OpenShift Container Platform 4.6

Logging

Configuring cluster logging in OpenShift Container Platform

Last Updated: 2022-10-24

OpenShift Container Platform 4.6 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

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 JSON OpenShift Container Platform Logging

You can use JSON logging to configure the Log Forwarding API to parse JSON strings into a structured object. You can perform the following tasks:

- Parse JSON logs
- Configure JSON log data for Elasticsearch
- Forward JSON logs to the Elasticsearch log store

For information, see About JSON Logging.

1.1.2. About collecting and storing Kubernetes events

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

For information, see About collecting and storing Kubernetes events .

1.1.3. About updating OpenShift Container Platform Logging

OpenShift Container Platform allows you to update OpenShift Container Platform logging. You must update the following operators while updating OpenShift Container Platform Logging:

- Elasticsearch Operator
- Cluster Logging Operator

For information, see About updating OpenShift Container Platform Logging .

1.1.4. About viewing the cluster dashboard

The OpenShift Container Platform Logging dashboard contains charts that show details about your Elasticsearch instance at the cluster level. These charts help you diagnose and anticipate problems.

For information, see About viewing the cluster dashboard.

1.1.5. About troubleshooting OpenShift Container Platform Logging

You can troubleshoot the logging issues by performing the following tasks:

- Viewing logging status
- Viewing the status of the log store
- Understanding logging alerts
- Collecting logging data for Red Hat Support
- Troubleshooting for critical alerts

1.1.6. About uninstalling OpenShift Container Platform Logging

You can stop log aggregation by deleting the ClusterLogging custom resource (CR). After deleting the CR, there are other cluster logging components that remain, which you can optionally remove.

For information, see About uninstalling OpenShift Container Platform Logging .

1.1.7. About exporting fields

The logging system exports fields. Exported fields are present in the log records and are available for searching from Elasticsearch and Kibana.

For information, see About exporting fields.

1.1.8. 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.

1.1.9. 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.

1.1.10. 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 OpenShift 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.

1.1.11. 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.

1.1.12. 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 .

1.1.13. 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 OpenShift Elasticsearch Operator and Cluster Logging Operator. The OpenShift 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 OpenShift 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 OpenShift Elasticsearch Operator and Cluster Logging Operator.

NOTE

If you do not want to 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. For more information, see Removing unused components if you do not use the default Elasticsearch log store.

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

1. Install the OpenShift Elasticsearch Operator:
   a. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
   b. Choose **OpenShift 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 **4.6** 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 OpenShift Elasticsearch Operator installed by switching to the **Operators → Installed Operators** page.

j. Ensure that **OpenShift 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 **4.6** 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.

   b. On the **Custom Resource Definitions** page, click **ClusterLogging**.

   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:

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.

```
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: 7
        limits:
          memory: "16Gi"
        requests:
          memory: "16Gi"
      proxy: 8
        resources:
          limits:
            memory: 256Mi
          requests:
            memory: 256Mi
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" 9
    kibana:
      replicas: 1
  curation:
    type: "curator"
    curator:
      schedule: "30 3 * * *" 10
  collection:
    logs:
      type: "fluentd" 11
      fluentd: {}
```

1. The name must be **instance**.

2. 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.

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

4. 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.

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

6. 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 Logging uses ephemeral storage.

7. Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the OpenShift 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.

8. Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift 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.

9 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

10 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.

11 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 control plane nodes (also known as the 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. Control plane 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
- fluentd-9z7kk
- fluentd-br7r2
- fluentd-fn2sb
- fluentd-pb2f8
- fluentd-zqgqx
- kibana-7fb4fd4cc9-bvt4p

Additional resources

- Installing Operators from the OperatorHub

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 OpenShift Elasticsearch Operator and Cluster Logging Operator.

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 OpenShift Elasticsearch Operator and Cluster Logging Operator using the CLI:

1. Create a namespace for the OpenShift Elasticsearch Operator.

   a. Create a namespace object YAML file (for example, **eo-namespace.yaml**) for the OpenShift Elasticsearch Operator:

   ```
   apiVersion: v1
   kind: Namespace
   metadata:
     name: openshift-operators-redhat (1)
     annotations:
       openshift.io/node-selector: ""
     labels:
       openshift.io/cluster-monitoring: "true" (2)
   ```

   (1) 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.

   (2) String. You must specify this label as shown to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.

   b. Create the namespace:

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

   For example:

   ```
   $ 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:

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

   b. Create the namespace:

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

   For example:

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

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

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

   ```
   apiVersion: operators.coreos.com/v1
   kind: OperatorGroup
   metadata:
     name: openshift-operators-redhat
     namespace: openshift-operators-redhat (1)
   spec: {}
   ```

   (1) You must specify the **openshift-operators-redhat** namespace.

   b. Create an Operator Group object:

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

   For example:

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

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

   Example Subscription

   ```
   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   metadata:
     name: "elasticsearch-operator"
     namespace: "openshift-operators-redhat" (1)
   spec:
     channel: "4.6" (2)
     installPlanApproval: "Automatic"
     source: "redhat-operators" (3)
     sourceNamespace: "openshift-marketplace"
     name: "elasticsearch-operator"
   ```

   (1) You must specify the **openshift-operators-redhat** namespace.

   (2) Specify **4.6** 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:

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

For example:

```
$ oc create -f eo-sub.yaml
```

The OpenShift Elasticsearch Operator is installed to the **openshift-operators-redhat** namespace and copied to each project in the cluster.

e. Verify the Operator installation:

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

Example output

```
NAMESPACE                                          NAME                                    DISPLAY
VERSION              REPLACES   PHASE
default                                        elasticsearch-operator.4.6.0-202007012112.p0
Elasticsearch Operator   4.6.0-202007012112.p0              Succeeded
kube-node-lease                                    elasticsearch-operator.4.6.0-202007012112.p0
Elasticsearch Operator   4.6.0-202007012112.p0              Succeeded
kube-public                                        elasticsearch-operator.4.6.0-202007012112.p0
Elasticsearch Operator   4.6.0-202007012112.p0              Succeeded
kube-system                                        elasticsearch-operator.4.6.0-202007012112.p0
Elasticsearch Operator   4.6.0-202007012112.p0              Succeeded
openshift-apiserver-operator                           elasticsearch-operator.4.6.0-
202007012112.p0    Elasticsearch Operator   4.6.0-202007012112.p0
Succeeded
openshift-apiserver                                elasticsearch-operator.4.6.0-202007012112.p0
Elasticsearch Operator   4.6.0-202007012112.p0              Succeeded
openshift-authentication-operator                      elasticsearch-operator.4.6.0-
202007012112.p0    Elasticsearch Operator   4.6.0-202007012112.p0
Succeeded
openshift-authentication                               elasticsearch-operator.4.6.0-
202007012112.p0    Elasticsearch Operator   4.6.0-202007012112.p0
Succeeded
...
```

There should be an OpenShift 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:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: cluster-logging
  namespace: openshift-logging (1)
spec:
  targetNamespaces:
  - openshift-logging (2)
```

(1) (2) You must specify the **openshift-logging** namespace.

b. Create the OperatorGroup object:

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

For example:

```
$ 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.

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: cluster-logging
  namespace: openshift-logging (1)
spec:
  channel: "4.6" (2)
  name: cluster-logging
  source: redhat-operators (3)
  sourceNamespace: openshift-marketplace
```

(1) You must specify the **openshift-logging** namespace.

(2) Specify **4.6** 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:

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

For example:

```
$ 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.

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

Example output

```
NAMESPACE                              NAME                             DISPLAY
VERSION           REPLACES   PHASE
...
```

```
openshift-logging                          clusterlogging.4.6.0-202007012112.p0
Cluster Logging        4.6.0-202007012112.p0          Succeeded
...
```

5. Create a Cluster Logging instance:

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

      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.

```
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: 7
        limits:
          memory: "16Gi"
        requests:
          memory: "16Gi"
      proxy: 8
        resources:
          limits:
            memory: 256Mi
          requests:
             memory: 256Mi
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" 9
    kibana:
      replicas: 1
  curation:
    type: "curator"
    curator:
      schedule: "30 3 * * *" (10)
  collection:
    logs:
      type: "fluentd" (11)
      fluentd: {}
```

1. The name must be **instance**.

2. 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.

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

4. 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.

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

6. 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 OpenShift Logging with ephemeral storage only.

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

8. Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift 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.

9. 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**.

10. 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.

11. 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 control plane 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. Control plane 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    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:

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

For example:

```
$ 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:

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

Example output

```
NAME                                            READY   STATUS    RESTARTS   AGE
cluster-logging-operator-66f77ffccb-ppzbg       1/1     Running   0          7m
elasticsearch-cdm-ftuhduuw-1-ffc4b9566-q6bhp    2/2     Running   0          2m40s
elasticsearch-cdm-ftuhduuw-2-7b4994dbfc-rd2gc   2/2     Running   0          2m36s
elasticsearch-cdm-ftuhduuw-3-84b5ff7ff8-gqnm2   2/2     Running   0          2m4s
fluentd-587vb                                   1/1     Running   0          2m26s
```

```
fluentd-7mpb9                        1/1     Running   0          2m30s
fluentd-flm6j                        1/1     Running   0          2m33s
fluentd-gn4rn                        1/1     Running   0          2m26s
fluentd-nlgb6                        1/1     Running   0          2m30s
fluentd-snpkt                        1/1     Running   0          2m28s
kibana-d6d5668c5-rppqm                         2/2     Running   0          2m39s
```

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 *Cluster 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:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-openshift-operators-redhat
  namespace: openshift-logging
spec:
  ingress:
    - from:
      - podSelector: {}
    - 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

CHAPTER 3. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT

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)

```
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: 5
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 736Mi
        requests:
          cpu: 100m
          memory: 736Mi
        replicas: 1
    curation: 6
      type: "curator"
      curator:
        resources:
          limits:
            memory: 256Mi
          requests:
            cpu: 100m
            memory: 256Mi
        schedule: "30 3 * * *"
    collection: 7
      logs:
        type: "fluentd"
        fluentd:
          resources:
            limits:
              memory: 736Mi
            requests:
              cpu: 100m
              memory: 736Mi
```

1. The CR name must be **instance**.
2. The CR must be installed to the **openshift-logging** namespace.
3. The Cluster Logging Operator management state. When set to **unmanaged** the operator is in an unsupported state and will not get updates.
4. Settings for the log store, including retention policy, the number of nodes, the resource requests and limits, and the storage class.
5. Settings for the visualizer, including the resource requests and limits, and the number of pod replicas.
6. Settings for curation, including the resource requests and limits, and curation schedule.
7. 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 OpenShift 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 OpenShift 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

```
NAME           READY  STATUS    RESTARTS   AGE     IP            NODE                  NOMINATED
NODE   READINESS GATES
fluentd-8d69v  1/1    Running   0          134m    10.130.2.30   master1.example.com   <none>
<none>
fluentd-bd225  1/1    Running   0          134m    10.131.1.11   master2.example.com   <none>
<none>
fluentd-cvrzs  1/1    Running   0          134m    10.130.0.21   master3.example.com   <none>
<none>
fluentd-gpqg2  1/1    Running   0          134m    10.128.2.27   worker1.example.com   <none>
<none>
fluentd-l9j7j  1/1    Running   0          134m    10.129.2.31   worker2.example.com   <none>
<none>
```

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"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   ```

```
....

spec:
  collection:
    logs:
      fluentd:
        resources:
          limits: (1)
            memory: 736Mi
          requests:
            cpu: 100m
            memory: 736Mi
```

(1) Specify the CPU and memory limits and requests as needed. The values shown are the default values.

3.2.4. Advanced configuration for the log forwarder

Cluster logging includes multiple Fluentd parameters that you can use for tuning the performance of the Fluentd log forwarder. With these parameters, you can change the following Fluentd behaviors:

- the size of Fluentd chunks and chunk buffer
- the Fluentd chunk flushing behavior
- the Fluentd chunk forwarding retry behavior

Fluentd collects log data in a single blob called a *chunk*. When Fluentd creates a chunk, the chunk is considered to be in the *stage*, where the chunk gets filled with data. When the chunk is full, Fluentd moves the chunk to the *queue*, where chunks are held before being flushed, or written out to their destination. Fluentd can fail to flush a chunk for a number of reasons, such as network issues or capacity issues at the destination. If a chunk cannot be flushed, Fluentd retries flushing as configured.

By default in OpenShift Container Platform, Fluentd uses the *exponential backoff* method to retry flushing, where Fluentd doubles the time it waits between attempts to retry flushing again, which helps reduce connection requests to the destination. You can disable exponential backoff and use the *periodic* retry method instead, which retries flushing the chunks at a specified interval. By default, Fluentd retries chunk flushing indefinitely. In OpenShift Container Platform, you cannot change the indefinite retry behavior.

These parameters can help you determine the trade-offs between latency and throughput.

- To optimize Fluentd for throughput, you could use these parameters to reduce network packet count by configuring larger buffers and queues, delaying flushes, and setting longer times between retries. Be aware that larger buffers require more space on the node file system.
- To optimize for low latency, you could use the parameters to send data as soon as possible, avoid the build-up of batches, have shorter queues and buffers, and use more frequent flush and retries.

You can configure the chunking and flushing behavior using the following parameters in the **ClusterLogging** custom resource (CR). The parameters are then automatically added to the Fluentd config map for use by Fluentd.

NOTE

These parameters are:

- Not relevant to most users. The default settings should give good general performance.
- Only for advanced users with detailed knowledge of Fluentd configuration and performance.
- Only for performance tuning. They have no effect on functional aspects of logging.

Table 3.1. Advanced Fluentd Configuration Parameters

Parmeter	Description	Default
chunkLimitSize	The maximum size of each chunk. Fluentd stops writing data to a chunk when it reaches this size. Then, Fluentd sends the chunk to the queue and opens a new chunk.	**8m**
totalLimitSize	The maximum size of the buffer, which is the total size of the stage and the queue. If the buffer size exceeds this value, Fluentd stops adding data to chunks and fails with an error. All data not in chunks is lost.	**8G**
flushInterval	The interval between chunk flushes. You can use **s** (seconds), **m** (minutes), **h** (hours), or **d** (days).	**1s**
flushMode	The method to perform flushes: • **lazy**: Flush chunks based on the **timekey** parameter. You cannot modify the **timekey** parameter. • **interval**: Flush chunks based on the **flushInterval** parameter. • **immediate**: Flush chunks immediately after data is added to a chunk.	**interval**

Parmeter	Description	Default
flushThreadCount	The number of threads that perform chunk flushing. Increasing the number of threads improves the flush throughput, which hides network latency.	**2**
overflowAction	The chunking behavior when the queue is full: • **throw_exception**: Raise an exception to show in the log. • **block**: Stop data chunking until the full buffer issue is resolved. • **drop_oldest_chunk**: Drop the oldest chunk to accept new incoming chunks. Older chunks have less value than newer chunks.	**block**
retryMaxInterval	The maximum time in seconds for the **exponential_backoff** retry method.	**300s**
retryType	The retry method when flushing fails: • **exponential_backoff**: Increase the time between flush retries. Fluentd doubles the time it waits until the next retry until the **retry_max_interval** parameter is reached. • **periodic**: Retries flushes periodically, based on the **retryWait** parameter.	**exponential_backoff**
retryWait	The time in seconds before the next chunk flush.	**1s**

For more information on the Fluentd chunk lifecycle, see Buffer Plugins in the Fluentd documentation.

