emergency shell. Use the following procedure to get debugging information.

Procedure

1. Run the following command to show which service units failed:

```
$ systemctl --failed
```

2. Optional: Run the following command on an individual service unit to find out more information:

```
$ journalctl -u <unit>.service
```

7.5. TROUBLESHOOTING NETWORK ISSUES

7.5.1. 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 **nodeip-configuration.service** selects the IP from the interface associated with the default route.

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 of 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, or if there is a networking layout where the node IP should not come from the default route interface, it is possible for the **nodeip-configuration.service** service to 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, you might receive the following error: **EtcdCertSignerControllerDegraded**. You can create a hint file that includes the **NODEIP_HINT** variable to override the default IP selection logic. For more information, see Optional: Overriding the default node IP selection logic.

7.5.1.1. Optional: Overriding the default node IP selection logic

To override the default IP selection logic, you can create a hint file that includes the **NODEIP_HINT** variable to override the default IP selection logic. Creating a hint file allows you to select a specific node IP address from the interface in the subnet of the IP address specified in the **NODEIP_HINT** variable.

For example, if a node has two interfaces, **eth0** with an address of **10.0.0.10/24**, and **eth1** with an address of **192.0.2.5/24**, and the default route points to **eth0** (**10.0.0.10**),the node IP address would normally use the **10.0.0.10** IP address.

Users can configure the **NODEIP_HINT** variable to point at a known IP in the subnet, for example, a subnet gateway such as **192.0.2.1** so that the other subnet, **192.0.2.0/24**, is selected. As a result, the **192.0.2.5** IP address on **eth1** is used for the node.

The following procedure shows how to override the default node IP selection logic.

Procedure

1. Add a hint file to your **/etc/default/nodeip-configuration** file, for example:

```
NODEIP_HINT=192.0.2.1
```

IMPORTANT

- Do not use the exact IP address of a node as a hint, for example, **192.0.2.5**. Using the exact IP address of a node causes the node using the hint IP address to fail to configure correctly.
- The IP address in the hint file is only used to determine the correct subnet. It will not receive traffic as a result of appearing in the hint file.

2. Generate the **base-64** encoded content by running the following command:

```
$ echo -n 'NODEIP_HINT=192.0.2.1' | base64 -w0
```

Example output

```
Tk9ERUlQX0hJTlQ9MTkyLjAuMCxxxx==
```

3. Activate the hint by creating a machine config manifest for both **master** and **worker** roles before deploying the cluster:

99-nodeip-hint-master.yaml

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-nodeip-hint-master
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,<encoded_content> ①
        mode: 0644
        overwrite: true
        path: /etc/default/nodeip-configuration
```

① Replace **<encoded_contents>** with the base64-encoded content of the **/etc/default/nodeip-configuration** file, for example, **Tk9ERUlQX0hJTlQ9MTkyLjAuMCxxxx==**. Note that a space is not acceptable after the comma and before the encoded content.

99-nodeip-hint-worker.yaml

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
 labels:
   machineconfiguration.openshift.io/role: worker
   name: 99-nodeip-hint-worker
spec:
 config:
   ignition:
     version: 3.2.0
   storage:
     files:
     - contents:
         source: data:text/plain;charset=utf-8;base64,<encoded_content> ①
       mode: 0644
       overwrite: true
       path: /etc/default/nodeip-configuration
```

① Replace **<encoded_contents>** with the base64-encoded content of the **/etc/default/nodeip-configuration** file, for example, **Tk9ERUlQX0hJTlQ9MTkyLjAuMCxxxx==**. Note that a space is not acceptable after the comma and before the encoded content.

4. Save the manifest to the directory where you store your cluster configuration, for example, **~/clusterconfigs**.

5. Deploy the cluster.

7.5.1.2. Configuring OVN-Kubernetes to use a secondary OVS bridge

You can create an additional or *secondary* Open vSwitch (OVS) bridge, **br-ex1**, that OVN-Kubernetes manages and the Multiple External Gateways (MEG) implementation uses for defining external gateways for an OpenShift Container Platform node. You can define a MEG in an **AdminPolicyBasedExternalRoute** custom resource (CR). The MEG implementation provides a pod with access to multiple gateways, equal-cost multipath (ECMP) routes, and the Bidirectional Forwarding Detection (BFD) implementation.

