



OpenShift Container Platform 4.1

Logging

Configuring cluster logging in OpenShift Container Platform 4.1

OpenShift Container Platform 4.1 Logging

Configuring cluster logging in OpenShift Container Platform 4.1

Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions for installing, configuring, and using the cluster logging feature. Cluster logging aggregates logs for a range of OpenShift Container Platform services.

Table of Contents

CHAPTER 1. ABOUT CLUSTER LOGGING AND OPENSIFT CONTAINER PLATFORM	5
1.1. ABOUT CLUSTER LOGGING	5
1.1.1. About cluster logging components	5
1.1.2. About Elasticsearch	5
1.1.3. About Fluentd	6
1.1.4. About Kibana	6
1.1.5. About Curator	6
1.1.6. About Event Router	7
1.1.7. About the Cluster Logging Custom Resource Definition	7
CHAPTER 2. ABOUT DEPLOYING CLUSTER LOGGING	9
2.1. ABOUT DEPLOYING AND CONFIGURING CLUSTER LOGGING	9
2.1.1. Configuring and Tuning Cluster Logging	9
2.1.2. Sample modified Cluster Logging Custom Resource	11
2.2. STORAGE CONSIDERATIONS FOR CLUSTER LOGGING AND OPENSIFT CONTAINER PLATFORM	12
2.3. ADDITIONAL RESOURCES	12
CHAPTER 3. DEPLOYING CLUSTER LOGGING	13
3.1. INSTALLING THE CLUSTER LOGGING AND ELASTICSEARCH OPERATORS	13
3.2. ADDITIONAL RESOURCES	20
CHAPTER 4. VIEWING THE KIBANA INTERFACE	21
4.1. LAUNCHING THE KIBANA INTERFACE	21
CHAPTER 5. WORKING WITH EVENT ROUTER	22
5.1. DEPLOYING AND CONFIGURING THE EVENT ROUTER	22
CHAPTER 6. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT	26
6.1. ABOUT CONFIGURING CLUSTER LOGGING	26
6.1.1. About deploying and configuring cluster logging	26
6.1.1.1. Configuring and Tuning Cluster Logging	26
6.1.1.2. Sample modified Cluster Logging Custom Resource	28
6.1.2. Moving the cluster logging resources	29
6.2. CHANGING CLUSTER LOGGING MANAGEMENT STATE	30
6.2.1. Changing the cluster logging management state	31
6.2.2. Changing the Elasticsearch management state	32
6.3. CONFIGURING CLUSTER LOGGING	33
6.3.1. Understanding the cluster logging component images	34
6.4. CONFIGURING ELASTICSEARCH TO STORE AND ORGANIZE LOG DATA	34
6.4.1. Configuring Elasticsearch CPU and memory limits	35
6.4.2. Configuring Elasticsearch replication policy	36
6.4.3. Configuring Elasticsearch storage	37
6.4.4. Configuring Elasticsearch for emptyDir storage	37
6.4.5. Exposing Elasticsearch as a route	38
6.4.6. About Elasticsearch alerting rules	41
6.5. CONFIGURING KIBANA	41
6.5.1. Configure Kibana CPU and memory limits	42
6.5.2. Scaling Kibana for redundancy	42
6.5.3. Installing the Kibana Visualize tool	43
6.6. CURATION OF ELASTICSEARCH DATA	44
6.6.1. Configuring the Curator schedule	44
6.6.2. Configuring Curator index deletion	45

6.6.3. Troubleshooting Curator	47
6.6.4. Configuring Curator in scripted deployments	48
6.6.5. Using the Curator Action file	49
6.7. CONFIGURING FLUENTD	50
6.7.1. Viewing Fluentd pods	51
6.7.2. Viewing Fluentd logs	51
6.7.3. Configure Fluentd CPU and memory limits	52
6.7.4. Configuring Fluentd log location	52
6.7.5. Throttling Fluentd logs	53
6.7.6. Understanding Buffer Chunk Limiting for Fluentd	54
6.7.7. Configuring Fluentd JSON parsing	55
6.7.8. Configuring how the log collector normalizes logs	56
6.7.9. Configuring Fluentd using environment variables	60
6.8. SENDING OPENSIFT CONTAINER PLATFORM LOGS TO EXTERNAL DEVICES	60
6.8.1. Configuring Fluentd to send logs to an external Elasticsearch instance	60
6.8.2. Configuring Fluentd to send logs to an external syslog server	62
6.8.3. Configuring Fluentd to send logs to an external log aggregator	63
6.9. CONFIGURING SYSTEMD-JOURNALD AND RSYSLOG	66
6.9.1. Configuring systemd-journald for cluster logging	66
CHAPTER 7. VIEWING ELASTICSEARCH STATUS	70
7.1. VIEWING ELASTICSEARCH STATUS	70
7.1.1. Example condition messages	71
7.2. VIEWING ELASTICSEARCH COMPONENT STATUS	72
CHAPTER 8. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS	76
8.1. SPECIFYING A NODE FOR CLUSTER LOGGING COMPONENTS USING NODE SELECTORS	76
CHAPTER 9. ADDITIONAL RESOURCES	78
CHAPTER 10. MANUALLY ROLLING OUT ELASTICSEARCH	79
10.1. PERFORMING AN ELASTICSEARCH ROLLING CLUSTER RESTART	79
CHAPTER 11. TROUBLESHOOTING KIBANA	83
11.1. TROUBLESHOOTING A KUBERNETES LOGIN LOOP	83
11.2. TROUBLESHOOTING A KUBERNETES CRYPTIC ERROR WHEN VIEWING THE KIBANA CONSOLE	83
11.3. TROUBLESHOOTING A KUBERNETES 503 ERROR WHEN VIEWING THE KIBANA CONSOLE	83
CHAPTER 12. EXPORTED FIELDS	85
12.1. DEFAULT EXPORTED FIELDS	85
Top Level Fields	85
collectd Fields	87
collectd.processes Fields	87
collectd.processes.ps_disk_ops Fields	88
collectd.processes.ps_cputime Fields	88
collectd.processes.ps_count Fields	89
collectd.processes.ps_pagefaults Fields	89
collectd.processes.ps_disk_octets Fields	89
collectd.disk Fields	90
collectd.disk.disk_merged Fields	90
collectd.disk.disk_octets Fields	90
collectd.disk.disk_time Fields	90
collectd.disk.disk_ops Fields	90
collectd.disk.disk_io_time Fields	91
collectd.interface Fields	91

collectd.interface.if_octets Fields	91
collectd.interface.if_packets Fields	91
collectd.interface.if_errors Fields	92
collectd.interface.if_dropped Fields	92
collectd.virt Fields	92
collectd.virt.if_octets Fields	92
collectd.virt.if_packets Fields	93
collectd.virt.if_errors Fields	93
collectd.virt.if_dropped Fields	93
collectd.virt.disk_ops Fields	93
collectd.virt.disk_octets Fields	94
collectd.CPU Fields	94
collectd.df Fields	94
collectd.entropy Fields	95
collectd.memory Fields	95
collectd.swap Fields	95
collectd.load Fields	95
collectd.load.load Fields	96
collectd.aggregation Fields	96
collectd.statsd Fields	96
collectd.postgresql Fields	99
12.2. RSYSLOG EXPORTED FIELDS	100
12.3. SYSTEMD EXPORTED FIELDS	100
systemd.k Fields	101
systemd.t Fields	101
systemd.u Fields	102
12.4. KUBERNETES EXPORTED FIELDS	103
kubernetes.labels Fields	103
kubernetes.annotations Fields	104
12.5. CONTAINER EXPORTED FIELDS	104
pipeline_metadata.collector Fields	104
pipeline_metadata.normalizer Fields	105
12.6. OVIRT EXPORTED FIELDS	105
ovirt.engine Fields	106
12.7. AUSHAPE EXPORTED FIELDS	106
aushape.data Fields	106
12.8. TLOG EXPORTED FIELDS	107
CHAPTER 13. UNINSTALLING CLUSTER LOGGING	108
13.1. UNINSTALLING CLUSTER LOGGING FROM OPENSIFT CONTAINER PLATFORM	108

CHAPTER 1. ABOUT CLUSTER LOGGING AND OPENSIFT CONTAINER PLATFORM

product-title} cluster logging aggregates all of the logs from your OpenShift Container Platform cluster, such as node system logs, application container logs, and so forth.

1.1. ABOUT 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 **instance**. 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. About 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 4 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.

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 Elasticsearch

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.

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.

For more information, see [Elasticsearch \(ES\)](#).

1.1.3. About Fluentd

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

Fluentd is deployed as a DaemonSet in OpenShift Container Platform that deploys pods to each OpenShift Container Platform node.

Fluentd uses **journald** as the system log source. These are 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, is 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 Kibana

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 Curator

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

```
spec:
  curation:
    type: "curator"
  resources:
    curator:
      schedule: "30 3 * * *" 1
```

- 1 Specify the Curator schedule in the [cron format](#).

For more information, see [Curator](#).