Procedure

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

```
$ oc edit ClusterLogging instance
```

2. Add or modify any of the following parameters:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  forwarder:
    fluentd:
      buffer:
        chunkLimitSize: 8m 1
        flushInterval: 5s 2
        flushMode: interval 3
        flushThreadCount: 3 4
        overflowAction: throw_exception 5
        retryMaxInterval: "300s" 6
        retryType: periodic 7
        retryWait: 1s 8
        totalLimitSize: 32m 9
...
```

1 Specify the maximum size of each chunk before it is queued for flushing.

2 Specify the interval between chunk flushes.

3 Specify the method to perform chunk flushes: **lazy**, **interval**, or **immediate**.

4 Specify the number of threads to use for chunk flushes.

5 Specify the chunking behavior when the queue is full: **throw_exception**, **block**, or **drop_oldest_chunk**.

6 Specify the maximum interval in seconds for the **exponential_backoff** chunk flushing method.

7 Specify the retry type when chunk flushing fails: **exponential_backoff** or **periodic**.

8 Specify the time in seconds before the next chunk flush.

9 Specify the maximum size of the chunk buffer.

3. Verify that the Fluentd pods are redeployed:

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

4. Check that the new values are in the **fluentd** config map:

```
$ oc extract configmap/fluentd --confirm
```

Example fluentd.conf

```
<buffer>
 @type file
 path '/var/lib/fluentd/default'
 flush_mode interval
 flush_interval 5s
 flush_thread_count 3
 retry_type periodic
 retry_wait 1s
 retry_max_interval 300s
 retry_timeout 60m
 queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '32'}"
 total_limit_size 32m
 chunk_limit_size 8m
 overflow_action throw_exception
</buffer>
```

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:

  ```
  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:

```
$ 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:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  collection:
    logs:
      type: "fluentd"
      fluentd: {}
```

4. 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 Forward API.

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 Forward API to forward audit logs to the internal Elasticsearch instance:

1. Create a **ClusterLogForwarder** 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:

     ```
     apiVersion: logging.openshift.io/v1
     kind: ClusterLogForwarder
     metadata:
       name: instance
       namespace: openshift-logging
     spec:
       pipelines: 1
       - name: all-to-default
         inputRefs:
         - infrastructure
         - application
         - audit
         outputRefs:
         - default
     ```

     1 A pipeline defines the type of logs to forward using the specified output. The default output forwards logs to the internal Elasticsearch instance.

     NOTE

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

   - If you have an existing **ClusterLogForwarder** CR, add a pipeline to the default output for the audit logs. You do not need to define the default output. For example:

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

```
 - name: elasticsearch-insecure
   type: "elasticsearch"
   url: http://elasticsearch-insecure.messaging.svc.cluster.local
   insecure: true
 - name: elasticsearch-secure
   type: "elasticsearch"
   url: https://elasticsearch-secure.messaging.svc.cluster.local
   secret:
     name: es-audit
 - name: secureforward-offcluster
   type: "fluentdForward"
   url: https://secureforward.offcluster.com:24224
   secret:
     name: secureforward
pipelines:
 - name: container-logs
   inputRefs:
   - application
   outputRefs:
   - secureforward-offcluster
 - name: infra-logs
   inputRefs:
   - infrastructure
   outputRefs:
   - elasticsearch-insecure
 - name: audit-logs
   inputRefs:
   - audit
   outputRefs:
   - elasticsearch-secure
   - default 1
```

1 This pipeline sends the audit logs to the internal Elasticsearch instance in addition to an external 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 configure a *retention policy* that specifies how long the default Elasticsearch log store keeps indices for each of the three log sources: infrastructure logs, application logs, and audit logs.

To configure the retention policy, you set a **maxAge** parameter for each log source in the **ClusterLogging** custom resource (CR). The CR applies these values to the Elasticsearch rollover schedule, which 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 based on the retention policy you configure. If you do not create a retention policy for any log sources, logs are deleted after seven days by default.

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:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
...
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    retentionPolicy: (1)
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3
...
```

(1) 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-over indices are deleted seven days after rollover. OpenShift Container Platform checks every 15 minutes to determine if the indices need to be rolled over.

```
apiVersion: "logging.openshift.io/v1"
kind: "Elasticsearch"
metadata:
  name: "elasticsearch"
spec:
...
  indexManagement:
    policies: (1)
      - name: infra-policy
        phases:
```

```
        delete:
          minAge: 7d (2)
        hot:
          actions:
            rollover:
              maxAge: 8h (3)
      pollInterval: 15m (4)
...
```

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

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

3. 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.

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

```
$ oc get cronjob
```

Example output

```
NAME                     SCHEDULE       SUSPEND   ACTIVE   LAST SCHEDULE   AGE
curator                 */10 * * * *   False     0        <none>          5s
elasticsearch-im-app     */15 * * * *   False     0        <none>          4s
elasticsearch-im-audit   */15 * * * *   False     0        <none>          4s
elasticsearch-im-infra   */15 * * * *   False     0        <none>          4s
```

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
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
....
spec:
    logStore:
      type: "elasticsearch"
      elasticsearch:
        resources: 1
          limits:
            memory: "16Gi"
          requests:
            cpu: "1"
            memory: "16Gi"
        proxy: 2
          resources:
            limits:
              memory: 100Mi
            requests:
              memory: 100Mi
```

1 Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the OpenShift 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.

2 Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift 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:

```
      resources:
        limits:
          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 memory you want, assuming the node has the 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:

```
$ oc edit clusterlogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"

....

spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      redundancyPolicy: "SingleRedundancy" (1)
```

(1) 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.

```
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** config map 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.

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. Scale down the Fluentd pods so they stop sending new logs to Elasticsearch:

```
$ oc -n openshift-logging patch daemonset/logging-fluentd -p '{"spec":{"template":{"spec":
{"nodeSelector":{"logging-infra-fluentd": "false"}}}}}'
```

4. Perform a shard synced flush using the OpenShift Container Platform es_util tool to ensure there are no pending 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:

```
$ 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}}
```

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

```
$ 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:

```
$ 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":
```

6. 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:

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

–

For example:

```
$ 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.

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

Example output

```
NAME                                        READY   STATUS    RESTARTS   AGE
elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6k    2/2     Running   0          22h
elasticsearch-cdm-5ceex6ts-2-f799564cb-l9mj7    2/2     Running   0          22h
elasticsearch-cdm-5ceex6ts-3-585968dc68-k7kjr   2/2     Running   0          22h
```

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", (1)
  "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.

7. If you changed the Elasticsearch configuration map, repeat these steps for each Elasticsearch pod.

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

```
$ 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:

```
$ 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

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

9. Scale up the Fluentd pods so they send new logs to Elasticsearch.

```
$ oc -n openshift-logging patch daemonset/logging-fluentd -p '{"spec":{"template":{"spec":
{"nodeSelector":{"logging-infra-fluentd": "true"}}}}}'
```

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:

```
$ 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

```
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
elasticsearch   ClusterIP   172.30.183.229   <none>        9200/TCP   22h
```

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

```
$ oc exec elasticsearch-cdm-oplnhinv-1-5746475887-fj2f8 -n openshift-logging -- curl -tlsv1.2 --
insecure -H "Authorization: Bearer ${token}" "https://172.30.183.229:9200/_cat/health"
```

Example output

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    29  100    29    0     0    108      0 --:--:-- --:--:-- --:--:--   108
```

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
       name: elasticsearch
     tls:
       termination: reencrypt
       destinationCACertificate: | (1)
   ```

   (1) Add the log store CA certifcate or use the command in the next step. You do not have to set the **spec.tls.key**, **spec.tls.certificate**, and **spec.tls.caCertificate** parameters required by some reencrypt routes.

   b. Run the following command to add the log store CA certificate to the route YAML you created in the previous step:

   ```
   $ cat ./admin-ca | sed -e "s/^/      /" >> <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:

```
curl -tlsv1.2 --insecure -H "Authorization: Bearer ${token}" "https://${routeES}"
```

The response appears similar to the following:

Example output

```
{
  "name" : "elasticsearch-cdm-i40ktba0-1",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "0eY-tJzcR3KOdpgeMJo-MQ",
  "version" : {
  "number" : "6.8.1",
  "build_flavor" : "oss",
  "build_type" : "zip",
  "build_hash" : "Unknown",
  "build_date" : "Unknown",
  "build_snapshot" : true,
  "lucene_version" : "7.7.0",
  "minimum_wire_compatibility_version" : "5.6.0",
  "minimum_index_compatibility_version" : "5.0.0"
},
  "<tagline>" : "<for search>"
}
```

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:

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

```
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
        requests:
          cpu: 100m
          memory: 100Mi
    replicas: 2
curation:
  type: "curator"
  curator:
    resources: (4)
      limits:
        memory: 200Mi
      requests:
        cpu: 200m
        memory: 200Mi
    schedule: "*/10 * * * *"
collection:
  logs:
    type: "fluentd"
    fluentd:
      resources: (5)
        limits:
          memory: 736Mi
        requests:
          cpu: 200m
          memory: 736Mi
```

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

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

5 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

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"

....

spec:
    visualization:
      type: "kibana"
      kibana:
        replicas: 1 1
```

1 Specify the number of Kibana nodes.

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 configuration. On OpenShift Container Platform this is achieved using persistent volume claims.

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.

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

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

Elasticsearch requires sufficient memory to perform large merge operations. If it does not have enough memory, it becomes unresponsive. To avoid this problem, evaluate how much application log data you need, and allocate approximately double that amount of free storage capacity.

By default, when storage capacity is 85% full, 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 a 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 default values. Although the alerts use the same default values, you cannot change these values in the alerts.

3.5.2. Additional resources

- Persistent Elasticsearch Storage

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:

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

```
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
      requests:
       cpu: 100m
       memory: 100Mi
    replicas: 2
  curation:
   type: "curator"
   curator:
    resources: 4
     limits:
      memory: 200Mi
     requests:
      cpu: 200m
      memory: 200Mi
    schedule: "*/10 * * * *"
  collection:
   logs:
    type: "fluentd"
    fluentd:
     resources: 5
      limits:
       memory: 736Mi
      requests:
       cpu: 200m
       memory: 736Mi
```

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

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

5 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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 1
      tolerations: 1
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
      resources:
        limits:
          memory: 8Gi
        requests:
          cpu: 100m
          memory: 1Gi
      storage: {}
      redundancyPolicy: "ZeroRedundancy"
  visualization:
    type: "kibana"
    kibana:
      tolerations: 2
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
      resources:
        limits:
          memory: 2Gi
        requests:
          cpu: 100m
          memory: 1Gi
      replicas: 1
  collection:
    logs:
      type: "fluentd"
      fluentd:
        tolerations: (3)
        - key: "logging"
          operator: "Exists"
          effect: "NoExecute"
          tolerationSeconds: 6000
        resources:
          limits:
            memory: 2Gi
          requests:
            cpu: 100m
            memory: 1Gi
```

1. This toleration is added to the Elasticsearch pods.
2. This toleration is added to the Kibana pod.
3. 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:

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

   For example:

```
$ 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:

```
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 1
      tolerations:
      - key: "elasticsearch"  1
        operator: "Exists"  2
        effect: "NoExecute"  3
        tolerationSeconds: 6000  4
```

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.

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 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:

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

For example:

```
$ 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:

```
visualization:
  type: "kibana"
  kibana:
    tolerations:
    - key: "kibana"  1
      operator: "Exists"  2
      effect: "NoExecute"  3
      tolerationSeconds: 6000 4
```

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**.

3.7.3. 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:

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

   For example:

   ```
   $ 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:

   ```
   collection:
     logs:
       type: "fluentd"
       fluentd:
         tolerations:
         - key: "collector" 1
           operator: "Exists" 2
           effect: "NoExecute" 3
           tolerationSeconds: 6000 4
   ```

   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**.

3.7.4. 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:

```
$ oc edit ClusterLogging instance
```

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

...

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  curation:
    curator:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      resources: null
      schedule: 30 3 * * *
    type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 2
        node-role.kubernetes.io/infra: ''
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: 500m
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage: {}
    type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      nodeSelector: 3
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana

...
```

1 2 3 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:

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

  #### Example output

  ```
  NAME                      READY   STATUS    RESTARTS   AGE   IP            NODE
  NOMINATED NODE   READINESS GATES
  kibana-5b8bdf44f9-ccpq9   2/2     Running   0          27s   10.129.2.18   ip-10-0-147-79.us-
  east-2.compute.internal   <none>           <none>
  ```

- 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

  ```
  NAME                                         STATUS   ROLES    AGE   VERSION
  ip-10-0-133-216.us-east-2.compute.internal   Ready    master   60m   v1.19.0
  ip-10-0-139-146.us-east-2.compute.internal   Ready    master   60m   v1.19.0
  ip-10-0-139-192.us-east-2.compute.internal   Ready    worker   51m   v1.19.0
  ip-10-0-139-241.us-east-2.compute.internal   Ready    worker   51m   v1.19.0
  ip-10-0-147-79.us-east-2.compute.internal    Ready    worker   51m   v1.19.0
  ip-10-0-152-241.us-east-2.compute.internal   Ready    master   60m   v1.19.0
  ip-10-0-139-48.us-east-2.compute.internal    Ready    infra    51m   v1.19.0
  ```

  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

  ```
  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:

```
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
```

(1) 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

```
NAME                                            READY   STATUS        RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7       1/1     Running       0          29m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running       0          28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running       0          28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running       0          28m
fluentd-42dzz                                   1/1     Running       0          28m
fluentd-d74rq                                   1/1     Running       0          28m
fluentd-m5vr9                                   1/1     Running       0          28m
fluentd-nkxl7                                   1/1     Running       0          28m
fluentd-pdvqb                                   1/1     Running       0          28m
fluentd-tflh6                                   1/1     Running       0          28m
kibana-5b8bdf44f9-ccpq9                         2/2     Terminating   0          4m11s
kibana-7d85dcffc8-bfpfp                         2/2     Running       0          33s
```

- The new pod is on the **ip-10-0-139-48.us-east-2.compute.internal** node:

  ```
  $ oc get pod kibana-7d85dcffc8-bfpfp -o wide
  ```

  Example output

  ```
  NAME                    READY   STATUS      RESTARTS   AGE   IP          NODE
  NOMINATED NODE   READINESS GATES
  kibana-7d85dcffc8-bfpfp   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

  ```
  NAME                                            READY   STATUS    RESTARTS   AGE
  cluster-logging-operator-84d98649c4-zb9g7       1/1     Running   0          30m
  elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running   0          29m
  elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running   0          29m
  elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running   0          29m
  fluentd-42dzz                                   1/1     Running   0          29m
  fluentd-d74rq                                   1/1     Running   0          29m
  fluentd-m5vr9                                   1/1     Running   0          29m
  fluentd-nkxl7                                   1/1     Running   0          29m
  fluentd-pdvqb                                   1/1     Running   0          29m
  fluentd-tflh6                                   1/1     Running   0          29m
  kibana-7d85dcffc8-bfpfp                         2/2     Running   0          62s
  ```

