OpenShift Container Platform 4.4

Logging

Configuring cluster logging in OpenShift Container Platform
OpenShift Container Platform 4.4 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 AND OPENSHIFT CONTAINER PLATFORM  ........ 6
  1. CLUSTER LOGGING .......................... 6
  1.1. Cluster logging components .................. 6
  1.2. About the logstore .......................... 6
  1.3. About the logging collector .................. 7
  1.4. About logging visualization .................. 7
  1.5. About logging curation ........................ 8
  1.6. About event routing ......................... 8
  1.7. About the Cluster Logging Custom Resource .. 8

CHAPTER 2. ABOUT DEPLOYING CLUSTER LOGGING ........................................ 10
  2.1. ABOUT DEPLOYING AND CONFIGURING CLUSTER LOGGING .................. 10
    2.1.1. Configuring and Tuning Cluster Logging .................. 10
    2.1.2. Sample modified Cluster Logging Custom Resource .......... 12
  2.2. STORAGE CONSIDERATIONS FOR CLUSTER LOGGING AND OPENSHIFT CONTAINER PLATFORM 13
    2.3. ADDITIONAL RESOURCES ......................... 13

CHAPTER 3. DEPLOYING CLUSTER LOGGING .................................................. 14
  3.1. INSTALLING CLUSTER LOGGING USING THE WEB CONSOLE .................. 14
  3.2. INSTALLING CLUSTER LOGGING USING THE CLI .................. 18
    3.3. ADDITIONAL RESOURCES ......................... 25

CHAPTER 4. UPDATING CLUSTER LOGGING ............................................. 26
  4.1. UPDATING CLUSTER LOGGING .................. 26

CHAPTER 5. WORKING WITH EVENT ROUTER ........................................... 30
  5.1. DEPLOYING AND CONFIGURING THE EVENT ROUTER .................. 30

CHAPTER 6. VIEWING CLUSTER LOGS ..................................................... 34
  6.1. VIEWING CLUSTER LOGS ......................... 34
  6.2. VIEWING CLUSTER LOGS IN THE OPENSHIFT CONTAINER PLATFORM WEB CONSOLE ... 35

CHAPTER 7. VIEWING CLUSTER LOGS USING KIBANA .................................. 36
  7.1. LAUNCHING KIBANA ......................... 36

CHAPTER 8. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT ..................... 37
  8.1. ABOUT CONFIGURING CLUSTER LOGGING .................. 37
    8.1.1. About deploying and configuring cluster logging .................. 37
    8.1.1.1. Configuring and Tuning Cluster Logging .................. 37
    8.1.1.2. Sample modified Cluster Logging Custom Resource .......... 39
  8.2. CHANGING CLUSTER LOGGING MANAGEMENT STATE .................. 40
    8.2.1. Changing the cluster logging management state .................. 41
    8.2.2. Changing the Elasticsearch management state .................. 42
  8.3. CONFIGURING CLUSTER LOGGING .................. 42
    8.3.1. Understanding the cluster logging component images .................. 44
  8.4. CONFIGURING ELASTICSEARCH TO STORE AND ORGANIZE LOG DATA .......... 44
    8.4.1. Configuring Elasticsearch CPU and memory limits .................. 45
    8.4.2. Configuring Elasticsearch replication policy .................. 46
    8.4.3. Configuring Elasticsearch storage .................. 47
    8.4.4. Configuring Elasticsearch for emptyDir storage .................. 48
    8.4.5. Exposing Elasticsearch as a route .................. 48
    8.4.6. About Elasticsearch alerting rules .................. 51
  8.5. CONFIGURING KIBANA .................. 52
## 8.5. CONFIGURING THE LOGGING COLLECTOR

- 8.5.1. Configure Kibana CPU and memory limits
- 8.5.2. Scaling Kibana for redundancy
- 8.5.3. Using tolerations to control the Kibana Pod placement
- 8.5.4. Installing the Kibana Visualize tool

## 8.6. CURATION OF ELASTICSEARCH DATA

- 8.6.1. Configuring the Curator schedule
- 8.6.2. Configuring Curator index deletion
- 8.6.3. Troubleshooting Curator
- 8.6.4. Configuring Curator in scripted deployments
- 8.6.5. Using the Curator Action file

## 8.7. CONFIGURING THE LOGGING COLLECTOR

- 8.7.1. Viewing logging collector pods
- 8.7.2. Configure log collector CPU and memory limits
- 8.7.3. Configuring the collected log location
- 8.7.4. Understanding Buffer Chunk Limiting for Fluentd
- 8.7.5. Configuring the logging collector using environment variables
- 8.7.6. About logging collector alerts

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

- 8.8.1. Using tolerations to control the Elasticsearch Pod placement
- 8.8.2. Using tolerations to control the Kibana Pod placement
- 8.8.3. Using tolerations to control the Curator Pod placement
- 8.8.4. Using tolerations to control the log collector Pod placement
- 8.8.5. Additional resources

## 8.9. FORWARD LOGS TO THIRD PARTY SYSTEMS

- 8.9.1. Forwarding logs using the Fluentd forward protocol
- 8.9.2. Forwarding logs using the syslog protocol
- 8.9.3. Forwarding logs using the Log Forwarding API
  - 8.9.3.1. Understanding the Log Forwarding API
  - 8.9.3.2. Enabling the Log Forwarding API
  - 8.9.3.3. Configuring log forwarding using the Log Forwarding API
  - 8.9.3.3.1. Example log forwarding custom resources
  - 8.9.3.4. Disabling the Log Forwarding API

## 8.10. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD

- 8.10.1. Configuring systemd-journald for cluster logging

## CHAPTER 9. VIEWING ELASTICSEARCH STATUS

- 9.1. VIEWING ELASTICSEARCH STATUS
- 9.1.1. Example condition messages

## CHAPTER 10. VIEWING CLUSTER LOGGING STATUS

- 10.1. VIEWING THE STATUS OF THE CLUSTER LOGGING OPERATOR
- 10.1.1. Example condition messages

## CHAPTER 11. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS

- 11.1. MOVING THE CLUSTER LOGGING RESOURCES

## CHAPTER 12. MANUALLY ROLLING OUT ELASTICSEARCH

- 12.1. PERFORMING AN ELASTICSEARCH ROLLING CLUSTER RESTART

## CHAPTER 13. TROUBLESHOOTING KIBANA

- 13.1. TROUBLESHOOTING A KUBERNETES LOGIN LOOP
- 13.2. TROUBLESHOOTING A KUBERNETES CRYPTIC ERROR WHEN VIEWING THE KIBANA CONSOLE
13.3. TROUBLESHOOTING A KUBERNETES 503 ERROR WHEN VIEWING THE KIBANA CONSOLE

CHAPTER 14. EXPORTED FIELDS

14.1. DEFAULT EXPORTED FIELDS

- Top Level Fields
- collectd Fields
- collectd.processes Fields
- collectd.processes.ps_disk_ops Fields
- collectd.processes.ps_cputime Fields
- collectd.processes.ps_count Fields
- collectd.processes.ps_pagefaults Fields
- collectd.processes.ps_disk_octets Fields
- collectd.disk Fields
- collectd.disk.disk_merged Fields
- collectd.disk.disk_octets Fields
- collectd.disk.disk_time Fields
- collectd.disk.disk_ops Fields
- collectd.disk.disk_io_time Fields
- collectd.interface Fields
- collectd.interface.if_octets Fields
- collectd.interface.if_packets Fields
- collectd.interface.if_errors Fields
- collectd.interface.if_dropped Fields
- collectd.virt Fields
- collectd.virt.if_octets Fields
- collectd.virt.if_packets Fields
- collectd.virt.if_errors Fields
- collectd.virt.if_dropped Fields
- collectd.virt.disk_ops Fields
- collectd.virt.disk_octets Fields
- collectd.CPU Fields
- collectd.df Fields
- collectd.entropy Fields
- collectd.memory Fields
- collectd.swap Fields
- collectd.load Fields
- collectd.load.load Fields
- collectd.aggregation Fields
- collectd.statsd Fields
- collectd.postgresql Fields

14.2. SYSTEMD EXPORTED FIELDS

- systemd.k Fields
- systemd.t Fields
- systemd.u Fields

14.3. KUBERNETES EXPORTED FIELDS

- kubernetes.labels Fields
- kubernetes.annotations Fields

14.4. CONTAINER EXPORTED FIELDS

- pipeline_metadata.collector Fields
- pipeline_metadata.normalizer Fields

14.5. OVIRT EXPORTED FIELDS

- ovirt.engine Fields

14.6. AUSHAPE EXPORTED FIELDS
CHAPTER 1. UNDERSTANDING CLUSTER LOGGING AND OPENSHIFT CONTAINER PLATFORM

As a cluster administrator, you can deploy cluster logging to aggregate all the logs from your OpenShift Container Platform cluster, such as node system logs, application container logs, and so forth.

1.1. CLUSTER LOGGING

OpenShift Container Platform cluster administrators can deploy cluster logging using a few CLI commands and the OpenShift Container Platform web console to install the Elasticsearch Operator and Cluster Logging Operator. When the operators are installed, create a Cluster Logging 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.

You can configure cluster logging by modifying the Cluster Logging Custom Resource (CR), named instances. The CR defines a complete cluster logging deployment that includes all the components of the logging stack to collect, store and visualize logs. The Cluster Logging Operator watches the ClusterLogging Custom Resource and adjusts the logging deployment accordingly.

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

1.1.1. Cluster logging components

The cluster logging components are based upon Elasticsearch, Fluentd, and Kibana (EFK). The collector, Fluentd, is deployed to each node in the OpenShift Container Platform cluster. It collects all node and container logs and writes them to Elasticsearch (ES). Kibana is the centralized, web UI where users and administrators can create rich visualizations and dashboards with the aggregated data.

There are currently 5 different types of cluster logging components:

- **logStore** - This is where the logs will be stored. The current implementation is Elasticsearch.
- **collection** - This is the component that collects logs from the node, formats them, and stores them in the logStore. The current implementation is Fluentd.
- **visualization** - This is the UI component used to view logs, graphs, charts, and so forth. The current implementation is Kibana.
- **curation** - This is the component that trims logs by age. The current implementation is Curator.
- **event routing** - This is the component forwards OpenShift Container Platform events to cluster logging. The current implementation is Event Router.

In this document, we may refer to logStore or Elasticsearch, visualization or Kibana, curation or Curator, collection or Fluentd, interchangeably, except where noted.

1.1.2. About the logstore

OpenShift Container Platform uses Elasticsearch (ES) to organize the log data from Fluentd into datastores, or indices.

Elasticsearch 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. Elasticsearch also spreads these replicas across the Elasticsearch nodes. The
**ClusterLogging** Custom Resource allows you to specify the replication policy in the Custom Resource Definition (CRD) to provide data redundancy and resilience to failure.

**NOTE**

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

The Cluster Logging Operator and companion Elasticsearch Operator ensure that each Elasticsearch node is deployed using a unique Deployment that includes its own storage volume. You can use a Cluster Logging Custom Resource (CR) to increase the number of Elasticsearch nodes. Refer to Elastic’s documentation for considerations involved in choosing storage and network location as directed below.

**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. Access to the indexes with the `project.{project_name}.{project_uuid}`. format is restricted based on the permissions of the user in the specific project.

For more information, see Elasticsearch (ES).

**1.1.3. About the logging collector**

OpenShift Container Platform uses Fluentd to collect data about your cluster.

The logging collector is deployed as a DaemonSet in OpenShift Container Platform that deploys pods to each OpenShift Container Platform node. `journald` is the system log source supplying log messages from the operating system, the container runtime, and 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 more information, see Fluentd.

**1.1.4. About logging visualization**

OpenShift Container Platform uses Kibana to display the log data collected by Fluentd and indexed by Elasticsearch.
Kibana is a browser-based console interface to query, discover, and visualize your Elasticsearch data through histograms, line graphs, pie charts, heat maps, built-in geospatial support, and other visualizations.

For more information, see Kibana.

1.1.5. About logging curation

The Elasticsearch Curator tool performs scheduled maintenance operations on a global and/or on a per-project basis. Curator performs actions based on its configuration. Only one Curator Pod is recommended per Elasticsearch cluster.

```yaml
spec:
curation:
type: "curator"
resources:
curator:
schedule: "30 3 * * *" ▼
```

Specify the Curator schedule in the cron format.

For more information, see Curator.

1.1.6. About event routing

The Event Router is a pod that forwards OpenShift Container Platform events to cluster logging. You must manually deploy Event Router.

The Event Router collects events and converts them into JSON format, which takes those events and pushes them to STDOUT. Fluentd indexes the events to the .operations index.

1.1.7. About the Cluster Logging Custom Resource

To make changes to your cluster logging deployment, create and modify the Cluster Logging 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 Cluster Logging CR

```yaml
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          memory: 16Gi
```
requests:
  cpu: "1"
  memory: 16Gi
storage:
  storageClassName: "gp2"
  size: "200G"
  redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana"
kibana:
  resources:
    limits:
      memory: 736Mi
    requests:
      cpu: 100m
      memory: 736Mi
  replicas: 1
curation:
  type: "curator"
curator:
  resources:
    limits:
      memory: 256Mi
    requests:
      cpu: 100m
      memory: 256Mi
    schedule: "30 3 * * *"
collection:
  logs:
    type: "fluentd"
fluentd:
  resources:
    limits:
      memory: 736Mi
    requests:
      cpu: 100m
      memory: 736Mi
CHAPTER 2. ABOUT DEPLOYING CLUSTER LOGGING

Before installing cluster logging into your OpenShift Container Platform cluster, review the following sections.

2.1. ABOUT DEPLOYING AND CONFIGURING CLUSTER LOGGING

OpenShift Container Platform cluster logging is designed to be used with the default configuration, which is tuned for small to medium sized OpenShift Container Platform clusters.

The installation instructions that follow include a sample Cluster Logging Custom Resource (CR), which you can use to create a cluster logging instance and configure your cluster logging deployment.

If you want to use the default cluster logging install, you can use the sample CR directly.

If you want to customize your deployment, make changes to the sample CR as needed. The following describes the configurations you can make when installing your cluster logging instance or modify after installation. See the Configuring sections for more information on working with each component, including modifications you can make outside of the Cluster Logging Custom Resource.

2.1.1. Configuring and Tuning Cluster Logging

You can configure your cluster logging environment by modifying the Cluster Logging Custom Resource deployed in the openshift-logging project.

You can modify any of the following components upon install or after install:

Memory and CPU

You can adjust both the CPU and memory limits for each component by modifying the resources block with valid memory and CPU values:

```yaml
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu: 1
          memory: 16Gi
        requests:
          cpu: 1
          memory: 16Gi
    collection:
      logs:
        fluentd:
          resources:
            limits:
              cpu: 1
              memory: 16Gi
            requests:
              cpu: 1
              memory: 16Gi
```

Type: "fluentd"

Visualization:

Kibana:
Elasticsearch storage

You can configure a persistent storage class and size for the Elasticsearch cluster using the `storageClass name` and `size` parameters. The Cluster Logging Operator creates a `PersistentVolumeClaim` for each data node in the Elasticsearch cluster based on these parameters.