Consider a use case for pods impacted by the Multiple External Gateways (MEG) feature and you want to egress traffic to a different interface, for example **br-ex1**, on a node. Egress traffic for pods not impacted by MEG get routed to the default OVS **br-ex** bridge.

IMPORTANT

Currently, MEG is unsupported for use with other egress features, such as egress IP, egress firewalls, or egress routers. Attempting to use MEG with egress features like egress IP can result in routing and traffic flow conflicts. This occurs because of how OVN-Kubernetes handles routing and source network address translation (SNAT). This results in inconsistent routing and might break connections in some environments where the return path must patch the incoming path.

You must define the additional bridge in an interface definition of a machine configuration manifest file. The Machine Config Operator uses the manifest to create a new file at **/etc/ovnk/extra_bridge** on the host. The new file includes the name of the network interface that the additional OVS bridge configures for a node.

IMPORTANT

Do not use the **nmstate** API to make configuration changes to the secondary interface that is defined in the **/etc/ovnk/extra_bridge** directory path. The **configure-ovs.sh** configuration script creates and manages OVS bridge interfaces, so any interruptive changes to these interfaces by the **nmstate** API can lead to network configuration instability.

After you create and edit the manifest file, the Machine Config Operator completes tasks in the following order:

1. Drains nodes in singular order based on the selected machine configuration pool.
2. Injects Ignition configuration files into each node, so that each node receives the additional **br-ex1** bridge network configuration.
3. Verify that the **br-ex** MAC address matches the MAC address for the interface that **br-ex** uses for the network connection.
4. Executes the **configure-ovs.sh** shell script that references the new interface definition.
5. Adds **br-ex** and **br-ex1** to the host node.
6. Uncordons the nodes.

NOTE

After all the nodes return to the **Ready** state and the OVN-Kubernetes Operator detects and configures **br-ex** and **br-ex1**, the Operator applies the **k8s.ovn.org/l3-gateway-config** annotation to each node.

For more information about useful situations for the additional **br-ex1** bridge and a situation that always requires the default **br-ex** bridge, see "Configuration for a localnet topology".

Procedure

1. Optional: Create an interface connection that your additional bridge, **br-ex1**, can use by completing the following steps. The example steps show the creation of a new bond and its dependent interfaces that are all defined in a machine configuration manifest file. The additional bridge uses the **MachineConfig** object to form a additional bond interface.

   IMPORTANT

   Do not use the Kubernetes NMState Operator or a **NodeNetworkConfigurationPolicy** (NNCP) manifest file to define the additional interface. Ensure that the additional interface or sub-interfaces when defining a **bond** interface are not used by an existing **br-ex** OVN Kubernetes network deployment.

   You cannot make configuration changes to the **br-ex** bridge or its underlying interfaces as a postinstallation task. As a workaround, use a secondary network interface connected to your host or switch.

   a. Create the following interface definition files. These files get added to a machine configuration manifest file so that host nodes can access the definition files.

   #### Example of the first interface definition file that is named eno1.config

   ```
   [connection]
   id=eno1
   type=ethernet
   interface-name=eno1
   master=bond1
   slave-type=bond
   autoconnect=true
   autoconnect-priority=20
   ```

   #### Example of the second interface definition file that is named eno2.config

   ```
   [connection]
   id=eno2
   type=ethernet
   interface-name=eno2
   master=bond1
   slave-type=bond
   autoconnect=true
   autoconnect-priority=20
   ```

