



OpenShift Container Platform 4.3

Logging

Configuring cluster logging in OpenShift Container Platform

OpenShift Container Platform 4.3 Logging

Configuring cluster logging in OpenShift Container Platform

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 cluster logging, which aggregates logs for a range of OpenShift Container Platform services.

Table of Contents

CHAPTER 1. UNDERSTANDING CLUSTER LOGGING AND OPENSIFT CONTAINER PLATFORM	6
1.1. CLUSTER LOGGING	6
1.1.1. Cluster logging components	6
1.1.2. About the log store	6
1.1.3. About the logging collector	7
1.1.4. About logging visualization	7
1.1.5. About logging curation	8
1.1.6. About event routing	8
1.1.7. About the Cluster Logging Custom Resource	8
CHAPTER 2. ABOUT DEPLOYING CLUSTER LOGGING	10
2.1. ABOUT DEPLOYING AND CONFIGURING CLUSTER LOGGING	10
2.1.1. Configuring and Tuning Cluster Logging	10
2.1.2. Sample modified Cluster Logging Custom Resource	12
2.2. STORAGE CONSIDERATIONS FOR CLUSTER LOGGING AND OPENSIFT CONTAINER PLATFORM	13
2.3. ADDITIONAL RESOURCES	13
CHAPTER 3. DEPLOYING CLUSTER LOGGING	14
3.1. INSTALL THE ELASTICSEARCH OPERATOR USING THE CLI	14
3.2. INSTALL THE CLUSTER LOGGING OPERATOR USING THE WEB CONSOLE	17
3.3. INSTALL THE CLUSTER LOGGING OPERATOR USING THE CLI	21
3.4. ADDITIONAL RESOURCES	26
CHAPTER 4. UPDATING CLUSTER LOGGING	27
4.1. UPDATING CLUSTER LOGGING	27
CHAPTER 5. VIEWING CLUSTER LOGS	30
5.1. VIEWING CLUSTER LOGS	30
5.2. VIEWING CLUSTER LOGS IN THE OPENSIFT CONTAINER PLATFORM WEB CONSOLE	30
CHAPTER 6. VIEWING CLUSTER LOGS USING KIBANA	32
6.1. LAUNCHING KIBANA	32
CHAPTER 7. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT	34
7.1. ABOUT CONFIGURING CLUSTER LOGGING	34
7.1.1. About deploying and configuring cluster logging	34
7.1.1.1. Configuring and Tuning Cluster Logging	34
7.1.1.2. Sample modified Cluster Logging Custom Resource	36
7.2. CHANGING CLUSTER LOGGING MANAGEMENT STATE	37
7.2.1. Changing the cluster logging management state	38
7.2.2. Changing the Elasticsearch management state	39
7.3. CONFIGURING CLUSTER LOGGING	39
7.3.1. Understanding the cluster logging component images	41
7.4. CONFIGURING ELASTICSEARCH TO STORE AND ORGANIZE LOG DATA	41
7.4.1. Configuring Elasticsearch CPU and memory limits	42
7.4.2. Configuring Elasticsearch replication policy	43
7.4.3. Configuring Elasticsearch storage	44
7.4.4. Configuring Elasticsearch for emptyDir storage	45
7.4.5. Exposing Elasticsearch as a route	45
7.4.6. About Elasticsearch alerting rules	48
7.5. CONFIGURING KIBANA	49
7.5.1. Configure Kibana CPU and memory limits	49
7.5.2. Scaling Kibana for redundancy	50

7.5.3. Using tolerations to control the Kibana Pod placement	51
7.5.4. Installing the Kibana Visualize tool	52
7.6. CURATION OF ELASTICSEARCH DATA	53
7.6.1. Configuring the Curator schedule	53
7.6.2. Configuring Curator index deletion	54
7.6.3. Troubleshooting Curator	56
7.6.4. Configuring Curator in scripted deployments	57
7.6.5. Using the Curator Action file	58
7.7. CONFIGURING THE LOGGING COLLECTOR	59
7.7.1. Viewing logging collector pods	60
7.7.2. Configure log collector CPU and memory limits	60
7.7.3. Configuring Buffer Chunk Limiting for Fluentd	61
7.7.4. Configuring the logging collector using environment variables	63
7.7.5. About logging collector alerts	63
7.8. COLLECTING AND STORING KUBERNETES EVENTS	64
7.8.1. Deploying and configuring the Event Router	64
7.9. USING TOLERATIONS TO CONTROL CLUSTER LOGGING POD PLACEMENT	68
7.9.1. Using tolerations to control the Elasticsearch Pod placement	69
7.9.2. Using tolerations to control the Kibana Pod placement	70
7.9.3. Using tolerations to control the Curator Pod placement	71
7.9.4. Using tolerations to control the log collector Pod placement	72
7.9.5. Additional resources	74
7.10. FORWARD LOGS TO THIRD PARTY SYSTEMS	74
7.10.1. Forwarding logs using the Fluentd forward protocol	74
7.10.2. Forwarding logs using the syslog protocol	78
7.10.3. Forwarding logs using the Log Forwarding API	81
7.10.3.1. Understanding the Log Forwarding API	82
7.10.3.2. Enabling the Log Forwarding API	84
7.10.3.3. Configuring log forwarding using the Log Forwarding API	85
7.10.3.3.1. Example log forwarding custom resources	87
7.10.3.4. Disabling the Log Forwarding API	88
7.11. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD	89
7.11.1. Configuring systemd-journald for cluster logging	89
CHAPTER 8. VIEWING ELASTICSEARCH STATUS	92
8.1. VIEWING ELASTICSEARCH STATUS	92
8.1.1. Example condition messages	94
8.2. VIEWING ELASTICSEARCH COMPONENT STATUS	95
CHAPTER 9. VIEWING CLUSTER LOGGING STATUS	99
9.1. VIEWING THE STATUS OF THE CLUSTER LOGGING OPERATOR	99
9.1.1. Example condition messages	101
9.2. VIEWING THE STATUS OF CLUSTER LOGGING COMPONENTS	103
CHAPTER 10. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS	105
10.1. MOVING THE CLUSTER LOGGING RESOURCES	105
CHAPTER 11. MANUALLY ROLLING OUT ELASTICSEARCH	109
11.1. PERFORMING AN ELASTICSEARCH ROLLING CLUSTER RESTART	109
CHAPTER 12. TROUBLESHOOTING KIBANA	113
12.1. TROUBLESHOOTING A KUBERNETES LOGIN LOOP	113
12.2. TROUBLESHOOTING A KUBERNETES CRYPTIC ERROR WHEN VIEWING THE KIBANA CONSOLE	113
12.3. TROUBLESHOOTING A KUBERNETES 503 ERROR WHEN VIEWING THE KIBANA CONSOLE	113

CHAPTER 13. EXPORTED FIELDS	115
13.1. DEFAULT EXPORTED FIELDS	115
Top Level Fields	115
collectd Fields	117
collectd.processes Fields	117
collectd.processes.ps_disk_ops Fields	117
collectd.processes.ps_cputime Fields	118
collectd.processes.ps_count Fields	118
collectd.processes.ps_pagefaults Fields	118
collectd.processes.ps_disk_octets Fields	119
collectd.disk Fields	119
collectd.disk.disk_merged Fields	119
collectd.disk.disk_octets Fields	119
collectd.disk.disk_time Fields	120
collectd.disk.disk_ops Fields	120
collectd.disk.disk_io_time Fields	120
collectd.interface Fields	121
collectd.interface.if_octets Fields	121
collectd.interface.if_packets Fields	121
collectd.interface.if_errors Fields	121
collectd.interface.if_dropped Fields	122
collectd.virt Fields	122
collectd.virt.if_octets Fields	122
collectd.virt.if_packets Fields	122
collectd.virt.if_errors Fields	122
collectd.virt.if_dropped Fields	123
collectd.virt.disk_ops Fields	123
collectd.virt.disk_octets Fields	123
collectd.CPU Fields	124
collectd.df Fields	124
collectd.entropy Fields	124
collectd.memory Fields	124
collectd.swap Fields	125
collectd.load Fields	125
collectd.load.load Fields	125
collectd.aggregation Fields	125
collectd.statsd Fields	126
collectd.postgresql Fields	129
13.2. SYSTEMD EXPORTED FIELDS	130
systemd.k Fields	130
systemd.t Fields	130
systemd.u Fields	132
13.3. KUBERNETES EXPORTED FIELDS	132
kubernetes.labels Fields	132
kubernetes.annotations Fields	133
13.4. CONTAINER EXPORTED FIELDS	133
pipeline_metadata.collector Fields	133
pipeline_metadata.normalizer Fields	134
13.5. OVIRT EXPORTED FIELDS	134
ovirt.engine Fields	135
13.6. AUSHAPE EXPORTED FIELDS	135
aushape.data Fields	135
13.7. TLOG EXPORTED FIELDS	136

CHAPTER 14. UNINSTALLING CLUSTER LOGGING	137
14.1. UNINSTALLING CLUSTER LOGGING FROM OPENSIFT CONTAINER PLATFORM	137

CHAPTER 1. UNDERSTANDING CLUSTER LOGGING AND OPENSIFT CONTAINER PLATFORM

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

1.1. CLUSTER LOGGING

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

You can configure cluster logging by modifying the Cluster Logging Custom Resource (CR), named **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. Cluster logging components

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

There are currently 5 different types of cluster logging components:

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

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

1.1.2. About the log store

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



NOTE

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

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



NOTE

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

Role-based access control (RBAC) applied on the Elasticsearch indices enables the controlled access of the logs to the developers. Access to the indexes with the `project.{project_name}.{project_uuid}.*` format is restricted based on the permissions of the user in the specific project.