This example specifies each data node in the cluster will be bound to a `PersistentVolumeClaim` that requests "200G" of "gp2" storage. Each primary shard will be backed by a single replica.

NOTE

Omitting the `storage` block results in a deployment that includes ephemeral storage only.

Elasticsearch replication policy

You can set the policy that defines how Elasticsearch shards are replicated across data nodes in the cluster:

- **FullRedundancy.** The shards for each index are fully replicated to every data node.
- **MultipleRedundancy.** The shards for each index are spread over half of the data nodes.
- **SingleRedundancy.** A single copy of each shard. Logs are always available and recoverable as long as at least two data nodes exist.

- **ZeroRedundancy.** No copies of any shards. Logs may be unavailable (or lost) in the event a node is down or fails.

**Curator schedule**

You specify the schedule for Curator in the cron format.

```yaml
spec:
curation:
type: "curator"
resources:
curator:
schedule: "30 3 * * *"
```

2.1.2. Sample modified Cluster Logging Custom Resource

The following is an example of a Cluster Logging Custom Resource modified using the options previously described.

**Sample modified Cluster Logging Custom Resource**

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 2
      resources:
        limits:
          memory: 2Gi
        requests:
          cpu: 200m
          memory: 2Gi
      storage: {}
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 1Gi
      replicas: 1
  curation:
    type: "curator"
```
2.2. STORAGE CONSIDERATIONS FOR CLUSTER LOGGING AND OPENSHIFT CONTAINER PLATFORM

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

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

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

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

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

**NOTE**

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

2.3. ADDITIONAL RESOURCES

For more information on installing operators, see Installing Operators from the OperatorHub.
CHAPTER 3. DEPLOYING CLUSTER LOGGING

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

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

- Reviewing the installation options in About deploying cluster logging.
- Reviewing the cluster logging storage considerations.
- Installing the Elasticsearch Operator and Cluster Logging Operator using the {project-title} web console or CLI.

3.1. INSTALLING CLUSTER LOGGING USING THE WEB CONSOLE

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

Prerequisites

- Ensure that you have the necessary persistent storage for Elasticsearch. Note that each Elasticsearch node requires its own storage volume. Elasticsearch is a memory-intensive application. Each Elasticsearch node needs 16G of memory for both memory requests and 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 deployments.

Procedure

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

1. Install the Elasticsearch Operator:
   b. Choose Elasticsearch Operator from the list of available Operators, and click Install.
   c. Ensure that the All namespaces on the cluster is selected under Installation Mode.
   d. Ensure that openshift-operators-redhat is selected under Installed Namespace. You must specify the openshift-operators-redhat Namespace. 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.
   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 an **Update Channel** and **Approval Strategy**.

g. Click **Subscribe**.

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

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

2. Install the Cluster Logging Operator:

   a. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.

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

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

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

   e. Select **Enable operator recommended cluster monitoring on this namespace**

   This option sets the **openshift.io/cluster-monitoring**: "true" label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-operators-redhat** Namespace.

   f. Select an **Update Channel** and **Approval Strategy**.

   g. Click **Subscribe**.

   h. Verify that the Cluster Logging Operator installed by switch to the **Operators → Installed Operators** page.

   i. 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"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: gp2
        size: 200G
        redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      replicas: 1
  curation:
    type: "curator"
    curator:
      schedule: "30 3 * * *"
  collection:
    logs:
      type: "fluentd"
      fluentd: {}
Settings for configuring Curator. Using the CR, you can set the Curator schedule. For more information, see Configuring Curator.

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

NOTE

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

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

```
$ oc get deployment
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Replicas</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster-logging-operator</td>
<td>1/1</td>
<td>18h</td>
</tr>
<tr>
<td>elasticsearch-cdm-x6kdekl-1</td>
<td>0/1</td>
<td>6m54s</td>
</tr>
<tr>
<td>elasticsearch-cdm-x6kdekl-2</td>
<td>0/1</td>
<td>6m49s</td>
</tr>
<tr>
<td>elasticsearch-cdm-x6kdekl-3</td>
<td>0/1</td>
<td>6m44s</td>
</tr>
<tr>
<td>fluentd-2c7dg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fluentd-9z7kk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 Custom Resource and Elasticsearch Custom Resource, which you can edit to make changes to your cluster logging cluster.

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-wtpv
   - elasticsearch-cdm-b3nqzchd-3-588c65-clg7g
   - fluentd-2c7dg
   - fluentd-9z7kk
3.2. INSTALLING CLUSTER LOGGING USING THE CLI

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

Prerequisites

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

Elasticsearch is a memory-intensive application. Each Elasticsearch node needs 16G of memory for both memory requests and 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 deployments.

Procedure

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

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

   ```yaml
   apiVersion: v1
   kind: Namespace
   metadata:
     name: openshift-operators-redhat
     annotations:
       openshift.io/node-selector: ""
     labels:
       openshift.io/cluster-monitoring: "true"
   
   You must specify the openshift-operators-redhat Namespace. To prevent possible conflicts with metrics, you should configure the Prometheus Cluster Monitoring stack to scrape metrics from the openshift-operators-redhat Namespace and not the openshift-operators Namespace. The openshift-operators Namespace might contain Community Operators, which are untrusted and could publish a metric with the same name as an OpenShift Container Platform metric, which would cause conflicts.

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

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

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

   b. Create the Namespace:

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

   For example:

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

3. Install the Elasticsearch Operator by creating the following objects:
   a. Create an Operator Group object YAML file (for example, `eo-og.yaml`) for the Elasticsearch operator:

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

   You must specify the `openshift-operators-redhat` Namespace.

   b. Create an Operator Group object:

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

   For example:

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

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

```yaml
apiVersion: operators.coreos.com/v1alpha1
class: Subscription
metadata:
  name: "elasticsearch-operator"
  namespace: "openshift-operators-redhat"
spec:
  channel: "4.4"
  installPlanApproval: "Automatic"
  source: "redhat-operators"
  sourceNamespace: "openshift-marketplace"
  name: "elasticsearch-operator"
```

1. You must specify the **openshift-operators-redhat** Namespace.
2. Specify **4.4** 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 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
```

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>REPLACES</th>
<th>VERSION</th>
<th>NAME</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td></td>
<td>4.4.0-202004222248</td>
<td>elasticsearch-operator.4.4.0-202004222248</td>
<td>Succeeded</td>
</tr>
<tr>
<td>Elasticsearch Operator</td>
<td>kube-node-lease</td>
<td>4.4.0-202004222248</td>
<td>elasticsearch-operator.4.4.0-202004222248</td>
<td>Succeeded</td>
</tr>
<tr>
<td>Elasticsearch Operator</td>
<td>kube-public</td>
<td>4.4.0-202004222248</td>
<td>elasticsearch-operator.4.4.0-202004222248</td>
<td>Succeeded</td>
</tr>
<tr>
<td>Elasticsearch Operator</td>
<td>kube-system</td>
<td>4.4.0-202004222248</td>
<td>elasticsearch-operator.4.4.0-202004222248</td>
<td>Succeeded</td>
</tr>
<tr>
<td>Elasticsearch Operator</td>
<td>openshift-apiversion-operator</td>
<td>4.4.0-202004222248</td>
<td>elasticsearch-operator.4.4.0-202004222248</td>
<td>Succeeded</td>
</tr>
<tr>
<td>Elasticsearch Operator</td>
<td>openshift-api-manager</td>
<td>4.4.0-202004222248</td>
<td>elasticsearch-operator.4.4.0-202004222248</td>
<td>Succeeded</td>
</tr>
<tr>
<td>Elasticsearch Operator</td>
<td>openshift-api-guardian</td>
<td>4.4.0-202004222248</td>
<td>elasticsearch-operator.4.4.0-202004222248</td>
<td>Succeeded</td>
</tr>
</tbody>
</table>
There should be an Elasticsearch Operator in each Namespace. The version number might be different than shown.

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

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

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

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
spec:
  channel: "4.4"
  name: cluster-logging
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

You must specify the `openshift-logging` Namespace.

Specify **4.4** as the channel.

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

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

For example:

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

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

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

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

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>openshift-logging</td>
<td>clusterlogging.4.4.0-202004222248</td>
<td>Cluster Logging</td>
</tr>
<tr>
<td></td>
<td>4.4.0-202004222248</td>
<td>Succeeded</td>
</tr>
</tbody>
</table>

5. Create a Cluster Logging instance:

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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: gp2
        size: 200G
        redundancyPolicy: "SingleRedundancy"
    visualization:
      type: "kibana"
```

NOTE

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

The Cluster Logging management state. In most 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 the **Managed** state.

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

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

Specify that each Elasticsearch node in the cluster is bound to a Persistent Volume Claim.

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

Settings for configuring Curator. Using the CR, you can set the Curator schedule. For more information, see **Configuring Curator**.

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

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

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

```
$ oc get deployment
```

```
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-cd-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
```

6. Verify the install 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
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster-logging-operator-66f77ffccbc-ppzbg</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>7m</td>
</tr>
<tr>
<td>elasticsearch-cdm-ftuhduuw-1-fcc4b9566-q6bhp</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>2m40s</td>
</tr>
<tr>
<td>elasticsearch-cdm-ftuhduuw-2-7b4994dbfc-rd2gc</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>2m36s</td>
</tr>
<tr>
<td>elasticsearch-cdm-ftuhduuw-3-84b5ff7ff8-gqnm2</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>2m4s</td>
</tr>
<tr>
<td>fluentd-587vb</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m26s</td>
</tr>
<tr>
<td>fluentd-7mbp9</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m30s</td>
</tr>
<tr>
<td>fluentd-flm6j</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m33s</td>
</tr>
<tr>
<td>fluentd-gn4rn</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m26s</td>
</tr>
<tr>
<td>fluentd-nlgb6</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m30s</td>
</tr>
<tr>
<td>fluentd-snkt</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>2m28s</td>
</tr>
<tr>
<td>kibana-d6d5666c5-rppqm</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>2m39s</td>
</tr>
</tbody>
</table>
3.3. ADDITIONAL RESOURCES

For more information on installing Operators, see Installing Operators from the OperatorHub.
CHAPTER 4. UPDATING CLUSTER LOGGING

After updating the OpenShift Container Platform cluster from 4.3 to 4.4, you must then upgrade cluster logging from 4.3 to 4.4.

4.1. UPDATING CLUSTER LOGGING

After updating the OpenShift Container Platform cluster, you can update cluster logging from 4.3 to 4.4 by updating the subscription for the Elasticsearch Operator and the Cluster Logging Operator.

IMPORTANT

Changes introduced by the new log forward feature modified the support for `out_forward` starting with the OpenShift Container Platform 4.3 release. You create a ConfigMap to configure `out_forward`. Any updates to the `secure-forward.conf` section of the Fluentd ConfigMap are removed.

If you use the `out_forward` plug-in, before updating, you can copy your current `secure-forward.conf` section from the Fluentd ConfigMap and use the copied data when you create the `secure-forward` ConfigMap.

Prerequisites

- Update the cluster from 4.3 to 4.4.
- Make sure the cluster logging status is healthy:
  - All Pods are ready.
  - Elasticsearch cluster is healthy.
- Optionally, copy your current `secure-forward.conf` section from the Fluentd ConfigMap for use if you want to create the `secure-forward` ConfigMap. See the note above.

Procedure

1. Update the Elasticsearch Operator:
   a. From the web console, click Operators → Installed Operators.
   b. Select the `openshift-operators-redhat` project.
   c. Click the Elasticsearch Operator.
   d. Click Subscription → Channel.
   e. In the Change Subscription Update Channel window, select 4.4 and click Save.
   f. Wait for a few seconds, then click Operators → Installed Operators.
      The Elasticsearch Operator is shown as 4.4. For example:
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 4.4 and click Save.
   f. Wait for a few seconds, then click Operators → Installed Operators.
      The Cluster Logging Operator is shown as 4.4. For example:
      
      Cluster Logging
      4.4.0-201909201915 provided
      by Red Hat, Inc

3. Check the logging components:
   a. Ensure that the Elasticsearch Pods are using a 4.4 image:
      
      $ oc get pod -o yaml -n openshift-logging --selector component=elasticsearch | grep 'image:'

      image: registry.redhat.io/openshift4/ose-logging-elasticsearch5@sha256:4e081ff048c3a56113bc672af5dfb8d29ea2ddca1fd79a3332a4446a461944f5
      image: registry.redhat.io/openshift4/ose-oauth-proxy@sha256:5fe478210770b21c1eb26c1570bcbda40bc5a79011580ff5ebd4c701a5b04eb2
      image: registry.redhat.io/openshift4/ose-logging-elasticsearch5@sha256:4e081ff048c3a56113bc672af5dfb8d29ea2ddca1fd79a3332a4446a461944f5
      image: registry.redhat.io/openshift4/ose-oauth-proxy@sha256:5fe478210770b21c1eb26c1570bcbda40bc5a79011580ff5ebd4c701a5b04eb2
      image: registry.redhat.io/openshift4/ose-logging-elasticsearch5@sha256:4e081ff048c3a56113bc672af5dfb8d29ea2ddca1fd79a3332a4446a461944f5
      image: registry.redhat.io/openshift4/ose-oauth-proxy@sha256:5fe478210770b21c1eb26c1570bcbda40bc5a79011580ff5ebd4c701a5b04eb2
      image: registry.redhat.io/openshift4/ose-logging-elasticsearch5@sha256:4e081ff048c3a56113bc672af5dfb8d29ea2ddca1fd79a3332a4446a461944f5
      image: registry.redhat.io/openshift4/ose-oauth-proxy@sha256:5fe478210770b21c1eb26c1570bcbda40bc5a79011580ff5ebd4c701a5b04eb2
      image: registry.redhat.io/openshift4/ose-logging-elasticsearch5@sha256:4e081ff048c3a56113bc672af5dfb8d29ea2ddca1fd79a3332a4446a461944f5
      image: registry.redhat.io/openshift4/ose-oauth-proxy@sha256:5fe478210770b21c1eb26c1570bcbda40bc5a79011580ff5ebd4c701a5b04eb2
      image: registry.redhat.io/openshift4/ose-logging-elasticsearch5@sha256:4e081ff048c3a56113bc672af5dfb8d29ea2ddca1fd79a3332a4446a461944f5
      image: registry.redhat.io/openshift4/ose-oauth-proxy@sha256:5fe478210770b21c1eb26c1570bcbda40bc5a79011580ff5ebd4c701a5b04eb2
      image: registry.redhat.io/openshift4/ose-logging-elasticsearch5@sha256:4e081ff048c3a56113bc672af5dfb8d29ea2ddca1fd79a3332a4446a461944f5
b. Ensure that all Elasticsearch Pods are in the **Ready** status:

```
$ oc get pod -n openshift-logging --selector component=elasticsearch
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>elasticsearch-cdm-1pbrl44l-1-55b7546f4c-mshhk</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>31m</td>
</tr>
<tr>
<td>elasticsearch-cdm-1pbrl44l-2-5c6d87589f-gx5hk</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>30m</td>
</tr>
<tr>
<td>elasticsearch-cdm-1pbrl44l-3-88df5d47-m45jc</td>
<td>2/2</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
</tbody>
</table>

c. 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",
  ...
```

d. Ensure that the logging collector Pods are using a 4.4 image:

```
$ oc get pod -n openshift-logging --selector logging-infra=fluentd -o yaml |grep 'image:'
```

```
image: registry.redhat.io/openshift4/ose-logging-fluentd@sha256:20f92b37685bc1003bb490002f7d8a1abbe2dd2d157e8532afa3830ce8da3483
image: registry.redhat.io/openshift4/ose-logging-fluentd@sha256:20f92b37685bc1003bb490002f7d8a1abbe2dd2d157e8532afa3830ce8da3483
image: registry.redhat.io/openshift4/ose-logging-fluentd@sha256:20f92b37685bc1003bb490002f7d8a1abbe2dd2d157e8532afa3830ce8da3483
```
e. Ensure that the Kibana Pods are using a 4.4 image:

```
$ oc get pod -n openshift-logging --selector logging-infra=kibana -o yaml |grep 'image:'
```

```
image: registry.redhat.io/openshift4/ose-logging-kibana5@sha256:3d657e3b90f9b3ef04a8351b1923250f93c04529f56b3e6ada0aba7c0a038ffe
f56e
image: registry.redhat.io/openshift4/ose-oauth-proxy@sha256:5fe478210770b21c1eb26c1570bcbda40bc5a79011580f5ebd4c701a5b04eb2
image: registry.redhat.io/openshift4/ose-logging-kibana5@sha256:3d657e3b90f9b3ef04a8351b1923250f93c04529f56b3e6ada0aba7c0a038ffe
f56e
image: registry.redhat.io/openshift4/ose-oauth-proxy@sha256:5fe478210770b21c1eb26c1570bcbda40bc5a79011580f5ebd4c701a5b04eb2
```

f. Ensure that the Curator CronJob is using a 4.4 image:

```
$ oc get CronJob curator -n openshift-logging -o yaml |grep 'image:'
```

```
image: registry.redhat.io/openshift4/ose-logging-curator5@sha256:330c3499e790d0e184414125a4843cd48849c601eb9f19ff82f30794c858b0bc
```
CHAPTER 5. WORKING WITH EVENT ROUTER

The Event Router communicates with the OpenShift Container Platform and prints OpenShift Container Platform events to log of the pod where the event occurs.

If Cluster Logging is deployed, you can view the OpenShift Container Platform events in Kibana.

5.1. DEPLOYING AND CONFIGURING THE EVENT ROUTER

Use the following steps to deploy Event Router into your cluster.

The following Template object creates the Service Account, ClusterRole, and ClusterRoleBinding required for the Event Router.

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.

Procedure

1. Create a template for the Event Router:

```yaml
kind: Template
apiVersion: v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to cluster logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
  - kind: ServiceAccount
    apiVersion: v1
    metadata:
      name: eventrouter
      namespace: ${NAMESPACE}
  - kind: ClusterRole
    apiVersion: v1
    metadata:
      name: event-reader
      rules:
        - apiGroups: [""
          resources: ["events"
          verbs: ["get", "watch", "list"]
  - kind: ClusterRoleBinding
    apiVersion: v1
    metadata:
      name: event-reader-binding
    subjects:
      - kind: ServiceAccount
        name: eventrouter
        namespace: ${NAMESPACE}
    roleRef:
      kind: ClusterRole
```
name: event-reader
- kind: ConfigMap
apiVersion: v1
metadata:
  name: eventrouter
  namespace: ${NAMESPACE}
data:
  config.json: |
  
  
  "sink": "stdout"
  
- kind: Deployment
apiVersion: apps/v1
metadata:
  name: eventrouter
  namespace: ${NAMESPACE}
labels:
  component: eventrouter
  logging-infra: eventrouter
  provider: openshift
spec:
  selector:
    matchLabels:
      component: eventrouter
      logging-infra: eventrouter
      provider: openshift
  replicas: 1
  template:
    metadata:
      labels:
        component: eventrouter
        logging-infra: eventrouter
        provider: openshift
      name: eventrouter
    spec:
      serviceAccount: eventrouter
      containers:
        - name: kube-eventrouter
          image: ${IMAGE}
          imagePullPolicy: IfNotPresent
          resources:
            limits:
              memory: ${MEMORY}
            requests:
              cpu: ${CPU}
              memory: ${MEMORY}
          volumeMounts:
          - name: config-volume
            mountPath: /etc/eventrouter
          volumes:
          - name: config-volume
            configMap:
              name: eventrouter
parameters:
  - name: IMAGE
    displayName: Image
1. Creates a Service Account for the Event Router.
2. Creates a cluster role to monitor for events in the cluster.
3. Allows the `get`, `watch`, and `list` permissions for the `events` resource.
4. Creates a ClusterRoleBinding to bind the ClusterRole to the ServiceAccount.
5. Specify the image version for the Event Router.
6. Specify the memory limit for the Event Router pods. Defaults to '128Mi'.
7. Specify the minimum amount of CPU to allocate to the Event Router. Defaults to '100m'.
8. Specify the namespace where `eventrouter` is deployed. Defaults to `openshift-logging`. The value must be the same as specified for the ServiceAccount and ClusterRoleBinding. The project indicates where in Kibana you can locate events:
   - If the event router pod is deployed in a default project, such as `kube-*` and `openshift-*`, you can find the events under the `.operation` index.
   - If the event router pod is deployed in other projects, you can find the event under the index using the project namespace.

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

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

   For example:

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

   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:

   ```bash
   $ oc get pods --selector component=eventrouter -o name
   pod/logging-eventrouter-d649f97c8-qvv8r
   ```
$ oc logs logging-eventrouter-d649f97c8-qvv8r

{"verb":"ADDED","event":{"metadata":{"name":"elasticsearch-operator.v0.0.1.158f402e25397146","namespace":"openshift-operators","selfLink":"/api/v1/namespaces/openshift-operators/events/elasticsearch-operator.v0.0.1.158f402e25397146","uid":"37b7ff11-4f1a-11e9-a7ad-0271b2ca69f0","resourceVersion":"523264","creationTimestamp":"2019-03-25T16:22:43Z"},"involvedObject":{"kind":"ClusterServiceVersion","namespace":"openshift-operators","name":"elasticsearch-operator.v0.0.1","uid":"27b2ca6d-4f1a-11e9-8fba-0ea949ad6f16","apiVersion":"operators.coreos.com/v1alpha1","resourceVersion":"523096"},"reason":"InstallSucceeded","message":"waiting for install components to report healthy","source":{"component":"operator-lifecycle-manager"},"firstTimestamp":"2019-03-25T16:22:43Z","lastTimestamp":"2019-03-25T16:22:43Z","count":1,"type":"Normal"}
CHAPTER 6. VIEWING CLUSTER LOGS

You can view OpenShift Container Platform cluster logs in the CLI or OpenShift Container Platform web console.

6.1. VIEWING CLUSTER LOGS

You can view cluster logs in the CLI.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

To view cluster logs:

1. Determine if the log location is a file or CONSOLE (stdout).

   ```bash
   $ oc -n openshift-logging set env daemonset/fluentd --list | grep LOGGING_FILE_PATH
   ```

2. Depending on the log location, execute the logging command:

   - If `LOGGING_FILE_PATH` points to a file, the default, use the `logs` utility, from the project, where the pod is located, to print out the contents of Fluentd log files:

     ```bash
     $ oc exec <any-fluentd-pod> -- logs  
     ```

     Specify the name of a log collector pod. Note the space before `logs`.

     For example:

     ```bash
     $ oc exec fluentd-ht42r -n openshift-logging -- logs
     ```

   - If you are using `LOGGING_FILE_PATH=console`, the log collector writes logs to stdout/stderr. You can retrieve the logs with the `oc logs [-f] <pod_name>` command, where the `-f` is optional.

     ```bash
     $ oc logs -f <any-fluentd-pod> -n openshift-logging
     ```

     Specify the name of a log collector pod. Use the `-f` option to follow what is being written into the logs.

     For example

     ```bash
     $ oc logs -f fluentd-ht42r -n openshift-logging
     ```

     The contents of log files are printed out.

     By default, Fluentd reads logs from the tail, or end, of the log.
6.2. VIEWING CLUSTER LOGS IN THE OPENSHIFT CONTAINER
PLATFORM WEB CONSOLE

You can view cluster logs in the OpenShift Container Platform web console.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

To view cluster logs:

1. In the OpenShift Container Platform console, navigate to Workloads → Pods.
2. Select the openshift-logging project from the drop-down menu.
3. Click one of the logging collector pods with the fluentd prefix.
4. Click Logs.

By default, Fluentd reads logs from the tail, or end, of the log.
CHAPTER 7. VIEWING CLUSTER LOGS USING KIBANA

The cluster logging installation deploys the Kibana web console.

7.1. LAUNCHING KIBANA

Kibana is a browser-based console to query, discover, and visualize your logs through histograms, line graphs, pie charts, heat maps, built-in geospatial support, and other visualizations.

Procedure

To launch Kibana:

1. In the OpenShift Container Platform console, click **Monitoring → Logging**.

2. Log in using the same credentials you use to log in to the OpenShift Container Platform console. The Kibana interface launches. You can now:
   - Search and browse your data using the Discover page.
   - Chart and map your data using the Visualize page.
   - Create and view custom dashboards using the Dashboard page.

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.
CHAPTER 8. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT

8.1. ABOUT CONFIGURING CLUSTER LOGGING

After installing cluster logging into your OpenShift Container Platform cluster, you can make the following configurations.

NOTE

You must set cluster logging to Unmanaged state before performing these configurations, unless otherwise noted. For more information, see Changing the cluster logging management state.

Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades. For more information, see Support policy for unmanaged Operators.

8.1.1. About deploying and configuring cluster logging

OpenShift Container Platform cluster logging is designed to be used with the default configuration, which is tuned for small to medium sized OpenShift Container Platform clusters.

The installation instructions that follow include a sample Cluster Logging Custom Resource (CR), which you can use to create a cluster logging instance and configure your cluster logging deployment.

If you want to use the default cluster logging install, you can use the sample CR directly.

If you want to customize your deployment, make changes to the sample CR as needed. The following describes the configurations you can make when installing your cluster logging instance or modify after installation. See the Configuring sections for more information on working with each component, including modifications you can make outside of the Cluster Logging Custom Resource.

8.1.1.1. Configuring and Tuning Cluster Logging

You can configure your cluster logging environment by modifying the Cluster Logging Custom Resource deployed in the openshift-logging project.

You can modify any of the following components upon install or after install:

Memory and CPU

You can adjust both the CPU and memory limits for each component by modifying the resources block with valid memory and CPU values:

```
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu: 1
          memory:
          requests:
            cpu: 1
```
Elasticsearch storage

You can configure a persistent storage class and size for the Elasticsearch cluster using the `storageClass` name and size parameters. The Cluster Logging Operator creates a `PersistentVolumeClaim` for each data node in the Elasticsearch cluster based on these parameters.

```yaml
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: "gp2"
        size: "200G"
```

This example specifies each data node in the cluster will be bound to a `PersistentVolumeClaim` that requests "200G" of "gp2" storage. Each primary shard will be backed by a single replica.
NOTE

Omitting the `storage` block results in a deployment that includes ephemeral storage only.

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

**Elasticsearch replication policy**

You can set the policy that defines how Elasticsearch shards are replicated across data nodes in the cluster:

- **FullRedundancy.** The shards for each index are fully replicated to every data node.
- **MultipleRedundancy.** The shards for each index are spread over half of the data nodes.
- **SingleRedundancy.** A single copy of each shard. Logs are always available and recoverable as long as at least two data nodes exist.
- **ZeroRedundancy.** No copies of any shards. Logs may be unavailable (or lost) in the event a node is down or fails.

**Curator schedule**

You specify the schedule for Curator in the cron format.

```
spec:
  curation:
    type: "curator"
    resources:
      curator:
        schedule: "30 3 * * *"
```

**8.1.1.2. Sample modified Cluster Logging Custom Resource**

The following is an example of a Cluster Logging Custom Resource modified using the options previously described.

**Sample modified Cluster Logging Custom Resource**

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 2
```
8.2. CHANGING CLUSTER LOGGING MANAGEMENT STATE

In order to modify certain components managed by the Cluster Logging Operator or the Elasticsearch Operator, you must set the operator to the unmanaged state.

In unmanaged state, the operators do not respond to changes in the CRs. The administrator assumes full control of individual component configurations and upgrades when in unmanaged state.

**IMPORTANT**

Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades. For more information, see Support policy for unmanaged Operators.

In managed state, the Cluster Logging Operator (CLO) responds to changes in the Cluster Logging Custom Resource (CR) and adjusts the logging deployment accordingly.
The OpenShift Container Platform documentation indicates in a prerequisite step when you must set the OpenShift Container Platform cluster to Unmanaged.

**NOTE**

If you set the Elasticsearch Operator (EO) to unmanaged and leave the Cluster Logging Operator (CLO) as managed, the CLO will revert changes you make to the EO, as the EO is managed by the CLO.

### 8.2.1. Changing the cluster logging management state

You must set the operator to the unmanaged state in order to modify the components managed by the Cluster Logging Operator:

- the Curator CronJob,
- the Elasticsearch CR,
- the Kibana Deployment,
- the log collector DaemonSet.

If you make changes to these components in managed state, the Cluster Logging Operator reverts those changes.

**NOTE**

An unmanaged cluster logging environment does not receive updates until you return the Cluster Logging Operator to Managed state.

#### Prerequisites

- The Cluster Logging Operator must be installed.

