Configuring cluster logging in OpenShift Container Platform
OpenShift Container Platform 4.5 Logging

Configuring cluster logging in OpenShift Container Platform
Abstract

This document provides instructions for installing, configuring, and using cluster logging, which aggregates logs for a range of OpenShift Container Platform services.
# Table of Contents

## CHAPTER 1. UNDERSTANDING CLUSTER LOGGING

1.1. ABOUT DEPLOYING CLUSTER LOGGING
   1.1.1. About cluster logging components
   1.1.2. About the logging collector
   1.1.3. About the log store
   1.1.4. About logging visualization
   1.1.5. About event routing
   1.1.6. About log forwarding

## CHAPTER 2. INSTALLING CLUSTER LOGGING

2.1. INSTALLING CLUSTER LOGGING USING THE WEB CONSOLE
2.2. POST-INSTALLATION TASKS
2.3. INSTALLING CLUSTER LOGGING USING THE CLI
2.4. POST-INSTALLATION TASKS
   2.4.1. Defining Kibana index patterns
   2.4.2. Allowing traffic between projects when network isolation is enabled

## CHAPTER 3. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT

3.1. ABOUT THE CLUSTER LOGGING CUSTOM RESOURCE
   3.1.1. About the ClusterLogging custom resource
3.2. CONFIGURING THE LOGGING COLLECTOR
   3.2.1. About unsupported configurations
   3.2.2. Viewing logging collector pods
   3.2.3. Configure log collector CPU and memory limits
   3.2.4. About logging collector alerts
   3.2.5. Removing unused components if you do not use the default Elasticsearch log store
3.3. CONFIGURING THE LOG STORE
   3.3.1. Forward audit logs to the log store
   3.3.2. Configuring log retention time
   3.3.3. Configuring CPU and memory requests for the log store
   3.3.4. Configuring replication policy for the log store
   3.3.5. Scaling down Elasticsearch pods
   3.3.6. Configuring persistent storage for the log store
   3.3.7. Configuring the log store for emptyDir storage
   3.3.8. Performing an Elasticsearch rolling cluster restart
   3.3.9. Exposing the log store service as a route
3.4. CONFIGURING THE LOG VISUALIZER
   3.4.1. Configuring CPU and memory limits
   3.4.2. Scaling redundancy for the log visualizer nodes
   3.4.3. Using tolerations to control the log visualizer pod placement
3.5. CONFIGURING CLUSTER LOGGING STORAGE
   3.5.1. Storage considerations for cluster logging and OpenShift Container Platform
   3.5.2. Additional resources
3.6. CONFIGURING CPU AND MEMORY LIMITS FOR CLUSTER LOGGING COMPONENTS
   3.6.1. Configuring CPU and memory limits
3.7. USING TOLERATIONS TO CONTROL CLUSTER LOGGING POD PLACEMENT
   3.7.1. Using tolerations to control the log store pod placement
   3.7.2. Using tolerations to control the log visualizer pod placement
   3.7.3. Using tolerations to control the log collector pod placement
   3.7.4. Additional resources
3.8. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS
3.8.1. Moving the cluster logging resources
3.9. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD
  3.9.1. Configuring systemd-journald for cluster logging
3.10. CONFIGURING THE LOG CURATOR
  3.10.1. Configuring the Curator schedule
  3.10.2. Configuring Curator index deletion
3.11. MAINTENANCE AND SUPPORT
  3.11.1. About unsupported configurations
  3.11.2. Unsupported configurations
  3.11.3. Support policy for unmanaged Operators

CHAPTER 4. VIEWING LOGS FOR A RESOURCE
  4.1. VIEWING RESOURCE LOGS

CHAPTER 5. VIEWING CLUSTER LOGS BY USING KIBANA
  5.1. DEFINING KIBANA INDEX PATTERNS
  5.2. VIEWING CLUSTER LOGS IN KIBANA

CHAPTER 6. FORWARDING LOGS TO THIRD PARTY SYSTEMS
  6.1. FORWARDING LOGS USING THE FLUENTD FORWARD PROTOCOL
  6.2. FORWARDING LOGS USING THE SYSLOG PROTOCOL
  6.3. FORWARDING LOGS USING THE LOG FORWARDING API
    6.3.1. Understanding the Log Forwarding API
    6.3.2. Enabling the Log Forwarding API
    6.3.3. Configuring log forwarding using the Log Forwarding API
      6.3.3.1. Example log forwarding custom resources
    6.3.4. Disabling the Log Forwarding API

CHAPTER 7. COLLECTING AND STORING KUBERNETES EVENTS
  7.1. DEPLOYING AND CONFIGURING THE EVENT ROUTER

CHAPTER 8. UPDATING CLUSTER LOGGING
  8.1. UPDATING CLUSTER LOGGING
    8.1.1. Post-update tasks
  8.2. DEFINING KIBANA INDEX PATTERNS

CHAPTER 9. TROUBLESHOOTING CLUSTER LOGGING
  9.1. VIEWING CLUSTER LOGGING STATUS
    9.1.1. Viewing the status of the Cluster Logging Operator
      9.1.1.1. Example condition messages
    9.1.2. Viewing the status of cluster logging components
  9.2. VIEWING THE STATUS OF THE LOG STORE
    9.2.1. Viewing the status of the log store
      9.2.1.1. Example condition messages
    9.2.2. Viewing the status of the log store components
  9.3. UNDERSTANDING CLUSTER LOGGING ALERTS
    9.3.1. Viewing logging collector alerts
    9.3.2. About logging collector alerts
    9.3.3. About Elasticsearch alerting rules
  9.4. TROUBLESHOOTING THE LOG CURATOR
    9.4.1. Troubleshooting log curation
  9.5. COLLECTING LOGGING DATA FOR RED HAT SUPPORT
    9.5.1. About the must-gather tool
    9.5.2. Prerequisites
9.5.3. Collecting cluster logging data

CHAPTER 10. UNINSTALLING CLUSTER LOGGING .................................................. 117
10.1. UNINSTALLING CLUSTER LOGGING FROM OPENShift CONTAINER PLATFORM 117

CHAPTER 11. EXPORTED FIELDS ........................................................................... 119
11.1. DEFAULT EXPORTED FIELDS
   Top Level Fields
   collectd Fields
   collectd.processes Fields
   collectd.processes.ps_disk_ops Fields
   collectd.processes.ps_cputime Fields
   collectd.processes.ps_count Fields
   collectd.processes.ps_pagefaults Fields
   collectd.processes.ps_disk_octets Fields
   collectd.disk Fields
   collectd.disk.disk_merged Fields
   collectd.disk.disk_octets Fields
   collectd.disk.disk_time Fields
   collectd.disk.disk_ops Fields
   collectd.disk.disk_io_time Fields
   collectd.interface Fields
   collectd.interface.if_octets Fields
   collectd.interface.if_packets Fields
   collectd.interface.if_errors Fields
   collectd.interface.if_dropped Fields
   collectd.virt Fields
   collectd.virt.if_octets Fields
   collectd.virt.if_packets Fields
   collectd.virt.if_errors Fields
   collectd.virt.if_dropped Fields
   collectd.virt.disk_ops Fields
   collectd.virt.disk_octets Fields
   collectd.CPU Fields
   collectd.df Fields
   collectd.entropy Fields
   collectd.memory Fields
   collectd.swap Fields
   collectd.load Fields
   collectd.load.load Fields
   collectd.aggregation Fields
   collectd.statsd Fields
   collectd.postgresql Fields
11.2. SYSTEMD EXPORTED FIELDS
   systemd.k Fields
   systemd.t Fields
   systemd.u Fields
11.3. KUBERNETES EXPORTED FIELDS
   kubernetes.labels Fields
   kubernetes.annotations Fields
11.4. CONTAINER EXPORTED FIELDS
   pipeline_metadata.collector Fields
   pipeline_metadata.normalizer Fields
11.5. OVIRT EXPORTED FIELDS
ovirt.engine Fields 138
11.6. AUSHAPE EXPORTED FIELDS
aushape.data Fields 139
11.7. TLOG EXPORTED FIELDS 140
CHAPTER 1. UNDERSTANDING CLUSTER LOGGING

As a cluster administrator, you can deploy cluster logging to aggregate all the logs from your OpenShift Container Platform cluster, such as node system audit logs, application container logs, and infrastructure logs. Cluster logging aggregates these logs from throughout your cluster and stores them in a default log store. You can use the Kibana web console to visualize log data.

Cluster logging aggregates the following types of logs:

- **application** - Container logs generated by user applications running in the cluster, except infrastructure container applications.
- **infrastructure** - Logs generated by infrastructure components running in the cluster and OpenShift Container Platform nodes, such as journal logs. Infrastructure components are pods that run in the *openshift*, *kube*, or *default* projects.
- **audit** - Logs generated by the node audit system (auditd), which are stored in the `/var/log/audit/audit.log` file, and the audit logs from the Kubernetes apiserver and the OpenShift apiserver.

NOTE

Because the internal OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs, audit logs are not stored in the internal Elasticsearch instance by default. If you want to send the audit logs to the internal log store, for example to view the audit logs in Kibana, you must use the Log Forwarding API as described in Forward audit logs to the log store.

1.1. ABOUT DEPLOYING CLUSTER LOGGING

OpenShift Container Platform cluster administrators can deploy cluster logging using the OpenShift Container Platform web console or CLI to install the Elasticsearch Operator and Cluster Logging Operator. When the operators are installed, you create a **ClusterLogging** custom resource (CR) to schedule cluster logging pods and other resources necessary to support cluster logging. The operators are responsible for deploying, upgrading, and maintaining cluster logging.

The **ClusterLogging** CR defines a complete cluster logging environment that includes all the components of the logging stack to collect, store and visualize logs. The Cluster Logging Operator watches the Cluster Logging CR and adjusts the logging deployment accordingly.

Administrators and application developers can view the logs of the projects for which they have view access.

For information, see Configuring the log collector.

1.1.1. About cluster logging components

The cluster logging components include a collector deployed to each node in the OpenShift Container Platform cluster that collects all node and container logs and writes them to a log store. You can use a centralized web UI to create rich visualizations and dashboards with the aggregated data.

The major components of cluster logging are:

- **collection** - This is the component that collects logs from the cluster, formats them, and forwards them to the log store. The current implementation is Fluentd.
• log store - This is where the logs are stored. The default implementation is Elasticsearch. You can use the default Elasticsearch log store or forward logs to external log stores. The default log store is optimized and tested for short-term storage.

• visualization - This is the UI component you can use to view logs, graphs, charts, and so forth. The current implementation is Kibana.

This document might refer to log store or Elasticsearch, visualization or Kibana, collection or Fluentd, interchangeably, except where noted.

1.1.2. About the logging collector

OpenShift Container Platform uses Fluentd to collect container and node logs.

By default, the log collector uses the following sources:

• journald for all system logs

• /var/log/containers/*.log for all container logs

The logging collector is deployed as a daemon set that deploys pods to each OpenShift Container Platform node. System and infrastructure logs are generated by journald log messages from the operating system, the container runtime, and OpenShift Container Platform. Application logs are generated by the CRI–O container engine. Fluentd collects the logs from these sources and forwards them internally or externally as you configure in OpenShift Container Platform.

The container runtimes provide minimal information to identify the source of log messages: project, pod name, and container id. This is not sufficient to uniquely identify the source of the logs. If a pod with a given name and project is deleted before the log collector begins processing its logs, information from the API server, such as labels and annotations, might not be available. There might not be a way to distinguish the log messages from a similarly named pod and project or trace the logs to their source. This limitation means log collection and normalization is considered best effort.

**IMPORTANT**

The available container runtimes provide minimal information to identify the source of log messages and do not guarantee unique individual log messages or that these messages can be traced to their source.

For information, see Configuring the log collector.

1.1.3. About the log store

By default, OpenShift Container Platform uses Elasticsearch (ES) to store log data. Optionally, you can use the log forwarding features to forward logs to external log stores using Fluentd protocols, syslog protocols, or the OpenShift Container Platform Log Forwarding API.

The cluster logging Elasticsearch instance is optimized and tested for short term storage, approximately seven days. If you want to retain your logs over a longer term, it is recommended you move the data to a third-party storage system.

Elasticsearch organizes the log data from Fluentd into datastores, or indices, then subdivides each index into multiple pieces called shards, which it spreads across a set of Elasticsearch nodes in an Elasticsearch cluster. You can configure Elasticsearch to make copies of the shards, called replicas, which Elasticsearch also spreads across the Elasticsearch nodes. The **ClusterLogging** custom resource (CR)
allows you to specify how the shards are replicated to provide data redundancy and resilience to failure. You can also specify how long the different types of logs are retained using a retention policy in the ClusterLogging CR.

NOTE
The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

The Cluster Logging Operator and companion Elasticsearch Operator ensure that each Elasticsearch node is deployed using a unique deployment that includes its own storage volume. You can use a ClusterLogging custom resource (CR) to increase the number of Elasticsearch nodes, as needed. Refer to the Elasticsearch documentation for considerations involved in configuring storage.

NOTE
A highly-available Elasticsearch environment requires at least three Elasticsearch nodes, each on a different host.

Role-based access control (RBAC) applied on the Elasticsearch indices enables the controlled access of the logs to the developers. Administrators can access all logs and developers can access only the logs in their projects.

For information, see Configuring the log store.

1.1.4. About logging visualization
OpenShift Container Platform uses Kibana to display the log data collected by Fluentd and indexed by Elasticsearch.

Kibana is a browser-based console interface to query, discover, and visualize your Elasticsearch data through histograms, line graphs, pie charts, and other visualizations.

For information, see Configuring the log visualizer.

1.1.5. About event routing
The Event Router is a pod that watches OpenShift Container Platform events so they can be collected by cluster logging. The Event Router collects events from all projects and writes them to STDOUT. Fluentd collects those events and forwards them into the OpenShift Container Platform Elasticsearch instance. Elasticsearch indexes the events to the infra index.

You must manually deploy the Event Router.

For information, see Collecting and storing Kubernetes events.

1.1.6. About log forwarding
By default, OpenShift Container Platform cluster logging sends logs to the default internal Elasticsearch log store, defined in the ClusterLogging custom resource (CR). If you want to forward logs to other log aggregators, you can use the log forwarding features to send logs to specific endpoints within or outside your cluster.

For information, see Forwarding logs to third party systems.
CHAPTER 2. INSTALLING CLUSTER LOGGING

You can install cluster logging by deploying the Elasticsearch and Cluster Logging Operators. The Elasticsearch Operator creates and manages the Elasticsearch cluster used by cluster logging. The Cluster Logging Operator creates and manages the components of the logging stack.

The process for deploying cluster logging to OpenShift Container Platform involves:

- Reviewing the cluster logging storage considerations.
- Installing the Elasticsearch Operator and Cluster Logging Operator using the OpenShift Container Platform web console or CLI.

2.1. INSTALLING CLUSTER LOGGING USING THE WEB CONSOLE

You can use the OpenShift Container Platform web console to install the Elasticsearch and Cluster Logging operators.

Prerequisites

- Ensure that you have the necessary persistent storage for Elasticsearch. Note that each Elasticsearch node requires its own storage volume.

NOTE

If you use a local volume for persistent storage, do not use a raw block volume, which is described with `volumeMode: block` in the `LocalVolume` object. Elasticsearch cannot use raw block volumes.

Elasticsearch is a memory-intensive application. By default, OpenShift Container Platform installs three Elasticsearch nodes with memory requests and limits of 16 GB. This initial set of three OpenShift Container Platform nodes might not have enough memory to run Elasticsearch within your cluster. If you experience memory issues that are related to Elasticsearch, add more Elasticsearch nodes to your cluster rather than increasing the memory on existing nodes.

Procedure

To install the Elasticsearch Operator and Cluster Logging Operator using the OpenShift Container Platform web console:

1. Install the Elasticsearch Operator:
   b. Choose Elasticsearch Operator from the list of available Operators, and click Install.
   c. Ensure that the All namespaces on the cluster is selected under Installation Mode.
   d. Ensure that openshift-operators-redhat is selected under Installed Namespace.
      You must specify the openshift-operators-redhat namespace. The openshift-operators namespace might contain Community Operators, which are untrusted and could publish a metric with the same name as an OpenShift Container Platform metric, which would cause conflicts.
   e. Select Enable operator recommended cluster monitoring on this namespace.
This option sets the `openshift.io/cluster-monitoring: "true"` label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the `openshift-operators-redhat` namespace.

f. Select 4.5 as the **Update Channel**

g. Select an **Approval Strategy**.
   - The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
   - The **Manual** strategy requires a user with appropriate credentials to approve the Operator update.

h. Click **Install**.

i. Verify that the Elasticsearch Operator installed by switching to the Operators → Installed Operators page.

j. Ensure that **Elasticsearch Operator** is listed in all projects with a **Status** of **Succeeded**.

2. Install the Cluster Logging Operator:
   
a. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.

b. Choose **Cluster Logging** from the list of available Operators, and click **Install**.

c. Ensure that the **A specific namespace on the cluster** is selected under **Installation Mode**.

d. Ensure that **Operator recommended namespace** is `openshift-logging` under **Installed Namespace**.

e. Select **Enable operator recommended cluster monitoring on this namespace**
   This option sets the `openshift.io/cluster-monitoring: "true"` label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the `openshift-logging` namespace.

f. Select 4.5 as the **Update Channel**

g. Select an **Approval Strategy**.
   - The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
   - The **Manual** strategy requires a user with appropriate credentials to approve the Operator update.

h. Click **Install**.

i. Verify that the Cluster Logging Operator installed by switching to the Operators → Installed Operators page.

j. Ensure that **Cluster Logging** is listed in the `openshift-logging` project with a **Status** of **Succeeded**.
   If the Operator does not appear as installed, to troubleshoot further:
   - Switch to the Operators → Installed Operators page and inspect the **Status** column for any errors or failures.
• Switch to the Workloads → Pods page and check the logs in any pods in the openshift-logging project that are reporting issues.

3. Create a cluster logging instance:
   a. Switch to the Administration → Custom Resource Definitions page.
   c. On the Custom Resource Definition Overview page, select View Instances from the Actions menu.
   d. On the ClusterLoggings page, click Create ClusterLogging.
      You might have to refresh the page to load the data.
   e. In the YAML field, replace the code with the following:

   ```yaml
   apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging"
spec:
  managementState: "Managed" 2
  logStore:
    type: "elasticsearch" 3
    retentionPolicy: 4
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
  elasticsearch:
    nodeCount: 3 5
  storage:
    storageClassName: "<storage-class-name>" 6
    size: 200G
  resources:
    limits:
      memory: 256Mi
    requests:
      memory: 8Gi
  proxy:
    resources:
      limits:
        memory: 256Mi
      requests:
        memory: 256Mi
   ```

   **NOTE**

   This default cluster logging configuration should support a wide array of environments. Review the topics on tuning and configuring the cluster logging components for information on modifications you can make to your cluster logging cluster.
The name must be instance.

The cluster logging management state. In some cases, if you change the cluster logging defaults, you must set this to Unmanaged. However, an unmanaged deployment does not receive updates until the cluster logging is placed back into a managed state.

Settings for configuring Elasticsearch. Using the CR, you can configure shard replication policy and persistent storage.

Specify the length of time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, 7d for seven days. Logs older than the maxAge are deleted. You must specify a retention policy for each log source or the Elasticsearch indices will not be created for that source.

Specify the number of Elasticsearch nodes. See the note that follows this list.

Enter the name of an existing storage class for Elasticsearch storage. For best performance, specify a storage class that allocates block storage. If you do not specify a storage class, OpenShift Container Platform deploys cluster logging with ephemeral storage only.

Enter the name of an existing storage class for Elasticsearch storage. For best performance, specify a storage class that allocates block storage. If you do not specify a storage class, OpenShift Container Platform deploys cluster logging with ephemeral storage only.

Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are 16G for the memory request and 1 for the CPU request.

Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are 256Mi for the memory request and 100m for the CPU request.

Settings for configuring Kibana. Using the CR, you can scale Kibana for redundancy and configure the CPU and memory for your Kibana nodes. For more information, see Configuring the log visualizer.
Settings for configuring the Curator schedule. Curator is used to remove data that is in the Elasticsearch index format prior to OpenShift Container Platform 4.5 and will be

Settings for configuring Fluentd. Using the CR, you can configure Fluentd CPU and memory limits. For more information, see Configuring Fluentd.

NOTE

The maximum number of Elasticsearch master nodes is three. If you specify a nodeCount greater than 3, OpenShift Container Platform creates three Elasticsearch nodes that are Master-eligible nodes, with the master, client, and data roles. The additional Elasticsearch nodes are created as Data-only nodes, using client and data roles. Master nodes perform cluster-wide actions such as creating or deleting an index, shard allocation, and tracking nodes. Data nodes hold the shards and perform data-related operations such as CRUD, search, and aggregations. Data-related operations are I/O-, memory-, and CPU-intensive. It is important to monitor these resources and to add more Data nodes if the current nodes are overloaded.

For example, if nodeCount=4, the following nodes are created:

```
$ oc get deployment
```

**Example output**

```
cluster-logging-operator 1/1 1 1 18h
elasticsearch-cd-x6kdekli-1 0/1 1 0 6m54s
elasticsearch-cdm-x6kdekli-1 1/1 1 1 18h
elasticsearch-cdm-x6kdekli-2 0/1 1 0 6m49s
elasticsearch-cdm-x6kdekli-3 0/1 1 0 6m44s
```

The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

f. Click Create. This creates the Cluster Logging components, the Elasticsearch custom resource and components, and the Kibana interface.

4. Verify the install:

a. Switch to the Workloads → Pods page.

b. Select the openshift-logging project.

You should see several pods for cluster logging, Elasticsearch, Fluentd, and Kibana similar to the following list:

- cluster-logging-operator-cb795f8dc-xkckc
- elasticsearch-cdm-b3nqzchd-1-5c6797-67kfz
- elasticsearch-cdm-b3nqzchd-2-6657f4-wtprv
- elasticsearch-cdm-b3nqzchd-3-588c65-clg7g
- fluentd-2c7dg
2.2. POST-INSTALLATION TASKS

If you plan to use Kibana, you must manually create your Kibana index patterns and visualizations to explore and visualize data in Kibana.

If your cluster network provider enforces network isolation, allow network traffic between the projects that contain the OpenShift Logging operators.

2.3. INSTALLING CLUSTER LOGGING USING THE CLI

You can use the OpenShift Container Platform CLI to install the Elasticsearch and Cluster Logging operators.

Prerequisites

- Ensure that you have the necessary persistent storage for Elasticsearch. Note that each Elasticsearch node requires its own storage volume.

  **NOTE**

  If you use a local volume for persistent storage, do not use a raw block volume, which is described with `volumeMode: block` in the `LocalVolume` object. Elasticsearch cannot use raw block volumes.

  Elasticsearch is a memory-intensive application. By default, OpenShift Container Platform installs three Elasticsearch nodes with memory requests and limits of 16 GB. This initial set of three OpenShift Container Platform nodes might not have enough memory to run Elasticsearch within your cluster. If you experience memory issues that are related to Elasticsearch, add more Elasticsearch nodes to your cluster rather than increasing the memory on existing nodes.