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

1.1.3. About the logging collector

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

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

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



IMPORTANT

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

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

1.1.4. About logging visualization

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

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

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

1.1.5. About logging curation

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

```
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 routing

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

You must manually deploy the Event Router.

1.1.7. About the Cluster Logging Custom Resource

To make changes to your cluster logging deployment, create and modify the Cluster Logging Custom Resource (CR). Instructions for creating or modifying a CR are provided in this documentation as appropriate.

The following is an example of a typical Custom Resource for cluster logging.

Sample Cluster Logging CR

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
  elasticsearch:
    nodeCount: 3
  resources:
```

```
limits:
  memory: 16Gi
requests:
  cpu: 500m
  memory: 16Gi
storage:
  storageClassName: "gp2"
  size: "200G"
  redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana"
kibana:
  resources:
    limits:
      memory: 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 OpenShift Container Platform cluster, review the following sections.

2.1. ABOUT DEPLOYING AND CONFIGURING CLUSTER LOGGING

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

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

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

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

2.1.1. Configuring and Tuning Cluster Logging

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

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

Memory and CPU

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

```
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu:
          memory: 16Gi
        requests:
          cpu: 500m
          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:
    nodeCount: 3
    storage:
      storageClassName: "gp2"
      size: "200G"

```

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



NOTE

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

```

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

```

Elasticsearch replication policy

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

- **FullRedundancy**. The shards for each index are fully replicated to every data node.
- **MultipleRedundancy**. The shards for each index are spread over half of the data nodes.

- **SingleRedundancy.** A single copy of each shard. Logs are always available and recoverable as long as at least two data nodes exist.
- **ZeroRedundancy.** No copies of any shards. Logs may be unavailable (or lost) in the event a node is down or fails.

Curator schedule

You specify the schedule for Curator in the [cron format](#).

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

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: 3
    resources:
      limits:
        memory: 32Gi
      requests:
        cpu: 3
        memory: 32Gi
    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

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

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

- Reviewing the installation options in [About deploying cluster logging](#).
- Reviewing the [cluster logging storage considerations](#).
- Installing the Elasticsearch Operator and Cluster Logging Operator.

3.1. INSTALL THE ELASTICSEARCH OPERATOR USING THE CLI

You **must** install the Elasticsearch Operator using the CLI following the directions below.

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

Procedure

To install the Elasticsearch Operator using the CLI:

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

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

1 You must specify the **openshift-operators-redhat** Namespace. To prevent possible conflicts with metrics, you should configure the Prometheus Cluster Monitoring stack to scrape metrics from the **openshift-operators-redhat** Namespace and not the **openshift-operators** Namespace. The **openshift-operators** Namespace might contain Community Operators, which are untrusted and could publish a metric with the same name as an OpenShift Container Platform metric, which would cause conflicts.

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

- b. Create the Namespace:

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

For example:

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

2. 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 <file-name>.yaml
```

For example:

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

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

Example Subscription

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: "elasticsearch-operator"
  namespace: "openshift-operators-redhat" 1
spec:
  channel: "4.3" 2
  installPlanApproval: "Automatic"
  source: "redhat-operators" 3
  sourceNamespace: "openshift-marketplace"
  name: "elasticsearch-operator"
```

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

- 2** Specify **4.3** as the channel.

- 3** Specify **redhat-operators**. If your OpenShift Container Platform cluster is installed on a restricted network, also known as a disconnected cluster, specify the name of the

CatalogSource object created when you configured the Operator Lifecycle Manager (OLM).

- d. Create the Subscription object:

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

For example:

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

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

```
$ oc project openshift-operators-redhat
```

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

- f. 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
```

- g. Create the RBAC object:

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

For example:

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

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

3. Verify the Operator installation:

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

NAMESPACE	VERSION	REPLACES	NAME	PHASE	DISPLAY
default			elasticsearch-operator.4.3.1-202002032140		
Elasticsearch Operator	4.3.1-202002032140			Succeeded	
kube-node-lease			elasticsearch-operator.4.3.1-202002032140		
Elasticsearch Operator	4.3.1-202002032140			Succeeded	
kube-public			elasticsearch-operator.4.3.1-202002032140		
Elasticsearch Operator	4.3.1-202002032140			Succeeded	
kube-system			elasticsearch-operator.4.3.1-202002032140		
Elasticsearch Operator	4.3.1-202002032140			Succeeded	
openshift-apiserver-operator			elasticsearch-operator.4.3.1-202002032140		
Elasticsearch Operator	4.3.1-202002032140			Succeeded	
openshift-apiserver			elasticsearch-operator.4.3.1-202002032140		
Elasticsearch Operator	4.3.1-202002032140			Succeeded	
openshift-authentication-operator			elasticsearch-operator.4.3.1-202002032140		
Elasticsearch Operator	4.3.1-202002032140			Succeeded	
openshift-authentication			elasticsearch-operator.4.3.1-202002032140		
Elasticsearch Operator	4.3.1-202002032140			Succeeded	
...					

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

Next step

Install the Cluster Logging Operator using the Console or the CLI using the steps in the following sections.

3.2. INSTALL THE CLUSTER LOGGING OPERATOR USING THE WEB CONSOLE

You can use the OpenShift Container Platform web console to install the Cluster Logging Operator.



NOTE

You cannot create a Project starting with **openshift-** using the web console or by using the **oc new-project** command. You must create a Namespace using a YAML object file and run the **oc create -f <file-name>.yaml** command, as shown.

Procedure

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

1. Create a Namespace for the Cluster Logging Operator. You must use the CLI to create the Namespace.
 - a. Create a Namespace object YAML file (for example, **clo-namespace.yaml**) for the Cluster Logging Operator:

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

1 2 3 Specify these values as shown.

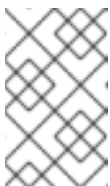
- b. Create the Namespace:

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

For example:

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

2. Install the Cluster Logging Operator:
 - a. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
 - b. Choose **Cluster Logging** from the list of available Operators, and click **Install**.
 - c. On the **Create Operator Subscription** page, under **A specific Namespace on the cluster** select **openshift-logging**. Then, click **Subscribe**.
3. Verify that the Cluster Logging Operator installed:
 - a. Switch to the **Operators → Installed Operators** page.
 - b. Ensure that **Cluster Logging** is listed in the **openshift-logging** project with a **Status** of **InstallSucceeded**.



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 the Operator does not appear as installed, to troubleshoot further:

- Switch to the **Operators → Installed Operators** page and inspect the **Status** column for any errors or failures.

- Switch to the **Workloads** → **Pods** page and check the logs in any Pods in the **openshift-logging** and **openshift-operators-redhat** projects that are reporting issues.
4. 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 field, replace the code with the following:



NOTE

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

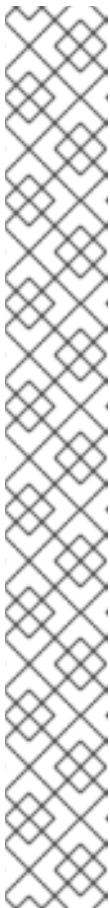
```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging"
spec:
  managementState: "Managed" 2
  logStore:
    type: "elasticsearch" 3
    elasticsearch:
      nodeCount: 3 4
      storage:
        storageClassName: "<storage-class-name>" 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 must be **instance**.

- 2 The cluster logging management state. In most cases, if you change the cluster logging defaults, you must set this to **Unmanaged**. However, an unmanaged deployment does
- 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 Enter the name of an existing StorageClass for Elasticsearch storage. For best performance, specify a StorageClass that allocates block storage.
- 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
```

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

- f. Click **Create**. This creates the Cluster Logging Custom Resource and Elasticsearch Custom Resource, which you can edit to make changes to your cluster logging cluster.
5. 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

3.3. INSTALL THE CLUSTER LOGGING OPERATOR USING THE CLI

You can use the OpenShift Container Platform CLI to install the Cluster Logging Operator. The Cluster Logging Operator creates and manages the components of the logging stack.

Procedure

To install the Cluster Logging Operator using the CLI:

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

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

- b. Create the Namespace:

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

For example:

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

2. Install the Cluster Logging Operator by creating the following objects:
 - a. Create an OperatorGroup object YAML file (for example, **clo-og.yaml**) for the Cluster Logging Operator:

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

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

- b. Create the OperatorGroup object:

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

For example:

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

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

Example Subscription

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: cluster-logging
  namespace: openshift-logging 1
spec:
  channel: "4.3" 2
  name: cluster-logging
  source: redhat-operators 3
  sourceNamespace: openshift-marketplace
```

- 1** You must specify the **openshift-logging** Namespace.
- 2** Specify **4.3** as the channel.
- 3** Specify **redhat-operators**. If your OpenShift Container Platform cluster is installed on a restricted network, also known as a disconnected cluster, specify the name of the CatalogSource object you created when you configured the Operator Lifecycle Manager (OLM).

- d. Create the Subscription object:

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

For example:

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

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

3. Verify the Operator installation.

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

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

NAMESPACE	VERSION	REPLACES	PHASE	NAME	DISPLAY
...					
openshift-logging	Logging	4.3.1-202002032140	Succeeded	clusterlogging.4.3.1-202002032140	Cluster
...					

4. Create a Cluster Logging instance:

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



NOTE

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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging"
spec:
  managementState: "Managed" 2
  logStore:
    type: "elasticsearch" 3
    elasticsearch:
      nodeCount: 3 4
      storage:
        storageClassName: "<storage-class-name>" 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 must be **instance**.
- 2** The Cluster Logging management state. In most cases, if you change the Cluster Logging defaults, you must set this to **Unmanaged**. However, an unmanaged deployment does not receive updates until Cluster Logging is placed back into the **Managed** state. For more information, see **Changing cluster logging management state**.
- 3** Settings for configuring Elasticsearch. Using the Custom Resource (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** Enter the name of an existing StorageClass for Elasticsearch storage. For best performance, specify a StorageClass that allocates block storage.
- 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  1/1   1      0      6m54s
elasticsearch-cdm-x6kdekli-1  1/1   1      1      18h
elasticsearch-cdm-x6kdekli-2  1/1   1      0      6m49s
elasticsearch-cdm-x6kdekli-3  1/1   1      0      6m44s
```