Example of the second bond interface definition file that is named bond1.config

```
[connection]
id=bond1
type=bond
interface-name=bond1
autoconnect=true
connection.autoconnect-slaves=1
autoconnect-priority=20

[bond]
mode=802.3ad
miimon=100
xmit_hash_policy="layer3+4"

[ipv4]
method=auto
```

b. Convert the definition files to Base64 encoded strings by running the following command:

```
$ base64 <directory_path>/en01.config
```

```
$ base64 <directory_path>/eno2.config
```

```
$ base64 <directory_path>/bond1.config
```

2. Prepare the environment variables. Replace **<machine_role>** with the node role, such as **worker**, and replace **<interface_name>** with the name of your additional **br-ex** bridge name.

```
$ export ROLE=<machine_role>
```

3. Define each interface definition in a machine configuration manifest file:

Example of a machine configuration file with definitions added for bond1, eno1, and en02

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: ${worker}
  name: 12-${ROLE}-sec-bridge-cni
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:;base64,<base-64-encoded-contents-for-bond1.conf>
        path: /etc/NetworkManager/system-connections/bond1.nmconnection
        filesystem: root
        mode: 0600
```

```
      - contents:
          source: data:;base64,<base-64-encoded-contents-for-eno1.conf>
        path: /etc/NetworkManager/system-connections/eno1.nmconnection
        filesystem: root
        mode: 0600
      - contents:
          source: data:;base64,<base-64-encoded-contents-for-eno2.conf>
        path: /etc/NetworkManager/system-connections/eno2.nmconnection
        filesystem: root
        mode: 0600
# ...
```

4. Create a machine configuration manifest file for configuring the network plugin by entering the following command in your terminal:

```
$ oc create -f <machine_config_file_name>
```

5. Create an Open vSwitch (OVS) bridge, **br-ex1**, on nodes by using the OVN-Kubernetes network plugin to create an **extra_bridge** file`. Ensure that you save the file in the **/etc/ovnk/extra_bridge** path of the host. The file must state the interface name that supports the additional bridge and not the default interface that supports **br-ex**, which holds the primary IP address of the node.

Example configuration for the **extra_bridge** file, **/etc/ovnk/extra_bridge**, that references a additional interface

```
bond1
```

6. Create a machine configuration manifest file that defines the existing static interface that hosts **br-ex1** on any nodes restarted on your cluster:

Example of a machine configuration file that defines **bond1** as the interface for hosting **br-ex1**

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: ${worker}
  name: 12-worker-extra-bridge
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/ovnk/extra_bridge
          mode: 0420
          overwrite: true
          contents:
            source: data:text/plain;charset=utf-8,bond1
          filesystem: root
```

7. Apply the machine-configuration to your selected nodes:

```
$ oc create -f <machine_config_file_name>
```

8. Optional: You can override the **br-ex** selection logic for nodes by creating a machine configuration file that in turn creates a **/var/lib/ovnk/iface_default_hint** resource.

NOTE

The resource lists the name of the interface that **br-ex** selects for your cluster. By default, **br-ex** selects the primary interface for a node based on boot order and the IP address subnet in the machine network. Certain machine network configurations might require that **br-ex** continues to select the default interfaces or bonds for a host node.

   a. Create a machine configuration file on the host node to override the default interface.

IMPORTANT

Only create this machine configuration file for the purposes of changing the **br-ex** selection logic. Using this file to change the IP addresses of existing nodes in your cluster is not supported.

Example of a machine configuration file that overrides the default interface

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: ${worker}
  name: 12-worker-br-ex-override
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /var/lib/ovnk/iface_default_hint
          mode: 0420
          overwrite: true
          contents:
            source: data:text/plain;charset=utf-8,bond0 ①
          filesystem: root
```

① Ensure **bond0** exists on the node before you apply the machine configuration file to the node.

   b. Before you apply the configuration to all new nodes in your cluster, reboot the host node to verify that **br-ex** selects the intended interface and does not conflict with the new interfaces that you defined on **br-ex1**.

   c. Apply the machine configuration file to all new nodes in your cluster:

```
$ oc create -f <machine_config_file_name>
```

Verification

1. Identify the IP addresses of nodes with the **exgw-ip-addresses** label in your cluster to verify that the nodes use the additional bridge instead of the default bridge:

   ```
   $ oc get nodes -o json | grep --color exgw-ip-addresses
   ```

   Example output

   ```
   "k8s.ovn.org/l3-gateway-config":
     \"exgw-ip-address\":\"172.xx.xx.yy/24\",\"next-hops\":[\"xx.xx.xx.xx\"],
   ```

2. Observe that the additional bridge exists on target nodes by reviewing the network interface names on the host node:

   ```
   $ oc debug node/<node_name> -- chroot /host sh -c "ip a | grep mtu | grep br-ex"
   ```

   Example output

   ```
   Starting pod/worker-1-debug ...
   To use host binaries, run `chroot /host`
   # ...
   5: br-ex: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
   UNKNOWN group default qlen 1000
   6: br-ex1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
   UNKNOWN group default qlen 1000
   ```

3. Optional: If you use **/var/lib/ovnk/iface_default_hint**, check that the MAC address of **br-ex** matches the MAC address of the primary selected interface:

   ```
   $ oc debug node/<node_name> -- chroot /host sh -c "ip a | grep -A1 -E 'br-ex|bond0'
   ```

   Example output that shows the primary interface for br-ex as bond0

   ```
   Starting pod/worker-1-debug ...
   To use host binaries, run `chroot /host`
   # ...
   sh-5.1# ip a | grep -A1 -E 'br-ex|bond0'
   2: bond0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
   ovs-system state UP group default qlen 1000
      link/ether fa:16:3e:47:99:98 brd ff:ff:ff:ff:ff:ff
   --
   5: br-ex: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
   UNKNOWN group default qlen 1000
      link/ether fa:16:3e:47:99:98 brd ff:ff:ff:ff:ff:ff
      inet 10.xx.xx.xx/21 brd 10.xx.xx.255 scope global dynamic noprefixroute br-ex
   ```

Additional resources

- Configure an external gateway on the default network

7.5.2. 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.

7.5.2.1. 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
```

7.5.2.2. 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 ①
  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 ②
              ExecReload=-/usr/bin/ovs-appctl vlog/set syslog:dbg
          name: 20-ovs-vswitchd-restart.conf
        name: ovs-vswitchd.service
```

① After you perform this procedure to configure control plane nodes, repeat the procedure and set the role to **worker** to configure worker nodes.

② 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

7.5.2.3. 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 <node_name> -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.

7.6. 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 4.16 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.

7.6.1. Operator subscription condition types

Subscriptions can report the following condition types:

Table 7.2. 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.
ResolutionFailed	The dependency resolution 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.

Additional resources

- Catalog health requirements

7.6.2. 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

```
Name:         cluster-logging
Namespace:    openshift-logging
Labels:       operators.coreos.com/cluster-logging.openshift-logging=
Annotations:  <none>
API Version:  operators.coreos.com/v1alpha1
Kind:         Subscription
# ...
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.

7.6.3. 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
Labels:       <none>
Annotations:  operatorframework.io/managed-by: marketplace-operator
              target.workload.openshift.io/management: {"effect": "PreferredDuringScheduling"}
API Version:  operators.coreos.com/v1alpha1
Kind:         CatalogSource
# ...
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
- States of Connectivity (gRPC documentation)
- Accessing images for Operators from private registries

7.6.4. 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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.

7.6.5. 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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>**.

7.6.6. 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

- 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 editing an **ImageDigestMirrorSet**, **ImageTagMirrorSet**, or **ImageContentSourcePolicy** object, it drains the corresponding nodes, applies the changes, and uncordons the nodes. The node drain does not happen for the following changes:
  - The addition of a registry with the **pull-from-mirror = "digest-only"** parameter set for each mirror.
  - The addition of a mirror with the **pull-from-mirror = "digest-only"** parameter set in a registry.
  - The addition of items to the **unqualified-search-registries** list.

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.