1.1.6. About Event Router

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 Definition

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

You should never have to modify this CRD. To make changes to your deployment, create and modify a specific 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

```
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:
    storageClassName: "gp2"
    size: "200G"
    redundancyPolicy: "SingleRedundancy"
  visualization:
```

```
type: "kibana"
kibana:
  resources:
    limits:
      memory: 1Gi
    requests:
      cpu: 500m
      memory: 1Gi
  proxy:
    resources:
      limits:
        memory: 100Mi
      requests:
        cpu: 100m
        memory: 100Mi
  replicas: 2
curation:
type: "curator"
curator:
  resources:
    limits:
      memory: 200Mi
    requests:
      cpu: 200m
      memory: 200Mi
  schedule: "*/10 * * * *"
collection:
logs:
type: "fluentd"
fluentd:
  resources:
    limits:
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 1Gi
```

CHAPTER 2. ABOUT DEPLOYING CLUSTER LOGGING

Before installing cluster logging into your 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:

```
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu:
          memory:
        requests:
          cpu: 1
          memory: 16Gi
      type: "elasticsearch"
  collection:
    logs:
      fluentd:
        resources:
          limits:
            cpu:
            memory:
          requests:
            cpu:
            memory:
        type: "fluentd"
  visualization:
    kibana:
      resources:
```

```

limits:
  cpu:
  memory:
requests:
  cpu:
  memory:
type: kibana
curation:
curator:
resources:
  limits:
    memory: 200Mi
    requests:
      cpu: 200m
      memory: 200Mi
  type: "curator"

```

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.

```

spec:
  logStore:
    type: "elasticsearch"
  elasticsearch:
    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:
    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](<https://en.wikipedia.org/wiki/Cron>).

```
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

```
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"
    curator:
      resources:
        limits:
```

```

    memory: 200Mi
    requests:
      cpu: 200m
      memory: 200Mi
    schedule: "*/5 * * * *"
collection:
logs:
  type: "fluentd"
  fluentd:
    resources:
      limits:
        memory: 1Gi
      requests:
        cpu: 200m
        memory: 1Gi

```

2.2. STORAGE CONSIDERATIONS FOR CLUSTER LOGGING AND OPENSIFT 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 **systemd journal** and `/var/log/containers/` to Elasticsearch.

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

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



NOTE

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

2.3. ADDITIONAL RESOURCES

For more information on installing operators, see [Installing Operators from the OperatorHub](#).

CHAPTER 3. DEPLOYING CLUSTER LOGGING

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

- Review the installation options in [About deploying cluster logging](#).
- Review the [cluster logging storage considerations](#).
- Install the Cluster Logging subscription using the web console.

3.1. INSTALLING THE CLUSTER LOGGING AND ELASTICSEARCH OPERATORS

You can use the OpenShift Container Platform console to install cluster logging, by deploying, the Cluster Logging and Elasticsearch Operators. The Cluster Logging Operator creates and manages the components of the logging stack. The Elasticsearch Operator creates and manages the Elasticsearch cluster used by cluster logging.



NOTE

The OpenShift Container Platform cluster logging solution requires that you install both the Cluster Logging Operator and Elasticsearch Operator. There is no use case in OpenShift Container Platform for installing the operators individually. You **must** install the Elasticsearch Operator using the CLI following the directions below. You can install the Cluster Logging Operator using the web console or CLI.

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.



NOTE

You **must** install the Elasticsearch Operator using the CLI following the directions below. You can install the Cluster Logging Operator using the web console or CLI.

Procedure

1. Create Namespaces for the Elasticsearch Operator and Cluster Logging Operator.



NOTE

You can also create the Namespaces in the web console using the **Administration** → **Namespaces** page. You must apply the **cluster-logging** and **cluster-monitoring** labels listed in the sample YAML to the namespaces you create.

- a. Create a Namespace for the Elasticsearch Operator (for example, **eo-namespace.yaml**):

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

- 1** You must specify the **openshift-operators-redhat** namespace. To prevent possible conflicts with metrics, you should configure the Prometheus Cluster Monitoring stack to scrape metrics from the **openshift-operators-redhat** namespace and not the **openshift-operators** namespace. The **openshift-operators** namespace can contain community operators, which are untrusted and could publish a metric with the same name as an OpenShift Container Platform metric, which would cause conflicts.
- 2** You must specify this label as shown to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.

- b. Run the following command to create the namespace:

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

For example:

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

- c. Create a Namespace for the Cluster Logging Operator (for example, **clo-namespace.yaml**):

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-logging 1
  annotations:
    openshift.io/node-selector: "" 2
  labels: 3
    openshift.io/cluster-logging: "true"
    openshift.io/cluster-monitoring: "true"
```

- 1 2 3** Specify these labels as shown.

- d. Run the following command to create the namespace:

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

For example:

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

2. 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:

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

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

- b. Create an Operator Group object:

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

- c. Create a CatalogSourceConfig object YAML file (for example, **eo-csc.yaml**) to enable the Elasticsearch Operator on the cluster.

Example CatalogSourceConfig

```
apiVersion: "operators.coreos.com/v1"
kind: "CatalogSourceConfig"
metadata:
  name: "elasticsearch"
  namespace: "openshift-marketplace"
spec:
  targetNamespace: "openshift-operators-redhat" 1
  packages: "elasticsearch-operator"
  source: "redhat-operators"
```

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

The Operator generates a CatalogSource from your CatalogSourceConfig in the namespace specified in **targetNamespace**.

- d. Create a CatalogSourceConfig object:

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

- e. Use the following command to get the **channel** value required for the next step.

```
$ oc get packagemanifest elasticsearch-operator -n openshift-marketplace -o
jsonpath='{.status.channels[].name}'

preview
```

- f. Create a Subscription object YAML file (for example, **eo-sub.yaml**) to subscribe a Namespace to an Operator.

Example Subscription

```

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

```

1 3 You must specify the **openshift-operators-redhat** namespace for **namespace** and **sourceNameSpace**.

2 Specify the **.status.channels[].name** value from the previous step.

g. Create the Subscription object:

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

h. Change to the **openshift-operators-redhat** project:

```
$ oc project openshift-operators-redhat

Now using project "openshift-operators-redhat"
```

i. Create a Role-based Access Control (RBAC) object file (for example, **eo-rbac.yaml**) to grant Prometheus permission to access the **openshift-operators-redhat** namespace:

```

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: prometheus-k8s
  namespace: openshift-operators-redhat
rules:
- apiGroups:
  - ""
  resources:
  - services
  - endpoints
  - pods
  verbs:
  - get
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: prometheus-k8s

```

```

namespace: openshift-operators-redhat
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: prometheus-k8s
subjects:
- kind: ServiceAccount
  name: prometheus-k8s
  namespace: openshift-operators-redhat

```

- j. Create the RBAC object:

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

The Elasticsearch operator is installed to each project in the cluster.

3. Install the Cluster Logging Operator using the OpenShift Container Platform web console for best results:
 - a. In the OpenShift Container Platform web console, click **Catalog** → **OperatorHub**.
 - b. Choose **Cluster Logging** from the list of available Operators, and click **Install**.
 - c. On the **Create Operator Subscription** page, under **A specific namespace on the cluster** select **openshift-logging**. Then, click **Subscribe**.
4. Verify the operator installations:
 - a. Switch to the **Catalog** → **Installed Operators** page.
 - b. Ensure that **Cluster Logging** is listed in the **openshift-logging** project with a **Status** of **InstallSucceeded**.
 - c. Ensure that **Elasticsearch Operator** is listed in the **openshift-operator-redhat** project with a **Status** of **InstallSucceeded**. The Elasticsearch Operator is copied to all other projects.



NOTE

During installation an operator might display a **Failed** status. If the operator then installs with an **InstallSucceeded** message, you can safely ignore the **Failed** message.

If either operator does not appear as installed, to troubleshoot further:

- Switch to the **Catalog** → **Operator Management** page and inspect the **Operator Subscriptions** and **Install Plans** tabs for any failure or errors under **Status**.
 - Switch to the **Workloads** → **Pods** page and check the logs in any Pods in the **openshift-logging** and **openshift-operators-redhat** projects that are reporting issues.
5. 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 **Cluster Loggings** page, click **Create Cluster Logging**. You might have to refresh the page to load the data.
- e. In the YAML, replace the code with the following:



NOTE

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

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging"
spec:
  managementState: "Managed" 2
  logStore:
    type: "elasticsearch" 3
    elasticsearch:
      nodeCount: 3 4
      storage:
        storageClassName: gp2 5
        size: 200G
        redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" 6
    kibana:
      replicas: 1
  curation:
    type: "curator" 7
    curator:
      schedule: "30 3 * * *"
  collection:
    logs:
      type: "fluentd" 8
      fluentd: {}

```

- 1 The name of the CR. This must be **instance**.
- 2 The cluster logging management state. In most cases, if you change the default cluster logging defaults, you must set this to **Unmanaged**. However, an unmanaged deployment does not receive updates until the cluster logging is placed back into a managed state. For more information, see **Changing cluster logging management state**.
- 3 Settings for configuring Elasticsearch. Using the CR, you can configure shard replication policy and persistent storage. For more information, see **Configuring Elasticsearch**.

- 4 Specify the number of Elasticsearch nodes. See the note that follows this list.
- 5 Specify that each Elasticsearch node in the cluster is bound to a Persistent Volume Claim.
- 6 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**.
- 7 Settings for configuring Curator. Using the CR, you can set the Curator schedule. For more information, see **Configuring Curator**.
- 8 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 0/1 1 0 6m54s
elasticsearch-cdm-x6kdekli-1 1/1 1 1 18h
elasticsearch-cdm-x6kdekli-2 0/1 1 0 6m49s
elasticsearch-cdm-x6kdekli-3 0/1 1 0 6m44s
```

- 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.
6. Verify the install:
- a. Switch to the **Workloads → Pods** page.
 - b. Select the **openshift-logging** project.
You should see several pods for cluster logging, Elasticsearch, Fluentd, and Kibana similar to the following list:
 - cluster-logging-operator-cb795f8dc-xkckc
 - elasticsearch-cdm-b3nqzchd-1-5c6797-67kfz
 - elasticsearch-cdm-b3nqzchd-2-6657f4-wtprv

- elasticsearch-cdm-b3nqzchd-3-588c65-clg7g
- fluentd-2c7dg
- fluentd-9z7kk
- fluentd-br7r2
- fluentd-fn2sb
- fluentd-pb2f8
- fluentd-zqgqx
- kibana-7fb4fd4cc9-bvt4p

3.2. ADDITIONAL RESOURCES

For more information on installing operators, see [Installing Operators from the OperatorHub](#).

CHAPTER 4. VIEWING THE KIBANA INTERFACE

The cluster logging installation deploys the Kibana interface.

4.1. LAUNCHING THE KIBANA INTERFACE

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

Procedure

To launch the Kibana interface:

1. In the OpenShift Container Platform console, click **Monitoring** → **Logging**.
2. Log in using the same credentials you use to log into 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 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.
- Set **TRANSFORM_EVENTS=true** in order to process and store event router events in Elasticsearch.
 - Set cluster logging to the unmanaged state.
 - Enable the **TRANSFORM_EVENTS** feature.

