OpenShift Container Platform 4.3

日志记录

在 OpenShift Container Platform 中配置集群日志记录

Last Updated: 2020-11-30

OpenShift Container Platform 4.3 日志记录

在 OpenShift Container Platform 中配置集群日志记录

法律通告

摘要

本文提供有关安装、配置和使用集群日志记录的说明，该功能可汇总多个 OpenShift Container Platform 服务的日志。

目录

第 1 章 关于集群日志记录和 OPENSHIFT CONTAINER PLATFORM

作为集群管理员，您可以部署集群日志记录来聚合 OpenShift Container Platform 集群中的所有日志，如节点系统日志、应用程序容器日志等。

1.1. 集群日志记录

OpenShift Container Platform 集群管理员可以使用一些 CLI 命令和 OpenShift Container Platform Web 控制台安装 Elasticsearch Operator 和 Cluster Logging Operator，以此部署集群日志记录。安装 Operator 后，可创建集群日志记录自定义资源 (CR) 以调度集群日志记录 pod 和支持集群日志记录所需的其他资源。Operator 负责部署、升级和维护集群日志记录。

您可以通过修改集群日志记录自定义资源 (CR)（名为 **instance**）来配置集群日志记录。CR 定义包括日志记录堆栈的所有组件在内的完整集群日志记录部署，以收集、存储和视觉化日志。Cluster Logging Operator 监控 **ClusterLogging** 自定义资源并相应地调整日志记录部署。

管理员和应用程序开发人员可以查看他们具有查看访问权限的项目的日志。

1.1.1. 关于集群日志记录组件

集群日志记录组件基于 Elasticsearch、Fluentd 或 Rsyslog 以及 Kibana（EFK）。收集器 Fluentd 部署到 OpenShift Container Platform 集群中的每个节点。它收集所有节点和容器日志，并将它们写入 Elasticsearch (ES)。Kibana 是一个集中式 Web UI，用户和管理员可以在其中使用汇总的数据创建丰富的视觉化和仪表板。

目前有 5 种不同类型的集群日志记录组件：

- logStore（存储） - 存储日志的位置。当前的实现是 Elasticsearch。
- collection（收集） - 此组件从节点收集日志，将日志格式化并存储到 logStore 中。当前的实现是 Fluentd。
- visualization（可视化） - 此 UI 组件用于查看日志、图形和图表等。当前的实现是 Kibana。
- curation（策展） - 此组件按日志时间进行筛检。当前的实现是 Curator。

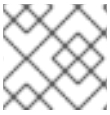
除非特别说明，在本文中我们可能会互换使用日志存储或 Elasticsearch、视觉化或 Kibana、策展或 Curator、收集或 Fluentd。

1.1.2. 关于日志存储

OpenShift Container Platform 使用 Elasticsearch (ES) 将日志数据从 Fluentd 整理到数据存储或 *索引*中。

Elasticsearch 将各个索引进一步划分成多个碎片（称为*分片*），分散到 Elasticsearch 集群中的一组 Elasticsearch 节点上。您可以配置 Elasticsearch 来为分片制作拷贝（称为*副本*）。Elasticsearch 也将这些副本分散到多个 Elasticsearch 节点上。借助 **ClusterLogging** 自定义资源，您可以在自定义资源定义 (CRD) 中指定复制策略，以提供数据冗余和故障恢复能力。

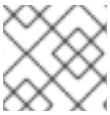
集群日志记录 Elasticsearch 实例经过优化并测试，用于大约 7 天的简短存储。如果要更长时间保留日志，建议您将数据移至第三方存储系统。



注意

索引模板的主分片数量等于 Elasticsearch 数据节点的数目。

Cluster Logging Operator 和相应的 Elasticsearch Operator 确保每个 Elasticsearch 节点都使用带有自身存储卷的唯一 Deployment 来进行部署。您可以使用集群日志记录自定义资源 (CR) 来增加 Elasticsearch 节点的数量。如需有关选择存储和网络位置的注意事项，请参考 Elastic 文档，如下所示。



注意

高可用性 Elasticsearch 环境需要至少三个 Elasticsearch 节点，各自在不同的主机上。

Elasticsearch 索引中应用的基于角色的访问控制 (RBAC) 可让开发人员控制对日志的访问。使用 **project.{project_name}.{project_uuid}.*** 格式对索引进行访问将会根据特定项目中的用户权限进行限制。

如需更多信息，请参阅 Elasticsearch (ES)。

1.1.3. 关于日志记录收集器

OpenShift Container Platform 使用 Fluentd 来收集集群的相关数据。

日志记录收集器部署为 OpenShift Container Platform 中的 DaemonSet，将各个 Pod 部署到每个 OpenShift Container Platform 节点中。**journald** 是系统日志源，提供来自操作系统、容器运行时和 OpenShift Container Platform 的日志消息。

容器运行时提供少许信息来标识日志消息的来源，如项目、容器名称和容器 ID。这不足以唯一地标识日志来源。如果在日志收集器开始处理日志之前删除了具有指定名称和项目的 Pod，则来自 API 服务器的信息（如标签和注解）可能会不可用。可能没有办法区分来自名称相似的 Pod 和项目的日志消息，也无法追溯日志的来源。这种局限性意味着日志收集和规范化仅属于**尽力而为**。

重要

可用的容器运行时提供少许信息来标识日志消息来源，无法确保唯一的个别日志消息，也不能保证可以追溯这些消息的来源。

如需更多信息，请参阅 Fluentd。

1.1.4. 关于日志记录视觉化

OpenShift Container Platform 使用 Kibana 显示由 Fluentd 收集并由 Elasticsearch 索引的日志数据。

Kibana 是基于浏览器的控制台界面，可通过直方图、折线图、饼图、热图、内置地理空间支持和其他视觉化方式，来查询、探索和视觉化您的 Elasticsearch 数据。

如需更多信息，请参阅 Kibana。

1.1.5. 关于日志记录策展

Elasticsearch Curator 工具在全局范围和/或以项目为基础执行调度的维护操作。Curator 根据其配置执行操作。建议每个 Elasticsearch 集群仅使用一个 Curator Pod。

```
spec:
  curation:
```

```
  type: "curator"
  resources:
  curator:
    schedule: "30 3 * * *" (1)
```