Procedure

To install the Elasticsearch Operator and Cluster Logging Operator using the CLI:

1. Create a Namespace for the Elasticsearch Operator.
   a. Create a Namespace object YAML file (for example, `eo-namespace.yaml`) for the Elasticsearch Operator:
You must specify the `openshift-operators-redhat` Namespace. To prevent possible conflicts with metrics, you should configure the Prometheus Cluster Monitoring stack to scrape metrics from the `openshift-operators-redhat` Namespace and not the `openshift-operators` Namespace. The `openshift-operators` Namespace might contain Community Operators, which are untrusted and could publish a metric with the same name as an OpenShift Container Platform metric, which would cause conflicts.

You must specify this label as shown to ensure that cluster monitoring scrapes the `openshift-operators-redhat` Namespace.

b. Create the Namespace:

```console
$ oc create -f <file-name>.yaml
```

For example:

```console
$ oc create -f eo-namespace.yaml
```

2. Create a Namespace for the Cluster Logging Operator:

   a. Create a Namespace object YAML file (for example, `clo-namespace.yaml`) for the Cluster Logging Operator:

       ```yaml
       apiVersion: v1
       kind: Namespace
       metadata:
         name: openshift-logging
       annotations:
         openshift.io/node-selector: ""
       labels:
         openshift.io/cluster-monitoring: "true"
       ```

   b. Create the Namespace:

       ```console
       $ oc create -f <file-name>.yaml
       ```

       For example:

       ```console
       $ oc create -f clo-namespace.yaml
       ```

3. Install the Elasticsearch Operator by creating the following objects:

   a. Create an Operator Group object YAML file (for example, `eo-og.yaml`) for the Elasticsearch operator:

       ```yaml
       apiVersion: v1
       kind: Namespace
       metadata:
         name: openshift-logging
       annotations:
         openshift.io/node-selector: ""
       labels:
         openshift.io/cluster-monitoring: "true"
       ```

       ```console
       $ oc create -f eo-namespace.yaml
       ```

       For example:

       ```console
       $ oc create -f eo-namespace.yaml
       ```
You must specify the `openshift-operators-redhat` Namespace.

b. Create an Operator Group object:

```bash
$ oc create -f <file-name>.yaml
```

For example:

```bash
$ oc create -f eo-og.yaml
```

c. Create a Subscription object YAML file (for example, `eo-sub.yaml`) to subscribe a Namespace to the Elasticsearch Operator.

**Example Subscription**

```yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: "elasticsearch-operator"
  namespace: "openshift-operators-redhat"  
spec:
  channel: "4.5"  
  installPlanApproval: "Automatic"
  source: "redhat-operators"
  sourceNamespace: "openshift-marketplace"
  name: "elasticsearch-operator"
```

1. You must specify the `openshift-operators-redhat` Namespace.
2. Specify 4.5 as the channel.
3. Specify `redhat-operators`. If your OpenShift Container Platform cluster is installed on a restricted network, also known as a disconnected cluster, specify the name of the CatalogSource object created when you configured the Operator Lifecycle Manager (OLM).

d. Create the Subscription object:

```bash
$ oc create -f <file-name>.yaml
```

For example:

```bash
$ oc create -f eo-sub.yaml
```
The Elasticsearch Operator is installed to the `openshift-operators-redhat` Namespace and copied to each project in the cluster.

e. Verify the Operator installation:

```bash
$ oc get csv --all-namespaces
```

**Example output**

```
<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>kube-node-lease</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>kube-public</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>kube-system</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>openshift-apiserver-operator</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>openshift-apiserver</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>openshift-authentication-operator</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>openshift-authentication</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>openshift-logging</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>openshift-logging</td>
<td>elasticsearch-operator.4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

There should be an Elasticsearch Operator in each Namespace. The version number might be different than shown.

4. Install the Cluster Logging Operator by creating the following objects:

a. Create an OperatorGroup object YAML file (for example, `clo-og.yaml`) for the Cluster Logging Operator:

```yaml
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: cluster-logging
spec:
  targetNamespaces:
    - openshift-logging
```

You must specify the `openshift-logging` namespace.

b. Create the OperatorGroup object:

```bash
$ oc create -f <file-name>.yaml
```
For example:

```bash
$ oc create -f clo-og.yaml
```

c. Create a Subscription object YAML file (for example, `clo-sub.yaml`) to subscribe a Namespace to the Cluster Logging Operator.

```yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: cluster-logging
  namespace: openshift-logging
spec:
  channel: "4.5"
  name: cluster-logging
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

1. You must specify the `openshift-logging` Namespace.

2. Specify `4.5` as the channel.

3. Specify `redhat-operators`. If your OpenShift Container Platform cluster is installed on a restricted network, also known as a disconnected cluster, specify the name of the `CatalogSource` object you created when you configured the Operator Lifecycle Manager (OLM).

d. Create the Subscription object:

```bash
$ oc create -f <file-name>.yaml
```

For example:

```bash
$ oc create -f clo-sub.yaml
```

The Cluster Logging Operator is installed to the `openshift-logging` Namespace.

e. Verify the Operator installation.

There should be a Cluster Logging Operator in the `openshift-logging` Namespace. The Version number might be different than shown.

```bash
$ oc get csv -n openshift-logging
```

**Example output**

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>openshift-logging</td>
<td>clusterlogging.4.5.0-202007012112.p0</td>
<td></td>
</tr>
<tr>
<td>Cluster Logging</td>
<td>4.5.0-202007012112.p0</td>
<td>Succeeded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. Create a Cluster Logging instance:
   a. Create an instance object YAML file (for example, `clo-instance.yaml`) for the Cluster Logging Operator:

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
     namespace: "openshift-logging"
   spec:
     managementState: "Managed"
     logStore:
       type: "elasticsearch"
       retentionPolicy:
         application:
           maxAge: 1d
         infra:
           maxAge: 7d
         audit:
           maxAge: 7d
       elasticsearch:
         nodeCount: 3
     storage:
       storageClassName: "<storage-class-name>"
       size: 200G
     resources:
       requests:
         memory: "8Gi"
       proxy:
         resources:
           limits:
             memory: 256Mi
           requests:
             memory: 256Mi
       redundancyPolicy: "SingleRedundancy"
     visualization:
       type: "kibana"
       kibana:
         replicas: 1
     curation:
       type: "curator"
       curator:
         schedule: "30 3 * * *"
   ```

   **NOTE**
   This default Cluster Logging configuration should support a wide array of environments. Review the topics on tuning and configuring the Cluster Logging components for information on modifications you can make to your Cluster Logging cluster.
The name must be instance.

The cluster logging management state. In some cases, if you change the cluster logging defaults, you must set this to Unmanaged. However, an unmanaged deployment does not receive updates until cluster logging is placed back into a managed state. Placing a deployment back into a managed state might revert any modifications you made.

Settings for configuring Elasticsearch. Using the custom resource (CR), you can configure shard replication policy and persistent storage.

Specify the length of time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, 7d for seven days. Logs older than the maxAge are deleted. You must specify a retention policy for each log source or the Elasticsearch indices will not be created for that source.

Specify the number of Elasticsearch nodes. See the note that follows this list.

Enter the name of an existing storage class for Elasticsearch storage. For best performance, specify a storage class that allocates block storage. If you do not specify a storage class, OpenShift Container Platform deploys cluster logging with ephemeral storage only.

Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are 16G for the memory request and 1 for the CPU request.

Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are 256Mi for the memory request and 100m for the CPU request.

Settings for configuring Kibana. Using the CR, you can scale Kibana for redundancy and configure the CPU and memory for your Kibana pods. For more information, see Configuring the log visualizer.

Settings for configuring the Curator schedule. Curator is used to remove data that is in the Elasticsearch index format prior to OpenShift Container Platform 4.5 and will be removed in a later release.

Settings for configuring Fluentd. Using the CR, you can configure Fluentd CPU and memory limits. For more information, see Configuring Fluentd.
NOTE

The maximum number of Elasticsearch master nodes is three. If you specify a **nodeCount** greater than 3, OpenShift Container Platform creates three Elasticsearch nodes that are Master-eligible nodes, with the master, client, and data roles. The additional Elasticsearch nodes are created as Data-only nodes, using client and data roles. Master nodes perform cluster-wide actions such as creating or deleting an index, shard allocation, and tracking nodes. Data nodes hold the shards and perform data-related operations such as CRUD, search, and aggregations. Data-related operations are I/O-, memory-, and CPU-intensive. It is important to monitor these resources and to add more Data nodes if the current nodes are overloaded.

For example, if **nodeCount=4**, the following nodes are created:

```bash
$ oc get deployment
```

**Example output**

```
cluster-logging-operator       1/1     1     1     18h
elasticsearch-cd-x6kdekli-1   1/1     1     0     6m54s
elasticsearch-cdm-x6kdekli-1  1/1     1     1     18h
elasticsearch-cdm-x6kdekli-2  1/1     1     0     6m49s
elasticsearch-cdm-x6kdekli-3  1/1     1     0     6m44s
```

The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

b. Create the instance:

```bash
$ oc create -f <file-name>.yaml
```

For example:

```bash
$ oc create -f clo-instance.yaml
```

This creates the Cluster Logging components, the Elasticsearch custom resource and components, and the Kibana interface.

6. Verify the installation by listing the pods in the **openshift-logging** project.
You should see several pods for Cluster Logging, Elasticsearch, Fluentd, and Kibana similar to the following list:

```bash
$ oc get pods -n openshift-logging
```

**Example output**

```
NAME                                            READY   STATUS     RESTARTS   AGE
cluster-logging-operator-66f77ffccbspzbg         1/1     Running   0          7m
elasticsearch-cdm-ftuhduuw-1-ff4b9566-q6bhp      2/2     Running   0          2m40s
elasticsearch-cdm-ftuhduuw-2-7b4994dbfcre2gc     2/2     Running   0          2m36s
elasticsearch-cdm-ftuhduuw-3-84b5ff7ff8-gqnm2    2/2     Running   0          2m4s
fluentd-587vb                                   1/1     Running   0          2m26s
```
2.4. POST-INSTALLATION TASKS

If you plan to use Kibana, you must manually create your Kibana index patterns and visualizations to explore and visualize data in Kibana.

If your cluster network provider enforces network isolation, allow network traffic between the projects that contain the OpenShift Logging operators.

2.4.1. Defining Kibana index patterns

An index pattern defines the Elasticsearch indices that you want to visualize. To explore and visualize data in Kibana, you must create an index pattern.

Prerequisites

- A user must have the cluster-admin role, the cluster-reader role, or both roles to view the infra and audit indices in Kibana. The default kubeadmin user has proper permissions to view these indices.
- If you can view the pods and logs in the default, kube- and openshift- projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

  $ oc auth can-i get pods/log -n <project>

Example output

  yes

NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the default output for audit logs.

Elasticsearch documents must be indexed before you can create index patterns. This is done automatically, but it might take a few minutes in a new or updated cluster.

Procedure

To define index patterns and create visualizations in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher and select Logging.
2. Create your Kibana index patterns by clicking Management → Index Patterns → Create index pattern:

- Each user must manually create index patterns when logging into Kibana the first time in order to see logs for their projects. Users must create an index pattern named app and use the @timestamp time field to view their container logs.

- Each admin user must create index patterns when logged into Kibana the first time for the app, infra, and audit indices using the @timestamp time field.

3. Create Kibana Visualizations from the new index patterns.

2.4.2. Allowing traffic between projects when network isolation is enabled

Your cluster network provider might enforce network isolation. If so, you must allow network traffic between the projects that contain the operators deployed by OpenShift Logging.

Network isolation blocks network traffic between pods or services that are in different projects. OpenShift Logging installs the OpenShift Elasticsearch Operator in the openshift-operators-redhat project and the Red Hat OpenShift Logging Operator in the openshift-logging project. Therefore, you must allow traffic between these two projects.

OpenShift Container Platform offers two supported choices for the default Container Network Interface (CNI) network provider, OpenShift SDN and OVN-Kubernetes. These two providers implement various network isolation policies.

OpenShift SDN has three modes:

network policy
This is the default mode. If no policy is defined, it allows all traffic. However, if a user defines a policy, they typically start by denying all traffic and then adding exceptions. This process might break applications that are running in different projects. Therefore, explicitly configure the policy to allow traffic to egress from one logging-related project to the other.

multitenant
This mode enforces network isolation. You must join the two logging-related projects to allow traffic between them.

subnet
This mode allows all traffic. It does not enforce network isolation. No action is needed.

OVN-Kubernetes always uses a network policy. Therefore, as with OpenShift SDN, you must configure the policy to allow traffic to egress from one logging-related project to the other.

Procedure

- If you are using OpenShift SDN in multitenant mode, join the two projects. For example:

  $ oc adm pod-network join-projects --to=openshift-operators-redhat openshift-logging

- Otherwise, for OpenShift SDN in network policy mode and OVN-Kubernetes, perform the following actions:

  a. Set a label on the openshift-operators-redhat namespace. For example:

  $ oc label namespace openshift-operators-redhat project=openshift-operators-redhat
b. Create a network policy object in the `openshift-logging` namespace that allows ingress from the `openshift-operators-redhat` project to the `openshift-logging` project. For example:

```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-openshift-operators-redhat
spec:
  ingress:
    - from:
      - namespaceSelector:
        matchLabels:
          project: openshift-operators-redhat
```

Additional resources

- About network policy
- About the OpenShift SDN default CNI network provider
- About the OVN-Kubernetes default Container Network Interface (CNI) network provider
3.1. ABOUT THE CLUSTER LOGGING CUSTOM RESOURCE

To configure OpenShift Container Platform cluster logging, you customize the `ClusterLogging` custom resource (CR).

3.1.1. About the ClusterLogging custom resource

To make changes to your cluster logging environment, create and modify the `ClusterLogging` custom resource (CR).

Instructions for creating or modifying a CR are provided in this documentation as appropriate.

The following is an example of a typical custom resource for cluster logging.

**Sample ClusterLogging custom resource (CR)**

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging" 2
spec:
  managementState: "Managed" 3
logStore:
  type: "elasticsearch" 4
  retentionPolicy:
    application:
      maxAge: 1d
    infra:
      maxAge: 7d
    audit:
      maxAge: 7d
  elasticsearch:
    nodeCount: 3
  resources:
    limits:
      memory: 16Gi
    requests:
      cpu: 500m
      memory: 16Gi
  storage:
    storageClassName: "gp2"
    size: "200G"
    redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana"
  kibana:
    resources:
      limits:
        memory: 736Mi
```
The CR name must be **instance**.

The CR must be installed to the **openshift-logging** namespace.

The Cluster Logging Operator management state. When set to **unmanaged** the operator is in an unsupported state and will not get updates.

Settings for the log store, including retention policy, the number of nodes, the resource requests and limits, and the storage class.

Settings for the visualizer, including the resource requests and limits, and the number of pod replicas.

Settings for curation, including the resource requests and limits, and curation schedule.

Settings for the log collector, including the resource requests and limits.

### 3.2. CONFIGURING THE LOGGING COLLECTOR

OpenShift Container Platform uses Fluentd to collect operations and application logs from your cluster and enriches the data with Kubernetes pod and project metadata.

You can configure the CPU and memory limits for the log collector and move the log collector pods to specific nodes. All supported modifications to the log collector can be performed though the **spec.collection.log.fluentd** stanza in the **ClusterLogging** custom resource (CR).

### 3.2.1. About unsupported configurations

The supported way of configuring cluster logging is by configuring it using the options described in this
documentation. Do not use other configurations, as they are unsupported. Configuration paradigms might change across OpenShift Container Platform releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this documentation, your changes will disappear because the Elasticsearch Operator and Cluster Logging Operator reconcile any differences. The Operators reverse everything to the defined state by default and by design.

**NOTE**

If you *must* perform configurations not described in the OpenShift Container Platform documentation, you *must* set your Cluster Logging Operator or Elasticsearch Operator to **Unmanaged**. An unmanaged cluster logging environment is **not supported** and does not receive updates until you return cluster logging to **Managed**.

### 3.2.2. Viewing logging collector pods

You can use the `oc get pods --all-namespaces -o wide` command to see the nodes where the Fluentd are deployed.

**Procedure**

Run the following command in the `openshift-logging` project:

```
$ oc get pods --selector component=fluentd -o wide -n openshift-logging
```

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
<th>IP</th>
<th>NODE</th>
<th>NOMINATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>fluentd-8d69v</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>134m</td>
<td>10.130.2.30</td>
<td>master1.example.com</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>fluentd-bd225</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>134m</td>
<td>10.131.1.11</td>
<td>master2.example.com</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>fluentd-cvrzs</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>134m</td>
<td>10.130.0.21</td>
<td>master3.example.com</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>fluentd-gpqg2</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>134m</td>
<td>10.128.2.27</td>
<td>worker1.example.com</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>fluentd-l9j7j</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>134m</td>
<td>10.129.2.31</td>
<td>worker2.example.com</td>
<td>&lt;none&gt;</td>
</tr>
</tbody>
</table>

### 3.2.3. Configure log collector CPU and memory limits

The log collector allows for adjustments to both the CPU and memory limits.

**Procedure**

1. Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

```
$ oc edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
```
Specify the CPU and memory limits and requests as needed. The values shown are the default values.

3.2.4. About logging collector alerts

The following alerts are generated by the logging collector. You can view these alerts in the OpenShift Container Platform web console, on the Alerts page of the Alerting UI.

<table>
<thead>
<tr>
<th>Alert</th>
<th>Message</th>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FluentdErrorsHigh</td>
<td>In the last minute, &lt;value&gt; errors reported by fluentd &lt;instance&gt;.</td>
<td>Fluentd is reporting a higher number of issues than the specified number, default 10.</td>
<td>Critical</td>
</tr>
<tr>
<td>FluentdNodeDown</td>
<td>Prometheus could not scrape fluentd &lt;instance&gt; for more than 10m.</td>
<td>Fluentd is reporting that Prometheus could not scrape a specific Fluentd instance.</td>
<td>Critical</td>
</tr>
<tr>
<td>FluentdQueueLengthBurst</td>
<td>In the last minute, fluentd &lt;instance&gt; buffer queue length increased more than 32. Current value is &lt;value&gt;.</td>
<td>Fluentd is reporting that it is overwhelmed.</td>
<td>Warning</td>
</tr>
<tr>
<td>FluentdQueueLengthInIncreasing</td>
<td>In the last 12h, fluentd &lt;instance&gt; buffer queue length constantly increased more than 1. Current value is &lt;value&gt;.</td>
<td>Fluentd is reporting queue usage issues.</td>
<td>Critical</td>
</tr>
</tbody>
</table>
3.2.5. Removing unused components if you do not use the default Elasticsearch log store

As an administrator, in the rare case that you forward logs to a third-party log store and do not use the default Elasticsearch log store, you can remove several unused components from your logging cluster.

In other words, if you do not use the default Elasticsearch log store, you can remove the internal Elasticsearch logStore, Kibana visualization, and log curation components from the ClusterLogging custom resource (CR). Removing these components is optional but saves resources.

Prerequisites

- Verify that your log forwarder does not send log data to the default internal Elasticsearch cluster. Inspect the ClusterLogForwarder CR YAML file that you used to configure log forwarding. Verify that it does not have an outputRefs element that specifies default. For example:

```yaml
outputRefs:
  - default
```

WARNING

Suppose the ClusterLogForwarder CR forwards log data to the internal Elasticsearch cluster, and you remove the logStore component from the ClusterLogging CR. In that case, the internal Elasticsearch cluster will not be present to store the log data. This absence can cause data loss.

Procedure

1. Edit the ClusterLogging custom resource (CR) in the openshift-logging project:

   ```bash
   $ oc edit ClusterLogging instance
   ```

2. If they are present, remove the logStore, visualization, curation stanzas from the ClusterLogging CR.

3. Preserve the collection stanza of the ClusterLogging CR. The result should look similar to the following example:

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   namespace: "openshift-logging"
   spec:
     managementState: "Managed"
   collection:
     logs:
       type: "fluentd"
     fluentd: {}
   ```
3. Verify that the Fluentd pods are redeployed:

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

Additional resources

- Forwarding logs to third party systems

## 3.3. CONFIGURING THE LOG STORE

OpenShift Container Platform uses Elasticsearch 6 (ES) to store and organize the log data.

You can make modifications to your log store, including:

- storage for your Elasticsearch cluster
- shard replication across data nodes in the cluster, from full replication to no replication
- external access to Elasticsearch data

Elasticsearch is a memory-intensive application. Each Elasticsearch node needs 16G of memory for both memory requests and limits, unless you specify otherwise in the `ClusterLogging` custom resource. The initial set of OpenShift Container Platform nodes might not be large enough to support the Elasticsearch cluster. You must add additional nodes to the OpenShift Container Platform cluster to run with the recommended or higher memory.

Each Elasticsearch node can operate with a lower memory setting, though this is not recommended for production environments.

### 3.3.1. Forward audit logs to the log store

Because the internal OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs, by default audit logs are not stored in the internal Elasticsearch instance.

If you want to send the audit logs to the internal log store, for example to view the audit logs in Kibana, you must use the Log Forwarding API. The Log Fowarding API is currently a Technology Preview feature.

**IMPORTANT**

The internal OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs. We recommend you ensure that the system to which you forward audit logs is compliant with your organizational and governmental regulations and is properly secured. OpenShift Container Platform cluster logging does not comply with those regulations.

Procedure

To use the Log Forwarding API to forward audit logs to the internal Elasticsearch instance:

1. If the Log Forwarding API is not enabled:
   a. Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

```
$ oc edit ClusterLogging instance
```
b. Add the `clusterlogging.openshift.io/logforwardingtechpreview` annotation and set to enabled:

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  annotations:
    clusterlogging.openshift.io/logforwardingtechpreview: enabled
  name: "instance"
  namespace: "openshift-logging"
spec:

  collection:
    logs:
      type: "fluentd"
      fluentd: {}

1. Enables and disables the Log Forwarding API. Set to `enabled` to use log forwarding.
2. The `spec.collection` block must be defined to use Fluentd in the `ClusterLogging` CR.

2. Create a LogForwarding CR YAML file or edit your existing CR:

- Create a CR to send all log types to the internal Elasticsearch instance. You can use the following example without making any changes:

```yaml
apiVersion: logging.openshift.io/v1alpha1
kind: LogForwarding
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
    - name: clo-es
      type: elasticsearch
      endpoint: 'elasticsearch.openshift-logging.svc:9200'
      secret:
        name: fluentd
  pipelines:
    - name: audit-pipeline
      inputSource: logs.audit
      outputRefs:
        - clo-es
    - name: app-pipeline
      inputSource: logs.app
      outputRefs:
        - clo-es
    - name: infra-pipeline
```

1. The `endpoint` field must point to the Elasticsearch instance.
2. The `inputSource` field must be set to the appropriate log source.
3. The `outputRefs` field must include the Elasticsearch instance.
4. The `pipeline` name must correspond to the log type.
The **endpoint** parameter points to the internal Elasticsearch instance.

This parameter sends the audit logs to the specified endpoint.

This parameter sends the application logs to the specified endpoint.

This parameter sends the infrastructure logs to the specified endpoint.

**NOTE**

You must configure a pipeline and output for all three types of logs: application, infrastructure, and audit. If you do not specify a pipeline and output for a log type, those logs are not stored and will be lost.

- If you have an existing **LogForwarding** CR, add an output for the internal Elasticsearch instance and a pipeline to that output for the audit logs. For example:

```yaml
apiVersion: "logging.openshift.io/v1alpha1"
kind: "LogForwarding"
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
    - name: elasticsearch
      type: "elasticsearch"
      endpoint: elasticsearch.openshift-logging.svc:9200
      secret:
        name: fluentd
    - name: elasticsearch-insecure
      type: "elasticsearch"
      endpoint: elasticsearch-insecure.messaging.svc.cluster.local
      insecure: true
    - name: secureforward-offcluster
      type: "forward"
      endpoint: https://secureforward.offcluster.com:24224
      secret:
        name: secureforward
  pipelines:
    - name: container-logs
      inputSource: logs.app
      outputRefs:
        - secureforward-offcluster
    - name: infra-logs
      inputSource: logs.infra
      outputRefs:
        - elasticsearch-insecure
        - name: audit-logs
```
CHAPTER 3. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT

A pipeline for sending the audit logs to the internal Elasticsearch instance.

Additional resources
For more information on the Log Forwarding API, see Forwarding logs using the Log Forwarding API.

3.3.2. Configuring log retention time

You can specify how long the default Elasticsearch log store keeps indices using a separate retention policy for each of the three log sources: infrastructure logs, application logs, and audit logs. The retention policy, which you configure using the maxAge parameter in the Cluster Logging Custom Resource (CR), is considered for the Elasticsearch roll over schedule and determines when Elasticsearch deletes the rolled-over indices.

Elasticsearch rolls over an index, moving the current index and creating a new index, when an index matches any of the following conditions:

- The index is older than the rollover.maxAge value in the Elasticsearch CR.
- The index size is greater than 40 GB × the number of primary shards.
- The index doc count is greater than 40960 KB × the number of primary shards.

Elasticsearch deletes the rolled-over indices are deleted based on the retention policy you configure.

If you do not create a retention policy for any of the log sources, logs are deleted after seven days by default.

**IMPORTANT**

If you do not specify a retention policy for all three log sources, only logs from the sources with a retention policy are stored. For example, if you set a retention policy for the infrastructure and application logs, but do not set a retention policy for audit logs, the audit logs will not be retained and there will be no audit-index in Elasticsearch or Kibana.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

To configure the log retention time:

1. Edit the ClusterLogging CR to add or modify the retentionPolicy parameter:

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
..."ClusterLogging"
spec:
```
Specify the time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, 1d for one day. Logs older than the maxAge are deleted. By default, logs are retained for seven days.