```
$ oc set env ds/fluentd TRANSFORM_EVENTS=true
```

Procedure

1. Create a template for the Event Router:

```
kind: Template
apiVersion: v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to cluster logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
  - kind: ServiceAccount 1
    apiVersion: v1
    metadata:
      name: eventrouter
      namespace: ${NAMESPACE}
  - kind: ClusterRole 2
    apiVersion: v1
    metadata:
      name: event-reader
    rules: 3
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "watch", "list"]
  - kind: ClusterRoleBinding 4
```

```

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 5
  displayName: Image
  value: "registry.redhat.io/openshift4/ose-logging-eventrouter:latest"
- name: MEMORY 6
  displayName: Memory
  value: "128Mi"
- name: CPU 7
  displayName: CPU
  value: "100m"
- name: NAMESPACE 8
  displayName: Namespace
  value: "openshift-logging"

```

- 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:

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

For example:

```

$ 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:

```
$ 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-
0ea949ad61f6","apiVersion":"operators.coreos.com/v1alpha1","resourceVersion":"523096"},"re
ason":"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. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT

6.1. ABOUT CONFIGURING CLUSTER LOGGING

After installing cluster logging into your 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](#).

6.1.1. 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.

6.1.1.1. 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:
          memory:
        requests:
          cpu: 1
          memory: 16Gi
      type: "elasticsearch"
  collection:
    logs:
      fluentd:
```

```

resources:
  limits:
    cpu:
    memory:
  requests:
    cpu:
    memory:
  type: "fluentd"
visualization:
  kibana:
    resources:
      limits:
        cpu:
        memory:
      requests:
        cpu:
        memory:
    type: kibana
curation:
  curator:
    resources:
      limits:
        memory: 200Mi
      requests:
        cpu: 200m
        memory: 200Mi
    type: "curator"

```

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.

```

spec:
  logStore:
    type: "elasticsearch"
  elasticsearch:
    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:
    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](<https://en.wikipedia.org/wiki/Cron>).

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

6.1.1.2. 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
    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"
curator:
  resources:
    limits:
      memory: 200Mi
    requests:
      cpu: 200m
      memory: 200Mi
  schedule: "*/5 * * * *"
collection:
  logs:
    type: "fluentd"
  fluentd:
    resources:
      limits:
        memory: 1Gi
      requests:
        cpu: 200m
        memory: 1Gi

```

6.1.2. 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:

```
$ oc edit ClusterLogging instance
```

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

```
....
```

```

spec:
  collection:
    logs:
      fluentd:
        resources: null
      rsyslog:
        resources: null
    type: fluentd
  curation:
    curator:
      nodeSelector: 1
        node-role.kubernetes.io/infra: "
      resources: null
      schedule: 30 3 * * *
    type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 2
        node-role.kubernetes.io/infra: "
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: 500m
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage: {}
    type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      nodeSelector: 3
        node-role.kubernetes.io/infra: " 4
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
  ....

```

- 1 2 3 4** 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.

6.2. 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.

In managed state, the Cluster Logging Operator (CLO) responds to changes in the Cluster Logging Custom Resource (CR) and attempts to update the cluster to match the CR.

The OpenShift Container Platform documentation indicates in a prerequisite step when you must set the 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.

6.2.1. 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:

```
$ oc edit ClusterLogging instance

$ oc edit ClusterLogging instance

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

....
```

```
spec:
  managementState: "Managed" 1
```

- 1 Specify the management state as **Managed** or **Unmanaged**.

6.2.2. 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:

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

Procedure

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

```
$ oc edit Elasticsearch elasticsearch

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

....

spec:
  managementState: "Managed" 1
```

- 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.

6.3. 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)

```

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
      rsyslog:
        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.
- You can specify specific nodes for the logging components using node selectors.

6.3.1. 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
```

```

ELASTICSEARCH_IMAGE=registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.1 1
FLUENTD_IMAGE=registry.redhat.io/openshift4/ose-logging-fluentd:v4.1 2
KIBANA_IMAGE=registry.redhat.io/openshift4/ose-logging-kibana5:v4.1 3
CURATOR_IMAGE=registry.redhat.io/openshift4/ose-logging-curator5:v4.1 4
OAUTH_PROXY_IMAGE=registry.redhat.io/openshift4/ose-oauth-proxy:v4.1 5

```

- 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.



NOTE

The values might be different depending on your environment.

6.4. CONFIGURING ELASTICSEARCH TO STORE AND ORGANIZE LOG DATA

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

You can configure your Elasticsearch deployment to:

- configure storage for your Elasticsearch cluster;
- define how shards are replicated across data nodes in the cluster, from full replication to no replication;
- configure 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 CPU limits, unless you specify otherwise in the ClusterLogging Custom Resource. The initial set of OpenShift Container Platform nodes might not be large enough to support the Elasticsearch cluster. You must add additional nodes to the OpenShift Container Platform cluster to run with the recommended or higher memory.

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

6.4.1. 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:

```
$ oc edit ClusterLogging instance
```

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

```

logStore:
  type: "elasticsearch"
  elasticsearch:
    resources: 1
    limits:
      memory: "16Gi"
    requests:
      cpu: "1"
      memory: "16Gi"

```

- 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.

6.4.2. 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:

```
oc edit clusterlogging instance
```

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
...
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      redundancyPolicy: "SingleRedundancy" 1

```

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

- **FullRedundancy.** Elasticsearch fully replicates the primary shards for each index to every data node. This provides the highest safety, but at the cost of the highest amount of disk required and the poorest performance.
- **MultipleRedundancy.** Elasticsearch fully replicates the primary shards for each index to half of the data nodes. This provides a good tradeoff between safety and performance.
- **SingleRedundancy.** Elasticsearch makes one copy of the primary shards for each index. Logs are always available and recoverable as long as at least two data nodes

exist. Better performance than `MultipleRedundancy`, when using 5 or more nodes. You cannot apply this policy on deployments of single Elasticsearch node.

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

6.4.3. Configuring Elasticsearch storage

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



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.

```
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.

6.4.4. 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:

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

6.4.5. 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:

```
$ 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

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

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

Prerequisites

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

Procedure

To expose Elasticsearch externally:

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

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

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

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

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

- a. Create a YAML file with the following:

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

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

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

- c. Create the route:

```
$ 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:

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

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

```
$ routeES=`oc get route elasticsearch -o jsonpath={.spec.host}`
```

- c. To verify the route was successfully created, run the following command that accesses Elasticsearch through the exposed route:

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

The response appears similar to the following:

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100  944    100  944    0    0    62    0 0:00:15 0:00:15 --:--:-- 204
{
  "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": "ODdiNWlyYzAtMjg5Ni0TAtNWE3MDY1MjMzNTc3",
        "_score": 1,
        "_source": {
          "_SOURCE_MONOTONIC_TIMESTAMP": "673396",
          "systemd": {
            "t": {
              "BOOT_ID": "246c34ee9cdeecb41a608e94",
              "MACHINE_ID": "e904a0bb5efd3e36badee0c",
              "TRANSPORT": "kernel"
            },
            "u": {
              "SYSLOG_FACILITY": "0",
              "SYSLOG_IDENTIFIER": "kernel"
            }
          }
        },
        "level": "info",
        "message": "acpiphp: Slot [30] registered",
        "hostname": "localhost.localdomain",
        "pipeline_metadata": {
          "collector": {
            "ipaddr4": "10.128.2.12",
            "ipaddr6": "fe80::xx:xxx:fe4c:5b09",
```

```

        "inputname": "fluent-plugin-systemd",
        "name": "fluentd",
        "received_at": "2019-03-15T20:25:06.273017+00:00",
        "version": "1.3.2 1.6.0"
    }
},
"@timestamp": "2019-03-15T20:00:13.808226+00:00",
"viaq_msg_id": "ODdiNWlyYzAtMYTAtNWE3MDY1MjMzNTc3"
}
}
]
}
}

```

6.4.6. About Elasticsearch alerting rules

You can view these alerting rules in Prometheus.

Alert	Description	Severity
ElasticsearchClusterNotHealthy	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.	critical
ElasticsearchClusterNotHealthy	Cluster health status has been YELLOW for at least 20m. Some shard replicas are not allocated.	warning
ElasticsearchBulkRequestsRejectionJumps	High Bulk Rejection Ratio at node in cluster. This node may not be keeping up with the indexing speed.	warning
ElasticsearchNodeDiskWatermarkReached	Disk Low Watermark Reached at node in cluster. Shards can not be allocated to this node anymore. You should consider adding more disk to the node.	alert
ElasticsearchNodeDiskWatermarkReached	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.	high
ElasticsearchJVMHeapUseHigh	JVM Heap usage on the node in cluster is <value>	alert
AggregatedLoggingSystemCPUHigh	System CPU usage on the node in cluster is <value>	alert
ElasticsearchProcessCPUHigh	ES process CPU usage on the node in cluster is <value>	alert

6.5. 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](#).

6.5.1. 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:

```
$ oc edit ClusterLogging instance

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
...
spec:
  visualization:
    type: "kibana"
    kibana:
      replicas:
      resources: 1
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    proxy: 2
      resources:
      limits:
        memory: 100Mi
      requests:
        cpu: 100m
        memory: 100Mi
```

- 1** Specify the CPU and memory limits to allocate for each node.
- 2** Specify the CPU and memory limits to allocate to the Kibana proxy.

6.5.2. 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:

```
$ oc edit ClusterLogging instance

$ oc edit ClusterLogging instance

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

....