7.6.6.1. 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.

7.6.6.2. 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.

7.6.7. 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>
```

7.6.8. Reinstalling Operators after failed uninstallation

You must successfully and completely uninstall an Operator prior to attempting to reinstall the same Operator. Failure to fully uninstall the Operator properly can leave resources, such as a project or namespace, stuck in a "Terminating" state and cause "error resolving resource" messages. For example:

Example Project resource description

```
...
    message: 'Failed to delete all resource types, 1 remaining: Internal error occurred:
      error resolving resource'
...
```

These types of issues can prevent an Operator from being reinstalled successfully.

> **WARNING**
>
> Forced deletion of a namespace is not likely to resolve "Terminating" state issues and can lead to unstable or unpredictable cluster behavior, so it is better to try to find related resources that might be preventing the namespace from being deleted. For more information, see the Red Hat Knowledgebase Solution #4165791 , paying careful attention to the cautions and warnings.

The following procedure shows how to troubleshoot when an Operator cannot be reinstalled because an existing custom resource definition (CRD) from a previous installation of the Operator is preventing a related namespace from deleting successfully.

Procedure

1. Check if there are any namespaces related to the Operator that are stuck in "Terminating" state:

   ```
   $ oc get namespaces
   ```

   #### Example output

   ```
   operator-ns-1                              Terminating
   ```

2. Check if there are any CRDs related to the Operator that are still present after the failed uninstallation:

   ```
   $ oc get crds
   ```

   > **NOTE**
   >
   > CRDs are global cluster definitions; the actual custom resource (CR) instances related to the CRDs could be in other namespaces or be global cluster instances.

3. If there are any CRDs that you know were provided or managed by the Operator and that should have been deleted after uninstallation, delete the CRD:

   ```
   $ oc delete crd <crd_name>
   ```

4. Check if there are any remaining CR instances related to the Operator that are still present after uninstallation, and if so, delete the CRs:

   a. The type of CRs to search for can be difficult to determine after uninstallation and can require knowing what CRDs the Operator manages. For example, if you are troubleshooting an uninstallation of the etcd Operator, which provides the **EtcdCluster** CRD, you can search for remaining **EtcdCluster** CRs in a namespace:

      ```
      $ oc get EtcdCluster -n <namespace_name>
      ```

      Alternatively, you can search across all namespaces:

      ```
      $ oc get EtcdCluster --all-namespaces
      ```

   b. If there are any remaining CRs that should be removed, delete the instances:

      ```
      $ oc delete <cr_name> <cr_instance_name> -n <namespace_name>
      ```

5. Check that the namespace deletion has successfully resolved:

   ```
   $ oc get namespace <namespace_name>
   ```

IMPORTANT

If the namespace or other Operator resources are still not uninstalled cleanly, contact Red Hat Support.

6. 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>
```

Additional resources

- Deleting Operators from a cluster
- Adding Operators to a cluster

7.7. 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 4.16.

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.

7.7.1. 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 7.3. Pod error states

Pod error state	Description
ErrImagePull	Generic image retrieval error.
ErrImagePullBackOff	Image retrieval failed and is backed off.

Pod error state	Description
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.
ErrTeardownNetwork	Network termination failed.

7.7.2. 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
   ```

7.7.3. 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
   ```

   Example output

   ```
   total 124K
   drwxr-xr-x. 1 root root   33 Aug 11 11:23 .
   drwxr-xr-x. 1 root root   28 Sep  6  2022 ..
   -rw-rw----. 1 root utmp    0 Jul 10 10:31 btmp
   -rw-r--r--. 1 root root  33K Jul 17 10:07 dnf.librepo.log
   -rw-r--r--. 1 root root  69K Jul 17 10:07 dnf.log
   -rw-r--r--. 1 root root 8.8K Jul 17 10:07 dnf.rpm.log
   -rw-r--r--. 1 root root  480 Jul 17 10:07 hawkey.log
   -rw-rw-r--. 1 root utmp    0 Jul 10 10:31 lastlog
   drwx------. 2 root root   23 Aug 11 11:14 openshift-apiserver
   drwx------. 2 root root    6 Jul 10 10:31 private
   drwxr-xr-x. 1 root root   22 Mar  9 08:05 rhsm
   -rw-rw-r--. 1 root utmp    0 Jul 10 10:31 wtmp
   ```