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 **RateLimitIntervalSec=30s** 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 (1)
ForwardToConsole=no (2)
ForwardToSyslog=no
MaxRetentionSec=1month (3)
RateLimitBurst=10000 (4)
RateLimitIntervalSec=30s
Storage=persistent (5)
SyncIntervalSec=1s (6)
SystemMaxUse=8g (7)
SystemKeepFree=20% (8)
SystemMaxFileSize=10M (9)
```

1. 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**.

2. 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.

3. Specify 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 **1month**.

4. 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 **RateLimitIntervalSec=30s** and **RateLimitBurst=10000**, which are the defaults.

5. Specify 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 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.

6. 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**.

7. Specify the maximum size the journal can use. The default is **8g**.

8. Specify how much disk space systemd must leave free. The default is **20%**.

9. 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 and store it in a variable that is named **jrnl_cnf** by running the following command:

```
$ export jrnl_cnf=$( cat journald.conf | base64 -w0 )
```

3. Create a **MachineConfig** object that includes the **jrnl_cnf** variable, which was created in the previous step. The following sample command creates a **MachineConfig** object for the worker:

```
$ cat << EOF > ./40-worker-custom-journald.yaml 1
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker 2
  name: 40-worker-custom-journald 3
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 3.1.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,${jrnl_cnf} 4
          verification: {}
        filesystem: root
        mode: 0644 5
        path: /etc/systemd/journald.conf.d/custom.conf
  osImageURL: ""
EOF
```

1 Optional: For control plane (also known as master) node, you can provide the file name as **40-master-custom-journald.yaml**.

2 Optional: For control plane (also known as master) node, provide the role as **master**.

3 Optional: For control plane (also known as master) node, you can provide the name as **40-master-custom-journald**.

4. Optional: To include a static copy of the parameters in the **journald.conf** file, replace **${jrnl_cnf}** with the output of the **echo $jrnl_cnf** command.

5. Set the permissions for the **journal.conf** file. It is recommended to set **0644** permissions.

4. Create the machine config:

```
$ oc apply -f <file_name>.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:

```
$ oc describe machineconfigpool/<node> 1
```

1. Specify the node as **master** or **worker**.

Example output for worker

```
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 * * * (1)
    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:

```
config.yaml: |
  project_name:
    action
      unit:value
```

The available parameters are:

Table 3.2. Project options

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

Table 3.3. Filter options

Variable Name	Description
.defaults	Use **.defaults** as the **project_name** to set the defaults for projects that are not specified.
.regex	The list of regular expressions that match project names.
pattern	The valid and properly escaped regular expression pattern enclosed by single quotation marks.

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
```

IMPORTANT

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 OpenShift 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 OpenShift 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 OpenShift 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:

  ```
  $ 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

```
{
  "_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/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",
        "received_at": "2020-09-23T20:47:15.007583+00:00",
        "version": "1.7.4 1.6.0"
      }
    },
    "@timestamp": "2020-09-23T20:47:03.422465+00:00",
    "viaq_msg_id": "YmJmYTBlNDktMDMGQtMjE3NmFiOGUyOWM3",
    "openshift": {
      "labels": {
        "logging": "infra"
      }
    }
  },
  "fields": {
    "@timestamp": [
      "2020-09-23T20:47:03.422Z"
    ],
    "pipeline_metadata.collector.received_at": [
      "2020-09-23T20:47:15.007Z"
    ]
  },
  "sort": [
    1600894023422
  ]
}
```

CHAPTER 6. FORWARDING LOGS TO THIRD PARTY SYSTEMS

By default, cluster logging sends container and infrastructure logs to the default internal Elasticsearch log store defined in the **ClusterLogging** custom resource. However, it does not send audit logs to the internal store because it does not provide secure storage. If this default configuration meets your needs, you do not need to configure the Log Forwarding API.

To send logs to other log aggregators, you use the OpenShift Container Platform Log Forwarding API. This API enables you to send container, infrastructure, and audit logs to specific endpoints within or outside your cluster. You can send different types of logs to various systems, so different individuals can access each type. You can also enable TLS support to send logs securely, as required by your organization.

NOTE

To send audit logs to the internal log store, use the Log Forwarding API as described in Forward audit logs to the log store .

When you forward logs externally, the Cluster Logging Operator creates or modifies a Fluentd config map to send logs using your desired protocols. You are responsible for configuring the protocol on the external log aggregator.

Alternatively, you can create a config map to use the Fluentd **forward** protocol or the syslog protocol to send logs to external systems. However, these methods for forwarding logs are deprecated in OpenShift Container Platform and will be removed in a future release.

IMPORTANT

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

6.1. ABOUT FORWARDING LOGS TO THIRD-PARTY SYSTEMS

Forwarding cluster logs to external third-party systems requires a combination of *outputs* and *pipelines* specified in a **ClusterLogForwarder** custom resource (CR) to send logs to specific endpoints inside and outside of your OpenShift Container Platform cluster. You can also use *inputs* to forward the application logs associated with a specific project to an endpoint.

- An *output* is the destination for log data that you define, or where you want the logs sent. An output can be one of the following types:
  - **elasticsearch**. An external Elasticsearch 6 (all releases) instance. The **elasticsearch** output can use a TLS connection.
  - **fluentdForward**. An external log aggregation solution that supports Fluentd. This option uses the Fluentd **forward** protocols. The **fluentForward** output can use a TCP or TLS connection and supports shared-key authentication by providing a **shared_key** field in a secret. Shared-key authentication can be used with or without TLS.
  - **syslog**. An external log aggregation solution that supports the syslog RFC3164 or RFC5424 protocols. The **syslog** output can use a UDP, TCP, or TLS connection.
  - **kafka**. A Kafka broker. The **kafka** output can use a TCP or TLS connection.

  - **default**. The internal OpenShift Container Platform Elasticsearch instance. You are not required to configure the default output. If you do configure a **default** output, you receive an error message because the **default** output is reserved for the Cluster Logging Operator.

  If the output URL scheme requires TLS (HTTPS, TLS, or UDPS), then TLS server-side authentication is enabled. To also enable client authentication, the output must name a secret in the **openshift-logging** project. The secret must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent.

- A *pipeline* defines simple routing from one log type to one or more outputs, or which logs you want to send. The log types are one of the following:
  - **application**. Container logs generated by user applications running in the cluster, except infrastructure container applications.
  - **infrastructure**. Container logs from pods that run in the **openshift***, **kube***, or **default** projects and journal logs sourced from node file system.
  - **audit**. Logs generated by auditd, the node audit system, and the audit logs from the Kubernetes API server and the OpenShift API server.

  You can add labels to outbound log messages by using **key:value** pairs in the pipeline. For example, you might add a label to messages that are forwarded to others data centers or label the logs by type. Labels that are added to objects are also forwarded with the log message.

- An *input* forwards the application logs associated with a specific project to a pipeline.

In the pipeline, you define which log types to forward using an **inputRef** parameter and where to forward the logs to using an **outputRef** parameter.

Note the following:

- If a **ClusterLogForwarder** object exists, logs are not forwarded to the default Elasticsearch instance, unless there is a pipeline with the **default** output.
- By default, cluster logging sends container and infrastructure logs to the default internal Elasticsearch log store defined in the **ClusterLogging** custom resource. However, it does not send audit logs to the internal store because it does not provide secure storage. If this default configuration meets your needs, do not configure the Log Forwarding API.
- If you do not define a pipeline for a log type, the logs of the undefined types are dropped. For example, if you specify a pipeline for the **application** and **audit** types, but do not specify a pipeline for the **infrastructure** type, **infrastructure** logs are dropped.
- You can use multiple types of outputs in the **ClusterLogForwarder** custom resource (CR) to send logs to servers that support different protocols.
- The internal OpenShift Container Platform Elasticsearch instance 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.
- You are responsible for creating and maintaining any additional configurations that external destinations might require, such as keys and secrets, service accounts, port openings, or global proxy configuration.

The following example forwards the audit logs to a secure external Elasticsearch instance, the infrastructure logs to an insecure external Elasticsearch instance, the application logs to a Kafka broker, and the application logs from the **my-apps-logs** project to the internal Elasticsearch instance.

Sample log forwarding outputs and pipelines

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: elasticsearch-secure (3)
     type: "elasticsearch"
     url: https://elasticsearch.secure.com:9200
     secret:
        name: elasticsearch
   - name: elasticsearch-insecure (4)
     type: "elasticsearch"
     url: http://elasticsearch.insecure.com:9200
   - name: kafka-app (5)
     type: "kafka"
     url: tls://kafka.secure.com:9093/app-topic
  inputs: (6)
   - name: my-app-logs
     application:
        namespaces:
        - my-project
  pipelines:
   - name: audit-logs (7)
     inputRefs:
      - audit
     outputRefs:
      - elasticsearch-secure
      - default
     labels:
       secure: "true" (8)
       datacenter: "east"
   - name: infrastructure-logs (9)
     inputRefs:
      - infrastructure
     outputRefs:
      - elasticsearch-insecure
     labels:
       datacenter: "west"
   - name: my-app (10)
     inputRefs:
      - my-app-logs
     outputRefs:
      - default
   - inputRefs: (11)
      - application
     outputRefs:
```

```
    - kafka-app
  labels:
    datacenter: "south"
```

1. The name of the **ClusterLogForwarder** CR must be **instance**.
2. The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
3. Configuration for an secure Elasticsearch output using a secret with a secure URL.
   - A name to describe the output.
   - The type of output: **elasticsearch**.
   - The secure URL and port of the Elasticsearch instance as a valid absolute URL, including the prefix.
   - The secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project.
4. Configuration for an insecure Elasticsearch output:
   - A name to describe the output.
   - The type of output: **elasticsearch**.
   - The insecure URL and port of the Elasticsearch instance as a valid absolute URL, including the prefix.
5. Configuration for a Kafka output using a client-authenticated TLS communication over a secure URL
   - A name to describe the output.
   - The type of output: **kafka**.
   - Specify the URL and port of the Kafka broker as a valid absolute URL, including the prefix.
6. Configuration for an input to filter application logs from the **my-project** namespace.
7. Configuration for a pipeline to send audit logs to the secure external Elasticsearch instance:
   - Optional. A name to describe the pipeline.
   - The **inputRefs** is the log type, in this example **audit**.
   - The **outputRefs** is the name of the output to use, in this example **elasticsearch-secure** to forward to the secure Elasticsearch instance and **default** to forward to the internal Elasticsearch instance.
   - Optional: Labels to add to the logs.
8. Optional: String. One or more labels to add to the logs. Quote values like "true" so they are recognized as string values, not as a boolean.
9. Configuration for a pipeline to send infrastructure logs to the insecure external Elasticsearch instance.

10 Configuration for a pipeline to send logs from the **my-project** project to the internal Elasticsearch instance.

- Optional. A name to describe the pipeline.
- The **inputRefs** is a specific input: **my-app-logs**.
- The **outputRefs** is **default**.
- Optional: String. One or more labels to add to the logs.

11 Configuration for a pipeline to send logs to the Kafka broker, with no pipeline name:

- The **inputRefs** is the log type, in this example **application**.
- The **outputRefs** is the name of the output to use.
- Optional: String. One or more labels to add to the logs.

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.

6.1.1. Forwarding logs to an external Elasticsearch instance

You can optionally forward logs to an external Elasticsearch instance in addition to, or instead of, the internal OpenShift Container Platform Elasticsearch instance. You are responsible for configuring the external log aggregator to receive log data from OpenShift Container Platform.

To configure log forwarding to an external Elasticsearch instance, create a **ClusterLogForwarder** custom resource (CR) with an output to that instance and a pipeline that uses the output. The external Elasticsearch output can use the HTTP (insecure) or HTTPS (secure HTTP) connection.

To forward logs to both an external and the internal Elasticsearch instance, create outputs and pipelines to the external instance and a pipeline that uses the **default** output to forward logs to the internal instance. You do not need to create a **default** output. If you do configure a **default** output, you receive an error message because the **default** output is reserved for the Cluster Logging Operator.

NOTE

If you want to forward logs to **only** the internal OpenShift Container Platform Elasticsearch instance, you do not need to create a **ClusterLogForwarder** CR.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Procedure

1. Create a **ClusterLogForwarder** CR YAML file similar to the following:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
   - name: elasticsearch-insecure 3
     type: "elasticsearch" 4
     url: http://elasticsearch.insecure.com:9200 5
   - name: elasticsearch-secure
     type: "elasticsearch"
     url: https://elasticsearch.secure.com:9200
     secret:
        name: es-secret 6
  pipelines:
   - name: application-logs 7
     inputRefs: 8
     - application
     - audit
     outputRefs:
     - elasticsearch-secure 9
     - default 10
     labels:
       myLabel: "myValue" 11
   - name: infrastructure-audit-logs 12
     inputRefs:
     - infrastructure
     outputRefs:
     - elasticsearch-insecure
     labels:
       logs: "audit-infra"
```

1 The name of the **ClusterLogForwarder** CR must be **instance**.

2 The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.

3 Specify a name for the output.

4 Specify the **elasticsearch** type.

5 Specify the URL and port of the external Elasticsearch instance as a valid absolute URL. You can use the **http** (insecure) or **https** (secure HTTP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP Address.

6 If using an **https** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent.

7 Optional: Specify a name for the pipeline.

8 Specify which log types should be forwarded using that pipeline: **application, infrastructure**, or **audit**.

9 Specify the output to use with that pipeline for forwarding the logs.

10 Optional: Specify the **default** output to send the logs to the internal Elasticsearch instance.

11 Optional: String. One or more labels to add to the logs.

12 Optional: Configure multiple outputs to forward logs to other external log aggregators of any supported type:

- Optional. A name to describe the pipeline.
- The **inputRefs** is the log type to forward using that pipeline: **application, infrastructure**, or **audit**.
- The **outputRefs** is the name of the output to use.
- Optional: String. One or more labels to add to the logs.

2. Create the CR object:

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

The Cluster Logging Operator redeploys 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
```

6.1.2. 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 that you have configured to accept the protocol. You can do this in addition to, or instead of, using the default Elasticsearch log store. You must also configure the external log aggregator to receive log data from OpenShift Container Platform.

To configure log forwarding using the **forward** protocol, create a **ClusterLogForwarder** custom resource (CR) with one or more outputs to the Fluentd servers and pipelines that use those outputs. The Fluentd output can use a TCP (insecure) or TLS (secure TCP) connection.

