OpenShift Container Platform 4.8

Logging

OpenShift Logging installation, usage, and release notes

Last Updated: 2021-09-09

OpenShift Container Platform 4.8 Logging

OpenShift Logging installation, usage, and release notes

Abstract

This document provides instructions for installing, configuring, and using OpenShift Logging, which aggregates logs for a range of OpenShift Container Platform services.

Table of Contents

CHAPTER 1. RELEASE NOTES FOR RED HAT OPENSHIFT LOGGING 5.2

1.1. SUPPORTED VERSIONS

- OpenShift Logging versions 5.0, 5.1, and 5.2 run on OpenShift Container Platform versions 4.7 and 4.8.

1.2. MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see Red Hat CTO Chris Wright's message .

1.3. OPENSHIFT LOGGING 5.2.0

This release includes RHBA-2021:3393 OpenShift Logging Bug Fix Release 5.2.0 .

1.3.1. New features and enhancements

- With this update, you can forward log data to Amazon CloudWatch, which provides application and infrastructure monitoring. For more information, see Forwarding logs to Amazon CloudWatch. (LOG-1173)

- With this update, you can forward log data to Grafana Loki, a horizontally scalable, highly available, multi-tenant log aggregation system. For more information, see Forwarding logs to Grafana Loki. (LOG-684)

- With this update, if you use the Fluentd forward protocol to forward log data over a TLS-encrypted connection, you can now use a password-encrypted private key file and specify the passphrase in the Cluster Log Forwarder configuration. For more information, see Forwarding logs using the Fluentd forward protocol. (LOG-1525)

- This enhancement enables you to use a username and password to authenticate a log forwarding connection to an external Elasticsearch instance. For example, if you cannot use mutual TLS (mTLS) because a third-party operates the Elasticsearch instance, you can use HTTP or HTTPS and set a secret that contains the username and password. For more information see Forwarding logs to an external Elasticsearch instance . (LOG-1022)

- With this update, you can collect OVN network policy audit logs for forwarding to a logging server. For more information, see Collecting OVN network policy audit logs . (LOG-1526)

- By default, the data model introduced in OpenShift Container Platform 4.5 gave logs from different namespaces a single index in common. This change made it harder to see which namespaces produced the most logs.
  The current release, OpenShift Logging 5.2, adds namespace metrics to the **Logging** dashboard in the OpenShift Container Platform console. With these metrics, you can see which namespaces produce logs and how many logs each namespace produces for a given timestamp.

  To see these metrics, open the **Administrator** perspective in the OpenShift Container Platform web console, and navigate to **Monitoring** → **Dashboards** → **Logging/Elasticsearch**. (LOG-1680)

- The current release, OpenShift Logging 5.2, enables two new metrics: For a given timestamp or duration, you can see the total logs produced or logged by individual containers, and the total logs collected by the collector. These metrics are labeled by namespace, pod, and container name, so you can see how many logs each namespace and pod collects and produces. (LOG-1213)

1.3.2. Bug fixes

- LOG-1130 "BZ#1927249 - fieldmanager.go:186 - SHOULD NOT HAPPEN - failed to update managedFields...duplicate entries for key 'name="POLICY_MAPPING"'"
- LOG-1268 "elasticsearch-im-{app,infra,audit} successfully run but fail to run their tasks."
- LOG-1271 "Logs Produced Line Chart does not display podname and namespace labels"
- LOG-1273 "The index management job status is always 'Completed' even when there has an error in the job log."
- LOG-1385 "APIRemovedInNextReleaseInUse alert for priorityclasses"
- LOG-1420 "Operators missing disconnected annotation"
- LOG-1440 "BZ#1966561 - OLM bug workaround for workload partitioning (PR#1042)"
- LOG-1499 "release-5.2 - Error 'com.fasterxml.jackson.core.JsonParseException: Invalid UTF-8 start byte 0x92' in Elasticsearch/Fluentd logs"
- LOG-1567 "Use correct variable for nextIndex"
- LOG-1446 "kibana-proxy CrashLoopBackoff with error Invalid configuration cookie_secret must be 16, 24, or 32 bytes to create an AES cipher"
- LOG-1625 "ds/fluentd is not created due to: 'system:serviceaccount:openshift-logging:cluster-logging-operator' cannot create resource 'securitycontextconstraints' in API group 'security.openshift.io' at the cluster scope"
- LOG-1071 "fluentd configuration posting all messages to its own log"
- LOG-1276 "Update Elasticsearch/kibana to use opendistro security plugin 2.10.5.1"
- LOG-1353 "No datapoints found on top 10 containers dashboard"
- LOG-1411 "Underestimate queued_chunks_limit_size value with chunkLimitSize and totalLimitSize tuning parameters"
- LOG-1558 "Update OD security dependency to resolve kibana index migration issue"
- LOG-1562 "CVE-2021-32740 logging-fluentd-container: rubygem-addressable: ReDoS in templates - openshift-logging-5"
- LOG-1570 "Bug 1981579: Fix built-in application behavior to collect all of logs"
- LOG-1589 "There are lots of dockercfg secrets and service-account-token secrets for ES and Kibana in openshift-logging namespace after deploying EFK pods."
- LOG-1590 "Vendored viaq/logerr dependency is missing a license file"

- LOG-1623 "Metric`log_collected_bytes_total` is not exposed"
- LOG-1624 "Index management cronjobs are using wrong image in CSV/elasticsearch-operator.5.2.0-1"
- LOG-1634 "Logging 5.2 - The CSV version is not changed in new bundles."
- LOG-1647 "Fluentd pods raise error **Prometheus::Client::LabelSetValidator::InvalidLabelSetError** when forward logs to external logStore"
- LOG-1657 "Index management jobs failing with error while attemping to determine the active write alias no permissions"
- LOG-1681 "Fluentd pod metric not able to scrape , Fluentd target down"
- LOG-1683 "Loki output not present in CLO CRD"
- LOG-1702 "Entry out of order when forward logs to loki"
- LOG-1714 "Memory/CPU spike issues seen with Logging 5.2 on Power"
- LOG-1722 "The value of card **Total Namespace Count** in Logging/Elasticsearch dashboard is not correct."
- LOG-1723 "In fluentd config, flush_interval can't be set with flush_mode=immediate"

1.3.3. Known issues

- If you forward logs to an external Elasticsearch server and then change a configured value in the pipeline secret, such as the username and password, the fluentd forwarder loads the new secret but uses the old value to connect to an external Elasticsearch server. This issue happens because the Red Hat OpenShift Logging Operator does not currently monitor secrets for content changes. (LOG-1652)
  As a workaround, if you change the secret, you can force the Fluentd pods to redeploy by entering:

  ```
  $ oc delete pod -l component=fluentd
  ```

1.4. OPENSHIFT LOGGING 5.1.0

This release includes:

- RHBA-2021:2885 - Bug Fix Advisory. Openshift Logging Bug Fix Release 5.1.1
- RHBA-2021:2112 - Bug Fix Advisory. OpenShift Logging Bug Fix Release 5.1.0

1.4.1. New features and enhancements

OpenShift Logging 5.1 now supports OpenShift Container Platform 4.7 and later running on:

- IBM Power Systems
- IBM Z and LinuxONE

This release adds improvements related to the following components and concepts.

- As a cluster administrator, you can use Kubernetes pod labels to gather log data from an application and send it to a specific log store. You can gather log data by configuring the **inputs[].application.selector.matchLabels** element in the **ClusterLogForwarder** custom resource (CR) YAML file. You can also filter the gathered log data by namespace. (LOG-883)
- This release adds the following new **ElasticsearchNodeDiskWatermarkReached** warnings to the OpenShift Elasticsearch Operator (EO):
    - Elasticsearch Node Disk Low Watermark Reached
    - Elasticsearch Node Disk High Watermark Reached
    - Elasticsearch Node Disk Flood Watermark Reached

  The alert applies the past several warnings when it predicts that an Elasticsearch node will reach the **Disk Low Watermark**, **Disk High Watermark**, or **Disk Flood Stage Watermark** thresholds in the next 6 hours. This warning period gives you time to respond before the node reaches the disk watermark thresholds. The warning messages also provide links to the troubleshooting steps, which you can follow to help mitigate the issue. The EO applies the past several hours of disk space data to a linear model to generate these warnings. (LOG-1100)
- JSON logs can now be forwarded as JSON objects, rather than quoted strings, to either Red Hat's managed Elasticsearch cluster or any of the other supported third-party systems. Additionally, you can now query individual fields from a JSON log message inside Kibana which increases the discoverability of specific logs. (LOG-785, LOG-1148)

1.4.2. Deprecated and removed features

Some features available in previous releases have been deprecated or removed.

Deprecated functionality is still included in OpenShift Logging and continues to be supported; however, it will be removed in a future release of this product and is not recommended for new deployments.

1.4.2.1. Elasticsearch Curator has been removed

With this update, the Elasticsearch Curator has been removed and is no longer supported. Elasticsearch Curator helped you curate or manage your indices on OpenShift Container Platform 4.4 and earlier. Instead of using Elasticsearch Curator, configure the log retention time.

1.4.3. Bug fixes

- Before this update, the **ClusterLogForwarder** CR did not show the **input[].selector** element after it had been created. With this update, when you specify a **selector** in the **ClusterLogForwarder** CR, it remains. Fixing this bug was necessary for LOG-883, which enables using pod label selectors to forward application log data. (LOG-1338)
- Before this update, an update in the cluster service version (CSV) accidentally introduced resources and limits for the OpenShift Elasticsearch Operator container. Under specific conditions, this caused an out-of-memory condition that terminated the Elasticsearch Operator pod. This update fixes the issue by removing the CSV resources and limits for the Operator container. The Operator gets scheduled without issues. (LOG-1254)
- Before this update, forwarding logs to Kafka using chained certificates failed with the following error message:

state=error: certificate verify failed (unable to get local issuer certificate)

Logs could not be forwarded to a Kafka broker with a certificate signed by an intermediate CA. This happened because the Fluentd Kafka plug-in could only handle a single CA certificate supplied in the **ca-bundle.crt** entry of the corresponding secret. This update fixes the issue by enabling the Fluentd Kafka plug-in to handle multiple CA certificates supplied in the **ca-bundle.crt** entry of the corresponding secret. Now, logs can be forwarded to a Kafka broker with a certificate signed by an intermediate CA. (LOG-1218, LOG-1216)

- Before this update, while under load, Elasticsearch responded to some requests with an HTTP 500 error, even though there was nothing wrong with the cluster. Retrying the request was successful. This update fixes the issue by updating the index management cron jobs to be more resilient when they encounter temporary HTTP 500 errors. The updated index management cron jobs will first retry a request multiple times before failing. (LOG-1215)

- Before this update, if you did not set the **.proxy** value in the cluster installation configuration, and then configured a global proxy on the installed cluster, a bug prevented Fluentd from forwarding logs to Elasticsearch. To work around this issue, in the proxy or cluster configuration, set the **no_proxy** value to **.svc.cluster.local** so it skips internal traffic. This update fixes the proxy configuration issue. If you configure the global proxy after installing an OpenShift Container Platform cluster, Fluentd forwards logs to Elasticsearch. (LOG-1187, BZ#1915448)

- Before this update, the logging collector created more socket connections than necessary. With this update, the logging collector reuses the existing socket connection to send logs. (LOG-1186)

- Before this update, if a cluster administrator tried to add or remove storage from an Elasticsearch cluster, the OpenShift Elasticsearch Operator (EO) incorrectly tried to upgrade the Elasticsearch cluster, displaying **scheduledUpgrade: "True"**, **shardAllocationEnabled: primaries**, and change the volumes. With this update, the EO does not try to upgrade the Elasticsearch cluster.
  The EO status displays the following new status information to indicate when you have tried to make an unsupported change to the Elasticsearch storage that it has ignored:

  - **StorageStructureChangeIgnored** when you try to change between using ephemeral and persistent storage structures.
  - **StorageClassNameChangeIgnored** when you try to change the storage class name.
  - **StorageSizeChangeIgnored** when you try to change the storage size.

  NOTE

  If you configure the **ClusterLogging** custom resource (CR) to switch from ephemeral to persistent storage, the EO creates a persistent volume claim (PVC) but does not create a persistent volume (PV). To clear the **StorageStructureChangeIgnored** status, you must revert the change to the **ClusterLogging** CR and delete the persistent volume claim (PVC).

  (LOG-1351)

- Before this update, if you redeployed a full Elasticsearch cluster, it got stuck in an unhealthy state, with one non-data node running and all other data nodes shut down. This issue happened because new certificates prevented the Elasticsearch Operator from scaling down the non-data nodes of the Elasticsearch cluster. With this update, Elasticsearch Operator can scale all the data and non-data nodes down and then back up again, so they load the new certificates. The Elasticsearch Operator can reach the new nodes after they load the new certificates. (LOG-1536)

CHAPTER 2. UNDERSTANDING RED HAT OPENSHIFT LOGGING

As a cluster administrator, you can deploy OpenShift Logging to aggregate all the logs from your OpenShift Container Platform cluster, such as node system audit logs, application container logs, and infrastructure logs. OpenShift Logging aggregates these logs from throughout your cluster and stores them in a default log store. You can use the Kibana web console to visualize log data .

OpenShift Logging aggregates the following types of logs:

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

NOTE

Because the internal OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs, audit logs are not stored in the internal Elasticsearch instance by default. If you want to send the audit logs to the default internal Elasticsearch log store, for example to view the audit logs in Kibana, you must use the Log Forwarding API as described in Forward audit logs to the log store .

2.1. ABOUT DEPLOYING OPENSHIFT LOGGING

OpenShift Container Platform cluster administrators can deploy OpenShift Logging using the OpenShift Container Platform web console or CLI to install the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator. When the operators are installed, you create a **ClusterLogging** custom resource (CR) to schedule OpenShift Logging pods and other resources necessary to support OpenShift Logging. The operators are responsible for deploying, upgrading, and maintaining OpenShift Logging.

The **ClusterLogging** CR defines a complete OpenShift Logging environment that includes all the components of the logging stack to collect, store and visualize logs. The Red Hat OpenShift Logging Operator watches the OpenShift Logging CR and adjusts the logging deployment accordingly.

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

For information, see Configuring the log collector.

2.1.1. About OpenShift Logging components

The OpenShift Logging components include a collector deployed to each node in the OpenShift Container Platform cluster that collects all node and container logs and writes them to a log store. You can use a centralized web UI to create rich visualizations and dashboards with the aggregated data.

The major components of OpenShift Logging are:

- collection - This is the component that collects logs from the cluster, formats them, and forwards them to the log store. The current implementation is Fluentd.
- log store - This is where the logs are stored. The default implementation is Elasticsearch. You can use the default Elasticsearch log store or forward logs to external log stores. The default log store is optimized and tested for short-term storage.
- visualization - This is the UI component you can use to view logs, graphs, charts, and so forth. The current implementation is Kibana.

This document might refer to log store or Elasticsearch, visualization or Kibana, collection or Fluentd, interchangeably, except where noted.

2.1.2. About the logging collector

OpenShift Container Platform uses Fluentd to collect container and node logs.

By default, the log collector uses the following sources:

- journald for all system logs
- **/var/log/containers/*.log** for all container logs

If you configure the log collector to collect audit logs, it gets them from **/var/log/audit/audit.log**.

The logging collector is a daemon set that deploys pods to each OpenShift Container Platform node. System and infrastructure logs are generated by journald log messages from the operating system, the container runtime, and OpenShift Container Platform. Application logs are generated by the CRI-O container engine. Fluentd collects the logs from these sources and forwards them internally or externally as you configure in OpenShift Container Platform.

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

IMPORTANT

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

For information, see Configuring the log collector.

2.1.3. About the log store

By default, OpenShift Container Platform uses Elasticsearch (ES) to store log data. Optionally, you can use the log forwarding features to forward logs to external log stores using Fluentd protocols, syslog protocols, or the OpenShift Container Platform Log Forwarding API.

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

Elasticsearch organizes the log data from Fluentd into datastores, or *indices*, then subdivides each index into multiple pieces called *shards*, which it spreads across a set of Elasticsearch nodes in an Elasticsearch cluster. You can configure Elasticsearch to make copies of the shards, called *replicas*, which Elasticsearch also spreads across the Elasticsearch nodes. The **ClusterLogging** custom resource (CR) allows you to specify how the shards are replicated to provide data redundancy and resilience to failure. You can also specify how long the different types of logs are retained using a retention policy in the **ClusterLogging** CR.

NOTE

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

The Red Hat OpenShift Logging Operator and companion OpenShift Elasticsearch Operator ensure that each Elasticsearch node is deployed using a unique deployment that includes its own storage volume. You can use a **ClusterLogging** custom resource (CR) to increase the number of Elasticsearch nodes, as needed. See the Elasticsearch documentation for considerations involved in configuring storage.

NOTE

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

Role-based access control (RBAC) applied on the Elasticsearch indices enables the controlled access of the logs to the developers. Administrators can access all logs and developers can access only the logs in their projects.

For information, see Configuring the log store.

2.1.4. About logging visualization

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

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

For information, see Configuring the log visualizer.

2.1.5. About event routing

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

You must manually deploy the Event Router.

For information, see Collecting and storing Kubernetes events.

2.1.6. About log forwarding

By default, OpenShift Logging sends logs to the default internal Elasticsearch log store, defined in the **ClusterLogging** custom resource (CR). If you want to forward logs to other log aggregators, you can use the log forwarding features to send logs to specific endpoints within or outside your cluster.

For information, see Forwarding logs to third-party systems .

CHAPTER 3. INSTALLING OPENSHIFT LOGGING

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

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

- Reviewing the OpenShift Logging storage considerations .
- Installing the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator using the OpenShift Container Platform web console or CLI.

3.1. INSTALLING OPENSHIFT LOGGING USING THE WEB CONSOLE

You can use the OpenShift Container Platform web console to install the OpenShift Elasticsearch and Red Hat OpenShift Logging Operators.

NOTE

If you do not want to use the default Elasticsearch log store, you can remove the internal Elasticsearch **logStore** and Kibana **visualization** components from the **ClusterLogging** custom resource (CR). Removing these components is optional but saves resources. For more information, see Removing unused components if you do not use the default Elasticsearch log store.

Prerequisites

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

  NOTE

  If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the **LocalVolume** object. Elasticsearch cannot use raw block volumes.

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

Procedure

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

1. Install the OpenShift Elasticsearch Operator:
   a. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
   b. Choose **OpenShift Elasticsearch Operator** from the list of available Operators, and click **Install**.

c. Ensure that the **All namespaces on the cluster**is selected under **Installation Mode**.

d. Ensure that **openshift-operators-redhat** is selected under **Installed Namespace**.
You must specify the **openshift-operators-redhat** namespace. The **openshift-operators** namespace might contain Community Operators, which are untrusted and could publish a metric with the same name as an OpenShift Container Platform metric, which would cause conflicts.

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

f. Select **stable-5.x** as the **Update Channel**.

g. Select an **Approval Strategy**.

- The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.

- The **Manual** strategy requires a user with appropriate credentials to approve the Operator update.

h. Click **Install**.

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

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

2. Install the Red Hat OpenShift Logging Operator:

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

b. Choose **Red Hat OpenShift Logging** from the list of available Operators, and click **Install**.

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

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

e. Select **Enable operator recommended cluster monitoring on this namespace**
This option sets the **openshift.io/cluster-monitoring: "true"** label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-logging** namespace.

f. Select **stable-5.x** as the **Update Channel**.

g. Select an **Approval Strategy**.

- The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.

- The **Manual** strategy requires a user with appropriate credentials to approve the Operator update.

h. Click **Install**.

i. Verify that the Red Hat OpenShift Logging Operator installed by switching to the **Operators** → **Installed Operators** page.

j. Ensure that **Red Hat OpenShift Logging** is listed in the **openshift-logging** project with a **Status** of **Succeeded**.
If the Operator does not appear as installed, to troubleshoot further:

- Switch to the **Operators** → **Installed Operators** page and inspect the **Status** column for any errors or failures.
- Switch to the **Workloads** → **Pods** page and check the logs in any pods in the **openshift-logging** project that are reporting issues.

3. Create a OpenShift Logging instance:

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

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

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

d. On the **ClusterLoggings** page, click **Create ClusterLogging**.
You might have to refresh the page to load the data.