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

- 1** Specify the number of Kibana nodes.

6.5.3. 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:

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

NAMESPACE      NAME      HOST/PORT
PATH  SERVICES      PORT  TERMINATION  WILDCARD
openshift-logging  kibana      kibana-openshift-logging.apps.openshift.com
kibana          <all> reencrypt/Redirect  None
```

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 1
```

- 1** 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>
```

6.6. 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 daily 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.

6.6.1. 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:

```
$ oc edit clusterlogging instance

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

curation:
  curator:
    schedule: 30 3 * * * 1
    type: curator
```

- 1 Specify the schedule for Curator in [cron format](#).



NOTE

The time zone is set based on the host node where the Curator pod runs.

6.6.2. 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 6.1. Project options

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

Table 6.2. Filter options

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

For example, to configure Curator to:

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

Use:

```
config.yaml: |
  .defaults:
    delete:
      days: 31

  .operations:
    delete:
      weeks: 8

  myapp-dev:
    delete:
```

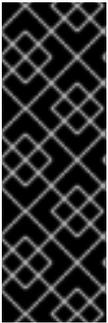
```

days: 1

myapp-qe:
  delete:
    weeks: 1

.regex:
- pattern: '^project\..+\-dev\..*$'
  delete:
    days: 1
- pattern: '^project\..+\-test\..*$'
  delete:
    days: 2

```



IMPORTANT

When you use **months** as the **\$UNIT** for an operation, Curator starts counting at the first day of the current month, not the current day of the current month. For example, if today is April 15, and you want to delete indices that are 2 months older than today (`delete: months: 2`), Curator does not delete indices that are dated older than February 15; it deletes indices older than February 1. That is, it goes back to the first day of the current month, then goes back two whole months from that date. If you want to be exact with Curator, it is best to use days (for example, `delete: days: 30`).

6.6.3. 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:

```

$ 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:

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

2. Use the following commands to control the CronJob:

- Suspend a CronJob:

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

- Resume a CronJob:

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

- Change a CronJob schedule:

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

- 1** The **schedule** option accepts schedules in [cron format](#).

6.6.4. 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:

```
$ 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.

```
$ 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.

6.6.5. 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.

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:
    action: delete_indices 1
    description: >-
      Delete .operations indices older than 30 days.
      Ignore the error if the filter does not
      result in an actionable list of indices (ignore_empty_list).
      See
      https://www.elastic.co/guide/en/elasticsearch/client/curator/5.2/ex_delete_indices.html
    options:
      # Swallow curator.exception.NoIndices exception
      ignore_empty_list: True
      # In seconds, default is 300
      timeout_override: ${CURATOR_TIMEOUT}
      # Don't swallow any other exceptions
      continue_if_exception: False
      # Optionally disable action, useful for debugging
      disable_action: False
      # All filters are bound by logical AND
    filters: 2
      - filtertype: pattern
        kind: regex
        value: '^\.operations\..*$'
        exclude: False 3
      - filtertype: age
        # Parse timestamp from index name
        source: name
        direction: older
        timestring: '%Y.%m.%d'
        unit: days
        unit_count: 30
        exclude: False
```

- 1 Specify **delete_indices** to delete the specified index.
- 2 Use the **filters** parameters to specify the index to be deleted. See the [Elastic Search curator documentation](#) for information on these parameters.
- 3 Specify **false** to allow the index to be deleted.

6.7. CONFIGURING FLUENTD

OpenShift Container Platform uses Fluentd to collect operations and application logs from your cluster which OpenShift Container Platform enriches with Kubernetes Pod and Namespace metadata.

You can configure log rotation, log location, use an external log aggregator, and make other 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](#).

6.7.1. Viewing Fluentd pods

You can use the **oc get pods -o wide** command to see the nodes where the Fluentd pod are deployed.

Procedure

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

```
$ oc get pods -o wide | grep fluentd
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
fluentd-5mr28	1/1	Running	0	4m56s	10.129.2.12	ip-10-0-164-233.ec2.internal
<none>						
fluentd-cnc4c	1/1	Running	0	4m56s	10.128.2.13	ip-10-0-155-142.ec2.internal
<none>						
fluentd-nlp8z	1/1	Running	0	4m56s	10.131.0.13	ip-10-0-138-77.ec2.internal
<none>						
fluentd-rknlk	1/1	Running	0	4m56s	10.128.0.33	ip-10-0-128-130.ec2.internal
<none>						
fluentd-rsm49	1/1	Running	0	4m56s	10.129.0.37	ip-10-0-163-191.ec2.internal
<none>						
fluentd-wjt8s	1/1	Running	0	4m56s	10.130.0.42	ip-10-0-156-251.ec2.internal
<none>						

6.7.2. Viewing Fluentd logs

How you view logs depends upon the **LOGGING_FILE_PATH** setting.

- 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:

```
$ oc exec <any-fluentd-pod> -- logs 1
```

- 1** Specify the name of a Fluentd pod. Note the space before **logs**.

For example:

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

To view the current setting:

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

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

```
$ oc logs -f <any-fluentd-pod> 1
```

- 1** Specify the name of a Fluentd pod. Use the **-f** option to follow what is being written into the logs.

For example

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

The contents of log files are printed out, starting with the oldest log.

6.7.3. Configure Fluentd 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:

```
$ oc edit ClusterLogging instance

$ oc edit ClusterLogging instance

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

....

spec:
  collection:
    logs:
      fluentd:
        resources:
          limits: 1
            cpu: 250m
            memory: 1Gi
          requests:
            cpu: 250m
            memory: 1Gi
```

- 1** Specify the CPU and memory limits as needed. The values shown are the default values.

6.7.4. Configuring Fluentd log location

Fluentd 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.

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**:

```
spec:
  template:
    spec:
      containers:
        env:
          - name: LOGGING_FILE_PATH
            value: console ❶
LOGGING_FILE_PATH= ❷
```

- ❶ 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:

```
oc -n openshift-logging set env daemonset/fluentd LOGGING_FILE_PATH=console
```

6.7.5. Throttling Fluentd logs

For projects that are especially verbose, an administrator can throttle down the rate at which the logs are read in by Fluentd before being processed. By throttling, you deliberately slow down the rate at which you are reading logs, so Kibana might take longer to display records.



WARNING

Throttling can contribute to log aggregation falling behind for the configured projects; log entries can be lost if a pod is deleted before Fluentd catches up.



NOTE

Throttling does not work when using the systemd journal as the log source. The throttling implementation depends on being able to throttle the reading of the individual log files for each project. When reading from the journal, there is only a single log source, no log files, so no file-based throttling is available. There is not a method of restricting the log entries that are read into the Fluentd process.

Prerequisite

Set cluster logging to the unmanaged state.

Procedure

1. To configure Fluentd to restrict specific projects, edit the throttle configuration in the Fluentd ConfigMap after deployment:

```
$ oc edit configmap/fluentd
```

The format of the *throttle-config.yaml* key is a YAML file that contains project names and the desired rate at which logs are read in on each node. The default is 1000 lines at a time per node. For example:

```
throttle-config.yaml: |
- opensift-logging:
  read_lines_limit: 10
- .operations:
  read_lines_limit: 100
```

6.7.6. 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*.

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 $(\text{FILE_BUFFER_LIMIT} / (\text{number_of_outputs} * \text{BUFFER_SIZE_LIMIT}))$.

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
- **4** - 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

6.7.7. Configuring Fluentd JSON parsing

You can configure Fluentd to inspect each log message to determine if the message is in **JSON** format and merge the message into the JSON payload document posted to Elasticsearch. This feature is disabled by default.

You can enable or disable this feature by editing the `MERGE_JSON_LOG` environment variable in the `fluentd` daemonset.



IMPORTANT

Enabling this feature comes with risks, including:

- Possible log loss due to Elasticsearch rejecting documents due to inconsistent type mappings.
- Potential buffer storage leak caused by rejected message cycling.
- Overwrite of data for field with same names.

The features in this topic should be used by only experienced Fluentd and Elasticsearch users.

Prerequisites

Set cluster logging to the unmanaged state.

Procedure

Use the following command to enable this feature:

```
oc set env ds/fluentd MERGE_JSON_LOG=true 1
```

1 Set this to **false** to disable this feature or **true** to enable this feature.

Setting MERGE_JSON_LOG and CDM_UNDEFINED_TO_STRING

If you set the **MERGE_JSON_LOG** and **CDM_UNDEFINED_TO_STRING** environment variables to **true**, you might receive an Elasticsearch **400** error. The error occurs because when `MERGE_JSON_LOG=true`, Fluentd adds fields with data types other than **string**. When you set **CDM_UNDEFINED_TO_STRING=true**, Fluentd attempts to add those fields as a **string** value resulting in the Elasticsearch **400** error. The error clears when the indices roll over for the next day.

When Fluentd rolls over the indices for the next day's logs, it will create a brand new index. The field definitions are updated and you will not get the **400** error.

Records that have **hard** errors, such as schema violations, corrupted data, and so forth, cannot be retried. Fluent sends the records for error handling. If you add a **<label @ERROR>** section to your Fluentd config, as the last **<label>**, you can handle these records as needed.

