OpenShift Container Platform 4.1

Logging

Configuring cluster logging in OpenShift Container Platform 4.1
Configuring cluster logging in OpenShift Container Platform 4.1
Abstract
This document provides instructions for installing, configuring, and using the cluster logging feature. Cluster logging aggregates logs for a range of OpenShift Container Platform services.
Table of Contents

CHAPTER 1. ABOUT CLUSTER LOGGING AND OPENShift CONTAINER PLATFORM .................................. 5
   1.1. ABOUT CLUSTER LOGGING ........................................... 5
       1.1.1. About cluster logging components ....................... 5
       1.1.2. About Elasticsearch ........................................ 5
       1.1.3. About Fluentd .............................................. 6
       1.1.4. About Kibana .............................................. 6
       1.1.5. About Curator ............................................. 6
       1.1.6. About Event Router ....................................... 7
       1.1.7. About the Cluster Logging Custom Resource .......... 7

CHAPTER 2. ABOUT DEPLOYING CLUSTER LOGGING .............................................................. 9
   2.1. ABOUT DEPLOYING AND CONFIGURING CLUSTER LOGGING ........................................ 9
       2.1.1. Configuring and Tuning Cluster Logging .................. 9
       2.1.2. Sample modified Cluster Logging Custom Resource .. 11
   2.2. STORAGE CONSIDERATIONS FOR CLUSTER LOGGING AND OPENShift CONTAINER PLATFORM .... 12
       2.3. ADDITIONAL RESOURCES ...................................... 12

CHAPTER 3. DEPLOYING CLUSTER LOGGING ........................................................................ 13
   3.1. INSTALLING THE CLUSTER LOGGING AND ELASTICSearch OPERATORS ..................... 13
       3.2. ADDITIONAL RESOURCES ...................................... 20

CHAPTER 4. VIEWING THE KIBANA INTERFACE .................................................................. 21
   4.1. LAUNCHING THE KIBANA INTERFACE ..................................... 21

CHAPTER 5. WORKING WITH EVENT ROUTER ................................................................. 22
   5.1. DEPLOYING AND CONFIGURING THE EVENT ROUTER ............................................. 22

CHAPTER 6. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT .................................. 26
   6.1. ABOUT CONFIGURING CLUSTER LOGGING .................................................. 26
       6.1.1. About deploying and configuring cluster logging ......... 26
       6.1.1.1. Configuring and Tuning Cluster Logging ............... 26
       6.1.1.2. Sample modified Cluster Logging Custom Resource .. 28
       6.1.2. Moving the cluster logging resources ................. 29
   6.2. CHANGING CLUSTER LOGGING MANAGEMENT STATE ........................................... 30
       6.2.1. Changing the cluster logging management state ........ 31
       6.2.2. Changing the Elastichsearch management state ....... 32
   6.3. CONFIGURING CLUSTER LOGGING ................................................. 33
       6.3.1. Understanding the cluster logging component images 34
   6.4. CONFIGURING ELASTICSearch TO STORE AND ORGANIZE LOG DATA ...................... 34
       6.4.1. Configuring Elastichsearch CPU and memory limits .... 35
       6.4.2. Configuring Elastichsearch replication policy ......... 36
       6.4.3. Configuring Elastichsearch storage ..................... 37
       6.4.4. Configuring Elastichsearch for emptyDir storage .... 37
       6.4.5. Exposing Elastichsearch as a route ..................... 38
       6.4.6. About Elastichsearch alerting rules ................. 41
   6.5. CONFIGURING KIBANA ........................................................................ 41
       6.5.1. Configure Kibana CPU and memory limits ............. 42
       6.5.2. Scaling Kibana for redundancy ............................ 42
       6.5.3. Installing the Kibana Visualize tool ..................... 43
   6.6. CURATION OF ELASTICSearch DATA ..................................................................... 44
       6.6.1. Configuring the Curator schedule ....................... 44
       6.6.2. Configuring Curator index deletion .................... 45
6.6.3. Troubleshooting Curator
6.6.4. Configuring Curator in scripted deployments
6.6.5. Using the Curator Action file

6.7. CONFIGURING FLUENTD
6.7.1. Viewing Fluentd pods
6.7.2. Viewing Fluentd logs
6.7.3. Configure Fluentd CPU and memory limits
6.7.4. Configuring Fluentd log location
6.7.5. Throttling Fluentd logs
6.7.6. Understanding Buffer Chunk Limiting for Fluentd
6.7.7. Configuring Fluentd JSON parsing
6.7.8. Configuring how the log collector normalizes logs
6.7.9. Configuring Fluentd using environment variables

6.8. SENDING OPENSHIFT CONTAINER PLATFORM LOGS TO EXTERNAL DEVICES
6.8.1. Configuring Fluentd to send logs to an external Elasticsearch instance
6.8.2. Configuring Fluentd to send logs to an external syslog server
6.8.3. Configuring Fluentd to send logs to an external log aggregator

6.9. CONFIGURING SYSTEMD-JOURNALD AND RSYSLOG

CHAPTER 7. VIEWING ELASTICSEARCH STATUS
7.1. VIEWING ELASTICSEARCH STATUS
7.1.1. Example condition messages
7.2. VIEWING ELASTICSEARCH COMPONENT STATUS

CHAPTER 8. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS
8.1. SPECIFYING A NODE FOR CLUSTER LOGGING COMPONENTS USING NODE SELECTORS

CHAPTER 9. ADDITIONAL RESOURCES

CHAPTER 10. MANUALLY ROLLING OUT ELASTICSEARCH
10.1. PERFORMING AN ELASTICSEARCH ROLLING CLUSTER RESTART

CHAPTER 11. TROUBLESHOOTING KIBANA
11.1. TROUBLESHOOTING A KUBERNETES LOGIN LOOP
11.2. TROUBLESHOOTING A KUBERNETES CRYPTIC ERROR WHEN VIEWING THE KIBANA CONSOLE
11.3. TROUBLESHOOTING A KUBERNETES 503 ERROR WHEN VIEWING THE KIBANA CONSOLE

CHAPTER 12. EXPORTED FIELDS
12.1. DEFAULT EXPORTED FIELDS
Top Level Fields
collectd Fields
collectd.processes Fields
collectd.processes.ps_disk_ops Fields
collectd.processes.ps_cputime Fields
collectd.processes.ps_count Fields
collectd.processes.ps_pagefaults Fields
collectd.processes.ps_disk_octets Fields
collectd.disk Fields
collectd.disk.disk_merged Fields
collectd.disk.disk_octets Fields
collectd.disk.disk_time Fields
collectd.disk.disk_ops Fields
collectd.disk.disk_io_time Fields
collectd.interface Fields
1. ABOUT CLUSTER LOGGING AND OPENSHIFT CONTAINER PLATFORM

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

1.1. ABOUT CLUSTER LOGGING

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

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

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

1.1.1. About cluster logging components

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

There are currently 4 different types of cluster logging components:

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

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

1.1.2. About Elasticsearch

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

Elasticsearch subdivides each index into multiple pieces called shards, which it spreads across a set of Elasticsearch nodes in an Elasticsearch cluster. You can configure Elasticsearch to make copies of the shards, called replicas. Elasticsearch also spreads these replicas across the Elasticsearch nodes. The ClusterLogging Custom Resource allows you to specify the replication policy in the Custom Resource Definition (CRD) to provide data redundancy and resilience to failure.
The Cluster Logging Operator and companion Elasticsearch Operator ensure that each Elasticsearch node is deployed using a unique Deployment that includes its own storage volume. You can use a Cluster Logging Custom Resource (CR) to increase the number of Elasticsearch nodes. Refer to Elastic’s documentation for considerations involved in choosing storage and network location as directed below.

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

For more information, see Elasticsearch (ES).

1.1.3. About Fluentd

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

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

Fluentd uses journald as the system log source. These are log messages from the operating system, the container runtime, and OpenShift Container Platform.