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

NOTE

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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging"
spec:
  managementState: "Managed" 2
  logStore:
    type: "elasticsearch" 3
    retentionPolicy: 4
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3 5
      storage:
        storageClassName: "<storage_class_name>" 6
        size: 200G
      resources: 7
        requests:
          memory: "8Gi"
      proxy: 8
        resources:
          limits:
            memory: 256Mi
          requests:
            memory: 256Mi
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" 9
    kibana:
      replicas: 1
  collection:
    logs:
      type: "fluentd" 10
      fluentd: {}
```

1 The name must be **instance**.

2 The OpenShift Logging management state. In some cases, if you change the OpenShift Logging defaults, you must set this to **Unmanaged**. However, an unmanaged deployment does not receive updates until OpenShift Logging is placed back into a managed state.

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

4 Specify the length of time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, **7d** for seven days. Logs older than the **maxAge** are deleted. You must specify a retention policy for each log source or the Elasticsearch indices will not be created for that source.

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

6 Enter the name of an existing storage class for Elasticsearch storage. For best performance, specify a storage class that allocates block storage. If you do not specify a storage class, OpenShift Logging uses ephemeral storage.

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

8 Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are **256Mi** for the memory request and **100m** for the CPU request.

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

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

memory limits. For more information, see **Configuring Fluentd**.

NOTE

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

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

```
$ oc get deployment
```

Example output

```
cluster-logging-operator       1/1     1            1           18h
elasticsearch-cd-x6kdekli-1     0/1     1            0           6m54s
elasticsearch-cdm-x6kdekli-1   1/1     1            1           18h
elasticsearch-cdm-x6kdekli-2   0/1     1            0           6m49s
elasticsearch-cdm-x6kdekli-3   0/1     1            0           6m44s
```

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

f. Click **Create**. This creates the OpenShift Logging components, the **Elasticsearch** custom resource and components, and the Kibana interface.

4. Verify the install:

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

b. Select the **openshift-logging** project.
You should see several pods for OpenShift Logging, Elasticsearch, Fluentd, and Kibana similar to the following list:

- cluster-logging-operator-cb795f8dc-xkckc
- elasticsearch-cdm-b3nqzchd-1-5c6797-67kfz
- elasticsearch-cdm-b3nqzchd-2-6657f4-wtprv
- elasticsearch-cdm-b3nqzchd-3-588c65-clg7g
- fluentd-2c7dg
- fluentd-9z7kk
- fluentd-br7r2

- fluentd-fn2sb
- fluentd-pb2f8
- fluentd-zqgqx
- kibana-7fb4fd4cc9-bvt4p

Additional resources

- Installing Operators from the OperatorHub

3.2. POST-INSTALLATION TASKS

If you plan to use Kibana, you must manually create your Kibana index patterns and visualizations to explore and visualize data in Kibana.

If your cluster network provider enforces network isolation, allow network traffic between the projects that contain the OpenShift Logging operators.

3.3. INSTALLING OPENSHIFT LOGGING USING THE CLI

You can use the OpenShift Container Platform CLI to install the OpenShift Elasticsearch and Red Hat OpenShift Logging Operators.

Prerequisites

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

  NOTE

  If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the **LocalVolume** object. Elasticsearch cannot use raw block volumes.

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

Procedure

To install the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator using the CLI:

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

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

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

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

b. Create the namespace:

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

For example:

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

2. Create a namespace for the Red Hat OpenShift Logging Operator:

a. Create a namespace object YAML file (for example, **olo-namespace.yaml**) for the Red Hat OpenShift Logging Operator:

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

b. Create the namespace:

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

For example:

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

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

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

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
```

```
metadata:
  name: openshift-operators-redhat
  namespace: openshift-operators-redhat (1)
spec: {}
```

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

b. Create an Operator Group object:

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

For example:

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

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

Example Subscription

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: "elasticsearch-operator"
  namespace: "openshift-operators-redhat" (1)
spec:
  channel: "stable-5.1" (2)
  installPlanApproval: "Automatic"
  source: "redhat-operators" (3)
  sourceNamespace: "openshift-marketplace"
  name: "elasticsearch-operator"
```

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

(2) Specify **5.0**, **stable**, or **stable-5.<x>** as the channel. See the following note.

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

NOTE

Specifying **stable** installs the current version of the latest stable release. Using **stable** with **installPlanApproval: "Automatic"**, will automatically upgrade your operators to the latest stable major and minor release.

Specifying **stable-5.<x>** installs the current minor version of a specific major release. Using **stable-5.<x>** with **installPlanApproval: "Automatic"**, will automatically upgrade your operators to the latest stable minor release within the major release you specify with **x**.

d. Create the Subscription object:

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

For example:

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

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

e. Verify the Operator installation:

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

Example output

```
NAMESPACE                                          NAME                                    DISPLAY
VERSION             REPLACES   PHASE
default                                        elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0            Succeeded
kube-node-lease                                elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0            Succeeded
kube-public                                    elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0            Succeeded
kube-system                                    elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0            Succeeded
openshift-apiserver-operator                   elasticsearch-operator.5.1.0-
202007012112.p0    OpenShift Elasticsearch Operator   5.1.0-202007012112.p0
Succeeded
openshift-apiserver                            elasticsearch-operator.5.1.0-202007012112.p0
OpenShift Elasticsearch Operator   5.1.0-202007012112.p0            Succeeded
openshift-authentication-operator              elasticsearch-operator.5.1.0-
202007012112.p0    OpenShift Elasticsearch Operator   5.1.0-202007012112.p0
Succeeded
openshift-authentication                       elasticsearch-operator.5.1.0-
202007012112.p0    OpenShift Elasticsearch Operator   5.1.0-202007012112.p0
Succeeded
...
```

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

4. Install the Red Hat OpenShift Logging Operator by creating the following objects:

a. Create an Operator Group object YAML file (for example, **olo-og.yaml**) for the Red Hat OpenShift Logging Operator:

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

```
spec:
  targetNamespaces:
  - openshift-logging (2)
```

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

b. Create the OperatorGroup object:

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

For example:

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

c. Create a Subscription object YAML file (for example, **olo-sub.yaml**) to subscribe a namespace to the Red Hat OpenShift Logging Operator.

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

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

(2) Specify **5.0**, **stable**, or **stable-5.<x>** as the channel.

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

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

For example:

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

The Red Hat OpenShift Logging Operator is installed to the **openshift-logging** namespace.

d. Verify the Operator installation.
There should be a Red Hat OpenShift Logging Operator in the **openshift-logging** namespace. The Version number might be different than shown.

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

Example output

```
NAMESPACE                                   NAME                              DISPLAY
VERSION           REPLACES   PHASE
...
openshift-logging                          clusterlogging.5.1.0-202007012112.p0
OpenShift Logging        5.1.0-202007012112.p0            Succeeded
...
```

5. Create a OpenShift Logging instance:

   a. Create an instance object YAML file (for example, **olo-instance.yaml**) for the Red Hat OpenShift Logging Operator:

NOTE

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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging"
spec:
  managementState: "Managed" 2
  logStore:
    type: "elasticsearch" 3
    retentionPolicy: 4
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3 5
      storage:
        storageClassName: "<storage-class-name>" 6
        size: 200G
      resources: 7
        requests:
          memory: "8Gi"
      proxy: 8
        resources:
          limits:
            memory: 256Mi
          requests:
             memory: 256Mi
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" (9)
    kibana:
      replicas: 1
  collection:
    logs:
      type: "fluentd" (10)
      fluentd: {}
```

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

2. The OpenShift Logging management state. In some cases, if you change the OpenShift Logging defaults, you must set this to **Unmanaged**. However, an unmanaged deployment does not receive updates until OpenShift Logging is placed back into a managed state. Placing a deployment back into a managed state might revert any modifications you made.

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

4. Specify the length of time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, **7d** for seven days. Logs older than the **maxAge** are deleted. You must specify a retention policy for each log source or the Elasticsearch indices will not be created for that source.

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

6. Enter the name of an existing storage class for Elasticsearch storage. For best performance, specify a storage class that allocates block storage. If you do not specify a storage class, OpenShift Container Platform deploys OpenShift Logging with ephemeral storage only.

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

8. Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are **256Mi** for the memory request and **100m** for the CPU request.

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

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

NOTE

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

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

```
$ oc get deployment
```

Example output

```
cluster-logging-operator       1/1     1            1           18h
elasticsearch-cd-x6kdekli-1    1/1     1            0           6m54s
elasticsearch-cdm-x6kdekli-1   1/1     1            1           18h
elasticsearch-cdm-x6kdekli-2   1/1     1            0           6m49s
elasticsearch-cdm-x6kdekli-3   1/1     1            0           6m44s
```

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

b. Create the instance:

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

For example:

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

This creates the OpenShift Logging components, the **Elasticsearch** custom resource and components, and the Kibana interface.

6. Verify the installation by listing the pods in the **openshift-logging** project.
You should see several pods for OpenShift Logging, Elasticsearch, Fluentd, and Kibana similar to the following list:

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

Example output

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

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

3.4. POST-INSTALLATION TASKS

If you plan to use Kibana, you must manually create your Kibana index patterns and visualizations to explore and visualize data in Kibana.

If your cluster network provider enforces network isolation, allow network traffic between the projects that contain the OpenShift Logging operators.

3.4.1. Defining Kibana index patterns

An index pattern defines the Elasticsearch indices that you want to visualize. To explore and visualize data in Kibana, you must create an index pattern.

Prerequisites

- A user must have the **cluster-admin** role, the **cluster-reader** role, or both roles to view the **infra** and **audit** indices in Kibana. The default **kubeadmin** user has proper permissions to view these indices.
  If you can view the pods and logs in the **default**, **kube-** and **openshift-** projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

  ```
  $ oc auth can-i get pods/log -n <project>
  ```

  Example output

  ```
  yes
  ```

  NOTE

  The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the **default** output for audit logs.

- Elasticsearch documents must be indexed before you can create index patterns. This is done automatically, but it might take a few minutes in a new or updated cluster.

Procedure

To define index patterns and create visualizations in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher ▦ and select **Logging**.

2. Create your Kibana index patterns by clicking **Management → Index Patterns → Create index pattern**:

   - Each user must manually create index patterns when logging into Kibana the first time to see logs for their projects. Users must create an index pattern named **app** and use the **@timestamp** time field to view their container logs.

   - Each admin user must create index patterns when logged into Kibana the first time for the **app**, **infra**, and **audit** indices using the **@timestamp** time field.

3. Create Kibana Visualizations from the new index patterns.

3.4.2. Allowing traffic between projects when network isolation is enabled

Your cluster network provider might enforce network isolation. If so, you must allow network traffic between the projects that contain the operators deployed by OpenShift Logging.

Network isolation blocks network traffic between pods or services that are in different projects. OpenShift Logging installs the *OpenShift Elasticsearch Operator* in the **openshift-operators-redhat** project and the *Red Hat OpenShift Logging Operator* in the **openshift-logging** project. Therefore, you must allow traffic between these two projects.

OpenShift Container Platform offers two supported choices for the default Container Network Interface (CNI) network provider, OpenShift SDN and OVN-Kubernetes. These two providers implement various network isolation policies.

OpenShift SDN has three modes:

network policy

This is the default mode. If no policy is defined, it allows all traffic. However, if a user defines a policy, they typically start by denying all traffic and then adding exceptions. This process might break applications that are running in different projects. Therefore, explicitly configure the policy to allow traffic to egress from one logging-related project to the other.

multitenant

This mode enforces network isolation. You must join the two logging-related projects to allow traffic between them.

subnet

This mode allows all traffic. It does not enforce network isolation. No action is needed.

OVN-Kubernetes always uses a **network policy**. Therefore, as with OpenShift SDN, you must configure the policy to allow traffic to egress from one logging-related project to the other.

Procedure

- If you are using OpenShift SDN in **multitenant** mode, join the two projects. For example:

```
$ oc adm pod-network join-projects --to=openshift-operators-redhat openshift-logging
```

- Otherwise, for OpenShift SDN in **network policy** mode and OVN-Kubernetes, perform the following actions:

  a. Set a label on the **openshift-operators-redhat** namespace. For example:

  ```
  $ oc label namespace openshift-operators-redhat project=openshift-operators-redhat
  ```

b. Create a network policy object in the **openshift-logging** namespace that allows ingress from the **openshift-operators-redhat** project to the **openshift-logging** project. For example:

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

Additional resources

- About network policy
- About the OpenShift SDN default CNI network provider
- About the OVN-Kubernetes default Container Network Interface (CNI) network provider

CHAPTER 4. CONFIGURING YOUR LOGGING DEPLOYMENT

4.1. ABOUT THE CLUSTER LOGGING CUSTOM RESOURCE

To configure OpenShift Logging, you customize the **ClusterLogging** custom resource (CR).

4.1.1. About the ClusterLogging custom resource

To make changes to your OpenShift Logging environment, create and modify the **ClusterLogging** custom resource (CR).

Instructions for creating or modifying a CR are provided in this documentation as appropriate.

The following example shows a typical custom resource for OpenShift Logging.

Sample ClusterLogging custom resource (CR)

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging" 2
spec:
  managementState: "Managed" 3
  logStore:
    type: "elasticsearch" 4
    retentionPolicy:
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage:
        storageClassName: "gp2"
        size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization: 5
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 736Mi
        requests:
          cpu: 100m
          memory: 736Mi
      replicas: 1
  collection: 6
    logs:
      type: "fluentd"
      fluentd:
        resources:
          limits:
            memory: 736Mi
          requests:
            cpu: 100m
            memory: 736Mi
```

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

4.2. CONFIGURING THE LOGGING COLLECTOR

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

You can configure the CPU and memory limits for the log collector and move the log collector pods to specific nodes. All supported modifications to the log collector can be performed though the **spec.collection.log.fluentd** stanza in the **ClusterLogging** custom resource (CR).

4.2.1. About unsupported configurations

The supported way of configuring OpenShift Logging is by configuring it using the options described in this documentation. Do not use other configurations, as they are unsupported. Configuration paradigms might change across OpenShift Container Platform releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this documentation, your changes will disappear because the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator reconcile any differences. The Operators reverse everything to the defined state by default and by design.

NOTE

If you *must* perform configurations not described in the OpenShift Container Platform documentation, you *must* set your Red Hat OpenShift Logging Operator or OpenShift Elasticsearch Operator to **Unmanaged**. An unmanaged OpenShift Logging environment is *not supported* and does not receive updates until you return OpenShift Logging to **Managed**.

4.2.2. Viewing logging collector pods

You can view the Fluentd logging collector pods and the corresponding nodes that they are running on. The Fluentd logging collector pods run only in the **openshift-logging** project.

Procedure

- Run the following command in the **openshift-logging** project to view the Fluentd logging collector pods and their details:

```
$ oc get pods --selector component=fluentd -o wide -n openshift-logging
```

Example output

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

4.2.3. Configure log collector CPU and memory limits

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

Procedure

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

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

```
$ oc edit ClusterLogging instance

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

...

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

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

4.2.4. Advanced configuration for the log forwarder

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

- Chunk and chunk buffer sizes
- Chunk flushing behavior
- Chunk forwarding retry behavior

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

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

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

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

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

NOTE

These parameters are:

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

Table 4.1. Advanced Fluentd Configuration Parameters

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

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

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

Procedure

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

   ```
   $ oc edit ClusterLogging instance
   ```

2. Add or modify any of the following parameters:

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

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

   2 Specify the interval between chunk flushes.

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

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

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

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

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

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

   9 Specify the maximum size of the chunk buffer.

3. Verify that the Fluentd pods are redeployed:

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

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

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

Example fluentd.conf

```
<buffer>
 @type file
 path '/var/lib/fluentd/default'
 flush_mode interval
 flush_interval 5s
 flush_thread_count 3
 flush_at_shutdown true
 retry_type periodic
 retry_wait 1s
 retry_max_interval 300s
 retry_forever true
 queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '32' }"
 total_limit_size "#{ENV['TOTAL_LIMIT_SIZE'] ||  8589934592 }" #32M
 chunk_limit_size "#{ENV['BUFFER_SIZE_LIMIT'] || '8m'}"
 overflow_action throw_exception
</buffer>
```

4.2.5. Removing unused components if you do not use the default Elasticsearch log store

As an administrator, in the rare case that you forward logs to a third-party log store and do not use the default Elasticsearch log store, you can remove several unused components from your logging cluster.

In other words, if you do not use the default Elasticsearch log store, you can remove the internal Elasticsearch **logStore** and Kibana **visualization** components from the **ClusterLogging** custom resource (CR). Removing these components is optional but saves resources.

Prerequisites

- Verify that your log forwarder does not send log data to the default internal Elasticsearch cluster. Inspect the **ClusterLogForwarder** CR YAML file that you used to configure log forwarding. Verify that it *does not* have an **outputRefs** element that specifies **default**. For example:

  ```
  outputRefs:
  - default
  ```

> **WARNING**
>
> Suppose the **ClusterLogForwarder** CR forwards log data to the internal Elasticsearch cluster, and you remove the **logStore** component from the **ClusterLogging** CR. In that case, the internal Elasticsearch cluster will not be present to store the log data. This absence can cause data loss.

Procedure

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

```
$ oc edit ClusterLogging instance
```

2. If they are present, remove the **logStore** and **visualization** stanzas from the **ClusterLogging** CR.

3. Preserve the **collection** stanza of the **ClusterLogging** CR. The result should look similar to the following example:

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

4. Verify that the Fluentd pods are redeployed:

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

Additional resources

- Forwarding logs to third-party systems

4.3. CONFIGURING THE LOG STORE

OpenShift Container Platform uses Elasticsearch 6 (ES) to store and organize the log data.

You can make modifications to your log store, including:

- storage for your Elasticsearch cluster
- shard replication across data nodes in the cluster, from full replication to no replication
- external access to Elasticsearch data

Elasticsearch is a memory-intensive application. Each Elasticsearch node needs at least 16G of memory for both memory requests and limits, unless you specify otherwise in the **ClusterLogging** custom resource. The initial set of OpenShift Container Platform nodes might not be large enough to support the Elasticsearch cluster. You must add additional nodes to the OpenShift Container Platform cluster to run with the recommended or higher memory, up to a maximum of 64G for each Elasticsearch node.

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

4.3.1. Forwarding audit logs to the log store

By default, OpenShift Logging does not store audit logs in the internal OpenShift Container Platform Elasticsearch log store. You can send audit logs to this log store so, for example, you can view them in Kibana.

To send the audit logs to the default internal Elasticsearch log store, for example to view the audit logs in Kibana, you must use the Log Forwarding API.

IMPORTANT

The internal OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs. Verify that the system to which you forward audit logs complies with your organizational and governmental regulations and is properly secured. OpenShift Logging does not comply with those regulations.

Procedure

To use the Log Forward API to forward audit logs to the internal Elasticsearch instance:

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

   - Create a CR to send all log types to the internal Elasticsearch instance. You can use the following example without making any changes:

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

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

NOTE

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

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

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

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

1 This pipeline sends the audit logs to the internal Elasticsearch instance in addition to an external instance.

Additional resources

For more information on the Log Forwarding API, see Forwarding logs using the Log Forwarding API .

4.3.2. Configuring log retention time

You can configure a *retention policy* that specifies how long the default Elasticsearch log store keeps indices for each of the three log sources: infrastructure logs, application logs, and audit logs.

To configure the retention policy, you set a **maxAge** parameter for each log source in the **ClusterLogging** custom resource (CR). The CR applies these values to the Elasticsearch rollover schedule, which determines when Elasticsearch deletes the rolled-over indices.

Elasticsearch rolls over an index, moving the current index and creating a new index, when an index matches any of the following conditions:

- The index is older than the **rollover.maxAge** value in the **Elasticsearch** CR.
- The index size is greater than 40 GB × the number of primary shards.
- The index doc count is greater than 40960 KB × the number of primary shards.

Elasticsearch deletes the rolled-over indices based on the retention policy you configure. If you do not create a retention policy for any log sources, logs are deleted after seven days by default.

Prerequisites

- OpenShift Logging and the OpenShift Elasticsearch Operator must be installed.

Procedure

To configure the log retention time:

1. Edit the **ClusterLogging** CR to add or modify the **retentionPolicy** parameter:

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

(1) Specify the time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, **1d** for one day. Logs older than the **maxAge** are deleted. By default, logs are retained for seven days.

2. You can verify the settings in the **Elasticsearch** custom resource (CR).
For example, the Red Hat OpenShift Logging Operator updated the following **Elasticsearch** CR to configure a retention policy that includes settings to roll over active indices for the infrastructure logs every eight hours and the rolled-over indices are deleted seven days after rollover. OpenShift Container Platform checks every 15 minutes to determine if the indices need to be rolled over.

```
apiVersion: "logging.openshift.io/v1"
kind: "Elasticsearch"
metadata:
  name: "elasticsearch"
spec:
...
  indexManagement:
    policies: (1)
      - name: infra-policy
        phases:
          delete:
            minAge: 7d (2)
        hot:
          actions:
            rollover:
              maxAge: 8h (3)
      pollInterval: 15m (4)
...
```

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

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

(3) The index age for OpenShift Container Platform to consider when rolling over the indices. This value is determined from the **maxAge** you set in the **ClusterLogging** CR.

(4) When OpenShift Container Platform checks if the indices should be rolled over. This setting is the default and cannot be changed.

NOTE

Modifying the **Elasticsearch** CR is not supported. All changes to the retention policies must be made in the **ClusterLogging** CR.