2. You can verify the settings in the Elasticsearch custom resource (CR).
   For example, the Cluster Logging Operator updated the following Elasticsearch CR to configure a retention policy that includes settings to roll over active indices for the infrastructure logs every eight hours and the rolled-ver indices are deleted seven days after rollover. OpenShift Container Platform checks every 15 minutes to determine if the indices need to be rolled over.

For each log source, the retention policy indicates when to delete and rollover logs for that source.

When OpenShift Container Platform deletes the rolled-over indices. This setting is the maxAge you set in the ClusterLogging CR.

The index age for OpenShift Container Platform to consider when rolling over the indices. This value is determined from the maxAge you set in the ClusterLogging CR.
When OpenShift Container Platform checks if the indices should be rolled over. This setting is the default and cannot be changed.

**NOTE**

Modifying the **Elasticsearch CR** is not supported. All changes to the retention policies must be made in the **ClusterLogging CR**.

The Elasticsearch Operator deploys a cron job to roll over indices for each mapping using the defined policy, scheduled using the **pollInterval**.

```
$ oc get cronjobs
```

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SCHEDULE</th>
<th>SUSPEND</th>
<th>ACTIVE</th>
<th>LAST SCHEDULE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>elasticsearch-delete-app</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>27s</td>
</tr>
<tr>
<td>elasticsearch-delete-audit</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>27s</td>
</tr>
<tr>
<td>elasticsearch-delete-infra</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>27s</td>
</tr>
<tr>
<td>elasticsearch-rollover-app</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>27s</td>
</tr>
<tr>
<td>elasticsearch-rollover-audit</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>27s</td>
</tr>
<tr>
<td>elasticsearch-rollover-infra</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>27s</td>
</tr>
</tbody>
</table>

### 3.3.3. Configuring CPU and memory requests for the log store

Each component specification allows for adjustments to both the CPU and memory requests. You should not have to manually adjust these values as the Elasticsearch Operator sets values sufficient for your environment.

**NOTE**

In large-scale clusters, the default memory limit for the Elasticsearch proxy container might not be sufficient, causing the proxy container to be OOMKilled. If you experience this issue, increase the memory requests and limits for the Elasticsearch proxy.

Each Elasticsearch node can operate with a lower memory setting though this is **not** recommended for production deployments. For production use, you should have no less than the default 16Gi allocated to each pod. Preferably you should allocate as much as possible, up to 64Gi per pod.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc edit ClusterLogging instance
```

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
```
Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are 16Gi for the memory request and 1 for the CPU request.

Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are 256Mi for the memory request and 100m for the CPU request.

If you adjust the amount of Elasticsearch memory, you must change both the request value and the limit value.

For example:

```yaml
resources:
  limits:
    cpu: "8"
    memory: "32Gi"
  requests:
    cpu: "8"
    memory: "32Gi"
```

Kubernetes generally adheres the node configuration and does not allow Elasticsearch to use the specified limits. Setting the same value for the requests and limits ensures that Elasticsearch can use the CPU and memory you want, assuming the node has the CPU and memory available.

### 3.3.4. Configuring replication policy for the log store

You can define how Elasticsearch shards are replicated across data nodes in the cluster.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.
Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

   ```bash
   $ oc edit clusterlogging instance
   ```

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   
   spec:
     logStore:
       type: "elasticsearch"
       elasticsearch:
         redundancyPolicy: "SingleRedundancy"
   ```

   Specify a redundancy policy for the shards. The change is applied upon saving the changes.

   - **FullRedundancy.** Elasticsearch fully replicates the primary shards for each index to every data node. This provides the highest safety, but at the cost of the highest amount of disk required and the poorest performance.

   - **MultipleRedundancy.** Elasticsearch fully replicates the primary shards for each index to half of the data nodes. This provides a good tradeoff between safety and performance.

   - **SingleRedundancy.** Elasticsearch makes one copy of the primary shards for each index. Logs are always available and recoverable as long as at least two data nodes exist. Better performance than MultipleRedundancy, when using 5 or more nodes. You cannot apply this policy on deployments of single Elasticsearch node.

   - **ZeroRedundancy.** Elasticsearch does not make copies of the primary shards. Logs might be unavailable or lost in the event a node is down or fails. Use this mode when you are more concerned with performance than safety, or have implemented your own disk/PVC backup/restore strategy.

   **NOTE**

   The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

3.3.5. Scaling down Elasticsearch pods

Reducing the number of Elasticsearch pods in your cluster can result in data loss or Elasticsearch performance degradation.

If you scale down, you should scale down by one pod at a time and allow the cluster to re-balance the shards and replicas. After the Elasticsearch health status returns to **green**, you can scale down by another pod.
NOTE
If your Elasticsearch cluster is set to **ZeroRedundancy**, you should not scale down your Elasticsearch pods.

### 3.3.6. Configuring persistent storage for the log store

Elasticsearch requires persistent storage. The faster the storage, the faster the Elasticsearch performance.

#### WARNING

Using NFS storage as a volume or a persistent volume (or via NAS such as Gluster) is not supported for Elasticsearch storage, as Lucene relies on file system behavior that NFS does not supply. Data corruption and other problems can occur.

### Prerequisites

- Cluster logging and Elasticsearch must be installed.

### Procedure

1. Edit the `ClusterLogging` CR to specify that each data node in the cluster is bound to a Persistent Volume Claim.

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   
   spec:
     logStore:
       type: "elasticsearch"
       elasticsearch:
         nodeCount: 3
         storage:
           storageClassName: "gp2"
           size: "200G"
   
   This example specifies each data node in the cluster is bound to a Persistent Volume Claim that requests "200G" of AWS General Purpose SSD (gp2) storage.

#### NOTE

If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the `LocalVolume` object. Elasticsearch cannot use raw block volumes.
3.3.7. Configuring the log store for emptyDir storage

You can use emptyDir with your log store, which creates an ephemeral deployment in which all of a pod’s data is lost upon restart.

**NOTE**

When using emptyDir, if log storage is restarted or redeployed, you will lose data.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

1. Edit the **ClusterLogging** CR to specify emptyDir:

   ```
   spec:
   logStore:
     type: "elasticsearch"
     elasticsearch:
       nodeCount: 3
       storage: {}
   ```

3.3.8. Performing an Elasticsearch rolling cluster restart

Perform a rolling restart when you change the **elasticsearch** configmap or any of the **elasticsearch-** deployment configurations.

Also, a rolling restart is recommended if the nodes on which an Elasticsearch pod runs requires a reboot.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.
- Install the OpenShift Container Platform **es_util** tool

**Procedure**

To perform a rolling cluster restart:

1. Change to the **openshift-logging** project:

   ```
   $ oc project openshift-logging
   ```

2. Get the names of the Elasticsearch pods:

   ```
   $ oc get pods | grep elasticsearch-
   ```

3. Perform a shard synced flush using the OpenShift Container Platform **es_util** tool to ensure there are no pending synced operations waiting to be written to disk prior to shutting down:

   ```
   $ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query="_flush/synced" -XPOST
   ```
For example:

```bash
$ oc exec -c elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --query="_flush/synced" -XPOST
```

**Example output**

```
{"_shards":{"total":4,"successful":4,"failed":0},".security":
{"total":2,"successful":2,"failed":0},".kibana_1":{"total":2,"successful":2,"failed":0}}
```

4. Prevent shard balancing when purposely bringing down nodes using the OpenShift Container Platform es_util tool:

```bash
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable": "primaries" } }'
```

For example:

```bash
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable": "primaries" } }'
```

**Example output**

```
{"acknowledged":true,"persistent":{"cluster":{"routing":{"allocation":
{"enable":"primaries"}}}},"transient":
```

5. After the command is complete, for each deployment you have for an ES cluster:

a. By default, the OpenShift Container Platform Elasticsearch cluster blocks rollouts to their nodes. Use the following command to allow rollouts and allow the pod to pick up the changes:

```bash
$ oc rollout resume deployment/<deployment-name>
```

For example:

```bash
$ oc rollout resume deployment/elasticsearch-cdm-0-1
```

**Example output**

```
deployment.extensions/elasticsearch-cdm-0-1 resumed
```

A new pod is deployed. After the pod has a ready container, you can move on to the next deployment.

```bash
$ oc get pods | grep elasticsearch-
```

**Example output**
b. After the deployments are complete, reset the pod to disallow rollouts:

$ oc rollout pause deployment/<deployment-name>

For example:

$ oc rollout pause deployment/elasticsearch-cdm-0-1

Example output

deployment.extensions/elasticsearch-cdm-0-1 paused

c. Check that the Elasticsearch cluster is in a green or yellow state:

$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query=_cluster/health?pretty=true

NOTE
If you performed a rollout on the Elasticsearch pod you used in the previous commands, the pod no longer exists and you need a new pod name here.

For example:

$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query=_cluster/health?pretty=true

```
{
  "cluster_name" : "elasticsearch",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 8,
  "active_shards" : 16,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 1,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

1 Make sure this parameter value is green or yellow before proceeding.
6. If you changed the Elasticsearch configuration map, repeat these steps for each Elasticsearch pod.

7. After all the deployments for the cluster have been rolled out, re-enable shard balancing:

```bash
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable": "all" } }
```

For example:

```bash
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable": "all" } }
```

**Example output**

```json
{
  "acknowledged": true,
  "persistent": {},
  "transient": {
    "cluster": {
      "routing": {
        "allocation": {
          "enable": "all"
        }
      }
    }
  }
}
```

### 3.3.9. Exposing the log store service as a route

By default, the log store that is deployed with cluster logging is not accessible from outside the logging cluster. You can enable a route with re-encryption termination for external access to the log store service for those tools that access its data.

Externally, you can access the log store by creating a reencrypt route, your OpenShift Container Platform token and the installed log store CA certificate. Then, access a node that hosts the log store service with a cURL request that contains:

- The **Authorization: Bearer ${token}**
- The Elasticsearch reencrypt route and an Elasticsearch API request.

Internally, you can access the log store service using the log store cluster IP, which you can get by using either of the following commands:

```bash
$ oc get service elasticsearch -o jsonpath={.spec.clusterIP} -n openshift-logging
```

**Example output**

```
172.30.183.229
```
$ oc get service elasticsearch -n openshift-logging

Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>elasticsearch</td>
<td>ClusterIP</td>
<td>172.30.183.229</td>
<td>&lt;none&gt;</td>
<td>9200/TCP</td>
<td>22h</td>
</tr>
</tbody>
</table>

You can check the cluster IP address with a command similar to the following:


Example output

<table>
<thead>
<tr>
<th>% Total</th>
<th>% Received</th>
<th>Xferd</th>
<th>Average Speed</th>
<th>Time</th>
<th>Time</th>
<th>Time</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>29</td>
<td>100</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>108</td>
<td>0</td>
</tr>
</tbody>
</table>

Prerequisites

- Cluster logging and Elasticsearch must be installed.
- You must have access to the project in order to be able to access to the logs.

Procedure

To expose the log store externally:

1. Change to the openshift-logging project:

   $ oc project openshift-logging

2. Extract the CA certificate from the log store and write to the admin-ca file:

   $ oc extract secret/elasticsearch --to=. --keys=admin-ca

Example output

admin-ca

3. Create the route for the log store service as a YAML file:

   a. Create a YAML file with the following:

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: elasticsearch
namespace: openshift-logging
spec:
  host: 
  to:
    kind: Service
```
b. Run the following command to add the log store CA certificate to the route YAML you created:

```
$ cat ./admin-ca | sed -e "s/\n/ /" >> <file-name>.yaml
```

c. Create the route:

```
$ oc create -f <file-name>.yaml
```

**Example output**

```
route.route.openshift.io/elasticsearch created
```

4. Check that the Elasticsearch service is exposed:
   a. Get the token of this service account to be used in the request:

```
$ token=$(oc whoami -t)
```
   b. Set the `elasticsearch` route you created as an environment variable.

```
$ routeES=`oc get route elasticsearch -o jsonpath={.spec.host}`
```
   c. To verify the route was successfully created, run the following command that accesses Elasticsearch through the exposed route:

```
```

The response appears similar to the following:

**Example output**

```
{
  "name": "elasticsearch-cdm-i40ktba0-1",
  "cluster_name": "elasticsearch",
  "cluster_uuid": "0eY-tJzcR3K0dpeMJo-MQ",
  "version": {
    "number": "6.8.1",
    "build_flavor": "oss",
    "build_type": "zip",
    "build_hash": "Unknown",
    "build_date": "Unknown",
    "build_snapshot": true,
  }
}```
3.4. CONFIGURING THE LOG VISUALIZER

OpenShift Container Platform uses Kibana to display the log data collected by cluster logging.

You can scale Kibana for redundancy and configure the CPU and memory for your Kibana nodes.

3.4.1. Configuring CPU and memory limits

The cluster logging components allow for adjustments to both the CPU and memory limits.

Procedure

1. Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

   ```bash
   $ oc edit ClusterLogging instance -n openshift-logging
   ```
Specify the CPU and memory limits and requests for the log store as needed. For Elasticsearch, you must adjust both the request value and the limit value.

Specify the CPU and memory limits and requests for the log visualizer as needed.

Specify the CPU and memory limits and requests for the log curator as needed.

Specify the CPU and memory limits and requests for the log collector as needed.

### 3.4.2. Scaling redundancy for the log visualizer nodes

You can scale the pod that hosts the log visualizer for redundancy.

**Procedure**

1. Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

```
$ oc edit ClusterLogging instance
$ oc edit ClusterLogging instance
```
Specify the number of Kibana nodes.

### 3.4.3. Using tolerations to control the log visualizer pod placement

You can control the node where the log visualizer pod runs and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to the log visualizer pod through the `ClusterLogging` custom resource (CR) and apply taints to a node through the node specification. A taint on a node is a key:value pair that instructs the node to repel all pods that do not tolerate the taint. Using a specific key:value pair that is not on other pods ensures only the Kibana pod can run on that node.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

1. Use the following command to add a taint to a node where you want to schedule the log visualizer pod:

   ```bash
   $ oc adm taint nodes <node-name> <key>=<value>:<effect>
   
   For example:
   
   ```bash
   $ oc adm taint nodes node1 kibana=node:NoExecute
   ```

   This example places a taint on `node1` that has key `kibana`, value `node`, and taint effect `NoExecute`. You must use the `NoExecute` taint effect. `NoExecute` schedules only pods that match the taint and remove existing pods that do not match.

2. Edit the `visualization` section of the `ClusterLogging` CR to configure a toleration for the Kibana pod:

   ```yaml
   visualization:
   type: "kibana"
   kibana:
   tolerations:
   - key: "kibana"
     operator: "Exists"
     effect: "NoExecute"
     tolerationSeconds: 6000
   ```
1. Specify the key that you added to the node.
2. Specify the `Exists` operator to require the `key/value/effect` parameters to match.
3. Specify the `NoExecute` effect.
4. Optionally, specify the `tolerationSeconds` parameter to set how long a pod can remain bound to a node before being evicted.

This toleration matches the taint created by the `oc adm taint` command. A pod with this toleration would be able to schedule onto node1.

### 3.5. CONFIGURING CLUSTER LOGGING STORAGE

Elasticsearch is a memory-intensive application. The default cluster logging installation deploys 16G of memory for both memory requests and memory limits. The initial set of OpenShift Container Platform nodes might not be large enough to support the Elasticsearch cluster. You must add additional nodes to the OpenShift Container Platform cluster to run with the recommended or higher memory. Each Elasticsearch node can operate with a lower memory setting, though this is not recommended for production environments.

#### 3.5.1. Storage considerations for cluster logging and OpenShift Container Platform

A persistent volume is required for each Elasticsearch deployment to have one data volume per data node. On OpenShift Container Platform this is achieved using persistent volume claims.

The Elasticsearch Operator names the PVCs using the Elasticsearch resource name. Refer to Persistent Elasticsearch Storage for more details.

**NOTE**

If you use a local volume for persistent storage, do not use a raw block volume, which is described with `volumeMode: block` in the `LocalVolume` object. Elasticsearch cannot use raw block volumes.

Fluentd ships any logs from `systemd journal` and `var/log/containers/` to Elasticsearch.

Therefore, consider how much data you need in advance and that you are aggregating application log data. Some Elasticsearch users have found that it is necessary to keep absolute storage consumption around 50% and below 70% at all times. This helps to avoid Elasticsearch becoming unresponsive during large merge operations.

By default, at 85% Elasticsearch stops allocating new data to the node, at 90% Elasticsearch attempts to relocate existing shards from that node to other nodes if possible. But if no nodes have free capacity below 85%, Elasticsearch effectively rejects creating new indices and becomes RED.

**NOTE**

These low and high watermark values are Elasticsearch defaults in the current release. You can modify these values, but you also must apply any modifications to the alerts also. The alerts are based on these defaults.

### 3.5.2. Additional resources
3.6. CONFIGURING CPU AND MEMORY LIMITS FOR CLUSTER LOGGING COMPONENTS

You can configure both the CPU and memory limits for each of the cluster logging components as needed.

3.6.1. Configuring CPU and memory limits

The cluster logging components allow for adjustments to both the CPU and memory limits.

**Procedure**

1. Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

   ```bash
   $ oc edit ClusterLogging instance -n openshift-logging
   ```

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   ...
   spec:
     managementState: "Managed"
     logStore:
       type: "elasticsearch"
       elasticsearch:
         nodeCount: 2
         resources: 1
         limits:
           memory: 2Gi
           requests:
             cpu: 200m
             memory: 2Gi
         storage:
           storageClassName: "gp2"
           size: "200G"
           redundancyPolicy: "SingleRedundancy"
     visualization:
       type: "kibana"
       kibana:
         resources: 2
         limits:
           memory: 1Gi
           requests:
             cpu: 500m
             memory: 1Gi
     proxy:
       resources: 3
       limits:
         memory: 100Mi
   ```
Specify the CPU and memory limits and requests for the log store as needed. For Elasticsearch, you must adjust both the request value and the limit value.

Specify the CPU and memory limits and requests for the log visualizer as needed.

Specify the CPU and memory limits and requests for the log curator as needed.

Specify the CPU and memory limits and requests for the log collector as needed.

### 3.7. USING TOLERATIONS TO CONTROL CLUSTER LOGGING POD PLACEMENT

You can use taints and tolerations to ensure that cluster logging pods run on specific nodes and that no other workload can run on those nodes.

Taints and tolerations are simple key: value pair. A taint on a node instructs the node to repel all pods that do not tolerate the taint.

The key is any string, up to 253 characters and the value is any string up to 63 characters. The string must begin with a letter or number, and may contain letters, numbers, hyphens, dots, and underscores.