For example:

```
data:
  fluent.conf:
  ....
  <label @ERROR>
  <match **>
    @type file
    path /var/log/fluent/dlq
    time_slice_format %Y%m%d
    time_slice_wait 10m
    time_format %Y%m%dT%H%M%S%z
    compress gzip
  </match>
</label>
```

This section writes error records to the [Elasticsearch dead letter queue \(DLQ\) file](#). See [the fluentd documentation](#) for more information about the file output.

Then you can edit the file to clean up the records manually, edit the file to use with the Elasticsearch **/_bulk index** API and use cURL to add those records. For more information on Elasticsearch Bulk API, see [the Elasticsearch documentation](#).

6.7.8. Configuring how the log collector normalizes logs

Cluster Logging uses a specific data model, like a database schema, to store log records and their metadata in the logging store. There are some restrictions on the data:

- There must be a **"message"** field containing the actual log message.
- There must be a **"@timestamp"** field containing the log record timestamp in RFC 3339 format, preferably millisecond or better resolution.

- There must be a **"level"** field with the log level, such as **err**, **info**, **unknown**, and so forth.



NOTE

For more information on the data model, see [Exported Fields](#).

Because of these requirements, conflicts and inconsistencies can arise with log data collected from different subsystems.

For example, if you use the **MERGE_JSON_LOG** feature (**MERGE_JSON_LOG=true**), it can be extremely useful to have your applications log their output in JSON, and have the log collector automatically parse and index the data in Elasticsearch. However, this leads to several problems, including:

- field names can be empty, or contain characters that are illegal in Elasticsearch;
- different applications in the same namespace might output the same field name with different value data types;
- applications might emit too many fields;
- fields may conflict with the cluster logging built-in fields.

You can configure how cluster logging treats fields from disparate sources by editing the log collector daemonset, Fluentd or Rsyslog, and setting environment variables in the table below.

- **Undefined fields.** One of the problems with log data from disparate systems is that some fields might be unknown to the ViaQ data model. Such fields are called *undefined*. ViaQ requires all top-level fields to be defined and described. Use the parameters to configure how OpenShift Container Platform moves any undefined fields under a top-level field called **undefined** to avoid conflicting with the *well known* ViaQ top-level fields. You can add undefined fields to the top-level fields and move others to an **undefined** container.

You can also replace special characters in undefined fields and convert undefined fields to their JSON string representation. Converting to JSON string preserves the structure of the value, so that you can retrieve the value later and convert it back to a map or an array.

- Simple scalar values like numbers and booleans are changed to a quoted string. For example: **10** becomes **"10"**, **3.1415** becomes **"3.1415"**, **false** becomes **"false"**.
- Map/dict values and array values are converted to their JSON string representation: **"mapfield":{"key":"value"}** becomes **"mapfield":{"key":"value\"}"** and **"arrayfield":[1,2,"three"]** becomes **"arrayfield":["1,2,\"three\"]"**.
- **Defined fields.** You can also configure which defined fields appear in the top levels of the logs. The default top-level fields, defined through the **CDM_DEFAULT_KEEP_FIELDS** parameter, are **CEE**, **time**, **@timestamp**, **aushape**, **ci_job**, **collectd**, **docker**, **fedora-ci**, **file**, **foreman**, **geoip**, **hostname**, **ipaddr4**, **ipaddr6**, **kubernetes**, **level**, **message**, **namespace_name**, **namespace_uuid**, **offset**, **openstack**, **ovirt**, **pid**, **pipeline_metadata**, **rsyslog**, **service**, **systemd**, **tags**, **testcase**, **tlog**, **viaq_msg_id**.

Any fields not included in **\${CDM_DEFAULT_KEEP_FIELDS}** or **\${CDM_EXTRA_KEEP_FIELDS}** are moved to **\${CDM_UNDEFINED_NAME}** if **CDM_USE_UNDEFINED** is **true**.

**NOTE**

The **CDM_DEFAULT_KEEP_FIELDS** parameter is for only advanced users, or if you are instructed to do so by Red Hat support.

- Empty fields. You can determine which empty fields to retain from disparate logs.

Table 6.3. Environment parameters for log normalization

Parameters	Definition	Example
CDM_EXTRA_KEEP_FIELDS	Specify an extra set of defined fields to be kept at the top level of the logs in addition to the CDM_DEFAULT_KEEP_FIELDS . The default is "".	CDM_EXTRA_KEEP_FIELDS="broker"
CDM_KEEP_EMPTY_FIELDS	Specify fields to retain even if empty in CSV format. Empty defined fields not specified are dropped. The default is "message", keep empty messages.	CDM_KEEP_EMPTY_FIELDS="message"
CDM_USE_UNDEFINED	Set to true to move undefined fields to the undefined top level field. The default is false . If true , values in CDM_DEFAULT_KEEP_FIELDS and CDM_EXTRA_KEEP_FIELDS are not moved to undefined .	CDM_USE_UNDEFINED=true
CDM_UNDEFINED_NAME	Specify a name for the undefined top level field if using CDM_USE_UNDEFINED . The default is `undefined`. Enabled only when CDM_USE_UNDEFINED is true .	CDM_UNDEFINED_NAME="undef"
CDM_UNDEFINED_MAX_NUM_FIELDS	If the number of undefined fields is greater than this number, all undefined fields are converted to their JSON string representation and stored in the CDM_UNDEFINED_NAME field. If the record contains more than this value of undefined fields, no further processing takes place on these fields. Instead, the fields will be converted to a single string JSON value, stored in the top-level CDM_UNDEFINED_NAME field. Keeping the default of -1 allows for an unlimited number of undefined fields, which is not recommended.	CDM_UNDEFINED_MAX_NUM_FIELDS=4
	 <p>NOTE</p> <p>This parameter is honored even if CDM_USE_UNDEFINED is false.</p>	
CDM_UNDEFINED_TO_STRING	Set to true to convert all undefined fields to their JSON string representation. The default is false .	CDM_UNDEFINED_TO_STRING=true

Parameters	Definition	Example
CDM_UNDEFINED_DOT_REPLACE_CHAR	Specify a character to use in place of a dot character '.' in an undefined field. MERGE_JSON_LOG must be true . The default is UNUSED . If you set the MERGE_JSON_LOG parameter to true , see the Note below.	CDM_UNDEFINED_DOT_REPLACE_CHAR="_"



NOTE

If you set the **MERGE_JSON_LOG** parameter in the log collector daemonset and **CDM_UNDEFINED_TO_STRING** environment variables to true, you might receive an Elasticsearch **400** error. The error occurs because when `MERGE_JSON_LOG=true`, the log collector adds fields with data types other than string. When you set **CDM_UNDEFINED_TO_STRING=true**, the log collector attempts to add those fields as a string value resulting in the Elasticsearch **400** error. The error clears when the log collector rolls over the indices for the next day's logs.

When the log collector rolls over the indices, it creates a brand new index. The field definitions are updated and you will not get the **400** error.

Procedure

Use the **CDM_*** parameters to configure undefined and empty field processing.

1. Configure how to process fields, as needed:
 - a. Specify the fields to move using **CDM_EXTRA_KEEP_FIELDS**.
 - b. Specify any empty fields to retain in the **CDM_KEEP_EMPTY_FIELDS** parameter in CSV format.
2. Configure how to process undefined fields, as needed:
 - a. Set **CDM_USE_UNDEFINED** to **true** to move undefined fields to the top-level **undefined** field:
 - b. Specify a name for the undefined fields using the **CDM_UNDEFINED_NAME** parameter.
 - c. Set **CDM_UNDEFINED_MAX_NUM_FIELDS** to a value other than the default **-1**, to set an upper bound on the number of undefined fields in a single record.
3. Specify **CDM_UNDEFINED_DOT_REPLACE_CHAR** to change any dot . characters in an undefined field name to another character. For example, if **CDM_UNDEFINED_DOT_REPLACE_CHAR=@@@" and there is a field named **foo.bar.baz** the field is transformed into **foo@@@bar@@@baz**.**
4. Set **UNDEFINED_TO_STRING** to **true** to convert undefined fields to their JSON string representation.



NOTE

If you configure the **CDM_UNDEFINED_TO_STRING** or **CDM_UNDEFINED_MAX_NUM_FIELDS** parameters, you use the **CDM_UNDEFINED_NAME** to change the undefined field name. This field is needed because **CDM_UNDEFINED_TO_STRING** or **CDM_UNDEFINED_MAX_NUM_FIELDS** could change the value type of the undefined field. When **CDM_UNDEFINED_TO_STRING** or **CDM_UNDEFINED_MAX_NUM_FIELDS** is set to true and there are more undefined fields in a log, the value type becomes **string**. Elasticsearch stops accepting records if the value type is changed, for example, from JSON to JSON string.

For example, when **CDM_UNDEFINED_TO_STRING** is **false** or **CDM_UNDEFINED_MAX_NUM_FIELDS** is the default, **-1**, the value type of the undefined field is **json**. If you change **CDM_UNDEFINED_MAX_NUM_FIELDS** to a value other than default and there are more undefined fields in a log, the value type becomes **string** (json string). Elasticsearch stops accepting records if the value type is changed.

6.7.9. Configuring Fluentd using environment variables

You can use [environment variables](#) to modify your Fluentd configuration.

Prerequisite

Set cluster logging to the unmanaged state.