NOTE

Alternately, you can use a config map to forward logs using the **forward** protocols. However, this method is deprecated in OpenShift Container Platform and will be removed in a future release.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Procedure

1. Create a **ClusterLogForwarder** CR YAML file similar to the following:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
   - name: fluentd-server-secure 3
     type: fluentdForward 4
     url: 'tls://fluentdserver.security.example.com:24224' 5
     secret: 6
        name: fluentd-secret
   - name: fluentd-server-insecure
     type: fluentdForward
     url: 'tcp://fluentdserver.home.example.com:24224'
  pipelines:
   - name: forward-to-fluentd-secure 7
     inputRefs:  8
     - application
     - audit
     outputRefs:
     - fluentd-server-secure 9
     - default 10
     labels:
       clusterId: "C1234" 11
   - name: forward-to-fluentd-insecure 12
     inputRefs:
     - infrastructure
     outputRefs:
     - fluentd-server-insecure
     labels:
       clusterId: "C1234"
```

1. The name of the **ClusterLogForwarder** CR must be **instance**.
2. The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
3. Specify a name for the output.
4. Specify the **fluentdForward** type.
5. Specify the URL and port of the external Fluentd instance as a valid absolute URL. You can use the **tcp** (insecure) or **tls** (secure TCP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP address.
6. If using a **tls** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent.
7. Optional. Specify a name for the pipeline.
8. Specify which log types should be forwarded using that pipeline: **application, infrastructure**, or **audit**.

9 Specify the output to use with that pipeline for forwarding the logs.

10 Optional. Specify the **default** output to forward logs to the internal Elasticsearch instance.

11 Optional: String. One or more labels to add to the logs.

12 Optional: Configure multiple outputs to forward logs to other external log aggregators of any supported type:

- Optional. A name to describe the pipeline.
- The **inputRefs** is the log type to forward using that pipeline: **application, infrastructure**, or **audit**.
- The **outputRefs** is the name of the output to use.
- Optional: String. One or more labels to add to the logs.

2. Create the CR object:

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

The Cluster Logging Operator redeploys 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
```

6.1.3. Forwarding logs using the syslog protocol

You can use the **syslog** RFC3164 or RFC5424 protocol to send a copy of your logs to an external log aggregator configured to accept the protocol instead of, or in addition to, the default Elasticsearch log store. You are responsible for configuring the external log aggregator, such as a syslog server, to receive the logs from OpenShift Container Platform.

To configure log forwarding using the **syslog** protocol, create a **ClusterLogForwarder** custom resource (CR) with one or more outputs to the syslog servers and pipelines that use those outputs. The syslog output can use a UDP, TCP, or TLS connection.

NOTE

Alternately, you can use a config map to forward logs using the **syslog** RFC3164 protocols. However, this method is deprecated in OpenShift Container Platform and will be removed in a future release.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Procedure

1. Create a **ClusterLogForwarder** CR YAML file similar to the following:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: rsyslog-east (3)
     type: syslog (4)
     syslog: (5)
       facility: local0
       rfc: RFC3164
       payloadKey: message
       severity: informational
     url: 'tls://rsyslogserver.east.example.com:514' (6)
     secret: (7)
        name: syslog-secret
   - name: rsyslog-west
     type: syslog
     syslog:
      appName: myapp
      facility: user
      msgID: mymsg
      procID: myproc
      rfc: RFC5424
      severity: debug
     url: 'udp://rsyslogserver.west.example.com:514'
  pipelines:
   - name: syslog-east (8)
     inputRefs: (9)
     - audit
     - application
     outputRefs: (10)
     - rsyslog-east
     - default (11)
     labels:
       secure: "true" (12)
       syslog: "east"
   - name: syslog-west (13)
     inputRefs:
     - infrastructure
     outputRefs:
     - rsyslog-west
     - default
     labels:
       syslog: "west"
```

1. The name of the **ClusterLogForwarder** CR must be **instance**.
2. The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
3. Specify a name for the output.
4. Specify the **syslog** type.

5 Optional. Specify the syslog parameters, listed below.

6 Specify the URL and port of the external syslog instance. You can use the **udp** (insecure), **tcp** (insecure) or **tls** (secure TCP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP address.

7 If using a **tls** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent.

8 Optional: Specify a name for the pipeline.

9 Specify which log types should be forwarded using that pipeline: **application,** **infrastructure**, or **audit**.

10 Specify the output to use with that pipeline for forwarding the logs.

11 Optional: Specify the **default** output to forward logs to the internal Elasticsearch instance.

12 Optional: String. One or more labels to add to the logs. Quote values like "true" so they are recognized as string values, not as a boolean.

13 Optional: Configure multiple outputs to forward logs to other external log aggregators of any supported type:

- Optional. A name to describe the pipeline.
- The **inputRefs** is the log type to forward using that pipeline: **application,** **infrastructure**, or **audit**.
- The **outputRefs** is the name of the output to use.
- Optional: String. One or more labels to add to the logs.

2. Create the CR object:

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

The Cluster Logging Operator redeploys 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
```

6.1.3.1. Syslog parameters

You can configure the following for the **syslog** outputs. For more information, see the syslog RFC3164 or RFC5424 RFC.

- facility: The syslog facility. The value can be a decimal integer or a case-insensitive keyword:
  - **0** or **kern** for kernel messages
  - **1** or **user** for user-level messages, the default.

- **2** or **mail** for the mail system
- **3** or **daemon** for system daemons
- **4** or **auth** for security/authentication messages
- **5** or **syslog** for messages generated internally by syslogd
- **6** or **lpr** for line printer subsystem
- **7** or **news** for the network news subsystem
- **8** or **uucp** for the UUCP subsystem
- **9** or **cron** for the clock daemon
- **10** or **authpriv** for security authentication messages
- **11** or **ftp** for the FTP daemon
- **12** or **ntp** for the NTP subsystem
- **13** or **security** for the syslog audit log
- **14** or **console** for the syslog alert log
- **15** or **solaris-cron** for the scheduling daemon
- **16–23** or **local0** – **local7** for locally used facilities

* Optional. **payloadKey**: The record field to use as payload for the syslog message.

NOTE

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

* rfc: The RFC to be used for sending log using syslog. The default is RFC5424.
* severity: The syslog severity to set on outgoing syslog records. The value can be a decimal integer or a case-insensitive keyword:
  - **0** or **Emergency** for messages indicating the system is unusable
  - **1** or **Alert** for messages indicating action must be taken immediately
  - **2** or **Critical** for messages indicating critical conditions
  - **3** or **Error** for messages indicating error conditions
  - **4** or **Warning** for messages indicating warning conditions
  - **5** or **Notice** for messages indicating normal but significant conditions
  - **6** or **Informational** for messages indicating informational messages
  - **7** or **Debug** for messages indicating debug-level messages, the default

- tag: Tag specifies a record field to use as tag on the syslog message.
- trimPrefix: Remove the specified prefix from the tag.

6.1.3.2. Additional RFC5424 syslog parameters

The following parameters apply to RFC5424:

- appName: The APP-NAME is a free-text string that identifies the application that sent the log. Must be specified for **RFC5424**.
- msgID: The MSGID is a free-text string that identifies the type of message. Must be specified for **RFC5424**.
- procID: The PROCID is a free-text string. A a change in the value indicates a discontinuity in syslog reporting. Must be specified for **RFC5424**.

6.1.4. Forwarding logs to a Kafka broker

You can forward logs to an external Kafka broker in addition to, or instead of, the default Elasticsearch log store.

To configure log forwarding to an external Kafka instance, create a **ClusterLogForwarder** custom resource (CR) with an output to that instance and a pipeline that uses the output. You can include a specific Kafka topic in the output or use the default. The Kafka output can use a TCP (insecure) or TLS (secure TCP) connection.

Procedure

1. Create a **ClusterLogForwarder** CR YAML file similar to the following:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: app-logs (3)
     type: kafka (4)
     url: tls://kafka.example.devlab.com:9093/app-topic (5)
     secret:
       name: kafka-secret (6)
   - name: infra-logs
     type: kafka
     url: tcp://kafka.devlab2.example.com:9093/infra-topic (7)
   - name: audit-logs
     type: kafka
     url: tls://kafka.qelab.example.com:9093/audit-topic
     secret:
        name: kafka-secret-qe
  pipelines:
   - name: app-topic (8)
     inputRefs: (9)
   - application
   outputRefs: (10)
   - app-logs
   labels:
     logType: "application" (11)
 - name: infra-topic (12)
   inputRefs:
   - infrastructure
   outputRefs:
   - infra-logs
   labels:
     logType: "infra"
 - name: audit-topic
   inputRefs:
   - audit
   outputRefs:
   - audit-logs
   - default (13)
   labels:
     logType: "audit"
```

1. The name of the **ClusterLogForwarder** CR must be **instance**.
2. The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
3. Specify a name for the output.
4. Specify the **kafka** type.
5. Specify the URL and port of the Kafka broker as a valid absolute URL, optionally with a specific topic. You can use the **tcp** (insecure) or **tls** (secure TCP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP address.
6. If using a **tls** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent.
7. Optional: To send an insecure output, use a **tcp** prefix in front of the URL. Also omit the **secret** key and its **name** from this output.
8. Optional: Specify a name for the pipeline.
9. Specify which log types should be forwarded using that pipeline: **application, infrastructure**, or **audit**.
10. Specify the output to use with that pipeline for forwarding the logs.
11. Optional: String. One or more labels to add to the logs.
12. Optional: Configure multiple outputs to forward logs to other external log aggregators of any supported type:
    - Optional. A name to describe the pipeline.

- The **inputRefs** is the log type to forward using that pipeline: **application, infrastructure**, or **audit**.
- The **outputRefs** is the name of the output to use.
- Optional: String. One or more labels to add to the logs.

13 Optional: Specify **default** to forward logs to the internal Elasticsearch instance.

2. Optional: To forward a single output to multiple kafka brokers, specify an array of kafka brokers as shown in this example:

```
...
spec:
  outputs:
  - name: app-logs
    type: kafka
    secret:
      name: kafka-secret-dev
    kafka: 1
      brokers: 2
        - tls://kafka-broker1.example.com:9093/
        - tls://kafka-broker2.example.com:9093/
      topic: app-topic 3
...
```

1 Specify a **kafka** key that has a **brokers** and **topic** key.

2 Use the **brokers** key to specify an array of one or more brokers.

3 Use the **topic** key to specify the target topic that will receive the logs.

3. Create the CR object:

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

The Cluster Logging Operator redeploys 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
```

6.1.5. Forwarding application logs from specific projects

You can use the Cluster Log Forwarder to send a copy of the application logs from specific projects to an external log aggregator. You can do this in addition to, or instead of, using the default Elasticsearch log store. You must also configure the external log aggregator to receive log data from OpenShift Container Platform.

To configure forwarding application logs from a project, create a **ClusterLogForwarder** custom resource (CR) with at least one input from a project, optional outputs for other log aggregators, and pipelines that use those inputs and outputs.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Procedure

1. Create a **ClusterLogForwarder** CR YAML file similar to the following:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: fluentd-server-secure (3)
     type: fluentdForward (4)
     url: 'tls://fluentdserver.security.example.com:24224' (5)
     secret: (6)
        name: fluentd-secret
   - name: fluentd-server-insecure
     type: fluentdForward
     url: 'tcp://fluentdserver.home.example.com:24224'
  inputs: (7)
   - name: my-app-logs
     application:
        namespaces:
        - my-project
  pipelines:
   - name: forward-to-fluentd-insecure (8)
     inputRefs: (9)
     - my-app-logs
     outputRefs: (10)
     - fluentd-server-insecure
     labels: (11)
       project: "my-project"
   - name: forward-to-fluentd-secure (12)
     inputRefs:
     - application
     - audit
     - infrastructure
     outputRefs:
     - fluentd-server-secure
     - default
     labels:
       clusterId: "C1234"
```

(1) The name of the **ClusterLogForwarder** CR must be **instance**.

(2) The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.

(3) Specify a name for the output.

4. Specify the output type: **elasticsearch**, **fluentdForward**, **syslog**, or **kafka**.

5. Specify the URL and port of the external log aggregator as a valid absolute URL. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP address.

6. If using a **tls** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent.

7. Configuration for an input to filter application logs from the specified projects.

8. Configuration for a pipeline to use the input to send project application logs to an external Fluentd instance.

9. The **my-app-logs** input.

10. The name of the output to use.

11. Optional: String. One or more labels to add to the logs.

12. Configuration for a pipeline to send logs to other log aggregators.

    - Optional: Specify a name for the pipeline.
    - Specify which log types should be forwarded using that pipeline: **application, infrastructure**, or **audit**.
    - Specify the output to use with that pipeline for forwarding the logs.
    - Optional: Specify the **default** output to forward logs to the internal Elasticsearch instance.
    - Optional: String. One or more labels to add to the logs.

2. Create the CR object:

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

6.1.6. Forwarding logs using the legacy Fluentd method

You can use the Fluentd **forward** protocol to send logs to destinations outside of your OpenShift Container Platform cluster by creating a configuration file and config map. You are responsible for configuring the external log aggregator to receive log data from OpenShift Container Platform.

IMPORTANT

This method for forwarding logs is deprecated in OpenShift Container Platform and will be removed in a future release.

To send logs using the Fluentd **forward** protocol, create a configuration file called **secure-forward.conf**, that points to an external log aggregator. Then, use that file to create a config map called called **secure-forward** in the **openshift-logging** project, which OpenShift Container Platform uses when forwarding the logs.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Sample Fluentd configuration file

```
<store>
  @type forward
  <security>
    self_hostname ${hostname}
    shared_key "fluent-receiver"
  </security>
  transport tls
  tls_verify_hostname false
  tls_cert_path '/etc/ocp-forward/ca-bundle.crt'
  <buffer>
    @type file
    path '/var/lib/fluentd/secureforwardlegacy'
    queued_chunks_limit_size "1024"
    chunk_limit_size "1m"
    flush_interval "5s"
    flush_at_shutdown "false"
    flush_thread_count "2"
    retry_max_interval "300"
    retry_forever true
    overflow_action "#{ENV['BUFFER_QUEUE_FULL_ACTION'] || 'throw_exception'}"
  </buffer>
  <server>
    host fluent-receiver.example.com
    port 24224
  </server>
</store>
```

Procedure

To configure OpenShift Container Platform to forward logs using the legacy Fluentd method:

1. Create a configuration file named **secure-forward** and specify parameters similar to the following within the **<store>** stanza:

```
<store>
  @type forward
  <security>
    self_hostname ${hostname}
    shared_key <key> (1)
  </security>
  transport tls (2)
  tls_verify_hostname <value> (3)
  tls_cert_path <path_to_file> (4)
  <buffer> (5)
    @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
  </buffer>
  <server>
    name 6
    host 7
    hostlabel 8
    port 9
  </server>
  <server> 10
    name
    host
  </server>
```

1. Enter the shared key between nodes.
2. Specify **tls** to enable TLS validation.
3. Set to **true** to verify the server cert hostname. Set to **false** to ignore server cert hostname.
4. Specify the path to the private CA certificate file as **/etc/ocp-forward/ca_cert.pem**.
5. Specify the Fluentd buffer parameters as needed.
6. Optionally, enter a name for this server.
7. Specify the hostname or IP of the server.
8. Specify the host label of the server.
9. Specify the port of the server.
10. Optionally, add additional servers. If you specify two or more servers, **forward** uses these server nodes in a round-robin order.

To use Mutual TLS (mTLS) authentication, see the Fluentd documentation for information about client certificate, key parameters, and other settings.

2. Create a config map named **secure-forward** in the **openshift-logging** project from the configuration file:

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

The Cluster Logging Operator redeploys 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
```

6.1.7. Forwarding logs using the legacy syslog method

You can use the **syslog** RFC3164 protocol to send logs to destinations outside of your OpenShift

Container Platform cluster by creating a configuration file and config map. You are responsible for configuring the external log aggregator, such as a syslog server, to receive the logs from OpenShift Container Platform.