**Sample cluster logging CR with tolerations**

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
spec:
```

1. Specify the CPU and memory limits and requests for the log store as needed. For Elasticsearch, you must adjust both the request value and the limit value.

2. Specify the CPU and memory limits and requests for the log visualizer as needed.

3. Specify the CPU and memory limits and requests for the log curator as needed.

4. Specify the CPU and memory limits and requests for the log collector as needed.
This toleration is added to the Elasticsearch pods.

This toleration is added to the Kibana pod.

This toleration is added to the logging collector pods.
3.7.1. Using tolerations to control the log store pod placement

You can control which nodes the log store pods runs on and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to the log store pods through the ClusterLogging custom resource (CR) and apply taints to a node through the node specification. A taint on a node is a key:value pair that instructs the node to repel all pods that do not tolerate the taint. Using a specific key:value pair that is not on other pods ensures only the log store pods can run on that node.

By default, the log store pods have the following toleration:

```
tolerations:
  - effect: "NoExecute"
    key: "node.kubernetes.io/disk-pressure"
    operator: "Exists"
```

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

1. Use the following command to add a taint to a node where you want to schedule the cluster logging pods:

   ```bash
   $ oc adm taint nodes <node-name> <key>=<value>:<effect>
   ```

   For example:

   ```bash
   $ oc adm taint nodes node1 elasticsearch=node:NoExecute
   ```

   This example places a taint on node1 that has key `elasticsearch`, value `node`, and taint effect `NoExecute`. Nodes with the `NoExecute` effect schedule only pods that match the taint and remove existing pods that do not match.

2. Edit the logstore section of the ClusterLogging CR to configure a toleration for the Elasticsearch pods:

   ```json
   logStore:
     type: "elasticsearch"
     elasticsearch:
       nodeCount: 1
       tolerations:
         - key: "elasticsearch"
           operator: "Exists"
           effect: "NoExecute"
           tolerationSeconds: 6000
   ```

   **1** Specify the key that you added to the node.

   **2** Specify the Exists operator to require a taint with the key `elasticsearch` to be present on the Node.
Specify the **NoExecute** effect.

Optionally, specify the **tolerationSeconds** parameter to set how long a pod can remain bound to a node before being evicted.

This toleration matches the taint created by the `oc adm taint` command. A pod with this toleration could be scheduled onto **node1**.

### 3.7.2. Using tolerations to control the log visualizer pod placement

You can control the node where the log visualizer pod runs and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to the log visualizer pod through the **ClusterLogging** custom resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all pods that do not tolerate the taint. Using a specific **key:value** pair that is not on other pods ensures only the Kibana pod can run on that node.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

1. Use the following command to add a taint to a node where you want to schedule the log visualizer pod:

   ```bash
   $ oc adm taint nodes <node-name> <key>=<value>:<effect>
   ```

   For example:

   ```bash
   $ oc adm taint nodes node1 kibana=node:NoExecute
   ```

   This example places a taint on **node1** that has key **kibana**, value **node**, and taint effect **NoExecute**. You must use the **NoExecute** taint effect. **NoExecute** schedules only pods that match the taint and remove existing pods that do not match.

2. Edit the **visualization** section of the **ClusterLogging** CR to configure a toleration for the Kibana pod:

   ```json
   visualization:
      type: "kibana"
      kibana:
         tolerations:
            - key: "kibana"  # Specify the key that you added to the node.
              operator: "Exists"  # Specify the **Exists** operator to require the **key/value/effect** parameters to match.
              effect: "NoExecute"
              tolerationSeconds: 6000
   ```
Specify the **NoExecute** effect.

Optionally, specify the **tolerationSeconds** parameter to set how long a pod can remain bound to a node before being evicted.

This toleration matches the taint created by the `oc adm taint` command. A pod with this toleration would be able to schedule onto `node1`.

### 3.7.3. Using tolerations to control the log collector pod placement

You can ensure which nodes the logging collector pods run on and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to logging collector pods through the **ClusterLogging** custom resource (CR) and apply taints to a node through the node specification. You can use taints and tolerations to ensure the pod does not get evicted for things like memory and CPU issues.

By default, the logging collector pods have the following toleration:

```
  tolerations:
  - key: "node-role.kubernetes.io/master"
    operator: "Exists"
    effect: "NoExecute"
```

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

1. Use the following command to add a taint to a node where you want logging collector pods to schedule logging collector pods:

   ```bash
   $ oc adm taint nodes <node-name> <key>=<value>:<effect>
   
   For example:
   ```

   ```bash
   $ oc adm taint nodes node1 collector=node:NoExecute
   ```

   This example places a taint on `node1` that has key `collector`, value `node`, and taint effect `NoExecute`. You must use the `NoExecute` taint effect. `NoExecute` schedules only pods that match the taint and removes existing pods that do not match.

2. Edit the **collection** stanza of the **ClusterLogging** custom resource (CR) to configure a toleration for the logging collector pods:

   ```yaml
   collection:
   logs:
     type: "fluentd"
     fluentd:
       tolerations:
       - key: "collector"
   ```
Specify the key that you added to the node.

Specify the **Exists** operator to require the `key/value/effect` parameters to match.

Specify the **NoExecute** effect.

Optionally, specify the `tolerationSeconds` parameter to set how long a pod can remain bound to a node before being evicted.

This toleration matches the taint created by the `oc adm taint` command. A pod with this toleration would be able to schedule onto `node1`.

### 3.7.4. Additional resources

For more information about taints and tolerations, see [Controlling pod placement using node taints](#).

### 3.8. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS

You can use node selectors to deploy the Elasticsearch, Kibana, and Curator pods to different nodes.

#### 3.8.1. Moving the cluster logging resources

You can configure the Cluster Logging Operator to deploy the pods for any or all of the Cluster Logging components, Elasticsearch, Kibana, and Curator to different nodes. You cannot move the Cluster Logging Operator pod from its installed location.

For example, you can move the Elasticsearch pods to a separate node because of high CPU, memory, and disk requirements.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed. These features are not installed by default.

**Procedure**

1. Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

   ```bash
   $ oc edit ClusterLogging instance
   ```

   ```yaml
   apiVersion: logging.openshift.io/v1
   kind: ClusterLogging
   ...
   spec:
     collection:
       logs:
   ```
Add a `nodeSelector` parameter with the appropriate value to the component you want to move. You can use a `nodeSelector` in the format shown or use `<key>: <value>` pairs, based on the value specified for the node.

**Verification**

To verify that a component has moved, you can use the `oc get pod -o wide` command.

For example:

- You want to move the Kibana pod from the `ip-10-0-147-79.us-east-2.compute.internal` node:

  ```bash
  $ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
  ```

  **Example output**
You want to move the Kibana Pod to the `ip-10-0-139-48.us-east-2.compute.internal` node, a dedicated infrastructure node:

```
$ oc get nodes
```

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
<th>IP</th>
<th>NODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>kibana-5b8bdff44f9-ccpq9</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>27s</td>
<td>10.129.2.18</td>
<td>ip-10-0-147-79.us-east-2.compute.internal &lt;none&gt; &lt;none&gt;</td>
</tr>
</tbody>
</table>

Note that the node has a `node-role.kubernetes.io/infra: ''` label:

```
$ oc get node ip-10-0-139-48.us-east-2.compute.internal -o yaml
```

**Example output**

```yaml
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-139-48.us-east-2.compute.internal
  selfLink: /api/v1/nodes/ip-10-0-139-48.us-east-2.compute.internal
  uid: 62038aa9-661f-41d7-ba93-b5f1b6ef8751
  resourceVersion: 39083
  creationTimestamp: '2020-04-13T19:07:55Z'
labels:
  node-role.kubernetes.io/infra: ''
...
```

To move the Kibana pod, edit the `ClusterLogging` CR to add a node selector:

```yaml
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:

...

visualization:
  kibana:
    nodeSelector: 1
```
node-role.kubernetes.io/infra: "
proxy:
  resources: null
replicas: 1
resources: null
type: kibana

Add a node selector to match the label in the node specification.

- After you save the CR, the current Kibana pod is terminated and new pod is deployed:

  ```
  $ oc get pods
  
  Example output
  
<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster-logging-operator-84d98649c4-zb9g7</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
<tr>
<td>elasticsearch-cdm-hwv01pfl7-1-5658f554f-kupmgl</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>elasticsearch-cdm-hwv01pfl7-2-84c877d75d-75wq</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>elasticsearch-cdm-hwv01pfl7-3-f5d95b87b-4nx78</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>fluentd-42dzz</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>fluentd-d74rq</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>fluentd-m5vrr9</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>fluentd-nkol7</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>fluentd-pdvqbb</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>fluentd-tflh6</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>28m</td>
</tr>
<tr>
<td>kibana-5b8bdf4f9-ccpq9</td>
<td>2/2</td>
<td>Terminating</td>
<td>0</td>
<td>4m11s</td>
</tr>
<tr>
<td>kibana-7d85dccc8-bcfpfb</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>33s</td>
</tr>
</tbody>
</table>
  
  The new pod is on the **ip-10-0-139-48.us-east-2.compute.internal** node:

  ```
  $ oc get pod kibana-7d85dccc8-bcfpfb -o wide
  
  Example output
  
  | NAME                      | READY | STATUS | RESTARTS | AGE   | IP            | NODE
  |---------------------------|-------|--------|----------|-------|---------------|------
  | kibana-7d85dccc8-bcfpfb   | 2/2   | Running| 0        | 43s   | 10.131.0.22   | ip-10-0-139-48.us-east-2.compute.internal
  |                           |       |        |          |       | <none>        | <none>

  After a few moments, the original Kibana pod is removed.

  ```
  $ oc get pods
  
  Example output
  
<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster-logging-operator-84d98649c4-zb9g7</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>30m</td>
</tr>
<tr>
<td>elasticsearch-cdm-hv01pfl7-1-5658f554f-kupmgl</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
<tr>
<td>elasticsearch-cdm-hv01pfl7-2-84c877d75d-75wq</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
<tr>
<td>elasticsearch-cdm-hv01pfl7-3-f5d95b87b-4nx78</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
<tr>
<td>fluentd-42dzz</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
</tbody>
</table>
3.9. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD

Because Fluentd reads from the journal, and the journal default settings are very low, journal entries can be lost because the journal cannot keep up with the logging rate from system services.

We recommend setting `RateLimitInterval=1s` and `RateLimitBurst=10000` (or even higher if necessary) to prevent the journal from losing entries.

3.9.1. Configuring systemd-journald for cluster logging

As you scale up your project, the default logging environment might need some adjustments.

For example, if you are missing logs, you might have to increase the rate limits for journald. You can adjust the number of messages to retain for a specified period of time to ensure that cluster logging does not use excessive resources without dropping logs.

You can also determine if you want the logs compressed, how long to retain logs, how or if the logs are stored, and other settings.

Procedure

1. Create a `journald.conf` file with the required settings:

   ```
   Compress=yes
   ForwardToConsole=no
   ForwardToSyslog=no
   MaxRetentionSec=1month
   RateLimitBurst=10000
   RateLimitInterval=1s
   Storage=persistent
   SyncIntervalSec=1s
   SystemMaxUse=8g
   SystemKeepFree=20%
   SystemMaxFileSize=10M
   ```

   - **Specify whether you want logs compressed before they are written to the file system. Specify yes to compress the message or no to not compress. The default is yes.**
   - **Configure whether to forward log messages. Defaults to no for each. Specify:**
     - **ForwardToConsole** to forward logs to the system console.
     - **ForwardToKsmg** to forward logs to the kernel log buffer.
     - **ForwardToSyslog** to forward to a syslog daemon.
     - **ForwardToWall** to forward messages as wall messages to all logged-in users.
Specifying the Maximum Time to Store Journal Entries

Enter a number to specify seconds. Or include a unit: "year", "month", "week", "day", "h" or "m". Enter 0 to disable. The default is 0.

Configure Rate Limiting

If, during the time interval defined by RateLimitIntervalSec, more logs than specified in RateLimitBurst are received, all further messages within the interval are dropped until the interval is over. It is recommended to set RateLimitInterval=1s and RateLimitBurst=10000, which are the defaults.

Specifying How Logs Are Stored

The default is persistent:

- volatile to store logs in memory in /var/log/journal/
- persistent to store logs to disk in /var/log/journal/. systemd creates the directory if it does not exist.
- auto to store logs in /var/log/journal/ if the directory exists. If it does not exist, systemd temporarily stores logs in /run/systemd/journal.
- none to not store logs. systemd drops all logs.

Specify the timeout before synchronizing journal files to disk for ERR, WARNING, NOTICE, INFO, and DEBUG logs. systemd immediately syncs after receiving a CRIT, ALERT, or EMERG log. The default is 1s.

Specify the Maximum Size the Journal Can Use

The default is 8g.

Specify How Much Disk Space Systemd Must Leave Free

The default is 20%.

Specify the Maximum Size for Individual Journal Files Stored Persistently in /var/log/journal

The default is 10M.

NOTE

If you are removing the rate limit, you might see increased CPU utilization on the system logging daemons as it processes any messages that would have previously been throttled.

For more information on systemd settings, see https://www.freedesktop.org/software/systemd/man/journald.conf.html. The default settings listed on that page might not apply to OpenShift Container Platform.

2. Convert the journal.conf file to base64:

   $ export jrn_cnf=$( cat /journald.conf | base64 -w0 )

3. Create a new MachineConfig object for master or worker and add the journal.conf parameters:

   For example:

   apiVersion: machineconfiguration.openshift.io/v1
   kind: MachineConfig
   metadata:
     labels:
       machineconfiguration.openshift.io/role: worker
1. Set the permissions for the `journal.conf` file. It is recommended to set 0644 permissions.

2. Specify the path to the base64-encoded `journal.conf` file.

4. Create the machine config:

   ```bash
   $ oc apply -f <filename>.yaml
   ```

   The controller detects the new `MachineConfig` object and generates a new rendered-worker-<hash> version.

5. Monitor the status of the rollout of the new rendered configuration to each node:

   ```bash
   $ oc describe machineconfigpool/worker
   ```

   **Example output**

   ```
   Name: worker
   Namespace: 
   Labels: machineconfiguration.openshift.io/mco-built-in= 
   Annotations: <none>
   API Version: machineconfiguration.openshift.io/v1
   Kind: MachineConfigPool
   ...
   Conditions:
   Message:
   Reason: All nodes are updating to rendered-worker-913514517bcea7c93bd446f4830bc64e
   ```

### 3.10. CONFIGURING THE LOG CURATOR

You can configure log retention time. That is, you can specify how long the default Elasticsearch log store keeps indices by configuring a separate retention policy for each of the three log sources: infrastructure logs, application logs, and audit logs. For instructions, see Configuring log retention time.
NOTE

Configuring log retention time is recommended method for curating log data: It works with both the current data model and the previous data model from OpenShift Container Platform 4.4 and earlier.

Optionally, to remove Elasticsearch indices that use the data model from OpenShift Container Platform 4.4 and earlier, you can also use the Elasticsearch Curator. The following sections explain how to use the Elasticsearch Curator.

IMPORTANT

The Elasticsearch Curator is deprecated in OpenShift Container Platform 4.7 (OpenShift Logging 5.0) and will be removed in OpenShift Logging 5.1.

3.10.1. Configuring the Curator schedule

You can specify the schedule for Curator using the Cluster Logging custom resource created by the OpenShift Logging installation.

IMPORTANT

The Elasticsearch Curator is deprecated in OpenShift Container Platform 4.7 (OpenShift Logging 5.0) and will be removed in OpenShift Logging 5.1.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

To configure the Curator schedule:

1. Edit the ClusterLogging custom resource in the openshift-logging project:

   ```
   $ oc edit clusterlogging instance
   
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   ...

   curation:
     curator:
       schedule: 30 3 * * *
       type: curator
   
   1 Specify the schedule for Curator in cron format.
   ```
NOTE
The time zone is set based on the host node where the Curator pod runs.

3.10.2. Configuring Curator index deletion
You can configure Elasticsearch Curator to delete Elasticsearch data that uses the data model prior to OpenShift Container Platform version 4.5. You can configure per-project and global settings. Global settings apply to any project not specified. Per-project settings override global settings.

IMPORTANT
The Elasticsearch Curator is deprecated in OpenShift Container Platform 4.7 (OpenShift Logging 5.0) and will be removed in OpenShift Logging 5.1.

Prerequisites
- Cluster logging must be installed.

Procedure
To delete indices:

1. Edit the OpenShift Container Platform custom Curator configuration file:

   $ oc edit configmap/curator

2. Set the following parameters as needed:

   ```yaml
   project_name: |
   action
   unit:value
   
   The available parameters are:

   Table 3.2. Project options

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>project_name</td>
<td>The actual name of a project, such as <code>myapp-devel</code> For OpenShift Container Platform operations logs, use the name <code>operations</code> as the project name.</td>
</tr>
<tr>
<td>action</td>
<td>The action to take, currently only <code>delete</code> is allowed.</td>
</tr>
<tr>
<td>unit</td>
<td>The period to use for deletion, <code>days</code>, <code>weeks</code>, or <code>months</code>.</td>
</tr>
<tr>
<td>value</td>
<td>The number of units.</td>
</tr>
</tbody>
</table>

   Table 3.3. Filter options
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.defaults</td>
<td>Use <code>.defaults</code> as the <code>project_name</code> to set the defaults for projects that are not specified.</td>
</tr>
<tr>
<td>.regex</td>
<td>The list of regular expressions that match project names.</td>
</tr>
<tr>
<td>pattern</td>
<td>The valid and properly escaped regular expression pattern enclosed by single quotation marks.</td>
</tr>
</tbody>
</table>

For example, to configure Curator to:

- Delete indices in the **myapp-dev** project older than 1 day
- Delete indices in the **myapp-qe** project older than 1 week
- Delete operations logs older than 8 weeks
- Delete all other projects indices after they are 31 days old
- Delete indices older than 1 day that are matched by the `^project\..+-dev\..*$` regex
- Delete indices older than 2 days that are matched by the `^project\..+-test\..*$` regex

Use:

```
config.yaml:
  .defaults:
    delete:
      days: 31
  .operations:
    delete:
      weeks: 8
  myapp-dev:
    delete:
      days: 1
  myapp-qe:
    delete:
      weeks: 1
  .regex:
    - pattern: `^project\..+-dev\..*$`
      delete:
        days: 1
    - pattern: `^project\..+-test\..*$`
      delete:
        days: 2
```
When you use **months** as the **unit** for an operation, Curator starts counting at the first day of the current month, not the current day of the current month. For example, if today is April 15, and you want to delete indices that are 2 months older than today (delete: months: 2), Curator does not delete indices that are dated older than February 15; it deletes indices older than February 1. That is, it goes back to the first day of the current month, then goes back two whole months from that date. If you want to be exact with Curator, it is best to use days (for example, **delete: days: 30**).

3.11. MAINTENANCE AND SUPPORT

3.11.1. About unsupported configurations

The supported way of configuring cluster logging is by configuring it using the options described in this documentation. Do not use other configurations, as they are unsupported. Configuration paradigms might change across OpenShift Container Platform releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this documentation, your changes will disappear because the Elasticsearch Operator and Cluster Logging Operator reconcile any differences. The Operators reverse everything to the defined state by default and by design.

**NOTE**

If you *must* perform configurations not described in the OpenShift Container Platform documentation, you *must* set your Cluster Logging Operator or Elasticsearch Operator to **Unmanaged**. An unmanaged cluster logging environment is *not supported* and does not receive updates until you return cluster logging to **Managed**.

3.11.2. Unsupported configurations

You must set the Cluster Logging Operator to the unmanaged state in order to modify the following components:

- the Curator cron job
- the Elasticsearch CR
- the Kibana deployment
- the fluent.conf file
- the Fluentd daemon set

You must set the Elasticsearch Operator to the unmanaged state in order to modify the following component:

- the Elasticsearch deployment files.

Explicitly unsupported cases include:

- **Configuring default log rotation** You cannot modify the default log rotation configuration.
- **Configuring the collected log location** You cannot change the location of the log collector output file, which by default is `/var/log/fluentd/fluentd.log`. 
Throttling log collection You cannot throttle down the rate at which the logs are read in by the log collector.

Configuring log collection JSON parsing You cannot format log messages in JSON.

Configuring the logging collector using environment variables You cannot use environment variables to modify the log collector.

Configuring how the log collector normalizes logs You cannot modify default log normalization.

Configuring Curator in scripted deployments You cannot configure log curation in scripted deployments.

Using the Curator Action file You cannot use the Curator config map to modify the Curator action file.

3.11.3. Support policy for unmanaged Operators

The management state of an Operator determines whether an Operator is actively managing the resources for its related component in the cluster as designed. If an Operator is set to an unmanaged state, it does not respond to changes in configuration nor does it receive updates.

While this can be helpful in non-production clusters or during debugging, Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades.

An Operator can be set to an unmanaged state using the following methods:

- **Individual Operator configuration**
  Individual Operators have a managementState parameter in their configuration. This can be accessed in different ways, depending on the Operator. For example, the Cluster Logging Operator accomplishes this by modifying a custom resource (CR) that it manages, while the Cluster Samples Operator uses a cluster-wide configuration resource.

  Changing the managementState parameter to Unmanaged means that the Operator is not actively managing its resources and will take no action related to the related component. Some Operators might not support this management state as it might damage the cluster and require manual recovery.

  **WARNING**
  Changing individual Operators to the Unmanaged state renders that particular component and functionality unsupported. Reported issues must be reproduced in Managed state for support to proceed.

- **Cluster Version Operator (CVO) overrides**
  The spec.overrides parameter can be added to the CVO’s configuration to allow administrators to provide a list of overrides to the CVO’s behavior for a component. Setting the spec.overrides[].unmanaged parameter to true for a component blocks cluster upgrades and alerts the administrator after a CVO override has been set:
Disabling ownership via cluster version overrides prevents upgrades. Please remove overrides before continuing.

**WARNING**

Setting a CVO override puts the entire cluster in an unsupported state. Reported issues must be reproduced after removing any overrides for support to proceed.
CHAPTER 4. VIEWING LOGS FOR A RESOURCE

You can view the logs for various resources, such as builds, deployments, and pods by using the OpenShift CLI (oc) and the web console.

NOTE

Resource logs are a default feature that provides limited log viewing capability. To enhance your log retrieving and viewing experience, it is recommended that you install OpenShift Container Platform cluster logging. Cluster logging aggregates all the logs from your OpenShift Container Platform cluster, such as node system audit logs, application container logs, and infrastructure logs, into a dedicated log store. You can then query, discover, and visualize your log data through the Kibana interface. Resource logs do not access the cluster logging log store.

4.1. VIEWING RESOURCE LOGS

You can view the log for various resources in the OpenShift CLI (oc) and web console. Logs read from the tail, or end, of the log.

Prerequisites

- Access to the OpenShift CLI (oc).

Procedure (UI)

1. In the OpenShift Container Platform console, navigate to Workloads → Pods or navigate to the pod through the resource you want to investigate.

NOTE

Some resources, such as builds, do not have pods to query directly. In such instances, you can locate the Logs link on the Details page for the resource.

2. Select a project from the drop-down menu.

3. Click the name of the pod you want to investigate.

4. Click Logs.

Procedure (CLI)

- View the log for a specific pod:

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

  where:

  -f

    Optional: Specifies that the output follows what is being written into the logs.

  <pod_name>

    Specifies the name of the pod.
<container_name>
Optional: Specifies the name of a container. When a pod has more than one container, you must specify the container name.

For example:

- $ oc logs ruby-58cd97df55-mww7r
- $ oc logs -f ruby-57f7f4855b-znl92 -c ruby

The contents of log files are printed out.

• View the log for a specific resource:

- $ oc logs <object_type>/<resource_name> ①

① Specifies the resource type and name.

For example:

- $ oc logs deployment/ruby

The contents of log files are printed out.
CHAPTER 5. VIEWING CLUSTER LOGS BY USING KIBANA

OpenShift Container Platform cluster logging includes a web console for visualizing collected log data. Currently, OpenShift Container Platform deploys the Kibana console for visualization.

Using the log visualizer, you can do the following with your data:

- search and browse the data using the Discover tab.
- chart and map the data using the Visualize tab.
- create and view custom dashboards using the Dashboard tab.

Use and configuration of the Kibana interface is beyond the scope of this documentation. For more information, on using the interface, see the Kibana documentation.

NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the default output for audit logs.

5.1. DEFINING KIBANA INDEX PATTERNS

An index pattern defines the Elasticsearch indices that you want to visualize. To explore and visualize data in Kibana, you must create an index pattern.

Prerequisites

- A user must have the cluster-admin role, the cluster-reader role, or both roles to view the infra and audit indices in Kibana. The default kubeadmin user has proper permissions to view these indices.

If you can view the pods and logs in the default, kube- and openshift- projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

```bash
$ oc auth can-i get pods/log -n <project>
```

Example output

```
yes
```

NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the default output for audit logs.

- Elasticsearch documents must be indexed before you can create index patterns. This is done automatically, but it might take a few minutes in a new or updated cluster.

Procedure
To define index patterns and create visualizations in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher and select Logging.

2. Create your Kibana index patterns by clicking Management → Index Patterns → Create index pattern:
   - Each user must manually create index patterns when logging into Kibana the first time in order to see logs for their projects. Users must create an index pattern named app and use the @timestamp time field to view their container logs.
   - Each admin user must create index patterns when logged into Kibana the first time for the app, infra, and audit indices using the @timestamp time field.

3. Create Kibana Visualizations from the new index patterns.

5.2. VIEWING CLUSTER LOGS IN KIBANA

You view cluster logs in the Kibana web console. The methods for viewing and visualizing your data in Kibana that are beyond the scope of this documentation. For more information, refer to the Kibana documentation.

Prerequisites

- Cluster logging and Elasticsearch must be installed.
- Kibana index patterns must exist.
- A user must have the cluster-admin role, the cluster-reader role, or both roles to view the infra and audit indices in Kibana. The default kubeadmin user has proper permissions to view these indices.

If you can view the pods and logs in the default, kube- and openshift- projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