(1) 以 cron 格式指定 Curator 计划。

如需更多信息，请参阅 Curator。

1.1.6. 关于事件路由

Event Router 是一个 Pod，它监视 OpenShift Container Platform 事件，以便通过集群日志记录来收集这些事件。Event Router 从所有项目收集事件，并将其写入 **STDOUT**。Fluentd 收集这些事件并将其转发到 OpenShift Container Platform Elasticsearch 实例。Elasticsearch 将事件索引到 **infra** 索引。

您必须手动部署 Event Router。

1.1.7. 集群日志记录自定义资源 (CR)

要更改集群日志记录部署，请创建并修改集群日志记录自定义资源 (CR) 。本文根据需要提供了有关创建或修改 CR 的说明。

以下是集群日志记录的典型自定义资源的示例。

集群日志记录 CR 示例

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage:
        storageClassName: "gp2"
        size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 1Gi
        requests:
```

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

第 2 章 关于部署集群日志记录

在将集群日志记录安装到 OpenShift Container Platform 集群前，请先查阅以下小节。

2.1. 关于部署和配置集群日志记录

OpenShift Container Platform 集群日志记录已设计为可搭配默认配置使用，该配置针对中小型 OpenShift Container Platform 集群进行了调优。

以下安装说明包括一个示例集群日志记录自定义资源 (CR)，您可以用它来创建集群日志记录实例并配置集群日志记录部署。

如果要使用默认集群日志记录安装，可直接使用示例 CR。

如果要自定义部署，请根据需要对示例 CR 进行更改。下文介绍了在安装集群日志记录实例或安装后修改时可以进行的配置。请参阅"配置"部分来了解有关使用各个组件的更多信息，包括可以在集群日志记录自定义资源之外进行的修改。

2.1.1. 配置和调优集群日志记录

您可以通过修改 **openshift-logging** 项目中部署的集群日志记录自定义资源来配置集群日志记录环境。

您可以在安装时或安装后修改以下任何组件：

内存和 CPU

您可以使用有效的内存和 CPU 值修改 **resources** 块，以此调整各个组件的 CPU 和内存限值：

```
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu:
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      type: "elasticsearch"
  collection:
    logs:
      fluentd:
        resources:
          limits:
            cpu:
            memory:
          requests:
            cpu:
            memory:
        type: "fluentd"
  visualization:
    kibana:
      resources:
        limits:
          cpu:
          memory:
```

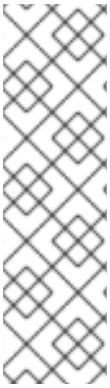
```
      requests:
        cpu:
        memory:
    type: kibana
  curation:
    curator:
      resources:
        limits:
          memory: 200Mi
        requests:
          cpu: 200m
          memory: 200Mi
      type: "curator"
```

Elasticsearch 存储

您可以使用 **storageClass name** 和 **size** 参数，为 Elasticsearch 集群配置持久性存储类和大小。Cluster Logging Operator 基于这些参数为 Elasticsearch 集群中的每个数据节点创建 **PersistentVolumeClaim**。

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

本例中指定，集群中的每个数据节点将绑定到请求 200G 的 gp2 存储的 **PersistentVolumeClaim**。每个主分片将由单个副本支持。



注意

省略 **storage** 块会导致部署中仅包含临时存储。

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

Elasticsearch 复制策略

您可以通过设置策略来定义如何在集群中的数据节点之间复制 Elasticsearch 分片：

- **FullRedundancy**。各个索引的分片完整复制到每个数据节点上。
- **MultipleRedundancy**。各个索引的分片分布到一半数据节点上。
- **SingleRedundancy**。各个分片具有单个副本。只要存在至少两个数据节点，日志就能始终可用且可恢复。
- **ZeroRedundancy**。所有分片均无副本。如果节点关闭或发生故障，则可能无法获得日志数据。

Curator 调度

以 cron 格式指定 Curator 的调度。

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

2.1.2. 修改后集群日志记录自定义资源示例

以下是使用前述选项修改的集群日志记录自定义资源的示例。

修改后集群日志记录自定义资源示例

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          memory: 32Gi
        requests:
          cpu: 3
          memory: 32Gi
      storage: {}
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 1Gi
      replicas: 1
  curation:
    type: "curator"
    curator:
      resources:
        limits:
          memory: 200Mi
        requests:
          cpu: 200m
          memory: 200Mi
```

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

2.2. 集群日志记录和 OPENSHIFT CONTAINER PLATFORM 的存储注意事项

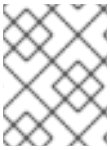
每个 Elasticsearch 部署都需要一个持久性卷，以便每个数据节点均有一个数据卷。在 OpenShift Container Platform 中，这可以使用 PVC 来实现。

Elasticsearch Operator 使用 Elasticsearch 资源名称来命名 PVC。如需更多详细信息，请参阅“Elasticsearch 持久性存储”。

Fluentd 将 **systemd journal** 和 **/var/log/containers/** 的所有日志都传输到 Elasticsearch。

因此，请提前考虑需要的数据量，并且注意要聚合应用程序的日志数据。一些 Elasticsearch 用户发现，有必要使绝对存储消耗始终保持在 50% 左右并处于 70% 以下。这有助于避免 Elasticsearch 在进行大型数据合并操作期间变得无响应。

默认情况下，达到 85%时 Elasticsearch 会停止向节点分配新数据，达到 90% 时 Elasticsearch 会尝试将现有分片从节点重新定位到其他节点（若有可能）。但是，如果存储消耗低于 85% 时无节点有可用存储空间，Elasticsearch 会拒绝创建新索引并且变为 RED。



注意

这些高、低水位线值是当前版本中的 Elasticsearch 默认值。您可以修改这些值，但需要同时将修改应用到警报。警报基于这些默认值。

2.3. 其他资源

如需有关安装 Operator 的更多信息，请参阅从 OperatorHub 安装 Operator。

第 3 章 部署集群日志记录

您可以通过部署 Elasticsearch 和 Cluster Logging Operator 来安装集群日志记录。Elasticsearch Operator 负责创建并管理由集群日志记录使用的 Elasticsearch 集群。Cluster Logging Operator 负责创建并管理日志记录堆栈的组件。

将集群日志记录部署到 OpenShift Container Platform 的过程涉及以下任务：

- 查阅关于部署集群日志记录中的安装选项。
- 查阅集群日志记录存储注意事项。
- 安装 Elasticsearch Operator 和 Cluster Logging Operator。