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

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

For more information, see Fluentd.

1.1.4. About Kibana

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

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

For more information, see Kibana.

1.1.5. About Curator

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

For more information, see Curator.

1.1.6. About Event Router

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

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

1.1.7. About the Cluster Logging Custom Resource Definition

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

You should never have to modify this CRD. To make changes to your deployment, create and modify a specific Custom Resource (CR). Instructions for creating or modifying a CR are provided in this documentation as appropriate.

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

Sample Cluster Logging CR

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 2
      resources:
        limits:
          memory: 2Gi
          cpu: 200m
          memory: 2Gi
        requests:
          cpu: 200m
          memory: 2Gi
    storage:
      storageClassName: "gp2"
      size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
```
type: "kibana"
kibana:
  resources:
    limits:
      memory: 1Gi
    requests:
      cpu: 500m
      memory: 1Gi
proxy:
  resources:
    limits:
      memory: 100Mi
    requests:
      cpu: 100m
      memory: 100Mi
replicas: 2
curation:
  type: "curator"
curator:
  resources:
    limits:
      memory: 200Mi
    requests:
      cpu: 200m
      memory: 200Mi
    schedule: "/10 * * *
collection:
  logs:
    type: "fluentd"
fluentd:
  resources:
    limits:
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 1Gi
CHAPTER 2. ABOUT DEPLOYING CLUSTER LOGGING

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

2.1. ABOUT DEPLOYING AND CONFIGURING CLUSTER LOGGING

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

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

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

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

2.1.1. Configuring and Tuning Cluster Logging

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

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

Memory and CPU

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

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

Elasticsearch storage

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

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

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

**NOTE**

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

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

Elasticsearch replication policy

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

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

**Curator schedule**
You specify the schedule for Curator in the [cron format](https://en.wikipedia.org/wiki/Cron).

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

### 2.1.2. Sample modified Cluster Logging Custom Resource

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

**Sample modified Cluster Logging Custom Resource**

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

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

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

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

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

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

NOTE

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

2.3. ADDITIONAL RESOURCES

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

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

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

3.1. INSTALLING THE CLUSTER LOGGING AND ELASTICSEARCH OPERATORS

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

NOTE

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

Prerequisites

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

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

NOTE

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

Procedure


NOTE

You can also create the Namespaces in the web console using the Administration → Namespaces page. You must apply the cluster-logging and cluster-monitoring labels listed in the sample YAML to the namespaces you create.
a. Create a Namespace for the Elasticsearch Operator (for example, `eo-namespac.yaml`):

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

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

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

b. Run the following command to create the namespace:

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

For example:

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

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

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

1 2 3 Specify these labels as shown.

d. Run the following command to create the namespace:

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

For example:

```bash
$ oc create -f clo-namespac.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:

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

   b. Create an Operator Group object:

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

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

   **Example CatalogSourceConfig**

   ```yaml
   apiVersion: "operators.coreos.com/v1"
   kind: "CatalogSourceConfig"
   metadata:
     name: "elasticsearch"
     namespace: "openshift-marketplace"
   spec:
     targetNamespace: "openshift-operators-redhat"
     packages: ["elasticsearch-operator"]
     source: "redhat-operators"
   ```
   
   1 You must specify the `openshift-operators-redhat` namespace.

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

d. Create a CatalogSourceConfig object:

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

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

   ```bash
   $ oc get packagemanifest elasticsearch-operator -n openshift-marketplace -o jsonpath='{.status.channels[0].name}'
   ```

   ```bash
   preview
   ```

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

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

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

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

g. Create the Subscription object:

$ oc create -f eo-sub.yaml

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

$ oc project openshift-operators-redhat
Now using project "openshift-operators-redhat"

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

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

j. Create the RBAC object:

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

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

3. Install the Cluster Logging Operator using the OpenShift Container Platform web console for best results:

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

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

c. On the Create Operator Subscription page, under A specific namespace on the cluster select openshift-logging. Then, click Subscribe.

4. Verify the operator installations:

a. Switch to the Catalog → Installed Operators page.

b. Ensure that Cluster Logging is listed in the openshift-logging project with a Status of InstallSucceeded.

c. Ensure that Elasticsearch Operator is listed in the openshift-operator-redhat project with a Status of InstallSucceeded. The Elasticsearch Operator is copied to all other projects.

NOTE

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

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

- Switch to the Catalog → Operator Management page and inspect the Operator Subscriptions and Install Plans tabs for any failure or errors under Status.

- Switch to the Workloads → Pods page and check the logs in any Pods in the openshift-logging and openshift-operators-redhat projects that are reporting issues.

5. Create a cluster logging instance:

a. Switch to the Administration → Custom Resource Definitions page.

c. On the **Custom Resource Definition Overview** page, select **View Instances** from the **Actions** menu.

d. On the **Cluster Loggings** page, click **Create Cluster Logging**. You might have to refresh the page to load the data.

e. In the YAML, replace the code with the following:

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

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

2. The cluster logging management state. In most cases, if you change the default cluster logging defaults, you must set this to **Unmanaged**. However, an unmanaged deployment does not receive updates until the cluster logging is placed back into a managed state. For more information, see **Changing cluster logging management state**.

3. Settings for configuring Elasticsearch. Using the CR, you can configure shard replication policy and persistent storage. For more information, see **Configuring Elasticsearch**.
Specify the number of Elasticsearch nodes. See the note that follows this list.

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

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

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

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

**NOTE**

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

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

```
$ oc get deployment

cluster-logging-operator       1/1     1            1           18h
elasticsearch-cd-x6dekli-1    0/1     1            0           6m54s
elasticsearch-cdm-x6dekli-1   1/1     1            1           18h
elasticsearch-cdm-x6dekli-2   0/1     1            0           6m49s
elasticsearch-cdm-x6dekli-3   0/1     1            0           6m44s
```

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

6. Verify the install:

a. Switch to the **Workloads → Pods** page.

b. Select the **openshift-logging** project.

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

- cluster-logging-operator-cb795f8dc-xkckc
- elasticsearch-cdm-b3nqzchd-1-5c6797-67kfz
- elasticsearch-cdm-b3nqzchd-2-6657f4-wtprv
• elasticsearch-cdm-b3nqzchd-3-588c65-clg7g
• fluentd-2c7dg
• fluentd-9z7kk
• fluentd-br7r2
• fluentd-fn2sb
• fluentd-pb2f8
• fluentd-zqgqx
• kibana-7fb4fd4cc9-bvt4p

3.2. ADDITIONAL RESOURCES

For more information on installing operators, see Installing Operators from the OperatorHub.
CHAPTER 4. VIEWING THE KIBANA INTERFACE

The cluster logging installation deploys the Kibana interface.

4.1. LAUNCHING THE KIBANA INTERFACE

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

Procedure

To launch the Kibana interface:

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

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

Use and configuration of the Kibana interface is beyond the scope of this documentation. For more information, on using the interface, see the Kibana documentation.
CHAPTER 5. WORKING WITH EVENT ROUTER

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

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

5.1. DEPLOYING AND CONFIGURING THE EVENT ROUTER

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

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

Prerequisites

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

- Set TRANSFORM_EVENTS=true in order to process and store event router events in Elasticsearch.
  
  - Set cluster logging to the unmanaged state.
  
  - Enable the TRANSFORM_EVENTS feature.