```
$ oc auth can-i get pods/log -n <project>
```

Example output

```
yes
```

NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the default output for audit logs.

Procedure

To view logs in Kibana:
1. In the OpenShift Container Platform console, click the Application Launcher and select Logging.

2. Log in using the same credentials you use to log in to the OpenShift Container Platform console. The Kibana interface launches.

3. In Kibana, click Discover.

4. Select the index pattern you created from the drop-down menu in the top-left corner: app, audit, or infra. The log data displays as time-stamped documents.

5. Expand one of the time-stamped documents.

6. Click the JSON tab to display the log entry for that document.

Example 5.1. Sample infrastructure log entry in Kibana

```json
{
  "_index": "infra-000001",
  "_type": "_doc",
  "_id": "YmJmYTBlNDkZTRmLTliMGQtMjE3NmFiOGUyOWM3",
  "_version": 1,
  "_score": null,
  "_source": {
    "docker": {
      "container_id": "f85fa55bbef7bb783f041066be1e7c267a6b88c4603dfce213e32c1"
    },
    "kubernetes": {
      "container_name": "registry-server",
      "namespace_name": "openshift-marketplace",
      "pod_name": "redhat-marketplace-n64gc",
      "container_image": "registry.redhat.io/redhat/redhat-marketplace-index:v4.6",
      "container_image_id": "registry.redhat.io/redhat-marketplace-index@sha256:65fc0c45aabb95809e376feb065771ecda9e5e59cc8b3024c4545c168f",
      "pod_id": "8f594ea2-c866-4b5c-a1c8-a50756704b2a",
      "host": "ip-10-0-182-28.us-east-2.compute.internal",
      "master_url": "https://kubernetes.default.svc",
      "namespace_id": "3abab127-7669-4eb3-b9ef-44c04ad68d38",
      "namespace_labels": {
        "openshift_io/cluster-monitoring": "true"
      },
      "flat_labels": {
        "catalogsource_operators_coreos_com/update=redhat-marketplace"
      }
    },
    "message": "time="2020-09-23T20:47:03Z" level=info msg="serving registry" database=/database/index.db port=50051",
    "level": "unknown",
    "hostname": "ip-10-0-182-28.internal",
    "pipeline_metadata": {
      "collector": {
        "ipaddr4": "10.0.182.28",
        "inputname": "fluent-plugin-systemd",
        "name": "fluentd",
      }
    }
  }
}
```
CHAPTER 5. VIEWING CLUSTER LOGS BY USING KIBANA
CHAPTER 6. FORWARDING LOGS TO THIRD PARTY SYSTEMS

By default, OpenShift Container Platform cluster logging sends logs to the default internal Elasticsearch log store, defined in the `ClusterLogging` custom resource (CR).

You can configure cluster logging to send logs to destinations outside of your OpenShift Container Platform cluster instead of the default Elasticsearch log store using the following methods:

- **Sending logs using the Fluentd forward protocol** You can create a config map to use the Fluentd forward protocol to securely send logs to an external logging aggregator that accepts the Fluent forward protocol.

- **Sending logs using syslog** You can create a config map to use the syslog protocol to send logs to an external syslog (RFC 3164) server.

Alternatively, you can use the Log Forwarding API, currently in Technology Preview. The Log Forwarding API, which is easier to configure than the Fluentd protocol and syslog, exposes configuration for sending logs to the internal Elasticsearch log store and to external Fluentd log aggregation solutions.

You cannot use the config map methods and the Log Forwarding API in the same cluster.

**IMPORTANT**

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

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

The methods for forwarding logs using a config map are deprecated and will be replaced by the Log Forwarding API in a future release.

### 6.1. FORWARDING LOGS USING THE FLUENTD FORWARD PROTOCOL

You can use the Fluentd forward protocol to send a copy of your logs to an external log aggregator, instead of the default Elasticsearch log store. On the OpenShift Container Platform cluster, you use the Fluentd forward protocol to send logs to a server configured to accept the protocol. You are responsible to configure the external log aggregator to receive the logs from OpenShift Container Platform.

**NOTE**

This method for forwarding logs is deprecated in OpenShift Container Platform and will be replaced by the Log Forwarding API in a future release.

To configure OpenShift Container Platform to send logs using the Fluentd forward protocol, create a ConfigMap called `secure-forward` in the `openshift-logging` namespace that points to an external log aggregator.
IMPORTANT

Starting with the OpenShift Container Platform 4.3, the process for using the Fluentd forward protocol has changed. You now need to create a ConfigMap, as described below.

Additionally, you can add any certificates required by your configuration to a secret named `secure-forward` that will be mounted to the Fluentd Pods.

Sample `secure-forward.conf`

```xml
<store>
  @type forward
  <security>
    self_hostname ${hostname} # ${hostname} is a placeholder.
    shared_key "fluent-receiver"
  </security>
  transport tls
  tls_verify_hostname false       # Set false to ignore server cert hostname.
  tls_cert_path '/etc/ocp-forward/ca-bundle.crt'
  <buffer>
    @type file
    path '/var/lib/fluentd/secureforwardlegacy'
    queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '1024' }"
    chunk_limit_size "#{ENV['BUFFER_SIZE_LIMIT'] || '1m' }"
    flush_interval "#{ENV['FORWARD_FLUSH_INTERVAL'] || '5s'}"
    flush_at_shutdown "#{ENV['FLUSH_AT_SHUTDOWN'] || 'false'}"
    flush_thread_count "#{ENV['FLUSH_THREAD_COUNT'] || 2}"
    retry_max_interval "#{ENV['FORWARD_RETRY_WAIT'] || '300'}"
    retry_forever true
    # the systemd journald 0.0.8 input plugin will just throw away records if the buffer
    # queue limit is hit - 'block' will halt further reads and keep retrying to flush the
    # buffer to the remote - default is 'exception' because in_tail handles that case
    overflow_action "#{ENV['BUFFER_QUEUE_FULL_ACTION'] || 'exception'}"
  </buffer>
  <server>
    host fluent-receiver.openshift-logging.svc  # or IP
    port 24224
  </server>
</store>
```

Sample `secure-forward` ConfigMap based on the configuration

```yaml
apiVersion: v1
data:
  secure-forward.conf: "<store>
    \@type forward
    \<security>
    \self_hostname ${hostname} # ${hostname} is a placeholder.
    \shared_key "fluent-receiver"
    \</security>
    \transport tls
    \tls_verify_hostname false       # Set false to ignore server cert hostname.
    \tls_cert_path '/etc/ocp-forward/ca-bundle.crt'
    \<buffer>
```
To configure OpenShift Container Platform to forward logs using the Fluentd `forward` protocol:

1. Create a configuration file named `secure-forward.conf` for the `forward` parameters:

   a. Configure the secrets and TLS information:

   ```yaml
   <store>
   @type forward
   
   self_hostname ${hostname}  # 1
   shared_key <SECRET_STRING>  # 2
   
   transport tls  # 3
   
   tls_verify_hostname true  # 4
   tls_cert_path <path_to_file>  # 5
   
   </store>
   
   @type file
   path '/var/lib/fluentd/secureforwardlegacy'
   queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '1024'}"
   chunk_limit_size "#{ENV['BUFFER_SIZE_LIMIT'] || '1m'}"
   flush_interval "#{ENV['FORWARD_FLUSH_INTERVAL'] || '5s'}"
   flush_at_shutdown "#{ENV['FLUSH_AT_SHUTDOWN'] || 'false'}"
   flush_thread_count "#{ENV['FLUSH_THREAD_COUNT'] || 2}"
   retry_max_interval "#{ENV['FORWARD_RETRY_WAIT'] || '300'}"
   retry_forever true
   # the systemd journald 0.0.8 input plugin will just throw away records if the buffer
   # queue limit is hit - 'block' will halt further reads and keep retrying to flush the
   # buffer to the remote - default is 'exception' because in_tail handles that case
   overflow_action "#{ENV['BUFFER_QUEUE_FULL_ACTION'] || 'exception'}"
   </buffer>
   </server>
   
   host fluent-receiver.openshift-logging.svc  # or IP
   port 24224
   </server>
   </store>
   
   kind: ConfigMap
   metadata:
   creationTimestamp: "2020-01-15T18:56:04Z"
   name: secure-forward
   namespace: openshift-logging
   resourceVersion: "19148"
   selfLink: /api/v1/namespaces/openshift-logging/configmaps/secure-forward
   uid: 6fd83202-93ab-d851b1d0f3e8
   ```

Specify the default value of the auto-generated certificate common name (CN).

Enter the Shared key between nodes.

Specify `tls` to enable TLS validation.

Set to `true` to verify the server cert hostname. Set to `false` to ignore server cert hostname.
Specify the path to private CA certificate file as `/etc/ocp-forward/ca_cert.pem`.

To use mTLS, see the Fluentd documentation for information about client certificate, key parameters, and other settings.

b. Configure the name, host, and port for your external Fluentd server:

```
<server>
  name 1
  host 2
  hostlabel 3
  port 4
</server>
<server>
  name 5
  host
</server>
```

1. Optionally, enter a name for this server.
2. Specify the host name or IP of the server.
3. Specify the host label of the server.
4. Specify the port of the server.
5. Optionally, add additional servers. If you specify two or more servers, forward uses these server nodes in a round-robin order.

For example:

```
<server>
  name externalsever1
  host 192.168.1.1
  hostlabel externalsever1.example.com
  port 24224
</server>
<server>
  name externalsever2
  host externalsever2.example.com
  port 24224
</server>
```

2. Create a ConfigMap named `secure-forward` in the openshift-logging namespace from the configuration file:

```
$ oc create configmap secure-forward --from-file=secure-forward.conf -n openshift-logging
```

3. Optional: Import any secrets required for the receiver:
For example:

```bash
$ oc create secret generic secure-forward --from-file=<arbitrary-name-of-key1>=cert_file_from_fluentd_receiver --from-literal=shared_key=value_from_fluentd_receiver
```

For example:

```bash
$ oc create secret generic secure-forward --from-file=ca-bundle.crt=ca-for-fluentd-receiver/ca.crt --from-literal=shared_key=fluentd-receiver
```

4. Refresh the fluentd Pods to apply the secure-forward secret and secure-forward ConfigMap:

```
$ oc delete pod --selector logging-infra=fluentd
```

5. Configure the external log aggregator to accept messages securely from OpenShift Container Platform.

### 6.2. FORWARDING LOGS USING THE SYSLOG PROTOCOL

You can use the syslog protocol to send a copy of your logs to an external syslog server, instead of the default Elasticsearch log store. Note the following about this syslog protocol:

- uses syslog protocol (RFC 3164), not RFC 5424;
- does not support TLS and thus, is not secure;
- does not provide Kubernetes metadata, systemd data, or other metadata.

**NOTE**

This method for forwarding logs is deprecated in OpenShift Container Platform and will be replaced by the Log Forwarding API in a future release.

There are two versions of the syslog protocol:

- **out_syslog**: The non-buffered implementation, which communicates through UDP, does not buffer data and writes out results immediately.
- **out_syslog_buffered**: The buffered implementation, which communicates through TCP, buffers data into chunks.

To configure log forwarding using the syslog protocol, create a configuration file, called `syslog.conf`, with the information needed to forward the logs. Then use that file to create a ConfigMap called `syslog` in the openshift-logging namespace, which OpenShift Container Platform uses when forwarding the logs. You are responsible to configure your syslog server to receive the logs from OpenShift Container Platform.

**IMPORTANT**

Starting with the OpenShift Container Platform 4.3, the process for using the syslog protocol has changed. You now need to create a ConfigMap, as described below.

You can forward logs to multiple syslog servers by specifying separate `<store>` stanzas in the configuration file.
### Sample syslog.conf

```xml
<store>
@type syslog_buffered
remote_syslog rsyslogserver.openshift-logging.svc.cluster.local
port 514
hostname ${hostname}
remove_tag_prefix tag
tag_key ident,systemd.u.SYSLOG_IDENTIFIER
facility local0
severity info
use_record true
payload_key message
</store>
```

1. The **syslog** protocol, either: **syslog** or **syslog_buffered**.
2. The fully qualified domain name (FQDN) or IP address of the syslog server.
3. The port number to connect on. Defaults to **514**.
4. The name of the syslog server.
5. Removes the prefix from the tag. Defaults to " (empty).
6. The field to set the syslog key.
7. The syslog log facility or source.
8. The syslog log severity.
9. Determines whether to use the severity and facility from the record if available.
10. Optional. The key to set the payload of the syslog message. Defaults to **message**.

**NOTE**

Configuring the **payload_key** parameter prevents other parameters from being forwarded to the syslog.

### Sample syslog ConfigMap based on the sample **syslog.conf**

```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: syslog
  namespace: openshift-logging
data:
syslog.conf: |
  <store>
    @type syslog_buffered
    remote_syslog syslogserver.openshift-logging.svc.cluster.local
```
Procedure

To configure OpenShift Container Platform to forward logs using the syslog protocol:

1. Create a configuration file named syslog.conf that contains the following parameters within the <store> stanza:
   a. Specify the syslog protocol type:

   ```yaml
   @type syslog_buffered 1
   ```

   1 Specify the protocol to use, either: syslog or syslog_buffered.

   b. Configure the name, host, and port for your external syslog server:

   ```yaml
   remote_syslog <remote> 1
   port <number> 2
   hostname <name> 3
   ```

   1 Specify the FQDN or IP address of the syslog server.
   2 Specify the port of the syslog server.
   3 Specify a name for this syslog server.

   For example:

   Example output

   ```yaml
   remote_syslog syslogserver.openshift-logging.svc.cluster.local
   port 514
   hostname fluentd-server
   ```

   c. Configure the other syslog variables as needed:

   ```yaml
   remove_tag_prefix 1
   tag_key <key> 2
   facility <value> 3
   severity <value> 4
   use_record <value> 5
   payload_key message 6
   ```
1. Add this parameter to remove the tag field from the syslog prefix.
2. Specify the field to set the syslog key.
3. Specify the syslog log facility or source. For values, see RTF 3164.
4. Specify the syslog log severity. For values, see link: RTF 3164.
5. Specify true to use the severity and facility from the record if available. If true, the container_name, namespace_name, and pod_name are included in the output content.
6. Specify the key to set the payload of the syslog message. Defaults to message.

**Example output**

```yaml
facility local0
severity info
```

The configuration file appears similar to the following:

```xml
<store>
  @type syslog_buffered
  remote_syslog syslogserver.openshift-logging.svc.cluster.local
  port 514
  hostname ${hostname}
  tag_key ident,systemd.u.SYSLOG_IDENTIFIER
  facility local0
  severity info
  use_record false
</store>
```

2. Create a ConfigMap named syslog in the openshift-logging namespace from the configuration file:

```
$ oc create configmap syslog --from-file=syslog.conf -n openshift-logging
```

The Cluster Logging Operator redeployes the Fluentd Pods. If the Pods do not redeploy, you can delete the Fluentd Pods to force them to redeploy.

```
$ oc delete pod --selector logging-infra=fluentd
```

### 6.3. FORWARDING LOGS USING THE LOG FORWARDING API

The Log Forwarding API enables you to configure custom pipelines to send container and node logs to specific endpoints within or outside of your cluster. You can send logs by type to the internal OpenShift Container Platform Elasticsearch instance and to remote destinations not managed by OpenShift Container Platform cluster logging, such as an existing logging service, an external Elasticsearch cluster, external log aggregation solutions, or a Security Information and Event Management (SIEM) system.
IMPORTANT

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

See the Red Hat Technology Preview features support scope for more information.

You can send different types of logs to different systems allowing you to control who in your organization can access each type. Optional TLS support ensures that you can send logs using secure communication as required by your organization.

Using the Log Forwarding API is optional. If you want to forward logs to only the internal OpenShift Container Platform Elasticsearch instance, do not configure the Log Forwarding API.

6.3.1. Understanding the Log Forwarding API

Forwarding cluster logs by using the Log Forwarding API requires a combination of outputs and pipelines. These resources send logs to specific endpoints inside and outside of your OpenShift Container Platform cluster.

NOTE

If you want to use only the default internal OpenShift Container Platform Elasticsearch logstore, do not configure any outputs and pipelines.

An output is the destination for log data and a pipeline defines simple routing for one source to one or more outputs.

An output can be either:

- **elasticsearch** to forward logs to an external Elasticsearch 6 (all releases) cluster, specified by server name or FQDN, and/or the internal OpenShift Container Platform Elasticsearch logstore.

- **forward** to forward logs to an external log aggregation solution. This option uses the Fluentd forward protocols.

A pipeline associates the type of data to an output. A type of data you can forward is one of the following:

- **logs.app** - Container logs generated by user applications running in the cluster, except infrastructure container applications.

- **logs.infra** - Logs generated by both infrastructure components running in the cluster and OpenShift Container Platform nodes, such as journal logs. Infrastructure components are pods that run in the openshift*, kube*, or default projects.

- **logs.audit** - Logs generated by the node audit system (auditd), which are stored in the /var/log/audit/audit.log file, and the audit logs from the Kubernetes apiserver and the OpenShift apiserver.

To use the Log Forwarding API, you create a custom logforwarding configuration file with outputs and pipelines to send logs to destinations you specify.
Note the following:

- The internal OpenShift Container Platform Elasticsearch logstore does not provide secure storage for audit logs. We recommend you ensure that the system to which you forward audit logs is compliant with your organizational and governmental regulations and is properly secured. OpenShift Container Platform cluster logging does not comply with those regulations.

- An output supports TLS communication using a secret. Secrets must have keys of: `tls.crt`, `tls.key`, and `ca-bundle.crt` which point to the respective certificates for which they represent. Secrets must have the key `shared_key` for use when using forward in a secure manner.

- You are responsible for creating and maintaining any additional configurations that external destinations might require, such as keys and secrets, service accounts, port opening, or global proxy configuration.

The following example creates three outputs:

- the internal OpenShift Container Platform Elasticsearch logstore,
- an unsecured externally-managed Elasticsearch logstore,
- a secured external log aggregator using the `forward` protocol.

Three pipelines send:

- the application logs to the internal OpenShift Container Platform Elasticsearch logstore,
- the infrastructure logs to an external Elasticsearch logstore,
- the audit logs to the secured device over the `forward` protocol.

**Sample log forwarding outputs and pipelines**

```yaml
apiVersion: "logging.openshift.io/v1alpha1"
kind: "LogForwarding"
metadata:
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
  - name: elasticsearch
type: "elasticsearch"
endpoint: elasticsearch.openshift-logging.svc:9200
secret:
  name: fluentd
- name: elasticsearch-insecure
type: "elasticsearch"
endpoint: elasticsearch-insecure.messaging.svc.cluster.local
insecure: true
- name: secureforward-offcluster
type: "forward"
endpoint: https://secureforward.offcluster.com:24224
secret:
  name: secureforward
```
The name of the log forwarding CR must be `instance`.

Parameter to enable log forwarding. Set to `true` to enable log forwarding.

Configuration for the outputs.

A name to describe the output.

The type of output, either `elasticsearch` or `forward`.

The log forwarding endpoint, either the server name or FQDN. For the internal OpenShift Container Platform Elasticsearch logstore, specify `elasticsearch.openshift-logging.svc:9200`.

Optional name of the secret required by the endpoint for TLS communication. The secret must exist in the `openshift-logging` project.

Optional setting if the endpoint does not use a secret, resulting in insecure communication.

Configuration for the pipelines.

A name to describe the pipeline.

The source type, `logs.app`, `logs.infra`, or `logs.audit`.

The name of one or more outputs configured in the CR.

**Fluentd log handling when the external log aggregator is unavailable**

If your external logging aggregator becomes unavailable and cannot receive logs, Fluentd continues to collect logs and stores them in a buffer. When the log aggregator becomes available, log forwarding resumes, including the buffered logs. If the buffer fills completely, Fluentd stops collecting logs. OpenShift Container Platform rotates the logs and deletes them. You cannot adjust the buffer size or add a persistent volume claim (PVC) to the Fluentd daemon set or pods.

**NOTE**

Because the internal OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs, audit logs are not stored in the internal Elasticsearch instance by default. If you want to send the audit logs to the internal log store, for example to view the audit logs in Kibana, you must use the Log Forwarding API as described in [Forward audit logs to the log store](#).
6.3.2. Enabling the Log Forwarding API

You must enable the Log Forwarding API before you can forward logs using the API.

Procedure

To enable the Log Forwarding API:

1. Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

   ```bash
   $ oc edit ClusterLogging instance
   ```

2. Add the `clusterlogging.openshift.io/logforwardingtechpreview` annotation and set to `enabled`:

   ```yaml
   spec:
     collection:
       logs:
         type: "fluentd"
         fluentd: {}  
   ```

   Enables and disables the Log Forwarding API. Set to `enabled` to use log forwarding. To use the only the OpenShift Container Platform Elasticsearch instance, set to disabled or do not add the annotation.

   The `spec.collection` block must be defined to use Fluentd in the `ClusterLogging` CR.

6.3.3. Configuring log forwarding using the Log Forwarding API

To configure the Log Forwarding, edit the `ClusterLogging` custom resource (CR) to add the `clusterlogging.openshift.io/logforwardingtechpreview: enabled` annotation and create a `LogForwarding` custom resource to specify the outputs, pipelines, and enable log forwarding.

If you enable Log Forwarding, you should define a pipeline all for three source types: `logs.app`, `logs.infra`, and `logs.audit`. The logs from any undefined source type are dropped. For example, if you specify a pipeline for the `logs.app` and `log-audit` types, but do not specify a pipeline for the `logs.infra` type, `logs.infra` logs are dropped.

Procedure

To configure log forwarding using the API:

1. Create a `LogForwarding` CR YAML file similar to the following:

   ```yaml
   apiVersion: "logging.openshift.io/v1alpha1"
   ```
kind: "LogForwarding"
metadata:
  name: instance
namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
    - name: elasticsearch
      type: "elasticsearch"
      endpoint: elasticsearch.openshift-logging.svc:9200
      secret:
        name: fluentd
    - name: elasticsearch-insecure
      type: "elasticsearch"
      endpoint: elasticsearch-insecure.messaging.svc.cluster.local
      insecure: true
    - name: secureforward-offcluster
      type: "forward"
      endpoint: https://secureforward.offcluster.com:24224
      secret:
        name: secureforward
pipelines:
  - name: container-logs
    inputSource: logs.app
    outputRefs:
      - elasticsearch
      - secureforward-offcluster
    - name: infra-logs
      inputSource: logs.infra
      outputRefs:
        - elasticsearch-insecure
    - name: audit-logs
      inputSource: logs.audit
      outputRefs:
        - secureforward-offcluster

1. The name of the log forwarding CR must be **instance**.
2. The namespace for the log forwarding CR must be **openshift-logging**.
3. Set to **true** to disable the default log forwarding behavior.
4. Add one or more endpoints:
   - Specify the type of output, either **elasticsearch** or **forward**.
   - Enter a name for the output.
   - Enter the endpoint, either the server name, FQDN, or IP address. If the cluster–wide proxy using the CIDR annotation is enabled, the endpoint must be a server name or FQDN, not an IP Address. For the internal OpenShift Container Platform Elasticsearch instance, specify `elasticsearch.openshift-logging.svc:9200`.
   - Optional: Enter the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project.
• Specify **insecure: true** if the endpoint does not use a secret, resulting in insecure communication.

5. Add one or more pipelines:

• Enter a name for the pipeline

• Specify the source type: **logs.app**, **logs.infra**, or **logs.audit**.

• Specify the name of one or more outputs configured in the CR.

**NOTE**

If you set **disableDefaultForwarding: true** you must configure a pipeline and output for all three types of logs, application, infrastructure, and audit. If you do not specify a pipeline and output for a log type, those logs are not stored and will be lost.

2. Create the CR object:

```
$ oc create -f <file-name>.yaml
```

6.3.3.1. Example log forwarding custom resources

A typical Log Forwarding configuration would be similar to the following examples.

The following Log Forwarding custom resource sends all logs to a secured external Elasticsearch log store:

**Sample custom resource to forward to an Elasticsearch log store**

```yaml
apiVersion: logging.openshift.io/v1alpha1
g kind: LogForwarding
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
    - name: user-created-es
      type: elasticsearch
      endpoint: 'elasticsearch-server.openshift-logging.svc:9200'
      secret:
        name: pipelinesecret
  pipelines:
    - name: app-pipeline
      inputSource: logs.app
      outputRefs:
        - user-created-es
    - name: infra-pipeline
      inputSource: logs.infra
      outputRefs:
        - user-created-es
    - name: audit-pipeline
```
The following Log Forwarding custom resource sends all logs to a secured Fluentd instance using the Fluentd **forward** protocol.

**Sample custom resource to use the forward protocol**

```yaml
apiVersion: logging.openshift.io/v1alpha1
kind: LogForwarding
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
  - name: fluentd-created-by-user
    type: forward
    endpoint: 'fluentdserver.openshift-logging.svc:24224'
    secret:
      name: fluentdserver
  pipelines:
  - name: app-pipeline
    inputSource: logs.app
    outputRefs:
      - fluentd-created-by-user
  - name: infra-pipeline
    inputSource: logs.infra
    outputRefs:
      - fluentd-created-by-user
  - name: clo-default-audit-pipeline
    inputSource: logs.audit
    outputRefs:
      - fluentd-created-by-user
```

**6.3.4. Disabling the Log Forwarding API**

To disable the Log Forwarding API and to stop forwarding logs to the specified endpoints, remove the `metadata.annotations.clusterlogging.openshift.io/logforwardingtechpreview:enabled` parameter from the **ClusterLogging** CR and delete the **LogForwarding** CR. The container and node logs will be forwarded to the internal OpenShift Container Platform Elasticsearch instance.

**NOTE**

Setting `disableDefaultForwarding=false` prevents cluster logging from sending logs to the specified endpoints and to default internal OpenShift Container Platform Elasticsearch instance.

**Procedure**

To disable the Log Forwarding API:

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:
2. Remove the `clusterlogging.openshift.io/logforwardingtechpreview` annotation:

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  annotations:
    clusterlogging.openshift.io/logforwardingtechpreview: enabled
  name: "instance"
  namespace: "openshift-logging"
....
```

1. Remove this annotation.

3. Delete the Log Forwarding custom resource:

   ```bash
   $ oc delete LogForwarding instance -n openshift-logging
   ```
CHAPTER 7. COLLECTING AND STORING KUBERNETES EVENTS

The OpenShift Container Platform Event Router is a pod that watches Kubernetes events and logs them for collection by cluster logging. You must manually deploy the Event Router.