IMPORTANT

This method for forwarding logs is deprecated in OpenShift Container Platform and will be removed 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 and buffers data into chunks.

To send logs 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 config map called **syslog** in the **openshift-logging** project, which OpenShift Container Platform uses when forwarding the logs.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Sample syslog configuration file

```
<store>
@type syslog_buffered
remote_syslog rsyslogserver.example.com
port 514
hostname ${hostname}
remove_tag_prefix tag
facility local0
severity info
use_record true
payload_key message
rfc 3164
</store>
```

You can configure the following **syslog** parameters. For more information, see the syslog RFC3164.

- facility: The syslog facility. The value can be a decimal integer or a case-insensitive keyword:
  - **0** or **kern** for kernel messages
  - **1** or **user** for user-level messages, the default.
  - **2** or **mail** for the mail system
  - **3** or **daemon** for the system daemons
  - **4** or **auth** for the security/authentication messages

- **5** or **syslog** for messages generated internally by syslogd
- **6** or **lpr** for the line printer subsystem
- **7** or **news** for the network news subsystem
- **8** or **uucp** for the UUCP subsystem
- **9** or **cron** for the clock daemon
- **10** or **authpriv** for security authentication messages
- **11** or **ftp** for the FTP daemon
- **12** or **ntp** for the NTP subsystem
- **13** or **security** for the syslog audit logs
- **14** or **console** for the syslog alert logs
- **15** or **solaris-cron** for the scheduling daemon
- **16**–**23** or **local0** – **local7** for locally used facilities

- payloadKey: The record field to use as payload for the syslog message.
- rfc: The RFC to be used for sending log using syslog.
- severity: The syslog severity to set on outgoing syslog records. The value can be a decimal integer or a case-insensitive keyword:
    - **0** or **Emergency** for messages indicating the system is unusable
    - **1** or **Alert** for messages indicating action must be taken immediately
    - **2** or **Critical** for messages indicating critical conditions
    - **3** or **Error** for messages indicating error conditions
    - **4** or **Warning** for messages indicating warning conditions
    - **5** or **Notice** for messages indicating normal but significant conditions
    - **6** or **Informational** for messages indicating informational messages
    - **7** or **Debug** for messages indicating debug-level messages, the default
- tag: The record field to use as tag on the syslog message.
- trimPrefix: The prefix to remove from the tag.

Procedure

To configure OpenShift Container Platform to forward logs using the legacy configuration methods:

1. Create a configuration file named **syslog.conf** and specify parameters similar to the following within the **<store>** stanza:

```
<store>
@type <type> (1)
remote_syslog <syslog-server> (2)
port 514 (3)
hostname ${hostname}
remove_tag_prefix <prefix> (4)
facility <value>
severity <value>
use_record <value>
payload_key message
rfc 3164 (5)
</store>
```

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

(2) Specify the FQDN or IP address of the syslog server.

(3) Specify the port of the syslog server.

(4) Optional: Specify the appropriate syslog parameters, for example:

- Parameter to remove the specified **tag** field from the syslog prefix.
- Parameter to set the specified field as the syslog key.
- Parameter to specify the syslog log facility or source.
- Parameter to specify the syslog log severity.
- Parameter 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.
- Parameter to specify the key to set the payload of the syslog message. Defaults to **message**.

(5) With the legacy syslog method, you must specify **3164** for the **rfc** value.

2. Create a config map named **syslog** in the **openshift-logging** project from the configuration file:

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

The Cluster Logging Operator redeploys 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
```

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:

```
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 (1)
    apiVersion: v1
    metadata:
      name: eventrouter
      namespace: ${NAMESPACE}
  - kind: ClusterRole (2)
    apiVersion: v1
    metadata:
      name: event-reader
    rules:
```

```
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "watch", "list"]
- kind: ClusterRoleBinding 3
  apiVersion: v1
  metadata:
    name: event-reader-binding
  subjects:
  - kind: ServiceAccount
    name: eventrouter
    namespace: ${NAMESPACE}
  roleRef:
    kind: ClusterRole
    name: event-reader
- kind: ConfigMap 4
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  data:
    config.json: |-
      {
        "sink": "stdout"
      }
- kind: Deployment 5
  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}
                memory: ${MEMORY}
              volumeMounts:
              - name: config-volume
                mountPath: /etc/eventrouter
            volumes:
              - name: config-volume
                configMap:
                  name: eventrouter
parameters:
  - name: IMAGE
    displayName: Image
    value: "registry.redhat.io/openshift4/ose-logging-eventrouter:latest"
  - name: CPU  6
    displayName: CPU
    value: "100m"
  - name: MEMORY  7
    displayName: Memory
    value: "128Mi"
  - name: NAMESPACE
    displayName: Namespace
    value: "openshift-logging"  8
```

1 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:

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

For example:

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

Example output

```
serviceaccount/logging-eventrouter created
clusterrole.authorization.openshift.io/event-reader created
```

```
clusterrolebinding.authorization.openshift.io/event-reader-binding created
configmap/logging-eventrouter created
deployment.apps/logging-eventrouter 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

   ```
   {"verb":"ADDED","event":{"metadata":{"name":"openshift-service-catalog-controller-
   manager-remover.1632d931e88fcd8f","namespace":"openshift-service-catalog-
   removed","selfLink":"/api/v1/namespaces/openshift-service-catalog-
   removed/events/openshift-service-catalog-controller-manager-
   remover.1632d931e88fcd8f","uid":"787d7b26-3d2f-4017-b0b0-
   420db4ae62c0","resourceVersion":"21399","creationTimestamp":"2020-09-
   08T15:40:26Z"},"involvedObject":{"kind":"Job","namespace":"openshift-service-catalog-
   removed","name":"openshift-service-catalog-controller-manager-
   remover","uid":"fac9f479-4ad5-4a57-8adc-
   cb25d3d9cf8f","apiVersion":"batch/v1","resourceVersion":"21280"},"reason":"Completed","
   message":"Job completed","source":{"component":"job-
   controller"},"firstTimestamp":"2020-09-08T15:40:26Z","lastTimestamp":"2020-09-
   08T15:40:26Z","count":1,"type":"Normal"}}
   ```

   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 OpenShift 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 4.5 to 4.6 by changing the subscription for the OpenShift Elasticsearch Operator and the Cluster Logging Operator.

When you update:

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

  If you update the Cluster Logging Operator before the OpenShift 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 4.5, you must upgrade cluster logging to 4.5 before updating to 4.6.

Prerequisites

- Update the OpenShift Container Platform cluster from 4.5 to 4.6.
- Make sure the cluster logging status is healthy:
  - All pods are **ready**.
  - The Elasticsearch cluster is healthy.

- Back up your Elasticsearch and Kibana data.

Procedure

1. Update the OpenShift Elasticsearch Operator:
   a. From the web console, click **Operators → Installed Operators**.
   b. Select the **openshift-operators-redhat** project.
   c. Click the **OpenShift Elasticsearch Operator**.
   d. Click **Subscription → Channel**.
   e. In the **Change Subscription Update Channel** window, select **4.6** and click **Save**.
   f. Wait for a few seconds, then click **Operators → Installed Operators**.
      The OpenShift Elasticsearch Operator is shown as 4.6. For example:

      ```
      OpenShift Elasticsearch Operator
      4.6.0-202007012112.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 **4.6** and click **Save**.
   f. Wait for a few seconds, then click **Operators → Installed Operators**.
      The Cluster Logging Operator is shown as 4.6. For example:

      ```
      Cluster Logging
      4.6.0-202007012112.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

      ```
      NAME                                           READY   STATUS    RESTARTS   AGE
      elasticsearch-cdm-1pbrl44l-1-55b7546f4c-mshhk   2/2     Running   0          31m
      ```

```
elasticsearch-cdm-1pbrl44l-2-5c6d87589f-gx5hk   2/2     Running   0          30m
elasticsearch-cdm-1pbrl44l-3-88df5d47-m45jc     2/2     Running   0          29m
```

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
```

```
NAME                    SCHEDULE        SUSPEND   ACTIVE   LAST SCHEDULE   AGE
curator               30 3,9,15,21 * * * False 0        <none>          20s
elasticsearch-im-app     */15 * * * *    False      0        <none>          56s
elasticsearch-im-audit   */15 * * * *    False      0        <none>          56s
elasticsearch-im-infra   */15 * * * *    False      0        <none>          56s
```

d. Verify that the log store is updated to 4.6 and the indices are **green**:

```
$ oc exec -c elasticsearch <any_es_pod_in_the_cluster> -- indices
```

Verify 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
bnBvUFEXTWi92z3zWAzieQ   3 1       222195            0        289            144
green  open   infra-000004
rtDSzoqsSl6saisSK7Au1Q   3 1       226717            0        297            148
green  open   infra-000012
RSf_kUwDSR2xEuKRZMPqZQ   3 1       227623            0        295            147
green  open   .kibana_7
1SJdCqlZTPWlIAaOUd78yg   1 1            4            0          0              0
green  open   infra-000010
iXwL3bnqTuGEABbUDa6OVw   3 1       248368            0        317            158
green  open   infra-000009
YN9EsULWSNaxWeeNvOs0RA   3 1       258799            0        337            168
green  open   infra-000014
YP0U6R7FQ_GVQVQZ6Yh9Ig   3 1       223788            0        292            146
green  open   infra-000015
```

```
JRBbAbEmSMqK5X40df9HbQ   3 1      224371          0       291           145
green  open   .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA   3 1           9          0         0             0
green  open   infra-000007
IlkkAVSzSOmosWTSAJM_hg   3 1      228584          0       296           148
green  open   infra-000005
d9BoGQdiQASsS3BBFm2iRA   3 1      227987          0       297           148
green  open   infra-000003                                            1-
goREK1QUKlQPAIVkWVaQ   3 1      226719          0       295           147
green  open   .security
zeT65uOuRTKZMjg_bbUc1g   1 1           5          0         0             0
green  open   .kibana-377444158_kubeadmin                                 wvMhDwJkR-
mRZQO84K0gUQ   3 1           1          0         0             0
green  open   infra-000006                                            5H-
KBSXGQKiO7hdapDE23g   3 1      226676          0       295           147
green  open   infra-000001                                            eH53BQ-
bSxSWR5xYZB6lVg   3 1      341800          0       443           220
green  open   .kibana-6
RVp7TemSSemGJcsSUmuf3A   1 1           4          0         0             0
green  open   infra-000011
J7XWBauWSTe0jnzX02fU6A   3 1      226100          0       293           146
green  open   app-000001
axSAFfONQDmKwatkjPXdtw   3 1      103186          0       126            57
green  open   infra-000016
m9c1iRLtStWSF1GopaRyCg   3 1       13685          0        19             9
green  open   infra-000002                                            Hz6WvINtTvKcQzw-
ewmbYg   3 1      228994          0       296           148
green  open   infra-000013                                            KR9mMFUpQl-
jraYtanyIGw   3 1      228166          0       298           148
green  open   audit-000001
eERqLdLmQOiQDFES1LBATQ   3 1           0          0         0             0
```

e. Verify that the log collector is updated to 4.6:

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

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

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

f. Verify that the log visualizer is updated to 4.6 using the Kibana CRD:

```
$ oc get kibana kibana -o json
```

Verify that the output includes a Kibana pod with the **ready** status:

Example 8.2. Sample output with a ready Kibana pod

```
[
{
"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 4.6:

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

```
cronjob.batch/curator
cronjob.batch/elasticsearch-im-app
cronjob.batch/elasticsearch-im-audit
cronjob.batch/elasticsearch-im-infra
```

Verify that the output includes the **elasticsearch-im-*** indices.

Post-update tasks

If you use the Log Forwarding API to forward logs, after the OpenShift Elasticsearch Operator and Cluster Logging Operator are fully updated to 4.6, you must replace your **LogForwarding** custom resource (CR) with a **ClusterLogForwarder** CR.

8.2. UPDATING LOG FORWARDING CUSTOM RESOURCES

The OpenShift Container Platform Log Forward API has been promoted from Technology Preview to Generally Available in OpenShift Container Platform 4.6. The GA release contains some improvements and enhancements that require you to make a change to your **ClusterLogging** custom resource (CR) and to replace your **LogForwarding** custom resource (CR) with a **ClusterLogForwarder** CR.

Sample ClusterLogForwarder instance in OpenShift Container Platform 4.6

```
apiVersion: logging.openshift.io/v1
```

```
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
....
spec:
  outputs:
  - url: http://remote.elasticsearch.com:9200
    name: elasticsearch
    type: elasticsearch
  - url: tls://fluentdserver.example.com:24224
    name: fluentd
    type: fluentdForward
    secret:
      name: fluentdserver
  pipelines:
  - inputRefs:
      - infrastructure
      - application
    name: mylogs
    outputRefs:
     - elasticsearch
  - inputRefs:
      - audit
    name: auditlogs
    outputRefs:
      - fluentd
      - default
...
```

Sample ClusterLogForwarder CR in OpenShift Container Platform 4.5

```
apiVersion: logging.openshift.io/v1alpha1
kind: LogForwarding
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
   - name: elasticsearch
     type: elasticsearch
     endpoint: remote.elasticsearch.com:9200
   - name: fluentd
     type: forward
     endpoint: fluentdserver.example.com:24224
     secret:
       name: fluentdserver
  pipelines:
   - inputSource: logs.infra
     name: infra-logs
     outputRefs:
     - elasticsearch
   - inputSource: logs.app
     name: app-logs
```

```
    outputRefs:
     - elasticsearch
  - inputSource: logs.audit
    name: audit-logs
    outputRefs:
     - fluentd
```

The following procedure shows each parameter you must change.

Procedure

To update the **ClusterLogForwarder** CR in 4.5 to the **ClusterLogForwarding** CR for 4.6, make the following modifications:

1. Edit the **ClusterLogging** custom resource (CR) to remove the **logforwardingtechpreview** annotation:

   Sample ClusterLogging CR

   ```
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     annotations:
       clusterlogging.openshift.io/logforwardingtechpreview: enabled (1)
     name: "instance"
     namespace: "openshift-logging"
   ....
   ```

   (1) Remove the **logforwardingtechpreview** annotation.

2. Export the **ClusterLogForwarder** CR to create a YAML file for the **ClusterLogForwarder** instance:

   ```
   $ oc get LogForwarding instance -n openshift-logging -o yaml| tee ClusterLogForwarder.yaml
   ```

3. Edit the YAML file to make the following modifications:

   Sample ClusterLogForwarder instance in OpenShift Container Platform 4.6

   ```
   apiVersion: logging.openshift.io/v1 (1)
   kind: ClusterLogForwarder (2)
   metadata:
     name: instance
     namespace: openshift-logging
   ....
   spec: (3)
     outputs:
     - url: http://remote.elasticsearch.com:9200 (4)
       name: elasticsearch
       type: elasticsearch
     - url: tls://fluentdserver.example.com:24224
       name: fluentd
       type: fluentdForward (5)
   ```