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

```
$ oc get cronjob
```

Example output

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

4.3.3. Configuring CPU and memory requests for the log store

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

NOTE

In large-scale clusters, the default memory limit for the Elasticsearch proxy container might not be sufficient, causing the proxy container to be OOMKilled. If you experience this issue, increase the memory requests and limits for the Elasticsearch proxy.

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

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

```
$ oc edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
....
spec:
    logStore:
      type: "elasticsearch"
      elasticsearch:
        resources: (1)
          limits:
            memory: 16Gi
          requests:
            cpu: "1"
            memory: "64Gi"
        proxy: (2)
          resources:
            limits:
              memory: 100Mi
            requests:
              memory: 100Mi
```

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

(2) Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are **256Mi** for the memory request and **100m** for the CPU request.

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

For example:

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

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

4.3.4. Configuring replication policy for the log store

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

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

```
$ oc edit clusterlogging instance
```

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

....

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

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

- **FullRedundancy**. Elasticsearch fully replicates the primary shards for each index to every data node. This provides the highest safety, but at the cost of the highest amount of disk required and the poorest performance.
- **MultipleRedundancy**. Elasticsearch fully replicates the primary shards for each index to half of the data nodes. This provides a good tradeoff between safety and performance.
- **SingleRedundancy**. Elasticsearch makes one copy of the primary shards for each index. Logs are always available and recoverable as long as at least two data nodes exist. Better performance than MultipleRedundancy, when using 5 or more nodes. You cannot apply this policy on deployments of single Elasticsearch node.
- **ZeroRedundancy**. Elasticsearch does not make copies of the primary shards. Logs might be unavailable or lost in the event a node is down or fails. Use this mode when you are more concerned with performance than safety, or have implemented your own disk/PVC backup/restore strategy.

NOTE

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

4.3.5. Scaling down Elasticsearch pods

Reducing the number of Elasticsearch pods in your cluster can result in data loss or Elasticsearch performance degradation.

If you scale down, you should scale down by one pod at a time and allow the cluster to re-balance the shards and replicas. After the Elasticsearch health status returns to **green**, you can scale down by another pod.

NOTE

If your Elasticsearch cluster is set to **ZeroRedundancy**, you should not scale down your Elasticsearch pods.

4.3.6. Configuring persistent storage for the log store

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

WARNING

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

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

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

....

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

This example specifies each data node in the cluster is bound to a Persistent Volume Claim that requests "200G" of AWS General Purpose SSD (gp2) storage.

NOTE

If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the **LocalVolume** object. Elasticsearch cannot use raw block volumes.

4.3.7. Configuring the log store for emptyDir storage

You can use emptyDir with your log store, which creates an ephemeral deployment in which all of a pod's data is lost upon restart.

NOTE

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

1. Edit the **ClusterLogging** CR to specify emptyDir:

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

4.3.8. Performing an Elasticsearch rolling cluster restart

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

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.
- Install the OpenShift Container Platform es_util tool

Procedure

To perform a rolling cluster restart:

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

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

2. Get the names of the Elasticsearch pods:

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

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

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

4. Perform a shard synced flush using the OpenShift Container Platform es_util tool to ensure there are no pending operations waiting to be written to disk prior to shutting down:

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query="_flush/synced" -
XPOST
```

For example:

```
$ oc exec -c elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6  -c elasticsearch -- es_util --
query="_flush/synced" -XPOST
```

Example output

```
{"_shards":{"total":4,"successful":4,"failed":0},".security":
{"total":2,"successful":2,"failed":0},".kibana_1":{"total":2,"successful":2,"failed":0}}
```

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

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable" :
"primaries" } }'
```

For example:

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable" :
"primaries" } }'
```

Example output

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

6. After the command is complete, for each deployment you have for an ES cluster:

   a. By default, the OpenShift Container Platform Elasticsearch cluster blocks rollouts to their nodes. Use the following command to allow rollouts and allow the pod to pick up the changes:

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

For example:

```
$ oc rollout resume deployment/elasticsearch-cdm-0-1
```

Example output

```
deployment.extensions/elasticsearch-cdm-0-1 resumed
```

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

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

Example output

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

b. After the deployments are complete, reset the pod to disallow rollouts:

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

For example:

```
$ oc rollout pause deployment/elasticsearch-cdm-0-1
```

Example output

```
deployment.extensions/elasticsearch-cdm-0-1 paused
```

c. Check that the Elasticsearch cluster is in a **green** or **yellow** state:

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

NOTE

If you performed a rollout on the Elasticsearch pod you used in the previous commands, the pod no longer exists and you need a new pod name here.

For example:

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

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

(1) Make sure this parameter value is **green** or **yellow** before proceeding.

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

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

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable" : "all" }
}'
```

For example:

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable" : "all" }
}'
```

Example output

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

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

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

4.3.9. Exposing the log store service as a route

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

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

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

Internally, you can access the log store service using the log store cluster IP, which you can get by using either of the following commands:

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

Example output

```
172.30.183.229
```

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

Example output

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

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

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

Example output

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.
- You must have access to the project to be able to access to the logs.

Procedure

To expose the log store externally:

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

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

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

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

   Example output

   ```
   admin-ca
   ```

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

   a. Create a YAML file with the following:

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

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

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

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

   c. Create the route:

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

      Example output

      ```
      route.route.openshift.io/elasticsearch created
      ```

4. Check that the Elasticsearch service is exposed:

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

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

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

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

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

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

The response appears similar to the following:

Example output

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

4.4. CONFIGURING THE LOG VISUALIZER

OpenShift Container Platform uses Kibana to display the log data collected by OpenShift Logging.

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

4.4.1. Configuring CPU and memory limits

The OpenShift Logging components allow for adjustments to both the CPU and memory limits.

Procedure

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

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

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

...

spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources: (1)
        limits:
          memory: 16Gi
        requests:
          cpu: 200m
          memory: 16Gi
      storage:
        storageClassName: "gp2"
        size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources: (2)
        limits:
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 1Gi
      proxy:
        resources: (3)
          limits:
            memory: 100Mi
          requests:
            cpu: 100m
            memory: 100Mi
      replicas: 2
  collection:
    logs:
      type: "fluentd"
      fluentd:
        resources: (4)
          limits:
            memory: 736Mi
          requests:
            cpu: 200m
            memory: 736Mi
```

(1) Specify the CPU and memory limits and requests for the log store as needed. For Elasticsearch, you must adjust both the request value and the limit value.

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

(4) Specify the CPU and memory limits and requests for the log collector as needed.

4.4.2. Scaling redundancy for the log visualizer nodes

You can scale the pod that hosts the log visualizer for redundancy.

Procedure

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

```
$ oc edit ClusterLogging instance
```

```
$ oc edit ClusterLogging instance

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

....

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

(1) Specify the number of Kibana nodes.

4.5. CONFIGURING OPENSHIFT LOGGING STORAGE

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

4.5.1. Storage considerations for OpenShift Logging and OpenShift Container Platform

A persistent volume is required for each Elasticsearch deployment configuration. On OpenShift Container Platform this is achieved using persistent volume claims.

> **NOTE**
>
> If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the **LocalVolume** object. Elasticsearch cannot use raw block volumes.

The OpenShift Elasticsearch Operator names the PVCs using the Elasticsearch resource name.

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

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

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

NOTE

These low and high watermark values are Elasticsearch defaults in the current release. You can modify these default values. Although the alerts use the same default values, you cannot change these values in the alerts.

4.5.2. Additional resources

- Configuring persistent storage for the log store

4.6. CONFIGURING CPU AND MEMORY LIMITS FOR OPENSHIFT LOGGING COMPONENTS

You can configure both the CPU and memory limits for each of the OpenShift Logging components as needed.

4.6.1. Configuring CPU and memory limits

The OpenShift Logging components allow for adjustments to both the CPU and memory limits.

Procedure

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

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

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

...

spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources: (1)
        limits:
          memory: 16Gi
        requests:
          cpu: 200m
          memory: 16Gi
      storage:
        storageClassName: "gp2"
        size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources: (2)
        limits:
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 1Gi
      proxy:
        resources: (3)
          limits:
            memory: 100Mi
          requests:
            cpu: 100m
            memory: 100Mi
      replicas: 2
  collection:
    logs:
      type: "fluentd"
      fluentd:
        resources: (4)
          limits:
            memory: 736Mi
          requests:
            cpu: 200m
            memory: 736Mi
```

1 Specify the CPU and memory limits and requests for the log store as needed. For Elasticsearch, you must adjust both the request value and the limit value.

2 3 Specify the CPU and memory limits and requests for the log visualizer as needed.

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

4.7. USING TOLERATIONS TO CONTROL OPENSHIFT LOGGING POD PLACEMENT

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

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

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

Sample OpenShift Logging CR with tolerations

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

...

spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      tolerations: 1
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
      resources:
        limits:
          memory: 16Gi
        requests:
          cpu: 200m
          memory: 16Gi
      storage: {}
      redundancyPolicy: "ZeroRedundancy"
  visualization:
    type: "kibana"
    kibana:
      tolerations: 2
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
      resources:
        limits:
          memory: 2Gi
        requests:
          cpu: 100m
          memory: 1Gi
      replicas: 1
  collection:
    logs:
      type: "fluentd"
      fluentd:
        tolerations: 3
        - key: "logging"
          operator: "Exists"
          effect: "NoExecute"
          tolerationSeconds: 6000
        resources:
          limits:
            memory: 2Gi
        requests:
          cpu: 100m
          memory: 1Gi
```

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

4.7.1. Using tolerations to control the log store pod placement

You can control which nodes the log store pods runs on and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to the log store pods through the **ClusterLogging** custom resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all pods that do not tolerate the taint. Using a specific **key:value** pair that is not on other pods ensures only the log store pods can run on that node.

By default, the log store pods have the following toleration:

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

1. Use the following command to add a taint to a node where you want to schedule the OpenShift Logging pods:

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

   For example:

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

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

2. Edit the **logstore** section of the **ClusterLogging** CR to configure a toleration for the Elasticsearch pods:

   ```
     logStore:
       type: "elasticsearch"
       elasticsearch:
         nodeCount: 1
   ```

```
tolerations:
- key: "elasticsearch" (1)
  operator: "Exists" (2)
  effect: "NoExecute" (3)
  tolerationSeconds: 6000 (4)
```

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

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

4.7.2. Using tolerations to control the log visualizer pod placement

You can control the node where the log visualizer pod runs and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to the log visualizer pod through the **ClusterLogging** custom resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all pods that do not tolerate the taint. Using a specific **key:value** pair that is not on other pods ensures only the Kibana pod can run on that node.

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

1. Use the following command to add a taint to a node where you want to schedule the log visualizer pod:

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

   For example:

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

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

2. Edit the **visualization** section of the **ClusterLogging** CR to configure a toleration for the Kibana pod:

   ```
   visualization:
     type: "kibana"
   ```

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

(1) Specify the key that you added to the node.

(2) Specify the **Exists** operator to require the **key**/**value**/**effect** parameters to match.

(3) Specify the **NoExecute** effect.

(4) Optionally, specify the **tolerationSeconds** parameter to set how long a pod can remain bound to a node before being evicted.

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

4.7.3. Using tolerations to control the log collector pod placement

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

You apply tolerations to logging collector pods through the **ClusterLogging** custom resource (CR) and apply taints to a node through the node specification. You can use taints and tolerations to ensure the pod does not get evicted for things like memory and CPU issues.

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

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

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

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

   For example:

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

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

2. Edit the **collection** stanza of the **ClusterLogging** custom resource (CR) to configure a toleration for the logging collector pods:

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

1 Specify the key that you added to the node.

2 Specify the **Exists** operator to require the **key**/**value**/**effect** parameters to match.

3 Specify the **NoExecute** effect.

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

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

4.7.4. Additional resources

For more information about taints and tolerations, see Controlling pod placement using node taints .

4.8. MOVING OPENSHIFT LOGGING RESOURCES WITH NODE SELECTORS

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

4.8.1. Moving OpenShift Logging resources

You can configure the Cluster Logging Operator to deploy the pods for OpenShift Logging components, such as Elasticsearch and Kibana, to different nodes. You cannot move the Cluster Logging Operator pod from its installed location.

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed. These features are not installed by default.

Procedure

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

```
$ oc edit ClusterLogging instance
```

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

...

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 1
        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: 2
        node-role.kubernetes.io/infra: ''
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana

...
```

1 2 Add a **nodeSelector** parameter with the appropriate value to the component you want to move. You can use a **nodeSelector** in the format shown or use **<key>: <value>** pairs, based on the value specified for the node.

Verification

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

For example:

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

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

Example output

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

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

```
$ oc get nodes
```

Example output

```
NAME                                        STATUS   ROLES          AGE   VERSION
ip-10-0-133-216.us-east-2.compute.internal   Ready    master         60m   v1.21.0
ip-10-0-139-146.us-east-2.compute.internal   Ready    master         60m   v1.21.0
ip-10-0-139-192.us-east-2.compute.internal   Ready    worker         51m   v1.21.0
ip-10-0-139-241.us-east-2.compute.internal   Ready    worker         51m   v1.21.0
ip-10-0-147-79.us-east-2.compute.internal    Ready    worker         51m   v1.21.0
ip-10-0-152-241.us-east-2.compute.internal   Ready    master         60m   v1.21.0
ip-10-0-139-48.us-east-2.compute.internal    Ready    infra          51m   v1.21.0
```

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

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

Example output

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

- To move the Kibana pod, edit the **ClusterLogging** CR to add a node selector:

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

...

spec:

...

  visualization:
```

```
  kibana:
    nodeSelector: (1)
      node-role.kubernetes.io/infra: ''
    proxy:
      resources: null
    replicas: 1
    resources: null
  type: kibana
```

(1) Add a node selector to match the label in the node specification.

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

```
$ oc get pods
```

Example output

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

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

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

Example output

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

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

```
$ oc get pods
```

Example output

```
NAME                                               READY   STATUS    RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7       1/1     Running   0          30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running   0          29m
```

```
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78   2/2     Running   0          29m
fluentd-42dzz                                  1/1     Running   0          29m
fluentd-d74rq                                  1/1     Running   0          29m
fluentd-m5vr9                                  1/1     Running   0          29m
fluentd-nkxl7                                  1/1     Running   0          29m
fluentd-pdvqb                                  1/1     Running   0          29m
fluentd-tflh6                                  1/1     Running   0          29m
kibana-7d85dcffc8-bfpfp                        2/2     Running   0          62s
```

4.9. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD

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

We recommend setting **RateLimitIntervalSec=30s** and **RateLimitBurst=10000** (or even higher if necessary) to prevent the journal from losing entries.

4.9.1. Configuring systemd-journald for OpenShift 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 OpenShift 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 Butane config file, **40-worker-custom-journald.bu**, that includes an **/etc/systemd/journald.conf** file with the required settings.

   NOTE

   See "Creating machine configs with Butane" for information about Butane.

```
variant: openshift
version: 4.8.0
metadata:
  name: 40-worker-custom-journald
  labels:
    machineconfiguration.openshift.io/role: "worker"
storage:
  files:
  - path: /etc/systemd/journald.conf
    mode: 0644 1
    overwrite: true
    contents:
      inline: |
        Compress=yes 2
        ForwardToConsole=no 3
        ForwardToSyslog=no
    MaxRetentionSec=1month (4)
    RateLimitBurst=10000 (5)
    RateLimitIntervalSec=30s
    Storage=persistent (6)
    SyncIntervalSec=1s (7)
    SystemMaxUse=8g (8)
    SystemKeepFree=20% (9)
    SystemMaxFileSize=10M (10)
```

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

(2) Specify whether you want logs compressed before they are written to the file system. Specify **yes** to compress the message or **no** to not compress. The default is **yes**.

(3) Configure whether to forward log messages. Defaults to **no** for each. Specify:

- **ForwardToConsole** to forward logs to the system console.
- **ForwardToKsmg** to forward logs to the kernel log buffer.
- **ForwardToSyslog** to forward to a syslog daemon.
- **ForwardToWall** to forward messages as wall messages to all logged-in users.

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

(5) Configure rate limiting. If more logs are received than what is specified in **RateLimitBurst** during the time interval defined by **RateLimitIntervalSec**, all further messages within the interval are dropped until the interval is over. It is recommended to set **RateLimitIntervalSec=30s** and **RateLimitBurst=10000**, which are the defaults.

(6) Specify how logs are stored. The default is **persistent**:

- **volatile** to store logs in memory in **/var/log/journal/**.
- **persistent** to store logs to disk in **/var/log/journal/**. systemd creates the directory if it does not exist.
- **auto** to store logs in **/var/log/journal/** if the directory exists. If it does not exist, systemd temporarily stores logs in **/run/systemd/journal**.
- **none** to not store logs. systemd drops all logs.

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

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

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

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

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

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

2. Use Butane to generate a **MachineConfig** object file, **40-worker-custom-journald.yaml**, containing the configuration to be delivered to the nodes:

```
$ butane 40-worker-custom-journald.bu -o 40-worker-custom-journald.yaml
```

3. Apply the machine config. For example:

```
$ oc apply -f 40-worker-custom-journald.yaml
```

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

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

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

Example output

```
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
```

4.10. MAINTENANCE AND SUPPORT

4.10.1. About unsupported configurations

The supported way of configuring OpenShift Logging is by configuring it using the options described in this documentation. Do not use other configurations, as they are unsupported. Configuration paradigms might change across OpenShift Container Platform releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this documentation, your changes will disappear because the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator reconcile any differences. The Operators reverse everything to the defined state by default and by design.

NOTE

If you *must* perform configurations not described in the OpenShift Container Platform documentation, you *must* set your Red Hat OpenShift Logging Operator or OpenShift Elasticsearch Operator to **Unmanaged**. An unmanaged OpenShift Logging environment is *not supported* and does not receive updates until you return OpenShift Logging to **Managed**.

4.10.2. Unsupported configurations

You must set the Red Hat OpenShift Logging Operator to the unmanaged state to modify the following components:

- The **Elasticsearch** CR
- The Kibana deployment
- The **fluent.conf** file
- The Fluentd daemon set

You must set the OpenShift Elasticsearch Operator to the unmanaged state to modify the following component:

- the Elasticsearch deployment files.

Explicitly unsupported cases include:

- **Configuring default log rotation**. You cannot modify the default log rotation configuration.
- **Configuring the collected log location**. You cannot change the location of the log collector output file, which by default is **/var/log/fluentd/fluentd.log**.
- **Throttling log collection**. You cannot throttle down the rate at which the logs are read in by the log collector.
- **Configuring the logging collector using environment variables**. You cannot use environment variables to modify the log collector.
- **Configuring how the log collector normalizes logs**. You cannot modify default log normalization.

4.10.3. Support policy for unmanaged Operators

The *management state* of an Operator determines whether an Operator is actively managing the resources for its related component in the cluster as designed. If an Operator is set to an *unmanaged* state, it does not respond to changes in configuration nor does it receive updates.

While this can be helpful in non-production clusters or during debugging, Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades.

An Operator can be set to an unmanaged state using the following methods:

- **Individual Operator configuration**
  Individual Operators have a **managementState** parameter in their configuration. This can be accessed in different ways, depending on the Operator. For example, the Red Hat OpenShift Logging Operator accomplishes this by modifying a custom resource (CR) that it manages, while the Cluster Samples Operator uses a cluster-wide configuration resource.

  Changing the **managementState** parameter to **Unmanaged** means that the Operator is not actively managing its resources and will take no action related to the related component. Some Operators might not support this management state as it might damage the cluster and require manual recovery.

  > **WARNING**
  >
  > Changing individual Operators to the **Unmanaged** state renders that particular component and functionality unsupported. Reported issues must be reproduced in **Managed** state for support to proceed.

- **Cluster Version Operator (CVO) overrides**
  The **spec.overrides** parameter can be added to the CVO's configuration to allow administrators to provide a list of overrides to the CVO's behavior for a component. Setting the **spec.overrides[].unmanaged** parameter to **true** for a component blocks cluster upgrades and alerts the administrator after a CVO override has been set:

  ```
  Disabling ownership via cluster version overrides prevents upgrades. Please remove
  overrides before continuing.
  ```

  > **WARNING**
  >
  > Setting a CVO override puts the entire cluster in an unsupported state. Reported issues must be reproduced after removing any overrides for support to proceed.

CHAPTER 5. VIEWING LOGS FOR A RESOURCE

You can view the logs for various resources, such as builds, deployments, and pods by using the OpenShift CLI (oc) and the web console.

NOTE

Resource logs are a default feature that provides limited log viewing capability. To enhance your log retrieving and viewing experience, it is recommended that you install OpenShift Logging. OpenShift Logging aggregates all the logs from your OpenShift Container Platform cluster, such as node system audit logs, application container logs, and infrastructure logs, into a dedicated log store. You can then query, discover, and visualize your log data through the Kibana interface. Resource logs do not access the OpenShift Logging log store.

5.1. VIEWING RESOURCE LOGS

You can view the log for various resources in the OpenShift CLI (oc) and web console. Logs read from the tail, or end, of the log.

Prerequisites

- Access to the OpenShift CLI (oc).

Procedure (UI)

1. In the OpenShift Container Platform console, navigate to **Workloads** → **Pods** or navigate to the pod through the resource you want to investigate.

   NOTE

   Some resources, such as builds, do not have pods to query directly. In such instances, you can locate the **Logs** link on the **Details** page for the resource.

2. Select a project from the drop-down menu.
3. Click the name of the pod you want to investigate.
4. Click **Logs**.

Procedure (CLI)

- View the log for a specific pod:

```
$ oc logs -f <pod_name> -c <container_name>
```

where:

-f

Optional: Specifies that the output follows what is being written into the logs.

<pod_name>

Specifies the name of the pod.

<container_name>

Optional: Specifies the name of a container. When a pod has more than one container, you must specify the container name.

For example:

```
$ oc logs ruby-58cd97df55-mww7r
```

```
$ oc logs -f ruby-57f7f4855b-znl92 -c ruby
```

The contents of log files are printed out.

- View the log for a specific resource:

```
$ oc logs <object_type>/<resource_name> (1)
```

(1) Specifies the resource type and name.

For example:

```
$ oc logs deployment/ruby
```

The contents of log files are printed out.

CHAPTER 6. VIEWING CLUSTER LOGS BY USING KIBANA

OpenShift Logging includes a web console for visualizing collected log data. Currently, OpenShift Container Platform deploys the Kibana console for visualization.

Using the log visualizer, you can do the following with your data:

- search and browse the data using the **Discover** tab.
- chart and map the data using the **Visualize** tab.
- create and view custom dashboards using the **Dashboard** tab.

Use and configuration of the Kibana interface is beyond the scope of this documentation. For more information, on using the interface, see the Kibana documentation.

NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the **default** output for audit logs.

6.1. DEFINING KIBANA INDEX PATTERNS

An index pattern defines the Elasticsearch indices that you want to visualize. To explore and visualize data in Kibana, you must create an index pattern.

Prerequisites

- A user must have the **cluster-admin** role, the **cluster-reader** role, or both roles to view the **infra** and **audit** indices in Kibana. The default **kubeadmin** user has proper permissions to view these indices.
  If you can view the pods and logs in the **default**, **kube-** and **openshift-** projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

  ```
  $ oc auth can-i get pods/log -n <project>
  ```

  Example output

  ```
  yes
  ```

  NOTE

  The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the **default** output for audit logs.

- Elasticsearch documents must be indexed before you can create index patterns. This is done automatically, but it might take a few minutes in a new or updated cluster.

Procedure

To define index patterns and create visualizations in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher ⊞ and select **Logging**.
2. Create your Kibana index patterns by clicking **Management → Index Patterns → Create index pattern**:
   - Each user must manually create index patterns when logging into Kibana the first time to see logs for their projects. Users must create an index pattern named **app** and use the **@timestamp** time field to view their container logs.
   - Each admin user must create index patterns when logged into Kibana the first time for the **app**, **infra**, and **audit** indices using the **@timestamp** time field.
3. Create Kibana Visualizations from the new index patterns.

6.2. VIEWING CLUSTER LOGS IN KIBANA

You view cluster logs in the Kibana web console. The methods for viewing and visualizing your data in Kibana that are beyond the scope of this documentation. For more information, refer to the Kibana documentation.

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.
- Kibana index patterns must exist.
- A user must have the **cluster-admin** role, the **cluster-reader** role, or both roles to view the **infra** and **audit** indices in Kibana. The default **kubeadmin** user has proper permissions to view these indices.
  If you can view the pods and logs in the **default**, **kube-** and **openshift-** projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

  ```
  $ oc auth can-i get pods/log -n <project>
  ```

  Example output

  ```
  yes
  ```

  NOTE

  The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the **default** output for audit logs.

Procedure

To view logs in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher ▦ and select **Logging**.
2. Log in using the same credentials you use to log in to the OpenShift Container Platform console.
   The Kibana interface launches.
3. In Kibana, click **Discover**.
4. Select the index pattern you created from the drop-down menu in the top-left corner: **app**, **audit**, or **infra**.
   The log data displays as time-stamped documents.
5. Expand one of the time-stamped documents.
6. Click the **JSON** tab to display the log entry for that document.

Example 6.1. Sample infrastructure log entry in Kibana

```
{
  "_index": "infra-000001",
  "_type": "_doc",
  "_id": "YmJmYTBlNDkZTRmLTliMGQtMjE3NmFiOGUyOWM3",
  "_version": 1,
  "_score": null,
  "_source": {
    "docker": {
      "container_id": "f85fa55bbef7bb783f041066be1e7c267a6b88c4603dfce213e32c1"
    },
    "kubernetes": {
      "container_name": "registry-server",
      "namespace_name": "openshift-marketplace",
      "pod_name": "redhat-marketplace-n64gc",
      "container_image": "registry.redhat.io/redhat/redhat-marketplace-index:v4.7",
      "container_image_id": "registry.redhat.io/redhat/redhat-marketplace-
index@sha256:65fc0c45aabb95809e376feb065771ecda9e5e59cc8b3024c4545c168f",
      "pod_id": "8f594ea2-c866-4b5c-a1c8-a50756704b2a",
      "host": "ip-10-0-182-28.us-east-2.compute.internal",
      "master_url": "https://kubernetes.default.svc",
      "namespace_id": "3abab127-7669-4eb3-b9ef-44c04ad68d38",
      "namespace_labels": {
        "openshift_io/cluster-monitoring": "true"
      },
      "flat_labels": [
        "catalogsource_operators_coreos_com/update=redhat-marketplace"
      ]
    },
    "message": "time=\"2020-09-23T20:47:03Z\" level=info msg=\"serving registry\"
database=/database/index.db port=50051",
    "level": "unknown",
    "hostname": "ip-10-0-182-28.internal",
    "pipeline_metadata": {
      "collector": {
        "ipaddr4": "10.0.182.28",
        "inputname": "fluent-plugin-systemd",
        "name": "fluentd",
          "received_at": "2020-09-23T20:47:15.007583+00:00",
          "version": "1.7.4 1.6.0"
        }
      },
      "@timestamp": "2020-09-23T20:47:03.422465+00:00",
      "viaq_msg_id": "YmJmYTBlNDktMDMGQtMjE3NmFiOGUyOWM3",
      "openshift": {
        "labels": {
          "logging": "infra"
        }
      }
    },
    "fields": {
      "@timestamp": [
        "2020-09-23T20:47:03.422Z"
      ],
      "pipeline_metadata.collector.received_at": [
        "2020-09-23T20:47:15.007Z"
      ]
    },
    "sort": [
      1600894023422
    ]
  }
```

CHAPTER 7. FORWARDING LOGS TO EXTERNAL THIRD-PARTY LOGGING SYSTEMS

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

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

NOTE

To send audit logs to the default internal Elasticsearch log store, use the Cluster Log Forwarder as described in Forward audit logs to the log store .

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

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

IMPORTANT

You cannot use the config map methods and the Cluster Log Forwarder in the same cluster.

7.1. ABOUT FORWARDING LOGS TO THIRD-PARTY SYSTEMS

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

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

  - **cloudwatch**. Amazon CloudWatch, a monitoring and log storage service hosted by Amazon Web Services (AWS).
  - **loki**. Grafana Loki, a horizontally scalable, highly available, multi-tenant log aggregation system.
  - **kafka**. A Kafka broker. The **kafka** output can use a TCP or TLS connection.
  - **default**. The internal OpenShift Container Platform Elasticsearch instance. You are not required to configure the default output. If you do configure a **default** output, you receive an error message because the **default** output is reserved for the Red Hat OpenShift Logging Operator.

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

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

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

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

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

Note the following:

- If a **ClusterLogForwarder** CR object exists, logs are not forwarded to the default Elasticsearch instance, unless there is a pipeline with the **default** output.
- By default, OpenShift Logging sends container and infrastructure logs to the default internal Elasticsearch log store defined in the **ClusterLogging** custom resource. However, it does not send audit logs to the internal store because it does not provide secure storage. If this default configuration meets your needs, do not configure the Log Forwarding API.
- If you do not define a pipeline for a log type, the logs of the undefined types are dropped. For example, if you specify a pipeline for the **application** and **audit** types, but do not specify a pipeline for the **infrastructure** type, **infrastructure** logs are dropped.
- You can use multiple types of outputs in the **ClusterLogForwarder** custom resource (CR) to send logs to servers that support different protocols.
- The internal OpenShift Container Platform Elasticsearch instance does not provide secure storage for audit logs. We recommend you ensure that the system to which you forward audit

logs is compliant with your organizational and governmental regulations and is properly secured. OpenShift Logging does not comply with those regulations.

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

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

Sample log forwarding outputs and pipelines

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

```
- name: my-app (11)
  inputRefs:
   - my-app-logs
  outputRefs:
   - default
- inputRefs: (12)
   - application
  outputRefs:
   - kafka-app
  labels:
    datacenter: "south"
```

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

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

3. Configuration for an secure Elasticsearch output using a secret with a secure URL.
    - A name to describe the output.
    - The type of output: **elasticsearch**.
    - The secure URL and port of the Elasticsearch instance as a valid absolute URL, including the prefix.
    - The secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project.

4. Configuration for an insecure Elasticsearch output:
    - A name to describe the output.
    - The type of output: **elasticsearch**.
    - The insecure URL and port of the Elasticsearch instance as a valid absolute URL, including the prefix.

5. Configuration for a Kafka output using a client-authenticated TLS communication over a secure URL
    - A name to describe the output.
    - The type of output: **kafka**.
    - Specify the URL and port of the Kafka broker as a valid absolute URL, including the prefix.

6. Configuration for an input to filter application logs from the **my-namespace** project.

7. Configuration for a pipeline to send audit logs to the secure external Elasticsearch instance:
    - A name to describe the pipeline.
    - The **inputRefs** is the log type, in this example **audit**.
    - The **outputRefs** is the name of the output to use, in this example **elasticsearch-secure** to forward to the secure Elasticsearch instance and **default** to forward to the internal Elasticsearch instance.

- Optional: Labels to add to the logs.

8 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.

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

10 Configuration for a pipeline to send infrastructure logs to the insecure external Elasticsearch instance.

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

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

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

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

Fluentd log handling when the external log aggregator is unavailable

If your external logging aggregator becomes unavailable and cannot receive logs, Fluentd continues to collect logs and stores them in a buffer. When the log aggregator becomes available, log forwarding resumes, including the buffered logs. If the buffer fills completely, Fluentd stops collecting logs. OpenShift Container Platform rotates the logs and deletes them. You cannot adjust the buffer size or add a persistent volume claim (PVC) to the Fluentd daemon set or pods.

7.2. SUPPORTED LOG DATA OUTPUT TYPES IN OPENSHIFT LOGGING 5.1

Red Hat OpenShift Logging version 5.1 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range target log collectors that ingest these protocols.

Output types	Protocols	Tested with

Output types	Protocols	Tested with
elasticsearch	elasticsearch	Elasticsearch 6.8.1 Elasticsearch 6.8.4 Elasticsearch 7.12.2
fluentdForward	fluentd forward v1	fluentd 1.7.4 logstash 7.10.1
kafka	kafka 0.11	kafka 2.4.1 kafka 2.7.0
syslog	RFC-3164, RFC-5424	rsyslog-8.39.0

NOTE

Previously, the syslog output supported only RFC-3164. The current syslog output adds support for RFC-5424.

7.3. SUPPORTED LOG DATA OUTPUT TYPES IN OPENSHIFT LOGGING 5.2

Red Hat OpenShift Logging version 5.2 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range target log collectors that ingest these protocols.

Output types	Protocols	Tested with
Amazon CloudWatch	REST over HTTPS	The current version of Amazon CloudWatch
elasticsearch	elasticsearch	Elasticsearch 6.8.1 Elasticsearch 6.8.4 Elasticsearch 7.12.2
fluentdForward	fluentd forward v1	fluentd 1.7.4 logstash 7.10.1
Grafana Loki	REST over HTTP and HTTPS	Loki 2.3.0 deployed on OCP and Grafana labs

Output types	Protocols	Tested with
kafka	kafka 0.11	kafka 2.4.1 kafka 2.7.0
syslog	RFC-3164, RFC-5424	rsyslog-8.39.0

NOTE

Previously, the syslog output supported only RFC-3164. The current syslog output adds support for RFC-5424.

7.4. FORWARDING LOGS TO AN EXTERNAL ELASTICSEARCH INSTANCE

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

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

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

NOTE

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

Prerequisites

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

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
```

```
  - name: elasticsearch-insecure (3)
    type: "elasticsearch" (4)
    url: http://elasticsearch.insecure.com:9200 (5)
  - name: elasticsearch-secure
    type: "elasticsearch"
    url: https://elasticsearch.secure.com:9200 (6)
    secret:
       name: es-secret (7)
pipelines:
  - name: application-logs (8)
    inputRefs: (9)
    - application
    - audit
    outputRefs:
    - elasticsearch-secure (10)
    - default (11)
    parse: json (12)
    labels:
      myLabel: "myValue" (13)
  - name: infrastructure-audit-logs (14)
    inputRefs:
    - infrastructure
    outputRefs:
    - elasticsearch-insecure
    labels:
      logs: "audit-infra"
```

1. The name of the **ClusterLogForwarder** CR must be **instance**.
2. The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
3. Specify a name for the output.
4. Specify the **elasticsearch** type.
5. Specify the URL and port of the external Elasticsearch instance as a valid absolute URL. You can use the **http** (insecure) or **https** (secure HTTP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP Address.
6. For a secure connection, you can specify an **https** or **http** URL that you authenticate by specifying a **secret**.
7. For an **https** prefix, specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project, and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent. Otherwise, for **http** and **https** prefixes, you can specify a secret that contains a username and password. For more information, see the following "Example: Setting secret that contains a username and password."
8. Optional: Specify a name for the pipeline.
9. Specify which log types to forward by using the pipeline: **application, infrastructure**, or **audit**.

10 Specify the name of the output to use when forwarding logs with this pipeline.

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

12 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.

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

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

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

2. Create the CR object:

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

Example: Setting a secret that contains a username and password

You can use a secret that contains a username and password to authenticate a secure connection to an external Elasticsearch instance.

For example, if you cannot use mutual TLS (mTLS) keys because a third party operates the Elasticsearch instance, you can use HTTP or HTTPS and set a secret that contains the username and password.

1. Create a **Secret** YAML file similar to the following example. Use base64-encoded values for the the **username** and **password** fields. The secret type is opaque by default.

```
apiVersion: v1
kind: Secret
metadata:
  name: openshift-test-secret
data:
  username: dGVzdHVzZXJuYW1lCg==
  password: dGVzdHBhc3N3b3JkCg==
```

2. Create the secret:

```
$ oc create secret -n openshift-logging openshift-test-secret.yaml
```

3. Specify the name of the secret in the **ClusterLogForwarder** CR:

```
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
   - name: elasticsearch
     type: "elasticsearch"
     url: https://elasticsearch.secure.com:9200
     secret:
        name: openshift-test-secret
```

NOTE

In the value of the **url** field, the prefix can be **http** or **https**.

4. Create the CR object:

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

7.5. FORWARDING LOGS USING THE FLUENTD FORWARD PROTOCOL

You can use the Fluentd **forward** protocol to send a copy of your logs to an external log aggregator that is configured to accept the protocol instead of, or in addition to, the default Elasticsearch log store. You are responsible for configuring the external log aggregator to receive the logs from OpenShift Container Platform.

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

NOTE

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

Prerequisites

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

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
```

```
  - name: fluentd-server-secure (3)
    type: fluentdForward (4)
    url: 'tls://fluentdserver.security.example.com:24224' (5)
    secret: (6)
       name: fluentd-secret
       passphrase: phrase (7)
  - name: fluentd-server-insecure
    type: fluentdForward
    url: 'tcp://fluentdserver.home.example.com:24224'
pipelines:
  - name: forward-to-fluentd-secure (8)
    inputRefs:  (9)
    - application
    - audit
    outputRefs:
    - fluentd-server-secure (10)
    - default (11)
    parse: json (12)
    labels:
      clusterId: "C1234" (13)
  - name: forward-to-fluentd-insecure (14)
    inputRefs:
    - infrastructure
    outputRefs:
    - fluentd-server-insecure
    labels:
      clusterId: "C1234"
```

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

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

3 Specify a name for the output.

4 Specify the **fluentdForward** type.

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

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

7 Optional: Specify the password or passphrase that protects the private key file.

8 Optional: Specify a name for the pipeline.

9 Specify which log types to forward by using the pipeline: **application, infrastructure**, or **audit**.

10 Specify the name of the output to use when forwarding logs with this pipeline.

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

12 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift

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

14 Optional: Configure multiple outputs to forward logs to other external log aggregtors of any supported type:

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

2. Create the CR object:

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

7.6. FORWARDING LOGS USING THE SYSLOG PROTOCOL

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

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

NOTE

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

Prerequisites

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

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
```

```
outputs:
 - name: rsyslog-east (3)
   type: syslog (4)
   syslog: (5)
     facility: local0
     rfc: RFC3164
     payloadKey: message
     severity: informational
   url: 'tls://rsyslogserver.east.example.com:514' (6)
   secret: (7)
      name: syslog-secret
 - name: rsyslog-west
   type: syslog
   syslog:
    appName: myapp
    facility: user
    msgID: mymsg
    procID: myproc
    rfc: RFC5424
    severity: debug
   url: 'udp://rsyslogserver.west.example.com:514'
pipelines:
 - name: syslog-east (8)
   inputRefs: (9)
   - audit
   - application
   outputRefs: (10)
   - rsyslog-east
   - default (11)
   parse: json (12)
   labels:
     secure: "true" (13)
     syslog: "east"
 - name: syslog-west (14)
   inputRefs:
   - infrastructure
   outputRefs:
   - rsyslog-west
   - default
   labels:
     syslog: "west"
```

1. The name of the **ClusterLogForwarder** CR must be **instance**.
2. The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
3. Specify a name for the output.
4. Specify the **syslog** type.
5. Optional: Specify the syslog parameters, listed below.
6. Specify the URL and port of the external syslog instance. You can use the **udp** (insecure), **tcp** (insecure) or **tls** (secure TCP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP address.

7 If using a **tls** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project, and must

8 Optional: Specify a name for the pipeline.

9 Specify which log types to forward by using the pipeline: **application, infrastructure**, or **audit**.

10 Specify the name of the output to use when forwarding logs with this pipeline.

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

12 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.

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

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

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

2. Create the CR object:

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

7.6.1. Syslog parameters

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

- facility: The syslog facility. The value can be a decimal integer or a case-insensitive keyword:
  - **0** or **kern** for kernel messages
  - **1** or **user** for user-level messages, the default.
  - **2** or **mail** for the mail system
  - **3** or **daemon** for system daemons
  - **4** or **auth** for security/authentication messages
  - **5** or **syslog** for messages generated internally by syslogd
  - **6** or **lpr** for the line printer subsystem

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

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

  NOTE

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

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

7.6.2. Additional RFC5424 syslog parameters

The following parameters apply to RFC5424:

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

7.7. FORWARDING LOGS TO AMAZON CLOUDWATCH

You can forward logs to Amazon CloudWatch, a monitoring and log storage service hosted by Amazon Web Services (AWS). You can forward logs to CloudWatch in addition to, or instead of, the default OpenShift Logging-managed Elasticsearch log store.

To configure log forwarding to CloudWatch, you must create a **ClusterLogForwarder** custom resource (CR) with an output for CloudWatch, and a pipeline that uses the output.

Procedure

1. Create a **Secret** YAML file that uses the **aws_access_key_id** and **aws_secret_access_key** fields to specify your base64-encoded AWS credentials. For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: cw-secret
  namespace: openshift-logging
data:
  aws_access_key_id: QUtJQUlPU0ZPRE5ON0VYQU1QTEUK
  aws_secret_access_key:
d0phbHJYVXRuRkVNSS9LN01ERU5HL2JQeFJmaUNZRVhBTVBMRUtFWQo=
```

2. Create the secret. For example:

```
$ oc apply -f cw-secret.yaml
```

3. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object. In the file, specify the name of the secret. For example:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance ①
  namespace: openshift-logging ②
spec:
  outputs:
   - name: cw ③
     type: cloudwatch ④
     cloudwatch:
       groupBy: logType ⑤
       groupPrefix: <group prefix> ⑥
       region: us-east-2 ⑦
     secret:
      name: cw-secret (8)
  pipelines:
    - name: infra-logs (9)
      inputRefs: (10)
        - infrastructure
        - audit
        - application
      outputRefs:
        - cw (11)
```

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

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

(3) Specify a name for the output.