3.1. 使用 CLI 安装 ELASTICSEARCH OPERATOR

您**必**须按照以下说明来使用 CLI 安装 Elasticsearch Operator。

先决条件

确保具有 Elasticsearch 所需的持久性存储。注意每个 Elasticsearch 节点都需要自己的存储卷。

Elasticsearch 是内存密集型应用程序。默认情况下，OpenShift Container Platform 安装 3 个 Elasticsearch 节点，其内存请求和限制为 16 GB。初始设置的三个 OpenShift Container Platform 节点可能没有足够的内存在集群中运行 Elasticsearch。如果遇到与 Elasticsearch 相关的内存问题，您应该在集群中添加更多 Elasticsearch 节点，而不是在退出的节点上增加内存。

流程

使用 CLI 安装 Elasticsearch Operator：

1. 为 Elasticsearch Operator 创建一个命名空间。

   a. 为 Elasticsearch Operator 创建一个命名空间对象 YAML 文件（例如 **eo-namespace.yaml**）：

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

(1) 您必须指定 **openshift-operators-redhat** 命名空间。为了防止可能与指标（metrics）冲突，您应该将 Prometheus Cluster Monitoring 堆栈配置为从 **openshift-operators-redhat** 命名空间中提取指标数据，而不是从 **openshift-operators** 命名空间中提取。**openshift-operators** 命名空间可能会包含社区提供的 operator。这些 operator 不被信任，其发布的 metric 可能与 OpenShift Container Platform metric 的名称相同，从而导致冲突。

(2) 您必须按照所示指定该标签，以确保集群监控提取 **openshift-operators-redhat** 命名空间。

b. 创建命名空间：

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

例如：

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

2. 通过创建以下对象来安装 Elasticsearch Operator：

a. 为 Elasticsearch Operator 创建 Operator Group 对象 YAML 文件（例如 **eo-og.yaml**）：

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

(1) 您必须指定 **openshift-operators-redhat** 命名空间。

b. 创建 Operator Group 对象：

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

例如：

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

c. 创建 Subscription 对象 YAML 文件（例如 **eo-sub.yaml**）来订阅 Operator 的命名空间。

订阅示例

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

(1) 您必须指定 **openshift-operators-redhat** 命名空间。

(2) 指定 **4.3** 作为频道。

(3) 指定 **redhat-operators**。如果 OpenShift Container Platform 集群安装在受限网络中（也称为断开连接的集群），请指定配置 Operator Lifecycle Manager (OLM) 时创建的 CatalogSource 对象的名称。

d. 创建订阅对象：

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

例如：

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

e. 更改到 **openshift-operators-redhat** 项目：

```
$ oc project openshift-operators-redhat

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

f. 创建基于角色的访问控制 (RBAC) 对象文件（例如 **eo-rbac.yaml**），向 Prometheus 授予 **openshift-operators-redhat** 命名空间的访问权限：

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

g. 创建 RBAC 对象：

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

例如：

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

Elasticsearch Operator 已安装到 **openshift-operators-redhat** 命名空间，并且复制到集群中的每个项目。

3. 验证 Operator 安装：

```
oc get csv --all-namespaces

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

每个命名空间中都应该有一个 Elasticsearch Operator。版本号可能与所示不同。

后续步骤

使用控制台或 CLI 安装 Cluster Logging Operator。

3.2. 使用 WEB 控制台安装 CLUSTER LOGGING OPERATOR。

使用 OpenShift Container Platform web 控制台安装 Cluster Logging Operator。

注意

您不能通过 web 控制台或使用 **oc new-project** 命令创建名称以 **openshift-** 开始的项目。您必须使用一个 YAML 对象文件创建一个命名空间（Namespace），并运行 **oc create -f <file-name>.yaml** 命令，如下所示。

流程

使用 OpenShift Container Platform web 控制台安装 Cluster Logging Operator：

1. 为 Cluster Logging Operator 创建一个命名空间。您必须使用 CLI 创建命名空间。

   a. 为 Cluster Logging Operator 创建一个命名空间对象 YAML 文件（例如，**clo-namespace.yaml**）：

```
apiVersion: v1
```

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

(1)(2)(3)如下所示指定这些值。

b. 创建命名空间：

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

例如：

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

2. 安装 Cluster Logging Operator：

a. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。

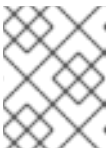
b. 从可用 Operator 列表中选择 **Cluster Logging**，再点击 **Install**。

c. 在 **Create Operator Subscription** 页面中，在 **A specific namespace on the cluster** 下选择 **openshift-logging**。然后，点击 **Subscribe**。

3. 验证 Cluster Logging Operator 已被安装。

a. 切换到 **Operators → Installed Operators** 页面。

b. 确保 **openshift-logging** 项目中列出的 **Cluster Logging** 的 **Status** 为 **InstallSucceeded**。



注意

在安装过程中，Operator 可能会显示 **Failed** 状态。如果 Operator 随后被安装并显示 **InstallSucceeded** 信息，则可以忽略 **Failed** 信息。

如果 Operator 没有被成功安装，请按照以下步骤进行故障排除：

- 切换到 **Operators → Installed Operators** 页面，并检查 **Status** 列中是否有任何错误或故障。
- 切换到 **Workloads → Pods** 页面，并检查 **openshift-logging** 和 **openshift-operators-redhat** 项目中报告问题的 Pod 的日志。

4. 创建集群日志记录实例：

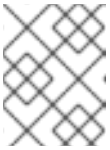
a. 切换到 **Administration → Custom Resource Definitions** 页面。

b. 在 **Custom Resource Definitions** 页面上，点 **ClusterLogging**。

c. 在 **Custom Resource Definition Overview** 页面上，从 **Actions** 菜单中选择 **View Instances**。

d. 在 **Cluster Loggings** 页面上，点击 **Create Cluster Logging**。
您可能需要刷新页面来加载数据。

e. 将 YAML 项中的代码替换为以下内容：



注意

此默认集群日志记录配置应该可以支持不同的环境。请参考有关调优和配置集群日志记录组件的主题，以了解有关可对集群日志记录集群进行修改的信息。

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

1. 名称必须是 **instance**。

2. 集群日志记录管理状态。在大多数情况下，如果更改了集群日志记录的默认值，则必须将其设置为 **Unmanaged**。但是，非受管部署不接收更新，直到集群日志记录重新变为受管状态为止。如需更多信息，请参阅**更改集群日志记录管理状态**。