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

- b. Create the instance:

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

For example:

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

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

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

```
NAME                                READY STATUS RESTARTS AGE
cluster-logging-operator-66f77fccb-ppzbg  1/1 Running 0       7m
elasticsearch-cdm-ftuhduuw-1-ffc4b9566-q6bhp  2/2 Running 0       2m40s
elasticsearch-cdm-ftuhduuw-2-7b4994dbfc-rd2gc  2/2 Running 0       2m36s
elasticsearch-cdm-ftuhduuw-3-84b5ff7ff8-gqnm2  2/2 Running 0       2m4s
fluentd-587vb                            1/1 Running 0       2m26s
fluentd-7mpb9                             1/1 Running 0       2m30s
fluentd-flm6j                             1/1 Running 0       2m33s
fluentd-gn4rn                             1/1 Running 0       2m26s
fluentd-nlgb6                             1/1 Running 0       2m30s
fluentd-snpkt                             1/1 Running 0       2m28s
kibana-d6d5668c5-rppqm                    2/2 Running 0       2m39s
```

3.4. ADDITIONAL RESOURCES

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

CHAPTER 4. UPDATING CLUSTER LOGGING

After updating the OpenShift Container Platform cluster from 4.2 to 4.3, you must then upgrade cluster logging from 4.2 to 4.3.

4.1. UPDATING CLUSTER LOGGING

After updating the OpenShift Container Platform cluster, you can update cluster logging from 4.2 to 4.3 by updating the subscription for the Elasticsearch Operator and the Cluster Logging Operator.



IMPORTANT

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

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

Prerequisites

- Update the cluster from 4.2 to 4.3.
- Make sure the cluster logging status is healthy:
 - All Pods are **ready**.
 - Elasticsearch cluster is healthy.
- Optionally, copy your current **secure-forward.conf** section from the Fluentd ConfigMap for use if you want to create the **secure-forward** ConfigMap. See the note above.

Procedure

1. Update the Elasticsearch Operator:
 - a. From the web console, click **Operators → Installed Operators**.
 - b. Select the **openshift-logging** project.
 - c. Click the **Elasticsearch Operator**.
 - d. Click **Subscription → Channel**.
 - e. In the **Change Subscription Update Channel** window, select **4.3** and click **Save**.
 - f. Wait for a few seconds, then click **Operators → Installed Operators**.
The Elasticsearch Operator is shown as 4.3. For example:

Elasticsearch Operator
4.3.0-201909201915 provided
by Red Hat, Inc

2. Update the Cluster Logging Operator:
 - a. From the web console, click **Operators** → **Installed Operators**.
 - b. Select the **openshift-logging** Project.
 - c. Click the **Cluster Logging Operator**.
 - d. Click **Subscription** → **Channel**.
 - e. In the **Change Subscription Update Channel** window, select **4.3** and click **Save**.
 - f. Wait for a few seconds, then click **Operators** → **Installed Operators**.
The Cluster Logging Operator is shown as 4.3. For example:

```
Cluster Logging
4.3.0-201909201915 provided
by Red Hat, Inc
```

3. Check the logging components:
 - a. Ensure that the Elasticsearch Pods are using a 4.3 image:

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

image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-oauth-proxy:v4.3.0-202001081344
```

- b. Ensure that all Elasticsearch Pods are in the **Ready** status:

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

NAME                                READY STATUS RESTARTS AGE
elasticsearch-cdm-1pbrl44l-1-55b7546f4c-mshhk 2/2 Running 0      31m
elasticsearch-cdm-1pbrl44l-2-5c6d87589f-gx5hk 2/2 Running 0      30m
elasticsearch-cdm-1pbrl44l-3-88df5d47-m45jc 2/2 Running 0      29m
```

- c. Ensure that the Elasticsearch cluster is healthy:

```
oc exec -n openshift-logging -c elasticsearch elasticsearch-cdm-1pbrl44l-1-55b7546f4c-
mshhk -- es_cluster_health

{
  "cluster_name" : "elasticsearch",
```



```
"status" : "green",
```

```
....
```

- d. Ensure that the logging collector Pods are using a 4.3 image:

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

```
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-fluentd:v4.3.0-202001081344
```

- e. Ensure that the Kibana Pods are using a 4.3 image:

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

```
image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
image: registry.redhat.io/openshift4/ose-logging-kibana5:v4.3.0-202001081344
```

- f. Ensure that the Curator CronJob is using a 4.3 image:

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

```
image: registry.redhat.io/openshift4/ose-logging-curator5:v4.3.0-202001081344
```

CHAPTER 5. VIEWING CLUSTER LOGS

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

5.1. VIEWING CLUSTER LOGS

You can view cluster logs in the CLI.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

To view cluster logs:

Use the **oc logs [-f] <pod_name>** command, where the **-f** is optional.

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

- 1** Specify the name of a log collector 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.

By default, Fluentd reads logs from the tail, or end, of the log.

5.2. VIEWING CLUSTER LOGS IN THE OPENSIFT CONTAINER PLATFORM WEB CONSOLE

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

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

To view cluster logs:

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

By default, Fluentd reads logs from the tail, or end, of the log.

CHAPTER 6. VIEWING CLUSTER LOGS USING KIBANA

The cluster logging installation deploys the Kibana web console.

6.1. LAUNCHING KIBANA

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

Prerequisites

If you installed OpenShift Container Platform with a proxy, you need to add **.apps.<cluster_name>.<base_domain>** to the **noProxy** list in your cluster-wide Proxy object.

For example:

```
$ oc edit proxy/cluster

apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  creationTimestamp: "2020-03-30T00:45:44Z"
  generation: 3
  name: cluster
  resourceVersion: "26654"
  selfLink: /apis/config.openshift.io/v1/proxies/cluster
  uid: 2213b41b-0721-4c9f-9586-0678c0058f85
spec:
  httpProxy: http://proxy.com
  httpsProxy: https://proxy.com
  noProxy: .apps.mycluster.example.com 1
  trustedCA:
    name: user-ca-bundle
```

1 Add **.apps.<cluster_name>.<base_domain>** to the **noProxy** list. This is a comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying.

Procedure

To launch Kibana:

1. In the OpenShift Container Platform console, click **Monitoring → Logging**.
2. Log in using the same credentials you use to log in to the OpenShift Container Platform console.

The Kibana interface launches. You can now:

- Search and browse your data using the Discover page.
 - Chart and map your data using the Visualize page.
 - Create and view custom dashboards using the Dashboard page.
- Use and configuration of the Kibana interface is beyond the scope of this documentation. For more information, on using the interface, see the [Kibana documentation](#).

**NOTE**

If you get a **security_exception** error in the Kibana console and cannot access your Kibana indices, you might have an expired OAuth token. If you see this error, log out of the Kibana console, and then log back in. This refreshes your OAuth tokens and you should be able to access your indices.

CHAPTER 7. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT

7.1. ABOUT CONFIGURING CLUSTER LOGGING

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



NOTE

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

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

For more information, see [Support policy for unmanaged Operators](#).

7.1.1. 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.

7.1.1.1. 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: 16Gi
        requests:
```

```

    cpu: 500m
    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:
    nodeCount: 3
    storage:
      storageClassName: "gp2"
      size: "200G"

```

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

**NOTE**

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

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

Elasticsearch replication policy

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

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

Curator schedule

You specify the schedule for Curator in the [cron format](#).

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

7.1.1.2. 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: 3
```



```

resources:
  limits:
    memory: 32Gi
  requests:
    cpu: 3
    memory: 32Gi
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

```

7.2. CHANGING CLUSTER LOGGING MANAGEMENT STATE

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

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



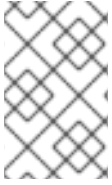
IMPORTANT

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

For more information, see [Support policy for unmanaged Operators](#).

In managed state, the Cluster Logging Operator (CLO) responds to changes in the Cluster Logging Custom Resource (CR) and adjusts the logging deployment accordingly.

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



NOTE

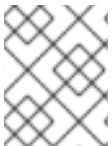
If you set the Elasticsearch Operator (EO) to unmanaged and leave the Cluster Logging Operator (CLO) as managed, the CLO will revert changes you make to the EO, as the EO is managed by the CLO.

7.2.1. 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**.

7.2.2. 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.

7.3. 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
      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.

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

The values might be different depending on your environment.



IMPORTANT

The logging routes are managed by the Cluster Logging Operator and cannot be modified by the user.

7.4. CONFIGURING ELASTICSEARCH TO STORE AND ORGANIZE LOG DATA

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

Some of the modifications you can make to your log store include:

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

**NOTE**

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

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

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

**NOTE**

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

7.4.1. 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: 500m
  memory: 16Gi
```

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

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

For example:

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

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

7.4.2. 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.



NOTE

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

7.4.3. Configuring Elasticsearch storage

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



WARNING

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

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

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

```
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.