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

Procedure

1. Create a template for the Event Router:

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

22
apiVersion: v1
metadata:
  name: event-reader-binding
subjects:
- kind: ServiceAccount
  name: eventrouter
  namespace: ${NAMESPACE}
roleRef:
  kind: ClusterRole
  name: event-reader
- kind: ConfigMap
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
data:
  config.json: |-
    
    "sink": "stdout"
  
- kind: Deployment
  apiVersion: apps/v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  labels:
    component: eventrouter
    logging-infra: eventrouter
    provider: openshift
  spec:
    selector:
      matchLabels:
        component: eventrouter
        logging-infra: eventrouter
        provider: openshift
    replicas: 1
    template:
      metadata:
        labels:
          component: eventrouter
          logging-infra: eventrouter
          provider: openshift
        name: eventrouter
      spec:
        serviceAccount: eventrouter
        containers:
        - name: kube-eventrouter
          image: ${IMAGE}
          imagePullPolicy: IfNotPresent
          resources:
            limits:
              memory: ${MEMORY}
            requests:
              cpu: ${CPU}
              memory: ${MEMORY}
          volumeMounts:
Creates a Service Account for the Event Router.

Creates a cluster role to monitor for events in the cluster.

Allows the get, watch, and list permissions for the events resource.

Creates a ClusterRoleBinding to bind the ClusterRole to the ServiceAccount.

Specify the image version for the Event Router.

Specify the memory limit for the Event Router pods. Defaults to '128Mi'.

Specify the minimum amount of CPU to allocate to the Event Router. Defaults to '100m'.

Specify the namespace where eventrouter is deployed. Defaults to openshift-logging. The value must be the same as specified for the ServiceAccount and ClusterRoleBinding. The project indicates where in Kibana you can locate events:

- If the event router pod is deployed in a default project, such as kube-* and openshift-*, you can find the events under the .operation index.

- If the event router pod is deployed in other projects, you can find the event under the index using the project namespace.

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

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

For example:

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

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

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

   $ oc logs logging-eventrouter-d649f97c8-qvv8r

   {"verb":"ADDED","event":{"metadata":{"name":"elasticsearch-operator.v0.0.1.158f402e25397146","namespace":"openshift-operators","selfLink":"/api/v1/namespaces/openshift-operators/events/elasticsearch-operator.v0.0.1.158f402e25397146","uid":"37b7ff11-4f1a-11e9-a7ad-0271b2ca69f0","resourceVersion":"523264","creationTimestamp":"2019-03-25T16:22:43Z"},"involvedObject":{"kind":"ClusterServiceVersion","namespace":"openshift-operators","name":"elasticsearch-operator.v0.0.1","uid":"27b2ca6d-4f1a-11e9-8fba-0ea949ad61f6","apiVersion":"operators.coreos.com/v1alpha1","resourceVersion":"523096"},"reason":"InstallSucceeded","message":"waiting for install components to report healthy","source":{"component":"operator-lifecycle-manager"},"firstTimestamp":"2019-03-25T16:22:43Z","lastTimestamp":"2019-03-25T16:22:43Z","count":1,"type":"Normal"}
CHAPTER 6. CONFIGURING YOUR CLUSTER LOGGING DEPLOYMENT

6.1. ABOUT CONFIGURING CLUSTER LOGGING

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

NOTE

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

6.1.1. About deploying and configuring cluster logging

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

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

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

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

6.1.1.1. Configuring and Tuning Cluster Logging

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

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

Memory and CPU

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

```yaml
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu: 1
          memory: 16Gi
        requests:
          cpu: 1
          memory: 16Gi
      type: "elasticsearch"
  collection:
    logs:
      fluentd:
```
resources:
  limits:
    cpu:
    memory:
  requests:
    cpu:
    memory:
  type: "fluentd"
visualization:
kibana:
  resources:
    limits:
    cpu:
    memory:
  requests:
    cpu:
    memory:
  type: kibana
curation:
curator:
  resources:
    limits:
      memory: 200Mi
    requests:
      cpu: 200m
      memory: 200Mi
  type: "curator"

Elasticsearch storage

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

```markdown
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      storage:
        storageClassName: "gp2"
        size: "200G"
```

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

**NOTE**

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

```markdown
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      storage: {}
```
Elasticsearch replication policy

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

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

Curator schedule

You specify the schedule for Curator in the [cron format](https://en.wikipedia.org/wiki/Cron).

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

6.1.1.2. Sample modified Cluster Logging Custom Resource

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

Sample modified Cluster Logging Custom Resource

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 2
      resources:
        limits:
          memory: 2Gi
        requests:
          cpu: 200m
          memory: 2Gi
      storage: {}
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
kibana:
  resources:
```
6.1.2. Moving the cluster logging resources

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

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

NOTE

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

Prerequisites

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

Procedure

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

   ```
   $ oc edit ClusterLogging instance
   ...
   ```

   ```
   apiVersion: logging.openshift.io/v1
   kind: ClusterLogging
   ...
   ```
spec:
collection:
  logs:
    fluentd:
      resources: null
    rsyslog:
      resources: null
      type: fluentd
curation:
curator:
  nodeSelector: 1
    node-role.kubernetes.io/infra: "
  resources: null
  schedule: 30 3 * * *
  type: curator
logStore:
elasticsearch:
  nodeCount: 3
  nodeSelector: 2
    node-role.kubernetes.io/infra: "
  redundancyPolicy: SingleRedundancy
  resources:
    limits:
      cpu: 500m
      memory: 16Gi
    requests:
      cpu: 500m
      memory: 16Gi
  storage: {}
  type: elasticsearch
managementState: Managed
visualization:
kibana:
  nodeSelector: 3
    node-role.kubernetes.io/infra: "
  proxy:
    resources: null
    replicas: 1
    resources: null
    type: kibana

6.2. CHANGING CLUSTER LOGGING MANAGEMENT STATE

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

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

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

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

**NOTE**

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

6.2.1. Changing the cluster logging management state

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

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

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

**NOTE**

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

**Prerequisites**

- The Cluster Logging Operator must be installed.

**Procedure**

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

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

Specify the management state as Managed or Unmanaged.

6.2.2. Changing the Elasticsearch management state

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

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

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

Prerequisite

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

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

Procedure

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

```
$ oc edit Elasticsearch elasticsearch

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

...

spec:
  managementState: "Managed" ①

① Specify the management state as Managed or Unmanaged.
```
NOTE

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

6.3. CONFIGURING CLUSTER LOGGING

Cluster logging is configurable using a Cluster Logging Custom Resource (CR) deployed in the openshift-logging project.

The Cluster Logging Operator watches for changes to Cluster Logging CRs, creates any missing logging components, and adjusts the logging deployment accordingly.

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

Sample Cluster Logging Custom Resource (CR)

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

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

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

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

### 6.3.1. Understanding the cluster logging component images

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

You can view the images by running the following command:

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

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

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

**NOTE**

The values might be different depending on your environment.

### 6.4. CONFIGURING ELASTICSEARCH TO STORE AND ORGANIZE LOG DATA

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

You can configure your Elasticsearch deployment to:
• configure storage for your Elasticsearch cluster;
• define how shards are replicated across data nodes in the cluster, from full replication to no replication;
• configure external access to Elasticsearch data.

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

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

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

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

### 6.4.1. Configuring Elasticsearch CPU and memory limits

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

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

**Prerequisites**

• Cluster logging and Elasticsearch must be installed.

**Procedure**

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

```sh
$ oc edit ClusterLogging instance
```

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  ....
spec:
```
Specify the CPU and memory limits as needed. If you leave these values blank, the Elasticsearch Operator sets default values that should be sufficient for most deployments.