3. 用于配置 Elasticsearch 的设置。通过使用 CR，您可以配置分片复制策略和持久性存储。如需更多信息，请参阅**配置 Elasticsearch**。

4. 指定 Elasticsearch 节点的数量。请参阅此列表后面的备注。

5. 为 Elasticsearch 存储输入现有 StorageClass 的名称。为获得最佳性能，请指定分配块存储的 StorageClass。

6. 用于配置 Kibana 的设置。通过使用 CR，您可以扩展 Kibana 来实现冗余性，并为 Kibana 节点配置 CPU 和内存。如需更多信息，请参阅**配置 Kibana**。

7 用于配置 Curator 的设置。通过使用 CR，您可以设置 Curator 调度。如需更多信息，请参阅**配置 Curator**。

8 用于配置 Fluentd 的设置。通过使用 CR，您可以配置 Fluentd CPU 和内存限值。如需更多信息，请参阅**配置 Fluentd**。

注意

Elasticsearch master 节点的最大数量为三个。如果您将 **nodeCount** 指定为大于 **3**，OpenShift Container Platform 只会创建三个符合 Master 节点条件的 Elasticsearch 节点（具有 master、client 和 data 角色）。其余 Elasticsearch 节点创建为"仅数据"节点，使用 client 和 data 角色。Master 节点执行集群范围的操作，如创建或删除索引、分配分片和跟踪节点等。数据节点保管分片，并执行与数据相关的操作，如 CRUD、搜索和聚合等。与数据相关的操作会占用大量 I/O、内存和 CPU。务必要监控这些资源，并在当前节点过载时添加更多数据节点。

例如，如果 **nodeCount = 4**，则创建以下节点：

```
$ oc get deployment

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

索引模板的主分片数量等于 Elasticsearch 数据节点的数目。

f. 点击 **Create**。这将创建集群日志记录自定义资源和 Elasticsearch 自定义资源，您可以通过编辑它们来更改用于集群日志记录的集群。

5. 验证安装：

a. 切换到 **Workloads → Pods** 页面。

b. 选择 **openshift-logging** 项目。
您应该会看到几个用于集群日志记录、Elasticsearch、Fluentd 和 Kibana 的 Pod，类似于以下列表：

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

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

3.3. 使用 CLI 安装 CLUSTER LOGGING OPERATOR

您可以使用 OpenShift Container Platform CLI 安装 Cluster Logging Operator。Cluster Logging Operator 负责创建并管理日志记录堆栈的组件。

流程

使用 CLI 安装 Cluster Logging Operator：

1. 为 Cluster Logging Operator 创建命名空间：
   a. 为 Cluster Logging Operator 创建一个命名空间对象 YAML 文件（例如，**clo-namespace.yaml**）：

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

   b. 创建命名空间：

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

   例如：

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

2. 通过创建以下对象来安装 Cluster Logging Operator：
   a. 为 Cluster Logging Operator 创建一个 OperatorGroup 对象 YAML 文件（例如，**clo-og.yaml**）：

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

   (1) (2) 您必须指定 **openshift-logging** 命名空间。

   b. 创建 OperatorGroup 对象：

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

例如：

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

c. 创建一个 Subscription 对象 YAML 文件（例如，**clo-sub.yaml**）来向 Operator 订阅一个 Namespace。

订阅示例

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

(1) 您必须指定 **openshift-logging** 命名空间。

(2) 指定 **4.3** 作为频道。

(3) 指定 **redhat-operators**。如果 OpenShift Container Platform 集群安装在受限网络中（也称为断开连接的集群），请指定配置 Operator Lifecycle Manager (OLM) 时创建的 CatalogSource 对象的名称。

d. 创建订阅对象：

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

例如：

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

Cluster Logging Operator 已安装到 **openshift-logging** 命名空间。

3. 验证 Operator 安装：
在 **openshift-logging** 命名空间中应该有一个 Cluster Logging Operator。版本号可能与所示不同。

```
oc get csv --all-namespaces

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

4. 创建集群日志记录（Cluster Logging）实例：

   a. 为 Cluster Logging Operator 创建实例对象 YAML 文件（如 **clo-instance.yaml**）：

注意

此默认集群日志记录配置应该可以支持不同的环境。请参考有关调优和配置集群日志记录组件的主题，以了解有关可对集群日志记录集群进行修改的信息。

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

1. 名称必须是 **instance**。
2. 集群日志记录管理状态。在大多数情况下，如果更改了集群日志记录的默认值，则必须将其设置为 **Unmanaged**。但是，在 Cluster Logging 返回到 **Managed** 状态前，非受管部署不接收更新。如需更多信息，请参阅**更改集群日志记录管理状态**。
3. 用于配置 Elasticsearch 的设置。通过使用子定义资源（CR），您可以配置分片复制策略和持久性存储。如需更多信息，请参阅**配置 Elasticsearch**。
4. 指定 Elasticsearch 节点的数量。请参阅此列表后面的备注。
5. 为 Elasticsearch 存储输入现有 StorageClass 的名称。为获得最佳性能，请指定分配块存储的 StorageClass。
6. 用于配置 Kibana 的设置。通过使用 CR，您可以扩展 Kibana 来实现冗余性，并为 Kibana 节点配置 CPU 和内存。如需更多信息，请参阅**配置 Kibana**。

7 用于配置 Curator 的设置。通过使用 CR，您可以设置 Curator 调度。如需更多信息，请参阅**配置 Curator**。

8 用于配置 Fluentd 的设置。通过使用 CR，您可以配置 Fluentd CPU 和内存限值。如需更多信息，请参阅**配置 Fluentd**。

注意

Elasticsearch master 节点的最大数量为三个。如果您将 **nodeCount** 指定为大于 **3**，OpenShift Container Platform 只会创建三个符合 Master 节点条件的 Elasticsearch 节点（具有 master、client 和 data 角色）。其余 Elasticsearch 节点创建为"仅数据"节点，使用 client 和 data 角色。Master 节点执行集群范围的操作，如创建或删除索引、分配分片和跟踪节点等。数据节点保管分片，并执行与数据相关的操作，如 CRUD、搜索和聚合等。与数据相关的操作会占用大量 I/O、内存和 CPU。务必要监控这些资源，并在当前节点过载时添加更多数据节点。

例如，如果 **nodeCount = 4**，则创建以下节点：

```
$ oc get deployment

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

索引模板的主分片数量等于 Elasticsearch 数据节点的数目。

b. 创建实例：

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

例如：

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

5. 通过列出 **openshift-logging** 项目中的 Pod 来验证安装。
您应该会看到几个用于 Cluster Logging、Elasticsearch、Fluentd 和 Kibana 的 Pod，类似于以下内容：