The Event Router collects events from all projects and writes them to STDOUT. Fluentd collects those events and forwards them into the OpenShift Container Platform Elasticsearch instance. Elasticsearch indexes the events to the infra index.

IMPORTANT
The Event Router adds additional load to Fluentd and can impact the number of other log messages that can be processed.

7.1. DEPLOYING AND CONFIGURING THE EVENT ROUTER

Use the following steps to deploy the Event Router into your cluster. You should always deploy the Event Router to the openshift-logging project to ensure it collects events from across the cluster.

The following Template object creates the service account, cluster role, and cluster role binding required for the Event Router. The template also configures and deploys the Event Router pod. You can use this template without making changes, or change the deployment object CPU and memory requests.

Prerequisites

- You need proper permissions to create service accounts and update cluster role bindings. For example, you can run the following template with a user that has the cluster-admin role.

- Cluster logging must be installed.

Procedure

1. Create a template for the Event Router:

```yaml
kind: Template
apiVersion: v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to cluster logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
- kind: ServiceAccount
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
- kind: ClusterRole
  apiVersion: v1
  metadata:
    name: event-reader
    namespace: ${NAMESPACE}
```
- apiGroups: [""]
  resources: ["events"]
  verbs: ["get", "watch", "list"]
- kind: ClusterRoleBinding
  apiVersion: v1
  metadata:
    name: event-reader-binding
  subjects:
    - kind: ServiceAccount
      name: eventrouter
      namespace: \${NAMESPACE}
      roleRef:
        kind: ClusterRole
        name: event-reader
- kind: ConfigMap
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: \${NAMESPACE}
  data:
    config.json: |
    {
      "sink": "stdout"
    }
- kind: Deployment
  apiVersion: apps/v1
  metadata:
    name: eventrouter
    namespace: \${NAMESPACE}
  labels:
    component: eventrouter
    logging-infra: eventrouter
    provider: openshift
  spec:
    selector:
      matchLabels:
        component: eventrouter
        logging-infra: eventrouter
        provider: openshift
    replicas: 1
    template:
      metadata:
        labels:
          component: eventrouter
          logging-infra: eventrouter
          provider: openshift
      name: eventrouter
    spec:
      serviceAccount: eventrouter
      containers:
        - name: kube-eventrouter
          image: \${IMAGE}
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: \${CPU}
Creates a Service Account in the **openshift-logging** project for the Event Router.

2. Creates a ClusterRole to monitor for events in the cluster.

3. Creates a ClusterRoleBinding to bind the ClusterRole to the service account.

4. Creates a config map in the **openshift-logging** project to generate the required `config.json` file.

5. Creates a deployment in the **openshift-logging** project to generate and configure the Event Router pod.

6. Specifies the minimum amount of memory to allocate to the Event Router pod. Defaults to 128Mi.

7. Specifies the minimum amount of CPU to allocate to the Event Router pod. Defaults to 100m.

8. Specifies the **openshift-logging** project to install objects in.

2. Use the following command to process and apply the template:

   ```bash
   $ oc process -f <templatefile> | oc apply -n openshift-logging -f -
   ```

   For example:

   ```bash
   $ oc process -f eventrouter.yaml | oc apply -n openshift-logging -f -
   ```

**Example output**

```
serviceaccount/logging-eventrouter created
clusterrole.authorization.openshift.io/event-reader created
```
3. Validate that the Event Router installed in the openshift-logging project:

a. View the new Event Router pod:

   $ oc get pods --selector component=eventrouter -o name -n openshift-logging

   **Example output**

   pod/cluster-logging-eventrouter-d649f97c8-qvv8r

b. View the events collected by the Event Router:

   $ oc logs <cluster_logging_eventrouter_pod> -n openshift-logging

   For example:

   $ oc logs cluster-logging-eventrouter-d649f97c8-qvv8r -n openshift-logging

   **Example output**

   ```json
   ```

You can also use Kibana to view events by creating an index pattern using the Elasticsearch infra index.
CHAPTER 8. UPDATING CLUSTER LOGGING

After updating the OpenShift Container Platform cluster from 4.4 to 4.5, you can then update the Elasticsearch Operator and Cluster Logging Operator from 4.4 to 4.5.

Cluster logging 4.5 introduces a new Elasticsearch version, Elasticsearch 6.8.1, and an enhanced security plug-in, Open Distro for Elasticsearch. The new Elasticsearch version introduces a new Elasticsearch data model, where the Elasticsearch data is indexed only by type: infrastructure, application, and audit. Previously, data was indexed by type (infrastructure and application) and project.

**IMPORTANT**

Because of the new data model, the update does not migrate existing custom Kibana index patterns and visualizations into the new version. You must re-create your Kibana index patterns and visualizations to match the new indices after updating.

Due to the nature of these changes, you are not required to update your cluster logging to 4.5. However, when you update to OpenShift Container Platform 4.6, you must update cluster logging to 4.6 at that time.

8.1. UPDATING CLUSTER LOGGING

After updating the OpenShift Container Platform cluster, you can update cluster logging from 4.4 to 4.5 by changing the subscription for the Elasticsearch Operator and the Cluster Logging Operator.

When you update:

- You must update the Elasticsearch Operator before updating the Cluster Logging Operator.

- You must update both the Elasticsearch Operator and the Cluster Logging Operator. Kibana is unusable when the Elasticsearch Operator has been updated but the Cluster Logging Operator has not been updated.

  If you update the Cluster Logging Operator before the Elasticsearch Operator, Kibana does not update and the Kibana custom resource (CR) is not created. To work around this problem, delete the Cluster Logging Operator pod. When the Cluster Logging Operator pod redeploys, the Kibana CR is created.

**IMPORTANT**

If your cluster logging version is prior to 4.4, you must upgrade cluster logging to 4.4 before updating to 4.5.

Prerequisites

- Update the OpenShift Container Platform cluster from 4.4 to 4.5.

- Make sure the cluster logging status is healthy:
  - All pods are **ready**.
  - The Elasticsearch cluster is healthy.

- Back up your Elasticsearch and Kibana data.
If your internal Elasticsearch instance uses persistent volume claims (PVCs), the PVCs must contain a `logging-cluster:elasticsearch` label. Without the label, during the upgrade the garbage collection process removes those PVCs and the Elasticsearch operator creates new PVCs.

- If you are updating from an OpenShift Container Platform version prior to version 4.4.30, you must manually add the label to the Elasticsearch PVCs.
  
  For example, you can use the following command to add a label to all the Elasticsearch PVCs:

  ```bash
  $ oc label pvc --all -n openshift-logging logging-cluster=elasticsearch
  ```

- After OpenShift Container Platform 4.4.30, the Elasticsearch operator automatically adds the label to the PVCs.

**Procedure**

1. Update the Elasticsearch Operator:

   a. From the web console, click **Operators → Installed Operators**.
   
   b. Select the `openshift-operators-redhat` project.
   
   c. Click the **Elasticsearch Operator**.
   
   d. Click **Subscription → Channel**.
   
   e. In the **Change Subscription Update Channel** window, select **4.5** and click **Save**.
   
   f. Wait for a few seconds, then click **Operators → Installed Operators**.
      
      The Elasticsearch Operator is shown as 4.5. For example:

      ```
      Elasticsearch Operator
      4.5.0-202007012112.p0 provided
      by Red Hat, Inc
      ```
      
      Wait for the **Status** field to report **Succeeded**.

2. Update the Cluster Logging Operator:

   a. From the web console, click **Operators → Installed Operators**.
   
   b. Select the `openshift-logging` project.
   
   c. Click the **Cluster Logging Operator**.
   
   d. Click **Subscription → Channel**.
   
   e. In the **Change Subscription Update Channel** window, select **4.5** and click **Save**.
   
   f. Wait for a few seconds, then click **Operators → Installed Operators**.
      
      The Cluster Logging Operator is shown as 4.5. For example:

      ```
      Cluster Logging
      4.5.0-202007012112.p0 provided
      by Red Hat, Inc
      ```
Wait for the **Status** field to report **Succeeded**.

3. Check the logging components:

   a. Ensure that all Elasticsearch pods are in the **Ready** status:

   ```
   $ oc get pod -n openshift-logging --selector component=elasticsearch
   
   **Example output**
   
<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>elasticsearch-cdm-1pbrl44l-1-55b7546f4c-mshhk</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>31m</td>
</tr>
<tr>
<td>elasticsearch-cdm-1pbrl44l-2-5c6d87589f-gx5hk</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>30m</td>
</tr>
<tr>
<td>elasticsearch-cdm-1pbrl44l-3-88df5d47-m45jc</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
</tbody>
</table>
   
   b. Ensure that the Elasticsearch cluster is healthy:

   ```
   
   ```
   $ oc exec -n openshift-logging -c elasticsearch elasticsearch-cdm-1pbrl44l-1-55b7546f4c-mshhk -- es_cluster_health
   
   ```
   
   ```
   {
   "cluster_name" : "elasticsearch",
   "status" : "green",
   }
   
   ...
   
   c. Ensure that the Elasticsearch cron jobs are created:

   ```
   $ oc project openshift-logging
   $ oc get cronjob
   
   **Example output**
   
<table>
<thead>
<tr>
<th>NAME</th>
<th>SCHEDULE</th>
<th>SUSPEND</th>
<th>ACTIVE</th>
<th>LAST SCHEDULE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>elasticsearch-im-app</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>56s</td>
</tr>
<tr>
<td>elasticsearch-im-audit</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>56s</td>
</tr>
<tr>
<td>elasticsearch-im-infra</td>
<td>*/15 * * * *</td>
<td>False</td>
<td>0</td>
<td>&lt;none&gt;</td>
<td>56s</td>
</tr>
</tbody>
</table>
   
   d. Verify that the log store is updated to 4.5 and the indices are **green**:

   ```
   $ oc exec -c elasticsearch <any_es_pod_in_the_cluster> -- indices
   
   ```
   
   Verifying that the output includes the **app-00000x, infra-00000x, audit-00000x, .security** indices.

   **Example 8.1. Sample output with indices in a green status**

   ```
   Tue Jun 30 14:30:54 UTC 2020
   health status index          uuid       pri rep
   docs.count     docs.deleted store.size pri.store.size
   green open     infra-000008
   bnBvUFEXTW92z3zWAzieQ  3 1  222195  0  289  144
   green open     infra-000004
   ```
Verify that the log collector is updated to 4.5:

```
$ oc get ds fluentd -o json | grep fluentd-init
```

Verify that the output includes a `fluentd-init` container:

```
"containerName": "fluentd-init"
```

Verify that the log visualizer is updated to 4.5 using the Kibana CRD:
Verify that the output includes a Kibana pod with the **ready** status:

**Example 8.2. Sample output with a ready Kibana pod**

```json
{
  "clusterCondition": {
    "kibana-5fdd766ffd-nb2jj": [
      {
        "lastTransitionTime": "2020-06-30T14:11:07Z",
        "reason": "ContainerCreating",
        "status": "True",
        "type": ""
      },
      {
        "lastTransitionTime": "2020-06-30T14:11:07Z",
        "reason": "ContainerCreating",
        "status": "True",
        "type": ""
      }
    ],
    "deployment": "kibana",
    "pods": {
      "failed": [],
      "notReady": [],
      "ready": []
    },
    "replicaSets": {
      "kibana-5fdd766ffd": ["replicas": 1]
    }
  }
}
```

g. Verify the Curator is updated to 4.5:

```bash
$ oc get cronjob -o name
```

- cronjob.batch/curator
- cronjob.batch/elasticsearch-delete-app
- cronjob.batch/elasticsearch-delete-audit
- cronjob.batch/elasticsearch-delete-infra
- cronjob.batch/elasticsearch-rollover-app
- cronjob.batch/elasticsearch-rollover-audit
- cronjob.batch/elasticsearch-rollover-infra

Verify that the output includes the **elasticsearch-delete-*** and **elasticsearch-rollover-*** indices.
8.1. Post-update tasks

If you use Kibana, after the Elasticsearch Operator and Cluster Logging Operator are fully updated to 4.5, you must recreate your Kibana index patterns and visualizations. Because of changes in the security plug-in, the cluster logging upgrade does not automatically create index patterns.

8.2. DEFINING KIBANA INDEX PATTERNS

An index pattern defines the Elasticsearch indices that you want to visualize. To explore and visualize data in Kibana, you must create an index pattern.

Prerequisites

- A user must have the cluster-admin role, the cluster-reader role, or both roles to view the infra and audit indices in Kibana. The default kubeadmin user has proper permissions to view these indices. If you can view the pods and logs in the default, kube- and openshift- projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

  ```
  $ oc auth can-i get pods/log -n <project>
  ```

  **Example output**

  ```
  yes
  ```

  **NOTE**

  The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the default output for audit logs.

- Elasticsearch documents must be indexed before you can create index patterns. This is done automatically, but it might take a few minutes in a new or updated cluster.

Procedure

To define index patterns and create visualizations in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher and select Logging.

2. Create your Kibana index patterns by clicking Management → Index Patterns → Create index pattern:

   - Each user must manually create index patterns when logging into Kibana the first time in order to see logs for their projects. Users must create an index pattern named app and use the @timestamp time field to view their container logs.

   - Each admin user must create index patterns when logged into Kibana the first time for the app, infra, and audit indices using the @timestamp time field.

3. Create Kibana Visualizations from the new index patterns.
9.1. VIEWING CLUSTER LOGGING STATUS

You can view the status of the Cluster Logging Operator and for a number of cluster logging components.

9.1.1. Viewing the status of the Cluster Logging Operator

You can view the status of your Cluster Logging Operator.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

1. Change to the openshift-logging project.

   $ oc project openshift-logging

2. To view the cluster logging status:

   a. Get the cluster logging status:

      $ oc get clusterlogging instance -o yaml

Example output

```yaml
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
...

status:
  collection:
    logs:
      fluentdStatus:
        daemonSet: fluentd
        nodes:
          fluentd-2rhqp: ip-10-0-169-13.ec2.internal
          fluentd-6fgjh: ip-10-0-165-244.ec2.internal
          fluentd-6l2ff: ip-10-0-128-218.ec2.internal
          fluentd-54nx5: ip-10-0-139-30.ec2.internal
          fluentd-flpnn: ip-10-0-147-228.ec2.internal
          fluentd-n2frh: ip-10-0-157-45.ec2.internal
        pods:
          failed: []
          notReady: []
          ready:
            - fluentd-2rhqp
            - fluentd-54nx5
            - fluentd-6fgjh
```
In the output, the cluster status fields appear in the **status** stanza.

1. **Information on the Fluentd pods.**

2. **Information on the Elasticsearch pods, including Elasticsearch cluster health, green, yellow, or red.**

3. **Information on the Kibana pods.**

4. **Information on the Fluentd pods.**
9.1.1.1. Example condition messages

The following are examples of some condition messages from the Status.Nodes section of the cluster logging instance.

A status message similar to the following indicates a node has exceeded the configured low watermark and no shard will be allocated to this node:

**Example output**

```
nodes:
- conditions:
    message: Disk storage usage for node is 27.5gb (36.74%). Shards will not be allocated on this node.
    reason: Disk Watermark Low
    status: "True"
    type: NodeStorage
deploymentName: example-elasticsearch-clientdatamaster-0-1
upgradeStatus: {}
```

A status message similar to the following indicates a node has exceeded the configured high watermark and shards will be relocated to other nodes:

**Example output**

```
nodes:
- conditions:
  - lastTransitionTime: 2019-03-15T16:04:45Z
    message: Disk storage usage for node is 27.5gb (36.74%). Shards will be relocated from this node.
    reason: Disk Watermark High
    status: "True"
    type: NodeStorage
deploymentName: cluster-logging-operator
upgradeStatus: {}
```

A status message similar to the following indicates the Elasticsearch node selector in the CR does not match any nodes in the cluster:

**Example output**

```
Elasticsearch Status:
  Shard Allocation Enabled: shard allocation unknown
Cluster:
  Active Primary Shards: 0
  Active Shards: 0
  Initializing Shards: 0
  Num Data Nodes: 0
  Num Nodes: 0
  Pending Tasks: 0
  Relocating Shards: 0
  Status: cluster health unknown
  Unassigned Shards: 0
Cluster Name: elasticsearch
```
A status message similar to the following indicates that the requested PVC could not bind to PV:

**Example output**

Node Conditions:
elasticsearch-cdm-mkkdys93-1:
  Last Transition Time: 2019-06-26T03:37:32Z
  Message: 0/5 nodes are available: 5 node(s) didn't match node selector.
  Reason: Unschedulable
  Status: True
  Type: Unschedulable

elasticsearch-cdm-mkkdys93-2:
Node Count: 2
Pods:
  Client:
    Failed:
    Not Ready:
      elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
      elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
    Ready:
    Data:
      Failed:
      Not Ready:
        elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
        elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
      Ready:
    Master:
      Failed:
      Not Ready:
        elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
        elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
      Ready:

A status message similar to the following indicates that the Fluentd pods cannot be scheduled because
the node selector did not match any nodes:

**Example output**

Node Conditions:
elasticsearch-cdm-mkkdys93-1:
  Last Transition Time: 2019-06-26T03:37:32Z
  Message: pod has unbound immediate PersistentVolumeClaims (repeated 5 times)
  Reason: Unschedulable
  Status: True
  Type: Unschedulable

A status message similar to the following indicates that the Fluentd pods cannot be scheduled because
the node selector did not match any nodes:
9.1.2. Viewing the status of cluster logging components

You can view the status for a number of cluster logging components.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

1. Change to the openshift-logging project.

   ```bash
   $ oc project openshift-logging
   ```

2. View the status of the cluster logging environment:

   ```bash
   $ oc describe deployment cluster-logging-operator
   ```

   **Example output**

   ```
   Name:                   cluster-logging-operator
   ....
   Conditions:             
   Type           Status  Reason
   ----           ------  ------
   Available      True    MinimumReplicasAvailable
   Progressing    True    NewReplicaSetAvailable
   ....
   Events:              
   Type    Reason             Age   From                   Message
   ----    ------             ----  ----                   -------
   Normal  ScalingReplicaSet 62m   deployment-controller  Scaled up replica set cluster-
   logging-operator-574b8987df to 1----
   ```

3. View the status of the cluster logging replica set:

   a. Get the name of a replica set:

      **Example output**

      ```bash
      $ oc get replicaset
      ```

      **Example output**

      ```
      NAME DESIRED CURRENT READY AGE
      ```
b. Get the status of the replica set:

   $ oc describe replicaset cluster-logging-operator-574b8987df

Example output

<table>
<thead>
<tr>
<th>Name:</th>
<th>cluster-logging-operator-574b8987df</th>
</tr>
</thead>
<tbody>
<tr>
<td>....</td>
<td></td>
</tr>
<tr>
<td>Replicas:</td>
<td>1 current / 1 desired</td>
</tr>
<tr>
<td>Pods Status:</td>
<td>1 Running / 0 Waiting / 0 Succeeded / 0 Failed</td>
</tr>
<tr>
<td>....</td>
<td></td>
</tr>
<tr>
<td>Events:</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Reason</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>Normal</td>
<td>SuccessfulCreate</td>
</tr>
</tbody>
</table>

9.2. VIEWING THE STATUS OF THE LOG STORE

You can view the status of the Elasticsearch Operator and for a number of Elasticsearch components.

9.2.1. Viewing the status of the log store

You can view the status of your log store.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

1. Change to the `openshift-logging` project.

   $ oc project openshift-logging

2. To view the status:

   a. Get the name of the log store instance:

      $ oc get Elasticsearch

Example output

```bash

```
b. Get the log store status:

```bash
$ oc get Elasticsearch <Elasticsearch-instance> -o yaml
```

For example:

```bash
$ oc get Elasticsearch elasticsearch -n openshift-logging -o yaml
```

The output includes information similar to the following:

**Example output**

```yaml
status: 1
  cluster: 2
    activePrimaryShards: 30
    activeShards: 60
    initializingShards: 0
    numDataNodes: 3
    numNodes: 3
    pendingTasks: 0
    relocatingShards: 0
    status: green
    unassignedShards: 0
    clusterHealth: ""
  conditions: [] 3
  nodes: 4
    - deploymentName: elasticsearch-cdm-zjf34ved-1
      upgradeStatus: {}
    - deploymentName: elasticsearch-cdm-zjf34ved-2
      upgradeStatus: {}
    - deploymentName: elasticsearch-cdm-zjf34ved-3
      upgradeStatus: {}
  pods: 5
    client:
      failed: []
      notReady: []
      ready:
        - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
        - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
        - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
    data:
      failed: []
      notReady: []
      ready:
        - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
        - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
        - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
    master:
      failed: []
      notReady: []
      ready:
```
In the output, the cluster status fields appear in the `status` stanza.