6.4.2. Configuring Elasticsearch replication policy

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

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

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

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

spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      redundancyPolicy: "SingleRedundancy"
```

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

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

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

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

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

### 6.4.3. Configuring Elasticsearch storage

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

***WARNING***

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

#### Prerequisites

- Cluster logging and Elasticsearch must be installed.

#### Procedure

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

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

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

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

**NOTE**

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

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

1. Edit the Cluster Logging CR to specify emptyDir:

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

**6.4.5. Exposing Elasticsearch as a route**

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

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

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

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

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

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

```
```

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

**Prerequisites**
Cluster logging and Elasticsearch must be installed.

You must have access to the project in order to be able to access to the logs. For example:

**Procedure**

**To expose Elasticsearch externally:**

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

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

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

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

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

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

   ✅ Add the Elasticsearch CA certificate or use the command in the next step. You do not have to set the `spec.tls.key`, `spec.tls.certificate`, and `spec.tls.caCertificate` parameters required by some reencrypt routes.

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

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

   c. Create the route:

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

4. Check that the Elasticsearch service is exposed:
   
   a. Get the token of this ServiceAccount to be used in the request:
b. Set the `elasticsearch` route you created as an environment variable.

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

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

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

The response appears similar to the following:

```json
{
  "took": 441,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 89157,
    "max_score": 1,
    "hits": [
      {
        "_index": ".operations.2019.03.15",
        "_type": "com.example.viaq.common",
        "_id": "ODdiNWIyYzAtMjg5Ni0TAtNWE3MDY1MjMzNTc3",
        "_score": 1,
        "_source": {
          ".SOURCE_MONOTONIC_TIMESTAMP": "673396",
          "systemd": {
            "t": {
              "BOOT_ID": "246c34ee9cdeecb41a608e94",
              "MACHINE_ID": "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:xxxx:fe4c:5b09",
```
6.4.6. About Elasticsearch alerting rules

You can view these alerting rules in Prometheus.

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

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

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

**NOTE**

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

### 6.5.1. Configure Kibana CPU and memory limits

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

**Procedure**

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

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

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

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

### 6.5.2. Scaling Kibana for redundancy
You can scale the Kibana deployment for redundancy.

Procedure

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

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

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

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

Specify the number of Kibana nodes.

6.5.3. Installing the Kibana Visualize tool

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

Procedure

To load dashboards and other Kibana UI objects:

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

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

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

2. Get the name of your Elasticsearch pods.

```bash
$ 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-k7kj    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.

     ![Image](https://kibana-openshift-logging.apps.openshift.com)

     Where the URL is Kibana route.

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

   c. Log out of the Kibana dashboard.

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

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

   For example:

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

### 6.6. CURATION OF ELASTICSEARCH DATA

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

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

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

#### 6.6.1. Configuring the Curator schedule

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

**Prerequisites**

- Cluster logging and Elasticsearch must be installed.

**Procedure**

To configure the Curator schedule:
1. Edit the Cluster Logging Custom Resource in the `openshift-logging` project:

```
$ oc edit clusterlogging instance
```

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

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

1 Specify the schedule for Curator in cron format.

**NOTE**

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

### 6.6.2. Configuring Curator index deletion

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

**Prerequisite**

- Cluster logging must be installed.

**Procedure**

To delete indices:

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

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

2. Set the following parameters as needed:

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

The available parameters are:

**Table 6.1. Project options**
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>project_name</strong></td>
<td>The actual name of a project, such as <code>myapp-devel</code>. For OpenShift Container Platform operations logs, use the name <code>.operations</code> as the project name.</td>
</tr>
<tr>
<td><strong>action</strong></td>
<td>The action to take, currently only <code>delete</code> is allowed.</td>
</tr>
<tr>
<td><strong>unit</strong></td>
<td>The period to use for deletion, <code>days</code>, <code>weeks</code>, or <code>months</code>.</td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>The number of units.</td>
</tr>
</tbody>
</table>

Table 6.2. Filter options

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

For example, to configure Curator to:

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

Use:

```yaml
config.yaml: |
  .defaults:
    delete:
      days: 31
  .operations:
    delete:
      weeks: 8
  myapp-dev:
    delete:
```
IMPORTANT

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

6.6.3. Troubleshooting Curator

You can use information in this section for debugging Curator. For example, if curator is in failed state,
but the log messages do not provide a reason, you could increase the log level and trigger a new job,
instead of waiting for another scheduled run of the cron job.

Prerequisites

Cluster logging and Elasticsearch must be installed.

Procedure

Enable the Curator debug log and trigger next Curator iteration manually

1. Enable debug log of Curator:

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

   Specify the log level:

   - **CRITICAL.** Curator displays only critical messages.

   - **ERROR.** Curator displays only error and critical messages.

   - **WARNING.** Curator displays only error, warning, and critical messages.

   - **INFO.** Curator displays only informational, error, warning, and critical messages.

   - **DEBUG.** Curator displays only debug messages, in addition to all of the above.
     The default value is INFO.
NOTE

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

1. Trigger next curator iteration:

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

2. Use the following commands to control the CronJob:

   - Suspend a CronJob:
     ```
     $ oc patch cronjob curator -p '{"spec":{"suspend":true}}'
     ```
   - Resume a CronJob:
     ```
     $ oc patch cronjob curator -p '{"spec":{"suspend":false}}'
     ```
   - Change a CronJob schedule:
     ```
     $ oc patch cronjob curator -p '{"spec":{"schedule":"0 0 * * *"}}'
     ```

   The `schedule` option accepts schedules in cron format.

6.6.4. Configuring Curator in scripted deployments

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

Prerequisites

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

Procedure

Use the following snippets to configure Curator in your scripts:

- For scripted deployments

  1. Create and modify the configuration:

     a. Copy the Curator configuration file and the OpenShift Container Platform custom configuration file from the Curator configuration map and create separate files for each:

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

     b. Edit the `/my/config/curator5.yaml` and `/my/config/config.yaml` files.
2. Delete the existing Curator config map and add the edited YAML files to a new Curator config map.

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

The next iteration will use this configuration.

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

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

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

The next iteration will use this configuration.

### 6.6.5. Using the Curator Action file

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

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

**IMPORTANT**

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

**Prerequisite**

- Cluster logging and Elasticsearch must be installed.
- Set cluster logging to the unmanaged state.
Procedure
To configure Curator to delete indices:

1. Edit the Curator ConfigMap:

   
   oc edit cm/curator -n openshift-logging

2. Make the following changes to the action file:

   
   actions:
   
   1:
   
   - action: delete_indices
     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
     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
     - filertype: pattern
       kind: regex
       value: '^\..operations\..*$'
       exclude: False
     - filertype: age
       # Parse timestamp from index name
       source: name
       direction: older
       timestring: '%Y.%m.%d'
       unit: days
       unit_count: 30
       exclude: False

   
   Specify delete_indices to delete the specified index.

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

   Specify false to allow the index to be deleted.

6.7. CONFIGURING FLUENTD

OpenShift Container Platform uses Fluentd to collect operations and application logs from your cluster which OpenShift Container Platform enriches with Kubernetes Pod and Namespace metadata.
You can configure log rotation, log location, use an external log aggregator, and make other configurations.

**NOTE**

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

### 6.7.1. Viewing Fluentd pods

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

**Procedure**

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

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

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

### 6.7.2. Viewing Fluentd logs

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

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

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

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

For example:

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

To view the current setting:

```bash
oc -n openshift-logging set env daemonset/fluentd --list | grep LOGGING_FILE_PATH
```
- If you are using \texttt{LOGGING_FILE_PATH=console}, Fluentd writes logs to stdout/stderr. You can retrieve the logs with the \texttt{oc logs [-f] <pod_name>} command, where the \texttt{-f} is optional, from the project where the pod is located.

\begin{verbatim}
$ oc logs -f <any-fluentd-pod> \textsuperscript{1}
\end{verbatim}

\textsuperscript{1} Specify the name of a Fluentd pod. Use the \texttt{-f} option to follow what is being written into the logs.

For example

\begin{verbatim}
$ oc logs -f fluentd-ht42r -n openshift-logging
\end{verbatim}

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

\subsection*{6.7.3. Configure Fluentd CPU and memory limits}

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

**Procedure**

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

\begin{verbatim}
$ oc edit ClusterLogging instance
\end{verbatim}

\begin{verbatim}
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"

spec:
  collection:
    logs:
      fluentd:
        resources:
          limits:
            \textsuperscript{1} cpu: 250m
            memory: 1Gi
          requests:
            cpu: 250m
            memory: 1Gi
\end{verbatim}

\textsuperscript{1} Specify the CPU and memory limits as needed. The values shown are the default values.