(4) Specify the **cloudwatch** type.

(5) Optional: Specify how to group the logs:

- **logType** creates log groups for each log type
- **namespaceName** creates a log group for each application name space. It also creates separate log groups for infrastructure and audit logs.
- **namespaceUUID** creates a new log groups for each application namespace UUID. It also creates separate log groups for infrastructure and audit logs.

(6) Optional: Specify a string to replace the default **infrastructureName** prefix in the names of the log groups.

(7) Specify the AWS region.

(8) Specify the name of the secret that contains your AWS credentials.

(9) Optional: Specify a name for the pipeline.

(10) Specify which log types to forward by using the pipeline: **application, infrastructure**, or **audit**.

(11) Specify the name of the output to use when forwarding logs with this pipeline.

4. Create the CR object:

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

Example: Using ClusterLogForwarder with Amazon CloudWatch

Here, you see an example **ClusterLogForwarder** custom resource (CR) and the log data that it outputs to Amazon CloudWatch.

Suppose that you are running an OpenShift Container Platform cluster named **mycluster**. The following command returns the cluster's **infrastructureName**, which you will use to compose **aws** commands later on:

```
$ oc get Infrastructure/cluster -ojson | jq .status.infrastructureName
"mycluster-7977k"
```

To generate log data for this example, you run a **busybox** Pod in a namespace called **app**. The **busybox** Pod writes a message to stdout every three seconds:

```
$ oc run busybox --image=busybox -- sh -c 'while true; do echo "My life is my message"; sleep 3;
done'
$ oc logs -f busybox
My life is my message
My life is my message
My life is my message
...
```

You can look up the UUID of the **app** namespace where the **busybox** Pod runs:

```
$ oc get ns/app -ojson | jq .metadata.uid
"794e1e1a-b9f5-4958-a190-e76a9b53d7bf"
```

In your **ClusterLogForwarder** custom resource (CR), you configure the **infrastructure**, **audit**, and **application** log types as inputs to the **all-logs** pipeline. You also connect this pipeline to **cw** output, which forwards the logs to a CloudWatch instance in the **us-east-2** region:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
   - name: cw
     type: cloudwatch
     cloudwatch:
       groupBy: logType
       region: us-east-2
     secret:
        name: cw-secret
  pipelines:
    - name: all-logs
      inputRefs:
        - infrastructure
        - audit
        - application
      outputRefs:
        - cw
```

Each region in CloudWatch contains three levels of objects:

- log group
  - log stream
    - log event

With **groupBy: logType** in the **ClusterLogForwarding** CR, the three log types in the **inputRefs** produce three log groups in Amazon Cloudwatch:

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.application"
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"
```

Each of the log groups contains log streams:

```
$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.application | jq
.logStreams[].logStreamName
"kubernetes.var.log.containers.busybox_app_busybox-
da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76.log"
```

```
$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.audit | jq
.logStreams[].logStreamName
"ip-10-0-131-228.us-east-2.compute.internal.k8s-audit.log"
"ip-10-0-131-228.us-east-2.compute.internal.linux-audit.log"
"ip-10-0-131-228.us-east-2.compute.internal.openshift-audit.log"
...
```

```
$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.infrastructure | jq
.logStreams[].logStreamName
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-69f9fd9b58-
zqzw5_openshift-oauth-apiserver_oauth-apiserver-
453c5c4ee026fe20a6139ba6b1cdd1bed25989c905bf5ac5ca211b7cbb5c3d7b.log"
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-797774f7c5-
lftrx_openshift-apiserver_openshift-apiserver-
ce51532df7d4e4d5f21c4f4be05f6575b93196336be0027067fd7d93d70f66a4.log"
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-797774f7c5-
lftrx_openshift-apiserver_openshift-apiserver-check-endpoints-
82a9096b5931b5c3b1d6dc4b66113252da4a6472c9fff48623baee761911a9ef.log"
...
```

Each log stream contains log events. To see a log event from the **busybox** Pod, you specify its log stream from the **application** log group:

```
$ aws logs get-log-events --log-group-name mycluster-7977k.application --log-stream-name
kubernetes.var.log.containers.busybox_app_busybox-
da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76.log
{
    "events": [
        {
            "timestamp": 1629422704178,
            "message": "{\"docker\":
{\"container_id\":\"da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76\"},\"kub
ernetes\":
{\"container_name\":\"busybox\",\"namespace_name\":\"app\",\"pod_name\":\"busybox\",\"container_ima
ge\":\"docker.io/library/busybox:latest\",\"container_image_id\":\"docker.io/library/busybox@sha256:0f35
4ec1728d9ff32edcd7d1b8bbdfc798277ad36120dc3dc683be44524c8b60\",\"pod_id\":\"870be234-
90a3-4258-b73f-4f4d6e2777c7\",\"host\":\"ip-10-0-216-3.us-east-2.compute.internal\",\"labels\":
{\"run\":\"busybox\"},\"master_url\":\"https://kubernetes.default.svc\",\"namespace_id\":\"794e1e1a-
b9f5-4958-a190-e76a9b53d7bf\",\"namespace_labels\":
```

```
{\"kubernetes_io/metadata_name\":\"app\"}},\"message\":\"My life is my
message\",\"level\":\"unknown\",\"hostname\":\"ip-10-0-216-3.us-east-
2.compute.internal\",\"pipeline_metadata\":{\"collector\":
{\"ipaddr4\":\"10.0.216.3\",\"inputname\":\"fluent-plugin-
systemd\",\"name\":\"fluentd\",\"received_at\":\"2021-08-
20T01:25:08.085760+00:00\",\"version\":\"1.7.4 1.6.0\"}},\"@timestamp\":\"2021-08-
20T01:25:04.178986+00:00\",\"viaq_index_name\":\"app-
write\",\"viaq_msg_id\":\"NWRjZmUyMWQtZjgzNC00MjI4LTk3MjMtNTk3NmY3ZjU4NDk1\",\"log_type\":
\"application\",\"time\":\"2021-08-20T01:25:04+00:00\"}",
            "ingestionTime": 1629422744016
        },
...
```

Example: Customizing the prefix in log group names

In the log group names, you can replace the default **infrastructureName** prefix, **mycluster-7977k**, with an arbitrary string like **demo-group-prefix**. To make this change, you update the **groupPrefix** field in the **ClusterLogForwarding** CR:

```
cloudwatch:
    groupBy: logType
    groupPrefix: demo-group-prefix
    region: us-east-2
```

The value of **groupPrefix** replaces the default **infrastructureName** prefix:

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"demo-group-prefix.application"
"demo-group-prefix.audit"
"demo-group-prefix.infrastructure"
```

Example: Naming log groups after application namespace names

For each application namespace in your cluster, you can create a log group in CloudWatch whose name is based on the name of the application namespace.

If you delete an application namespace object and create a new one that has the same name, CloudWatch continues using the same log group as before.

If you consider successive application namespace objects that have the same name as equivalent to each other, use the approach described in this example. Otherwise, if you need to distinguish the resulting log groups from each other, see the following "Naming log groups for application namespace UUIDs" section instead.

To create application log groups whose names are based on the names of the application namespaces, you set the value of the **groupBy** field to **namespaceName** in the **ClusterLogForwarder** CR:

```
cloudwatch:
    groupBy: namespaceName
    region: us-east-2
```

Setting **groupBy** to **namespaceName** affects the application log group only. It does not affect the **audit** and **infrastructure** log groups.

In Amazon Cloudwatch, the namespace name appears at the end of each log group name. Because there is a single application namespace, "app", the following output shows a new **mycluster-7977k.app** log group instead of **mycluster-7977k.application**:

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.app"
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"
```

If the cluster in this example had contained multiple application namespaces, the output would show multiple log groups, one for each namespace.

The **groupBy** field affects the application log group only. It does not affect the **audit** and **infrastructure** log groups.

Example: Naming log groups after application namespace UUIDs

For each application namespace in your cluster, you can create a log group in CloudWatch whose name is based on the UUID of the application namespace.

If you delete an application namespace object and create a new one, CloudWatch creates a new log group.

If you consider successive application namespace objects with the same name as different from each other, use the approach described in this example. Otherwise, see the preceding "Example: Naming log groups for application namespace names" section instead.

To name log groups after application namespace UUIDs, you set the value of the **groupBy** field to **namespaceUUID** in the **ClusterLogForwarder** CR:

```
cloudwatch:
   groupBy: namespaceUUID
   region: us-east-2
```

In Amazon Cloudwatch, the namespace UUID appears at the end of each log group name. Because there is a single application namespace, "app", the following output shows a new **mycluster-7977k.794e1e1a-b9f5-4958-a190-e76a9b53d7bf** log group instead of **mycluster-7977k.application**:

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.794e1e1a-b9f5-4958-a190-e76a9b53d7bf" // uid of the "app" namespace
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"
```

The **groupBy** field affects the application log group only. It does not affect the **audit** and **infrastructure** log groups.

7.8. FORWARDING LOGS TO GRAFANA LOKI

You can forward logs to an external Loki logging system in addition to, or instead of, the internal default OpenShift Container Platform Elasticsearch instance.

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

Prerequisites

- You must have a Loki logging system running at the URL you specify with the **url** field in the CR.

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: loki-insecure (3)
     type: "loki" (4)
     url: http://loki.insecure.com:9200 (5)
   - name: loki-secure
     type: "loki"
     url: https://loki.secure.com:9200 (6)
     secret:
        name: loki-secret (7)
  pipelines:
   - name: application-logs (8)
     inputRefs: (9)
     - application
     - audit
     outputRefs:
     - loki-secure (10)
     loki:
       tenantKey: kubernetes.namespace_name (11)
       labelKeys: kubernetes.labels.foo (12)
```

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

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

(3) Specify a name for the output.

(4) Specify the type as **"loki"**.

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

(6) For a secure connection, you can specify an **https** or **http** URL that you authenticate by specifying a **secret**.

(7) For an **https** prefix, specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project, and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent. Otherwise, for **http** and **https** prefixes, you can specify a secret that contains a username and password. For more information, see the following "Example: Setting secret that contains a username and password."

8 Optional: Specify a name for the pipeline.

9 Specify which log types to forward by using the pipeline: **application, infrastructure**, or **audit**.

10 Specify the name of the output to use when forwarding logs with this pipeline.

11 Optional: Specify a meta-data key field to generate values for the **TenantID** field in Loki. For example, setting **tenantKey: kubernetes.namespace_name** uses the names of the Kubernetes namespaces as values for tenant IDs in Loki. To see which other log record fields you can specify, see the "Log Record Fields" link in the following "Additional resources" section.

12 Optional: Specify a list of meta-data field keys to replace the default Loki labels. Loki label names must match the regular expression **[a-zA-Z_:][a-zA-Z0-9_:]***. Illegal characters in meta-data keys are replaced with _ to form the label name. For example, the **kubernetes.labels.foo** meta-data key becomes Loki label **kubernetes_labels_foo**. If you do not set **labelKeys**, the default value is: **[log_type, kubernetes.namespace_name, kubernetes.pod_name, kubernetes_host]**. Keep the set of labels small because Loki limits the size and number of labels allowed. See Configuring Loki, limits_config. You can still query based on any log record field using query filters.

NOTE

Because Loki requires log streams to be correctly ordered by timestamp, **labelKeys** always includes the **kubernetes_host** label set, even if you do not specify it. This inclusion ensures that each stream originates from a single host, which prevents timestamps from becoming disordered due to clock differences on different hosts.

2. Create the CR object:

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

Additional resources

- Log Record Fields.

7.9. FORWARDING APPLICATION LOGS FROM SPECIFIC PROJECTS

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

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

Prerequisites

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

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  outputs:
   - name: fluentd-server-secure (3)
     type: fluentdForward (4)
     url: 'tls://fluentdserver.security.example.com:24224' (5)
     secret: (6)
        name: fluentd-secret
   - name: fluentd-server-insecure
     type: fluentdForward
     url: 'tcp://fluentdserver.home.example.com:24224'
  inputs: (7)
   - name: my-app-logs
     application:
        namespaces:
        - my-project
  pipelines:
   - name: forward-to-fluentd-insecure (8)
     inputRefs: (9)
     - my-app-logs
     outputRefs: (10)
     - fluentd-server-insecure
     parse: json (11)
     labels:
       project: "my-project" (12)
   - name: forward-to-fluentd-secure (13)
     inputRefs:
     - application
     - audit
     - infrastructure
     outputRefs:
     - fluentd-server-secure
     - default
     labels:
       clusterId: "C1234"
```

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

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

(3) Specify a name for the output.

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

(5)

Specify the URL and port of the external log aggregator as a valid absolute URL. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name

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

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

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

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

10 The name of the output to use.

11 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.

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

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

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

2. Create the CR object:

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

7.10. FORWARDING APPLICATION LOGS FROM SPECIFIC PODS

As a cluster administrator, you can use Kubernetes pod labels to gather log data from specific pods and forward it to a log collector.

Suppose that you have an application composed of pods running alongside other pods in various namespaces. If those pods have labels that identify the application, you can gather and output their log data to a specific log collector.

To specify the pod labels, you use one or more **matchLabels** key-value pairs. If you specify multiple key-value pairs, the pods must match all of them to be selected.

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object. In the file, specify the pod labels using simple equality-based selectors under **inputs[].name.application.selector.matchLabels**, as shown in the following example.

Example ClusterLogForwarder CR YAML file

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  pipelines:
    - inputRefs: [ myAppLogData ] 3
      outputRefs: [ default ] 4
      parse: json 5
  inputs: 6
    - name: myAppLogData
      application:
        selector:
          matchLabels: 7
            environment: production
            app: nginx
        namespaces: 8
        - app1
        - app2
  outputs: 9
    - default
    ...
```

1. The name of the **ClusterLogForwarder** CR must be **instance**.
2. The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
3. Specify one or more comma-separated values from **inputs[].name**.
4. Specify one or more comma-separated values from **outputs[]**.
5. Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.
6. Define a unique **inputs[].name** for each application that has a unique set of pod labels.
7. Specify the key-value pairs of pod labels whose log data you want to gather. You must specify both a key and value, not just a key. To be selected, the pods must match all the key-value pairs.
8. Optional: Specify one or more namespaces.
9. Specify one or more outputs to forward your log data to. The optional **default** output shown here sends log data to the internal Elasticsearch instance.

2. Optional: To restrict the gathering of log data to specific namespaces, use **inputs[].name.application.namespaces**, as shown in the preceding example.

3. Optional: You can send log data from additional applications that have different pod labels to the same pipeline.

   a. For each unique combination of pod labels, create an additional **inputs[].name** section similar to the one shown.

   b. Update the **selectors** to match the pod labels of this application.

   c. Add the new **inputs[].name** value to **inputRefs**. For example:

   ```
   - inputRefs: [ myAppLogData, myOtherAppLogData ]
   ```

4. Create the CR object:

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

Additional resources

- For more information on **matchLabels** in Kubernetes, see Resources that support set-based requirements.

7.11. COLLECTING OVN NETWORK POLICY AUDIT LOGS

You can collect the OVN network policy audit logs from the **/var/log/ovn/acl-audit-log.log** file on OVN-Kubernetes pods and forward them to logging servers.

Prerequisites

- You are using OpenShift Container Platform version 4.8 or later.
- You are using Cluster Logging version 5.2 or later.
- You have already set up a **ClusterLogForwarder** custom resource (CR) object.
- The OpenShift Container Platform cluster is configured for OVN-Kubernetes network policy audit logging. See the following "Additional resources" section.

NOTE

Often, logging servers that store audit data must meet organizational and governmental requirements for compliance and security.

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object as described in other topics on forwarding logs to third-party systems.

2. In the YAML file, add the **audit** log type to the **inputRefs** element in a pipeline. For example:

   ```
   pipelines:
    - name: audit-logs
   ```

```
inputRefs:
 - audit 1
outputRefs:
 - secure-logging-server 2
```

1 Specify **audit** as one of the log types to input.

2 Specify the output that connects to your logging server.

3. Recreate the updated CR object:

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

Verification

Verify that audit log entries from the nodes that you are monitoring are present among the log data gathered by the logging server.

Find an original audit log entry in **/var/log/ovn/acl-audit-log.log** and compare it with the corresponding log entry on the logging server.

For example, an original log entry in **/var/log/ovn/acl-audit-log.log** might look like this:

```
2021-07-06T08:26:58.687Z|00004|acl_log(ovn_pinctrl0)|INFO|name="verify-audit-
logging_deny-all", verdict=drop, severity=alert:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:12,dl_dst=0a:58:0a:81:02:14,nw_src=10
.129.2.18,nw_dst=10.129.2.20,nw_tos=0,nw_ecn=0,nw_ttl=64,icmp_type=8,icmp_code=0
```

And the corresponding OVN audit log entry you find on the logging server might look like this:

```
{
  "@timestamp" : "2021-07-06T08:26:58..687000+00:00",
  "hostname":"ip.abc.iternal",
  "level":"info",
  "message" : "2021-07-06T08:26:58.687Z|00004|acl_log(ovn_pinctrl0)|INFO|name=\"verify-audit-
logging_deny-all\", verdict=drop, severity=alert:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:12,dl_dst=0a:58:0a:81:02:14,nw_src=10.129.2.18,nw_dst=
10.129.2.20,nw_tos=0,nw_ecn=0,nw_ttl=64,icmp_type=8,icmp_code=0"
}
```

Where:

- **@timestamp** is the timestamp of the log entry.
- **hostname** is the node from which the log originated.
- **level** is the log entry.
- **message** is the original audit log message.

NOTE

On an Elasticsearch server, look for log entries whose indices begin with **audit-00000**.

Troubleshooting

1. Verify that your OpenShift Container Platform cluster meets all the prerequisites.
2. Verify that you have completed the procedure.
3. Verify that the nodes generating OVN logs are enabled and have **/var/log/ovn/acl-audit-log.log** files.
4. Check the Fluentd pod logs for issues.

Additional resources

- Network policy audit logging

7.12. FORWARDING LOGS USING THE LEGACY FLUENTD METHOD

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

IMPORTANT

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

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

Prerequisites

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

Sample Fluentd configuration file

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

Procedure

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

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

```
<store>
  @type forward
  <security>
    self_hostname ${hostname}
    shared_key <key> (1)
  </security>
  transport tls (2)
  tls_verify_hostname <value> (3)
  tls_cert_path <path_to_file> (4)
  <buffer> (5)
    @type file
    path '/var/lib/fluentd/secureforwardlegacy'
    queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '1024' }"
    chunk_limit_size "#{ENV['BUFFER_SIZE_LIMIT'] || '1m' }"
    flush_interval "#{ENV['FORWARD_FLUSH_INTERVAL'] || '5s'}"
    flush_at_shutdown "#{ENV['FLUSH_AT_SHUTDOWN'] || 'false'}"
    flush_thread_count "#{ENV['FLUSH_THREAD_COUNT'] || 2}"
    retry_max_interval "#{ENV['FORWARD_RETRY_WAIT'] || '300'}"
    retry_forever true
  </buffer>
  <server>
    name (6)
    host (7)
    hostlabel (8)
    port (9)
  </server>
  <server> (10)
    name
    host
  </server>
```

(1) Enter the shared key between nodes.

(2) Specify **tls** to enable TLS validation.

(3) Set to **true** to verify the server cert hostname. Set to **false** to ignore server cert hostname.

4. Specify the path to the private CA certificate file as **/etc/ocp-forward/ca_cert.pem**.
5. Specify the Fluentd buffer parameters as needed.
6. Optionally, enter a name for this server.
7. Specify the hostname or IP of the server.
8. Specify the host label of the server.
9. Specify the port of the server.
10. Optionally, add additional servers. If you specify two or more servers, **forward** uses these server nodes in a round-robin order.

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

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

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

7.13. FORWARDING LOGS USING THE LEGACY SYSLOG METHOD

You can use the **syslog** RFC3164 protocol to send logs to destinations outside of your OpenShift Container Platform cluster by creating a configuration file and config map. You are responsible for configuring the external log aggregator, such as a syslog server, to receive the logs from OpenShift Container Platform.

IMPORTANT

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

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

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

To send logs using the **syslog** protocol, create a configuration file called **syslog.conf**, with the information needed to forward the logs. Then, use that file to create a config map called **syslog** in the **openshift-logging** project, which OpenShift Container Platform uses when forwarding the logs.