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

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

   Example output

   ```
   2023-07-10T10:29:38+0000 INFO --- logging initialized ---
   2023-07-10T10:29:38+0000 DDEBUG timer: config: 13 ms
   2023-07-10T10:29:38+0000 DEBUG Loaded plugins: builddep, changelog, config-
   manager, copr, debug, debuginfo-install, download, generate_completion_cache, groups-
   manager, needs-restarting, playground, product-id, repoclosure, repodiff, repograph,
   ```

```
repomanage, reposync, subscription-manager, uploadprofile
2023-07-10T10:29:38+0000 INFO Updating Subscription Management repositories.
2023-07-10T10:29:38+0000 INFO Unable to read consumer identity
2023-07-10T10:29:38+0000 INFO Subscription Manager is operating in container mode.
2023-07-10T10:29:38+0000 INFO
```

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>
```

7.7.4. 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.

7.7.5. 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.

7.7.6. 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.

7.8. TROUBLESHOOTING THE SOURCE-TO-IMAGE PROCESS

7.8.1. 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.

Procedure

1. 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:
    a. **During the build configuration stage**, a build pod is used to create an application container image from a base image and application source code.
    b. **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.
    c. **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.
2. When troubleshooting S2I issues, follow this strategy:
    a. Monitor build, deployment, and application pod status.
    b. Determine the stage of the S2I process where the problem occurred.
    c. Review logs corresponding to the failed stage.

7.8.2. 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>
```

7.8.3. 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 4.16 cluster nodes running Red Hat Enterprise Linux CoreOS (RHCOS) are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. 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.

7.8.4. Additional resources

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

7.9. TROUBLESHOOTING STORAGE ISSUES

7.9.1. 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=0
```

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

7.10. TROUBLESHOOTING WINDOWS CONTAINER WORKLOAD ISSUES

7.10.1. 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 .

7.10.2. 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 compute 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
```

7.10.3. 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.

7.10.3.1. 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 compute 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}
```

7.10.3.2. 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 compute 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.

7.10.4. 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 compute 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
```

7.10.5. 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 compute 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 containerd container runtime service:

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

7.10.6. Collecting containerd logs for Windows containers

The Windows containerd container service does not stream log data to stdout, but instead, it stream log data to the Windows event log. You can view the containerd event logs to investigate issues you think might be caused by the Windows containerd container service.

Prerequisites

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

Procedure

- View the containerd logs by running the following command:

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

7.10.7. Additional resources

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

7.11. INVESTIGATING MONITORING ISSUES

OpenShift Container Platform includes a preconfigured, preinstalled, and self-updating monitoring stack that provides monitoring for core platform components. In OpenShift Container Platform 4.16, cluster administrators can optionally enable monitoring for user-defined projects.

Use these procedures if the following issues occur:

- Your own metrics are unavailable.
- Prometheus is consuming a lot of disk space.
- The **KubePersistentVolumeFillingUp** alert is firing for Prometheus.

7.11.1. Investigating why user-defined project 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 projects.
- You have created a **ServiceMonitor** resource.

Procedure

1. Ensure that your project and resources are not excluded from user workload monitoring. The following examples use the **ns1** project.

   a. Verify that the project *does not* have the **openshift.io/user-monitoring=false** label attached:

   ```
   $ oc get namespace ns1 --show-labels | grep 'openshift.io/user-monitoring=false'
   ```

   NOTE

   The default label set for user workload projects is **openshift.io/user-monitoring=true**. However, the label is not visible unless you manually apply it.

   b. Verify that the **ServiceMonitor** and **PodMonitor** resources *do not* have the **openshift.io/user-monitoring=false** label attached. The following example checks the **prometheus-example-monitor** service monitor.