```
    secret:
      name: fluentdserver
  pipelines:
  - inputRefs: 6
      - infrastructure
      - application
    name: mylogs
    outputRefs:
     - elasticsearch
  - inputRefs:
      - audit
    name: auditlogs
    outputRefs:
      - fluentd
      - default 7
...
```

1. Change the **apiVersion** from **"logging.openshift.io/v1alpha1"** to **"logging.openshift.io/v1"**.

2. Change the object kind from **kind: "LogForwarding"** to **kind: "ClusterLogForwarder"**.

3. Remove the **disableDefaultForwarding: true** parameter.

4. Change the output parameter from **spec.outputs.endpoint** to **spec.outputs.url**. Add a prefix to the URL, such as **https:**//, **tcp:**//, and so forth, if a prefix is not present.

5. For Fluentd outputs, change the **type** from **forward** to **fluentdForward**.

6. Change the pipelines:

    - Change **spec.pipelines.inputSource** to **spec.pipelines.inputRefs**
    - Change **logs.infra** to **infrastructure**
    - Change **logs.app** to **application**
    - Change **logs.audit** to **audit**

7. Optional: Add a **default** pipeline to send logs to the internal Elasticsearch instance. You are not required to configure a **default** output.

NOTE

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

4. Create the CR object:

```
$ oc create -f ClusterLogForwarder.yaml
```

For information on the new capabilities of the Log Forwarding API, see Forwarding logs to third party systems.

CHAPTER 9. VIEWING CLUSTER DASHBOARDS

The **Logging/Elasticsearch Nodes** and **Openshift Logging** dashboards in the OpenShift Container Platform web console show in-depth details about your Elasticsearch instance and the individual Elasticsearch nodes that you can use to prevent and diagnose problems.

The **OpenShift Logging** dashboard contains charts that show details about your Elasticsearch instance at a cluster level, including cluster resources, garbage collection, shards in the cluster, and Fluentd statistics.

The **Logging/Elasticsearch Nodes** dashboard contains charts that show details about your Elasticsearch instance, many at node level, including details on indexing, shards, resources, and so forth.

NOTE

For more detailed data, click the **Grafana UI** link in a dashboard to launch the Grafana dashboard. Grafana is shipped with OpenShift cluster monitoring.

9.1. ACCESSING THE ELASTISEARCH AND OPENSHIFT LOGGING DASHBOARDS

You can view the **Logging/Elasticsearch Nodes** and **OpenShift Logging** dashboards in the OpenShift Container Platform web console.

Procedure

To launch the dashboards:

1. In the OpenShift Container Platform web console, click **Monitoring** → **Dashboards**.
2. On the **Dashboards** page, select **Logging/Elasticsearch Nodes** or **OpenShift Logging** from the **Dashboard** menu.
   For the **Logging/Elasticsearch Nodes** dashboard, you can select the Elasticsearch node you want to view and set the data resolution.

   The appropriate dashboard is displayed, showing multiple charts of data.
3. Optionally, select a different time range to display or refresh rate for the data from the **Time Range** and **Refresh Interval** menus.

NOTE

For more detailed data, click the **Grafana UI** link to launch the Grafana dashboard.

For information on the dashboard charts, see About the OpenShift Logging dashboard and About the Logging/Elastisearch Nodes dashboard.

9.2. ABOUT THE OPENSHIFT LOGGING DASHBOARD

The **OpenShift Logging** dashboard contains charts that show details about your Elasticsearch instance at a cluster-level that you can use to diagnose and anticipate problems.

Table 9.1. OpenShift Logging charts

Metric	Description
Elastic Cluster Status	The current Elasticsearch status: ● ONLINE - Indicates that the Elasticsearch instance is online. ● OFFLINE - Indicates that the Elasticsearch instance is offline.
Elastic Nodes	The total number of Elasticsearch nodes in the Elasticsearch instance.
Elastic Shards	The total number of Elasticsearch shards in the Elasticsearch instance.
Elastic Documents	The total number of Elasticsearch documents in the Elasticsearch instance.
Total Index Size on Disk	The total disk space that is being used for the Elasticsearch indices.
Elastic Pending Tasks	The total number of Elasticsearch changes that have not been completed, such as index creation, index mapping, shard allocation, or shard failure.
Elastic JVM GC time	The amount of time that the JVM spent executing Elasticsearch garbage collection operations in the cluster.
Elastic JVM GC Rate	The total number of times that JVM executed garbage activities per second.
Elastic Query/Fetch Latency Sum	● Query latency: The average time each Elasticsearch search query takes to execute. ● Fetch latency: The average time each Elasticsearch search query spends fetching data. Fetch latency typically takes less time than query latency. If fetch latency is consistently increasing, it might indicate slow disks, data enrichment, or large requests with too many results.
Elastic Query Rate	The total queries executed against the Elasticsearch instance per second for each Elasticsearch node.
CPU	The amount of CPU used by Elasticsearch, Fluentd, and Kibana, shown for each component.

Metric	Description
Elastic JVM Heap Used	The amount of JVM memory used. In a healthy cluster, the graph shows regular drops as memory is freed by JVM garbage collection.
Elasticsearch Disk Usage	The total disk space used by the Elasticsearch instance for each Elasticsearch node.
File Descriptors In Use	The total number of file descriptors used by Elasticsearch, Fluentd, and Kibana.
FluentD emit count	The total number of Fluentd messages per second for the Fluentd default output, and the retry count for the default output.
FluentD Buffer Availability	The percent of the Fluentd buffer that is available for chunks. A full buffer might indicate that Fluentd is not able to process the number of logs received.
Elastic rx bytes	The total number of bytes that Elasticsearch has received from FluentD, the Elasticsearch nodes, and other sources.
Elastic Index Failure Rate	The total number of times per second that an Elasticsearch index fails. A high rate might indicate an issue with indexing.
FluentD Output Error Rate	The total number of times per second that FluentD is not able to output logs.

9.3. CHARTS ON THE LOGGING/ELASTICSEARCH NODES DASHBOARD

The **Logging/Elasticsearch Nodes** dashboard contains charts that show details about your Elasticsearch instance, many at node-level, for further diagnostics.

Elasticsearch status

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about the status of your Elasticsearch instance.

Table 9.2. Elasticsearch status fields

Metric	Description

Metric	Description
Cluster status	The cluster health status during the selected time period, using the Elasticsearch green, yellow, and red statuses: ● 0 – Indicates that the Elasticsearch instance is in green status, which means that all shards are allocated. ● 1 – Indicates that the Elasticsearch instance is in yellow status, which means that replica shards for at least one shard are not allocated. ● 2 – Indicates that the Elasticsearch instance is in red status, which means that at least one primary shard and its replicas are not allocated.
Cluster nodes	The total number of Elasticsearch nodes in the cluster.
Cluster data nodes	The number of Elasticsearch data nodes in the cluster.
Cluster pending tasks	The number of cluster state changes that are not finished and are waiting in a cluster queue, for example, index creation, index deletion, or shard allocation. A growing trend indicates that the cluster is not able to keep up with changes.

Elasticsearch cluster index shard status

Each Elasticsearch index is a logical group of one or more shards, which are basic units of persisted data. There are two types of index shards: primary shards, and replica shards. When a document is indexed into an index, it is stored in one of its primary shards and copied into every replica of that shard. The number of primary shards is specified when the index is created, and the number cannot change during index lifetime. You can change the number of replica shards at any time.

The index shard can be in several states depending on its lifecycle phase or events occurring in the cluster. When the shard is able to perform search and indexing requests, the shard is active. If the shard cannot perform these requests, the shard is non–active. A shard might be non-active if the shard is initializing, reallocating, unassigned, and so forth.

Index shards consist of a number of smaller internal blocks, called index segments, which are physical representations of the data. An index segment is a relatively small, immutable Lucene index that is created when Lucene commits newly-indexed data. Lucene, a search library used by Elasticsearch, merges index segments into larger segments in the background to keep the total number of segments low. If the process of merging segments is slower than the speed at which new segments are created, it could indicate a problem.

When Lucene performs data operations, such as a search operation, Lucene performs the operation against the index segments in the relevant index. For that purpose, each segment contains specific data structures that are loaded in the memory and mapped. Index mapping can have a significant impact on the memory used by segment data structures.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about the Elasticsearch index shards.

Table 9.3. Elasticsearch cluster shard status charts

Metric	Description
Cluster active shards	The number of active primary shards and the total number of shards, including replicas, in the cluster. If the number of shards grows higher, the cluster performance can start degrading.
Cluster initializing shards	The number of non-active shards in the cluster. A non-active shard is one that is initializing, being reallocated to a different node, or is unassigned. A cluster typically has non–active shards for short periods. A growing number of non–active shards over longer periods could indicate a problem.
Cluster relocating shards	The number of shards that Elasticsearch is relocating to a new node. Elasticsearch relocates nodes for multiple reasons, such as high memory use on a node or after a new node is added to the cluster.
Cluster unassigned shards	The number of unassigned shards. Elasticsearch shards might be unassigned for reasons such as a new index being added or the failure of a node.

Elasticsearch node metrics

Each Elasticsearch node has a finite amount of resources that can be used to process tasks. When all the resources are being used and Elasticsearch attempts to perform a new task, Elasticsearch put the tasks into a queue until some resources become available.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about resource usage for a selected node and the number of tasks waiting in the Elasticsearch queue.

Table 9.4. Elasticsearch node metric charts

Metric	Description
ThreadPool tasks	The number of waiting tasks in individual queues, shown by task type. A long–term accumulation of tasks in any queue could indicate node resource shortages or some other problem.
CPU usage	The amount of CPU being used by the selected Elasticsearch node as a percentage of the total CPU allocated to the host container.
Memory usage	The amount of memory being used by the selected Elasticsearch node.

Metric	Description
Disk usage	The total disk space being used for index data and metadata on the selected Elasticsearch node.
Documents indexing rate	The rate that documents are indexed on the selected Elasticsearch node.
Indexing latency	The time taken to index the documents on the selected Elasticsearch node. Indexing latency can be affected by many factors, such as JVM Heap memory and overall load. A growing latency indicates a resource capacity shortage in the instance.
Search rate	The number of search requests run on the selected Elasticsearch node.
Search latency	The time taken to complete search requests on the selected Elasticsearch node. Search latency can be affected by many factors. A growing latency indicates a resource capacity shortage in the instance.
Documents count (with replicas)	The number of Elasticsearch documents stored on the selected Elasticsearch node, including documents stored in both the primary shards and replica shards that are allocated on the node.
Documents deleting rate	The number of Elasticsearch documents being deleted from any of the index shards that are allocated to the selected Elasticsearch node.
Documents merging rate	The number of Elasticsearch documents being merged in any of index shards that are allocated to the selected Elasticsearch node.

Elasticsearch node fielddata

Fielddata is an Elasticsearch data structure that holds lists of terms in an index and is kept in the JVM Heap. Because fielddata building is an expensive operation, Elasticsearch caches the fielddata structures. Elasticsearch can evict a fielddata cache when the underlying index segment is deleted or merged, or if there is not enough JVM HEAP memory for all the fielddata caches.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about Elasticsearch fielddata.

Table 9.5. Elasticsearch node fielddata charts

Metric	Description

Metric	Description
Fielddata memory size	The amount of JVM Heap used for the fielddata cache on the selected Elasticsearch node.
Fielddata evictions	The number of fielddata structures that were deleted from the selected Elasticsearch node.

Elasticsearch node query cache

If the data stored in the index does not change, search query results are cached in a node-level query cache for reuse by Elasticsearch.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about the Elasticsearch node query cache.

Table 9.6. Elasticsearch node query charts

Metric	Description
Query cache size	The total amount of memory used for the query cache for all the shards allocated to the selected Elasticsearch node.
Query cache evictions	The number of query cache evictions on the selected Elasticsearch node.
Query cache hits	The number of query cache hits on the selected Elasticsearch node.
Query cache misses	The number of query cache misses on the selected Elasticsearch node.

Elasticsearch index throttling

When indexing documents, Elasticsearch stores the documents in index segments, which are physical representations of the data. At the same time, Elasticsearch periodically merges smaller segments into a larger segment as a way to optimize resource use. If the indexing is faster then the ability to merge segments, the merge process does not complete quickly enough, which can lead to issues with searches and performance. To prevent this situation, Elasticsearch throttles indexing, typically by reducing the number of threads allocated to indexing down to a single thread.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about Elasticsearch index throttling.

Table 9.7. Index throttling charts

Metric	Description
Indexing throttling	The amount of time that Elasticsearch has been throttling the indexing operations on the selected Elasticsearch node.

Metric	Description
Merging throttling	The amount of time that Elasticsearch has been throttling the segment merge operations on the selected Elasticsearch node.

Node JVM Heap statistics

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about JVM Heap operations.

Table 9.8. JVM Heap statistic charts

Metric	Description
Heap used	The amount of the total allocated JVM Heap space that is used on the selected Elasticsearch node.
GC count	The number of garbage collection operations that have been run on the selected Elasticsearch node, by old and young garbage collection.
GC time	The amount of time that the JVM spent running garbage collection operations on the selected Elasticsearch node, by old and young garbage collection.

CHAPTER 10. TROUBLESHOOTING CLUSTER LOGGING

10.1. VIEWING CLUSTER LOGGING STATUS

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

10.1.1. 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

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

....

status: (1)
  collection:
    logs:
      fluentdStatus:
        daemonSet: fluentd (2)
        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
```

```
     - fluentd-6l2ff
     - fluentd-flpnn
     - fluentd-n2frh
 logstore: 3
  elasticsearchStatus:
  - ShardAllocationEnabled:  all
   cluster:
    activePrimaryShards:    5
    activeShards:           5
    initializingShards:     0
    numDataNodes:           1
    numNodes:               1
    pendingTasks:           0
    relocatingShards:       0
    status:                 green
    unassignedShards:       0
   clusterName:             elasticsearch
   nodeConditions:
    elasticsearch-cdm-mkkdys93-1:
   nodeCount:  1
   pods:
    client:
     failed:
     notReady:
     ready:
     - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
    data:
     failed:
     notReady:
     ready:
     - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
    master:
     failed:
     notReady:
     ready:
     - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
visualization:  4
  kibanaStatus:
  - deployment: kibana
   pods:
    failed: []
    notReady: []
    ready:
    - kibana-7fb4fd4cc9-f2nls
   replicaSets:
   - kibana-7fb4fd4cc9
   replicas: 1
```

1. In the output, the cluster status fields appear in the **status** stanza.

2. Information on the Fluentd pods.

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

4. Information on the Kibana pods.

10.1.1.1. 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:
  - lastTransitionTime: 2019-03-15T15:57:22Z
    message: Disk storage usage for node is 27.5gb (36.74%). Shards will be 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
    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-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
      Data:
        Failed:
        Not Ready:
          elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
      Master:
        Failed:
        Not Ready:
          elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
```

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:               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:

Example output

```
Status:
  Collection:
    Logs:
      Fluentd Status:
        Daemon Set:  fluentd
        Nodes:
        Pods:
```

```
        Failed:
        Not Ready:
        Ready:
```

10.1.2. 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.

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

2. View the status of the cluster logging environment:

   ```
   $ 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

      ```
      $ oc get replicaset
      ```

      Example output

      ```
      NAME                                DESIRED   CURRENT   READY   AGE
      ```

```
cluster-logging-operator-574b8987df       1         1         1       159m
elasticsearch-cdm-uhr537yu-1-6869694fb    1         1         1       157m
elasticsearch-cdm-uhr537yu-2-857b6d676f   1         1         1       156m
elasticsearch-cdm-uhr537yu-3-5b6fdd8cfd   1         1         1       155m
kibana-5bd5544f87                         1         1         1       157m
```

b. Get the status of the replica set:

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

Example output

```
Name:          cluster-logging-operator-574b8987df

....

Replicas:       1 current / 1 desired
Pods Status:    1 Running / 0 Waiting / 0 Succeeded / 0 Failed

....

Events:
  Type    Reason           Age   From                   Message
  ----    ------           ----  ----                   -------
  Normal  SuccessfulCreate  66m   replicaset-controller  Created pod: cluster-logging-
operator-574b8987df-qjhqv----
```

10.2. VIEWING THE STATUS OF THE LOG STORE

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

10.2.1. 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

```
NAME           AGE
elasticsearch   5h9m
```

b. Get the log store status:

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

For example:

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

The output includes information similar to the following:

Example output

```
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:
```

```
    - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
  shardAllocationEnabled: all
```

1. In the output, the cluster status fields appear in the **status** stanza.
2. 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.
3. 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**.
4. The log store nodes in the cluster, with **upgradeStatus**.
5. The log store client, data, and master pods in the cluster, listed under 'failed`, **notReady** or **ready** state.