Prerequisites

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

Sample syslog configuration file

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

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

- facility: The syslog facility. The value can be a decimal integer or a case-insensitive keyword:
  - **0** or **kern** for kernel messages
  - **1** or **user** for user-level messages, the default.
  - **2** or **mail** for the mail system
  - **3** or **daemon** for the system daemons
  - **4** or **auth** for the security/authentication messages
  - **5** or **syslog** for messages generated internally by syslogd
  - **6** or **lpr** for the line printer subsystem
  - **7** or **news** for the network news subsystem
  - **8** or **uucp** for the UUCP subsystem
  - **9** or **cron** for the clock daemon
  - **10** or **authpriv** for security authentication messages
  - **11** or **ftp** for the FTP daemon
  - **12** or **ntp** for the NTP subsystem
  - **13** or **security** for the syslog audit logs
  - **14** or **console** for the syslog alert logs
  - **15** or **solaris-cron** for the scheduling daemon
  - **16**–**23** or **local0** – **local7** for locally used facilities
- payloadKey: The record field to use as payload for the syslog message.
- rfc: The RFC to be used for sending logs using syslog.
- severity: The syslog severity to set on outgoing syslog records. The value can be a decimal integer or a case-insensitive keyword:

- **0** or **Emergency** for messages indicating the system is unusable
- **1** or **Alert** for messages indicating action must be taken immediately
- **2** or **Critical** for messages indicating critical conditions
- **3** or **Error** for messages indicating error conditions
- **4** or **Warning** for messages indicating warning conditions
- **5** or **Notice** for messages indicating normal but significant conditions
- **6** or **Informational** for messages indicating informational messages
- **7** or **Debug** for messages indicating debug-level messages, the default

- tag: The record field to use as a tag on the syslog message.
- trimPrefix: The prefix to remove from the tag.

Procedure

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

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

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

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

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

(3) Specify the port of the syslog server.

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

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

- Parameter to use the severity and facility from the record if available. If **true**, the **container_name**, **namespace_name**, and **pod_name** are included in the output content.

- Parameter to specify the key to set the payload of the syslog message. Defaults to **message**.

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

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

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

7.14. TROUBLESHOOTING LOG FORWARDING

When you create a **ClusterLogForwarder** custom resource (CR), if the Red Hat OpenShift Logging Operator does not redeploy the Fluentd pods automatically, you can delete the Fluentd pods to force them to redeploy.

Prerequisites

- You have created a **ClusterLogForwarder** custom resource (CR) object.

Procedure

- Delete the Fluentd pods to force them to redeploy.

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

CHAPTER 8. ENABLING JSON LOGGING

You can configure the Log Forwarding API to parse JSON strings into a structured object.

8.1. PARSING JSON LOGS

Logs including JSON logs are usually represented as a string inside the **message** field. That makes it hard for users to query specific fields inside a JSON document. OpenShift Logging's Log Forwarding API enables you to parse JSON logs into a structured object and forward them to either OpenShift Logging-managed Elasticsearch or any other third-party system supported by the Log Forwarding API.

To illustrate how this works, suppose that you have the following structured JSON log entry.

Example structured JSON log entry

```
{"level":"info","name":"fred","home":"bedrock"}
```

Normally, the **ClusterLogForwarder** custom resource (CR) forwards that log entry in the **message** field. The **message** field contains the JSON-quoted string equivalent of the JSON log entry, as shown in the following example.

Example message field

```
{"message":"{\"level\":\"info\",\"name\":\"fred\",\"home\":\"bedrock\"",
 "more fields..."}
```

To enable parsing JSON log, you add **parse: json** to a pipeline in the **ClusterLogForwarder** CR, as shown in the following example.

Example snippet showing parse: json

```
pipelines:
- inputRefs: [ application ]
  outputRefs: myFluentd
  parse: json
```

When you enable parsing JSON logs by using **parse: json**, the CR copies the JSON-structured log entry in a **structured** field, as shown in the following example. This does not modify the original **message** field.

Example structured output containing the structured JSON log entry

```
{"structured": { "level": "info", "name": "fred", "home": "bedrock" },
 "more fields..."}
```

IMPORTANT

If the log entry does not contain valid structured JSON, the **structured** field will be absent.

To enable parsing JSON logs for specific logging platforms, see Forwarding logs to third-party systems .

8.2. CONFIGURING JSON LOG DATA FOR ELASTICSEARCH

If your JSON logs follow more than one schema, storing them in a single index might cause type conflicts and cardinality problems. To avoid that, you must configure the **ClusterLogForwarder** custom resource (CR) to group each schema into a single output definition. This way, each schema is forwarded to a separate index.

IMPORTANT

If you forward JSON logs to the default Elasticsearch instance managed by OpenShift Logging, it generates new indices based on your configuration. To avoid performance issues associated with having too many indices, consider keeping the number of possible schemas low by standardizing to common schemas.

Structure types

You can use the following structure types in the **ClusterLogForwarder** CR to construct index names for the Elasticsearch log store:

- **structuredTypeKey** (string, optional) is the name of a message field. The value of that field, if present, is used to construct the index name.
  - **kubernetes.labels.<key>** is the Kubernetes Pod label whose value is used to construct the index name.
  - **openshift.labels.<key>** is the **pipeline.label.<key>** element in the **ClusterLogForwarder** CR whose value is used to construct the index name.
  - **kubernetes.container_name** uses the container name to construct the index name.
- **structuredTypeName**: (string, optional) If **structuredTypeKey** is not set or its key is not present, OpenShift Logging uses the value of **structuredTypeName** as the structured type. When you use both **structuredTypeKey** and **structuredTypeName** together, **structuredTypeName** provides a fallback index name if the key in **structuredTypeKey** is missing from the JSON log data.

NOTE

Although you can set the value of **structuredTypeKey** to any field shown in the "Log Record Fields" topic, the most useful fields are shown in the preceding list of structure types.

A structuredTypeKey: kubernetes.labels.<key> example

Suppose the following:

- Your cluster is running application pods that produce JSON logs in two different formats, "apache" and "google".
- The user labels these application pods with **logFormat=apache** and **logFormat=google**.
- You use the following snippet in your **ClusterLogForwarder** CR YAML file.

```
outputDefaults:
- elasticsearch:
    structuredTypeKey: kubernetes.labels.logFormat (1)
    structuredTypeName: nologformat
pipelines:
- inputRefs: <application>
  outputRefs: default
  parse: json (2)
```

(1) Uses the value of the key-value pair that is formed by the Kubernetes **logFormat** label.

(2) Enables parsing JSON logs.

In that case, the following structured log record goes to the **app-apache-write** index:

```
{
  "structured":{"name":"fred","home":"bedrock"},
  "kubernetes":{"labels":{"logFormat": "apache", ...}}
}
```

And the following structured log record goes to the **app-google-write** index:

```
{
  "structured":{"name":"wilma","home":"bedrock"},
  "kubernetes":{"labels":{"logFormat": "google", ...}}
}
```

A structuredTypeKey: openshift.labels.<key> example

Suppose that you use the following snippet in your **ClusterLogForwarder** CR YAML file.

```
outputDefaults:
- elasticsearch:
    structuredTypeKey: openshift.labels.myLabel (1)
    structuredTypeName: nologformat
pipelines:
 - name: application-logs
   inputRefs:
   - application
   - audit
   outputRefs:
   - elasticsearch-secure
   - default
   parse: json
   labels:
     myLabel: myValue (2)
```

(1) Uses the value of the key-value pair that is formed by the OpenShift **myLabel** label.

(2) The **myLabel** element gives its string value, **myValue**, to the structured log record.

In that case, the following structured log record goes to the **app-myValue-write** index:

```
{
  "structured":{"name":"fred","home":"bedrock"},
  "openshift":{"labels":{"myLabel": "myValue", ...}}
}
```

Additional considerations

- The Elasticsearch *index* for structured records is formed by prepending "app-" to the structured type and appending "-write".
- Unstructured records are not sent to the structured index. They are indexed as usual in the application, infrastructure, or audit indices.
- If there is no non-empty structured type, forward an *unstructured* record with no **structured** field.

It is important not to overload Elasticsearch with too many indices. Only use distinct structured types for distinct log *formats*, **not** for each application or namespace. For example, most Apache applications use the same JSON log format and structured type, such as **LogApache**.

8.3. FORWARDING JSON LOGS TO THE ELASTICSEARCH LOG STORE

For an Elasticsearch log store, if your JSON log entries *follow different schemas*, configure the **ClusterLogForwarder** custom resource (CR) to group each JSON schema into a single output definition. This way, Elasticsearch uses a separate index for each schema.

IMPORTANT

Because forwarding different schemas to the same index can cause type conflicts and cardinality problems, you must perform this configuration before you forward data to the Elasticsearch store.

To avoid performance issues associated with having too many indices, consider keeping the number of possible schemas low by standardizing to common schemas.

Procedure

1. Add the following snippet to your **ClusterLogForwarder** CR YAML file.

```
outputDefaults:
- elasticsearch:
    structuredTypeKey: <log record field>
    structuredTypeName: <name>
pipelines:
- inputRefs:
  - application
  outputRefs: default
  parse: json
```

2. Optional: Use **structuredTypeKey** to specify one of the log record fields, as described in the preceding topic, Configuring JSON log data for Elasticsearch. Otherwise, remove this line.

3. Optional: Use **structuredTypeName** to specify a **<name>**, as described in the preceding topic, Configuring JSON log data for Elasticsearch. Otherwise, remove this line.

IMPORTANT

To parse JSON logs, you must set either **structuredTypeKey** or **structuredTypeName**, or both **structuredTypeKey** and **structuredTypeName**.

4. For **inputRefs**, specify which log types to forward by using that pipeline, such as **application, infrastructure**, or **audit**.

5. Add the **parse: json** element to pipelines.

6. Create the CR object:

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

The Red Hat OpenShift Logging Operator redeploys the Fluentd pods. However, if they do not redeploy, delete the Fluentd pods to force them to redeploy.

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

Additional resources

- Forwarding logs to third-party systems

CHAPTER 9. COLLECTING AND STORING KUBERNETES EVENTS

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

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

IMPORTANT

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

9.1. DEPLOYING AND CONFIGURING THE EVENT ROUTER

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

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

Prerequisites

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

Procedure

1. Create a template for the Event Router:

```
kind: Template
apiVersion: v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to OpenShift Logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
  - kind: ServiceAccount (1)
    apiVersion: v1
    metadata:
      name: eventrouter
      namespace: ${NAMESPACE}
  - kind: ClusterRole (2)
    apiVersion: v1
    metadata:
      name: event-reader
    rules:
```

```
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "watch", "list"]
- kind: ClusterRoleBinding  3
  apiVersion: v1
  metadata:
    name: event-reader-binding
  subjects:
  - kind: ServiceAccount
    name: eventrouter
    namespace: ${NAMESPACE}
  roleRef:
    kind: ClusterRole
    name: event-reader
- kind: ConfigMap  4
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  data:
    config.json: |-
      {
        "sink": "stdout"
      }
- kind: Deployment  5
  apiVersion: apps/v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
    labels:
      component: "eventrouter"
      logging-infra: "eventrouter"
      provider: "openshift"
  spec:
    selector:
      matchLabels:
        component: "eventrouter"
        logging-infra: "eventrouter"
        provider: "openshift"
    replicas: 1
    template:
      metadata:
        labels:
          component: "eventrouter"
          logging-infra: "eventrouter"
          provider: "openshift"
        name: eventrouter
      spec:
        serviceAccount: eventrouter
        containers:
          - name: kube-eventrouter
            image: ${IMAGE}
            imagePullPolicy: IfNotPresent
            resources:
              requests:
                cpu: ${CPU}
                memory: ${MEMORY}
              volumeMounts:
              - name: config-volume
                mountPath: /etc/eventrouter
          volumes:
            - name: config-volume
              configMap:
                name: eventrouter
parameters:
  - name: IMAGE
    displayName: Image
    value: "registry.redhat.io/openshift-logging/eventrouter-rhel8:latest"
  - name: CPU  6
    displayName: CPU
    value: "100m"
  - name: MEMORY  7
    displayName: Memory
    value: "128Mi"
  - name: NAMESPACE
    displayName: Namespace
    value: "openshift-logging"  8
```

1 Creates a Service Account in the **openshift-logging** project for the Event Router.

2 Creates a ClusterRole to monitor for events in the cluster.

3 Creates a ClusterRoleBinding to bind the ClusterRole to the service account.

4 Creates a config map in the **openshift-logging** project to generate the required **config.json** file.

5 Creates a deployment in the **openshift-logging** project to generate and configure the Event Router pod.

6 Specifies the minimum amount of memory to allocate to the Event Router pod. Defaults to **128Mi**.

7 Specifies the minimum amount of CPU to allocate to the Event Router pod. Defaults to **100m**.

8 Specifies the **openshift-logging** project to install objects in.

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

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

For example:

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

Example output

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

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

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

   a. View the new Event Router pod:

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

   Example output

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

   b. View the events collected by the Event Router:

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

   For example:

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

   Example output

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

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

CHAPTER 10. UPDATING OPENSHIFT LOGGING

OpenShift Container Platform versions 4.7 and 4.8 support OpenShift Logging versions 5.0, 5.1, and 5.2.

To upgrade from cluster logging in OpenShift Container Platform version 4.6 and earlier to OpenShift Logging 5.x, you update the OpenShift Container Platform cluster to version 4.7 or 4.8. Then, you update the following operators:

- From Elasticsearch Operator 4.x to OpenShift Elasticsearch Operator 5.x
- From Cluster Logging Operator 4.x to Red Hat OpenShift Logging Operator 5.x

To upgrade from a previous version of OpenShift Logging to the current version, you update OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator to their current versions.

10.1. UPDATING FROM CLUSTER LOGGING IN OPENSHIFT CONTAINER PLATFORM 4.6 OR EARLIER TO OPENSHIFT LOGGING 5.X

OpenShift Container Platform 4.7 made the following name changes:

- The *cluster logging* feature became the *Red Hat OpenShift Logging* 5.x product.
- The *Cluster Logging* Operator became the *Red Hat OpenShift Logging* Operator.
- The *Elasticsearch* Operator became *OpenShift Elasticsearch* Operator.

To upgrade from cluster logging in OpenShift Container Platform version 4.6 and earlier to OpenShift Logging 5.x, you update the OpenShift Container Platform cluster to version 4.7 or 4.8. Then, you update the following operators:

- From Elasticsearch Operator 4.x to OpenShift Elasticsearch Operator 5.x
- From Cluster Logging Operator 4.x to Red Hat OpenShift Logging Operator 5.x

IMPORTANT

You must update the OpenShift Elasticsearch Operator *before* you update the Red Hat OpenShift Logging Operator. You must also update *both* Operators to the same version.

If you update the operators in the wrong order, Kibana does not update and the Kibana custom resource (CR) is not created. To work around this problem, you delete the Red Hat OpenShift Logging Operator pod. When the Red Hat OpenShift Logging Operator pod redeploys, it creates the Kibana CR and Kibana becomes available again.

Prerequisites

- The OpenShift Container Platform version is 4.7 or later.
- The OpenShift Logging status is healthy:
  - All pods are **ready**.
  - The Elasticsearch cluster is healthy.

- Your Elasticsearch and Kibana data is backed up.

Procedure

1. Update the OpenShift Elasticsearch Operator:
   a. From the web console, click **Operators** → **Installed Operators**.
   b. Select the **openshift-operators-redhat** project.
   c. Click the **OpenShift Elasticsearch Operator**.
   d. Click **Subscription** → **Channel**.
   e. In the **Change Subscription Update Channel** window, select **5.0** or **stable-5.x** and click **Save**.
   f. Wait for a few seconds, then click **Operators** → **Installed Operators**.
      Verify that the OpenShift Elasticsearch Operator version is 5.x.x.

      Wait for the **Status** field to report **Succeeded**.
2. Update the Cluster Logging Operator:
   a. From the web console, click **Operators** → **Installed Operators**.
   b. Select the **openshift-logging** project.
   c. Click the **Cluster Logging Operator**.
   d. Click **Subscription** → **Channel**.
   e. In the **Change Subscription Update Channel** window, select **5.0** or **stable-5.x** and click **Save**.
   f. Wait for a few seconds, then click **Operators** → **Installed Operators**.
      Verify that the Red Hat OpenShift Logging Operator version is 5.0.x or 5.x.x.

      Wait for the **Status** field to report **Succeeded**.
3. Check the logging components:
   a. Ensure that all Elasticsearch pods are in the **Ready** status:

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

Example output

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

   b. Ensure that the Elasticsearch cluster is healthy:

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

```
{
  "cluster_name" : "elasticsearch",
  "status" : "green",
}
```

c. Ensure that the Elasticsearch cron jobs are created:

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

```
$ oc get cronjob
```

```
NAME                     SCHEDULE       SUSPEND   ACTIVE   LAST SCHEDULE   AGE
elasticsearch-im-app     */15 * * * *   False     0        <none>          56s
elasticsearch-im-audit   */15 * * * *   False     0        <none>          56s
elasticsearch-im-infra   */15 * * * *   False     0        <none>          56s
```

d. Verify that the log store is updated to 5.0 or 5.x and the indices are **green**:

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

Verify that the output includes the **app-00000x**, **infra-00000x**, **audit-00000x**, **.security** indices.

Example 10.1. Sample output with indices in a green status

```
Tue Jun 30 14:30:54 UTC 2020
health status index                                                                uuid                 pri rep
docs.count docs.deleted store.size pri.store.size
green  open   infra-000008
bnBvUFEXTWi92z3zWAzieQ   3 1       222195           0       289           144
green  open   infra-000004
rtDSzoqsSl6saisSK7Au1Q   3 1       226717           0       297           148
green  open   infra-000012
RSf_kUwDSR2xEuKRZMPqZQ   3 1       227623           0       295           147
green  open   .kibana_7
1SJdCqlZTPWlIAaOUd78yg   1 1           4           0        0            0
green  open   infra-000010
iXwL3bnqTuGEABbUDa6OVw   3 1       248368           0       317           158
green  open   infra-000009
YN9EsULWSNaxWeeNvOs0RA   3 1       258799           0       337           168
green  open   infra-000014
YP0U6R7FQ_GVQVQZ6Yh9Ig   3 1       223788           0       292           146
green  open   infra-000015
JRBbAbEmSMqK5X40df9HbQ   3 1       224371           0       291           145
green  open   .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA   3 1           9           0        0            0
green  open   infra-000007
llkkAVSzSOmosWTSAJM_hg   3 1       228584           0       296           148
green  open   infra-000005
d9BoGQdiQASsS3BBFm2iRA   3 1       227987           0       297           148
```

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

e. Verify that the log collector is updated to 5.0 or 5.x:

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

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

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

f. Verify that the log visualizer is updated to 5.0 or 5.x using the Kibana CRD:

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

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

Example 10.2. Sample output with a ready Kibana pod

```
[
{
"clusterCondition": {
"kibana-5fdd766ffd-nb2jj": [
{
"lastTransitionTime": "2020-06-30T14:11:07Z",
"reason": "ContainerCreating",
"status": "True",
"type": ""
},
{
"lastTransitionTime": "2020-06-30T14:11:07Z",
"reason": "ContainerCreating",
"status": "True",
"type": ""
}
]
},
"deployment": "kibana",
"pods": {
"failed": [],
"notReady": []
"ready": []
},
"replicaSets": [
"kibana-5fdd766ffd"
],
"replicas": 1
}
]
```

10.2. UPDATING OPENSHIFT LOGGING TO THE CURRENT VERSION

To update OpenShift Logging from 5.x to the current version, you change the subscriptions for the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator.

IMPORTANT

You must update the OpenShift Elasticsearch Operator *before* you update the Red Hat OpenShift Logging Operator. You must also update *both* Operators to the same version.

If you update the operators in the wrong order, Kibana does not update and the Kibana custom resource (CR) is not created. To work around this problem, you delete the Red Hat OpenShift Logging Operator pod. When the Red Hat OpenShift Logging Operator pod redeploys, it creates the Kibana CR and Kibana becomes available again.

Prerequisites

- The OpenShift Container Platform version is 4.7 or later.
- The OpenShift Logging status is healthy:
  - All pods are **ready**.
  - The Elasticsearch cluster is healthy.
- Your Elasticsearch and Kibana data is backed up.

Procedure

1. Update the OpenShift Elasticsearch Operator:
   a. From the web console, click **Operators → Installed Operators**.
   b. Select the **openshift-operators-redhat** project.

c. Click the **OpenShift Elasticsearch Operator**.

d. Click **Subscription** → **Channel**.

e. In the **Change Subscription Update Channel** window, select **stable-5.x** and click **Save**.

f. Wait for a few seconds, then click **Operators** → **Installed Operators**.
Verify that the OpenShift Elasticsearch Operator version is 5.x.x.

Wait for the **Status** field to report **Succeeded**.

2. Update the Red Hat OpenShift Logging Operator:

a. From the web console, click **Operators** → **Installed Operators**.

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

c. Click the **Red Hat OpenShift Logging Operator**.

d. Click **Subscription** → **Channel**.

e. In the **Change Subscription Update Channel** window, select **stable-5.x** and click **Save**.

f. Wait for a few seconds, then click **Operators** → **Installed Operators**.
Verify that the Red Hat OpenShift Logging Operator version is 5.x.x.

Wait for the **Status** field to report **Succeeded**.