\subsection*{6.7.4. Configuring Fluentd log location}

Fluentd writes logs to a specified file or to the default location, \texttt{/var/log/fluentd/fluentd.log}, based on the \texttt{LOGGING_FILE_PATH} environment variable.
Prerequisite
Set cluster logging to the unmanaged state.

Procedure
To set the output location for the Fluentd logs:

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

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

2. Specify the log output method:

- **use** `console` to use the Fluentd default location. Retrieve the logs with the `oc logs [-f] <pod_name>` command.

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

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

6.7.5. Throttling Fluentd logs

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

**WARNING**

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

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

Prerequisite

Set cluster logging to the unmanaged state.

Procedure

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

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

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

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

6.7.6. Understanding Buffer Chunk Limiting for Fluentd

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

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

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

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

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

```
$ egrep "buffer_type|buffer_path" *.conf
output-es-config.conf:
  buffer_type file
  buffer_path `/var/lib/fluentd/buffer-output-es-config`
output-es-ops-config.conf:
  buffer_type file
  buffer_path `/var/lib/fluentd/buffer-output-es-ops-config`
```
The Fluentd buffer_queue_limit is the value of the variable BUFFER_QUEUE_LIMIT. This value is 32 by default.

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

If the BUFFER_QUEUE_LIMIT variable has the default set of values:

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

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

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

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

### 6.7.7. Configuring Fluentd JSON parsing

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

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

**IMPORTANT**

Enabling this feature comes with risks, including:

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

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

**Prerequisites**

Set cluster logging to the unmanaged state.

**Procedure**
Use the following command to enable this feature:

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

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

### Setting MERGE_JSON_LOG and CDM_UNDEFINED_TO_STRING

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

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

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

For example:

```yaml
data:
  fluent.conf:
    ....

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

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

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

### 6.7.8. Configuring how the log collector normalizes logs

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

- There must be a "**message**" field containing the actual log message.
- There must be a "**@timestamp**" field containing the log record timestamp in RFC 3339 format, preferably millisecond or better resolution.
There must be a "level" field with the log level, such as err, info, unknown, and so forth.

**NOTE**

For more information on the data model, see Exported Fields.

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

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

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

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

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

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

- Simple scalar values like numbers and booleans are changed to a quoted string. For example: 10 becomes "10", 3.1415 becomes "3.1415", false becomes "false".

- Map/dict values and array values are converted to their JSON string representation: "mapfield":{"key":"value"} becomes "mapfield":{"key":"value"} and "arrayfield": [1,2,"three"] becomes "arrayfield": [1,2,"three"].

- **Defined fields.** You can also configure which defined fields appear in the top levels of the logs. The default top-level fields, defined through the CDM_DEFAULT_KEEP_FIELDS parameter, are CEE, time, @timestamp, aushape, ci_job, collectd, docker, fedora-ci, file, foreman, geoip, hostname, ipaddr4, ipaddr6, kubernetes, level, message, namespace_name, namespace_uuid, offset, openstack, ovirt, pid, pipeline_metadata, rsyslog, service, systemd, tags, testcase, tlog, viaq_msg_id.

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

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

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

### Table 6.3. Environment parameters for log normalization

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

**NOTE**

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

**NOTE**

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

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

**Procedure**

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

1. Configure how to process fields, as needed:
   a. Specify the fields to move using CDM_EXTRA_KEEP_FIELDS.
   b. Specify any empty fields to retain in the CDM_KEEP_EMPTY_FIELDS parameter in CSV format.

2. Configure how to process undefined fields, as needed:
   a. Set CDM_USE_UNDEFINED to `true` to move undefined fields to the top-level undefined field:
   b. Specify a name for the undefined fields using the CDM_UNDEFINED_NAME parameter.
   c. Set CDM_UNDEFINED_MAX_NUM_FIELDS to a value other than the default `-1`, to set an upper bound on the number of undefined fields in a single record.

3. Specify CDM_UNDEFINED_DOT_REPLACE_CHAR to change any dot `.` characters in an undefined field name to another character. For example, if CDM_UNDEFINED_DOT_REPLACE_CHAR=@@@ and there is a field named foo.bar.baz the field is transformed into foo@@@bar@@@baz.

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

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

6.7.9. Configuring Fluentd using environment variables

You can use environment variables to modify your Fluentd configuration.

Prerequisite
Set cluster logging to the unmanaged state.

Procedure
Set any of the Fluentd environment variables as needed:

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

For example:

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

6.8. SENDING OPENSIFHT CONTAINER PLATFORM LOGS TO EXTERNAL DEVICES

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

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

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

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

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

Prerequisite

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

Procedure

To direct logs to a specific Elasticsearch instance:

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

```bash
$ oc edit ds/fluentd
```

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

2. Set `ES_HOST` and `OPS_HOST` to the same destination, while ensuring that `ES_PORT` and `OPS_PORT` also have the same value for an external Elasticsearch instance to contain both application and operations logs.

3. Configure your externally-hosted Elasticsearch instance for TLS. Only externally-hosted Elasticsearch instances that use Mutual TLS are allowed.
NOTE

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

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

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

Prerequisite

Set cluster logging to the unmanaged state.

Procedure

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

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

The desired remote syslog host. Required for each host.

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

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

Fluentd Environment Variables

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE_REMOTE_SYSLOG</td>
<td>Defaults to false. Set to true to enable use of the fluent-plugin-remote-syslog gem</td>
</tr>
<tr>
<td>REMOTE_SYSLOG_HOST</td>
<td>(Required) Hostname or IP address of the remote syslog server.</td>
</tr>
<tr>
<td>REMOTE_SYSLOG_PORT</td>
<td>Port number to connect on. Defaults to 514.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SYSLOG_SEVERITY</td>
<td>Set the syslog severity level. Defaults to <code>debug</code>.</td>
</tr>
<tr>
<td>REMOTE_SYSLOG_FACILITY</td>
<td>Set the syslog facility. Defaults to <code>local0</code>.</td>
</tr>
<tr>
<td>REMOTE_SYSLOG_USE_RECORD</td>
<td>Defaults to <code>false</code>. Set to <code>true</code> to use the record’s severity and facility fields to set on the syslog message.</td>
</tr>
<tr>
<td>REMOTE_SYSLOG_REMOVE_TAG_PREFIX</td>
<td>Removes the prefix from the tag, defaults to &quot; (empty).</td>
</tr>
<tr>
<td>REMOTE_SYSLOG_TAG_KEY</td>
<td>If specified, uses this field as the key to look on the record, to set the tag on the syslog message.</td>
</tr>
<tr>
<td>REMOTE_SYSLOG_PAYLOAD_KEY</td>
<td>If specified, uses this field as the key to look on the record, to set the payload on the syslog message.</td>
</tr>
<tr>
<td>REMOTE_SYSLOG_TYPE</td>
<td>Set the transport layer protocol type. Defaults to <code>syslog_buffered</code>, which sets the TCP protocol. To switch to UDP, set this to <code>syslog</code>.</td>
</tr>
</tbody>
</table>

**WARNING**

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

---

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

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

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

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

**Default secure-forward.conf section**

```bash
# <store>
```
Procedure

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

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

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

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

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

For example:
Optionally, enter a name for this external aggregator.

Specify the host name or IP of the external aggregator.

Specify the port of the external aggregator.

Optionally, add additional external aggregator.

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

```
#   <security>
#     self_hostname $(hostname) # $(hostname) is a placeholder. 1
#     shared_key <shared_key_between_forwarder_and_forwarder> 2
#   </security>

#   tls_cert_path /path/for/certificate/ca_cert.pem 3
```

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

2. Specify a shared key for authentication.

3. Specify the path to your CA certificate.

For example:

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

tls_cert_path /etc/fluent/keys/ca.crt
```

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

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

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

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

For example:

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

5. Configure the secure-forward.conf file on the external aggregator to accept messages securely from Fluentd.
   When configuring the external aggregator, it must be able to accept messages securely from Fluentd.