```
oc get pods -n openshift-logging

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. 其他资源

如需有关安装 Operator 的更多信息，请参阅从 OperatorHub 安装 Operator。

第 4 章 更新集群日志记录

在将 OpenShift Container Platform 集群从 4.2 升级到 4.3 后，必须将集群日志记录从 4.2 升级到 4.3。

4.1. 更新集群日志记录

升级 OpenShift Container Platform 集群后，您可以通过更新 Elasticsearch Operator 和 Cluster Logging Operator 的订阅将集群日志记录从 4.2 升级到 4.3。

重要

从 OpenShift Container Platform 4.3 开始，新引进的日志转发功能带来的变化改变了 **out_forward**。在 OpenShift Container Platform 4.3 中，您可以创建一个 ConfigMap 来配置 **out_forward**。任何对 Fluentd ConfigMap 中的 **secure-forward.conf** 部分的更新都会被删除。

如果使用 **out_forward** 插件，在更新前，您可以从 Fluentd ConfigMap 中复制当前的 **secure-forward.conf** 部分，并在创建 **secure-forward** ConfigMap 时使用复制的数据。

先决条件

- 将集群从 4.2 更新至 4.3。
- 确保集群日志记录具有健康状态：
  - 所有 Pod 都为 **Ready** 状态。
  - Elasticsearch 集群处于健康状态。
- 另外，如果您需要创建 **secure-forward** ConfigMap，复制当前 Fluentd ConfigMap 中的 **secure-forward.conf** 部分的内容。请参阅上述备注。

流程

1. 更新 Elasticsearch Operator：
   a. 在 Web 控制台中，点 **Operators → Installed Operators**。
   b. 选择 **openshift-logging** 项目。
   c. 点 **Elasticsearch Operator**。
   d. 点 **Subscription → Channel**。
   e. 在 **Change Subscription Update Channel** 窗口中，选择 **4.3** 并点 **Save**。
   f. 等待几秒钟，然后点 **Operators → Installed Operators**。
   Elasticsearch Operator 显示为 4.3 版本。例如：

   ```
   Elasticsearch Operator
   4.3.0-201909201915 provided
   by Red Hat, Inc
   ```

2. 更新 Cluster Logging Operator：

a. 在 Web 控制台中，点 **Operators → Installed Operators**。

b. 选择 **openshift-logging** 项目。

c. 点 **Cluster Logging Operator**。

d. 点 **Subscription → Channel**。

e. 在 **Change Subscription Update Channel**窗口中，选择 **4.3** 并点 **Save**。

f. 等待几秒钟，然后点 **Operators → Installed Operators**。
Cluster Logging Operator 显示为 4.3 版本。例如：

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

3. 检查日志记录组件：

a. 确保 Elasticsearch Pod 使用的是 4.3 镜像：

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

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

b. 确保所有 Elasticsearch Pod 都处于 **Ready** 状态：

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

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

c. 确保 Elasticsearch 集群健康：

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

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

```
  "status" : "green",

....
```

d. 确保日志记录收集器 Pod 使用的是 4.3 镜像：

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

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

e. 确保 Kibana Pod 使用的是 4.3 镜像：

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

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

f. 确保 Curator CronJob 使用的是 4.3 镜像：

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

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

第 5 章 查看集群日志

您可以使用 CLI 或通过 OpenShift Container Platform web 控制台查看 OpenShift Container Platform 集群的日志。

5.1. 查看集群日志

您可以在 CLI 中查看集群日志。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

查看集群日志：

使用 **oc logs [-f] <pod_name>** 命令，其中的 **-f** 是可选的。

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

1 指定日志收集器 Pod 的名称。使用 **-f** 选项可以跟踪正在写进日志中的内容。

例如：

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

输出的日志文件内容。

默认情况下，Fluentd 从日志的末尾开始读取日志。

5.2. 在 OPENSHIFT CONTAINER PLATFORM WEB 控制台中查看集群日志

您可以在 OpenShift Container Platform Web 控制台中查看集群日志。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

查看集群日志：

1. 在 OpenShift Container Platform 控制台中，进入 **Workloads → Pods**。
2. 从下拉菜单中选择 **openshift-logging** 项目。
3. 点带有 **fluentd** 前缀的某一个日志记录收集器 Pod。
4. 点击 **Logs**。

默认情况下，Fluentd 从日志的末尾开始读取日志。

第 6 章 使用 KIBANA 查看集群日志

在安装集群日志记录时会部署 Kibana web 控制台。

6.1. 启动 KIBANA

Kibana 是基于浏览器的控制台界面，可通过直方图、折线图、饼图、热图、内置地理空间支持和其他视觉化方式，来查询、探索和视觉化您的日志数据。

先决条件

如果使用代理安装 OpenShift Container Platform，需要将 **.apps.<cluster_name>.<base_domain>** 添加到集群范围的 Proxy 对象的 **noProxy** 列表中。

例如：

```
$ oc edit proxy/cluster

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

(1) 把 **.apps.<cluster_name>.<base_domain>** 添加到 **noProxy** 列表。排除代理的目标域名、域、IP 地址或其他网络 CIDR 的逗号分隔列表。

流程

启动 Kibana：

1. 在 OpenShift Container Platform 控制台中点 **Monitoring → Logging**。
2. 使用用来登录到 OpenShift Container Platform 控制台的相同凭证进行登录。Kibana 界面将出现。您现在可以：
   - 使用 Discover 页面搜索并浏览您的数据。
   - 使用 Visualize 页面对数据进行图表显示。
   - 使用 Dashboard 页面创建并查看自定义仪表板。

使用并配置 Kibana 界面的内容超出了本文档的范围。相关信息，请参阅 Kibana 文档。

注意

如果您在 Kibana 控制台中看到 **security_exception** 错误，且无法访问 Kibana 索引，则代表您可能有已过期的 OAuth 令牌。如果您看到这个错误，请登出 Kibana 控制台后再重新登录。这会刷新您的 OAuth 令牌，您应该能够访问您的索引。

第 7 章 配置集群日志记录部署

7.1. 关于配置集群日志记录

将集群日志记录安装到 OpenShift Container Platform 集群中后，您可以进行以下配置。

注意

除非另有说明，否则在执行这些配置之前，必须将集群日志记录设置为非受管状态。如需更多信息，请参阅更改集群日志记录管理状态。

处于非受管状态的 Operator 不被正式支持，集群管理员需要完全掌控各个组件的配置和升级。

如需了解更多信息，请参阅非受管 Operator 的支持策略

7.1.1. 关于部署和配置集群日志记录

OpenShift Container Platform 集群日志记录已设计为可搭配默认配置使用，该配置针对中小型 OpenShift Container Platform 集群进行了调优。

以下安装说明包括一个示例集群日志记录自定义资源 (CR)，您可以用它来创建集群日志记录实例并配置集群日志记录部署。

如果要使用默认集群日志记录安装，可直接使用示例 CR。

如果要自定义部署，请根据需要对示例 CR 进行更改。下文介绍了在安装集群日志记录实例或安装后修改时可以进行的配置。请参阅"配置"部分来了解有关使用各个组件的更多信息，包括可以在集群日志记录自定义资源之外进行的修改。

7.1.1.1. 配置和调优集群日志记录

您可以通过修改 **openshift-logging** 项目中部署的集群日志记录自定义资源来配置集群日志记录环境。

您可以在安装时或安装后修改以下任何组件：

内存和 CPU

您可以使用有效的内存和 CPU 值修改 **resources** 块，以此调整各个组件的 CPU 和内存限值：