3. Check the logging components:

a. Ensure that all Elasticsearch pods are in the **Ready** status:

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

Example output

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

b. Ensure that the Elasticsearch cluster is healthy:

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

```
{
  "cluster_name" : "elasticsearch",
  "status" : "green",
}
```

c. Ensure that the Elasticsearch cron jobs are created:

```
$ oc project openshift-logging
$ oc get cronjob
```

```
NAME                     SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
elasticsearch-im-app     */15 * * * *   False     0        <none>          56s
elasticsearch-im-audit   */15 * * * *   False     0        <none>          56s
elasticsearch-im-infra   */15 * * * *   False     0        <none>          56s
```

d. Verify that the log store is updated to 5.x and the indices are **green**:

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

Verify that the output includes the **app-00000x**, **infra-00000x**, **audit-00000x**, **.security** indices.

Example 10.3. Sample output with indices in a green status

```
Tue Jun 30 14:30:54 UTC 2020
health status index                                                      uuid                pri rep
docs.count docs.deleted store.size pri.store.size
green  open   infra-000008
bnBvUFEXTWi92z3zWAzieQ   3 1       222195           0       289           144
green  open   infra-000004
rtDSzoqsSl6saisSK7Au1Q   3 1       226717           0       297           148
green  open   infra-000012
RSf_kUwDSR2xEuKRZMPqZQ   3 1       227623           0       295           147
green  open   .kibana_7
1SJdCqlZTPWlIAaOUd78yg   1 1           4           0         0             0
green  open   infra-000010
iXwL3bnqTuGEABbUDa6OVw   3 1       248368           0       317           158
green  open   infra-000009
YN9EsULWSNaxWeeNvOs0RA   3 1       258799           0       337           168
green  open   infra-000014
YP0U6R7FQ_GVQVQZ6Yh9Ig   3 1       223788           0       292           146
green  open   infra-000015
JRBbAbEmSMqK5X40df9HbQ   3 1       224371           0       291           145
green  open   .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA   3 1           9           0         0             0
green  open   infra-000007
llkkAVSzSOmosWTSAJM_hg   3 1       228584           0       296           148
green  open   infra-000005
d9BoGQdiQASsS3BBFm2iRA   3 1       227987           0       297           148
green  open   infra-000003                                                 1-
goREK1QUKlQPAIVkWVaQ   3 1       226719           0       295           147
green  open   .security
zeT65uOuRTKZMjg_bbUc1g   1 1           5           0         0             0
green  open   .kibana-377444158_kubeadmin                                        wvMhDwJkR-
mRZQO84K0gUQ   3 1           1           0         0             0
green  open   infra-000006                                                 5H-
KBSXGQKiO7hdapDE23g   3 1       226676           0       295           147
green  open   infra-000001                                                 eH53BQ-
bSxSWR5xYZB6lVg   3 1       341800           0       443           220
green  open   .kibana-6
RVp7TemSSemGJcsSUmuf3A   1 1           4           0         0             0
green  open   infra-000011
```

```
J7XWBauWSTe0jnzX02fU6A   3 1      226100          0       293           146
green  open   app-000001
axSAFfONQDmKwatkjPXdtw   3 1      103186          0       126           57
green  open   infra-000016
m9c1iRLtStWSF1GopaRyCg   3 1      13685           0       19            9
green  open   infra-000002                                              Hz6WvINtTvKcQzw-
ewmbYg   3 1      228994          0       296           148
green  open   infra-000013                                              KR9mMFUpQl-
jraYtanyIGw   3 1      228166          0       298           148
green  open   audit-000001
eERqLdLmQOiQDFES1LBATQ   3 1          0          0        0            0
```

e. Verify that the log collector is updated to 5.x:

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

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

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

f. Verify that the log visualizer is updated to 5.x using the Kibana CRD:

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

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

Example 10.4. Sample output with a ready Kibana pod

```
[
{
"clusterCondition": {
"kibana-5fdd766ffd-nb2jj": [
{
"lastTransitionTime": "2020-06-30T14:11:07Z",
"reason": "ContainerCreating",
"status": "True",
"type": ""
},
{
"lastTransitionTime": "2020-06-30T14:11:07Z",
"reason": "ContainerCreating",
"status": "True",
"type": ""
}
]
},
"deployment": "kibana",
"pods": {
"failed": [],
"notReady": []
"ready": []
},
"replicaSets": [
"kibana-5fdd766ffd"
],
"replicas": 1
}
]
```

CHAPTER 11. VIEWING CLUSTER DASHBOARDS

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

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

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

NOTE

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

11.1. ACCESSING THE ELASTISEARCH AND OPENSHIFT LOGGING DASHBOARDS

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

Procedure

To launch the dashboards:

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

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

NOTE

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

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

11.2. ABOUT THE OPENSHIFT LOGGING DASHBOARD

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

Table 11.1. OpenShift Logging charts

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

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

11.3. CHARTS ON THE LOGGING/ELASTICSEARCH NODES DASHBOARD

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

Elasticsearch status

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

Table 11.2. Elasticsearch status fields

Metric	Description

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

Elasticsearch cluster index shard status

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

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

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

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

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

Table 11.3. Elasticsearch cluster shard status charts

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

Elasticsearch node metrics

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

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

Table 11.4. Elasticsearch node metric charts

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

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

Elasticsearch node fielddata

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

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

Table 11.5. Elasticsearch node fielddata charts

Metric	Description

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

Elasticsearch node query cache

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

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

Table 11.6. Elasticsearch node query charts

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

Elasticsearch index throttling

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

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

Table 11.7. Index throttling charts

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

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

Node JVM Heap statistics

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

Table 11.8. JVM Heap statistic charts

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

CHAPTER 12. TROUBLESHOOTING LOGGING

12.1. VIEWING OPENSHIFT LOGGING STATUS

You can view the status of the Red Hat OpenShift Logging Operator and for a number of OpenShift Logging components.

12.1.1. Viewing the status of the Red Hat OpenShift Logging Operator

You can view the status of your Red Hat OpenShift Logging Operator.

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

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

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

2. To view the OpenShift Logging status:

   a. Get the OpenShift Logging status:

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

Example output

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

....

status: 1
  collection:
    logs:
      fluentdStatus:
        daemonSet: fluentd 2
        nodes:
          fluentd-2rhqp: ip-10-0-169-13.ec2.internal
          fluentd-6fgjh: ip-10-0-165-244.ec2.internal
          fluentd-6l2ff: ip-10-0-128-218.ec2.internal
          fluentd-54nx5: ip-10-0-139-30.ec2.internal
          fluentd-flpnn: ip-10-0-147-228.ec2.internal
          fluentd-n2frh: ip-10-0-157-45.ec2.internal
        pods:
          failed: []
          notReady: []
          ready:
          - fluentd-2rhqp
          - fluentd-54nx5
          - fluentd-6fgjh
          - fluentd-6l2ff
          - fluentd-flpnn
          - fluentd-n2frh
  logstore: 3
    elasticsearchStatus:
    - ShardAllocationEnabled:  all
      cluster:
        activePrimaryShards:    5
        activeShards:           5
        initializingShards:     0
        numDataNodes:           1
        numNodes:               1
        pendingTasks:           0
        relocatingShards:       0
        status:                 green
        unassignedShards:       0
      clusterName:              elasticsearch
      nodeConditions:
        elasticsearch-cdm-mkkdys93-1:
      nodeCount:  1
      pods:
        client:
          failed:
          notReady:
          ready:
          - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
        data:
          failed:
          notReady:
          ready:
          - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
        master:
          failed:
          notReady:
          ready:
          - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
visualization:  4
    kibanaStatus:
    - deployment: kibana
      pods:
        failed: []
        notReady: []
        ready:
        - kibana-7fb4fd4cc9-f2nls
      replicaSets:
      - kibana-7fb4fd4cc9
      replicas: 1
```

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

2 Information on the Fluentd pods.

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

4 Information on the Kibana pods.

12.1.1.1. Example condition messages

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

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

Example output

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

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

Example output

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

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

Example output

```
  Elasticsearch Status:
    Shard Allocation Enabled:  shard allocation unknown
    Cluster:
      Active Primary Shards:  0
      Active Shards:          0
      Initializing Shards:    0
      Num Data Nodes:         0
      Num Nodes:              0
      Pending Tasks:          0
      Relocating Shards:      0
      Status:                 cluster health unknown
      Unassigned Shards:      0
    Cluster Name:             elasticsearch
    Node Conditions:
      elasticsearch-cdm-mkkdys93-1:
        Last Transition Time:  2019-06-26T03:37:32Z
        Message:               0/5 nodes are available: 5 node(s) didn't match node selector.
        Reason:                Unschedulable
        Status:                True
        Type:                  Unschedulable
      elasticsearch-cdm-mkkdys93-2:
    Node Count:  2
    Pods:
      Client:
        Failed:
        Not Ready:
          elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
      Data:
        Failed:
        Not Ready:
          elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
      Master:
        Failed:
        Not Ready:
          elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
          elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
        Ready:
```

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

Example output

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

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

Example output

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

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

12.1.2. Viewing the status of OpenShift Logging components

You can view the status for a number of OpenShift Logging components.

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

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

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

2. View the status of the OpenShift Logging environment:

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

   Example output

   ```
   Name:                   cluster-logging-operator

   ....

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

   ....

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

3. View the status of the OpenShift Logging replica set:

   a. Get the name of a replica set:

      Example output

      ```
      $ oc get replicaset
      ```

      Example output

      ```
      NAME                                DESIRED   CURRENT   READY   AGE
      ```

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

b. Get the status of the replica set:

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

Example output

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

....

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

....

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

12.2. VIEWING THE STATUS OF THE LOG STORE

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

12.2.1. Viewing the status of the log store

You can view the status of your log store.

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

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

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

2. To view the status:

a. Get the name of the log store instance:

```
$ oc get Elasticsearch
```

Example output

```
NAME           AGE
elasticsearch   5h9m
```

b. Get the log store status:

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

For example:

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

The output includes information similar to the following:

Example output

```
status: 1
  cluster: 2
    activePrimaryShards: 30
    activeShards: 60
    initializingShards: 0
    numDataNodes: 3
    numNodes: 3
    pendingTasks: 0
    relocatingShards: 0
    status: green
    unassignedShards: 0
  clusterHealth: ""
  conditions: [] 3
  nodes: 4
  - deploymentName: elasticsearch-cdm-zjf34ved-1
    upgradeStatus: {}
  - deploymentName: elasticsearch-cdm-zjf34ved-2
    upgradeStatus: {}
  - deploymentName: elasticsearch-cdm-zjf34ved-3
    upgradeStatus: {}
  pods: 5
    client:
      failed: []
      notReady: []
      ready:
      - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
      - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
      - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
    data:
      failed: []
      notReady: []
      ready:
      - elasticsearch-cdm-zjf34ved-1-6d7fbf844f-sn422
      - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
      - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
    master:
      failed: []
      notReady: []
      ready:
```

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

1. In the output, the cluster status fields appear in the **status** stanza.
2. The status of the log store:
    - The number of active primary shards.
    - The number of active shards.
    - The number of shards that are initializing.
    - The number of log store data nodes.
    - The total number of log store nodes.
    - The number of pending tasks.
    - The log store status: **green**, **red**, **yellow**.
    - The number of unassigned shards.
3. Any status conditions, if present. The log store status indicates the reasons from the scheduler if a pod could not be placed. Any events related to the following conditions are shown:
    - Container Waiting for both the log store and proxy containers.
    - Container Terminated for both the log store and proxy containers.
    - Pod unschedulable. Also, a condition is shown for a number of issues; see **Example condition messages**.
4. The log store nodes in the cluster, with **upgradeStatus**.
5. The log store client, data, and master pods in the cluster, listed under 'failed`, **notReady**, or **ready** state.

12.2.1.1. Example condition messages

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

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

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

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

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

The following status message indicates that the log store node selector in the CR does not match any nodes in the cluster:

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

The following status message indicates that the log store CR uses a non-existent persistent volume claim (PVC).

```
status:
   nodes:
   - conditions:
     - last Transition Time:  2019-04-10T05:55:51Z
       message:               pod has unbound immediate PersistentVolumeClaims (repeated 5 times)
       reason:                Unschedulable
       status:                True
       type:                  Unschedulable
```

The following status message indicates that your log store cluster does not have enough nodes to support the redundancy policy.

```
status:
  clusterHealth: ""
  conditions:
  - lastTransitionTime: 2019-04-17T20:01:31Z
    message: Wrong RedundancyPolicy selected. Choose different RedundancyPolicy or
      add more nodes with data roles
    reason: Invalid Settings
    status: "True"
    type: InvalidRedundancy
```

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

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

The following status message indicates that Elasticsearch storage does not support the change you tried to make.

For example:

```
status:
  clusterHealth: green
  conditions:
    - lastTransitionTime: "2021-05-07T01:05:13Z"
      message: Changing the storage structure for a custom resource is not supported
      reason: StorageStructureChangeIgnored
      status: 'True'
      type: StorageStructureChangeIgnored
```

The **reason** and **type** fields specify the type of unsupported change:

StorageClassNameChangeIgnored
: Unsupported change to the storage class name.

StorageSizeChangeIgnored
: Unsupported change the storage size.

StorageStructureChangeIgnored
: Unsupported change between ephemeral and persistent storage structures.

IMPORTANT

If you try to configure the **ClusterLogging** custom resource (CR) to switch from ephemeral to persistent storage, the OpenShift Elasticsearch Operator creates a persistent volume claim (PVC) but does not create a persistent volume (PV). To clear the **StorageStructureChangeIgnored** status, you must revert the change to the **ClusterLogging** CR and delete the PVC.

12.2.2. Viewing the status of the log store components

You can view the status for a number of the log store components.

Elasticsearch indices

You can view the status of the Elasticsearch indices.

1. Get the name of an Elasticsearch pod:

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

Example output

```
pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. Get the status of the indices:

```
$ oc exec elasticsearch-cdm-4vjor49p-2-6d4d7db474-q2w7z -- indices
```

Example output

```
Defaulting container name to elasticsearch.
Use 'oc describe pod/elasticsearch-cdm-4vjor49p-2-6d4d7db474-q2w7z -n openshift-
logging' to see all of the containers in this pod.

green  open   infra-000002                                            S4QANnf1QP6NgCegfnrnbQ
3   1     119926           0       157            78
green  open   audit-000001                                            8_EQx77iQCSTzFOXtxRqFw
3   1        0           0         0             0
green  open   .security                                           iDjscH7aSUGhIdq0LheLBQ   1
1         5           0         0             0
green  open   .kibana_-377444158_kubeadmin
yBywZ9GfSrKebz5gWBZbjw   3   1         1           0         0             0
green  open   infra-000001                                            z6Dpe__ORgiopEpW6Yl44A
3   1     871000           0       874           436
green  open   app-000001                                              hIrazQCeSISewG3c2VIvsQ
3   1      2453           0         3             1
green  open   .kibana_1                                              JCitcBMSQxKOvIq6iQW6wg
1   1        0           0         0             0
green  open   .kibana_-1595131456_user1                                     gIYFIEGRRe-
ka0W3okS-mQ   3   1         1           0         0             0
```

Log store pods

You can view the status of the pods that host the log store.

1. Get the name of a pod:

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

Example output

```
pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. Get the status of a pod:

```
$ oc describe pod elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
```

The output includes the following status information:

Example output

```
....
Status:             Running

....

Containers:
  elasticsearch:
    Container ID:   cri-o://b7d44e0a9ea486e27f47763f5bb4c39dfd2
    State:          Running
      Started:      Mon, 08 Jun 2020 10:17:56 -0400
    Ready:          True
    Restart Count:  0
    Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
period=5s #success=1 #failure=3

....

  proxy:
    Container ID:  cri-
o://3f77032abaddbb1652c116278652908dc01860320b8a4e741d06894b2f8f9aa1
    State:          Running
      Started:      Mon, 08 Jun 2020 10:18:38 -0400
    Ready:          True
    Restart Count:  0

....

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

....

Events:          <none>
```

Log storage pod deployment configuration

You can view the status of the log store deployment configuration.

1. Get the name of a deployment configuration:

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

Example output

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

2. Get the deployment configuration status:

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

The output includes the following status information:

Example output

```
....
  Containers:
   elasticsearch:
    Image:      registry.redhat.io/openshift-logging/elasticsearch6-rhel8
    Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
period=5s #success=1 #failure=3

....

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

....

Events:          <none>
```

Log store replica set

You can view the status of the log store replica set.

1. Get the name of a replica set:

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

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

2. Get the status of the replica set:

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

The output includes the following status information:

Example output

```
....
  Containers:
   elasticsearch:
    Image:      registry.redhat.io/openshift-logging/elasticsearch6-
rhel8@sha256:4265742c7cdd85359140e2d7d703e4311b6497eec7676957f455d6908e7b1
c25
    Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
period=5s #success=1 #failure=3

....

Events:          <none>
```

12.3. UNDERSTANDING OPENSHIFT LOGGING ALERTS

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

12.3.1. Viewing logging collector alerts

Alerts are shown in the OpenShift Container Platform web console, on the **Alerts** tab of the Alerting UI. Alerts are in one of the following states:

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

Procedure

To view OpenShift Logging and other OpenShift Container Platform alerts:

1. In the OpenShift Container Platform console, click **Monitoring → Alerting**.
2. Click the **Alerts** tab. The alerts are listed, based on the filters selected.

Additional resources

- For more information on the Alerting UI, see Managing alerts.

12.3.2. About logging collector alerts

The following alerts are generated by the logging collector. You can view these alerts in the OpenShift Container Platform web console, on the **Alerts** page of the Alerting UI.

Table 12.1. Fluentd Prometheus alerts

Alert	Message	Description	Severity

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

12.3.3. About Elasticsearch alerting rules

You can view these alerting rules in Prometheus.

Table 12.2. Alerting rules

Alert	Description	Severity
ElasticsearchClusterNotHealthy	The cluster health status has been RED for at least 2 minutes. The cluster does not accept writes, shards may be missing, or the master node hasn't been elected yet.	Critical
ElasticsearchClusterNotHealthy	The cluster health status has been YELLOW for at least 20 minutes. Some shard replicas are not allocated.	Warning
ElasticsearchDiskSpaceRunningLow	The cluster is expected to be out of disk space within the next 6 hours.	Critical

Alert	Description	Severity
ElasticsearchHighFileDescriptorUsage	The cluster is predicted to be out of file descriptors within the next hour.	Warning
ElasticsearchJVMHeapUseHigh	The JVM Heap usage on the specified node is high.	Alert
ElasticsearchNodeDiskWatermarkReached	The specified node has hit the low watermark due to low free disk space. Shards can not be allocated to this node anymore. You should consider adding more disk space to the node.	Info
ElasticsearchNodeDiskWatermarkReached	The specified node has hit the high watermark due to low free disk space. Some shards will be re-allocated to different nodes if possible. Make sure more disk space is added to the node or drop old indices allocated to this node.	Warning
ElasticsearchNodeDiskWatermarkReached	The specified node has hit the flood watermark due to low free disk space. Every index that has a shard allocated on this node is enforced a read-only block. The index block must be manually released when the disk use falls below the high watermark.	Critical
ElasticsearchJVMHeapUseHigh	The JVM Heap usage on the specified node is too high.	Alert
ElasticsearchWriteRequestsRejectionJumps	Elasticsearch is experiencing an increase in write rejections on the specified node. This node might not be keeping up with the indexing speed.	Warning
AggregatedLoggingSystemCPUHigh	The CPU used by the system on the specified node is too high.	Alert
ElasticsearchProcessCPUHigh	The CPU used by Elasticsearch on the specified node is too high.	Alert

12.4. COLLECTING LOGGING DATA FOR RED HAT SUPPORT

When opening a support case, it is helpful to provide debugging information about your cluster to Red Hat Support.

The **must-gather** tool enables you to collect diagnostic information for project-level resources, cluster-level resources, and each of the OpenShift Logging components.

For prompt support, supply diagnostic information for both OpenShift Container Platform and OpenShift Logging.

NOTE

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

12.4.1. About the must-gather tool

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

For your OpenShift Logging environment, **must-gather** collects the following information:

- Project-level resources, including pods, configuration maps, service accounts, roles, role bindings, and events at the project level
- Cluster-level resources, including nodes, roles, and role bindings at the cluster level
- OpenShift Logging resources in the **openshift-logging** and **openshift-operators-redhat** namespaces, including health status for the log collector, the log store, and the log visualizer

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

12.4.2. Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

12.4.3. Collecting OpenShift Logging data

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

Procedure

To collect OpenShift Logging information with **must-gather**:

1. Navigate to the directory where you want to store the **must-gather** information.
2. Run the **oc adm must-gather** command against the OpenShift Logging image:

```
$ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator -o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-operator")].image}')
```

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

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

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

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

12.5. TROUBLESHOOTING FOR CRITICAL ALERTS

12.5.1. Elasticsearch Cluster Health is Red

At least one primary shard and its replicas are not allocated to a node.