#### Procedure

1. Edit the Cluster Logging Custom Resource (CR) in the openshift-logging project:

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

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   ....
   spec:
     managementState: "Managed" 1
   ```

   1 Specify the management state as **Managed** or **Unmanaged**.
8.2.2. Changing the Elasticsearch management state

You must set the operator to the unmanaged state in order to modify the Elasticsearch deployment files, which are managed by the Elasticsearch Operator.

If you make changes to these components in managed state, the Elasticsearch Operator reverts those changes.

NOTE
An unmanaged Elasticsearch cluster does not receive updates until you return the Elasticsearch Operator to Managed state.

Prerequisite

- The Elasticsearch Operator must be installed.
- Have the name of the Elasticsearch CR, in the openshift-logging project:

  ```bash
  $ oc get -n openshift-logging Elasticsearch
  NAME    AGE
  elasticsearch  28h
  ```

Procedure

Edit the Elasticsearch Custom Resource (CR) in the openshift-logging project:

```bash
$ oc edit Elasticsearch elasticsearch
```

```yaml
apiVersion: logging.openshift.io/v1
kind: Elasticsearch
metadata:
  name: elasticsearch
...

spec:
  managementState: "Managed"
```

1 Specify the management state as Managed or Unmanaged.

NOTE
If you set the Elasticsearch Operator (EO) to unmanaged and leave the Cluster Logging Operator (CLO) as managed, the CLO will revert changes you make to the EO, as the EO is managed by the CLO.

8.3. CONFIGURING CLUSTER LOGGING

Cluster logging is configurable using a Cluster Logging Custom Resource (CR) deployed in the openshift-logging project.
The Cluster Logging Operator watches for changes to Cluster Logging CRs, creates any missing logging components, and adjusts the logging deployment accordingly.

The Cluster Logging CR is based on the Cluster Logging Custom Resource Definition (CRD), which defines a complete cluster logging deployment and includes all the components of the logging stack to collect, store and visualize logs.

Sample Cluster Logging Custom Resource (CR)

```yaml
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  creationTimestamp: 2019-03-20T18:07:02Z
  generation: 1
  name: instance
  namespace: openshift-logging
spec:
  collection:
    logs:
      fluentd:
        resources: null
        type: fluentd
  curation:
    curator:
      resources: null
      schedule: 30 3 * * *
      type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: {}
          memory: {}
        requests:
          cpu: {}
          memory: {}
          storage: {}
      type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      proxy:
        resources: null
      replicas: 1
      resources: null
      type: kibana
```

You can configure the following for cluster logging:

- You can place cluster logging into an unmanaged state that allows an administrator to assume full control of individual component configurations and upgrades.

- You can overwrite the image for each cluster logging component by modifying the appropriate environment variable in the `cluster-logging-operator` Deployment.
8.3.1. Understanding the cluster logging component images

There are several components in cluster logging, each one implemented with one or more images. Each image is specified by an environment variable defined in the `cluster-logging-operator` deployment in the `openshift-logging` project and should not be changed.

You can view the images by running the following command:

```
$ oc -n openshift-logging set env deployment/cluster-logging-operator --list | grep _IMAGE
```

1. **ELASTICSEARCH_IMAGE**=registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3
2. **FLUENTD_IMAGE**=registry.redhat.io/openshift4/ose-logging-fluentd:v4.3
4. **CURATOR_IMAGE**=registry.redhat.io/openshift4/ose-logging-curator5:v4.3
5. **OAUTH_PROXY_IMAGE**=registry.redhat.io/openshift4/ose-oauth-proxy:v4.3

1. **ELASTICSEARCH_IMAGE** deploys Elasticsearch.
2. **FLUENTD_IMAGE** deploys Fluentd.
3. **KIBANA_IMAGE** deploys Kibana.
4. **CURATOR_IMAGE** deploys Curator.
5. **OAUTH_PROXY_IMAGE** defines OAUTH for OpenShift Container Platform.

The values might be different depending on your environment.

8.4. CONFIGURING ELASTICSEARCH TO STORE AND ORGANIZE LOG DATA

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

Some of the modifications you can make to your Elasticsearch deployment include:

- storage for your Elasticsearch cluster;
- how shards are replicated across data nodes in the cluster, from full replication to no replication;
- allowing external access to Elasticsearch data.

**NOTE**

Scaling down Elasticsearch nodes is not supported. When scaling down, Elasticsearch pods can be accidentally deleted, possibly resulting in shards not being allocated and replica shards being lost.

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 Cluster Logging 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 deployments.

**NOTE**

If you set the Elasticsearch Operator (EO) to unmanaged and leave the Cluster Logging Operator (CLO) as managed, the CLO will revert changes you make to the EO, as the EO is managed by the CLO.

### 8.4.1. Configuring Elasticsearch CPU and memory limits

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

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 Cluster Logging Custom Resource (CR) in the `openshift-logging` project:

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

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

1. Specify the CPU and memory limits as needed. If you leave these values blank, the Elasticsearch Operator sets default values that should be sufficient for most deployments.

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

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

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

### 8.4.2. Configuring Elasticsearch replication policy

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 Cluster Logging Custom Resource (CR) in the `openshift-logging` project:

   ```
ooc edit clusterlogging instance
   ```

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

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.

### 8.4.3. Configuring Elasticsearch storage

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 Cluster Logging CR to specify that each data node in the cluster is bound to a Persistent Volume Claim.

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

spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: "gp2"
        size: "200G"
```
This example specifies each data node in the cluster is bound to a Persistent Volume Claim that requests “200G” of AWS General Purpose SSD (gp2) storage.

### 8.4.4. Configuring Elasticsearch for emptyDir storage

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

**NOTE**

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

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

1. Edit the Cluster Logging CR to specify emptyDir:

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

### 8.4.5. Exposing Elasticsearch as a route

By default, Elasticsearch 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 Elasticsearch for those tools that access its data.

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

- The **Authorization: Bearer ${token}**
- The Elasticsearch reencrypt route and an [Elasticsearch API request](#).

Internally, you can access Elasticsearch using the Elasticsearch cluster IP:

You can get the Elasticsearch cluster IP using either of the following commands:

```
$ oc get service elasticsearch -o jsonpath={.spec.clusterIP} -n openshift-logging
172.30.183.229
```

```
oc get service elasticsearch
NAME        TYPE     CLUSTER-IP     EXTERNAL-IP    PORT(S)    AGE
elasticsearch  ClusterIP   172.30.183.229 <none>     9200/TCP   22h
```
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 Elasticsearch externally:

1. Change to the openshift-logging project:

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

2. Extract the CA certificate from Elasticsearch and write to the `admin-ca` file:

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

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

   a. Create a YAML file with the following:

   ```yaml
   apiVersion: route.openshift.io/v1
   kind: Route
   metadata:
     name: elasticsearch
     namespace: openshift-logging
   spec:
     host: 
     to:
       kind: Service
       name: elasticsearch
     tls:
       termination: reencrypt
       destinationCA: | 1
   ```

   1. Add the Elasticsearch CA certificate 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 Elasticsearch CA certificate to the route YAML you created:

   ```bash
   cat ./admin-ca | sed -e "s/\n/ /" >> <file-name>.yaml
   ```
c. Create the route:

```bash
$ oc create -f <file-name>.yaml
route.route.openshift.io/elasticsearch created
```

4. Check that the Elasticsearch service is exposed:

a. Get the token of this ServiceAccount to be used in the request:

```bash
$ token=$(oc whoami -t)
```

b. Set the `elasticsearch` route you created as an environment variable.

```bash
$ 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:

```bash
curl -tlsv1.2 --insecure -H "Authorization: Bearer ${token}"
  "https://${routeES}/.operations.*/_search?size=1" | jq
```

The response appears similar to the following:

```json
{
  "took": 441,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 89157,
    "max_score": 1,
    "hits": [
      {
        "_index": ".operations.2019.03.15",
        "_type": "com.example.viaq.common",
        "_id": "ODdiNWIyYzAtMjg5Ni0TAtNWE3MDY1MjMzNTc3",
        "_score": 1,
        "_source": {
          "_SOURCE_MONOTONIC_TIMESTAMP": "673396",
          "systemd": {
            "t": {
              "BOOT_ID": "246c34ee9cdeecb41a608e94",
              "MACHINE_ID": "e904a0bb5efd3e6badee0c",
              "TRANSPORT": "kernel"
            },
            "u": {
```
8.4.6. About Elasticsearch alerting rules

You can view these alerting rules in Prometheus.

<table>
<thead>
<tr>
<th>Alert</th>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElasticsearchClusterNotHealthy</td>
<td>Cluster health status has been RED for at least 2m. Cluster does not accept writes, shards may be missing or master node hasn’t been elected yet.</td>
<td>critical</td>
</tr>
<tr>
<td>ElasticsearchClusterNotHealthy</td>
<td>Cluster health status has been YELLOW for at least 20m. Some shard replicas are not allocated.</td>
<td>warning</td>
</tr>
<tr>
<td>ElasticsearchBulkRequestsRejectionJumps</td>
<td>High Bulk Rejection Ratio at node in cluster. This node may not be keeping up with the indexing speed.</td>
<td>warning</td>
</tr>
<tr>
<td>ElasticsearchNodeDiskWatermarkReached</td>
<td>Disk Low Watermark Reached at node in cluster. Shards can not be allocated to this node anymore. You should consider adding more disk space to the node.</td>
<td>alert</td>
</tr>
<tr>
<td>ElasticsearchNodeDiskWatermarkReached</td>
<td>Disk High Watermark Reached at node in cluster. 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.</td>
<td>high</td>
</tr>
<tr>
<td>Alert</td>
<td>Description</td>
<td>Severity</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>ElasticsearchJVMHeapUseHigh</td>
<td>JVM Heap usage on the node in cluster is &lt;value&gt;</td>
<td>alert</td>
</tr>
<tr>
<td>AggregatedLoggingSystemCPUHigh</td>
<td>System CPU usage on the node in cluster is &lt;value&gt;</td>
<td>alert</td>
</tr>
<tr>
<td>ElasticsearchProcessCPUHigh</td>
<td>ES process CPU usage on the node in cluster is &lt;value&gt;</td>
<td>alert</td>
</tr>
</tbody>
</table>

### 8.5. CONFIGURING KIBANA

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

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

**NOTE**

You must set cluster logging to Unmanaged state before performing these configurations, unless otherwise noted. For more information, see Changing the cluster logging management state.

Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades. For more information, see Support policy for unmanaged Operators.

### 8.5.1. Configure Kibana CPU and memory limits

Each component specification allows for adjustments to both the CPU and memory limits.

**Procedure**

1. Edit the Cluster Logging Custom Resource (CR) in the openshift-logging project:

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

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   ...
   spec:
     visualization:
       type: "kibana"
     kibana:
       replicas: 1
       resources:
         limits:
   ```
8.5.2. Scaling Kibana for redundancy

You can scale the Kibana deployment for redundancy.

Procedure

1. Edit the Cluster Logging Custom Resource (CR) in the `openshift-logging` project:

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

2. Specify the CPU and memory limits to allocate for each node.

3. Specify the CPU and memory limits to allocate to the Kibana proxy.

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

8.5.3. Using tolerations to control the Kibana Pod placement

You can control which nodes the Kibana Pods run on and prevent other workloads from using those nodes by using tolerations on the Pods.

You apply tolerations to the Kibana Pods through the Cluster Logging 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 Kibana 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 Cluster Logging Custom Resource (CR) to configure a toleration for the Kibana Pod:

   ```
   visualization:
   type: "kibana"
   kibana:
   tolerations:
   - key: "kibana"
     operator: "Exists"
     effect: "NoExecute"
     tolerationSeconds: 6000
   ```

   1. Specify the key that you added to the node.
   2. Specify the `Exists` operator to require the `key/value/effect` parameters to match.
   3. Specify the `NoExecute` effect.
   4. Optionally, specify the `tolerationSeconds` parameter to set how long a Pod can remain bound to a node before being evicted.

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

8.5.4. Installing the Kibana Visualize tool

Kibana’s `Visualize` tab enables you to create visualizations and dashboards for monitoring container logs, allowing administrator users (`cluster-admin` or `cluster-reader`) to view logs by deployment, namespace, pod, and container.

Procedure

To load dashboards and other Kibana UI objects:

1. If necessary, get the Kibana route, which is created by default upon installation of the Cluster Logging Operator:
2. Get the name of your Elasticsearch pods.

   $ oc get pods -l component=elasticsearch

   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

3. Create the necessary per-user configuration that this procedure requires:
   a. Log in to the Kibana dashboard as the user you want to add the dashboards to.

      https://kibana-openshift-logging.apps.openshift.com

      Where the URL is Kibana route.

   b. If the Authorize Access page appears, select all permissions and click Allow selected permissions.

   c. Log out of the Kibana dashboard.

4. Run the following command from the project where the pod is located using the name of any of your Elasticsearch pods:

   $ oc exec <es-pod> -- es_load_kibana_ui_objects <user-name>

   For example:

   $ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6k -- es_load_kibana_ui_objects <user-name>

**NOTE**

The metadata of the Kibana objects such as visualizations, dashboards, and so forth are stored in Elasticsearch with the .kibana.{user_hash} index format. You can obtain the user_hash using the userhash=$(echo -n $username | sha1sum | awk '{print $1}’) command. By default, the Kibana shared_ops index mode enables all users with cluster admin roles to share the index, and this Kibana object metadata is saved to the .kibana index.

Any custom dashboard can be imported for a particular user either by using the import/export feature or by inserting the metadata onto the Elasticsearch index using the curl command.
8.6. CURATION OF ELASTICSEARCH DATA

The Elasticsearch Curator tool performs scheduled maintenance operations on a global and/or on a per-project basis. Curator performs actions based on its configuration.

The Cluster Logging Operator installs Curator and its configuration. You can configure the Curator cron schedule using the Cluster Logging Custom Resource and further configuration options can be found in the Curator ConfigMap, curator in the openshift-logging project, which incorporates the Curator configuration file, curator5.yaml and an OpenShift Container Platform custom configuration file, config.yaml.

OpenShift Container Platform uses the config.yaml internally to generate the Curator action file.

Optionally, you can use the action file, directly. Editing this file allows you to use any action that Curator has available to it to be run periodically. However, this is only recommended for advanced users as modifying the file can be destructive to the cluster and can cause removal of required indices/settings from Elasticsearch. Most users only must modify the Curator configuration map and never edit the action file.

8.6.1. Configuring the Curator schedule

You can specify the schedule for Curator using the cluster logging Custom Resource created by the cluster logging installation.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

To configure the Curator schedule:

1. Edit the Cluster Logging Custom Resource in the openshift-logging project:

   ```yaml
   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.
   ```
8.6.2. Configuring Curator index deletion

You can configure Curator to delete Elasticsearch data based on retention settings. You can configure per-project and global settings. Global settings apply to any project not specified. Per-project settings override global settings.

Prerequisite

- 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 8.1. Project options**

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

   **Table 8.2. Filter options**

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

For example, to configure Curator to:

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

Use:

```yaml
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
```

OpenShift Container Platform 4.4 Logging
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**).

8.6.3. Troubleshooting Curator

You can use information in this section for debugging Curator. For example, if curator is in 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**

Enable the Curator debug log and trigger next Curator iteration manually

1. Enable debug log of Curator:

   ```bash
   $ oc set env cronjob/curator CURATOR_LOG_LEVEL=DEBUG
   CURATOR_SCRIPT_LOG_LEVEL=DEBUG
   ```

   Specify the log level:

   - **CRITICAL**. Curator displays only critical messages.
   - **ERROR**. Curator displays only error and critical messages.
   - **WARNING**. Curator displays only error, warning, and critical messages.
   - **INFO**. Curator displays only informational, error, warning, and critical messages.
   - **DEBUG**. Curator displays only debug messages, in addition to all of the above. The default value is **INFO**.

   **NOTE**

   Cluster logging uses the OpenShift Container Platform custom environment variable **CURATOR_SCRIPT_LOG_LEVEL** in OpenShift Container Platform wrapper scripts (**run.sh** and **convert.py**). The environment variable takes the same values as **CURATOR_LOG_LEVEL** for script debugging, as needed.

1. Trigger next curator iteration:

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

2. Use the following commands to control the CronJob:

   - Suspend a CronJob:
Resume a CronJob:

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

Change a CronJob schedule:

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

The **schedule** option accepts schedules in **cron format**.