7.4.4. 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: {}

```

7.4.5. Exposing Elasticsearch as a route

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

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

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

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

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

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

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

```
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 -n openshift-logging -- 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.

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. Run the following command to 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"
  }
}
]
}

```

7.4.6. 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

Alert	Description	Severity
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 space 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
AggregatedLoggingSystemCPUPHigh	System CPU usage on the node in cluster is <value>	alert
ElasticsearchProcessCPUPHigh	ES process CPU usage on the node in cluster is <value>	alert

7.5. CONFIGURING KIBANA

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

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



NOTE

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

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

For more information, see [Support policy for unmanaged Operators](#).

7.5.1. 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.

7.5.2. 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.

7.5.3. Using tolerations to control the Kibana Pod placement

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

You apply tolerations to the Kibana Pods through the Cluster Logging Custom Resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all Pods that do not tolerate the taint. Using a specific **key:value** pair that is not on other Pods ensures only the Kibana Pod can run on that node.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

- Use the following command to add a taint to a node where you want to schedule the Kibana Pod:

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

For example:

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

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

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

```

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

```

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

bound to a node before being evicted.

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

7.5.4. Installing the Kibana Visualize tool

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

Procedure

To load dashboards and other Kibana UI objects:

1. If necessary, get the Kibana route, which is created by default upon installation of the Cluster Logging Operator:

```
$ 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>
```



NOTE

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

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

7.6. CURATION OF ELASTICSEARCH DATA

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

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

OpenShift Container Platform uses the `config.yaml` internally to generate the Curator [action file](#).

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

7.6.1. 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.

7.6.2. 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:

- Edit the OpenShift Container Platform custom Curator configuration file:

```
$ oc edit configmap/curator
```

- Set the following parameters as needed:

```

config.yaml: |
  project_name:
  action
  unit:value

```

The available parameters are:

Table 7.1. 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.

Variable Name	Description
unit	The period to use for deletion, days , weeks , or months .
value	The number of units.

Table 7.2. 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-qa:
    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**).

7.6.3. 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.

2. Trigger next curator iteration:

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

3. 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](#).

7.6.4. 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.

7.6.5. Using the Curator Action file

The **Curator** ConfigMap in the **openshift-logging** project includes a [Curator action file](#) where you configure any Curator action to be run periodically.

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



IMPORTANT

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

Prerequisite

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

Procedure

To configure Curator to delete indices:

1. Edit the Curator ConfigMap:

```
oc edit cm/curator -n openshift-logging
```

2. Make the following changes to the **action** file:

```
actions:
  1:
    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.

7.7. CONFIGURING THE LOGGING COLLECTOR

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

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

**NOTE**

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

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

For more information, see [Support policy for unmanaged Operators](#).

7.7.1. Viewing logging collector pods

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

Procedure

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

```
$ oc get pods --all-namespaces -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>	<none>					
fluentd-cnc4c	1/1	Running	0	4m56s	10.128.2.13	ip-10-0-155-142.ec2.internal
<none>	<none>					
fluentd-nlp8z	1/1	Running	0	4m56s	10.131.0.13	ip-10-0-138-77.ec2.internal
<none>	<none>					
fluentd-rknlk	1/1	Running	0	4m56s	10.128.0.33	ip-10-0-128-130.ec2.internal
<none>	<none>					
fluentd-rsm49	1/1	Running	0	4m56s	10.129.0.37	ip-10-0-163-191.ec2.internal
<none>	<none>					
fluentd-wjt8s	1/1	Running	0	4m56s	10.130.0.42	ip-10-0-156-251.ec2.internal
<none>	<none>					

7.7.2. Configure log collector CPU and memory limits

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

Procedure

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

```
$ 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
            memory: 736Mi
          requests:
            cpu: 100m
            memory: 736Mi
```

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

7.7.3. Configuring Buffer Chunk Limiting for Fluentd

If the Fluentd log collector is unable to keep up with a high number of logs, Fluentd performs file buffering to reduce memory usage and prevent data loss.

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

You can tune file buffering in your cluster by editing environment variables in the Fluentd Daemonset:



NOTE

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

- **BUFFER_SIZE_LIMIT**. This parameter determines the maximum size of each chunk file before Fluentd creates a new chunk. The default is **8M**. This parameter sets the Fluentd **chunk_limit_size** variable.
A high **BUFFER_SIZE_LIMIT** can collect more records per chunk file. However, bigger records take longer to be sent to the logstore.
- **FILE_BUFFER_LIMIT**. This parameter determines the file buffer size per logging output. This value is only a request based on the available space on the node where a Fluentd pod is scheduled. OpenShift Container Platform does not allow Fluentd to exceed the node capacity. The default is **256Mi**.
A high **FILE_BUFFER_LIMIT** could translate to a higher **BUFFER_QUEUE_LIMIT** based the number of outputs. However, if the node's space is under pressure, Fluentd can fail.

By default, the **number_of_outputs** is **1** if all the logs are sent to a single resource, and is incremented by **1** for each additional resource. You might have multiple outputs if you use the Log Forwarding API, the Fluentd **Forward** protocol, or syslog protocol to forward logs to external locations.

The permanent volume size must be larger than **FILE_BUFFER_LIMIT** multiplied by the number of outputs.

- **BUFFER_QUEUE_LIMIT**. This parameter is the maximum number of buffer chunks allowed. The **BUFFER_QUEUE_LIMIT** parameter is not directly tunable. OpenShift Container Platform

calculates this value based on the number of logging outputs, the chunk size, and the filesystem space available. The default is **32** chunks. To change the **BUFFER_QUEUE_LIMIT**, you must change the value of **FILE_BUFFER_LIMIT**. The **BUFFER_QUEUE_LIMIT** parameter sets the Fluentd **queue_limit_length** parameter.

OpenShift Container Platform calculates the **BUFFER_QUEUE_LIMIT** as **(FILE_BUFFER_LIMIT / (number_of_outputs * BUFFER_SIZE_LIMIT))**.

Using the default set of values, the value of **BUFFER_QUEUE_LIMIT** is 32:

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

OpenShift Container Platform uses the Fluentd [file buffer plug-in](#) to configure how the chunks are stored. You can see the location of the buffer file using the following command:

```
$ oc get cm fluentd -o json | jq -r '.data."fluent.conf"'
```

```
<buffer>
  @type file 1
  path '/var/lib/flunetd/retry-elasticseach' 2
```

1 The Fluentd **file** buffer plugin. Do not change this value.

2 The path where buffer chunks are stored.

Prerequisite

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

Procedure

To configure Buffer Chunk Limiting:

1. Edit either of the following parameters in the **fluentd** Daemonset.

```
spec:
  template:
    spec:
      containers:
        env:
          - name: FILE_BUFFER_LIMIT 1
            value: "256"
          - name: BUFFER_SIZE_LIMIT 2
            value: 8Mi
```

1 Specify the Fluentd file buffer size per output.

2 Specify the maximum size of each Fluentd buffer chunk.

7.7.4. Configuring the logging collector using environment variables

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

See the [Fluentd README](#) in Github for lists of the available environment variables.

Prerequisite

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

Procedure

Set any of the Fluentd environment variables as needed:

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

For example:

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

7.7.5. About logging collector alerts

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

All the logging collector alerts are listed on the **Monitoring** → **Alerts** page of the OpenShift Container Platform web console. Alerts are in one of the following states:

- **Firing.** The alert condition is true for the duration of the timeout. Click the **Options** menu at the end of the firing alert to view more information or silence the alert.
- **Pending** The alert condition is currently true, but the timeout has not been reached.
- **Not Firing.** The alert is not currently triggered.

Table 7.3. Fluentd Prometheus alerts

Alert	Message	Description	Severity
FluentdErrorsHigh	In the last minute, <value> errors reported by fluentd <instance>.	Fluentd is reporting a higher number of issues than the specified number, default 10.	Critical
FluentdNodeDown	Prometheus could not scrape fluentd <instance> for more than 10m.	Fluentd is reporting that Prometheus could not scrape a specific Fluentd instance.	Critical

Alert	Message	Description	Severity
FluentdQueueLengthBurst	In the last minute, fluentd <instance> buffer queue length increased more than 32. Current value is <value>.	Fluentd is reporting that it is overwhelmed.	Warning
FluentdQueueLengthIncreasing	In the last 12h, fluentd <instance> buffer queue length constantly increased more than 1. Current value is <value>.	Fluentd is reporting queue usage issues.	Critical

7.8. COLLECTING AND STORING KUBERNETES EVENTS

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

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



IMPORTANT

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

7.8.1. Deploying and configuring the Event Router

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

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

Prerequisites

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

Procedure

1. Create a template for the Event Router:

```

kind: Template
apiVersion: v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to cluster logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
- kind: ServiceAccount 1
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
- kind: ClusterRole 2
  apiVersion: v1
  metadata:
    name: event-reader
  rules:
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "watch", "list"]
- kind: ClusterRoleBinding 3
  apiVersion: v1
  metadata:
    name: event-reader-binding
  subjects:
  - kind: ServiceAccount
    name: eventrouter
    namespace: ${NAMESPACE}
  roleRef:
    kind: ClusterRole
    name: event-reader
- kind: ConfigMap 4
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  data:
    config.json: |-
      {
        "sink": "stdout"
      }
- kind: Deployment 5
  apiVersion: apps/v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  labels:
    component: eventrouter
    logging-infra: eventrouter
    provider: openshift
  spec:
    selector:
      matchLabels:
        component: eventrouter

```