   ```
   $ oc -n ns1 get servicemonitor prometheus-example-monitor --show-labels | grep
   'openshift.io/user-monitoring=false'
   ```

   c. If the label is attached, remove the label:

Example of removing the label from the project

```
$ oc label namespace ns1 'openshift.io/user-monitoring-'
```

Example of removing the label from the resource

```
$ oc -n ns1 label servicemonitor prometheus-example-monitor 'openshift.io/user-monitoring-'
```

Example output

```
namespace/ns1 unlabeled
```

2. Check that the corresponding labels match in the service and **ServiceMonitor** resource configurations. The following examples use the **prometheus-example-app** service, the **prometheus-example-monitor** service monitor, and the **ns1** project.
   a. Obtain the label defined in the service.

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

Example output

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

   b. Check that the **matchLabels** definition in the **ServiceMonitor** resource configuration matches the label output in the previous step.

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

Example output

```
apiVersion: v1
kind: ServiceMonitor
metadata:
  name: prometheus-example-monitor
  namespace: ns1
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.

3. 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
    ```

4. Review the target status for your endpoint on the **Metrics targets** page in the OpenShift Container Platform web console UI.

    a. Log in to the OpenShift Container Platform web console and navigate to **Observe** → **Targets** in the **Administrator** perspective.

    b. Locate the metrics endpoint in the list, and review the status of the target in the **Status** column.

    c. If the **Status** is **Down**, click the URL for the endpoint to view more information on the **Target Details** page for that metrics target.

5. 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. The affected **prometheus-operator** pod is automatically redeployed.

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

- Enabling monitoring for user-defined projects
- See Specifying how a service is monitored for details on how to create a service monitor or pod monitor
- See Getting detailed information about a metrics target

7.11.2. 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 time series database (TSDB) status using the Prometheus HTTP API** for more information about which labels are creating the most time series data. Doing so requires cluster administrator privileges.
- **Check the number of scrape samples** that are being collected.
- **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** cluster role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. In the **Administrator** perspective, navigate to **Observe → Metrics**.
2. Enter a Prometheus Query Language (PromQL) query in the **Expression** field. The following example queries help to identify high cardinality metrics that might result in high disk space consumption:
   - By running the following query, you can identify the ten jobs that have the highest number of scrape samples:

     ```
     topk(10, max by(namespace, job) (topk by(namespace, job) (1,
     scrape_samples_post_metric_relabeling)))
     ```

   - By running the following query, you can pinpoint time series churn by identifying the ten jobs that have created the most time series data in the last hour:

     ```
     topk(10, sum by(namespace, job) (sum_over_time(scrape_series_added[1h])))
     ```

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. Review the TSDB status using the Prometheus HTTP API by following these steps when logged in as a cluster administrator:
   a. Get the Prometheus API route URL by running the following command:

```
$ HOST=$(oc -n openshift-monitoring get route prometheus-k8s -
ojsonpath='{.status.ingress[].host}')
```

   b. Extract an authentication token by running the following command:

```
$ TOKEN=$(oc whoami -t)
```

   c. Query the TSDB status for Prometheus by running the following command:

```
$ curl -H "Authorization: Bearer $TOKEN" -k "https://$HOST/api/v1/status/tsdb"
```

Example output