```
spec:
  logStore:
    elasticsearch:
      resources:
        limits:
          cpu:
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      type: "elasticsearch"
  collection:
    logs:
      fluentd:
        resources:
```

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

Elasticsearch 存储

您可以使用 **storageClass name** 和 **size** 参数，为 Elasticsearch 集群配置持久性存储类和大小。Cluster Logging Operator 基于这些参数为 Elasticsearch 集群中的每个数据节点创建 **PersistentVolumeClaim**。

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

本例中指定，集群中的每个数据节点将绑定到请求 200G 的 gp2 存储的 **PersistentVolumeClaim**。每个主分片将由单个副本支持。

注意

省略 **storage** 块会导致部署中仅包含临时存储。

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

Elasticsearch 复制策略

您可以通过设置策略来定义如何在集群中的数据节点之间复制 Elasticsearch 分片：

- **FullRedundancy**。各个索引的分片完整复制到每个数据节点上。
- **MultipleRedundancy**。各个索引的分片分布到一半数据节点上。
- **SingleRedundancy**。各个分片具有单个副本。只要存在至少两个数据节点，日志就能始终可用且可恢复。
- **ZeroRedundancy**。所有分片均无副本。如果节点关闭或发生故障， 则可能无法获得日志数据。

Curator 调度

以 cron 格式指定 Curator 的调度。

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

7.1.1.2. 修改后集群日志记录自定义资源示例

以下是使用前述选项修改的集群日志记录自定义资源的示例。

修改后集群日志记录自定义资源示例

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          memory: 32Gi
        requests:
          cpu: 3
          memory: 32Gi
      storage: {}
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 1Gi
```

```
      requests:
        cpu: 500m
        memory: 1Gi
    replicas: 1
curation:
  type: "curator"
  curator:
    resources:
      limits:
        memory: 200Mi
      requests:
        cpu: 200m
        memory: 200Mi
    schedule: "*/5 * * * *"
collection:
  logs:
    type: "fluentd"
    fluentd:
      resources:
        limits:
          memory: 1Gi
        requests:
          cpu: 200m
          memory: 1Gi
```

7.2. 更改集群日志记录管理状态

若要修改由 Cluster Logging Operator 或 Elasticsearch Operator 管理的特定组件，您必须将 Operator 设置为*非受管*状态。

在非受管状态下，Operator 不响应 CR 中的变化。处于非托管状态时，管理员完全掌控各个组件的配置和升级。

重要

处于非受管状态的 Operator 不被正式支持，集群管理员需要完全掌控各个组件的配置和升级。

如需了解更多信息，请参阅非受管 Operator 的支持策略

在受管状态下，Cluster Logging Operator (CLO) 会对集群日志记录自定义资源 (CR) 中的更改进行相应，并对日志部署进行相应的调整。

OpenShift Container Platform 文档在预备步骤中指明何时您必须将 OpenShift Container Platform 集群设置为“非受管（Unmanaged）”状态。

注意

如果将 Elasticsearch Operator (EO) 设置为非受管状态，并将 Cluster Logging Operator (CLO) 保留为受管状态，则 CLO 会还原您对 EO 进行的更改，因为 EO 由 CLO 进行管理。

7.2.1. 更改集群日志记录管理状态

您必须将 Cluster Logging Operator 设置为*非受管*状态，才能修改由此 Operator 管理的组件：

- Curator CronJob；
- Elasticsearch CR；
- Kibana Deployment；
- 日志收集器 DaemonSet。

如果您在受管状态下对这些组件进行更改，则 Cluster Logging Operator 将还原这些更改。

注意

在 Cluster Logging Operator 返回到受管状态之前，非受管集群日志记录环境不会接收更新。

先决条件

- 必须安装 Cluster Logging Operator。

流程

1. 在 **openshift-logging** 项目中编辑集群日志记录自定义资源 (CR)：

```
$ oc edit ClusterLogging instance
```

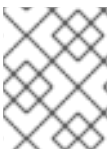
```
$ oc edit ClusterLogging instance

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

....

spec:
  managementState: "Managed" (1)
```

(1) 将管理状态指定为 **Managed** 或 **Unmanaged**。

7.2.2. 更改 Elasticsearch 管理状态

您必须将 Elasticsearch Operator 设置为*非受管*状态，才能修改由此 Operator 管理的 Elasticsearch 部署文件：

如果您在受管状态下对这些组件进行更改，则 Elasticsearch Operator 将还原这些更改。



注意

在 Elasticsearch Operator 返回到受管状态之前，非受管 Elasticsearch 集群不会接收更新。

先决条件

- 必须安装 Elasticsearch Operator。
- 具有 **openshift-logging** 项目中 Elasticsearch CR 的名称：

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

流程

编辑 **openshift-logging** 项目中的 Elasticsearch 自定义资源 (CR)：

```
$ oc edit Elasticsearch elasticsearch

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


....

spec:
  managementState: "Managed" 1
```

1 将管理状态指定为 **Managed** 或 **Unmanaged**。

注意

如果将 Elasticsearch Operator (EO) 设置为非受管状态，并将 Cluster Logging Operator (CLO) 保留为受管状态，则 CLO 会还原您对 EO 进行的更改，因为 EO 由 CLO 进行管理。