```

    logging-infra: eventrouter
    provider: openshift
  replicas: 1
  template:
    metadata:
      labels:
        component: eventrouter
        logging-infra: eventrouter
        provider: openshift
        name: eventrouter
    spec:
      serviceAccount: eventrouter
      containers:
        - name: kube-eventrouter
          image: ${IMAGE}
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: ${CPU}
              memory: ${MEMORY}
          volumeMounts:
            - name: config-volume
              mountPath: /etc/eventrouter
      volumes:
        - name: config-volume
          configMap:
            name: eventrouter
  parameters:
    - name: IMAGE
      displayName: Image
      value: "registry.redhat.io/openshift4/ose-logging-eventrouter:latest"
    - name: CPU 6
      displayName: CPU
      value: "100m"
    - name: MEMORY 7
      displayName: Memory
      value: "128Mi"
    - name: NAMESPACE
      displayName: Namespace
      value: "openshift-logging" 8

```

- 1** Creates a Service Account in the **openshift-logging** project for the Event Router.
- 2** Creates a ClusterRole to monitor for events in the cluster.
- 3** Creates a ClusterRoleBinding to bind the ClusterRole to the ServiceAccount.
- 4** Creates a ConfigMap in the **openshift-logging** project to generate the required **config.json** file.
- 5** Creates a Deployment in the **openshift-logging** project to generate and configure the Event Router Pod.
- 6** Specifies the minimum amount of memory to allocate to the Event Router Pod. Defaults to **128Mi**.

- 7 Specifies the minimum amount of CPU to allocate to the Event Router Pod. Defaults to **100m**.
- 8 Specifies the **openshift-logging** project to install objects in.

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

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

For example:

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

Example output

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

3. Validate that the Event Router installed in the **openshift-logging** project:

a. View the new Event Router Pod:

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

Example output

```
pod/cluster-logging-eventrouter-d649f97c8-qvv8r
```

b. View the events collected by the Event Router:

```
$ oc logs <cluster_logging_eventrouter_pod> -n openshift-logging
```

For example:

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

Example output

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

```
message:"Job completed","source":{"component":"job-
controller"},"firstTimestamp":"2020-09-08T15:40:26Z","lastTimestamp":"2020-09-
08T15:40:26Z","count":1,"type":"Normal"}}
```

You can also use Kibana to view events by creating an index pattern using the Elasticsearch **infra** index.

7.9. USING TOLERATIONS TO CONTROL CLUSTER LOGGING POD PLACEMENT

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

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

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

Sample cluster logging CR with tolerations

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



```

    memory: 2Gi
    requests:
      cpu: 100m
      memory: 1Gi
    replicas: 1
  curation:
    type: "curator"
    curator:
      tolerations: 3
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
    resources:
      limits:
        memory: 200Mi
      requests:
        cpu: 100m
        memory: 100Mi
    schedule: "*/5 * * * *"
  collection:
    logs:
      type: "fluentd"
      fluentd:
        tolerations: 4
        - key: "logging"
          operator: "Exists"
          effect: "NoExecute"
          tolerationSeconds: 6000
    resources:
      limits:
        memory: 2Gi
      requests:
        cpu: 100m
        memory: 1Gi

```

- 1 This toleration is added to the Elasticsearch pods.
- 2 This toleration is added to the Kibana pod.
- 3 This toleration is added to the Curator pod.
- 4 This toleration is added to the logging collector pods.

7.9.1. Using tolerations to control the Elasticsearch Pod placement

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

You apply tolerations to Elasticsearch Pods through the Cluster Logging Custom Resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all Pods that do not tolerate the taint. Using a specific **key:value pair** that is not on other Pods ensures only Elasticsearch Pods can run on that node.

By default, the Elasticsearch Pods have the following toleration:

```

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

```

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

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

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

For example:

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

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

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

```

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

```

- Specify the key that you added to the node.
- Specify the **Exists** operator to require a taint with the key **elasticsearch** to be present on the Node.
- Specify the **NoExecute** effect.
- Optionally, specify the **tolerationSeconds** parameter to set how long a Pod can remain bound to a node before being evicted.

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

7.9.2. Using tolerations to control the Kibana Pod placement

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

You apply tolerations to the Kibana Pods through the Cluster Logging Custom Resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all Pods that do not tolerate the taint. Using a specific **key:value** pair that is not on other Pods ensures only the Kibana Pod can run on that node.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

1. Use the following command to add a taint to a node where you want to schedule the Kibana Pod:

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

For example:

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

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

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

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

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

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

7.9.3. Using tolerations to control the Curator Pod placement

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

You apply tolerations to the Curator Pod through the Cluster Logging Custom Resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all Pods that do not tolerate the taint. Using a specific **key:value** pair that is not on other Pods ensures only the Curator Pod can run on that node.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

- Use the following command to add a taint to a node where you want to schedule the Curator Pod:

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

For example:

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

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

- Edit the **curation** section of the Cluster Logging Custom Resource (CR) to configure a toleration for the Curator Pod:

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

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

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

7.9.4. Using tolerations to control the log collector Pod placement

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

You apply tolerations to logging collector Pods through the Cluster Logging Custom Resource (CR) and apply taints to a node through the node specification. You can use taints and tolerations to ensure the Pod does not get evicted for things like memory and CPU issues.

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

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

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

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

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

For example:

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

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

2. Edit the **collection** section of the Cluster Logging Custom Resource (CR) to configure a toleration for the logging collector Pods:

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

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

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

7.9.5. Additional resources

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

7.10. FORWARD LOGS TO THIRD PARTY SYSTEMS

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

- **Sending logs using the Fluentd forward protocol** You can create a Configmap to use the [Fluentd forward protocol](#) to securely send logs to an external logging aggregator that accepts the Fluent **forward** protocol.
- **Sending logs using syslog** You can create a Configmap to use the [syslog protocol](#) to send logs to an external syslog (RFC 3164) server.

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



IMPORTANT

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

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

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

7.10.1. Forwarding logs using the Fluentd forward protocol

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



NOTE

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

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



IMPORTANT

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

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

Sample **secure-forward.conf**

```
<store>
  @type forward
  <security>
    self_hostname ${hostname} # ${hostname} is a placeholder.
    shared_key "fluent-receiver"
  </security>
  transport tls
  tls_verify_hostname false      # Set false to ignore server cert hostname.

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

Sample **secure-forward** ConfigMap based on the configuration

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

```

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

```

kind: ConfigMap

metadata:

creationTimestamp: "2020-01-15T18:56:04Z"

name: secure-forward

namespace: openshift-logging

resourceVersion: "19148"

selfLink: /api/v1/namespaces/openshift-logging/configmaps/secure-forward

uid: 6fd83202-93ab-d851b1d0f3e8

Procedure

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

1. Create a configuration file named **secure-forward.conf** for the **forward** parameters:
 - a. Configure the secrets and TLS information:

```

<store>
@type forward

self_hostname ${hostname} 1
shared_key <SECRET_STRING> 2

transport tls 3

tls_verify_hostname true 4
tls_cert_path <path_to_file> 5

```

- 1 Specify the default value of the auto-generated certificate common name (CN).
- 2 Enter the Shared key between nodes
- 3 Specify **tls** to enable TLS validation.
- 4 Set to **true** to verify the server cert hostname. Set to **false** to ignore server cert hostname.

- 5 Specify the path to private CA certificate file as `/etc/ocp-forward/ca_cert.pem`.

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

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

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

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

For example:

```
<server>
  name externalserver1
  host 192.168.1.1
  hostlabel externalserver1.example.com
  port 24224
</server>
<server>
  name externalserver2
  host externalserver2.example.com
  port 24224
</server>
</store>
```

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

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

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

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

For example:

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

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

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

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

7.10.2. Forwarding logs using the syslog protocol

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

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



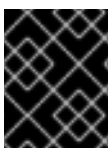
NOTE

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

There are two versions of the **syslog** protocol:

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

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



IMPORTANT

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

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

Sample syslog.conf

```

<store>
@type syslog_buffered ❶
remote_syslog rsyslogserver.openshift-logging.svc.cluster.local ❷
port 514 ❸
hostname fluentd-4nzfz ❹
remove_tag_prefix tag ❺
tag_key ident,systemd.u.SYSLOG_IDENTIFIER ❻
facility local0 ❼
severity info ❽
use_record true ❾
payload_key message ❿
</store>

```

- ❶ The **syslog** protocol, either: **syslog** or **syslog_buffered**.
- ❷ The fully qualified domain name (FQDN) or IP address of the syslog server.
- ❸ The port number to connect on. Defaults to **514**.
- ❹ The name of the syslog server.
- ❺ Removes the prefix from the tag. Defaults to "" (empty).
- ❻ The field to set the syslog key.
- ❼ The syslog log facility or source.
- ❽ The syslog log severity.
- ❾ Determines whether to use the severity and facility from the record if available.
- ❿ The key to set the payload of the syslog message. Defaults to **message**.

Sample syslog ConfigMap based on the sample syslog.conf

```

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

```

```
use_record true
payload_key message
</store>
```

Procedure

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

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

- a. Specify the **syslog** protocol type:

```
@type syslog_buffered 1
```

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

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

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

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

For example:

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

- c. Configure the other syslog variables as needed:

```
remove_tag_prefix 1
tag_key <key> 2
facility <value> 3
severity <value> 4
use_record <value> 5
payload_key message 6
```

- 1 Add this parameter to remove the **tag** field from the syslog prefix.
- 2 Specify the field to set the syslog key.
- 3 Specify the syslog log facility or source. For values, see [RTF 3164](#).
- 4 Specify the syslog log severity. For values, see [link:RTF 3164](#).