You can find further explanation of how to set up the inforward plugin and the out_forward plugin.

6.9. CONFIGURING SYSTEMD-JOURNALD AND RSYSLG

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

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

6.9.1. Configuring systemd-journald for cluster logging

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

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

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

Procedure

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

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

Specify the maximum time to store journal entries. Enter digits to specify seconds. Or include a unit: "year", "month", "week", "day", "h" or "m". Enter **0** to disable. The default is **1month**.

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.

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.

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

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

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

Specify the maximum size for individual journal files stored persistently in `/var/log/journal`. The default is **10M**.
NOTE

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

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

2. Convert the journal.conf file to base64:

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

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

   For example:

   ```yaml
   ... 
   config: 
     storage: 
       files: 
         - contents: 
             source: data:text/plain;charset=utf-8;base64,${jrnl_cnf} 
             verification: {} 
             filesystem: root 
             mode: 0644 
             path: /etc/systemd/journald.conf 
             systemd: {} 
   1 Set the permissions for the journal.conf file. It is recommended to set 0644 permissions.
   2 Specify the path to the base64-encoded journal.conf file.
   ```

4. Create the MachineConfig:

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

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

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

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

   Name: worker 
   Namespace: 
   Labels: machineconfiguration.openshift.io/mco-built-in= 
   Annotations: <none> 
   API Version: machineconfiguration.openshift.io/v1 
   Kind: MachineConfigPool
Conditions:
Message:
Reason: All nodes are updating to rendered-worker-913514517bcea7c93bd446f4830bc64e
CHAPTER 7. VIEWING ELASTICSEARCH STATUS

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

7.1. VIEWING ELASTICSEARCH STATUS

You can view the status of your Elasticsearch cluster.

Prerequisites

- Cluster logging and Elasticsearch must be installed.

Procedure

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

   $ oc project openshift-logging

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

      $ oc get Elasticsearch

      NAME   AGE
      elasticsearch 5h9m

   b. Get the Elasticsearch status:

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

      For example:

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

      The output includes information similar to the following:

      status: 1
      clusterHealth: green 2
      conditions: 3
          .....  
      nodes: 4
          .....  
      pods: 5
          .....  
      shardAllocationEnabled: "All" 6

      1 In the output, the cluster status fields appear in the `status` stanza.
      2 The status of the Elasticsearch cluster, `green`, `red`, `yellow`.
      3 Any status conditions, if present. The Elasticsearch cluster status indicates the reasons from the scheduler if a pod could not be placed. Any events related to the
The following conditions are shown:

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

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

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

7.1.1. Example condition messages

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

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

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

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

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

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

```
status:
  nodes:
```
This status message indicates that the Elasticsearch CR uses a non-existent PVC.

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

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

7.2. VIEWING ELASTICSEARCH COMPONENT STATUS

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

Elasticsearch indices

You can view the status of the Elasticsearch indices.

1. Get the name of an Elasticsearch pod:

   $ oc get pods --selector component=elasticsearch -o name
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 ausZHoKxTNOoBvv9RlXfrw 3 1 1494624 0 2947 1475
green open .kibana 9Fltn1D0QHSnFMXpphZ--Q 1 1 1 0 0 0
green open .searchguard chOwDnQlSsqhfSPcot1Yiw 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:
estacksearch:
  Container ID: cri-o://b7d44e0a9ea486e27f47763f5bb4c39dff2
  State: Running
  Started: Mon, 08 Apr 2019 10:17:56 -0400
  Ready: True
```
Elasticsearch deployment configuration

You can view the status of the Elasticsearch deployment configuration.

1. Get the name of a deployment configuration:

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

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

2. Get the deployment configuration status:

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

   The output includes the following status information:

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

You can view the status of the Elasticsearch ReplicaSet.

1. Get the name of a replica set:

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

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

2. Get the status of the replica set:

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

   The output includes the following status information:

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

   Events: <none>
CHAPTER 8. MOVING THE CLUSTER LOGGING RESOURCES WITH NODE SELECTORS

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

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

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

Prerequisites

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

Procedure

1. Add the desired label to your nodes:

   ```
   $ oc label <resource> <name> <key>=<value>
   
   For example, to label a node:
   
   $ oc label nodes ip-10-0-142-25.ec2.internal type=elasticsearch
   ```

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

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

nodeSelector:  
  logging: curator 
  schedule: "*/10 * * * *" 

nodeSelector:  
  logging: fluentd 

---

1. Node selector for Elasticsearch.
CHAPTER 9. ADDITIONAL RESOURCES

- For more information on node selectors, see: Placing pods on specific nodes using node selectors.
CHAPTER 10. MANUALLY ROLLING OUT ELASTICSEARCH

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

10.1. PERFORMING AN ELASTICSEARCH ROLLING CLUSTER RESTART

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

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

Prerequisite

- Cluster logging and Elasticsearch must be installed.

Procedure

To perform a rolling cluster restart:

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

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

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

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

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

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

   For example:

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

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

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

   For example:

   ```bash
   -
   ```

{
  "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-k7kj    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

   $ 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-k7kj    2/2     Running   0          22h

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

   OpenShift Container Platform 4.1 Logging
   80
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",
  "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:

```
  "transient": { "cluster.routing.allocation.enable": "all" } }
```

```json
{
  "acknowledged" : true,
  "persistent" : {
  },
  "transient" : {
    "cluster" : {
      "routing" : {
        "allocation" : {
```
"enable" : "all"
}
CHAPTER 11. TROUBLESHOOTING KIBANA

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

11.1. TROUBLESHOOTING A KUBERNETES LOGIN LOOP

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

Procedure

To fix this issue:

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

   $ oc delete oauthclient/kibana-proxy

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

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

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

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

Fix this issue by replacing the OAuthClient entry.

Procedure

To replace the OAuthClient entry:

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

   $ oc delete oauthclient/kibana-proxy

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

   $ oc edit oauthclient/kibana-proxy

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

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

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

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

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

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


CHAPTER 12. EXPORTED FIELDS

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

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

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

12.1. DEFAULT EXPORTED FIELDS

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

Top Level Fields

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@timestamp</td>
<td>The UTC value marking when the log payload was created, or when the log payload was first collected if the creation time is not known. This is the log processing pipeline’s best effort determination of when the log payload was generated. Add the @ prefix convention to note a field as being reserved for a particular use. With Elasticsearch, most tools look for @timestamp by default. For example, the format would be 2015-01-24 14:06:05.071000.</td>
</tr>
<tr>
<td>geoip</td>
<td>This is geo-ip of the machine.</td>
</tr>
<tr>
<td>hostname</td>
<td>The hostname is the fully qualified domain name (FQDN) of the entity generating the original payload. This field is an attempt to derive this context. Sometimes the entity generating it knows the context. While other times that entity has a restricted namespace itself, which is known by the collector or normalizer.</td>
</tr>
<tr>
<td>ipaddr4</td>
<td>The IP address V4 of the source server, which can be an array.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>ipaddr6</td>
<td>The IP address V6 of the source server, if available.</td>
</tr>
</tbody>
</table>
| level | The logging level as provided by `rsyslog` (severitytext property), python's logging module. Possible values are as listed at `misc/sys/syslog.h` plus `trace` and `unknown`. For example, `alert` `crit` `debug` `emerg` `err` `info` `notice` `trace` `unknown` `warning`. Note that `trace` is not in the `syslog.h` list but many applications use it.  

* You should only use `unknown` when the logging system gets a value it does not understand, and note that it is the highest level.  
* Consider `trace` as higher or more verbose, than `debug`.  
* `error` is deprecated, use `err`.  
* Convert `panic` to `emerg`.  
* Convert `warn` to `warning`.  

Numeric values from `syslog/journal PRIORITY` can usually be mapped using the priority values as listed at `misc/sys/syslog.h`.  

Log levels and priorities from other logging systems should be mapped to the nearest match. See `python logging` for an example. |
| message | A typical log entry message, or payload. It can be stripped of metadata pulled out of it by the collector or normalizer, that is UTF-8 encoded. |
| pid | This is the process ID of the logging entity, if available. |
| service | The name of the service associated with the logging entity, if available. For example, the `syslog APP-NAME` and `rsyslog programname` property are mapped to the service field. |
| tags | Optionally provided operator defined list of tags placed on each log by the collector or normalizer. The payload can be a string with whitespace-delimited string tokens, or a JSON list of string tokens. |
| file | Optional path to the file containing the log entry local to the collector TODO analyzer for file paths. |
| offset | The offset value can represent bytes to the start of the log line in the file (zero or one based), or log line numbers (zero or one based), as long as the values are strictly monotonically increasing in the context of a single log file. The values are allowed to wrap, representing a new version of the log file (rotation). |
### CHAPTER 12. EXPORTED FIELDS

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

**collectd Fields**
The following fields represent namespace metrics metadata.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| collectd.interval          | type: float
The collectd interval.                                                                                                                             |
| collectd.plugin            | type: string
The collectd plug-in.                                                                                                                                    |
| collectd.plugin_instance   | type: string
The collectd plugin_instance.                                                                                                                          |
| collectd.type_instance     | type: string
The collectd type_instance.                                                                                                                             |
| collectd.type              | type: string
The collectd type.                                                                                                                                            |
| collectd.dstypes           | type: string
The collectd dstypes.                                                                                                                             |

**collectd.processes Fields**
The following field corresponds to the collectd processes plug-in.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.processes.ps_state</td>
<td>type: integer The <code>collectd ps_state</code> type of processes plug-in.</td>
</tr>
</tbody>
</table>

**collectd.processes.ps_disk_ops Fields**

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

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

**collectd.processes.ps_cputime Fields**

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| collectd.processes.ps_cputime.user | type: float  
TODO |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `collectd.processes.ps_cpu_time.syst` | type: float  
| TODO |

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `collectd.processes.ps_disk_octets.read` | type: float  
| TODO |
| `collectd.processes.ps_disk_octets.write` | type: float  
| TODO |
| `collectd.processes.fork_rate` | type: float  
| TODO |

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.

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

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

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

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

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

#### collectd.disk.disk_ops Fields
The `collectd disk_ops` type of disk plug-in.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `collectd.disk.disk_io_time.io_time` | type: float  
TODO |
| `collectd.disk.disk_io_time.weighted_io_time` | type: float  
TODO |

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

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

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

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

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

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

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

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

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

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

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

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

**collectd.virt.disk_ops** Fields
The **collectd disk_ops** type of virt plug-in.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collectd.virt.disk_ops.read</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
<tr>
<td><code>collectd.virt.disk_ops.write</code></td>
<td>type: float</td>
</tr>
<tr>
<td></td>
<td>TODO</td>
</tr>
</tbody>
</table>

**collectd.virt.disk_octets Fields**

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

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

**collectd.CPU Fields**

Corresponds to the `collectd` CPU plug-in.

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

**collectd.df Fields**

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

**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.

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

#### collectd.memory Fields

Corresponds to the `collectd` memory plug-in.

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

#### collectd.swap Fields

Corresponds to the `collectd` swap plug-in.

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

#### collectd.load Fields

Corresponds to the `collectd` load plug-in.
**collectd.load.load Fields**
The **collectd** load type of load plug-in.

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

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.statsd.host_cpu</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <strong>collectd</strong> CPU type of <strong>statsd</strong> plug-in.</td>
</tr>
<tr>
<td>collectd.statsd.host_elapsed_time</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <strong>collectd elapsed_time</strong> type of <strong>statsd</strong> plug-in.</td>
</tr>
<tr>
<td>collectd.statsd.host_memory</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <strong>collectd</strong> memory type of <strong>statsd</strong> plug-in.</td>
</tr>
<tr>
<td>collectd.statsd.host_nic_speed</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <strong>collectd nic_speed</strong> type of <strong>statsd</strong> plug-in.</td>
</tr>
<tr>
<td>collectd.statsd.host_nic_rx</td>
<td>type: integer</td>
</tr>
<tr>
<td></td>
<td>The <strong>collectd nic_rx</strong> type of <strong>statsd</strong> plug-in.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `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. |
| `collectd.statsd.host_storage` | type: integer  
The `collectd` storage type of statsd plug-in. |
| `collectd.statsd.host_swap` | type: integer  
The `collectd` swap type of statsd plug-in. |
| `collectd.statsd.host_vds` | type: integer  
The `collectd` VDSM type of statsd plug-in. |
| `collectd.statsd.host_vms` | type: integer  
The `collectd` VMS type of statsd plug-in. |
| `collectd.statsd.vm_nic_tx_dropped` | type: integer  
The `collectd nic_tx_dropped` type of statsd plug-in. |
| `collectd.statsd.vm_nic_rx_bytes` | type: integer  
The `collectd nic_rx_bytes` type of statsd plug-in. |