7.3. 配置集群日志记录

可以使用部署在 **openshift-logging** 项目中的集群日志记录自定义资源 (CR) 来配置集群日志记录。

Cluster Logging Operator 会监控集群日志记录 CR 的更改，创建任何缺少的日志记录组件，并相应地调整日志记录部署。

集群日志记录 CR 基于集群日志记录自定义资源定义 (CRD)，后者定义完整的集群日志记录部署，包括日志记录堆栈中用于收集、存储和视觉化日志的所有组件。

集群日志记录自定义资源 (CR) 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  creationTimestamp: '2019-03-20T18:07:02Z'
  generation: 1
  name: instance
  namespace: openshift-logging
```

```
spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  curation:
    curator:
      resources: null
      schedule: 30 3 * * *
    type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu:
          memory:
        requests:
          cpu:
          memory:
      storage: {}
    type: elasticsearch
  managementState: Managed
  visualization:
    kibana:
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

您可以对集群日志记录进行以下配置：

- 可以将集群日志记录置于非受管状态，使管理员能够完全掌控各个组件的配置和升级。
- 可以通过在 **cluster-logging-operator** 部署中修改适当的环境变量，覆盖每个集群日志记录组件的镜像。
- 可以使用节点选择器为日志记录组件指定特定的节点。

7.3.1. 了解集群日志记录组件镜像

集群日志记录中有多个组件，各自通过一个或多个镜像实施。每个镜像都由 **openshift-logging** 项目的 **cluster-logging-operator** 部署中定义的环境变量指定的，并且不应更改。

您可以通过运行以下命令来查看镜像：

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

```
ELASTICSEARCH_IMAGE=registry.redhat.io/openshift4/ose-logging-elasticsearch5:v4.3 (1)
FLUENTD_IMAGE=registry.redhat.io/openshift4/ose-logging-fluentd:v4.3 (2)
KIBANA_IMAGE=registry.redhat.io/openshift4/ose-logging-kibana5:v4.3 (3)
```

```
CURATOR_IMAGE=registry.redhat.io/openshift4/ose-logging-curator5:v4.3 4
OAUTH_PROXY_IMAGE=registry.redhat.io/openshift4/ose-oauth-proxy:v4.3 5
```

1. **ELASTICSEARCH_IMAGE** 部署 Elasticsearch。
2. **FLUENTD_IMAGE** 部署 Fluentd。
3. **KIBANA_IMAGE** 部署 Kibana。
4. **CURATOR_IMAGE** 部署 Curator。
5. **OAUTH_PROXY_IMAGE** 为 OpenShift Container Platform 定义 OAUTH。

这些值可能视您的环境而异。

重要

日志记录路由由 Cluster Logging Operator 管理，用户无法修改它。

7.4. 配置 ELASTICSEARCH 以存储和整理日志数据

OpenShift Container Platform 使用 Elasticsearch (ES) 来存储和整理日志数据。

您可以对日志存储进行的一些修改包括：

- Elasticsearch 集群存储；
- 如何在集群中的数据节点之间复制分片，包括从完整复制到不复制；
- 允许外部对 Elasticsearch 数据进行访问。

注意

不支持缩减 Elasticsearch 节点。缩减规模时，Elasticsearch Pod 可能会被意外删除，这可能导致未分配分片，并且丢失副本分片。

Elasticsearch 是内存密集型应用程序。每个 Elasticsearch 节点需要 16G 内存来满足内存请求（requests）和限值（limits）的需要，除非集群日志记录自定义资源中另有指定。最初的 OpenShift Container Platform 节点组可能不足以支持 Elasticsearch 集群。您必须在 OpenShift Container Platform 集群中添加额外的节点，才能使用建议或更高的内存来运行。

每个 Elasticsearch 节点都可以在较低的内存设置下运行，但在生产部署中不建议这样做。

注意

如果将 Elasticsearch Operator (EO) 设置为非受管状态，并将 Cluster Logging Operator (CLO) 保留为受管状态，则 CLO 会还原您对 EO 进行的更改，因为 EO 由 CLO 进行管理。

7.4.1. 配置 Elasticsearch CPU 和内存限值

每个组件规格都允许调整 CPU 和内存限值。您应该无需手动调整这些值，因为 Elasticsearch Operator 会设置适当的值以满足环境的要求。

每个 Elasticsearch 节点都可以在较低的内存设置下运行，但在生产部署中**不建议**这样做。对于生产环境，为每个 Pod 应该分配的数量应不少于默认的 16Gi。最好为每个 Pod 分配不超过 64Gi 的尽量多的数量。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 在 **openshift-logging** 项目中编辑集群日志记录自定义资源 (CR)：

```
$ oc edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
....
spec:
    logStore:
      type: "elasticsearch"
      elasticsearch:
        resources: 1
          limits:
            memory: 16Gi
          requests:
            cpu: 500m
            memory: 16Gi
```

1 根据需要指定 CPU 和内存限值。如果这些值留白，则 Elasticsearch Operator 会设置默认值，它们应足以满足大多数部署的需要。

如果调整了 Elasticsearch CPU 和内存的数量，您必须同时更改请求值和限制值。

例如：

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

Kubernetes 一般遵循节点 CPU 配置，DOES 不允许 Elasticsearch 使用指定的限制。为 **requests** 和 **limits** 设置相同的值可以确保 Elasticseach 可以使用您需要它们使用的 CPU 和内存数量（假定节点有足够可用 CPU 和内存）。

7.4.2. 配置 Elasticsearch 复制策略

您可以定义如何在集群中的数据节点之间复制 Elasticsearch 分片：

先决条件

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 在 **openshift-logging** 项目中编辑集群日志记录自定义资源 (CR)：

```
oc edit clusterlogging instance
```

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

....