### 8.6.4. Configuring Curator in scripted deployments

Use the information in this section if you must configure Curator in scripted deployments.

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.
- Set cluster logging to the unmanaged state.

**Procedure**

Use the following snippets to configure Curator in your scripts:

- For scripted deployments
  1. Create and modify the configuration:
     a. Copy the Curator configuration file and the OpenShift Container Platform custom configuration file from the Curator configuration map and create separate files for each:

     ```bash
     $ oc extract configmap/curator --keys=curator5.yaml,config.yaml --to=/my/config
     ```

     b. Edit the `/my/config/curator5.yaml` and `/my/config/config.yaml` files.

  2. Delete the existing Curator config map and add the edited YAML files to a new Curator config map.

     ```bash
     $ oc delete configmap curator ; sleep 1
     $ oc create configmap curator \
       --from-file=curator5.yaml=/my/config/curator5.yaml \
       --from-file=config.yaml=/my/config/config.yaml \
       ; sleep 1
     ```

     The next iteration will use this configuration.

- If you are using the **action** file:

  1. Create and modify the configuration:
a. Copy the Curator configuration file and the action file from the Curator configuration map and create separate files for each:

   $ oc extract configmap/curator --keys=curator5.yaml,actions.yaml --to=/my/config

b. Edit the /my/config/curator5.yaml and /my/config/actions.yaml files.

2. Delete the existing Curator config map and add the edited YAML files to a new Curator config map.

   $ oc delete configmap curator ; sleep 1
   $ oc create configmap curator
   --from-file=curator5.yaml=/my/config/curator5.yaml
   --from-file=actions.yaml=/my/config/actions.yaml
   ; sleep 1

   The next iteration will use this configuration.

8.6.5. Using the Curator Action file

The Curator ConfigMap in the openshift-logging project includes a Curator action file where you configure any Curator action to be run periodically.

However, when you use the action file, OpenShift Container Platform ignores the config.yaml section of the curator ConfigMap, which is configured to ensure important internal indices do not get deleted by mistake. In order to use the action file, you should add an exclude rule to your configuration to retain these indices. You also must manually add all the other patterns following the steps in this topic.

**IMPORTANT**

The actions and config.yaml are mutually-exclusive configuration files. Once the actions file exist, OpenShift Container Platform ignores the config.yaml file. Using the action file is recommended only for advanced users as using this file can be destructive to the cluster and can cause removal of required indices/settings from Elasticsearch.

**Prerequisite**

- Cluster logging and Elasticsearch must be installed.
- Set cluster logging to the unmanaged state. Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades.

**Procedure**

To configure Curator to delete indices:

1. Edit the Curator ConfigMap:

   oc edit cm/curator -n openshift-logging

2. Make the following changes to the action file:

   actions:
   1:
Specify `delete_indices` to delete the specified index.

Use the `filers` parameters to specify the index to be deleted. See the Elastic Search curator documentation for information on these parameters.

Specify `false` to allow the index to be deleted.

### 8.7. 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 Namespace metadata.

You can configure log rotation, log location, use an external log aggregator, and make other configurations for the log collector.
NOTE

You must set cluster logging to Unmanaged state before performing these configurations, unless otherwise noted. For more information, see Changing the cluster logging management state.

Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades. For more information, see Support policy for unmanaged Operators.

8.7.1. 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:

```bash
$ oc get pods --all-namespaces -o wide | grep fluentd
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
<th>IP</th>
<th>NODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>fluentd-5mr28</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>4m56s</td>
<td>10.129.2.12</td>
<td>ip-10-0-164-233.ec2.internal</td>
</tr>
<tr>
<td>fluentd-cnc4c</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>4m56s</td>
<td>10.128.2.13</td>
<td>ip-10-0-155-142.ec2.internal</td>
</tr>
<tr>
<td>fluentd-nlp8z</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>4m56s</td>
<td>10.131.0.13</td>
<td>ip-10-0-138-77.ec2.internal</td>
</tr>
<tr>
<td>fluentd-rknlk</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>4m56s</td>
<td>10.128.0.33</td>
<td>ip-10-0-128-130.ec2.internal</td>
</tr>
<tr>
<td>fluentd-rsm49</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>4m56s</td>
<td>10.129.0.37</td>
<td>ip-10-0-163-191.ec2.internal</td>
</tr>
<tr>
<td>fluentd-wjt8s</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>4m56s</td>
<td>10.130.0.42</td>
<td>ip-10-0-156-251.ec2.internal</td>
</tr>
</tbody>
</table>

8.7.2. Configure log collector CPU and memory limits

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

Procedure

1. Edit the Cluster Logging Custom Resource (CR) in the openshift-logging project:

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

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     name: "instance"
   ...
   ```
8.7.3. Configuring the collected log location

The log collector writes logs to a specified file or to the default location, `/var/log/fluentd/fluentd.log` based on the `LOGGING_FILE_PATH` environment variable.

Prerequisite

- Set cluster logging to the unmanaged state. Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades.

Procedure

To set the output location for the Fluentd logs:

1. Edit the `LOGGING_FILE_PATH` parameter in the `fluentd` daemonset. You can specify a particular file or `console`:

   ```yaml
   spec:
     template:
       spec:
         containers:
           env:
             - name: LOGGING_FILE_PATH
               value: console
   ```

   Specify the log output method:

   - use `console` to use the Fluentd default location. Retrieve the logs with the `oc logs [-f] <pod_name>` command.
   - use `<path-to-log/fluentd.log>` to send the log output to the specified file. Retrieve the logs with the `oc exec <pod_name> — logs` command. This is the default setting.
   
   Or, use the CLI:

   ```bash
   $ oc -n openshift-logging set env daemonset/fluentd
   LOGGING_FILE_PATH=/logs/fluentd.log
   ```

Specify the CPU and memory limits and requests as needed. The values shown are the default values.
8.7.4. Understanding Buffer Chunk Limiting for Fluentd

If the Fluentd logger is unable to keep up with a high number of logs, it will need to switch to file buffering to reduce memory usage and prevent data loss.

Fluentd file buffering stores records in **chunks**. Chunks are stored in **buffers**.

**NOTE**

To modify the FILE_BUFFER_LIMIT or BUFFER_SIZE_LIMIT parameters in the Fluentd daemonset as described below, you must set cluster logging to the unmanaged state. Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades.

The Fluentd buffer_chunk_limit is determined by the environment variable BUFFER_SIZE_LIMIT, which has the default value **8m**. The file buffer size per output is determined by the environment variable FILE_BUFFER_LIMIT, which has the default value **256Mi**. The permanent volume size must be larger than FILE_BUFFER_LIMIT multiplied by the output.

On the Fluentd pods, permanent volume /var/lib/fluentd should be prepared by the PVC or hostmount, for example. That area is then used for the file buffers.

The buffer_type and buffer_path are configured in the Fluentd configuration files as follows:

```
$ egrep "buffer_type|buffer_path" *.conf
output-es-config.conf:
  buffer_type file
  buffer_path `/var/lib/fluentd/buffer-output-es-config`
output-es-ops-config.conf:
  buffer_type file
  buffer_path `/var/lib/fluentd/buffer-output-es-ops-config`
```

The Fluentd buffer_queue_limit is the value of the variable BUFFER_QUEUE_LIMIT. This value is **32** by default.

The environment variable BUFFER_QUEUE_LIMIT is calculated as (FILE_BUFFER_LIMIT / (number_of_outputs * BUFFER_SIZE_LIMIT)).

If the BUFFER_QUEUE_LIMIT variable has the default set of values:

- **FILE_BUFFER_LIMIT** = **256Mi**
- **number_of_outputs** = **1**
- **BUFFER_SIZE_LIMIT** = **8Mi**

The value of buffer_queue_limit will be **32**. To change the buffer_queue_limit, you must change the value of FILE_BUFFER_LIMIT.

In this formula, number_of_outputs is **1** if all the logs are sent to a single resource, and it is incremented by **1** for each additional resource. For example, the value of number_of_outputs is:

- **1** - if all logs are sent to a single Elasticsearch pod
- **2** - if application logs are sent to an Elasticsearch pod and ops logs are sent to another Elasticsearch pod
- if application logs are sent to an Elasticsearch pod, ops logs are sent to another Elasticsearch pod, and both of them are forwarded to other Fluentd instances

8.7.5. Configuring the logging collector using environment variables

You can use environment variables to modify the configuration of the Fluentd log collector.

See the Fluentd README in Github for lists of the available environment variables.

Prerequisite

- Set cluster logging to the unmanaged state. Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades.

Procedure

Set any of the Fluentd environment variables as needed:

```
oc set env ds/fluentd <env-var>=<value>
```

For example:

```
oc set env ds/fluentd LOGGING_FILE_AGE=30
```

8.7.6. About logging collector alerts

The following alerts are generated by the logging collector and can be viewed on the Alerts tab of the Prometheus UI.

All the logging collector alerts are listed on the Monitoring → Alerts page of the OpenShift Container Platform web console. 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.

<table>
<thead>
<tr>
<th>Alert</th>
<th>Message</th>
<th>Description</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FluentdErrorsHigh</td>
<td>In the last minute, &lt;value&gt; errors reported by fluentd &lt;instance&gt;.</td>
<td>Fluentd is reporting a higher number of issues than the specified number, default 10.</td>
<td>Critical</td>
</tr>
<tr>
<td>FluentdNodeDown</td>
<td>Prometheus could not scrape fluentd &lt;instance&gt; for more than 10m.</td>
<td>Fluentd is reporting that Prometheus could not scrape a specific Fluentd instance.</td>
<td>Critical</td>
</tr>
<tr>
<td>Alert</td>
<td>Message</td>
<td>Description</td>
<td>Severity</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>FluentdQueueLength Burst</td>
<td>In the last minute, fluentd &lt;instance&gt; buffer queue length increased more than 32. Current value is &lt;value&gt;.</td>
<td>Fluentd is reporting that it is overwhelmed.</td>
<td>Warning</td>
</tr>
<tr>
<td>FluentdQueueLength Increasing</td>
<td>In the last 12h, fluentd &lt;instance&gt; buffer queue length constantly increased more than 1. Current value is &lt;value&gt;.</td>
<td>Fluentd is reporting queue usage issues.</td>
<td>Critical</td>
</tr>
</tbody>
</table>

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

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

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

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

**Sample cluster logging CR with tolerations**

```yaml
cr.yaml
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
curation:
type: "curator"
curator:
tolerations: 3
  - key: "logging"
    operator: "Exists"
    effect: "NoExecute"
    tolerationSeconds: 6000
resources:
  limits:
    memory: 200Mi
  requests:
    cpu: 100m
    memory: 100Mi
schedule: "*/5 * * * *"
collection:
logs:
type: "fluentd"
fluentd:
tolerations: 4
  - 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 Curator pod.
4  This toleration is added to the logging collector pods.
8.8.1. Using tolerations to control the Elasticsearch Pod placement

You can control which nodes the Elasticsearch Pods runs on and prevent other workloads from using those nodes by using tolerations on the Pods.

You apply tolerations to Elasticsearch Pods through the Cluster Logging 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 Elasticsearch Pods can run on that node.

By default, the Elasticsearch Pods have the following toleration:

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

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

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

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

   For example:

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

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

2. Edit the `logstore` section of the Cluster Logging Custom Resource (CR) to configure a toleration for the Elasticsearch Pods:

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

   - **Specify the key that you added to the node.**
   - **Specify the **Exists** operator to require a taint with the key `elasticsearch` to be present on the Node.**
Specify the NoExecute effect.

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

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

### 8.8.2. Using tolerations to control the Kibana Pod placement

You can control which nodes the Kibana Pods run on and prevent other workloads from using those nodes by using tolerations on the Pods.

You apply tolerations to the Kibana Pods through the Cluster Logging 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 Kibana Pod:

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

   ```bash
   $ oc adm taint nodes node1 kibana=node:NoExecute
   
   This example places a taint on node1 that has key kibana, value node, and taint effect NoExecute. You must use the NoExecute taint effect. NoExecute schedules only Pods that match the taint and remove existing Pods that do not match.
   ```

2. Edit the visualization section of the Cluster Logging Custom Resource (CR) to configure a toleration for the Kibana Pod:

   ```yaml
   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.
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`.

### 8.8.3. Using tolerations to control the Curator Pod placement

You can control which node the Curator Pod runs on and prevent other workloads from using those nodes by using tolerations on the Pod.

You apply tolerations to the Curator Pod through the Cluster Logging 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 Curator 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 Curator Pod:

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

   For example:

   ```sh
   $ oc adm taint nodes node1 curator=node:NoExecute
   ```

   This example places a taint on `node1` that has key `curator`, 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 `curation` section of the Cluster Logging Custom Resource (CR) to configure a toleration for the Curator Pod:

   ```yaml
   curation:
     type: "curator"
     curator:
       tolerations:
       - key: "curator"
         operator: "Exists"
         effect: "NoExecute"
         tolerationSeconds: 6000
   ```

   - **1** Specify the key that you added to the node.
   - **2** Specify the `Exists` operator to require the `key/value/effect` parameters to match.
   - **3** Specify the `NoExecute` effect.
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 that is created by the `oc adm taint` command. A Pod with this toleration would be able to schedule onto `node1`.

8.8.4. 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 Cluster Logging 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` section of the Cluster Logging Custom Resource (CR) to configure a toleration for the logging collector Pods:

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

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

### 8.8.5. Additional resources

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

### 8.9. FORWARD LOGS TO THIRD PARTY SYSTEMS

By default, OpenShift Container Platform cluster logging sends logs to the default internal Elasticsearch logstore, defined in the Cluster Logging Custom Resource.

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

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

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

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

**IMPORTANT**

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

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

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

### 8.9.1. Forwarding logs using the Fluentd forward protocol

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

**NOTE**

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

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

**IMPORTANT**

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

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

Sample **secure-forward.conf**

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

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

**Procedure**

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

1. Create a configuration file named `secure-forward.conf` for the `forward` parameters:
   a. Configure the secrets and TLS information:
      ```yaml
      <store>
        @type forward

        self_hostname ${hostname} \[1
        shared_key <SECRET_STRING> \[2
      </store>
      ```
transport tls 3

tls_verify_hostname true 4
tls_cert_path <path_to_file> 5

1. Specify the default value of the auto-generated certificate common name (CN).
2. Enter the Shared key between nodes
3. Specify tls to enable TLS validation.
4. Set to true to verify the server cert hostname. Set to false to ignore server cert hostname.
5. Specify the path to private CA certificate file as /etc/ocp-forward/ca_cert.pem.