```
"status": "success","data":{"headStats":{"numSeries":507473,
"numLabelPairs":19832,"chunkCount":946298,"minTime":1712253600010,
"maxTime":1712257935346},"seriesCountByMetricName":
[{"name":"etcd_request_duration_seconds_bucket","value":51840},
{"name":"apiserver_request_sli_duration_seconds_bucket","value":47718},
...
```

Additional resources

- Setting a scrape sample limit for user-defined projects

7.11.3. Resolving the KubePersistentVolumeFillingUp alert firing for Prometheus

As a cluster administrator, you can resolve the **KubePersistentVolumeFillingUp** alert being triggered for Prometheus.

The critical alert fires when a persistent volume (PV) claimed by a **prometheus-k8s-*** pod in the **openshift-monitoring** project has less than 3% total space remaining. This can cause Prometheus to function abnormally.

NOTE

There are two **KubePersistentVolumeFillingUp** alerts:

- **Critical alert**: The alert with the **severity="critical"** label is triggered when the mounted PV has less than 3% total space remaining.
- **Warning alert**: The alert with the **severity="warning"** label is triggered when the mounted PV has less than 15% total space remaining and is expected to fill up within four days.

To address this issue, you can remove Prometheus time-series database (TSDB) blocks to create more space for the PV.

Prerequisites

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

Procedure

1. List the size of all TSDB blocks, sorted from oldest to newest, by running the following command:

```
$ oc debug <prometheus_k8s_pod_name> -n openshift-monitoring \ ①
-c prometheus --image=$(oc get po -n openshift-monitoring <prometheus_k8s_pod_name> \
②
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}') \
-- sh -c 'cd /prometheus/;du -hs $(ls -dtr */ | grep -Eo "[0-9|A-Z]{26}")'
```

① ② Replace **<prometheus_k8s_pod_name>** with the pod mentioned in the **KubePersistentVolumeFillingUp** alert description.

Example output

```
308M    01HVKMPKQWZYWS8WVDAYQHNMW6
52M     01HVK64DTDA81799TBR9QDECEZ
102M    01HVK64DS7TRZRWF2756KHST5X
140M    01HVJS59K11FBVAPVY57K88Z11
90M     01HVH2A5Z58SKT810EM6B9AT50
152M    01HV8ZDVQMX41MKCN84S32RRZ1
354M    01HV6Q2N26BK63G4RYTST71FBF
156M    01HV664H9J9Z1FTZD73RD1563E
216M    01HTHXB60A7F239HN7S2TENPNS
104M    01HTHMGRXGS0WXA3WATRXHR36B
```

2. Identify which and how many blocks could be removed, then remove the blocks. The following example command removes the three oldest Prometheus TSDB blocks from the **prometheus-k8s-0** pod:

```
$ oc debug prometheus-k8s-0 -n openshift-monitoring \
-c prometheus --image=$(oc get po -n openshift-monitoring prometheus-k8s-0 \
```

```
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}') \
-- sh -c 'ls -latr /prometheus/ | egrep -o "[0-9|A-Z]{26}" | head -3 | \
while read BLOCK; do rm -r /prometheus/$BLOCK; done'
```

3. Verify the usage of the mounted PV and ensure there is enough space available by running the following command:

```
$ oc debug <prometheus_k8s_pod_name> -n openshift-monitoring \①
--image=$(oc get po -n openshift-monitoring <prometheus_k8s_pod_name> \②
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}') -- df -h /prometheus/
```

① ② Replace **<prometheus_k8s_pod_name>** with the pod mentioned in the **KubePersistentVolumeFillingUp** alert description.

The following example output shows the mounted PV claimed by the **prometheus-k8s-0** pod that has 63% of space remaining:

Example output

```
Starting pod/prometheus-k8s-0-debug-j82w4 ...
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme0n1p4  40G   15G  40G  37% /prometheus

Removing debug pod ...
```

7.12. DIAGNOSING OPENSHIFT CLI (OC) ISSUES

7.12.1. 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 7.4. 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.

Log level	Description
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.

7.12.2. Specifying OpenShift CLI (oc) log levels

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

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.

Prerequisites

- Install the OpenShift CLI (**oc**).

Procedure

- Specify the **oc** log level when running an **oc** command:

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

  where:

  <command>

  Specifies the command you are running.

  <log_level>

  Specifies the log level to apply to the command.

- To obtain the current user's session token, run the following command:

  ```
  $ oc whoami -t
  ```

  #### Example output

  ```
  sha256~RCV3Qcn7H-OEfqCGVI0CvnZ6...
  ```