Troubleshooting

1. Check the Elasticsearch cluster health and verify that the cluster **status** is red.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- health
   ```

2. List the nodes that have joined the cluster.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
   query=_cat/nodes?v
   ```

3. List the Elasticsearch pods and compare them with the nodes in the command output from the previous step.

   ```
   oc -n openshift-logging get pods -l component=elasticsearch
   ```

4. If some of the Elasticsearch nodes have not joined the cluster, perform the following steps.

   a. Confirm that Elasticsearch has an elected master node.

      ```
      oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
      query=_cat/master?v
      ```

   b. Review the pod logs of the elected master node for issues.

      ```
      oc logs <elasticsearch_master_pod_name> -c elasticsearch -n openshift-logging
      ```

   c. Review the logs of nodes that have not joined the cluster for issues.

      ```
      oc logs <elasticsearch_node_name> -c elasticsearch -n openshift-logging
      ```

5. If all the nodes have joined the cluster, perform the following steps, check if the cluster is in the process of recovering.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
   query=_cat/recovery?active_only=true
   ```

   If there is no command output, the recovery process might be delayed or stalled by pending tasks.

6. Check if there are pending tasks.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- health |grep
   number_of_pending_tasks
   ```

7. If there are pending tasks, monitor their status.

If their status changes and indicates that the cluster is recovering, continue waiting. The recovery time varies according to the size of the cluster and other factors.

Otherwise, if the status of the pending tasks does not change, this indicates that the recovery has stalled.

8. If it seems like the recovery has stalled, check if **cluster.routing.allocation.enable** is set to **none**.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
   query=_cluster/settings?pretty
   ```

9. If **cluster.routing.allocation.enable** is set to **none**, set it to **all**.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
   query=_cluster/settings?pretty -X PUT -d '{"persistent":
   {"cluster.routing.allocation.enable":"all"}}'
   ```

10. Check which indices are still red.

    ```
    oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
    query=_cat/indices?v
    ```

11. If any indices are still red, try to clear them by performing the following steps.

    a. Clear the cache.

       ```
       oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
       query=<elasticsearch_index_name>/_cache/clear?pretty
       ```

    b. Increase the max allocation retries.

       ```
       oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
       query=<elasticsearch_index_name>/_settings?pretty -X PUT -d
       '{"index.allocation.max_retries":10}'
       ```

    c. Delete all the scroll items.

       ```
       oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
       query=_search/scroll/_all -X DELETE
       ```

    d. Increase the timeout.

       ```
       oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
       query=<elasticsearch_index_name>/_settings?pretty -X PUT -d
       '{"index.unassigned.node_left.delayed_timeout":"10m"}'
       ```

12. If the preceding steps do not clear the red indices, delete the indices individually.

    a. Identify the red index name.

       ```
       oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
       query=_cat/indices?v
       ```

b. Delete the red index.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_red_index_name> -X DELETE
```

13. If there are no red indices and the cluster status is red, check for a continuous heavy processing load on a data node.

    a. Check if the Elasticsearch JVM Heap usage is high.

    ```
    oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
    query=_nodes/stats?pretty
    ```

    In the command output, review the **node_name.jvm.mem.heap_used_percent** field to determine the JVM Heap usage.

    b. Check for high CPU utilization.

Additional resources

- Search for "Free up or increase disk space" in the Elasticsearch topic, Fix a red or yellow cluster status.

12.5.2. Elasticsearch Cluster Health is Yellow

Replica shards for at least one primary shard are not allocated to nodes.

Troubleshooting

1. Increase the node count by adjusting **nodeCount** in the **ClusterLogging** CR.

Additional resources

- About the Cluster Logging custom resource
- Configuring persistent storage for the log store
- Search for "Free up or increase disk space" in the Elasticsearch topic, Fix a red or yellow cluster status.

12.5.3. Elasticsearch Node Disk Low Watermark Reached

Elasticsearch does not allocate shards to nodes that reach the low watermark.

Troubleshooting

1. Identify the node on which Elasticsearch is deployed.

```
oc -n openshift-logging get po -o wide
```

2. Check if there are **unassigned shards**.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/health?pretty | grep unassigned_shards
```

3. If there are unassigned shards, check the disk space on each node.

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

4. Check the **nodes.node_name.fs** field to determine the free disk space on that node.
   If the used disk percentage is above 85%, the node has exceeded the low watermark, and shards can no longer be allocated to this node.

5. Try to increase the disk space on all nodes.

6. If increasing the disk space is not possible, try adding a new data node to the cluster.

7. If adding a new data node is problematic, decrease the total cluster redundancy policy.

   a. Check the current **redundancyPolicy**.

   ```
   oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
   ```

   NOTE

   If you are using a **ClusterLogging** CR, enter:

   ```
   oc -n openshift-logging get cl -o
   jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
   ```

   b. If the cluster **redundancyPolicy** is higher than **SingleRedundancy**, set it to **SingleRedundancy** and save this change.

8. If the preceding steps do not fix the issue, delete the old indices.

   a. Check the status of all indices on Elasticsearch.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
   ```

   b. Identify an old index that can be deleted.

   c. Delete the index.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
   query=<elasticsearch_index_name> -X DELETE
   ```

Additional resources

- Search for "redundancyPolicy" in the "Sample **ClusterLogging** custom resource (CR)" in About the Cluster Logging custom resource

12.5.4. Elasticsearch Node Disk High Watermark Reached

Elasticsearch attempts to relocate shards away from a node that has reached the high watermark.

Troubleshooting

1. Identify the node on which Elasticsearch is deployed.

   ```
   oc -n openshift-logging get po -o wide
   ```

2. Check the disk space on each node.

   ```
   for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
   jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
   elasticsearch $pod -- df -h /elasticsearch/persistent; done
   ```

3. Check if the cluster is rebalancing.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
   query=_cluster/health?pretty | grep relocating_shards
   ```

   If the command output shows relocating shards, the High Watermark has been exceeded. The default value of the High Watermark is 90%.

   The shards relocate to a node with low disk usage that has not crossed any watermark threshold limits.

4. To allocate shards to a particular node, free up some space.

5. Try to increase the disk space on all nodes.

6. If increasing the disk space is not possible, try adding a new data node to the cluster.

7. If adding a new data node is problematic, decrease the total cluster redundancy policy.

   a. Check the current **redundancyPolicy**.

      ```
      oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
      ```

      NOTE

      If you are using a **ClusterLogging** CR, enter:

      ```
      oc -n openshift-logging get cl -o
      jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
      ```

   b. If the cluster **redundancyPolicy** is higher than **SingleRedundancy**, set it to **SingleRedundancy** and save this change.

8. If the preceding steps do not fix the issue, delete the old indices.

   a. Check the status of all indices on Elasticsearch.

      ```
      oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
      ```

   b. Identify an old index that can be deleted.

   c. Delete the index.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

Additional resources

- Search for "redundancyPolicy" in the "Sample **ClusterLogging** custom resource (CR)" in About the Cluster Logging custom resource

12.5.5. Elasticsearch Node Disk Flood Watermark Reached

Elasticsearch enforces a read-only index block on every index that has both of these conditions:

- One or more shards are allocated to the node.
- One or more disks exceed the flood stage.

Troubleshooting

1. Check the disk space of the Elasticsearch node.

   ```
   for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
   jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
   elasticsearch $pod -- df -h /elasticsearch/persistent; done
   ```

   Check the **nodes.node_name.fs** field to determine the free disk space on that node.

2. If the used disk percentage is above 95%, it signifies that the node has crossed the flood watermark. Writing is blocked for shards allocated on this particular node.

3. Try to increase the disk space on all nodes.

4. If increasing the disk space is not possible, try adding a new data node to the cluster.

5. If adding a new data node is problematic, decrease the total cluster redundancy policy.

   a. Check the current **redundancyPolicy**.

      ```
      oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
      ```

      NOTE

      If you are using a **ClusterLogging** CR, enter:

      ```
      oc -n openshift-logging get cl -o
      jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
      ```

   b. If the cluster **redundancyPolicy** is higher than **SingleRedundancy**, set it to **SingleRedundancy** and save this change.

6. If the preceding steps do not fix the issue, delete the old indices.

   a. Check the status of all indices on Elasticsearch.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

b. Identify an old index that can be deleted.

c. Delete the index.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

7. Continue freeing up and monitoring the disk space until the used disk space drops below 90%. Then, unblock write to this particular node.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_all/_settings?pretty -X PUT -d '{"index.blocks.read_only_allow_delete": null}'
```

Additional resources

- Search for "redundancyPolicy" in the "Sample **ClusterLogging** custom resource (CR)" in About the Cluster Logging custom resource

12.5.6. Elasticsearch JVM Heap Use is High

The Elasticsearch node JVM Heap memory used is above 75%.

Troubleshooting

Consider increasing the heap size.

12.5.7. Aggregated Logging System CPU is High

System CPU usage on the node is high.

Troubleshooting

Check the CPU of the cluster node. Consider allocating more CPU resources to the node.

12.5.8. Elasticsearch Process CPU is High

Elasticsearch process CPU usage on the node is high.

Troubleshooting

Check the CPU of the cluster node. Consider allocating more CPU resources to the node.

12.5.9. Elasticsearch Disk Space is Running Low

The Elasticsearch Cluster is predicted to be out of disk space within the next 6 hours based on current disk usage.

Troubleshooting

1. Get the disk space of the Elasticsearch node.

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

2. In the command output, check the **nodes.node_name.fs** field to determine the free disk space on that node.

3. Try to increase the disk space on all nodes.

4. If increasing the disk space is not possible, try adding a new data node to the cluster.

5. If adding a new data node is problematic, decrease the total cluster redundancy policy.

   a. Check the current **redundancyPolicy**.

   ```
   oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
   ```

   NOTE

   If you are using a **ClusterLogging** CR, enter:

   ```
   oc -n openshift-logging get cl -o
   jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
   ```

   b. If the cluster **redundancyPolicy** is higher than **SingleRedundancy**, set it to **SingleRedundancy** and save this change.

6. If the preceding steps do not fix the issue, delete the old indices.

   a. Check the status of all indices on Elasticsearch.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
   ```

   b. Identify an old index that can be deleted.

   c. Delete the index.

   ```
   oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
   query=<elasticsearch_index_name> -X DELETE
   ```

Additional resources

- Search for "redundancyPolicy" in the "Sample **ClusterLogging** custom resource (CR)" in About the Cluster Logging custom resource

- Search for "ElasticsearchDiskSpaceRunningLow" in About Elasticsearch alerting rules.

- Search for "Free up or increase disk space" in the Elasticsearch topic, Fix a red or yellow cluster status.

12.5.10. Elasticsearch FileDescriptor Usage is high

Based on current usage trends, the predicted number of file descriptors on the node is insufficient.

Troubleshooting

Check and, if needed, configure the value of **max_file_descriptors** for each node, as described in the Elasticsearch File descriptors topic.

Additional resources

- Search for "ElasticsearchHighFileDescriptorUsage" in About Elasticsearch alerting rules.
- Search for "File Descriptors In Use" in OpenShift Logging dashboards.

CHAPTER 13. UNINSTALLING OPENSHIFT LOGGING

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

13.1. UNINSTALLING OPENSHIFT LOGGING FROM OPENSHIFT CONTAINER PLATFORM

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

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

Prerequisites

- OpenShift Logging and Elasticsearch must be installed.

Procedure

To remove OpenShift Logging:

1. Use the OpenShift Container Platform web console to remove the **ClusterLogging** CR:
   a. Switch to the **Administration** → **Custom Resource Definitions** page.
   b. On the **Custom Resource Definitions** page, click **ClusterLogging**.
   c. On the **Custom Resource Definition Details** page, click **Instances**.
   d. Click the Options menu ⋮ next to the instance and select **Delete ClusterLogging**.
2. Optional: Delete the custom resource definitions (CRD):
   a. Switch to the **Administration** → **Custom Resource Definitions** page.
   b. Click the Options menu ⋮ next to **ClusterLogForwarder** and select **Delete Custom Resource Definition**.
   c. Click the Options menu ⋮ next to **ClusterLogging** and select **Delete Custom Resource Definition**.
   d. Click the Options menu ⋮ next to **Elasticsearch** and select **Delete Custom Resource Definition**.
3. Optional: Remove the Red Hat OpenShift Logging Operator and OpenShift Elasticsearch Operator:
   a. Switch to the **Operators** → **Installed Operators** page.

b. Click the Options menu ⋮ next to the Red Hat OpenShift Logging Operator and select **Uninstall Operator**.

c. Click the Options menu ⋮ next to the OpenShift Elasticsearch Operator and select **Uninstall Operator**.

4. Optional: Remove the OpenShift Logging and Elasticsearch projects.

   a. Switch to the **Home** → **Projects** page.

   b. Click the Options menu ⋮ next to the **openshift-logging** project and select **Delete Project**.

   c. Confirm the deletion by typing **openshift-logging** in the dialog box and click **Delete**.

   d. Click the Options menu ⋮ next to the **openshift-operators-redhat** project and select **Delete Project**.

   IMPORTANT

   Do not delete the **openshift-operators-redhat** project if other global operators are installed in this namespace.

   e. Confirm the deletion by typing **openshift-operators-redhat** in the dialog box and click **Delete**.

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

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

   WARNING

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

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

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

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

Additional resources

- Reclaiming a persistent volume manually

CHAPTER 14. LOG RECORD FIELDS

The following fields can be present in log records exported by OpenShift Logging. Although log records are typically formatted as JSON objects, the same data model can be applied to other encodings.

To search these fields from Elasticsearch and Kibana, use the full dotted field name when searching. For example, with an Elasticsearch **/_search URL**, to look for a Kubernetes pod name, use **/_search/q=kubernetes.pod_name:name-of-my-pod**.

The top level fields may be present in every record.

CHAPTER 15. MESSAGE

The original log entry text, UTF-8 encoded. This field may be absent or empty if a non-empty **structured** field is present. See the description of **structured** for more.

Data type	text
Example value	**HAPPY**

CHAPTER 16. STRUCTURED

Original log entry as a structured object. This field may be present if the forwarder was configured to parse structured JSON logs. If the original log entry was a valid structured log, this field will contain an equivalent JSON structure. Otherwise this field will be empty or absent, and the **message** field will contain the original log message. The **structured** field can have any subfields that are included in the log message, there are no restrictions defined here.

Data type	group
Example value	map[message:starting fluentd worker pid=21631 ppid=21618 worker=0 pid:21631 ppid:21618 worker:0]

CHAPTER 17. @TIMESTAMP

A UTC value that marks when the log payload was created or, if the creation time is not known, when the log payload was first collected. The “@” prefix denotes a field that is reserved for a particular use. By default, most tools look for “@timestamp” with ElasticSearch.

Data type	date
Example value	**2015-01-24 14:06:05.071000000 Z**

CHAPTER 18. HOSTNAME

The name of the host where this log message originated. In a Kubernetes cluster, this is the same as **kubernetes.host**.

Data type	keyword

CHAPTER 19. IPADDR4

The IPv4 address of the source server. Can be an array.

Data type	ip

CHAPTER 20. IPADDR6

The IPv6 address of the source server, if available. Can be an array.

Data type	ip

CHAPTER 21. LEVEL

The logging level from various sources, including **rsyslog(severitytext property)**, a Python logging module, and others.

The following values come from **syslog.h**, and are preceded by their numeric equivalents:

- **0** = **emerg**, system is unusable.
- **1** = **alert**, action must be taken immediately.
- **2** = **crit**, critical conditions.
- **3** = **err**, error conditions.
- **4** = **warn**, warning conditions.
- **5** = **notice**, normal but significant condition.
- **6** = **info**, informational.
- **7** = **debug**, debug-level messages.

The two following values are not part of **syslog.h** but are widely used:

- **8** = **trace**, trace-level messages, which are more verbose than **debug** messages.
- **9** = **unknown**, when the logging system gets a value it doesn't recognize.

Map the log levels or priorities of other logging systems to their nearest match in the preceding list. For example, from python logging, you can match **CRITICAL** with **crit**, **ERROR** with **err**, and so on.

Data type	keyword
Example value	**info**

CHAPTER 22. PID

The process ID of the logging entity, if available.

Data type	keyword

CHAPTER 23. SERVICE

The name of the service associated with the logging entity, if available. For example, syslog's **APP-NAME** and rsyslog's **programname** properties are mapped to the service field.

Data type	keyword

CHAPTER 24. TAGS

Optional. An 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.

Data type	text

CHAPTER 25. FILE

The path to the log file from which the collector reads this log entry. Normally, this is a path in the **/var/log** file system of a cluster node.

Data type	text

CHAPTER 26. 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), so 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).

Data type	long

CHAPTER 27. KUBERNETES

The namespace for Kubernetes-specific metadata

Data type	group

27.1. KUBERNETES.POD_NAME

The name of the pod

Data type	keyword

27.2. KUBERNETES.POD_ID

The Kubernetes ID of the pod

Data type	keyword

27.3. KUBERNETES.NAMESPACE_NAME

The name of the namespace in Kubernetes

Data type	keyword

27.4. KUBERNETES.NAMESPACE_ID

The ID of the namespace in Kubernetes

Data type	keyword

27.5. KUBERNETES.HOST

The Kubernetes node name

Data type	keyword

27.6. KUBERNETES.CONTAINER_NAME

The name of the container in Kubernetes

Data type	keyword

27.7. KUBERNETES.ANNOTATIONS

Annotations associated with the Kubernetes object

Data type	group

27.8. KUBERNETES.LABELS

Labels present on the original Kubernetes Pod

Data type	group

27.9. KUBERNETES.EVENT

The Kubernetes event obtained from the Kubernetes master API. This event description loosely follows **type Event** in Event v1 core.

Data type	group

27.9.1. kubernetes.event.verb

The type of event, **ADDED**, **MODIFIED**, or **DELETED**

Data type	keyword
Example value	**ADDED**

27.9.2. kubernetes.event.metadata

Information related to the location and time of the event creation

Data type	group

27.9.2.1. kubernetes.event.metadata.name

The name of the object that triggered the event creation

Data type	keyword
Example value	**java-mainclass-1.14d888a4cfc24890**

27.9.2.2. kubernetes.event.metadata.namespace

The name of the namespace where the event originally occurred. Note that it differs from **kubernetes.namespace_name**, which is the namespace where the **eventrouter** application is deployed.

Data type	keyword
Example value	**default**

27.9.2.3. kubernetes.event.metadata.selfLink

A link to the event

Data type	keyword
Example value	**/api/v1/namespaces/javaj/events/java-mainclass-1.14d888a4cfc24890**

27.9.2.4. kubernetes.event.metadata.uid

The unique ID of the event

Data type	keyword
Example value	**d828ac69-7b58-11e7-9cf5-5254002f560c**

27.9.2.5. kubernetes.event.metadata.resourceVersion

A string that identifies the server's internal version of the event. Clients can use this string to determine when objects have changed.

Data type	integer
Example value	**311987**

27.9.3. kubernetes.event.involvedObject

The object that the event is about.

Data type	group

27.9.3.1. kubernetes.event.involvedObject.kind

The type of object

Data type	keyword
Example value	**ReplicationController**

27.9.3.2. kubernetes.event.involvedObject.namespace

The namespace name of the involved object. Note that it may differ from **kubernetes.namespace_name**, which is the namespace where the **eventrouter** application is deployed.

Data type	keyword
Example value	**default**

27.9.3.3. kubernetes.event.involvedObject.name

The name of the object that triggered the event

Data type	keyword
Example value	**java-mainclass-1**

27.9.3.4. kubernetes.event.involvedObject.uid

The unique ID of the object

Data type	keyword
Example value	**e6bff941-76a8-11e7-8193-5254002f560c**

27.9.3.5. kubernetes.event.involvedObject.apiVersion

The version of kubernetes master API

Data type	keyword
Example value	**v1**

27.9.3.6. kubernetes.event.involvedObject.resourceVersion

A string that identifies the server's internal version of the pod that triggered the event. Clients can use this string to determine when objects have changed.

Data type	keyword
Example value	**308882**

27.9.4. kubernetes.event.reason

A short machine-understandable string that gives the reason for generating this event

Data type	keyword
Example value	**SuccessfulCreate**

27.9.5. kubernetes.event.source_component

The component that reported this event

Data type	keyword
Example value	**replication-controller**

27.9.6. kubernetes.event.firstTimestamp

The time at which the event was first recorded

Data type	date
Example value	**2017-08-07 10:11:57.000000000 Z**

27.9.7. kubernetes.event.count

The number of times this event has occurred

Data type	integer
Example value	**1**

27.9.8. kubernetes.event.type

The type of event, **Normal** or **Warning**. New types could be added in the future.

Data type	keyword
Example value	**Normal**

CHAPTER 28. OPENSHIFT

The namespace for openshift-logging specific metadata

Data type	group

28.1. OPENSHIFT.LABELS

Labels added by the Cluster Log Forwarder configuration

Data type	group