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

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

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

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

For example:

```xml
<server>
  name externalsever1
  host 192.168.1.1
  hostlabel externalsever1.example.com
  port 24224
</server>
<server>
  name externalsever2
  host externalsever2.example.com
</server>
```
2. Create a ConfigMap named `secure-forward` in the `openshift-logging` namespace from the configuration file:

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

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

   ```bash
   $ oc create secret generic secure-forward --from-file=<arbitrary-name-of-key1>=cert_file_from_fluentd_receiver --from-literal=shared_key=value_from_fluentd_receiver
   $ oc create secret generic secure-forward --from-file=ca-bundle.crt=ca-for-fluentd-receiver/ca.crt --from-literal=shared_key=fluentd-receiver
   ```

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

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

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

8.9.2. Forwarding logs using the syslog protocol

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

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

   **NOTE**

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

   There are two versions of the `syslog` protocol:

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

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

**IMPORTANT**

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

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

**Sample syslog.conf**

```xml
<store>
  @type syslog_buffered
  remote_syslog rsyslogserver.openshift-logging.svc.cluster.local
  port 514
  hostname fluentd-4nzfz
  remove_tag_prefix tag
  tag_key ident,systemd.u.SYSLOG_IDENTIFIER
  facility local0
  severity info
  use_record true
  payload_key message
</store>
```

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

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

```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: syslog
```
namespace: openshift-logging
data:
syslog.conf: |
  <store>
    @type syslog_buffered
    remote_syslog syslogserver.openshift-logging.svc.cluster.local
    port 514
    hostname fluentd-4nzfz
    remove_tag_prefix tag
    tag_key ident,systemd.u.SYSLOG_IDENTIFIER
    facility local0
    severity info
    use_record true
    payload_key message
  </store>

Procedure

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

1. Create a configuration file named `syslog.conf` that contains the following parameters within the `<store>` stanza:
   a. Specify the `syslog` protocol type:
      ```
      @type syslog_buffered
      ```
      Specify the protocol to use, either: `syslog` or `syslog_buffered`.
   b. Configure the name, host, and port for your external syslog server:
      ```
      remote_syslog <remote>  
      port <number>  
      hostname <name>  
      ```
      Specify the FQDN or IP address of the syslog server.
      Specify the port of the syslog server.
      Specify a name for this syslog server.
      For example:
      ```
      remote_syslog syslogserver.openshift-logging.svc.cluster.local
      port 514
      hostname fluentd-server
      ```
   c. Configure the other syslog variables as needed:
      ```
      remove_tag_prefix
      tag_key <key>
      facility <value>
      ```
severity <value> 4
use_record <value> 5
payload_key message 6

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

For example:

    facility local0
    severity info

The configuration file appears similar to the following:

    <store>
    @type syslog_buffered
    remote_syslog syslogserver.openshift-logging.svc.cluster.local
    port 514
    hostname fluentd-4nzfz
    tag_key ident,systemd.u.SYSLOG_IDENTIFIER
    facility local0
    severity info
    use_record false
    </store>

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

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

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

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

    // Module included in the following assemblies:
    //
    // * logging/cluster-logging-external.adoc

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

**IMPORTANT**

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

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

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

**NOTE**

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

### 8.9.3.1. Understanding the Log Forwarding API

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

**NOTE**

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

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

An output can be either:

- **elasticsearch** to forward logs to an external Elasticsearch v5.x cluster, specified by server name or FQDN, and/or the internal OpenShift Container Platform Elasticsearch logstore.
- **forward** to forward logs to an external log aggregation solution. This option uses the Fluentd forward protocols.

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

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

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

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

Note the following:

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

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

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

The following example creates three outputs:

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

Three pipelines send:

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

**Sample log forwarding outputs and pipelines**

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

1. The name of the log forwarding CR must be **instance**.
2. Parameter to enable log forwarding. Set to **true** to enable log forwarding.
3. Configuration for the outputs.
4. A name to describe the output.
5. The type of output, either **elasticsearch** or **forward**.
6. The log forwarding endpoint, either the server name or FQDN. For the internal OpenShift Container Platform Elasticsearch logstore, specify `elasticsearch.openshift-logging.svc:9200`.
7. Optional name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project.
8. Optional setting if the endpoint does not use a secret, resulting in insecure communication.
9. Configuration for the pipelines.
10. A name to describe the pipeline.
11. The source type, **logs.app**, **logs.infra**, or **logs.audit**.
12. The name of one or more outputs configured in the CR.

### 8.9.3.2. Enabling the Log Forwarding API

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

To enable the Log Forwarding API:

1. Edit the Cluster Logging Custom Resource (CR) in the openshift-logging project:

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

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

   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   metadata:
     annotations:
       clusterlogging.openshift.io/logforwardingtechpreview: enabled
     name: "instance"
     namespace: "openshift-logging"
   spec:
     ...
     collection:
       logs:
         type: "fluentd"
         fluentd: {}  
   ```

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

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

8.9.3.3. Configuring log forwarding using the Log Forwarding API

To configure the Log Forwarding, edit the Cluster Logging Custom Resource (CR) to add the `clusterlogging.openshift.io/logforwardingtechpreview: enabled` annotation and create a Log Forwarding Custom Resource to specify the outputs, pipelines, and enable log forwarding.

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

Procedure

To configure log forwarding using the API:

1. Create a Log Forwarding CR YAML file similar to the following:

   ```yaml
   apiVersion: "logging.openshift.io/v1alpha1"
   kind: "LogForwarding"
   metadata:
     name: instance
     namespace: openshift-logging 
   ```
The name of the log forwarding CR must be **instance**.

The namespace for the log forwarding CR must be **openshift-logging**.

Set to **true** disable the default log forwarding behavior.

Add one or more endpoints:

- Specify the type of output, either **elasticsearch** or **forward**.
- Enter a name for the output.
- Enter the endpoint, either the server name or FQDN.
- Optionally, enter the name of the secret required by the endpoint for TLS communication. The secret must exist in the openshift-logging project. For the default Elasticsearch logstore, use **fluentd** as the secret name.
- Specify **insecure: true** if the endpoint does not use a secret, resulting in insecure communication.

Add one or more pipelines:

- Enter a name for the pipeline
• Specify the source type: `logs.app`, `logs.infra`, or `logs.audit`.
• Specify the name of one or more outputs configured in the CR.

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

2. Create the CR object:

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

**8.9.3.3.1. Example log forwarding custom resources**

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

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

**Sample custom resource to forward to an Elasticsearch logstore**

```
apiVersion: logging.openshift.io/v1alpha1
customResource: LogForwarding
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
    - name: user-created-es
      type: elasticsearch
      endpoint: 'elasticsearch-server.openshift-logging.svc:9200'
      secret:
        name: pipelinesecret
  pipelines:
    - name: app-pipeline
      inputSource: logs.app
      outputRefs:
        - user-created-es
    - name: infra-pipeline
      inputSource: logs.infra
      outputRefs:
        - user-created-es
    - name: audit-pipeline
      inputSource: logs.audit
      outputRefs:
        - user-created-es
```

The following Log Forwarding custom resource sends all logs to a secured Fluentd instance using the Fluentd **forward** protocol.
Sample custom resource to use the forward protocol

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

### 8.9.3.4. Disabling the Log Forwarding API

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

**NOTE**

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

**Procedure**

To disable the Log Forwarding API:

1. Edit the Cluster Logging Custom Resource (CR) in the `openshift-logging` project:
   ```bash
   $ oc edit ClusterLogging instance
   ```

2. Remove the `clusterlogging.openshift.io/logforwardingtechpreview` annotation:
   ```yaml
   apiVersion: "logging.openshift.io/v1"
   kind: "ClusterLogging"
   ```
8.10. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD

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

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

8.10.1. 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=no  
   ForwardToConsole=yes  
   ForwardToSyslog=no  
   MaxRetentionSec=30  
   RateLimitBurst=10000  
   RateLimitInterval=1s  
   Storage=volatile  
   SyncIntervalSec=1s  
   SystemMaxUse=8g  
   SystemKeepFree=20%  
   SystemMaxFileSize=10M

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.

4. 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 1 month.

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

6. 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 **/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.

7. 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**.

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

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

10. 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](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:

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

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

```yaml
...  
config:
  storage:
    files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,$(jrnl_cnf)
          verification: {}
          filesystem: root
          mode: 0644
          path: /etc/systemd/journald.conf
          systemd: {}

1. Set the permissions for the `journal.conf` file. It is recommended to set 0644 permissions.
2. Specify the path to the base64-encoded `journal.conf` file.

4. Create the MachineConfig:

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

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

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

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

   ```yaml
   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
   ```
CHAPTER 9. VIEWING ELASTICSEARCH STATUS

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

9.1. VIEWING ELASTICSEARCH STATUS

You can view the status of your Elasticsearch cluster.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

1. Change to the openshift-logging project.

   `$ oc project openshift-logging`

2. To view the Elasticsearch cluster status:

   a. Get the name of the Elasticsearch instance:

      `$ oc get Elasticsearch`

      | NAME       | AGE     |
      |------------|---------|
      | elasticsearch | 5h9m   |

   b. Get the Elasticsearch 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:

      ```
      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-1-6d7f844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
data:
  failed: []
  notReady: []
  ready:
    - elasticsearch-cdm-zjf34ved-1-6d7f844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
master:
  failed: []
  notReady: []
  ready:
    - elasticsearch-cdm-zjf34ved-1-6d7f844f-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 Elasticsearch cluster:**
   
   - The number of active primary shards.
   - The number of active shards.
   - The number of shards that are initializing.
   - The number of Elasticsearch data nodes.
   - The total number of Elasticsearch nodes.
   - The number of pending tasks.
   - The Elasticsearch status: **green, red, yellow**.
   - The number of unassigned shards.

3. **Any status conditions, if present.** The Elasticsearch cluster 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 Elasticsearch and proxy containers.
   - Container Terminated for both the Elasticsearch and proxy containers.
• Pod unschedulable. Also, a condition is shown for a number of issues, see Example condition messages.

The Elasticsearch nodes in the cluster, with **upgradeStatus**.

The Elasticsearch client, data, and master pods in the cluster, listed under ‘failed’, **notReady** or **ready** state.

### 9.1.1. 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.

```yaml
status:
  nodes:
    - conditions:
        message: Disk storage usage for node is 27.5gb (36.74%). Shards will be not be allocated on this node.
        reason: Disk Watermark Low
        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.

```yaml
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 Elasticsearch node selector in the CR does not match any nodes in the cluster:

```yaml
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 Elasticsearch 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 Elasticsearch cluster does not have enough nodes to support your Elasticsearch 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 master nodes:

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

9.2. VIEWING ELASTICSEARCH COMPONENT STATUS

You can view the status for a number of Elasticsearch 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

   pod/elasticsearch-cdm-1godmszn-1-6f8495-yp4lw
   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-1godmszn-1-6f8495-vp4lw -- indices

Defaulting container name to elasticsearch.
Use `oc describe pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw -n openshift-logging` to see all of the containers in this pod.

Wed Apr 10 05:42:12 UTC 2019

<table>
<thead>
<tr>
<th>health status index</th>
<th>uuid</th>
<th>pri</th>
<th>rep</th>
<th>docs.count</th>
</tr>
</thead>
<tbody>
<tr>
<td>docs.deleted store.size</td>
<td>pri.store.size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>red open .kibana.647a750f1787408bf50008234ec0edd5a6a9b2ac</td>
<td>GTewEJEZqaus9QjvBBnGg</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>green open .operations.2019.04.10</td>
<td>ausZHoKxTNOoBvv9RlXfrw</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2176114</td>
<td>0</td>
<td>1956</td>
<td></td>
<td></td>
</tr>
<tr>
<td>green open .operations.2019.04.11</td>
<td>chOwDnQlSsqhfSPcot1Yiw</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1494624</td>
<td>0</td>
<td>1475</td>
<td></td>
<td></td>
</tr>
<tr>
<td>green open .kibana</td>
<td>chOwDnQlSsqhfSPcot1Yiw</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>green open .searchguard</td>
<td>chOwDnQlSsqhfSPcot1Yiw</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Elasticsearch pods

You can view the status of the Elasticsearch pods.

1. Get the name of a pod:

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

      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:

   ....
   Status: Running
   ....

   Containers:
   elasticsearch:
      Container ID: cri-o://b7d44e0a9ea486e27f47763f5bb4c39dfd2
      State: Running
      Started: Mon, 08 Apr 2019 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://3f77032abaddbb1652c116278652908dc01860320b8a4e741d06894b2f8f9aa1
State: Running
Started: Mon, 08 Apr 2019 10:18:38 -0400
Ready: True
Restart Count: 0

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

Events: <none>

Elasticsearch deployment configuration
You can view the status of the Elasticsearch deployment configuration.

1. Get the name of a deployment configuration:

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

   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
Elasticsearch ReplicaSet

You can view the status of the Elasticsearch ReplicaSet.

1. Get the name of a replica set:

   ```bash
   $ 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:

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

   The output includes the following status information:

   ```bash
   ....
   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
   ....
   Events: <none>
   ```
CHAPTER 10. VIEWING CLUSTER LOGGING STATUS

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

10.1. 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

The output includes information similar to the following:

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

status:
  collection:
  logs:
    fluentdStatus:
      daemonSet: fluentd
      nodes:
        fluentd-2rhqp: ip-10-0-169-13.ec2.internal
        fluentd-6fgjh: ip-10-0-165-244.ec2.internal
        fluentd-6l2ff: ip-10-0-128-218.ec2.internal
        fluentd-54nx5: ip-10-0-139-30.ec2.internal
        fluentd-flpnn: ip-10-0-147-228.ec2.internal
        fluentd-n2frh: ip-10-0-157-45.ec2.internal

  pods:
  failed: []
  notReady: []
  ready:
  - fluentd-2rhqp
  - fluentd-54nx5
  - fluentd-6fgjh
  - fluentd-6l2ff
  - fluentd-flpnn
  - fluentd-n2frh
```
In the output, the cluster status fields appear in the **status** stanza.

Information on the Fluentd pods.

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

Information on the Kibana pods.

### 10.1.1. 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:

```
nodes:
- conditions:
    message: Disk storage usage for node is 27.5gb (36.74%). Shards will be not be allocated on this node.
    reason: Disk Watermark Low
    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:

```
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:

```
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
  Reason: Unschedulable
Message: 0/5 nodes are available: 5 node(s) didn't match node selector.
Status: True
Type: Unschedulable
elasticsearch-cdm-mkkdys93-2:
Node Count: 2
Pods:
Client:
  Failed:
  Not Ready:
    elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
    elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
  Ready:
Data:
  Failed:
  Not Ready:
    elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
    elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
  Ready:
Master:
  Failed:
  Not Ready:
    elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49
    elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
  Ready:

A status message similar to the following indicates that the requested PVC could not bind to PV:

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

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

Curation:
Curator Status:
Cluster Condition:
curator-1561518900-cjx8d:
  Last Transition Time: 2019-06-26T03:20:08Z
  Reason: Completed
  Status: True
  Type: ContainerTerminated
curator-1561519200-zqxxj:
  Last Transition Time: 2019-06-26T03:20:01Z
  Message: 0/5 nodes are available: 1 Insufficient cpu, 5 node(s) didn't match node selector.
  Reason: 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:

Status:
Collection:
Logs:
Fluentd Status:
  Daemon Set: fluentd
Nodes:
  Pods:
  Failed:
  Not Ready:
Ready:

10.2. 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 deployment:

   $ oc describe deployment cluster-logging-operator

   The output includes the following status information:

   | Name:          cluster-logging-operator
   | Conditions:    Status Reason                    
   | Available:     True   MinimumReplicasAvailable  
   | Progressing:   True   NewReplicaSetAvailable    

   Events:
3. View the status of the cluster logging ReplicaSet:

   a. Get the name of a ReplicaSet:

   ```bash
   $ oc get replicaset
   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 ReplicaSet:

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

   The output includes the following status information:

   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----
   ```
CHAPTER 11. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS

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

11.1. 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.