10.2.1.1. 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:
    - lastTransitionTime: 2019-03-15T15:57:22Z
      message: Disk storage usage for node is 27.5gb (36.74%). Shards will be not
        be allocated on this node.
      reason: Disk Watermark Low
      status: "True"
      type: NodeStorage
    deploymentName: example-elasticsearch-cdm-0-1
    upgradeStatus: {}
```

This status message indicates a node has exceeded the configured high watermark and shards will be relocated to other nodes.

```
status:
  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: 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:
     - last Transition Time:  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
    reason: Invalid Settings
    status: "True"
    type: InvalidRedundancy
```

This status message indicates your cluster has too many control plane nodes (also known as the 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
```

10.2.2. 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                                          S4QANnf1QP6NgCegfnrnbQ
   3   1     119926          0        157            78
   green  open   audit-000001                                          8_EQx77iQCSTzFOXtxRqFw
   3   1          0          0          0             0
   green  open   .security                                         iDjscH7aSUGhIdq0LheLBQ   1
   1          5          0          0             0
   green  open   .kibana_-377444158_kubeadmin
   yBywZ9GfSrKebz5gWBZbjw   3   1          1          0          0             0
   ```

```
green  open   infra-000001                                     z6Dpe__ORgiopEpW6Yl44A
3   1     871000           0       874           436
green  open   app-000001                                       hIrazQCeSISewG3c2VIvsQ
3   1       2453           0         3             1
green  open   .kibana_1                                        JCitcBMSQxKOvIq6iQW6wg
1   1          0           0         0             0
green  open   .kibana_-1595131456_user1                            gIYFIEGRRe-
ka0W3okS-mQ   3   1          1           0         0             0
```

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://3f77032abaddbb1652c116278652908dc01860320b8a4e741d06894b2f8f9aa1
    State:          Running
      Started:      Mon, 08 Jun 2020 10:18:38 -0400
    Ready:          True
    Restart Count:  0
```

```
....

Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True

....

Events:          <none>
```

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@sha256:4265742c7cdd85359140e2d7d703e4311b6497eec7676957f455d6
908e7b1c25
    Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
period=5s #success=1 #failure=3

....

Events:          <none>
```

10.3. UNDERSTANDING CLUSTER LOGGING ALERTS

All of the logging collector alerts are listed on the Alerting UI of the OpenShift Container Platform web console.

10.3.1. 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 alerts.

10.3.2. 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 10.1. Fluentd Prometheus alerts

Alert	Message	Description	Severity
FluentDHighErrorRate	**<value> of records have resulted in an error by fluentd <instance>.**	The number of FluentD output errors is high, by default more than 10 in the previous 15 minutes.	Warning
FluentdNodeDown	**Prometheus could not scrape fluentd <instance> for more than 10m.**	Fluentd is reporting that Prometheus could not scrape a specific Fluentd instance.	Critical
FluentdQueueLengthBurst	**In the last minute, fluentd <instance> buffer queue length increased more than 32. Current value is <value>.**	Fluentd is reporting that it cannot keep up with the data being indexed.	Warning
FluentdQueueLengthIncreasing	**In the last 12h, fluentd <instance> buffer queue length constantly increased more than 1. Current value is <value>.**	Fluentd is reporting that the queue size is increasing.	Critical
FluentDVeryHighErrorRate	**<value> of records have resulted in an error by fluentd <instance>.**	The number of FluentD output errors is very high, by default more than 25 in the previous 15 minutes.	Critical

10.3.3. About Elasticsearch alerting rules

You can view these alerting rules in Prometheus.

Alert	Description	Severity
ElasticsearchClusterNotHealthy	The cluster health status has been RED for at least 2 minutes. The cluster does not accept writes, shards may be missing, or the master node hasn’t been elected yet.	critical
ElasticsearchClusterNotHealthy	The cluster health status has been YELLOW for at least 20 minutes. Some shard replicas are not allocated.	warning
ElasticsearchDiskSpaceRunningLow	The cluster is expected to be out of disk space within the next 6 hours.	Critical
ElasticsearchHighFileDescriptorUsage	The cluster is predicted to be out of file descriptors within the next hour.	warning
ElasticsearchJVMHeapUseHigh	The JVM Heap usage on the specified node is high.	Alert
ElasticsearchNodeDiskWatermarkReached	The specified node has hit the low watermark due to low free disk space. Shards can not be allocated to this node anymore. You should consider adding more disk space to the node.	info
ElasticsearchNodeDiskWatermarkReached	The specified node has hit the high watermark due to low free disk space. Some shards will be re-allocated to different nodes if possible. Make sure more disk space is added to the node or drop old indices allocated to this node.	warning
ElasticsearchNodeDiskWatermarkReached	The specified node has hit the flood watermark due to low free disk space. Every index that has a shard allocated on this node is enforced a read-only block. The index block must be manually released when the disk use falls below the high watermark.	critical
ElasticsearchJVMHeapUseHigh	The JVM Heap usage on the specified node is too high.	alert
ElasticsearchWriteRequestsRejectionJumps	Elasticsearch is experiencing an increase in write rejections on the specified node. This node might not be keeping up with the indexing speed.	Warning
AggregatedLoggingSystemCPUHigh	The CPU used by the system on the specified node is too high.	alert
ElasticsearchProcessCPUHigh	The CPU used by Elasticsearch on the specified node is too high.	alert

10.4. 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 4.6, and will be removed in a later release.

10.4.1. 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:

    ```
    $ 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:

    ```
    $ oc create job --from=cronjob/curator <job_name>
    ```

3. Use the following commands to control the cron job:

    - Suspend a cron job:

      ```
      $ oc patch cronjob curator -p '{"spec":{"suspend":true}}'
      ```

    - Resume a cron job:

      ```
      $ oc patch cronjob curator -p '{"spec":{"suspend":false}}'
      ```

    - Change a cron job schedule:

```
$ oc patch cronjob curator -p '{"spec":{"schedule":"0 0 * * *"}}' (1)
```

(1) The **schedule** option accepts schedules in cron format.

10.5. 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 **hack/logging-dump.sh** script. The script is no longer supported and does not collect data.

10.5.1. About the must-gather tool

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

For your cluster logging environment, **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 **openshift-logging** and **openshift-operators-redhat** namespaces, including health status for the log collector, the log store, the curator, and the log visualizer

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

10.5.2. Prerequisites

- Cluster logging and Elasticsearch must be installed.

10.5.3. Collecting cluster logging data

You can use the **oc adm must-gather** CLI command to collect information about your cluster logging environment.

Procedure

To collect cluster logging information with **must-gather**:

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

2. Run the **oc adm must-gather** command against the cluster logging image:

   ```
   $ 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 **must-gather** tool creates a new directory that starts with **must-gather.local** within the current directory. For example: **must-gather.local.4157245944708210408**.

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

   ```
   $ tar -cvaf must-gather.tar.gz must-gather.local.4157245944708210408
   ```

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

CHAPTER 11. UNINSTALLING CLUSTER LOGGING

You can remove cluster logging from your OpenShift Container Platform cluster.

11.1. UNINSTALLING CLUSTER LOGGING FROM OPENSHIFT CONTAINER PLATFORM

You can stop log aggregation by deleting the **ClusterLogging** custom resource (CR). After deleting the CR, there are other cluster logging components that remain, which you can optionally remove.

Deleting the **ClusterLogging** CR does not remove the persistent volume claims (PVCs). To preserve or delete the remaining PVCs, persistent volumes (PVs), and associated data, you must take further action.

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.
    b. On the **Custom Resource Definitions** page, click **ClusterLogging**.
    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 **ClusterLogForwarder** and select **Delete Custom Resource Definition**.
    c. Click the Options menu ⋮ next to **ClusterLogging** and select **Delete Custom Resource Definition**.
    d. Click the Options menu ⋮ next to **Elasticsearch** and select **Delete Custom Resource Definition**.
3. Optional: Remove the Cluster Logging Operator and OpenShift Elasticsearch Operator:
    a. Switch to the **Operators** → **Installed Operators** page.

b. Click the Options menu ⋮ next to the Cluster Logging Operator and select **Uninstall Operator**.

c. Click the Options menu ⋮ next to the OpenShift 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**.

5. To keep the PVCs for reuse with other pods, keep the labels or PVC names that you need to reclaim the PVCs.

6. Optional: If you do not want to keep the PVCs, you can delete them.

   WARNING

   Releasing or deleting PVCs can delete PVs and cause data loss.

   a. Switch to the **Storage** → **Persistent Volume Claims** page.

   b. Click the Options menu ⋮ next to each PVC and select **Delete Persistent Volume Claim**.

   c. If you want to recover storage space, you can delete the PVs.

Additional resources

- Reclaiming a persistent volume manually

CHAPTER 12. 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**

12.1. 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.

Parameter	Description
@timestamp	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.
geoip	This is geo-ip of the machine.
hostname	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.
ipaddr4	The IP address V4 of the source server, which can be an array.
ipaddr6	The IP address V6 of the source server, if available.

Parameter	Description
level	The logging level as provided by rsyslog (severitytext property), python's logging module. Possible values are as listed at **misc/sys/syslog.h** plus **trace** and **unknown**. For example, "alert crit debug emerg err info notice trace unknown warning". Note that **trace** is not in the **syslog.h** 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 **syslog/journal PRIORITY** can usually be mapped using the priority values as listed at misc/sys/syslog.h. Log levels and priorities from other logging systems should be mapped to the nearest match. See python logging for an example.
message	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.
pid	This is the process ID of the logging entity, if available.
service	The name of the service associated with the logging entity, if available. For example, the **syslog APP-NAME** property is mapped to the service field.
tags	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.
file	Optional path to the file containing the log entry local to the collector **TODO** analyzer for file paths.
offset	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).
namespace_name	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 **kubernetes.namespace_name**.
namespace_uuid	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 **kubernetes.namespace_uuid**. This will also cause the Kubernetes metadata lookup to be skipped for this log record.

collectd Fields
The following fields represent namespace metrics metadata.

Parameter	Description
collectd.interval	type: float The **collectd** interval.
collectd.plugin	type: string The **collectd** plug-in.
collectd.plugin_instance	type: string The **collectd** plugin_instance.
collectd.type_instance	type: string The **collectd type_instance**.
collectd.type	type: string The **collectd** type.
collectd.dstypes	type: string The **collectd** dstypes.

collectd.processes Fields
The following field corresponds to the **collectd** processes plug-in.

Parameter	Description
collectd.processes.ps_state	type: integer The **collectd ps_state** type of processes plug-in.

collectd.processes.ps_disk_ops Fields
The **collectd ps_disk_ops** type of processes plug-in.

Parameter	Description
collectd.processes.ps_disk_ops.read	type: float **TODO**
collectd.processes.ps_disk_ops.write	type: float **TODO**

Parameter	Description
collectd.processes.ps_vm	type: integer The **collectd ps_vm** type of processes plug-in.
collectd.processes.ps_rss	type: integer The **collectd ps_rss** type of processes plug-in.
collectd.processes.ps_data	type: integer The **collectd ps_data** type of processes plug-in.
collectd.processes.ps_code	type: integer The **collectd ps_code** type of processes plug-in.
collectd.processes.ps_stacksize	type: integer The **collectd ps_stacksize** type of processes plug-in.

collectd.processes.ps_cputime Fields
The **collectd ps_cputime** type of processes plug-in.

Parameter	Description
collectd.processes.ps_cputime.user	type: float **TODO**
collectd.processes.ps_cputime.syst	type: float **TODO**

collectd.processes.ps_count Fields
The **collectd ps_count** type of processes plug-in.

Parameter	Description
collectd.processes.ps_count.processes	type: integer **TODO**
collectd.processes.ps_count.threads	type: integer **TODO**

collectd.processes.ps_pagefaults Fields

The **collectd ps_pagefaults** type of processes plug-in.

Parameter	Description
collectd.processes.ps_pagefaults.majflt	type: float **TODO**
collectd.processes.ps_pagefaults.minflt	type: float **TODO**

collectd.processes.ps_disk_octets Fields
The **collectd ps_disk_octets** type of processes plug-in.

Parameter	Description
collectd.processes.ps_disk_octets.read	type: float **TODO**
collectd.processes.ps_disk_octets.write	type: float **TODO**
collectd.processes.fork_rate	type: float The **collectd fork_rate** type of processes plug-in.

collectd.disk Fields
Corresponds to **collectd** disk plug-in.

collectd.disk.disk_merged Fields
The **collectd disk_merged** type of disk plug-in.

Parameter	Description
collectd.disk.disk_merged.read	type: float **TODO**
collectd.disk.disk_merged.write	type: float **TODO**

collectd.disk.disk_octets Fields
The **collectd disk_octets** type of disk plug-in.

Parameter	Description
collectd.disk.disk_octets.read	type: float **TODO**
collectd.disk.disk_octets.write	type: float **TODO**

collectd.disk.disk_time Fields
The **collectd disk_time** type of disk plug-in.

Parameter	Description
collectd.disk.disk_time.read	type: float **TODO**
collectd.disk.disk_time.write	type: float **TODO**

collectd.disk.disk_ops Fields
The **collectd disk_ops** type of disk plug-in.

Parameter	Description
collectd.disk.disk_ops.read	type: float **TODO**
collectd.disk.disk_ops.write	type: float **TODO**
collectd.disk.pending_operations	type: integer The **collectd pending_operations** type of disk plug-in.

collectd.disk.disk_io_time Fields
The **collectd disk_io_time** type of disk plug-in.

Parameter	Description
collectd.disk.disk_io_time.io_time	type: float **TODO**

Parameter	Description
collectd.disk.disk_io_time.weighted_io_time	type: float **TODO**

collectd.interface Fields
Corresponds to the **collectd** interface plug-in.

collectd.interface.if_octets Fields
The **collectd if_octets** type of interface plug-in.