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
```

6.8. SENDING OPENSIFT CONTAINER PLATFORM LOGS TO EXTERNAL DEVICES

You can send Elasticsearch logs to external devices, such as an externally-hosted Elasticsearch instance or an external syslog server. You can also configure Fluentd to send logs to an external log aggregator.



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](#).

6.8.1. Configuring Fluentd to send logs to an external Elasticsearch instance

Fluentd sends logs to the value of the **ES_HOST**, **ES_PORT**, **OPS_HOST**, and **OPS_PORT** environment variables of the Elasticsearch deployment configuration. The application logs are directed to the **ES_HOST** destination, and operations logs to **OPS_HOST**.



NOTE

Sending logs directly to an AWS Elasticsearch instance is not supported. Use Fluentd Secure Forward to direct logs to an instance of Fluentd that you control and that is configured with the **fluent-plugin-aws-elasticsearch-service** plug-in.

Prerequisite

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

Procedure

To direct logs to a specific Elasticsearch instance:

1. Edit the **fluentd** DaemonSet in the **openshift-logging** project:

```
$ oc edit ds/fluentd

spec:
  template:
    spec:
      containers:
        env:
          - name: ES_HOST
            value: elasticsearch
          - name: ES_PORT
            value: '9200'
          - name: ES_CLIENT_CERT
            value: /etc/fluent/keys/app-cert
          - name: ES_CLIENT_KEY
            value: /etc/fluent/keys/app-key
          - name: ES_CA
            value: /etc/fluent/keys/app-ca
          - name: OPS_HOST
            value: elasticsearch
          - name: OPS_PORT
            value: '9200'
          - name: OPS_CLIENT_CERT
            value: /etc/fluent/keys/infra-cert
          - name: OPS_CLIENT_KEY
            value: /etc/fluent/keys/infra-key
          - name: OPS_CA
            value: /etc/fluent/keys/infra-ca
```

2. Set **ES_HOST** and **OPS_HOST** to the same destination, while ensuring that **ES_PORT** and **OPS_PORT** also have the same value for an external Elasticsearch instance to contain both application and operations logs.
3. Configure your externally-hosted Elasticsearch instance for TLS. Only externally-hosted Elasticsearch instances that use Mutual TLS are allowed.

**NOTE**

If you are not using the provided Kibana and Elasticsearch images, you will not have the same multi-tenant capabilities and your data will not be restricted by user access to a particular project.

6.8.2. Configuring Fluentd to send logs to an external syslog server

Use the **fluent-plugin-remote-syslog** plug-in on the host to send logs to an external syslog server.

Prerequisite

Set cluster logging to the unmanaged state.

Procedure

1. Set environment variables in the **fluentd** daemonset in the **openshift-logging** project:

```
spec:
  template:
    spec:
      containers:
      - name: fluentd
        image: 'registry.redhat.io/openshift4/ose-logging-fluentd:v4.1'
        env:
        - name: REMOTE_SYSLOG_HOST 1
          value: host1
        - name: REMOTE_SYSLOG_HOST_BACKUP
          value: host2
        - name: REMOTE_SYSLOG_PORT_BACKUP
          value: 5555
```

- 1** The desired remote syslog host. Required for each host.

This will build two destinations. The syslog server on **host1** will be receiving messages on the default port of **514**, while **host2** will be receiving the same messages on port **5555**.

2. Alternatively, you can configure your own custom the **fluentd** daemonset in the **openshift-logging** project.

Fluentd Environment Variables

Parameter	Description
USE_REMOTE_SYSLOG	Defaults to false . Set to true to enable use of the fluent-plugin-remote-syslog gem
REMOTE_SYSLOG_HOST	(Required) Hostname or IP address of the remote syslog server.
REMOTE_SYSLOG_PORT	Port number to connect on. Defaults to 514 .

Parameter	Description
REMOTE_SYSLOG_SEVERITY	Set the syslog severity level. Defaults to debug .
REMOTE_SYSLOG_FACILITY	Set the syslog facility. Defaults to local0 .
REMOTE_SYSLOG_USE_RECORD	Defaults to false . Set to true to use the record's severity and facility fields to set on the syslog message.
REMOTE_SYSLOG_REMOVE_TAG_PREFIX	Removes the prefix from the tag, defaults to "" (empty).
REMOTE_SYSLOG_TAG_KEY	If specified, uses this field as the key to look on the record, to set the tag on the syslog message.
REMOTE_SYSLOG_PAYLOAD_KEY	If specified, uses this field as the key to look on the record, to set the payload on the syslog message.
REMOTE_SYSLOG_TYPE	Set the transport layer protocol type. Defaults to syslog_buffered , which sets the TCP protocol. To switch to UDP, set this to syslog .

**WARNING**

This implementation is insecure, and should only be used in environments where you can guarantee no snooping on the connection.

6.8.3. Configuring Fluentd to send logs to an external log aggregator

You can configure Fluentd to send a copy of its logs to an external log aggregator, and not the default Elasticsearch, using the **out_forward** plug-in. From there, you can further process log records after the locally hosted Fluentd has processed them.

The **forward** plug-in is supported by Fluentd only. The **out_forward** plug-in implements the client side (sender) and the **in_forward** plug-in implements the server side (receiver).

To configure OpenShift Container Platform to send logs using **out_forward**, create a ConfigMap called **secure-forward** in the **openshift-logging** namespace that points to a receiver. On the receiver, configure the **in_forward** plug-in to receive the logs from OpenShift Container Platform. For more information on using the **in_forward** plug-in, see the [Fluentd documentation](#).

Default **secure-forward.conf** section

```
# <store>
```

```

# @type forward
# <security>
# self_hostname ${hostname} # ${hostname} is a placeholder.
# shared_key <shared_key_between_forwarder_and_forwardee>
# </security>
# transport tls
# tls_verify_hostname true      # Set false to ignore server cert hostname.

# tls_cert_path /path/for/certificate/ca_cert.pem
# <buffer>
# @type file
# path '/var/lib/fluentd/forward'
# 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 server.fqdn.example.com # or IP
# port 24284
# </server>
# <server>
# host 203.0.113.8 # ip address to connect
# name server.fqdn.example.com # The name of the server. Used for logging and certificate
# verification in TLS transport (when host is address).
# </server>
# </store>

```

Procedure

To send a copy of Fluentd logs to an external log aggregator:

1. Edit the **secure-forward.conf** section of the Fluentd configuration map:

```
$ oc edit configmap/fluentd -n openshift-logging
```

2. Enter the name, host, and port for your external Fluentd server:

```

# <server>
# host server.fqdn.example.com # or IP
# port 24284
# </server>
# <server>
# host 203.0.113.8 # ip address to connect
# name server.fqdn.example.com # The name of the server. Used for logging and
# certificate verification in TLS transport (when host is address).
# </server>

```

For example:

```

<server>
  name externalserver1 ❶
  host 192.168.1.1 ❷
  port 24224 ❸
</server>
<server> ❹
  name externalserver1
  host 192.168.1.2
  port 24224
</server>
</store>

```

- ❶ Optionally, enter a name for this external aggregator.
- ❷ Specify the host name or IP of the external aggregator.
- ❸ Specify the port of the external aggregator.
- ❹ Optionally, add additional external aggregator.

3. Add the path to your CA certificate and private key to the **secure-forward.conf** section:

```

# <security>
# self_hostname ${hostname} # ${hostname} is a placeholder. ❶
# shared_key <shared_key_between_forwarder_and_forwardee> ❷
# </security>

# tls_cert_path /path/for/certificate/ca_cert.pem ❸

```

- ❶ Specify the default value of the auto-generated certificate common name (CN).
- ❷ Specify a shared key for authentication.
- ❸ Specify the path to your CA certificate.

For example:

```

<security>
  self_hostname client.fqdn.local
  shared_key cluster_logging_key
</security>

tls_cert_path /etc/fluent/keys/ca.crt

```

To use mTLS, see the [Fluentd documentation](#) for information about client certificate and key parameters and other settings.

4. Add certificates to be used in **secure-forward.conf** to the existing secret that is mounted on the Fluentd pods. The **your_ca_cert** and **your_private_key** values must match what is specified in **secure-forward.conf** in **configmap/logging-fluentd**:

```

$ oc patch secrets/fluentd --type=json \
  --patch "[{'op':'add','path':'/data/your_ca_cert','value':$(base64 -w0

```

```
/path/to/your_ca_cert.pem)']}]"
```

```
$ oc patch secrets/fluentd --type=json \
  --patch "[{'op':'add','path':'/data/your_private_key','value':$(base64 -w0
/path/to/your_private_key.pem)'}]"
```



NOTE

Replace **your_private_key** with a generic name. This is a link to the JSON path, not a path on your host system.

For example:

```
$ oc patch secrets/fluentd --type=json \
  --patch "[{'op':'add','path':'/data/ca.crt','value':$(base64 -w0 /etc/fluent/keys/ca.crt)'}]"
```

```
$ oc patch secrets/fluentd --type=json \
  --patch "[{'op':'add','path':'/data/ext-agg','value':$(base64 -w0 /etc/fluent/keys/ext-agg.pem)'}]"
```

5. Configure the **secure-forward.conf** file on the external aggregator to accept messages securely from Fluentd.

When configuring the external aggregator, it must be able to accept messages securely from Fluentd.

You can find further explanation of [how to set up the inforward plugin](#) and [the out_forward plugin](#).

6.9. CONFIGURING SYSTEMD-JOURNALD AND RSYSLOG

Because Fluentd and rsyslog read 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.

6.9.1. Configuring systemd-journald for cluster logging

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

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

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

Procedure

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

```
Compress=no 1
ForwardToConsole=yes 2
ForwardToSyslog=no 3
MaxRetentionSec=30s 4
RateLimitBurst=10000 5
```

```

RateLimitInterval=1s 6
Storage=volatile 7
SyncIntervalSec=1s 8
SystemMaxUse=8g 9
SystemKeepFree=20% 10
SystemMaxFileSize10M 11

```

- 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** **3** 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 digits to specify seconds. Or include a unit: "year", "month", "week", "day", "h" or "m". Enter **0** to disable. The default is **1month**.
- 5** **6** 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.
- 7** 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.
- 8** 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**.
- 9** Specify the maximum size the journal can use. The default is **8g**.
- 10** Specify how much disk space `systemd` must leave free. The default is **20%**.
- 11** Specify the maximum size for individual journal files stored persistently in `/var/log/journal`. The default is **10M**.

**NOTE**

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

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

- Convert the **journal.conf** file to base64:

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

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

```
...
config:
  storage:
    files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,{jrnl_cnf}
          verification: {}
        filesystem: root
        mode: 0644 ①
        path: /etc/systemd/journald.conf ②
      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.