NOTE

You should set your MachineSet to use at least 6 replicas.

Prerequisites

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

Procedure

1. Edit the Cluster Logging Custom Resource in the openshift-logging project:

   ```bash
   $ oc edit ClusterLogging instance
   
   apiVersion: logging.openshift.io/v1
   kind: ClusterLogging
   ....
   spec:
   collection:
   logs:
   fluentd:
   resources: null
   type: fluentd
   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: " 4
proxy:
  resources: null
replicas: 1
resources: null
type: kibana

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 steps
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
  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-2compute.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
  NAME                                         STATUS   ROLES          AGE   VERSION
  ip-10-0-133-216.us-east-2.compute.internal   Ready    master         60m   v1.17.1
  ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.17.1
  ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.17.1
  ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.17.1
  ip-10-0-147-79.us-east-2.compute.internal    Ready    worker         51m   v1.17.1
  ip-10-0-152-241.us-east-2.compute.internal   Ready    infra          51m   v1.17.1
  ip-10-0-139-48.us-east-2.compute.internal    Ready    infra          51m   v1.17.1

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

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'
lables:
  node-role.kubernetes.io/infra: "

- To move the Kibana Pod, edit the Cluster Logging CR to add a node selector:

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

....

spec:
....

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

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

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

$ oc get pods

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-7d85dcfc8-bfpcf                           2/2     Running       0          33s
• The new pod is on the **ip-10-0-139-48.us-east-2.compute.internal** node:

```bash
$ oc get pod kibana-7d85dcff8-bfpfp -o wide
NAME                      READY   STATUS      RESTARTS   AGE   IP                  NODE
NOMINATED NODE   READINESS GATES
kibana-7d85dcff8-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.

```bash
$ oc get pods
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-7d85dcff8-bfpfp                         2/2     Running   0          62s
```
CHAPTER 12. MANUALLY ROLLING OUT ELASTICSEARCH

OpenShift Container Platform supports the Elasticsearch rolling cluster restart. A rolling restart applies appropriate changes to the Elasticsearch cluster without down time (if three masters are configured). The Elasticsearch cluster remains online and operational, with nodes taken offline one at a time.

12.1. PERFORMING AN ELASTICSEARCH ROLLING CLUSTER RESTART

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

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

Prerequisite

- Cluster logging and Elasticsearch must be installed.

Procedure

To perform a rolling cluster restart:

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

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

2. Use the following command to extract the CA certificate from Elasticsearch and write to the `admin-ca` file:

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

3. Perform a shard synced flush to ensure there are no pending operations waiting to be written to disk prior to shutting down:

   ```bash
   $ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- curl -s --cacert /etc/elasticsearch/secret/admin-ca --cert /etc/elasticsearch/secret/admin-cert --key /etc/elasticsearch/secret/admin-key -XPOST 'https://localhost:9200/_flush/synced'
   ```

   For example:

   ```bash
   oc exec -c elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -- curl -s --cacert /etc/elasticsearch/secret/admin-ca --cert /etc/elasticsearch/secret/admin-cert --key /etc/elasticsearch/secret/admin-key -XPOST 'https://localhost:9200/_flush/synced'
   ```

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

   ```bash
   $ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query=_cluster/settings -XPUT 'https://localhost:9200/_cluster/settings' -d '{ "transient": { "cluster.routing.allocation.enable": "none" } }'
   ```

   For example:

   ```bash
   ```
  "transient": { "cluster.routing.allocation.enable" : "none" } }
{
  "acknowledged" : true,
  "persistent" : {},
  "transient" : {
    "cluster" : {
      "routing" : {
        "allocation" : {
          "enable" : "none"
        }
      }
    }
  }
}

5. Once 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
   deployment.extensions/elasticsearch-cdm-0-1 resumed

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

   $ oc get pods | grep elasticsearch-*

   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. Once complete, reset the pod to disallow rollouts:

   $ oc rollout pause deployment/<deployment-name>

   For example:

   $ oc rollout pause deployment/elasticsearch-cdm-0-1
   deployment.extensions/elasticsearch-cdm-0-1 paused

c. Check that the Elasticsearch cluster is in green 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:

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

{
  "cluster_name" : "elasticsearch",
  "status" : "green",  
  "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 is green before proceeding.

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

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

```bash
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query=_cluster/settings -XPUT 'https://localhost:9200/_cluster/settings' -d '{ "transient": { "cluster.routing.allocation.enable": "none" }}'
```

For example:

```bash
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --query=_cluster/settings?pretty=true -XPUT 'https://localhost:9200/_cluster/settings' -d '{ "transient": { "cluster.routing.allocation.enable": "all" }}'

{
  "acknowledged" : true,
  "persistent" : {},
  "transient" : {
    "cluster" : {
      "routing" : {
        "allocation" : {
```
"enable": "all"
}
}
}
}
CHAPTER 13. TROUBLESHOOTING KIBANA

Using the Kibana console with OpenShift Container Platform can cause problems that are easily solved, but are not accompanied with useful error messages. Check the following troubleshooting sections if you are experiencing any problems when deploying Kibana on OpenShift Container Platform.

13.1. TROUBLESHOOTING A KUBERNETES LOGIN LOOP

The OAuth2 proxy on the Kibana console must share a secret with the master host’s OAuth2 server. If the secret is not identical on both servers, it can cause a login loop where you are continuously redirected back to the Kibana login page.

Procedure

To fix this issue:

1. Run the following command to delete the current OAuthClient:

   $ oc delete oauthclient/kibana-proxy

13.2. TROUBLESHOOTING A KUBERNETES CRYPTIC ERROR WHEN VIEWING THE KIBANA CONSOLE

When attempting to visit the Kibana console, you may receive a browser error instead:

   {"error":"invalid_request","error_description":"The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed."}

This can be caused by a mismatch between the OAuth2 client and server. The return address for the client must be in a whitelist so the server can securely redirect back after logging in.

Fix this issue by replacing the OAuthClient entry.

Procedure

To replace the OAuthClient entry:

1. Run the following command to delete the current OAuthClient:

   $ oc delete oauthclient/kibana-proxy

If the problem persists, check that you are accessing Kibana at a URL listed in the OAuth client. This issue can be caused by accessing the URL at a forwarded port, such as 1443 instead of the standard 443 HTTPS port. You can adjust the server whitelist by editing the OAuth client:

   $ oc edit oauthclient/kibana-proxy

13.3. TROUBLESHOOTING A KUBERNETES 503 ERROR WHEN VIEWING THE KIBANA CONSOLE

If you receive a proxy error when viewing the Kibana console, it could be caused by one of two issues:
Kibana might not be recognizing pods. If Elasticsearch is slow in starting up, Kibana may timeout trying to reach it. Check whether the relevant service has any endpoints:

```
$ oc describe service kibana
Name:                   kibana
[...]                     
Endpoints:              <none>
```

If any Kibana pods are live, endpoints are listed. If they are not, check the state of the Kibana pods and deployment. You might have to scale the deployment down and back up again.

- The route for accessing the Kibana service is masked. This can happen if you perform a test deployment in one project, then deploy in a different project without completely removing the first deployment. When multiple routes are sent to the same destination, the default router will only route to the first created. Check the problematic route to see if it is defined in multiple places:

```
$ oc get route --all-namespaces --selector logging-infra=support
```
CHAPTER 14. 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

14.1. DEFAULT EXPORTED FIELDS

These are the default fields exported by the logging system and available for searching from Elasticsearch and Kibana. The default fields are Top Level and collectd.*

Top Level Fields
The top level fields are common to every application, and may be present in every record. For the Elasticsearch template, top level fields populate the actual mappings of default in the template’s mapping section.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@timestamp</td>
<td>The UTC value marking when the log payload was created, or when the log payload was first collected if the creation time is not known. This is the log processing pipeline’s best effort determination of when the log payload was generated. Add the @ prefix convention to note a field as being reserved for a particular use. With Elasticsearch, most tools look for @timestamp by default. For example, the format would be 2015-01-24 14:06:05.071000.</td>
</tr>
<tr>
<td>geoip</td>
<td>This is geo-ip of the machine.</td>
</tr>
<tr>
<td>hostname</td>
<td>The hostname is the fully qualified domain name (FQDN) of the entity generating the original payload. This field is an attempt to derive this context. Sometimes the entity generating it knows the context. While other times that entity has a restricted namespace itself, which is known by the collector or normalizer.</td>
</tr>
<tr>
<td>ipaddr4</td>
<td>The IP address V4 of the source server, which can be an array.</td>
</tr>
<tr>
<td>ipaddr6</td>
<td>The IP address V6 of the source server, if available.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>level</td>
<td>The logging level as provided by rsyslog (severitytext property), python’s logging module. Possible values are as listed at 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.</td>
</tr>
<tr>
<td>message</td>
<td>A typical log entry message, or payload. It can be stripped of metadata pulled out of it by the collector or normalizer, that is UTF-8 encoded.</td>
</tr>
<tr>
<td>pid</td>
<td>This is the process ID of the logging entity, if available.</td>
</tr>
<tr>
<td>service</td>
<td>The name of the service associated with the logging entity, if available. For example, the syslog APP-NAME property is mapped to the service field.</td>
</tr>
<tr>
<td>tags</td>
<td>Optionally provided operator defined list of tags placed on each log by the collector or normalizer. The payload can be a string with whitespace-delimited string tokens, or a JSON list of string tokens.</td>
</tr>
<tr>
<td>file</td>
<td>Optional path to the file containing the log entry local to the collector TODO analyzer for file paths.</td>
</tr>
<tr>
<td>offset</td>
<td>The offset value can represent bytes to the start of the log line in the file (zero or one based), or log line numbers (zero or one based), as long as the values are strictly monotonically increasing in the context of a single log file. The values are allowed to wrap, representing a new version of the log file (rotation).</td>
</tr>
<tr>
<td>namespace_name</td>
<td>Associate this record with the namespace that shares it’s name. This value will not be stored, but it is used to associate the record with the appropriate namespace for access control and visualization. Normally this value will be given in the tag, but if the protocol does not support sending a tag, this field can be used. If this field is present, it will override the namespace given in the tag or in kubernetes.namespace_name.</td>
</tr>
<tr>
<td>namespace_uuid</td>
<td>This is the uuid associated with the namespace_name. This value will not be stored, but is used to associate the record with the appropriate namespace for access control and visualization. If this field is present, it will override the uuid given in kubernetes.namespace_uuid. This will also cause the Kubernetes metadata lookup to be skipped for this log record.</td>
</tr>
</tbody>
</table>
### collectd Fields
The following fields represent namespace metrics metadata.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.processes.ps_state</td>
<td>type: integer The <code>collectd</code> <code>ps_state</code> type of processes plug-in.</td>
</tr>
</tbody>
</table>

### collectd.processes.ps_disk_ops Fields
The `collectd` `ps_disk_ops` type of processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| collectd.processes.ps_disk_ops.read | type: float
|                            | TODO |
| collectd.processes.ps_disk_ops.write  | type: float
<p>|                            | TODO |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.processes.ps_vm</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd ps_vm</code> type of processes plug-in.</td>
</tr>
<tr>
<td>collectd.processes.ps_rss</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd ps_rss</code> type of processes plug-in.</td>
</tr>
<tr>
<td>collectd.processes.ps_data</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd ps_data</code> type of processes plug-in.</td>
</tr>
<tr>
<td>collectd.processes.ps_code</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd ps_code</code> type of processes plug-in.</td>
</tr>
<tr>
<td>collectd.processes.ps_stacksize</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd ps_stacksize</code> type of processes plug-in.</td>
</tr>
</tbody>
</table>

**collectd.processes.ps_cputime Fields**

The `collectd ps_cputime` type of processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.processes.ps_cputime.user</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.processes.ps_cputime.syst</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.processes.ps_count Fields**

The `collectd ps_count` type of processes plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.processes.ps_count.processes</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.processes.ps_count.threads</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.processes.ps_pagefaults.majflt</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.processes.ps_pagefaults.minflt</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.processes.ps_disk_octets** Fields

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.processes.ps_disk_octets.read</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.processes.ps_disk_octets.write</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.processes.fork_rate</td>
<td>type: float</td>
</tr>
<tr>
<td>The <strong>collectd fork_rate</strong> type of processes plug-in.</td>
<td></td>
</tr>
</tbody>
</table>

**collectd.disk** Fields

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

**collectd.disk.disk_merged** Fields

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.disk.disk_merged.read</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.disk.disk_merged.write</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.disk.disk_octets** Fields

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_octets.read</code></td>
<td>type: float, TODO</td>
</tr>
<tr>
<td><code>collectd.disk.disk_octets.write</code></td>
<td>type: float, TODO</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_time.read</code></td>
<td>type: float, TODO</td>
</tr>
<tr>
<td><code>collectd.disk.disk_time.write</code></td>
<td>type: float, TODO</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_ops.read</code></td>
<td>type: float, TODO</td>
</tr>
<tr>
<td><code>collectd.disk.disk_ops.write</code></td>
<td>type: float, TODO</td>
</tr>
<tr>
<td><code>collectd.disk.pending_operations</code></td>
<td>type: integer</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.disk.disk_io_time.io_time</code></td>
<td>type: float, TODO</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>collectd.disk.disk_io_time.weighted_io_time</code></td>
<td>type: float TODO</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.interface.if_octets.rx</code></td>
<td>type: float TODO</td>
</tr>
<tr>
<td><code>collectd.interface.if_octets.tx</code></td>
<td>type: float TODO</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.interface.if_packets.rx</code></td>
<td>type: float TODO</td>
</tr>
<tr>
<td><code>collectd.interface.if_packets.tx</code></td>
<td>type: float TODO</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.interface.if_errors.rx</code></td>
<td>type: float TODO</td>
</tr>
<tr>
<td><code>collectd.interface.if_errors.tx</code></td>
<td>type: float TODO</td>
</tr>
</tbody>
</table>
**collectd.interface.if_dropped Fields**
The **collectd if_dropped** type of interface plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.interface.if_dropped.rx</td>
<td>type: float</td>
</tr>
<tr>
<td>collectd.interface.if_dropped.tx</td>
<td>type: float</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.if_octets.rx</td>
<td>type: float</td>
</tr>
<tr>
<td>collectd.virt.if_octets.tx</td>
<td>type: float</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.if_packets.rx</td>
<td>type: float</td>
</tr>
<tr>
<td>collectd.virt.if_packets.tx</td>
<td>type: float</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>collectd.virt.if_errors.rx</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.virt.if_errors.tx</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.virt.if_dropped Fields**

The `collectd if_dropped` type of virt plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.if_dropped.rx</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.virt.if_dropped.tx</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.virt.disk_ops Fields**