| `collectd.statsd.vm_nic_tx_bytes` | type: integer  
The `collectd nic_tx_bytes` type of statsd plug-in. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collectd.statsd.vm_balloo_min</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd balloon_min</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_balloo_max</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd balloon_max</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_balloon_target</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd balloon_target</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_balloon_cur</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd balloon_cur</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_cpu_sys</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd cpu_sys</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_cpu_usage</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd cpu_usage</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_read_ops</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd disk_read_ops</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_write_ops</td>
<td>type: integer</td>
</tr>
<tr>
<td>The collectd` <strong>disk_write_ops</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_flush_latency</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd disk_flush_latency</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_apparent_size</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd disk_apparent_size</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_write_bytes</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd disk_write_bytes</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>collectd.statsd.vm_disk_write_rate</td>
<td>type: integer</td>
</tr>
<tr>
<td>The <strong>collectd disk_write_rate</strong> type of <strong>statsd</strong> plug-in.</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `collectd.statsd.vm_disk_true_size` | type: integer  
The `collectd disk_true_size` type of `statsd` plug-in. |
| `collectd.statsd.vm_disk_read_rate` | type: integer  
The `collectd disk_read_rate` type of `statsd` plug-in. |
| `collectd.statsd.vm_disk_write_latency` | type: integer  
The `collectd disk_write_latency` type of `statsd` plug-in. |
| `collectd.statsd.vm_disk_read_latency` | type: integer  
The `collectd disk_read_latency` type of `statsd` plug-in. |
| `collectd.statsd.vm_disk_read_bytes` | type: integer  
The `collectd disk_read_bytes` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_rx_dropped` | type: integer  
The `collectd nic_rx_dropped` type of `statsd` plug-in. |
| `collectd.statsd.vm_cpu_user` | type: integer  
The `collectd cpu_user` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_rx_errors` | type: integer  
The `collectd nic_rx_errors` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_tx_errors` | type: integer  
The `collectd nic_tx_errors` type of `statsd` plug-in. |
| `collectd.statsd.vm_nic_speed` | type: integer  
The `collectd nic_speed` type of `statsd` plug-in. |

**collectd.postgresql Fields**  
Corresponds to `collectd postgreSQL` plug-in.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `collectd.postgresql.pg_n_tup_g` | type: integer  
The `collectd` type `pg_n_tup_g` of plug-in `postgresql`. |
### Parameter | Description
--- | ---
`collectd.postgresql.pg_n_tup_c` | type: integer. The `collectd` type `pg_n_tup_c` of plug-in postgresql.

`collectd.postgresql.pg_numbackends` | type: integer. The `collectd` type `pg_numbackends` of plug-in postgresql.

`collectd.postgresql.pg_xact` | type: integer. The `collectd` type `pg_xact` of plug-in postgresql.

`collectd.postgresql.pg_db_size` | type: integer. The `collectd` type `pg_db_size` of plug-in postgresql.

`collectd.postgresql.pg_blks` | type: integer. The `collectd` type `pg_blks` of plug-in postgresql.

### 12.2. RSYSLOG EXPORTED FIELDS

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

The following fields are RFC5424 based metadata.

### Parameter | Description
--- | ---
`rsyslog.facility` | See syslog specification for more information on `rsyslog`.

`rsyslog.protocol-version` | This is the `rsyslog` protocol version.

`rsyslog.structured-data` | See syslog specification for more information on `syslog` structured-data.

`rsyslog.msgid` | This is the `syslog` msgid field.

`rsyslog.appname` | If `app-name` is the same as `programname`, then only fill top-level field `service`. If `app-name` is not equal to `programname`, this field will hold `app-name`. See syslog specifications for more information.

### 12.3. SYSTEMD EXPORTED FIELDS

These are the `systemd` fields exported by the OpenShift Container Platform cluster logging available for searching from Elasticsearch and Kibana.
Contains common fields specific to \texttt{systemd} journal. \textit{Applications} may write their own fields to the journal. These will be available under the \texttt{systemd.u} namespace. \texttt{RESULT} and \texttt{UNIT} are two such fields.

\textbf{systemd.k Fields}

The following table contains \texttt{systemd} kernel-specific metadata.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{systemd.k.KERNEL DEVICE}</td>
<td>\texttt{systemd.k.KERNEL_DEVICE} is the kernel device name.</td>
</tr>
<tr>
<td>\texttt{systemd.k.KERNEL SUBSYSTEM}</td>
<td>\texttt{systemd.k.KERNEL_SUBSYSTEM} is the kernel subsystem name.</td>
</tr>
<tr>
<td>\texttt{systemd.k.UDEV_DEVLINK}</td>
<td>\texttt{systemd.k.UDEV_DEVLINK} includes additional symlink names that point to the node.</td>
</tr>
<tr>
<td>\texttt{systemd.k.UDEV_DEVNODE}</td>
<td>\texttt{systemd.k.UDEV_DEVNODE} is the node path of the device.</td>
</tr>
<tr>
<td>\texttt{systemd.k.UDEV_SYSNAME}</td>
<td>\texttt{systemd.k.UDEV_SYSNAME} is the kernel device name.</td>
</tr>
</tbody>
</table>

\textbf{systemd.t Fields}

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

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

**systemd.u Fields**  
**systemd.u Fields** are directly passed from clients and stored in the journal.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>systemd.u.CODE_FILE</td>
<td><code>systemd.u.CODE_FILE</code> is the code location containing the filename of the source.</td>
</tr>
<tr>
<td>systemd.u.CODE_FUNCTION</td>
<td><code>systemd.u.CODE_FUNCTION</code> is the code location containing the function of the source.</td>
</tr>
<tr>
<td>systemd.u.CODE_LINE</td>
<td><code>systemd.u.CODE_LINE</code> is the code location containing the line number of the source.</td>
</tr>
<tr>
<td>systemd.u.ERRNO</td>
<td><code>systemd.u.ERRNO</code>, if present, is the low-level error number formatted in numeric value, as a decimal string.</td>
</tr>
<tr>
<td>systemd.u.MESSAGE_ID</td>
<td><code>systemd.u.MESSAGE_ID</code> is the message identifier ID for recognizing message types.</td>
</tr>
<tr>
<td>systemd.u.RESULT</td>
<td>For private use only.</td>
</tr>
<tr>
<td>systemd.u.UNIT</td>
<td>For private use only.</td>
</tr>
</tbody>
</table>

### 12.4. KUBERNETES EXPORTED FIELDS

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kubernetes.pod_id</td>
<td>Kubernetes ID of the pod.</td>
</tr>
<tr>
<td>kubernetes.namespace_name</td>
<td>The name of the namespace in Kubernetes.</td>
</tr>
<tr>
<td>kubernetes.namespace_id</td>
<td>ID of the namespace in Kubernetes.</td>
</tr>
<tr>
<td>kubernetes.host</td>
<td>Kubernetes node name.</td>
</tr>
<tr>
<td>kubernetes.container_name</td>
<td>The name of the container in Kubernetes.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>kubernetes.labels.deployment</code></td>
<td>The deployment associated with the Kubernetes object.</td>
</tr>
<tr>
<td><code>kubernetes.labels.deploymentconfig</code></td>
<td>The deploymentconfig associated with the Kubernetes object.</td>
</tr>
<tr>
<td><code>kubernetes.labels.component</code></td>
<td>The component associated with the Kubernetes object.</td>
</tr>
<tr>
<td><code>kubernetes.labels.provider</code></td>
<td>The provider associated with the Kubernetes object.</td>
</tr>
</tbody>
</table>

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

### 12.5. CONTAINER EXPORTED FIELDS

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

**pipeline_metadata.collector Fields**
This section contains metadata specific to the collector.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pipeline_metadata.collector.hostname</code></td>
<td>FQDN of the collector. It might be different from the FQDN of the actual emitter of the logs.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.name</code></td>
<td>Name of the collector.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.version</code></td>
<td>Version of the collector.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.ipaddr4</code></td>
<td>IP address v4 of the collector server, can be an array.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.ipaddr6</code></td>
<td>IP address v6 of the collector server, can be an array.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.inputname</code></td>
<td>How the log message was received by the collector whether it was TCP/UDP, or imjournal/imfile.</td>
</tr>
<tr>
<td><code>pipeline_metadata.collector.received_at</code></td>
<td>Time when the message was received by the collector.</td>
</tr>
</tbody>
</table>
pipeline_metadata.collect or.original_raw_message
The original non-parsed log message, collected by the collector or as close to the source as possible.

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

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

pipeline_metadata.trace
The field records the trace of the message. Each collector and normalizer appends information about itself and the date and time when the message was processed.

12.6. OVAIRT 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.
ovirt.entity | The type of the data source, hosts, VMS, and engine.
---|---
ovirt.host_id | The oVirt host UUID.

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

12.7. AUSHAPE EXPORTED FIELDS

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
aushape.data.avc        | type: nested |
aushape.data.execve     | type: string |
aushape.data.netfilter_cnf | type: nested |
aushape.data.obj_pid    | type: nested |
aushape.data.path       | type: nested |
12.8. TLOG EXPORTED FIELDS

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tlog.ver</td>
<td>Message format version number.</td>
</tr>
<tr>
<td>tlog.user</td>
<td>Recorded user name.</td>
</tr>
<tr>
<td>tlog.term</td>
<td>Terminal type name.</td>
</tr>
<tr>
<td>tlog.session</td>
<td>Audit session ID of the recorded session.</td>
</tr>
<tr>
<td>tlog.id</td>
<td>ID of the message within the session.</td>
</tr>
<tr>
<td>tlog.pos</td>
<td>Message position in the session, milliseconds.</td>
</tr>
<tr>
<td>tlog.timing</td>
<td>Distribution of this message’s events in time.</td>
</tr>
<tr>
<td>tlog.in_txt</td>
<td>Input text with invalid characters scrubbed.</td>
</tr>
<tr>
<td>tlog.in_bin</td>
<td>Scrubbed invalid input characters as bytes.</td>
</tr>
<tr>
<td>tlog.out_txt</td>
<td>Output text with invalid characters scrubbed.</td>
</tr>
<tr>
<td>tlog.out_bin</td>
<td>Scrubbed invalid output characters as bytes.</td>
</tr>
</tbody>
</table>
CHAPTER 13. UNINSTALLING CLUSTER LOGGING

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

13.1. UNINSTALLING CLUSTER LOGGING FROM OPENSШIFT 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