- Create the MachineConfig:

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

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

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

```
$ oc describe machineconfigpool/worker

Name:      worker
Namespace:
Labels:    machineconfiguration.openshift.io/mco-built-in=
Annotations: <none>
API Version: machineconfiguration.openshift.io/v1
Kind:      MachineConfigPool
```

...

Conditions:

Message:

Reason: All nodes are updating to rendered-worker-
913514517bcea7c93bd446f4830bc64e

CHAPTER 7. VIEWING ELASTICSEARCH STATUS

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

7.1. 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
clusterHealth: green 2
conditions: 3
.....
nodes: 4
.....
pods: 5
.....
shardAllocationEnabled: "All" 6
```

- 1** In the output, the cluster status fields appear in the **status** stanza.
- 2** The status of the Elasticsearch cluster, **green, red, yellow**.
- 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

reasons from the controller. 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**.

- 4 The Elasticsearch nodes in the cluster, with **upgradeStatus**.
- 5 The Elasticsearch client, data, and master pods in the cluster, listed under 'failed', **notReady** or **ready** state.

7.1.1. Example condition messages

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

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

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

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

```
status:
  nodes:
  - conditions:
    - lastTransitionTime: 2019-03-15T16:04:45Z
      message: Disk storage usage for node is 27.5gb (36.74%). Shards will be relocated
        from this node.
      reason: Disk Watermark High
      status: "True"
      type: NodeStorage
      deploymentName: example-elasticsearch-cdm-0-1
      upgradeStatus: {}
```

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

```
status:
  nodes:
```

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

This status message indicates that the 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
```

7.2. 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-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. Get the status of the indices:

```
$ oc exec elasticsearch-cdm-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
health status index                uuid                pri rep docs.count
docs.deleted store.size pri.store.size
red open .kibana.647a750f1787408bf50088234ec0edd5a6a9b2ac
N7iCbRjSSc2bGhn8Cpc7Jg 2 1
green open .operations.2019.04.10          GTewEJEzQjaus9QjvBBnGg 3 1
2176114 0 3929 1956
green open .operations.2019.04.11          ausZHoKxTNOoBvv9RIXfrw 3 1
1494624 0 2947 1475
green open .kibana                      9Fltn1D0QHSnFMXpphZ--Q 1 1 1
0 0 0
green open .searchguard                  chOwDnQISsqhfSPcot1Yiw 1 1
5 1 0 0
```

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-
o://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:v4.1
    Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
period=5s #success=1 #failure=3

....

Conditions:

```

```

Type      Status Reason
----      -
Progressing Unknown DeploymentPaused
Available  True   MinimumReplicasAvailable
....
Events:    <none>

```

Elasticsearch ReplicaSet

You can view the status of the Elasticsearch ReplicaSet.

1. Get the name of a replica set:

```

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

replicaset.extensions/elasticsearch-cdm-1gon-1-6f8495
replicaset.extensions/elasticsearch-cdm-1gon-2-5769cf
replicaset.extensions/elasticsearch-cdm-1gon-3-f66f7d

```

2. Get the status of the replica set:

```

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

```

The output includes the following status information:

```

....
Containers:
  elasticsearch:
    Image:   registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.1
    Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
              period=5s #success=1 #failure=3
....
Events:     <none>

```

CHAPTER 8. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS

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

8.1. SPECIFYING A NODE FOR CLUSTER LOGGING COMPONENTS USING NODE SELECTORS

Each component specification allows the component to target a specific node.

Prerequisites

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

Procedure

1. Add the desired label to your nodes:

```
$ oc label <resource> <name> <key>=<value>
```

For example, to label a node:

```
$ oc label nodes ip-10-0-142-25.ec2.internal type=elasticsearch
```

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

```
$ oc edit ClusterLogging instance

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "nodeselector"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeSelector: 1
        logging: es
      nodeCount: 1
    resources:
      limits:
        memory: 2Gi
      requests:
        cpu: 200m
        memory: 2Gi
    storage:
      size: "20G"
      storageClassName: "gp2"
      redundancyPolicy: "ZeroRedundancy"
  visualization:
    type: "kibana"
  kibana:
```

```
nodeSelector: 2
  logging: kibana
  replicas: 1
curation:
  type: "curator"
  curator:
    nodeSelector: 3
    logging: curator
    schedule: "*/10 * * * *"
collection:
  logs:
    type: "fluentd"
    fluentd:
      nodeSelector: 4
      logging: fluentd
```

- 1 Node selector for Elasticsearch.
- 2 Node selector for Kibana.
- 3 Node selector for Curator.
- 4 Node selector for Fluentd.

CHAPTER 9. ADDITIONAL RESOURCES

- For more information on node selectors, see: [Placing pods on specific nodes using node selectors](#).

CHAPTER 10. 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.

10.1. 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:

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

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

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

```
$ 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:

```
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:

```
$ 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:

-

```
$ 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" : "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:

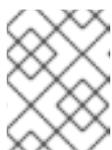
```
$ oc rollout pause deployment/<deployment-name>
```

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:

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

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

1 Make sure this parameter 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:

```
$ 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:

```
$ 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 11. 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.

11.1. 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
```