The `collectd disk_ops` type of virt plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.disk_ops.read</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.virt.disk_ops.write</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.virt.disk_octets Fields**

The `collectd disk_octets` type of virt plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.virt.disk_octets.read</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.virt.disk_octets.write</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>collectd.virt.memory</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> memory type of <code>virt</code> plug-in.</td>
</tr>
<tr>
<td>collectd.virt.vcpu</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> <code>vcpu</code> type of <code>virt</code> plug-in.</td>
</tr>
<tr>
<td>collectd.virt.vcpu_total</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> <code>vcpu_total</code> type of <code>virt</code> plug-in.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.CPU.percent</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> type percent of plug-in CPU.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.df.df_complex</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> type <code>df_complex</code> of plug-in <code>df</code>.</td>
</tr>
<tr>
<td>collectd.df.percent_bytes</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> type <code>percent_bytes</code> of plug-in <code>df</code>.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.entropy.entropy</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <code>collectd</code> entropy type of entropy plug-in.</td>
</tr>
</tbody>
</table>

**collectd.memory Fields**
Corresponds to the `collectd` memory plug-in.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.memory.memory</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd</em> memory type of memory plug-in.</td>
</tr>
<tr>
<td>collectd.memory.percent</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd</em> percent type of memory plug-in.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.swap.swap</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd</em> swap type of swap plug-in.</td>
</tr>
<tr>
<td>collectd.swap.swap_io</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <em>collectd swap_io</em> type of swap plug-in.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.load.load.shortterm</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.load.load midterm</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td>collectd.load.load.longterm</td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

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

---

OpenShift Container Platform 4.4 Logging

124
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `collectd.aggregation.percent` | type: float  
  TODO |

**collectd.statsd Fields**

Corresponds to `collectd statsd` plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `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. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `collectd.statsd.host_storage` | type: integer  
The `collectd` storage type of `statsd` plug-in. |
| `collectd.statsd.host_swap` | type: integer  
The `collectd` swap type of `statsd` plug-in. |
| `collectd.statsd.host_vds` | type: integer  
The `collectd` VDSM type of `statsd` plug-in. |
| `collectd.statsd.host_vms` | type: integer  
The `collectd` VMS type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_tx_dropped` | type: integer  
The `collectd nic_tx_dropped` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_rx_bytes` | type: integer  
The `collectd nic_rx_bytes` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_tx_bytes` | type: integer  
The `collectd nic_tx_bytes` type of `statsd` plug-in. |
| `collectd.statsd.vm_balloon_min` | type: integer  
The `collectd balloon_min` type of `statsd` plug-in. |
| `collectd.statsd.vm_balloon_max` | type: integer  
The `collectd balloon_max` type of `statsd` plug-in. |
| `collectd.statsd.vm_balloon_target` | type: integer  
The `collectd balloon_target` type of `statsd` plug-in. |
| `collectd.statsd.vm_balloon_cur` | type: integer  
The `collectd balloon_cur` type of `statsd` plug-in. |
| `collectd.statsd.vm_cpu_sys` | type: integer  
The `collectd cpu_sys` type of `statsd` plug-in. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. |
## collectd.statsd Fields

Corresponds to `collectd` postgresql plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.statsd.vm_nic_rx_dropped</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> <code>nic_rx_dropped</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.vm_cpu_user</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> <code>cpu_user</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.vm_nic_rx_errors</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> <code>nic_rx_errors</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.vm_nic_tx_errors</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> <code>nic_tx_errors</code> type of <code>statsd</code> plug-in.</td>
</tr>
<tr>
<td><code>collectd.statsd.vm_nic_speed</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> <code>nic_speed</code> type of <code>statsd</code> plug-in.</td>
</tr>
</tbody>
</table>

## collectd.postgresql Fields

Corresponds to `collectd` postgresql plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.postgresql.pg_n_tup_g</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> type <code>pg_n_tup_g</code> of plug-in postgresql.</td>
</tr>
<tr>
<td><code>collectd.postgresql.pg_n_tup_c</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> type <code>pg_n_tup_c</code> of plug-in postgresql.</td>
</tr>
<tr>
<td><code>collectd.postgresql.pg_numbackends</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> type <code>pg_numbackends</code> of plug-in postgresql.</td>
</tr>
<tr>
<td><code>collectd.postgresql.pg_xact</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> type <code>pg_xact</code> of plug-in postgresql.</td>
</tr>
<tr>
<td><code>collectd.postgresql.pg_db_size</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> type <code>pg_db_size</code> of plug-in postgresql.</td>
</tr>
<tr>
<td><code>collectd.postgresql.pg_blks</code></td>
<td>type: integer  &lt;br&gt;The <code>collectd</code> type <code>pg_blks</code> of plug-in postgresql.</td>
</tr>
</tbody>
</table>
14.2. SYSTEMD EXPORTED FIELDS

These are the **systemd** fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana.

Contains common fields specific to **systemd** journal. Applications may write their own fields to the journal. These will be available under the **systemd.u** namespace. **RESULT** and **UNIT** are two such fields.

**systemd.k** Fields
The following table contains **systemd** kernel-specific metadata.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>systemd.k.KERNEL_DEVICE</strong></td>
<td><strong>systemd.k.KERNELDEVICE</strong> is the kernel device name.</td>
</tr>
<tr>
<td><strong>systemd.k.KERNEL_SUBSYSTEM</strong></td>
<td><strong>systemd.k.KERNEL_SUBSYSTEM</strong> is the kernel subsystem name.</td>
</tr>
<tr>
<td><strong>systemd.k.UDEV_DEVLINK</strong></td>
<td><strong>systemd.k.UDEV_DEVLINK</strong> includes additional symlink names that point to the node.</td>
</tr>
<tr>
<td><strong>systemd.k.UDEV_DEVNODE</strong></td>
<td><strong>systemd.k.UDEV_DEVNODE</strong> is the node path of the device.</td>
</tr>
<tr>
<td><strong>systemd.k.UDEV_SYSNAME</strong></td>
<td><strong>systemd.k.UDEV_SYSNAME</strong> is the kernel device name.</td>
</tr>
</tbody>
</table>

**systemd.t** Fields
**systemd.t** Fields are trusted journal fields, fields that are implicitly added by the journal, and cannot be altered by client code.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>systemd.t.AUDIT_LOGINUID</strong></td>
<td><strong>systemd.t.AUDIT_LOGINUID</strong> is the user ID for the journal entry process.</td>
</tr>
<tr>
<td><strong>systemd.t.BOOT_ID</strong></td>
<td><strong>systemd.t.BOOT_ID</strong> is the kernel boot ID.</td>
</tr>
<tr>
<td><strong>systemd.t.AUDIT_SESSION</strong></td>
<td><strong>systemd.t.AUDIT_SESSION</strong> is the session for the journal entry process.</td>
</tr>
<tr>
<td><strong>systemd.t.CAP_EFFECTIVE</strong></td>
<td><strong>systemd.t.CAP_EFFECTIVE</strong> represents the capabilities of the journal entry process.</td>
</tr>
<tr>
<td><strong>systemd.t.CMDLINE</strong></td>
<td><strong>systemd.t.CMDLINE</strong> is the command line of the journal entry process.</td>
</tr>
<tr>
<td><strong>systemd.t.COMM</strong></td>
<td><strong>systemd.t.COMM</strong> is the name of the journal entry process.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>systemd.t.EXE</td>
<td>systemd.t.EXE is the executable path of the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.GID</td>
<td>systemd.t.GID is the group ID for the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.HOSTNAME</td>
<td>systemd.t.HOSTNAME is the name of the host.</td>
</tr>
<tr>
<td>systemd.t.MACHINE_ID</td>
<td>systemd.t.MACHINE_ID is the machine ID of the host.</td>
</tr>
<tr>
<td>systemd.t.PID</td>
<td>systemd.t.PID is the process ID for the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.SELINUX_CONTEXT</td>
<td>systemd.t.SELINUX_CONTEXT is the security context, or label, for the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.SOURCE_REALTIME_TIMESTAMP</td>
<td>systemd.t.SOURCE_REALTIME_TIMESTAMP is the earliest and most reliable timestamp of the message. This is converted to RFC 3339 NS format.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_CGROUP</td>
<td>systemd.t.SYSTEMD_CGROUP is the systemd control group path.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_OWNER_UID</td>
<td>systemd.t.SYSTEMD_OWNER_UID is the owner ID of the session.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_SESSION</td>
<td>systemd.t.SYSTEMD_SESSION, if applicable, is the systemd session ID.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_SLICE</td>
<td>systemd.t.SYSTEMD_SLICE is the slice unit of the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_UNIT</td>
<td>systemd.t.SYSTEMD_UNIT is the unit name for a session.</td>
</tr>
<tr>
<td>systemd.t.SYSTEMD_USER_UNIT</td>
<td>systemd.t.SYSTEMD_USER_UNIT, if applicable, is the user unit name for a session.</td>
</tr>
<tr>
<td>systemd.t.TRANSPORT</td>
<td>systemd.t.TRANSPORT is the method of entry by the journal service. This includes, audit, driver, syslog, journal, stdout, and kernel.</td>
</tr>
<tr>
<td>systemd.t.UID</td>
<td>systemd.t.UID is the user ID for the journal entry process.</td>
</tr>
<tr>
<td>systemd.t.SYSLOG_FACILITY</td>
<td>systemd.t.SYSLOG_FACILITY is the field containing the facility, formatted as a decimal string, for syslog.</td>
</tr>
<tr>
<td>systemd.t.SYSLOG_IDENTIFIER</td>
<td>systemd.t.systemd.t.SYSLOG_IDENTIFIER is the identifier for syslog.</td>
</tr>
</tbody>
</table>
### 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. |

### 14.3. 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. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kubernetes.host</td>
<td>Kubernetes node name.</td>
</tr>
<tr>
<td>kubernetes.container_name</td>
<td>The name of the container in Kubernetes.</td>
</tr>
<tr>
<td>kubernetes.labels.deployment</td>
<td>The deployment associated with the Kubernetes object.</td>
</tr>
<tr>
<td>kubernetes.labels.deploymentconfig</td>
<td>The deploymentconfig associated with the Kubernetes object.</td>
</tr>
<tr>
<td>kubernetes.labels.component</td>
<td>The component associated with the Kubernetes object.</td>
</tr>
<tr>
<td>kubernetes.labels.provider</td>
<td>The provider associated with the Kubernetes object.</td>
</tr>
</tbody>
</table>

**kubernetes.annotations Fields**

Annotations associated with the OpenShift object are `kubernetes.annotations` fields.

### 14.4. CONTAINER EXPORTED FIELDS

These are the Docker fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana. Namespace for docker container-specific metadata. The `docker.container_id` is the Docker container ID.

**pipeline_metadata.collector Fields**

This section contains metadata specific to the collector.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipeline_metadata.collector.hostname</td>
<td>FQDN of the collector. It might be different from the FQDN of the actual emitter of the logs.</td>
</tr>
<tr>
<td>pipeline_metadata.collector.name</td>
<td>Name of the collector.</td>
</tr>
<tr>
<td>pipeline_metadata.collector.version</td>
<td>Version of the collector.</td>
</tr>
<tr>
<td>pipeline_metadata.collector.ipaddr4</td>
<td>IP address v4 of the collector server, can be an array.</td>
</tr>
<tr>
<td>pipeline_metadata.collector.ipaddr6</td>
<td>IP address v6 of the collector server, can be an array.</td>
</tr>
</tbody>
</table>
### Parameter | Description
--- | ---
**pipeline_metadata.collect** or **inputname** | How the log message was received by the collector whether it was TCP/UDP, or imjournal/imfile.

**pipeline_metadata.collect** or **received_at** | Time when the message was received by the collector.

**pipeline_metadata.collect** or **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.

### 14.5. oVIRT EXPORTED FIELDS
These are the oVirt fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana.

Namespace for oVirt metadata.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovirt.entity</td>
<td>The type of the data source, hosts, VMS, and engine.</td>
</tr>
<tr>
<td>ovirt.host_id</td>
<td>The oVirt host UUID.</td>
</tr>
</tbody>
</table>

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

14.6. AUSHAPE EXPORTED FIELDS

These are the Aushape fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana.

Audit events converted with Aushape. For more information, see Aushape.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aushape.serial</td>
<td>Audit event serial number.</td>
</tr>
<tr>
<td>aushape.node</td>
<td>Name of the host where the audit event occurred.</td>
</tr>
<tr>
<td>aushape.error</td>
<td>The error aushape encountered while converting the event.</td>
</tr>
<tr>
<td>aushape.trimmed</td>
<td>An array of JSONPath expressions relative to the event object, specifying objects or arrays with the content removed as the result of event size limiting. An empty string means the event removed the content, and an empty array means the trimming occurred by unspecified objects and arrays.</td>
</tr>
<tr>
<td>aushape.text</td>
<td>An array log record strings representing the original audit event.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aushape.data.avc</td>
<td>type: nested</td>
</tr>
<tr>
<td>aushape.data.execve</td>
<td>type: string</td>
</tr>
<tr>
<td>aushape.data.netfilterCfg</td>
<td>type: nested</td>
</tr>
</tbody>
</table>
14.7. TLOG EXPORTED FIELDS

These are the Tlog fields exported by the OpenShift Container Platform cluster logging system and available for searching from Elasticsearch and Kibana.

Tlog terminal I/O recording messages. For more information see Tlog.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tlog.ver</td>
<td>Message format version number.</td>
</tr>
<tr>
<td>tlog.user</td>
<td>Recorded user name.</td>
</tr>
<tr>
<td>tlog.term</td>
<td>Terminal type name.</td>
</tr>
<tr>
<td>tlog.session</td>
<td>Audit session ID of the recorded session.</td>
</tr>
<tr>
<td>tlog.id</td>
<td>ID of the message within the session.</td>
</tr>
<tr>
<td>tlog.pos</td>
<td>Message position in the session, milliseconds.</td>
</tr>
<tr>
<td>tlog.timing</td>
<td>Distribution of this message’s events in time.</td>
</tr>
<tr>
<td>tlog.in_txt</td>
<td>Input text with invalid characters scrubbed.</td>
</tr>
<tr>
<td>tlog.in_bin</td>
<td>Scrubbed invalid input characters as bytes.</td>
</tr>
<tr>
<td>tlog.out_txt</td>
<td>Output text with invalid characters scrubbed.</td>
</tr>
<tr>
<td>tlog.out_bin</td>
<td>Scrubbed invalid output characters as bytes.</td>
</tr>
</tbody>
</table>
CHAPTER 15. UNINSTALLING CLUSTER LOGGING

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

15.1. UNINSTALLING CLUSTER LOGGING FROM OPENSHIFT CONTAINER PLATFORM

You can remove cluster logging from your cluster.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

To remove cluster logging:

1. Use the following command to remove everything generated during the deployment.

   $ oc delete clusterlogging instance -n openshift-logging

2. Use the following command to remove the Persistent Volume Claims that remain after the Operator instances are deleted:

   $ oc delete pvc --all -n openshift-logging