- 5 Specify **true** to use the severity and facility from the record if available. If **true**, the **container_name**, **namespace_name**, and **pod_name** are included in the output
- 6 Specify the key to set the payload of the syslog message. Defaults to **message**.

For example:

```
facility local0
severity info
```

The configuration file appears similar to the following:

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

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

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

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

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

7.10.3. Forwarding logs using the Log Forwarding API

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

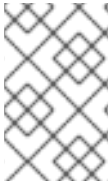


IMPORTANT

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

See the [Red Hat Technology Preview features support scope](#) for more information.

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

**NOTE**

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

7.10.3.1. Understanding the Log Forwarding API

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

**NOTE**

If you want to use only the default internal OpenShift Container Platform Elasticsearch logstore, do not configure the Log Forwarding feature.

By default, the Cluster Logging Operator sends logs to the default internal Elasticsearch logstore, as defined in the Cluster Logging Custom Resource. To use the Log Forwarding feature, you create a custom **logforwarding** configuration file to send logs to endpoints you specify.

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

An output can be either:

- **elasticsearch** to forward logs to an external Elasticsearch v5.x cluster and/or the internal OpenShift Container Platform Elasticsearch instance.
- **forward** to forward logs to an external log aggregation solution. This option uses the Fluentd **forward** protocols.

**NOTE**

The endpoint must be a server name or FQDN, not an IP Address, if the cluster-wide proxy using the CIDR annotation is enabled.

A *pipeline* associates the source of the data to an output. The source of the data is one of the following:

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

Note the following:

- The internal OpenShift Container Platform Elasticsearch instance does not provide secure storage for audit logs. We recommend you ensure that the system to which you forward audit logs is compliant with your organizational and governmental regulations and is properly secured. OpenShift Container Platform cluster logging does not comply with those regulations.
- An output supports TLS communication using a secret. Secrets must have keys of: **tls.crt**, **tls.key**, and **ca-bundler.crt** which point to the respective certificates for which they represent. Secrets must have the key **shared_key** for use when using forward in a secure manner.
- You are responsible to create and maintain any additional configurations that external destinations might require, such as keys and secrets, service accounts, port opening, or global proxy configuration.

The following example creates three outputs:

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

Three pipelines send:

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

Sample log forwarding outputs and pipelines

```

apiVersion: "logging.openshift.io/v1alpha1"
kind: "LogForwarding"
metadata:
  name: instance 1
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true 2
  outputs: 3
  - name: elasticsearch 4
    type: "elasticsearch" 5
    endpoint: elasticsearch.openshift-logging.svc:9200 6
    secret: 7
      name: fluentd
  - name: elasticsearch-insecure
    type: "elasticsearch"
    endpoint: elasticsearch-insecure.svc.messaging.cluster.local
    insecure: true 8
  - name: secureforward-offcluster
    type: "forward"
    endpoint: https://secureforward.offcluster.com:24224
    secret:
      name: secureforward
  pipelines: 9
  - name: container-logs 10

```

```

inputSource: logs.app 11
outputRefs: 12
- elasticsearch
- secureforward-offcluster
- name: infra-logs
inputSource: logs.infra
outputRefs:
- elasticsearch-insecure
- name: audit-logs
inputSource: logs.audit
outputRefs:
- secureforward-offcluster

```

- 1** The name of the log forwarding CR must be **instance**.
- 2** Parameter to disable the default log forwarding behavior.
- 3** Configuration for the outputs.
- 4** A name to describe the output.
- 5** The type of output, either **elasticsearch** or **forward**.
- 6** Enter the endpoint, either the server name, FQDN, or IP address. If the cluster-wide proxy using the CIDR annotation is enabled, the endpoint must be a server name or FQDN, not an IP Address. For the internal OpenShift Container Platform Elasticsearch instance, specify **elasticsearch.openshift-logging.svc:9200**.
- 7** Optional name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project.
- 8** Optional setting if the endpoint does not use a secret, resulting in insecure communication.
- 9** Configuration for the pipelines.
- 10** A name to describe the pipeline.
- 11** The data source: **logs.app**, **logs.infra**, or **logs.audit**.
- 12** The name of one or more outputs configured in the CR.

7.10.3.2. Enabling the Log Forwarding API

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

Procedure

To enable the Log Forwarding API:

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

```
$ oc edit ClusterLogging instance
```

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


```

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

  collection: 2
    logs:
      type: "fluentd"
      fluentd: {}

```

- 1** Enables and disables the Log Forwarding API. Set to **enabled** to use log forwarding. To use the only the OpenShift Container Platform Elasticsearch instance, set to disabled or do not add the annotation.
- 2** The **spec.collection** block must be defined to use Fluentd in the Cluster Logging CR.

7.10.3.3. Configuring log forwarding using the Log Forwarding API

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

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

Procedure

To configure log forwarding using the API:

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

```

apiVersion: "logging.openshift.io/v1alpha1"
kind: "LogForwarding"
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  disableDefaultForwarding: true 3
  outputs: 4
  - name: elasticsearch
    type: "elasticsearch"
    endpoint: elasticsearch.openshift-logging.svc:9200
    secret:
      name: elasticsearch
  - name: elasticsearch-insecure
    type: "elasticsearch"

```

```

  endpoint: elasticsearch-insecure.svc.messaging.cluster.local
  insecure: true
- name: secureforward-offcluster
  type: "forward"
  endpoint: https://secureforward.offcluster.com:24224
  secret:
    name: secureforward
pipelines: 5
- name: container-logs
  inputSource: logs.app
  outputRefs:
  - elasticsearch
  - secureforward-offcluster
- name: infra-logs
  inputSource: logs.infra
  outputRefs:
  - elasticsearch-insecure
- name: audit-logs
  inputSource: logs.audit
  outputRefs:
  - secureforward-offcluster

```

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

**NOTE**

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

2. Create the CR object:

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

7.10.3.3.1. Example log forwarding custom resources

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

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

Sample custom resource to forward to an Elasticsearch logstore

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

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

Sample custom resource to use the forward protocol

```
apiVersion: logging.openshift.io/v1alpha1
```

```

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

```

7.10.3.4. Disabling the Log Forwarding API

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



NOTE

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

Procedure

To disable the Log Forwarding API:

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

```
$ oc edit ClusterLogging instance
```

2. Remove the **clusterlogging.openshift.io/logforwardingtechpreview** annotation:

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  annotations:
    clusterlogging.openshift.io/logforwardingtechpreview: enabled 1

```

```
name: "instance"
namespace: "openshift-logging"
....
```

- 1 Remove this annotation.

3. Delete the Log Forwarding Custom Resource:

```
$ oc delete LogForwarding instance -n openshift-logging
```

7.11. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD

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

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

7.11.1. Configuring systemd-journald for cluster logging

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

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

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

Procedure

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

```
Compress=yes 1
ForwardToConsole=no 2
ForwardToSyslog=no
MaxRetentionSec=1month 3
RateLimitBurst=10000 4
RateLimitInterval=1s
Storage=persistent 5
SyncIntervalSec=1s 6
SystemMaxUse=8g 7
SystemKeepFree=20% 8
SystemMaxFileSize=10M 9
```

- 1 Specify whether you want logs compressed before they are written to the file system. Specify **yes** to compress the message or **no** to not compress. The default is **yes**.
- 2 Configure whether to forward log messages. Defaults to **no** for each. Specify:
 - **ForwardToConsole** to forward logs to the system console.

- **ForwardToKsmg** to forward logs to the kernel log buffer.
 - **ForwardToSyslog** to forward to a syslog daemon.
 - **ForwardToWall** to forward messages as wall messages to all logged-in users.
- 3 Specify the maximum time to store journal entries. Enter a number to specify seconds. Or include a unit: "year", "month", "week", "day", "h" or "m". Enter **0** to disable. The default is **1month**.
 - 4 Configure rate limiting. If, during the time interval defined by **RateLimitIntervalSec**, more logs than specified in **RateLimitBurst** are received, all further messages within the interval are dropped until the interval is over. It is recommended to set **RateLimitInterval=1s** and **RateLimitBurst=10000**, which are the defaults.
 - 5 Specify how logs are stored. The default is **persistent**:
 - **volatile** to store logs in memory in `/var/log/journal/`.
 - **persistent** to store logs to disk in `/var/log/journal/`. systemd creates the directory if it does not exist.
 - **auto** to store logs in in `/var/log/journal/` if the directory exists. If it does not exist, systemd temporarily stores logs in `/run/systemd/journal`.
 - **none** to not store logs. systemd drops all logs.
 - 6 Specify the timeout before synchronizing journal files to disk for **ERR**, **WARNING**, **NOTICE**, **INFO**, and **DEBUG** logs. systemd immediately syncs after receiving a **CRIT**, **ALERT**, or **EMERG** log. The default is **1s**.
 - 7 Specify the maximum size the journal can use. The default is **8g**.
 - 8 Specify how much disk space systemd must leave free. The default is **20%**.
 - 9 Specify the maximum size for individual journal files stored persistently in `/var/log/journal`. The default is **10M**.



NOTE

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

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

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

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

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

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 50-corp-journald
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,{jrnI_cnf}
          mode: 0644 1
          overwrite: true
          path: /etc/systemd/journald.conf 2

```

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

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

4. Create the MachineConfig:

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

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

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

```

$ 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 8. VIEWING ELASTICSEARCH STATUS

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

8.1. VIEWING ELASTICSEARCH STATUS

You can view the status of your Elasticsearch cluster.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

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

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

2. To view the Elasticsearch cluster status:
 - a. Get the name of the Elasticsearch instance:

```
$ oc get Elasticsearch  
  
NAME          AGE  
elasticsearch 5h9m
```

- b. Get the Elasticsearch status:

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

For example:

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

The output includes information similar to the following:

```
status: 1  
cluster: 2  
  activePrimaryShards: 30  
  activeShards: 60  
  initializingShards: 0  
  numDataNodes: 3  
  numNodes: 3  
  pendingTasks: 0  
  relocatingShards: 0  
  status: green  
  unassignedShards: 0  
  clusterHealth: ""  
  conditions: [] 3  
  nodes: 4  
  - deploymentName: elasticsearch-cdm-zjf34ved-1  
    upgradeStatus: {}
```



```

- deploymentName: elasticsearch-cdm-zjf34ved-2
  upgradeStatus: {}
- deploymentName: elasticsearch-cdm-zjf34ved-3
  upgradeStatus: {}
pods: 5
client:
  failed: []
  notReady: []
  ready:
    - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
data:
  failed: []
  notReady: []
  ready:
    - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
master:
  failed: []
  notReady: []
  ready:
    - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
    - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
    - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
shardAllocationEnabled: all

```

- 1 In the output, the cluster status fields appear in the **status** stanza.
- 2 The status of the Elasticsearch cluster:
 - The number of active primary shards.
 - The number of active shards.
 - The number of shards that are initializing.
 - The number of Elasticsearch data nodes.
 - The total number of Elasticsearch nodes.
 - The number of pending tasks.
 - The Elasticsearch status: **green, red, yellow**.
 - The number of unassigned shards.
- 3 Any status conditions, if present. The Elasticsearch cluster status indicates the reasons from the scheduler if a pod could not be placed. Any events related to the following conditions are shown:
 - Container Waiting for both the Elasticsearch and proxy containers.
 - Container Terminated for both the Elasticsearch and proxy containers.

- Pod unschedulable. Also, a condition is shown for a number of issues, see **Example condition messages**.

- 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.

8.1.1. Example condition messages

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

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

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

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

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

This status message indicates the 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
```

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

```

....

```

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

```

```
....  
Events:      <none>
```

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:v4.3  
    Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s  
              period=5s #success=1 #failure=3  
  
....  
Events:      <none>
```

CHAPTER 9. VIEWING CLUSTER LOGGING STATUS

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

9.1. VIEWING THE STATUS OF THE CLUSTER LOGGING OPERATOR

You can view the status of your Cluster Logging Operator.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

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

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

2. To view the cluster logging status:

- a. Get the cluster logging status:

```
$ oc get clusterlogging instance -o yaml
```

The output includes information similar to the following:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
...
status: 1
collection:
logs:
  fluentdStatus:
    daemonSet: fluentd 2
    nodes:
      fluentd-2rhqp: ip-10-0-169-13.ec2.internal
      fluentd-6fgjh: ip-10-0-165-244.ec2.internal
      fluentd-6l2ff: ip-10-0-128-218.ec2.internal
      fluentd-54nx5: ip-10-0-139-30.ec2.internal
      fluentd-flpnn: ip-10-0-147-228.ec2.internal
      fluentd-n2frh: ip-10-0-157-45.ec2.internal
    pods:
      failed: []
      notReady: []
      ready:
        - fluentd-2rhqp
        - fluentd-54nx5
        - fluentd-6fgjh
        - fluentd-6l2ff
        - fluentd-flpnn
        - fluentd-n2frh
```

```

curation: 3
  curatorStatus:
    - cronJobs: curator
      schedules: 30 3 * * *
      suspended: false
logstore: 4
  elasticsearchStatus:
    - ShardAllocationEnabled: all
      cluster:
        activePrimaryShards: 5
        activeShards: 5
        initializingShards: 0
        numDataNodes: 1
        numNodes: 1
        pendingTasks: 0
        relocatingShards: 0
        status: green
        unassignedShards: 0
      clusterName: elasticsearch
      nodeConditions:
        elasticsearch-cdm-mkkdys93-1:
          nodeCount: 1
      pods:
        client:
          failed:
          notReady:
          ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
        data:
          failed:
          notReady:
          ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
        master:
          failed:
          notReady:
          ready:
            - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
  visualization: 5
    kibanaStatus:
      - deployment: kibana
      pods:
        failed: []
        notReady: []
        ready:
          - kibana-7fb4fd4cc9-f2nls
      replicaSets:
        - kibana-7fb4fd4cc9
      replicas: 1

```

- 1 In the output, the cluster status fields appear in the **status** stanza.
- 2 Information on the Fluentd pods.
- 3 Information on the Curator pods.

- 4 Information on the Elasticsearch pods, including Elasticsearch cluster health, **green**, **yellow**, or **red**.
- 5 Information on the Kibana pods.

9.1.1. Example condition messages

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

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

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

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

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

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

```
Elasticsearch Status:
Shard Allocation Enabled: shard allocation unknown
Cluster:
  Active Primary Shards: 0
  Active Shards:        0
  Initializing Shards:  0
  Num Data Nodes:      0
  Num Nodes:           0
  Pending Tasks:       0
  Relocating Shards:   0
  Status:               cluster health unknown
  Unassigned Shards:   0
  Cluster Name:         elasticsearch
```

Node Conditions:

elasticsearch-cdm-mkkdys93-1:

Last Transition Time: 2019-06-26T03:37:32Z

Message: 0/5 nodes are available: 5 node(s) didn't match node selector.

Reason: Unschedulable

Status: True

Type: Unschedulable

elasticsearch-cdm-mkkdys93-2:

Node Count: 2

Pods:

Client:

Failed:

Not Ready:

elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49

elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl

Ready:

Data:

Failed:

Not Ready:

elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49

elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl

Ready:

Master:

Failed:

Not Ready:

elasticsearch-cdm-mkkdys93-1-75dd69dccc-f7f49

elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl

Ready:

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

Node Conditions:

elasticsearch-cdm-mkkdys93-1:

Last Transition Time: 2019-06-26T03:37:32Z

Message: pod has unbound immediate PersistentVolumeClaims (repeated 5 times)

Reason: Unschedulable

Status: True

Type: Unschedulable

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

Curation:

Curator Status:

Cluster Condition:

curator-1561518900-cjx8d:

Last Transition Time: 2019-06-26T03:20:08Z

Reason: Completed

Status: True

Type: ContainerTerminated

curator-1561519200-zqxxj:

Last Transition Time: 2019-06-26T03:20:01Z

Message: 0/5 nodes are available: 1 Insufficient cpu, 5 node(s) didn't match node selector.

Reason: Unschedulable

```
Status:      True
Type:       Unschedulable
Cron Jobs:  curator
Schedules:  */5 * * * *
Suspended:  false
```

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

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

9.2. VIEWING THE STATUS OF CLUSTER LOGGING COMPONENTS

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

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

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

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

2. View the status of the cluster logging deployment:

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

The output includes the following status information:

```
Name:          cluster-logging-operator
...

Conditions:
  Type          Status Reason
  ----          -
  Available     True   MinimumReplicasAvailable
  Progressing   True   NewReplicaSetAvailable
...

Events:
```

```

Type      Reason           Age From           Message
----      -
Normal    ScalingReplicaSet 62m deployment-controller Scaled up replica set cluster-logging-operator-574b8987df to 1----

```

3. View the status of the cluster logging ReplicaSet:

- a. Get the name of a ReplicaSet:

```

$ oc get replicaset
NAME                                DESIRED  CURRENT  READY  AGE
cluster-logging-operator-574b8987df 1         1        1      159m
elasticsearch-cdm-uhr537yu-1-6869694fb 1         1        1      157m
elasticsearch-cdm-uhr537yu-2-857b6d676f 1         1        1      156m
elasticsearch-cdm-uhr537yu-3-5b6fdd8cfd 1         1        1      155m
kibana-5bd5544f87                    1         1        1      157m

```

- b. Get the status of the ReplicaSet:

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

The output includes the following status information:

```

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

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

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

```

CHAPTER 10. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS

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

10.1. MOVING THE CLUSTER LOGGING RESOURCES

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

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



NOTE

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

Prerequisites

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

Procedure

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

```
$ oc edit ClusterLogging instance

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

....

spec:
  collection:
    logs:
      fluentd:
        resources: null
        type: fluentd
  curation:
    curator:
      nodeSelector: 1
        node-role.kubernetes.io/infra: "
      resources: null
      schedule: 30 3 * * *
      type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 2
        node-role.kubernetes.io/infra: "
      redundancyPolicy: SingleRedundancy
      resources:
```

```

limits:
  cpu: 500m
  memory: 16Gi
requests:
  cpu: 500m
  memory: 16Gi
storage: {}
type: elasticsearch
managementState: Managed
visualization:
  kibana:
    nodeSelector: 3
      node-role.kubernetes.io/infra: " 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.

Verification steps

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

For example:

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

```

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

```

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

```

$ oc get nodes
NAME                                                    STATUS ROLES    AGE VERSION
ip-10-0-133-216.us-east-2.compute.internal             Ready master    60m v1.16.2
ip-10-0-139-146.us-east-2.compute.internal             Ready master    60m v1.16.2
ip-10-0-139-192.us-east-2.compute.internal             Ready worker    51m v1.16.2
ip-10-0-139-241.us-east-2.compute.internal             Ready worker    51m v1.16.2
ip-10-0-147-79.us-east-2.compute.internal              Ready worker    51m v1.16.2
ip-10-0-152-241.us-east-2.compute.internal             Ready master    60m v1.16.2
ip-10-0-139-48.us-east-2.compute.internal              Ready infra     51m v1.16.2

```

Note that the node has a **node-role.kubernetes.io/infra: "** label:

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

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

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

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

...

spec:

...