The status of the log store:

- The number of active primary shards.
- The number of active shards.
- The number of shards that are initializing.
- The number of log store data nodes.
- The total number of log store nodes.
- The number of pending tasks.
- The log store status: `green`, `red`, `yellow`.
- The number of unassigned shards.

Any status conditions, if present. The log store status indicates the reasons from the scheduler if a pod could not be placed. Any events related to the following conditions are shown:

- Container Waiting for both the log store and proxy containers.
- Container Terminated for both the log store and proxy containers.
- Pod unschedulable. Also, a condition is shown for a number of issues, see Example condition messages.

The log store nodes in the cluster, with `upgradeStatus`.

The log store client, data, and master pods in the cluster, listed under 'failed', `notReady` or `ready` state.

### 9.2.1.1. Example condition messages

The following are examples of some condition messages from the `Status` section of the Elasticsearch instance.

This status message indicates a node has exceeded the configured low watermark and no shard will be allocated to this node.

```
status:
  nodes:
    - conditions:
        message: Disk storage usage for node is 27.5gb (36.74%). Shards will be not
                be allocated on this node.
        reason: Disk Watermark Low
```
This status message indicates a node has exceeded the configured high watermark and shards will be relocated to other nodes.

```
status: "True"
type: NodeStorage
deploymentName: example-elasticsearch-cdm-0-1
upgradeStatus: {}
```

This status message indicates the log store node selector in the CR does not match any nodes in the cluster:

```
status:
  nodes:
    - conditions:
        - lastTransitionTime: 2019-04-10T02:26:24Z
          message: '0/8 nodes are available: 8 node(s) didn"t match node selector.'
          reason: Unschedulable
          status: "True"
          type: Unschedulable
```

This status message indicates that the log store CR uses a non-existent PVC.

```
status:
  nodes:
    - conditions:
        - lastTransitionTime: 2019-04-10T05:55:51Z
          message: pod has unbound immediate PersistentVolumeClaims (repeated 5 times)
          reason: Unschedulable
          status: True
          type: Unschedulable
```

This status message indicates that your log store cluster does not have enough nodes to support your log store redundancy policy.

```
status:
  clusterHealth: ""
  conditions:
    - lastTransitionTime: 2019-04-17T20:01:31Z
      message: Wrong RedundancyPolicy selected. Choose different RedundancyPolicy or add more nodes with data roles
```

OpenShift Container Platform 4.5 Logging
reason: Invalid Settings
status: "True"
type: InvalidRedundancy

This status message indicates your cluster has too many master nodes:

status:
  clusterHealth: green
conditions:
  - lastTransitionTime: '2019-04-17T20:12:34Z'
    message: >-
      Invalid master nodes count. Please ensure there are no more than 3 total
      nodes with master roles
    reason: Invalid Settings
    status: 'True'
    type: InvalidMasters

9.2.2. Viewing the status of the log store components

You can view the status for a number of the log store components.

Elasticsearch indices

You can view the status of the Elasticsearch indices.

1. Get the name of an Elasticsearch pod:

   $ oc get pods --selector component=elasticsearch -o name

   Example output

   pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
   pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
   pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7

2. Get the status of the indices:

   $ oc exec elasticsearch-cdm-4vjor49p-2-6d4d7db474-q2w7z -- indices

   Example output

   Defaulting container name to elasticsearch.
   Use 'oc describe pod/elasticsearch-cdm-4vjor49p-2-6d4d7db474-q2w7z -n openshift-logging' to see all of the containers in this pod.

   green open infra-000002 S4QAnnf1QP6NgCegfnrbQ
   3 1 119926 0 157 78
   green open audit-000001 _EQx77iQCSTzFOXtxRqFw
   3 1 0 0 0 0
   green open .security iDjscH7aSUGhldq0LheLBQ 1
   1 5 0 0 0
   green open .kibana_-377444158_kubeadmin yBywZ9GfSrkKebz5gWBZbjw 3 1 1 0 0 0
   green open infra-000001 z6Dpe__ORgiopEpW6YI44A
Log store pods
You can view the status of the pods that host the log store.

1. Get the name of a pod:

   $ oc get pods --selector component=elasticsearch -o name

Example output

   pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
   pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
   pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7

2. Get the status of a pod:

   $ oc describe pod elasticsearch-cdm-1godmszn-1-6f8495-vp4lw

The output includes the following status information:

Example output

   ....
   Status:    Running
   ....

Containers:
  elasticsearch:
      Container ID:  cri-o://b7d44e0a9ea486e27f47763f5bb4c39dfd2
      State:        Running
      Started:      Mon, 08 Jun 2020 10:17:56 -0400
      Ready:        True
      Restart Count:  0
      Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s period=5s #success=1 #failure=3

   ....
   proxy:
      Container ID:  cri-o://3f77032abaddbb1652c116278652908dc01860320b8a4741d06894b2f8f9aa1
      State:        Running
      Started:      Mon, 08 Jun 2020 10:18:38 -0400
      Ready:        True
      Restart Count:  0
Log storage pod deployment configuration
You can view the status of the log store deployment configuration.

1. Get the name of a deployment configuration:

   $ oc get deployment --selector component=elasticsearch -o name

   **Example output**

   deployment.extensions/elasticsearch-cdm-1gon-1
   deployment.extensions/elasticsearch-cdm-1gon-2
   deployment.extensions/elasticsearch-cdm-1gon-3

2. Get the deployment configuration status:

   $ oc describe deployment elasticsearch-cdm-1gon-1

   The output includes the following status information:

   **Example output**

   Containers:
   elasticsearch:
   Image:  registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3
   Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
   period=5s #success=1 #failure=3

   Conditions:
   Type             Status         Reason
   ----             ------         ------
   Progressing     Unknown       DeploymentPaused
   Available       True          MinimumReplicasAvailable

   Events:      <none>
Log store replica set
You can view the status of the log store replica set.

1. Get the name of a replica set:

   ```
   $ oc get replicaSet --selector component=elasticsearch -o name
   replicaset.extensions/elasticsearch-cdm-1gon-1-6f8495
   replicaset.extensions/elasticsearch-cdm-1gon-2-5769cf
   replicaset.extensions/elasticsearch-cdm-1gon-3-f66f7d
   ```

2. Get the status of the replica set:

   ```
   $ oc describe replicaSet elasticsearch-cdm-1gon-1-6f8495
   ```

   The output includes the following status information:

   **Example output**

   ```
   ....
   Containers:
   elasticsearch:
      Image:      registry.redhat.io/openshift4/ose-logging-elasticsearch6@sha256:4265742c7cdd85359140e2d7d703e4311b6497eec7676957f455d6908e7b1c25
      Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s period=5s #success=1 #failure=3
   ....
   Events:          <none>
   ```

### 9.3. UNDERSTANDING CLUSTER LOGGING ALERTS

All of the logging collector alerts are listed on the Alerting UI of the OpenShift Container Platform web console.

### 9.3.1. Viewing logging collector alerts

Alerts are shown in the OpenShift Container Platform web console, on the **Alerts** tab of the Alerting UI. Alerts are in one of the following states:

- **Firing**. The alert condition is true for the duration of the timeout. Click the **Options** menu at the end of the firing alert to view more information or silence the alert.

- **Pending**. The alert condition is currently true, but the timeout has not been reached.

- **Not Firing**. The alert is not currently triggered.

**Procedure**

To view cluster logging and other OpenShift Container Platform alerts:
1. In the OpenShift Container Platform console, click Monitoring → Alerting.

2. Click the Alerts tab. The alerts are listed, based on the filters selected.

Additional resources

- For more information on the Alerting UI, see Managing cluster alerts.

9.3.2. About logging collector alerts

The following alerts are generated by the logging collector. You can view these alerts in the OpenShift Container Platform web console, on the Alerts page of the Alerting UI.

Table 9.1. Fluentd Prometheus alerts

<table>
<thead>
<tr>
<th>Alert</th>
<th>Message</th>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FluentdErrorsHigh</td>
<td>In the last minute, &lt;value&gt; errors reported by fluentd &lt;instance&gt;.</td>
<td>Fluentd is reporting a higher number of issues than the specified number, default 10.</td>
<td>Critical</td>
</tr>
<tr>
<td>FluentdNodeDown</td>
<td>Prometheus could not scrape fluentd &lt;instance&gt; for more than 10m.</td>
<td>Fluentd is reporting that Prometheus could not scrape a specific Fluentd instance.</td>
<td>Critical</td>
</tr>
<tr>
<td>FluentdQueueLengthBurst</td>
<td>In the last minute, fluentd &lt;instance&gt; buffer queue length increased more than 32. Current value is &lt;value&gt;.</td>
<td>Fluentd is reporting that it is overwhelmed.</td>
<td>Warning</td>
</tr>
<tr>
<td>FluentdQueueLengthIncreasing</td>
<td>In the last 12h, fluentd &lt;instance&gt; buffer queue length constantly increased more than 1. Current value is &lt;value&gt;.</td>
<td>Fluentd is reporting queue usage issues.</td>
<td>Critical</td>
</tr>
</tbody>
</table>

9.3.3. About Elasticsearch alerting rules

You can view these alerting rules in Prometheus.

<table>
<thead>
<tr>
<th>Alert</th>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElasticsearchClusterNotHealthy</td>
<td>Cluster health status has been RED for at least 2m. Cluster does not accept writes, shards may be missing or master node hasn’t been elected yet.</td>
<td>critical</td>
</tr>
</tbody>
</table>
### 9.4. TROUBLESHOOTING THE LOG CURATOR

You can use information in this section for debugging log curation. Curator is used to remove data that is in the Elasticsearch index format prior to OpenShift Container Platform 4.5, and will be removed in a later release.

#### 9.4.1. Troubleshooting log curation

You can use information in this section for debugging log curation. For example, if curator is in a failed state, but the log messages do not provide a reason, you could increase the log level and trigger a new job, instead of waiting for another scheduled run of the cron job.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

To enable the Curator debug log and trigger next Curator iteration manually:

1. Enable debug log of Curator:

   ```bash
   $ oc set env cronjob/curator CURATOR_LOG_LEVEL=DEBUG
   CURATOR_SCRIPT_LOG_LEVEL=DEBUG
   ```
Specify the log level:

- **CRITICAL**. Curator displays only critical messages.
- **ERROR**. Curator displays only error and critical messages.
- **WARNING**. Curator displays only error, warning, and critical messages.
- **INFO**. Curator displays only informational, error, warning, and critical messages.
- **DEBUG**. Curator displays only debug messages, in addition to all of the above. The default value is INFO.

**NOTE**

Cluster logging uses the OpenShift Container Platform custom environment variable `CURATOR_SCRIPT_LOG_LEVEL` in OpenShift Container Platform wrapper scripts (`run.sh` and `convert.py`). The environment variable takes the same values as `CURATOR_LOG_LEVEL` for script debugging, as needed.

2. Trigger next curator iteration:

```bash
$ oc create job --from=cronjob/curator <job_name>
```

3. Use the following commands to control the cron job:

- **Suspend a cron job:**
  ```bash
  $ oc patch cronjob curator -p '{"spec":{"suspend":true}}'
  ```

- **Resume a cron job:**
  ```bash
  $ oc patch cronjob curator -p '{"spec":{"suspend":false}}'
  ```

- **Change a cron job schedule:**
  ```bash
  $ oc patch cronjob curator -p '{"spec":{"schedule":"0 0 * * *"}}'
  ```

  The **schedule** option accepts schedules in cron format.

9.5. COLLECTING LOGGING DATA FOR RED HAT SUPPORT

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

The **must-gather** tool enables you to collect diagnostic information for project-level resources, cluster-level resources, and each of the cluster logging components.

For prompt support, supply diagnostic information for both OpenShift Container Platform and cluster logging.
NOTE

Do not use the \texttt{hack/logging-dump.sh} script. The script is no longer supported and does not collect data.

9.5.1. About the must-gather tool

The \texttt{oc adm must-gather} CLI command collects the information from your cluster that is most likely needed for debugging issues.

For your cluster logging environment, \texttt{must-gather} collects the following information:

- project-level resources, including pods, configuration maps, service accounts, roles, role bindings, and events at the project level
- cluster-level resources, including nodes, roles, and role bindings at the cluster level
- cluster logging resources in the \texttt{openshift-logging} and \texttt{openshift-operators-redhat} namespaces, including health status for the log collector, the log store, the curator, and the log visualizer

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

9.5.2. Prerequisites

- Cluster logging and Elasticsearch must be installed.

9.5.3. Collecting cluster logging data

You can use the \texttt{oc adm must-gather} CLI command to collect information about your cluster logging environment.

Procedure

To collect cluster logging information with \texttt{must-gather}:

1. Navigate to the directory where you want to store the \texttt{must-gather} information.

2. Run the \texttt{oc adm must-gather} command against the cluster logging image:

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

The \texttt{must-gather} tool creates a new directory that starts with \texttt{must-gather.local} within the current directory. For example: \texttt{must-gather.local.415724594708210408}.

3. Create a compressed file from the \texttt{must-gather} directory that was just created. For example, on a computer that uses a Linux operating system, run the following command:

\[
\texttt{tar -cvaf must-gather.tar.gz must-gather.local.415724594708210408}
\]

4. Attach the compressed file to your support case on the \texttt{Red Hat Customer Portal}.
You can remove cluster logging from your OpenShift Container Platform cluster.

### 10.1. UNINSTALLING CLUSTER LOGGING FROM OPENShift CONTAINER PLATFORM

You can stop log aggregation by deleting the `ClusterLogging` custom resource (CR). However, after deleting the CR there are other cluster logging components that remain, which you can optionally remove.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

To remove cluster logging:

1. Use the OpenShift Container Platform web console to remove the `ClusterLogging` CR:
   a. Switch to the Administration → Custom Resource Definitions page.
   c. On the Custom Resource Definition Details page, click `Instances`.
   d. Click the Options menu next to the instance and select **Delete ClusterLogging**.

2. Optional: Delete the custom resource definitions (CRD):
   a. Switch to the Administration → Custom Resource Definitions page.
   b. Click the Options menu next to `ClusterLogging` and select **Delete Custom Resource Definition**.
   c. Click the Options menu next to `Elasticsearch` and select **Delete Custom Resource Definition**.
   d. Click the Options menu next to `LogForwarding` and select **Delete Custom Resource Definition**.

3. Optional: Remove the Cluster Logging Operator and Elasticsearch Operator:
   a. Switch to the Operators → Installed Operators page.
   b. Select the `openshift-logging` project.
c. Click the Options menu next to the Cluster Logging Operator and select Uninstall Operator.

d. Select the openshift-operators-redhat project.

e. Click the Options menu next to the Elasticsearch Operator and select Uninstall Operator.

4. Optional: Remove the Cluster Logging and Elasticsearch projects.

   a. Switch to the Home → Projects page.

   b. Click the Options menu next to the openshift-logging project and select Delete Project.

   c. Confirm the deletion by typing openshift-logging in the dialog box and click Delete.

   d. Click the Options menu next to the openshift-operators-redhat project and select Delete Project.

   **IMPORTANT**

   Do not delete the openshift-operators-redhat project if other global operators are installed in this namespace.

   e. Confirm the deletion by typing openshift-operators-redhat in the dialog box and click Delete.
CHAPTER 11. EXPORTED FIELDS

These are the fields exported by the logging system and available for searching from Elasticsearch and Kibana. Use the full, dotted field name when searching. For example, for an Elasticsearch /_search URL, to look for a Kubernetes pod name, use /_search?q=kubernetes.pod_name:name-of-my-pod.

The following sections describe fields that may not be present in your logging store. Not all of these fields are present in every record. The fields are grouped in the following categories:

- exported-fields-Default
- exported-fields-systemd
- exported-fields-kubernetes
- exported-fields-pipeline_metadata
- exported-fields-ovirt
- exported-fields-aushape
- exported-fields-tlog

11.1. DEFAULT EXPORTED FIELDS

These are the default fields exported by the logging system and available for searching from Elasticsearch and Kibana. The default fields are Top Level and collectd*

Top Level Fields
The top level fields are common to every application, and may be present in every record. For the Elasticsearch template, top level fields populate the actual mappings of default in the template’s mapping section.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@timestamp</td>
<td>The UTC value marking when the log payload was created, or when the log payload was first collected if the creation time is not known. This is the log processing pipeline’s best effort determination of when the log payload was generated. Add the @ prefix convention to note a field as being reserved for a particular use. With Elasticsearch, most tools look for @timestamp by default. For example, the format would be 2015-01-24 14:06:05.071000.</td>
</tr>
<tr>
<td>geoip</td>
<td>This is geo-ip of the machine.</td>
</tr>
<tr>
<td>hostname</td>
<td>The hostname is the fully qualified domain name (FQDN) of the entity generating the original payload. This field is an attempt to derive this context. Sometimes the entity generating it knows the context. While other times that entity has a restricted namespace itself, which is known by the collector or normalizer.</td>
</tr>
<tr>
<td>ipaddr4</td>
<td>The IP address V4 of the source server, which can be an array.</td>
</tr>
<tr>
<td>ipaddr6</td>
<td>The IP address V6 of the source server, if available.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>level</td>
<td>The logging level as provided by rsyslog (severitytext property), python’s logging module. Possible values are as listed at <code>misc/sys/syslog.h</code> plus trace and unknown. For example, “alert crit debug emerg err info notice trace unknown warning”. Note that trace is not in the <code>syslog.h</code> list but many applications use it. You should only use unknown when the logging system gets a value it does not understand, and note that it is the highest level. Consider trace as higher or more verbose, than debug. error is deprecated, use err. Convert panic to emerg. Convert warn to warning. Numeric values from <code>syslog/journal PRIORITY</code> can usually be mapped using the priority values as listed at <code>misc/sys/syslog.h</code>. Log levels and priorities from other logging systems should be mapped to the nearest match. See python logging for an example.</td>
</tr>
<tr>
<td>message</td>
<td>A typical log entry message, or payload. It can be stripped of metadata pulled out of it by the collector or normalizer, that is UTF-8 encoded.</td>
</tr>
<tr>
<td>pid</td>
<td>This is the process ID of the logging entity, if available.</td>
</tr>
<tr>
<td>service</td>
<td>The name of the service associated with the logging entity, if available. For example, the <code>syslog APP-NAME</code> property is mapped to the service field.</td>
</tr>
<tr>
<td>tags</td>
<td>Optionally provided operator defined list of tags placed on each log by the collector or normalizer. The payload can be a string with whitespace-delimited string tokens, or a JSON list of string tokens.</td>
</tr>
<tr>
<td>file</td>
<td>Optional path to the file containing the log entry local to the collector. <strong>TODO</strong> analyzer for file paths.</td>
</tr>
<tr>
<td>offset</td>
<td>The offset value can represent bytes to the start of the log line in the file (zero or one based), or log line numbers (zero or one based), as long as the values are strictly monotonically increasing in the context of a single log file. The values are allowed to wrap, representing a new version of the log file (rotation).</td>
</tr>
<tr>
<td>namespace_name</td>
<td>Associate this record with the namespace that shares it’s name. This value will not be stored, but it is used to associate the record with the appropriate namespace for access control and visualization. Normally this value will be given in the tag, but if the protocol does not support sending a tag, this field can be used. If this field is present, it will override the namespace given in the tag or in <code>kubernetes.namespace_name</code>.</td>
</tr>
<tr>
<td>namespace_uuid</td>
<td>This is the uuid associated with the namespace_name. This value will not be stored, but is used to associate the record with the appropriate namespace for access control and visualization. If this field is present, it will override the uuid given in <code>kubernetes.namespace_uuid</code>. This will also cause the Kubernetes metadata lookup to be skipped for this log record.</td>
</tr>
</tbody>
</table>
**collectd Fields**  
The following fields represent namespace metrics metadata.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.interval</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> interval.</td>
</tr>
<tr>
<td><code>collectd.plugin</code></td>
<td>type: string</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.plugin_instance</code></td>
<td>type: string</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> plugin_instance.</td>
</tr>
<tr>
<td><code>collectd.type_instance</code></td>
<td>type: string</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd type_instance</code>.</td>
</tr>
<tr>
<td><code>collectd.type</code></td>
<td>type: string</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> type.</td>
</tr>
<tr>
<td><code>collectd.dstypes</code></td>
<td>type: string</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> dstypes.</td>
</tr>
</tbody>
</table>

**collectd.processes Fields**  
The following field corresponds to the `collectd` processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.processes.ps_state</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd ps_state</code> type of processes plug-in.</td>
</tr>
</tbody>
</table>

**collectd.processes.ps_disk_ops Fields**  
The `collectd ps_disk_ops` type of processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.processes.ps_disk_ops.read</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.processes.ps_disk_ops.write</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td><code>collectd.processes.ps_vm</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd ps_vm</strong> type of processes plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.processes.ps_rss</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd ps_rss</strong> type of processes plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.processes.ps_data</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd ps_data</strong> type of processes plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.processes.ps_code</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd ps_code</strong> type of processes plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.processes.ps_stacksize</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd ps_stacksize</strong> type of processes plug-in.</td>
<td></td>
</tr>
</tbody>
</table>

**collectd.processes.ps_cputime Fields**
The **collectd ps_cputime** type of processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.processes.ps_cputime.user</code></td>
<td>type: float</td>
</tr>
<tr>
<td>TODO</td>
<td></td>
</tr>
<tr>
<td><code>collectd.processes.ps_cputime.syst</code></td>
<td>type: float</td>
</tr>
<tr>
<td>TODO</td>
<td></td>
</tr>
</tbody>
</table>

**collectd.processes.ps_count Fields**
The **collectd ps_count** type of processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.processes.ps_count.processes</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>TODO</td>
<td></td>
</tr>
<tr>
<td><code>collectd.processes.ps_count.threads</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>TODO</td>
<td></td>
</tr>
</tbody>
</table>

**collectd.processes.ps_pagefaults Fields**

The `collectd ps_pagefaults` type of processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.processes.ps_pagefaults.majflt</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.processes.ps_pagefaults.minflt</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

`collectd.processes.ps_disk_octets` Fields
The `collectd ps_disk_octets` type of processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.processes.ps_disk_octets.read</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.processes.ps_disk_octets.write</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.processes.fork_rate</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd fork_rate</code> type of processes plug-in.</td>
</tr>
</tbody>
</table>

`collectd.disk` Fields
Corresponds to `collectd` disk plug-in.