Parameter	Description
collectd.interface.if_octets.rx	type: float **TODO**
collectd.interface.if_octets.tx	type: float **TODO**

collectd.interface.if_packets Fields
The **collectd if_packets** type of interface plug-in.

Parameter	Description
collectd.interface.if_packets.rx	type: float **TODO**
collectd.interface.if_packets.tx	type: float **TODO**

collectd.interface.if_errors Fields
The **collectd if_errors** type of interface plug-in.

Parameter	Description
collectd.interface.if_errors.rx	type: float **TODO**
collectd.interface.if_errors.tx	type: float **TODO**

collectd.interface.if_dropped Fields
The **collectd if_dropped** type of interface plug-in.

Parameter	Description
collectd.interface.if_dropped.rx	type: float **TODO**
collectd.interface.if_dropped.tx	type: float **TODO**

collectd.virt Fields
Corresponds to **collectd** virt plug-in.

collectd.virt.if_octets Fields
The **collectd if_octets** type of virt plug-in.

Parameter	Description
collectd.virt.if_octets.rx	type: float **TODO**
collectd.virt.if_octets.tx	type: float **TODO**

collectd.virt.if_packets Fields
The **collectd if_packets** type of virt plug-in.

Parameter	Description
collectd.virt.if_packets.rx	type: float **TODO**
collectd.virt.if_packets.tx	type: float **TODO**

collectd.virt.if_errors Fields
The **collectd if_errors** type of virt plug-in.

Parameter	Description

Parameter	Description
collectd.virt.if_errors.rx	type: float **TODO**
collectd.virt.if_errors.tx	type: float **TODO**

collectd.virt.if_dropped Fields
The **collectd if_dropped** type of virt plug-in.

Parameter	Description
collectd.virt.if_dropped.rx	type: float **TODO**
collectd.virt.if_dropped.tx	type: float **TODO**

collectd.virt.disk_ops Fields
The **collectd disk_ops** type of virt plug-in.

Parameter	Description
collectd.virt.disk_ops.read	type: float **TODO**
collectd.virt.disk_ops.write	type: float **TODO**

collectd.virt.disk_octets Fields
The **collectd disk_octets** type of virt plug-in.

Parameter	Description
collectd.virt.disk_octets.read	type: float **TODO**
collectd.virt.disk_octets.write	type: float **TODO**

Parameter	Description
collectd.virt.memory	type: float The **collectd** memory type of virt plug-in.
collectd.virt.virt_vcpu	type: float The **collectd virt_vcpu** type of virt plug-in.
collectd.virt.virt_cpu_tota l	type: float The **collectd virt_cpu_total** type of virt plug-in.

collectd.CPU Fields
Corresponds to the **collectd** CPU plug-in.

Parameter	Description
collectd.CPU.percent	type: float The **collectd** type percent of plug-in CPU.

collectd.df Fields
Corresponds to the **collectd df** plug-in.

Parameter	Description
collectd.df.df_complex	type: float The **collectd** type **df_complex** of plug-in **df**.
collectd.df.percent_bytes	type: float The **collectd** type **percent_bytes** of plug-in **df**.

collectd.entropy Fields
Corresponds to the **collectd** entropy plug-in.

Parameter	Description
collectd.entropy.entropy	type: integer The **collectd** entropy type of entropy plug-in.

collectd.memory Fields
Corresponds to the **collectd** memory plug-in.

Parameter	Description
collectd.memory.memory	type: float The **collectd** memory type of memory plug-in.
collectd.memory.percent	type: float The **collectd** percent type of memory plug-in.

collectd.swap Fields
Corresponds to the **collectd** swap plug-in.

Parameter	Description
collectd.swap.swap	type: integer The **collectd** swap type of swap plug-in.
collectd.swap.swap_io	type: integer The **collectd swap_io** type of swap plug-in.

collectd.load Fields
Corresponds to the **collectd** load plug-in.

collectd.load.load Fields
The **collectd** load type of load plug-in

Parameter	Description
collectd.load.load.shortterm	type: float **TODO**
collectd.load.load.midterm	type: float **TODO**
collectd.load.load.longterm	type: float **TODO**

collectd.aggregation Fields
Corresponds to **collectd** aggregation plug-in.

Parameter	Description
collectd.aggregation.percent	type: float **TODO**

collectd.statsd Fields
Corresponds to **collectd statsd** plug-in.

Parameter	Description
collectd.statsd.host_cpu	type: integer The **collectd** CPU type of **statsd** plug-in.
collectd.statsd.host_elapsed_time	type: integer The **collectd elapsed_time** type of **statsd** plug-in.
collectd.statsd.host_memory	type: integer The **collectd** memory type of **statsd** plug-in.
collectd.statsd.host_nic_speed	type: integer The **collectd nic_speed** type of **statsd** plug-in.
collectd.statsd.host_nic_rx	type: integer The **collectd nic_rx** type of **statsd** plug-in.
collectd.statsd.host_nic_tx	type: integer The **collectd nic_tx** type of **statsd** plug-in.
collectd.statsd.host_nic_rx_dropped	type: integer The **collectd nic_rx_dropped** type of **statsd** plug-in.
collectd.statsd.host_nic_tx_dropped	type: integer The **collectd nic_tx_dropped** type of **statsd** plug-in.
collectd.statsd.host_nic_rx_errors	type: integer The **collectd nic_rx_errors** type of **statsd** plug-in.
collectd.statsd.host_nic_tx_errors	type: integer The **collectd nic_tx_errors** type of **statsd** plug-in.

Parameter	Description
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_vdsm	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.

Parameter	Description
collectd.statsd.vm_cpu_usage	type: integer The **collectd cpu_usage** type of **statsd** plug-in.
collectd.statsd.vm_disk_read_ops	type: integer The **collectd disk_read_ops** type of **statsd** plug-in.
collectd.statsd.vm_disk_write_ops	type: integer The **collectd disk_write_ops** type of **statsd** plug-in.
collectd.statsd.vm_disk_flush_latency	type: integer The **collectd disk_flush_latency** type of **statsd** plug-in.
collectd.statsd.vm_disk_apparent_size	type: integer The **collectd disk_apparent_size** type of **statsd** plug-in.
collectd.statsd.vm_disk_write_bytes	type: integer The **collectd disk_write_bytes** type of **statsd** plug-in.
collectd.statsd.vm_disk_write_rate	type: integer The **collectd disk_write_rate** type of **statsd** plug-in.
collectd.statsd.vm_disk_true_size	type: integer The **collectd disk_true_size** type of **statsd** plug-in.
collectd.statsd.vm_disk_read_rate	type: integer The **collectd disk_read_rate** type of **statsd** plug-in.
collectd.statsd.vm_disk_write_latency	type: integer The **collectd disk_write_latency** type of **statsd** plug-in.
collectd.statsd.vm_disk_read_latency	type: integer The **collectd disk_read_latency** type of **statsd** plug-in.
collectd.statsd.vm_disk_read_bytes	type: integer The **collectd disk_read_bytes** type of **statsd** plug-in.

Parameter	Description
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.

Parameter	Description
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.

12.2. 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.

Parameter	Description
systemd.k.KERNEL_DEVICE	**systemd.k.KERNEL_DEVICE** is the kernel device name.
systemd.k.KERNEL_SUBSYSTEM	**systemd.k.KERNEL_SUBSYSTEM** is the kernel subsystem name.
systemd.k.UDEV_DEVLINK	**systemd.k.UDEV_DEVLINK** includes additional symlink names that point to the node.
systemd.k.UDEV_DEVNODE	**systemd.k.UDEV_DEVNODE** is the node path of the device.
systemd.k.UDEV_SYSNAME	**systemd.k.UDEV_SYSNAME** is the kernel device name.

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.

Parameter	Description
systemd.t.AUDIT_LOGINUID	**systemd.t.AUDIT_LOGINUID** is the user ID for the journal entry process.
systemd.t.BOOT_ID	**systemd.t.BOOT_ID** is the kernel boot ID.
systemd.t.AUDIT_SESSION	**systemd.t.AUDIT_SESSION** is the session for the journal entry process.
systemd.t.CAP_EFFECTIVE	**systemd.t.CAP_EFFECTIVE** represents the capabilities of the journal entry process.
systemd.t.CMDLINE	**systemd.t.CMDLINE** is the command line of the journal entry process.
systemd.t.COMM	**systemd.t.COMM** is the name of the journal entry process.

Parameter	Description
systemd.t.EXE	**systemd.t.EXE** is the executable path of the journal entry process.
systemd.t.GID	**systemd.t.GID** is the group ID for the journal entry process.
systemd.t.HOSTNAME	**systemd.t.HOSTNAME** is the name of the host.
systemd.t.MACHINE_ID	**systemd.t.MACHINE_ID** is the machine ID of the host.
systemd.t.PID	**systemd.t.PID** is the process ID for the journal entry process.
systemd.t.SELINUX_CONTEXT	**systemd.t.SELINUX_CONTEXT** is the security context, or label, for the journal entry process.
systemd.t.SOURCE_REALTIME_TIMESTAMP	**systemd.t.SOURCE_REALTIME_TIMESTAMP** is the earliest and most reliable timestamp of the message. This is converted to RFC 3339 NS format.
systemd.t.SYSTEMD_CGROUP	**systemd.t.SYSTEMD_CGROUP** is the **systemd** control group path.
systemd.t.SYSTEMD_OWNER_UID	**systemd.t.SYSTEMD_OWNER_UID** is the owner ID of the session.
systemd.t.SYSTEMD_SESSION	**systemd.t.SYSTEMD_SESSION**, if applicable, is the**systemd** session ID.
systemd.t.SYSTEMD_SLICE	**systemd.t.SYSTEMD_SLICE** is the slice unit of the journal entry process.
systemd.t.SYSTEMD_UNIT	**systemd.t.SYSTEMD_UNIT** is the unit name for a session.
systemd.t.SYSTEMD_USER_UNIT	**systemd.t.SYSTEMD_USER_UNIT**, if applicable, is the user unit name for a session.
systemd.t.TRANSPORT	**systemd.t.TRANSPORT** is the method of entry by the journal service. This includes, **audit**, **driver**, **syslog**, **journal**, **stdout**, and **kernel**.
systemd.t.UID	**systemd.t.UID** is the user ID for the journal entry process.
systemd.t.SYSLOG_FACILITY	**systemd.t.SYSLOG_FACILITY** is the field containing the facility, formatted as a decimal string, for **syslog**.
systemd.t.SYSLOG_IDENTIFIER	**systemd.t.systemd.t.SYSLOG_IDENTIFIER** is the identifier for **syslog**.

Parameter	Description
systemd.t.SYSLOG_PID	**SYSLOG_PID** is the client process ID for**syslog**.

systemd.u Fields
systemd.u Fields are directly passed from clients and stored in the journal.

Parameter	Description
systemd.u.CODE_FILE	**systemd.u.CODE_FILE** is the code location containing the filename of the source.
systemd.u.CODE_FUNCTION	**systemd.u.CODE_FUNCTION** is the code location containing the function of the source.
systemd.u.CODE_LINE	**systemd.u.CODE_LINE** is the code location containing the line number of the source.
systemd.u.ERRNO	**systemd.u.ERRNO**, if present, is the low-level error number formatted in numeric value, as a decimal string.
systemd.u.MESSAGE_ID	**systemd.u.MESSAGE_ID** is the message identifier ID for recognizing message types.
systemd.u.RESULT	For private use only.
systemd.u.UNIT	For private use only.

12.3. 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.

Parameter	Description
kubernetes.pod_id	Kubernetes ID of the pod.
kubernetes.namespace_name	The name of the namespace in Kubernetes.
kubernetes.namespace_id	ID of the namespace in Kubernetes.

Parameter	Description
kubernetes.host	Kubernetes node name.
kubernetes.container_name	The name of the container in Kubernetes.
kubernetes.labels.deployment	The deployment associated with the Kubernetes object.
kubernetes.labels.deploymentconfig	The deploymentconfig associated with the Kubernetes object.
kubernetes.labels.component	The component associated with the Kubernetes object.
kubernetes.labels.provider	The provider associated with the Kubernetes object.

kubernetes.annotations Fields
Annotations associated with the OpenShift object are **kubernetes.annotations** fields.

12.4. 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.

Parameter	Description
pipeline_metadata.collector.hostname	FQDN of the collector. It might be different from the FQDN of the actual emitter of the logs.
pipeline_metadata.collector.name	Name of the collector.
pipeline_metadata.collector.version	Version of the collector.
pipeline_metadata.collector.ipaddr4	IP address v4 of the collector server, can be an array.
pipeline_metadata.collector.ipaddr6	IP address v6 of the collector server, can be an array.

Parameter	Description
pipeline_metadata.collector.inputname	How the log message was received by the collector whether it was TCP/UDP, or imjournal/imfile.
pipeline_metadata.collector.received_at	Time when the message was received by the collector.
pipeline_metadata.collector.original_raw_message	The original non-parsed log message, collected by the collector or as close to the source as possible.

pipeline_metadata.normalizer Fields
This section contains metadata specific to the normalizer.

Parameter	Description
pipeline_metadata.normalizer.hostname	FQDN of the normalizer.
pipeline_metadata.normalizer.name	Name of the normalizer.
pipeline_metadata.normalizer.version	Version of the normalizer.
pipeline_metadata.normalizer.ipaddr4	IP address v4 of the normalizer server, can be an array.
pipeline_metadata.normalizer.ipaddr6	IP address v6 of the normalizer server, can be an array.
pipeline_metadata.normalizer.inputname	how the log message was received by the normalizer whether it was TCP/UDP.
pipeline_metadata.normalizer.received_at	Time when the message was received by the normalizer.
pipeline_metadata.normalizer.original_raw_message	The original non-parsed log message as it is received by the normalizer.
pipeline_metadata.trace	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.

12.5. 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.

Parameter	Description
ovirt.entity	The type of the data source, hosts, VMS, and engine.
ovirt.host_id	The oVirt host UUID.

ovirt.engine Fields
Namespace for metadata related to the Manager. The FQDN of the Manager is **ovirt.engine.fqdn**

12.6. 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.

Parameter	Description
aushape.serial	Audit event serial number.
aushape.node	Name of the host where the audit event occurred.
aushape.error	The error aushape encountered while converting the event.
aushape.trimmed	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.
aushape.text	An array log record strings representing the original audit event.

aushape.data Fields
Parsed audit event data related to Aushape.

Parameter	Description
aushape.data.avc	type: nested
aushape.data.execve	type: string
aushape.data.netfilter_cfg	type: nested

Parameter	Description
aushape.data.obj_pid	type: nested
aushape.data.path	type: nested

12.7. 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.

Parameter	Description
tlog.ver	Message format version number.
tlog.user	Recorded user name.
tlog.term	Terminal type name.
tlog.session	Audit session ID of the recorded session.
tlog.id	ID of the message within the session.
tlog.pos	Message position in the session, milliseconds.
tlog.timing	Distribution of this message's events in time.
tlog.in_txt	Input text with invalid characters scrubbed.
tlog.in_bin	Scrubbed invalid input characters as bytes.
tlog.out_txt	Output text with invalid characters scrubbed.
tlog.out_bin	Scrubbed invalid output characters as bytes.