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

- ❶ ❷ Add a node selector to match the label in the node specification.

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

```
$ oc get pods

NAME                                READY STATUS   RESTARTS AGE
cluster-logging-operator-84d98649c4-zb9g7  1/1 Running    0      29m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg  2/2 Running    0      28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj  2/2 Running    0      28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78  2/2 Running    0      28m
fluentd-42dzz                            1/1 Running    0      28m
fluentd-d74rq                             1/1 Running    0      28m
fluentd-m5vr9                             1/1 Running    0      28m
fluentd-nkx17                             1/1 Running    0      28m
fluentd-pdvqb                             1/1 Running    0      28m
fluentd-tflh6                             1/1 Running    0      28m
kibana-5b8bdf44f9-ccpq9                   2/2 Terminating 0      4m11s
kibana-7d85dcffc8-bfpfp                   2/2 Running    0      33s
```

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

```
$ oc get pod kibana-7d85dcffc8-bfpfp -o wide
NAME                READY STATUS   RESTARTS AGE IP          NODE
NOMINATED NODE     READINESS GATES
kibana-7d85dcffc8-bfpfp 2/2   Running    0      43s 10.131.0.22 ip-10-0-139-48.us-
east-2.compute.internal <none>      <none>
```

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

```
$ oc get pods
NAME                                READY STATUS   RESTARTS AGE
cluster-logging-operator-84d98649c4-zb9g7 1/1   Running    0      30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg 2/2   Running    0      29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj 2/2   Running    0      29m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78 2/2   Running    0      29m
fluentd-42dzz                          1/1   Running    0      29m
fluentd-d74rq                          1/1   Running    0      29m
fluentd-m5vr9                          1/1   Running    0      29m
fluentd-nkx17                          1/1   Running    0      29m
fluentd-pdvqb                          1/1   Running    0      29m
fluentd-tflh6                          1/1   Running    0      29m
kibana-7d85dcffc8-bfpfp                2/2   Running    0      62s
```


CHAPTER 11. 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.

11.1. 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 12. 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.

12.1. 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
```

12.2. 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
```

12.3. 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 13. EXPORTED FIELDS

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

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

- `exported-fields-Default`
- `exported-fields-systemd`
- `exported-fields-kubernetes`
- `exported-fields-pipeline_metadata`
- `exported-fields-ovirt`
- `exported-fields-aushape`
- `exported-fields-tlog`

13.1. 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
<code>@timestamp</code>	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 <code>@</code> prefix convention to note a field as being reserved for a particular use. With Elasticsearch, most tools look for <code>@timestamp</code> by default. For example, the format would be <code>2015-01-24 14:06:05.071000</code> .
<code>geoip</code>	This is geo-ip of the machine.
<code>hostname</code>	The <code>hostname</code> 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.
<code>ipaddr4</code>	The IP address V4 of the source server, which can be an array.
<code>ipaddr6</code>	The IP address V6 of the source server, if available.

Parameter	Description
level	<p>The logging level as provided by rsyslog (severitytext property), python's logging module. Possible values are as listed at misc/sys/syslog.h plus trace and unknown. For example, "alert crit debug emerg err info notice trace unknown warning". Note that trace is not in the syslog.h list but many applications use it.</p> <p>. 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> <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 property is mapped to the service field.
tags	Optionally provided operator defined list of tags placed on each log by the collector or normalizer. The payload can be a string with whitespace-delimited string tokens, or a JSON list of string tokens.
file	Optional path to the file containing the log entry local to the collector TODO analyzer for file paths.
offset	The offset value can represent bytes to the start of the log line in the file (zero or one based), or log line numbers (zero or one based), as long as the values are strictly monotonically increasing in the context of a single log file. The values are allowed to wrap, representing a new version of the log file (rotation).
namespace_name	Associate this record with the namespace that shares it's name. This value will not be stored, but it is used to associate the record with the appropriate namespace for access control and visualization. Normally this value will be given in the tag, but if the protocol does not support sending a tag, this field can be used. If this field is present, it will override the namespace given in the tag or in kubernetes.namespace_name .
namespace_uuid	This is the uuid associated with the namespace_name . This value will not be stored, but is used to associate the record with the appropriate namespace for access control and visualization. If this field is present, it will override the uuid given in kubernetes.namespace_uuid . This will also cause the Kubernetes metadata lookup to be skipped for this log record.

collectd Fields

The following fields represent namespace metrics metadata.

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

collectd.processes Fields

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

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

collectd.processes.ps_disk_ops Fields

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

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

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

collectd.processes.ps_cputime Fields

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

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

collectd.processes.ps_count Fields

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

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

collectd.processes.ps_pagefaults Fields

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

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

collectd.processes.ps_disk_octets Fields

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

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

collectd.disk Fields

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

collectd.disk.disk_merged Fields

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

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

collectd.disk.disk_octets Fields

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

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

collectd.disk.disk_time Fields

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

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

collectd.disk.disk_ops Fields

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

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

collectd.disk.disk_io_time Fields

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

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

Parameter	Description
collectd.disk.disk_io_time_weighted_io_time	type: float TODO

collectd.interface Fields

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

collectd.interface.if_octets Fields

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

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

collectd.interface.if_packets Fields

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

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

collectd.interface.if_errors Fields

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

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

collectd.interface.if_dropped Fields

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

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

collectd.virt Fields

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

collectd.virt.if_octets Fields

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

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

collectd.virt.if_packets Fields

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

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

collectd.virt.if_errors Fields

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

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

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

collectd.virt.if_dropped Fields

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

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

collectd.virt.disk_ops Fields

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

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

collectd.virt.disk_octets Fields

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

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

Parameter	Description
collectd.virt.memory	type: float The collectd memory type of virt plug-in.
collectd.virt.virt_vcpu	type: float The collectd virt_vcpu type of virt plug-in.
collectd.virt.virt_cpu_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.
collectd.statsd.host_nic_tx	type: integer The collectd nic_tx type of statsd plug-in.
collectd.statsd.host_nic_rx_dropped	type: integer The collectd nic_rx_dropped type of statsd plug-in.
collectd.statsd.host_nic_tx_dropped	type: integer The collectd nic_tx_dropped type of statsd plug-in.
collectd.statsd.host_nic_rx_errors	type: integer The collectd nic_rx_errors type of statsd plug-in.
collectd.statsd.host_nic_tx_errors	type: integer The collectd nic_tx_errors type of statsd plug-in.

Parameter	Description
collectd.statsd.host_storage	type: integer The collectd storage type of statsd plug-in.
collectd.statsd.host_swap	type: integer The collectd swap type of statsd plug-in.
collectd.statsd.host_vdsm	type: integer The collectd VDSM type of statsd plug-in.
collectd.statsd.host_vms	type: integer The collectd VMS type of statsd plug-in.
collectd.statsd.vm_nic_tx_dropped	type: integer The collectd nic_tx_dropped type of statsd plug-in.
collectd.statsd.vm_nic_rx_bytes	type: integer The collectd nic_rx_bytes type of statsd plug-in.
collectd.statsd.vm_nic_tx_bytes	type: integer The collectd nic_tx_bytes type of statsd plug-in.
collectd.statsd.vm_balloon_min	type: integer The collectd balloon_min type of statsd plug-in.
collectd.statsd.vm_balloon_max	type: integer The collectd balloon_max type of statsd plug-in.
collectd.statsd.vm_balloon_target	type: integer The collectd balloon_target type of statsd plug-in.
collectd.statsd.vm_balloon_cur	type: integer The collectd balloon_cur type of statsd plug-in.
collectd.statsd.vm_cpu_sys	type: integer The collectd cpu_sys type of statsd plug-in.

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

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

collectd.postgresql Fields

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

Parameter	Description
collectd.postgresql.pg_n_tup_g	type: integer The collectd type pg_n_tup_g of plug-in postgresql.
collectd.postgresql.pg_n_tup_c	type: integer The collectd type pg_n_tup_c of plug-in postgresql.
collectd.postgresql.pg_n_umbackends	type: integer The collectd type pg_numbackends of plug-in postgresql.
collectd.postgresql.pg_xact	type: integer The collectd type pg_xact of plug-in postgresql.
collectd.postgresql.pg_db_size	type: integer The collectd type pg_db_size of plug-in postgresql.
collectd.postgresql.pg_blks	type: integer The collectd type pg_blks of plug-in postgresql.

13.2. SYSTEMD EXPORTED FIELDS

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

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

systemd.k Fields

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

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

systemd.t Fields

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

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

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

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

systemd.u Fields

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

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

13.3. KUBERNETES EXPORTED FIELDS

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

The namespace for Kubernetes-specific metadata. The **kubernetes.pod_name** is the name of the pod.

kubernetes.labels Fields

Labels attached to the OpenShift object are **kubernetes.labels**. Each label name is a subfield of labels field. Each label name is de-dotted, meaning dots in the name are replaced with underscores.

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

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

kubernetes.annotations Fields

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

13.4. CONTAINER EXPORTED FIELDS

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

pipeline_metadata.collector Fields

This section contains metadata specific to the collector.

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

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

pipeline_metadata.normalizer Fields

This section contains metadata specific to the normalizer.

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

13.5. OVIRT EXPORTED FIELDS

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

Namespace for oVirt metadata.

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

ovirt.engine Fields

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

13.6. AUSHAPE EXPORTED FIELDS

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

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

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

aushape.data Fields

Parsed audit event data related to Aushape.

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

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

13.7. TLOG EXPORTED FIELDS

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

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

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

CHAPTER 14. UNINSTALLING CLUSTER LOGGING

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

14.1. 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
```

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

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