`collectd.disk.disk_merged` Fields
The `collectd disk_merged` type of disk plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_merged.read</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.disk.disk_merged.write</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

`collectd.disk.disk_octets` Fields
The `collectd disk_octets` type of disk plug-in.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_octets.read</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.disk.disk_octets.write</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.disk.disk_time Fields**
The `collectd disk_time` type of disk plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_time.read</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.disk.disk_time.write</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.disk.disk_ops Fields**
The `collectd disk_ops` type of disk plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_ops.read</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.disk.disk_ops.write</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.disk.pending_operations</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd pending_operations</code> type of disk plug-in.</td>
</tr>
</tbody>
</table>

**collectd.disk.disk_io_time Fields**
The `collectd disk_io_time` type of disk plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_io_time.io_time</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>collectd.disk.disk_io_time.weighted_io_time</code></td>
<td>type: float&lt;br&gt;TODO</td>
</tr>
</tbody>
</table>

**collectd.interface Fields**
Corresponds to the **collectd** interface plug-in.

**collectd.interface.if_octets Fields**
The **collectd if_octets** type of interface plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.interface.if_octets.rx</code></td>
<td>type: float&lt;br&gt;TODO</td>
</tr>
<tr>
<td><code>collectd.interface.if_octets.tx</code></td>
<td>type: float&lt;br&gt;TODO</td>
</tr>
</tbody>
</table>

**collectd.interface.if_packets Fields**
The **collectd if_packets** type of interface plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.interface.if_packets.rx</code></td>
<td>type: float&lt;br&gt;TODO</td>
</tr>
<tr>
<td><code>collectd.interface.if_packets.tx</code></td>
<td>type: float&lt;br&gt;TODO</td>
</tr>
</tbody>
</table>

**collectd.interface.if_errors Fields**
The **collectd if_errors** type of interface plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.interface.if_errors.rx</code></td>
<td>type: float&lt;br&gt;TODO</td>
</tr>
<tr>
<td><code>collectd.interface.if_errors.tx</code></td>
<td>type: float&lt;br&gt;TODO</td>
</tr>
</tbody>
</table>
**collectd.interface.if_dropped Fields**
The *collectd if_dropped* type of interface plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.interface.if_dropped.rx</td>
<td>type: float, TODO</td>
</tr>
<tr>
<td>collectd.interface.if_dropped.tx</td>
<td>type: float, TODO</td>
</tr>
</tbody>
</table>

**collectd.virt Fields**
Corresponds to *collectd* virt plug-in.

**collectd.virt.if_octets Fields**
The *collectd if_octets* type of virt plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.if_octets.rx</td>
<td>type: float, TODO</td>
</tr>
<tr>
<td>collectd.virt.if_octets.tx</td>
<td>type: float, TODO</td>
</tr>
</tbody>
</table>

**collectd.virt.if_packets Fields**
The *collectd if_packets* type of virt plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.if_packets.rx</td>
<td>type: float, TODO</td>
</tr>
<tr>
<td>collectd.virt.if_packets.tx</td>
<td>type: float, TODO</td>
</tr>
</tbody>
</table>

**collectd.virt.if_errors Fields**
The *collectd if_errors* type of virt plug-in.
### collectd.virt.if_errors

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.if_errors.rx</td>
<td>type: float</td>
</tr>
<tr>
<td>collectd.virt.if_errors.tx</td>
<td>type: float</td>
</tr>
</tbody>
</table>

### collectd.virt.if_dropped

The `collectd if_dropped` type of virt plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.if_dropped.rx</td>
<td>type: float</td>
</tr>
<tr>
<td>collectd.virt.if_dropped.tx</td>
<td>type: float</td>
</tr>
</tbody>
</table>

### collectd.virt.disk_ops

The `collectd disk_ops` type of virt plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.disk_ops.read</td>
<td>type: float</td>
</tr>
<tr>
<td>collectd.virt.disk_ops.write</td>
<td>type: float</td>
</tr>
</tbody>
</table>

### collectd.virt.disk_octets

The `collectd disk_octets` type of virt plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.disk_octets.read</td>
<td>type: float</td>
</tr>
<tr>
<td>collectd.virt.disk_octets.write</td>
<td>type: float</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>collectd.virt.memory</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd</em> memory type of virt plug-in.</td>
</tr>
<tr>
<td>collectd.virt.vcpu_vcpu</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd vcpu_vcpu</em> type of virt plug-in.</td>
</tr>
<tr>
<td>collectd.virt.vcpu_total</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd vcpu_total</em> type of virt plug-in.</td>
</tr>
</tbody>
</table>

**collectd.CPU Fields**
Corresponds to the *collectd* CPU plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.CPU.percent</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd</em> type percent of plug-in CPU.</td>
</tr>
</tbody>
</table>

**collectd.df Fields**
Corresponds to the *collectd df* plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.df.df_complex</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd</em> type df_complex of plug-in df.</td>
</tr>
<tr>
<td>collectd.df.percent_bytes</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd</em> type percent_bytes of plug-in df.</td>
</tr>
</tbody>
</table>

**collectd.entropy Fields**
Corresponds to the *collectd* entropy plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.entropy.entropy</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd</em> entropy type of entropy plug-in.</td>
</tr>
</tbody>
</table>

**collectd.memory Fields**
Corresponds to the *collectd* memory plug-in.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.memory.memory</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> memory type of memory plug-in.</td>
</tr>
<tr>
<td><code>collectd.memory.percent</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> percent type of memory plug-in.</td>
</tr>
</tbody>
</table>

**collectd.swap** Fields
Corresponds to the `collectd` swap plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.swap.swap</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> swap type of swap plug-in.</td>
</tr>
<tr>
<td><code>collectd.swap.swap_io</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd swap_io</code> type of swap plug-in.</td>
</tr>
</tbody>
</table>

**collectd.load** Fields
Corresponds to the `collectd` load plug-in.

**collectd.load.load** Fields
The `collectd` load type of load plug-in

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.load.load.shortterm</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.load.load.midterm</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.load.load.longterm</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.aggregation** Fields
Corresponds to `collectd` aggregation plug-in.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.aggregation.perc</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**`collectd.statsd` Fields**

Corresponds to `collectd statsd` plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.statsd.host_cpu</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> CPU type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_elapsed_time</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd elapsed_time</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_memory</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> memory type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_nic_speed</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd nic_speed</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_nic_rx</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd nic_rx</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_nic_tx</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd nic_tx</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_nic_rx_dropped</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd nic_rx_dropped</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_nic_tx_dropped</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd nic_tx_dropped</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_nic_rx_errors</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd nic_rx_errors</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.host_nic_tx_errors</code></td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd nic_tx_errors</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| `collectd.statsd.host_storage` | type: integer  
The `collectd` storage type of `statsd` plug-in.                          |
| `collectd.statsd.host_swap`   | type: integer  
The `collectd` swap type of `statsd` plug-in.                            |
| `collectd.statsd.host_vds`    | type: integer  
The `collectd` VDSM type of `statsd` plug-in.                            |
| `collectd.statsd.host_vms`    | type: integer  
The `collectd` VMS type of `statsd` plug-in.                             |
| `collectd.statsd.vm_nic_tx_dropped` | type: integer  
The `collectd nic_tx_dropped` type of `statsd` plug-in.          |
| `collectd.statsd.vm_nic_rx_bytes` | type: integer  
The `collectd nic_rx_bytes` type of `statsd` plug-in.        |
| `collectd.statsd.vm_nic_tx_bytes` | type: integer  
The `collectd nic_tx_bytes` type of `statsd` plug-in.      |
| `collectd.statsd.vm_balloon_min` | type: integer  
The `collectd balloon_min` type of `statsd` plug-in.  |
| `collectd.statsd.vm_balloon_max` | type: integer  
The `collectd balloon_max` type of `statsd` plug-in.  |
| `collectd.statsd.vm_balloon_target` | type: integer  
The `collectd balloon_target` type of `statsd` plug-in. |
| `collectd.statsd.vm_balloon_cur` | type: integer  
The `collectd balloon_cur` type of `statsd` plug-in.   |
| `collectd.statsd.vm_cpu_sys`  | type: integer  
The `collectd cpu_sys` type of `statsd` plug-in.                           |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.statsd.vm_cpu_usage</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd cpu_usage</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_read_ops</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_read_ops</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_write_ops</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_write_ops</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_flush_latency</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_flush_latency</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_apparent_size</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_apparent_size</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_write_bytes</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_write_bytes</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_write_rate</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_write_rate</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_true_size</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_true_size</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_read_rate</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_read_rate</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_write_latency</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_write_latency</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_read_latency</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_read_latency</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
<tr>
<td><code>collectd.statsd.vm_disk_read_bytes</code></td>
<td>type: integer</td>
</tr>
<tr>
<td>The <code>collectd disk_read_bytes</code> type of <code>statsd</code> plug-in.</td>
<td></td>
</tr>
</tbody>
</table>
## Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `collectd.statsd.vm_nic_rx_dropped` | type: integer  
The `collectd` `nic_rx_dropped` type of `statsd` plug-in. |
| `collectd.statsd.vm_cpu_user` | type: integer  
The `collectd` `cpu_user` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_rx_errors` | type: integer  
The `collectd` `nic_rx_errors` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_tx_errors` | type: integer  
The `collectd` `nic_tx_errors` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_speed` | type: integer  
The `collectd` `nic_speed` type of `statsd` plug-in. |

### collectd.postgresql Fields

Corresponds to `collectd postgresql` plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `collectd.postgresql.pg_n_tup_g` | type: integer  
The `collectd` type `pg_n_tup_g` of plug-in `postgresql`. |
| `collectd.postgresql.pg_n_tup_c` | type: integer  
The `collectd` type `pg_n_tup_c` of plug-in `postgresql`. |
| `collectd.postgresql.pg_numbackends` | type: integer  
The `collectd` type `pg_numbackends` of plug-in `postgresql`. |
| `collectd.postgresql.pg_xact` | type: integer  
The `collectd` type `pg_xact` of plug-in `postgresql`. |
| `collectd.postgresql.pg_db_size` | type: integer  
The `collectd` type `pg_db_size` of plug-in `postgresql`. |
| `collectd.postgresql.pg_blks` | type: integer  
The `collectd` type `pg_blks` of plug-in `postgresql`. |
11.2. SYSTEMD EXPORTED FIELDS

These are the `systemd` fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana.

Contains common fields specific to `systemd` journal. Applications may write their own fields to the journal. These will be available under the `systemd.u` namespace. `RESULT` and `UNIT` are two such fields.

`systemd.k` Fields
The following table contains `systemd` kernel-specific metadata.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>systemd.k.KERNEL_DEVICE</code></td>
<td><code>systemd.k.KERNEL_DEVICE</code> is the kernel device name.</td>
</tr>
<tr>
<td><code>systemd.k.KERNEL_SUBSYSTEM</code></td>
<td><code>systemd.k.KERNEL_SUBSYSTEM</code> is the kernel subsystem name.</td>
</tr>
<tr>
<td><code>systemd.k.UDEV_DEVLINK</code></td>
<td><code>systemd.k.UDEV_DEVLINK</code> includes additional symlink names that point to the node.</td>
</tr>
<tr>
<td><code>systemd.k.UDEV_DEVNODE</code></td>
<td><code>systemd.k.UDEV_DEVNODE</code> is the node path of the device.</td>
</tr>
<tr>
<td><code>systemd.k.UDEV_SYSNAME</code></td>
<td><code>systemd.k.UDEV_SYSNAME</code> is the kernel device name.</td>
</tr>
</tbody>
</table>

`systemd.t` Fields
`systemd.t` Fields are trusted journal fields, fields that are implicitly added by the journal, and cannot be altered by client code.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>systemd.t.AUDIT_LOGINUID</code></td>
<td><code>systemd.t.AUDIT_LOGINUID</code> is the user ID for the journal entry process.</td>
</tr>
<tr>
<td><code>systemd.t.BOOT_ID</code></td>
<td><code>systemd.t.BOOT_ID</code> is the kernel boot ID.</td>
</tr>
<tr>
<td><code>systemd.t.AUDIT_SESSION</code></td>
<td><code>systemd.t.AUDIT_SESSION</code> is the session for the journal entry process.</td>
</tr>
<tr>
<td><code>systemd.t.CAP_EFFECTIVE</code></td>
<td><code>systemd.t.CAP_EFFECTIVE</code> represents the capabilities of the journal entry process.</td>
</tr>
<tr>
<td><code>systemd.t.CMDLINE</code></td>
<td><code>systemd.t.CMDLINE</code> is the command line of the journal entry process.</td>
</tr>
<tr>
<td><code>systemd.t.COMM</code></td>
<td><code>systemd.t.COMM</code> is the name of the journal entry process.</td>
</tr>
</tbody>
</table>
### CHAPTER 11. EXPORTED FIELDS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>systemd.t.EXE</td>
<td><strong>systemd.t.EXE</strong> is the executable path of the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.GID</td>
<td><strong>systemd.t.GID</strong> is the group ID for the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.HOSTNAME</td>
<td><strong>systemd.t.HOSTNAME</strong> is the name of the host.</td>
</tr>
<tr>
<td>systemd.t.MACHINE_ID</td>
<td><strong>systemd.t.MACHINE_ID</strong> is the machine ID of the host.</td>
</tr>
<tr>
<td>systemd.t.PID</td>
<td><strong>systemd.t.PID</strong> is the process ID for the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.SELINUX_CONTEXT</td>
<td><strong>systemd.t.SELINUX_CONTEXT</strong> is the security context, or label, for the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.SOURCE_REALTIME_TIMESTAMP</td>
<td><strong>systemd.t.SOURCE_REALTIME_TIMESTAMP</strong> is the earliest and most reliable timestamp of the message. This is converted to RFC 3339 NS format.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_CGROUP</td>
<td><strong>systemd.t.SYSTEMD_CGROUP</strong> is the <strong>systemd</strong> control group path.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_OWNER_UID</td>
<td><strong>systemd.t.SYSTEMD_OWNER_UID</strong> is the owner ID of the session.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_SESSION</td>
<td><strong>systemd.t.SYSTEMD_SESSION</strong>, if applicable, is the <strong>systemd</strong> session ID.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_SLICE</td>
<td><strong>systemd.t.SYSTEMD_SLICE</strong> is the slice unit of the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_UNIT</td>
<td><strong>systemd.t.SYSTEMD_UNIT</strong> is the unit name for a session.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_USER_UNIT</td>
<td><strong>systemd.t.SYSTEMD_USER_UNIT</strong>, if applicable, is the user unit name for a session.</td>
</tr>
<tr>
<td>systemd.t.TRANSPORT</td>
<td><strong>systemd.t.TRANSPORT</strong> is the method of entry by the journal service. This includes, <strong>audit</strong>, <strong>driver</strong>, <strong>syslog</strong>, <strong>journal</strong>, <strong>stdout</strong>, and <strong>kernel</strong>.</td>
</tr>
<tr>
<td>systemd.t.UID</td>
<td><strong>systemd.t.UID</strong> is the user ID for the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.SYSLOG_FACILITY</td>
<td><strong>systemd.t.SYSLOG_FACILITY</strong> is the field containing the facility, formatted as a decimal string, for <strong>syslog</strong>.</td>
</tr>
<tr>
<td>systemd.t.SYSLOG_IDENTIFIER</td>
<td><strong>systemd.t.SYSLOG_IDENTIFIER</strong> is the identifier for <strong>syslog</strong>.</td>
</tr>
</tbody>
</table>
**systemd.t** Fields

**systemd.t** fields are directly passed from clients and stored in the journal.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>systemd.t.SYSLOG_PID</td>
<td><strong>SYSLOG_PID</strong> is the client process ID for <strong>syslog</strong>.</td>
</tr>
</tbody>
</table>

**systemd.u** Fields

**systemd.u** fields are directly passed from clients and stored in the journal.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>systemd.u.CODE_FILE</td>
<td><strong>systemd.u.CODE_FILE</strong> is the code location containing the filename of the source.</td>
</tr>
<tr>
<td>systemd.u.CODE_FUNCTION</td>
<td><strong>systemd.u.CODE_FUNCTION</strong> is the code location containing the function of the source.</td>
</tr>
<tr>
<td>systemd.u.CODE_LINE</td>
<td><strong>systemd.u.CODE_LINE</strong> is the code location containing the line number of the source.</td>
</tr>
<tr>
<td>systemd.u.ERRNO</td>
<td><strong>systemd.u.ERRNO</strong>, if present, is the low-level error number formatted in numeric value, as a decimal string.</td>
</tr>
<tr>
<td>systemd.u.MESSAGE_ID</td>
<td><strong>systemd.u.MESSAGE_ID</strong> is the message identifier ID for recognizing message types.</td>
</tr>
<tr>
<td>systemd.u.RESULT</td>
<td>For private use only.</td>
</tr>
<tr>
<td>systemd.u.UNIT</td>
<td>For private use only.</td>
</tr>
</tbody>
</table>

**11.3. KUBERNETES EXPORTED FIELDS**

These are the Kubernetes fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana.

The namespace for Kubernetes-specific metadata. The **kubernetes.pod_name** is the name of the pod.

**kubernetes.labels** Fields

Labels attached to the OpenShift object are **kubernetes.labels**. Each label name is a subfield of labels field. Each label name is de-dotted, meaning dots in the name are replaced with underscores.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kubernetes.pod_id</td>
<td>Kubernetes ID of the pod.</td>
</tr>
<tr>
<td>kubernetes.namespace_name</td>
<td>The name of the namespace in Kubernetes.</td>
</tr>
<tr>
<td>kubernetes.namespace_id</td>
<td>ID of the namespace in Kubernetes.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>kubernetes.host</code></td>
<td>Kubernetes node name.</td>
</tr>
<tr>
<td><code>kubernetes.container_name</code></td>
<td>The name of the container in Kubernetes.</td>
</tr>
<tr>
<td><code>kubernetes.labels.deployment</code></td>
<td>The deployment associated with the Kubernetes object.</td>
</tr>
<tr>
<td><code>kubernetes.labels.deploymentconfig</code></td>
<td>The deploymentconfig associated with the Kubernetes object.</td>
</tr>
<tr>
<td><code>kubernetes.labels.component</code></td>
<td>The component associated with the Kubernetes object.</td>
</tr>
<tr>
<td><code>kubernetes.labels.provider</code></td>
<td>The provider associated with the Kubernetes object.</td>
</tr>
</tbody>
</table>

**kubernetes.annotations Fields**
Annotations associated with the OpenShift object are `kubernetes.annotations` fields.

### 11.4. CONTAINER EXPORTED FIELDS

These are the Docker fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana. Namespace for docker container-specific metadata. The docker.container_id is the Docker container ID.

**pipeline_metadata.collector Fields**
This section contains metadata specific to the collector.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pipeline_metadata.collector.hostname</code></td>
<td>FQDN of the collector. It might be different from the FQDN of the actual emitter of the logs.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.name</code></td>
<td>Name of the collector.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.version</code></td>
<td>Version of the collector.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.ipaddr4</code></td>
<td>IP address v4 of the collector server, can be an array.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.ipaddr6</code></td>
<td>IP address v6 of the collector server, can be an array.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pipeline_metadata.collector.inputname</td>
<td>How the log message was received by the collector whether it was TCP/UDP, or imjournal/imfile.</td>
</tr>
<tr>
<td>pipeline_metadata.collector.received_at</td>
<td>Time when the message was received by the collector.</td>
</tr>
<tr>
<td>pipeline_metadata.collector.original_raw_message</td>
<td>The original non-parsed log message, collected by the collector or as close to the source as possible.</td>
</tr>
</tbody>
</table>

**pipeline_metadata.normalizer Fields**
This section contains metadata specific to the normalizer.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipeline_metadata.normalizer.hostname</td>
<td>FQDN of the normalizer.</td>
</tr>
<tr>
<td>pipeline_metadata.normalizer.name</td>
<td>Name of the normalizer.</td>
</tr>
<tr>
<td>pipeline_metadata.normalizer.version</td>
<td>Version of the normalizer.</td>
</tr>
<tr>
<td>pipeline_metadata.normalizer.ipaddr4</td>
<td>IP address v4 of the normalizer server, can be an array.</td>
</tr>
<tr>
<td>pipeline_metadata.normalizer.ipaddr6</td>
<td>IP address v6 of the normalizer server, can be an array.</td>
</tr>
<tr>
<td>pipeline_metadata.normalizer.inputname</td>
<td>how the log message was received by the normalizer whether it was TCP/UDP.</td>
</tr>
<tr>
<td>pipeline_metadata.normalizer.received_at</td>
<td>Time when the message was received by the normalizer.</td>
</tr>
<tr>
<td>pipeline_metadata.normalizer.original_raw_message</td>
<td>The original non-parsed log message as it is received by the normalizer.</td>
</tr>
<tr>
<td>pipeline_metadata.trace</td>
<td>The field records the trace of the message. Each collector and normalizer appends information about itself and the date and time when the message was processed.</td>
</tr>
</tbody>
</table>

11.5. OVIRT EXPORTED FIELDS
These are the oVirt fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana.

Namespace for oVirt metadata.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovirt.entity</td>
<td>The type of the data source, hosts, VMS, and engine.</td>
</tr>
<tr>
<td>ovirt.host_id</td>
<td>The oVirt host UUID.</td>
</tr>
</tbody>
</table>

**ovirt.engine Fields**

Namespace for oVirt engine related metadata. The FQDN of the oVirt engine is `ovirt.engine.fqdn`.

**11.6. AUSHAPE EXPORTED FIELDS**

These are the Aushape fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana.

Audit events converted with Aushape. For more information, see Aushape.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aushape.serial</td>
<td>Audit event serial number.</td>
</tr>
<tr>
<td>aushape.node</td>
<td>Name of the host where the audit event occurred.</td>
</tr>
<tr>
<td>aushape.error</td>
<td>The error aushape encountered while converting the event.</td>
</tr>
<tr>
<td>aushape.trimmed</td>
<td>An array of JSONPath expressions relative to the event object, specifying objects or arrays with the content removed as the result of event size limiting. An empty string means the event removed the content, and an empty array means the trimming occurred by unspecified objects and arrays.</td>
</tr>
<tr>
<td>aushape.text</td>
<td>An array log record strings representing the original audit event.</td>
</tr>
</tbody>
</table>

**aushape.data Fields**

Parsed audit event data related to Aushape.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aushape.data.avc</td>
<td>type: nested</td>
</tr>
<tr>
<td>aushape.data.execve</td>
<td>type: string</td>
</tr>
<tr>
<td>aushape.data.netfilter_cfg</td>
<td>type: nested</td>
</tr>
</tbody>
</table>
11.7. TLOG EXPORTED FIELDS

These are the Tlog fields exported by the OpenShift Container Platform cluster logging system and available for searching from Elasticsearch and Kibana.

Tlog terminal I/O recording messages. For more information see Tlog.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tlog.ver</code></td>
<td>Message format version number.</td>
</tr>
<tr>
<td><code>tlog.user</code></td>
<td>Recorded user name.</td>
</tr>
<tr>
<td><code>tlog.term</code></td>
<td>Terminal type name.</td>
</tr>
<tr>
<td><code>tlog.session</code></td>
<td>Audit session ID of the recorded session.</td>
</tr>
<tr>
<td><code>tlog.id</code></td>
<td>ID of the message within the session.</td>
</tr>
<tr>
<td><code>tlog.pos</code></td>
<td>Message position in the session, milliseconds.</td>
</tr>
<tr>
<td><code>tlog.timing</code></td>
<td>Distribution of this message’s events in time.</td>
</tr>
<tr>
<td><code>tlog.in_txt</code></td>
<td>Input text with invalid characters scrubbed.</td>
</tr>
<tr>
<td><code>tlog.in_bin</code></td>
<td>Scrubbed invalid input characters as bytes.</td>
</tr>
<tr>
<td><code>tlog.out_txt</code></td>
<td>Output text with invalid characters scrubbed.</td>
</tr>
<tr>
<td><code>tlog.out_bin</code></td>
<td>Scrubbed invalid output characters as bytes.</td>
</tr>
</tbody>
</table>