11.2. 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
```

11.3. 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 12. EXPORTED FIELDS

These are the fields exported by the logging system and available for searching from Elasticsearch and Kibana. Use the full, dotted field name when searching. For example, for an Elasticsearch `/_search` URL, to look for a Kubernetes Pod name, use `/_search/q=kubernetes.pod_name:name-of-my-pod`.

The following sections describe fields that may not be present in your logging store. Not all of these fields are present in every record. The fields are grouped in the following categories:

- **exported-fields-Default**
- **exported-fields-rsyslog**
- **exported-fields-systemd**
- **exported-fields-kubernetes**
- **exported-fields-pipeline_metadata**
- **exported-fields-ovirt**
- **exported-fields-aushape**
- **exported-fields-tlog**

12.1. DEFAULT EXPORTED FIELDS

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

Top Level Fields

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

Parameter	Description
@timestamp	The UTC value marking when the log payload was created, or when the log payload was first collected if the creation time is not known. This is the log processing pipeline's best effort determination of when the log payload was generated. Add the @ prefix convention to note a field as being reserved for a particular use. With Elasticsearch, most tools look for @timestamp by default. For example, the format would be 2015-01-24 14:06:05.071000.
geoip	This is geo-ip of the machine.
hostname	The hostname is the fully qualified domain name (FQDN) of the entity generating the original payload. This field is an attempt to derive this context. Sometimes the entity generating it knows the context. While other times that entity has a restricted namespace itself, which is known by the collector or normalizer.
ipaddr4	The IP address V4 of the source server, which can be an array.

Parameter	Description
ipaddr6	The IP address V6 of the source server, if available.
level	<p>The logging level as provided by rsyslog (<code>severitytext</code> property), python's logging module. Possible values are as listed at misc/sys/syslog.h plus trace and unknown. For example, <i>alert crit debug emerg err info notice trace unknown warning</i>. Note that trace is not in the syslog.h list but many applications use it.</p> <ul style="list-style-type: none"> * You should only use unknown when the logging system gets a value it does not understand, and note that it is the highest level. * Consider trace as higher or more verbose, than debug. * error is deprecated, use err. * Convert panic to emerg. * Convert warn to warning. <p>Numeric values from syslog/journal PRIORITY can usually be mapped using the priority values as listed at misc/sys/syslog.h.</p> <p>Log levels and priorities from other logging systems should be mapped to the nearest match. See python logging for an example.</p>
message	A typical log entry message, or payload. It can be stripped of metadata pulled out of it by the collector or normalizer, that is UTF-8 encoded.
pid	This is the process ID of the logging entity, if available.
service	The name of the service associated with the logging entity, if available. For example, the syslog APP-NAME and rsyslog programname property are mapped to the service field.
tags	Optionally provided operator defined list of tags placed on each log by the collector or normalizer. The payload can be a string with whitespace-delimited string tokens, or a JSON list of string tokens.
file	Optional path to the file containing the log entry local to the collector TODO analyzer for file paths.
offset	The offset value can represent bytes to the start of the log line in the file (zero or one based), or log line numbers (zero or one based), as long as the values are strictly monotonically increasing in the context of a single log file. The values are allowed to wrap, representing a new version of the log file (rotation).

Parameter	Description
namespace_name	Associate this record with the namespace that shares it's name. This value will not be stored, but it is used to associate the record with the appropriate namespace for access control and visualization. Normally this value will be given in the tag, but if the protocol does not support sending a tag, this field can be used. If this field is present, it will override the namespace given in the tag or in kubernetes.namespace_name .
namespace_uuid	This is the uuid associated with the namespace_name . This value will not be stored, but is used to associate the record with the appropriate namespace for access control and visualization. If this field is present, it will override the uuid given in kubernetes.namespace_uuid . This will also cause the Kubernetes metadata lookup to be skipped for this log record.

collectd Fields

The following fields represent namespace metrics metadata.

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

collectd.processes Fields

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

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

collectd.processes.ps_disk_ops Fields

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

Parameter	Description
collectd.processes.ps_disk_ops.read	type: float TODO
collectd.processes.ps_disk_ops.write	type: float TODO
collectd.processes.ps_vm	type: integer The collectd ps_vm type of processes plug-in.
collectd.processes.ps_rss	type: integer The collectd ps_rss type of processes plug-in.
collectd.processes.ps_data	type: integer The collectd ps_data type of processes plug-in.
collectd.processes.ps_code	type: integer The collectd ps_code type of processes plug-in.
collectd.processes.ps_stacksize	type: integer The collectd ps_stacksize type of processes plug-in.

collectd.processes.ps_cputime Fields

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

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

Parameter	Description
collectd.processes.ps_cpu.uptime.syst	type: float TODO

collectd.processes.ps_count Fields

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

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

collectd.processes.ps_pagefaults Fields

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

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

collectd.processes.ps_disk_octets Fields

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

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

collectd.disk Fields

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

collectd.disk.disk_merged Fields

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

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

collectd.disk.disk_octets Fields

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

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

collectd.disk.disk_time Fields

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

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

collectd.disk.disk_ops Fields

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

Parameter	Description
-----------	-------------

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

collectd.disk.disk_io_time Fields

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

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

collectd.interface Fields

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

collectd.interface.if_octets Fields

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

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

collectd.interface.if_packets Fields

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

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

collectd.interface.if_errors Fields

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

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

collectd.interface.if_dropped Fields

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

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

collectd.virt Fields

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

collectd.virt.if_octets Fields

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

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

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

collectd.virt.if_packets Fields

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

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

collectd.virt.if_errors Fields

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

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

collectd.virt.if_dropped Fields

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

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

collectd.virt.disk_ops Fields

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

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

collectd.virt.disk_octets Fields

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

Parameter	Description
collectd.virt.disk_octets.read	type: float TODO
collectd.virt.disk_octets.write	type: float TODO
collectd.virt.memory	type: float The collectd memory type of virt plug-in.
collectd.virt.virt_vcpu	type: float The collectd virt_vcpu type of virt plug-in.
collectd.virt.virt_cpu_total	type: float The collectd virt_cpu_total type of virt plug-in.

collectd.CPU Fields

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

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

collectd.df Fields

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

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

collectd.entropy Fields

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

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

collectd.memory Fields

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

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

collectd.swap Fields

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

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

collectd.load Fields

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

collectd.load.load Fields

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

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

collectd.aggregation Fields

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

Parameter	Description
collectd.aggregation.percent	type: float TODO

collectd.statsd Fields

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

Parameter	Description
collectd.statsd.host_cpu	type: integer The collectd CPU type of statsd plug-in.
collectd.statsd.host_elapsed_time	type: integer The collectd elapsed_time type of statsd plug-in.
collectd.statsd.host_memory	type: integer The collectd memory type of statsd plug-in.
collectd.statsd.host_nic_speed	type: integer The collectd nic_speed type of statsd plug-in.
collectd.statsd.host_nic_rx	type: integer The collectd nic_rx type of statsd plug-in.

Parameter	Description
<code>collectd.statsd.host_nic_tx</code>	type: integer The <code>collectd nic_tx</code> type of <code>statsd</code> plug-in.
<code>collectd.statsd.host_nic_rx_dropped</code>	type: integer The <code>collectd nic_rx_dropped</code> type of <code>statsd</code> plug-in.
<code>collectd.statsd.host_nic_tx_dropped</code>	type: integer The <code>collectd nic_tx_dropped</code> type of <code>statsd</code> plug-in.
<code>collectd.statsd.host_nic_rx_errors</code>	type: integer The <code>collectd nic_rx_errors</code> type of <code>statsd</code> plug-in.
<code>collectd.statsd.host_nic_tx_errors</code>	type: integer The <code>collectd nic_tx_errors</code> type of <code>statsd</code> plug-in.
<code>collectd.statsd.host_storage</code>	type: integer The <code>collectd</code> storage type of <code>statsd</code> plug-in.
<code>collectd.statsd.host_swap</code>	type: integer The <code>collectd</code> swap type of <code>statsd</code> plug-in.
<code>collectd.statsd.host_vdsm</code>	type: integer The <code>collectd</code> VDSM type of <code>statsd</code> plug-in.
<code>collectd.statsd.host_vms</code>	type: integer The <code>collectd</code> VMS type of <code>statsd</code> plug-in.
<code>collectd.statsd.vm_nic_tx_dropped</code>	type: integer The <code>collectd nic_tx_dropped</code> type of <code>statsd</code> plug-in.
<code>collectd.statsd.vm_nic_rx_bytes</code>	type: integer The <code>collectd nic_rx_bytes</code> type of <code>statsd</code> plug-in.
<code>collectd.statsd.vm_nic_tx_bytes</code>	type: integer The <code>collectd nic_tx_bytes</code> type of <code>statsd</code> plug-in.

Parameter	Description
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.
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.

Parameter	Description
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.vm_nic_rx_dropped	type: integer The collectd nic_rx_dropped type of statsd plug-in.
collectd.statsd.vm_cpu_user	type: integer The collectd cpu_user type of statsd plug-in.
collectd.statsd.vm_nic_rx_errors	type: integer The collectd nic_rx_errors type of statsd plug-in.
collectd.statsd.vm_nic_tx_errors	type: integer The collectd nic_tx_errors type of statsd plug-in.
collectd.statsd.vm_nic_speed	type: integer The collectd nic_speed type of statsd plug-in.

collectd.postgresql Fields

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

Parameter	Description
collectd.postgresql.pg_n_tup_g	type: integer The collectd type pg_n_tup_g of plug-in postgresql.

Parameter	Description
collectd.postgresql.pg_n_tup_c	type: integer The collectd type pg_n_tup_c of plug-in postgresql.
collectd.postgresql.pg_n_umbakends	type: integer The collectd type pg_numbakends of plug-in postgresql.
collectd.postgresql.pg_xact	type: integer The collectd type pg_xact of plug-in postgresql.
collectd.postgresql.pg_db_size	type: integer The collectd type pg_db_size of plug-in postgresql.
collectd.postgresql.pg_blks	type: integer The collectd type pg_blks of plug-in postgresql.

12.2. RSYSLOG EXPORTED FIELDS

These are the **rsyslog** fields exported by the logging system and available for searching from Elasticsearch and Kibana.

The following fields are RFC5424 based metadata.

Parameter	Description
rsyslog.facility	See syslog specification for more information on rsyslog .
rsyslog.protocol-version	This is the rsyslog protocol version.
rsyslog.structured-data	See syslog specification for more information on syslog structured-data.
rsyslog.msgid	This is the syslog msgid field.
rsyslog.appname	If app-name is the same as programname , then only fill top-level field service . If app-name is not equal to programname , this field will hold app-name . See syslog specifications for more information.

12.3. SYSTEMD EXPORTED FIELDS

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

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

systemd.k Fields

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

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

systemd.t Fields

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

Parameter	Description
systemd.t.AUDIT_LOGINUID	systemd.t.AUDIT_LOGINUID is the user ID for the journal entry process.
systemd.t.BOOT_ID	systemd.t.BOOT_ID is the kernel boot ID.
systemd.t.AUDIT_SESSION	systemd.t.AUDIT_SESSION is the session for the journal entry process.
systemd.t.CAP_EFFECTIVE	systemd.t.CAP_EFFECTIVE represents the capabilities of the journal entry process.
systemd.t.CMDLINE	systemd.t.CMDLINE is the command line of the journal entry process.
systemd.t.COMM	systemd.t.COMM is the name of the journal entry process.
systemd.t.EXE	systemd.t.EXE is the executable path of the journal entry process.
systemd.t.GID	systemd.t.GID is the group ID for the journal entry process.
systemd.t.HOSTNAME	systemd.t.HOSTNAME is the name of the host.

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

systemd.u Fields

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

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

12.4. 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.
kubernetes.host	Kubernetes node name.
kubernetes.container_name	The name of the container in Kubernetes.

Parameter	Description
kubernetes.labels.deployment	The deployment associated with the Kubernetes object.
kubernetes.labels.deploymentconfig	The deploymentconfig associated with the Kubernetes object.
kubernetes.labels.component	The component associated with the Kubernetes object.
kubernetes.labels.provider	The provider associated with the Kubernetes object.

kubernetes.annotations Fields

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

12.5. CONTAINER EXPORTED FIELDS

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

pipeline_metadata.collector Fields

This section contains metadata specific to the collector.

Parameter	Description
pipeline_metadata.collector.hostname	FQDN of the collector. It might be different from the FQDN of the actual emitter of the logs.
pipeline_metadata.collector.name	Name of the collector.
pipeline_metadata.collector.version	Version of the collector.
pipeline_metadata.collector.ipaddr4	IP address v4 of the collector server, can be an array.
pipeline_metadata.collector.ipaddr6	IP address v6 of the collector server, can be an array.
pipeline_metadata.collector.inputname	How the log message was received by the collector whether it was TCP/UDP, or imjournal/imfile.
pipeline_metadata.collector.received_at	Time when the message was received by the collector.

Parameter	Description
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.

12.6. OVIRT EXPORTED FIELDS

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

Namespace for oVirt metadata.

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

ovirt.engine Fields

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

12.7. AUSHAPE EXPORTED FIELDS

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

Audit events converted with Aushape. For more information, see [Aushape](#).

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

aushape.data Fields

Parsed audit event data related to Aushape.

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

12.8. TLOG EXPORTED FIELDS

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

Tlog terminal I/O recording messages. For more information see [Tlog](#).

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

CHAPTER 13. UNINSTALLING CLUSTER LOGGING

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

13.1. UNINSTALLING CLUSTER LOGGING FROM OPENSIFT 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
```