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

1 为分片指定冗余策略。更改会在保存后应用。

- **FullRedundancy**：Elasticsearch 将每个索引的主分片完整复制到每个数据节点。这可提供最高的安全性，但代价是需要最大数量的磁盘并且性能最差。
- **MultipleRedundancy**：Elasticsearch 将每个索引的主分片完整复制到一半的数据节点。这可在安全性和性能之间提供很好的折衷。
- **SingleRedundancy**：Elasticsearch 为每个索引的主分片制作一个副本。只要存在至少两个数据节点，日志就能始终可用且可恢复。使用 5 个或更多节点时，性能胜过 MultipleRedundancy。您不能将此策略应用于单个 Elasticsearch 节点的部署。
- **ZeroRedundancy**：Elasticsearch 不制作主分片的副本。如果节点关闭或发生故障，则可能无法获得日志数据。如果您更关注性能而非安全性，或者实施了自己的磁盘/PVC 备份/恢复策略，可以考虑使用此模式。

注意

索引模板的主分片数量等于 Elasticsearch 数据节点的数目。

7.4.3. 配置 Elasticsearch 存储

Elasticsearch 需要持久性存储。存储速度越快，Elasticsearch 性能越高。

警告

在 Elasticsearch 存储中不支持将 NFS 存储用作卷或持久性卷（或者通过 NAS 比如 Gluster），因为 Lucene 依赖于 NFS 不提供的文件系统行为。数据崩溃和其他问题可能会发生。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 编辑集群日志记录 CR，将集群中的每个数据节点指定为绑定到 PVC。

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

....

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

本例中指定，集群中的每个数据节点都绑定到请求"200G"的 AWS 通用 SSD (gp2) 存储的 PVC。

7.4.4. 为 Elasticsearch 配置 emptyDir 存储

您可以将 emptyDir 与 Elasticsearch 搭配使用来创建一个临时部署，临时部署一旦重启其中所有 Pod 的数据都会丢失。

注意

使用 emptyDir 时，如果重启或重新部署 Elasticsearch，数据将会丢失。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 编辑集群日志记录 CR 以指定 emptyDir：

```
spec:
```

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

7.4.5. 将 Elasticsearch 公开为路由

默认情况下，无法从日志记录集群外部访问部署了集群日志记录的 Elasticsearch。您可以启用一个 re-encryption termination 模式的路由，以实现外部对 Elasticsearch 的访问来获取数据。

另外，还可以在外部创建一个重新加密路由，使用 OpenShift Container Platform 令牌和已安装的 Elasticsearch CA 证书以从外部访问 Elasticsearch。然后，使用包含以下信息的 cURL 请求来访问 Elasticsearch 节点：

- **Authorization: Bearer ${token}**
- Elasticsearch 重新加密路由和 Elasticsearch API 请求。

在内部，您可以使用 Elasticsearch 集群 IP 访问 Elastiscearch：

您可以使用以下命令之一获取 Elasticsearch 集群 IP：

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

172.30.183.229
```

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

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

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

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

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。
- 您必须具有项目的访问权限，以便能访问其日志。

流程

对外部公开 Elasticsearch：

1. 进入 **openshift-logging** 项目：

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

2. 从 Elasticsearch 提取 CA 证书并写入 ***admin-ca*** 文件：

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

admin-ca
```

3. 以 YAML 文件形式创建 Elasticsearch 服务的路由：

   a. 使用以下内容创建一个 YAML文件：

```
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 添加 Elasticsearch CA 证书或使用下一步中的命令。您不必设置一些重新加密路由所需的 **spec.tls.key**、**spec.tls.certificate** 和 **spec.tls.caCertificate** 参数。

   b. 运行以下命令将 Elasticsearch CA 证书添加到您创建的路由 YAML 中：

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

   c. 创建路由：

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

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

4. 检查是否公开了 Elasticsearch 服务：

   a. 获取此 ServiceAccount 的令牌，以便在请求中使用：

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

   b. 将您创建的 **Elasticsearch** 路由设置为环境变量。

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

   c. 要验证路由是否创建成功，请运行以下命令来通过公开的路由访问 Elasticsearch：

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

   其响应类似于如下：

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

```
                                 Dload  Upload   Total   Spent    Left  Speed
100   944  100   944    0     0     62      0  0:00:15  0:00:15 --:--:--   204
{
  "took": 441,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 89157,
    "max_score": 1,
    "hits": [
      {
        "_index": ".operations.2019.03.15",
        "_type": "com.example.viaq.common",
        "_id": "ODdiNWIyYzAtMjg5Ni0TAtNWE3MDY1MjMzNTc3",
        "_score": 1,
        "_source": {
          "_SOURCE_MONOTONIC_TIMESTAMP": "673396",
          "systemd": {
            "t": {
              "BOOT_ID": "246c34ee9cdeecb41a608e94",
              "MACHINE_ID": "e904a0bb5efd3e36badee0c",
              "TRANSPORT": "kernel"
            },
            "u": {
              "SYSLOG_FACILITY": "0",
              "SYSLOG_IDENTIFIER": "kernel"
            }
          },
          "level": "info",
          "message": "acpiphp: Slot [30] registered",
          "hostname": "localhost.localdomain",
          "pipeline_metadata": {
            "collector": {
              "ipaddr4": "10.128.2.12",
              "ipaddr6": "fe80::xx:xxxx:fe4c:5b09",
              "inputname": "fluent-plugin-systemd",
              "name": "fluentd",
              "received_at": "2019-03-15T20:25:06.273017+00:00",
              "version": "1.3.2 1.6.0"
            }
          },
          "@timestamp": "2019-03-15T20:00:13.808226+00:00",
          "viaq_msg_id": "ODdiNWIyYzAtMYTAtNWE3MDY1MjMzNTc3"
        }
      }
    ]
  }
}
```

7.4.6. 关于 Elasticsearch 警报规则

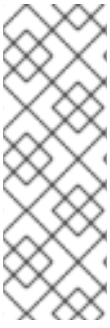
您可以在 Prometheus 中查看这些警报规则。

警报	描述	重要性
ElasticsearchClusterNotHealthy	集群健康状态为 RED 至少有 2 分钟。集群不接受写操作，分片可能缺失或者 master 节点尚未选定。	critical
ElasticsearchClusterNotHealthy	集群健康状态为 YELLOW 至少有 20 分钟。某些分片副本尚未分配。	warning
ElasticsearchBulkRequestsRejectionJumps	集群中节点的批量拒绝率高。此节点可能无法跟上索引速度。	warning
ElasticsearchNodeDiskWatermarkReached	集群中节点已达到磁盘低水位线。分片无法再分配给此节点。应该考虑向节点添加更多磁盘空间。	alert
ElasticsearchNodeDiskWatermarkReached	集群中节点已达到磁盘高水位线。若有可能，某些分片将重新分配到其他节点。确保向节点添加更多磁盘空间，或者丢弃分配给此节点的旧索引。	high
ElasticsearchJVMHeapUseHigh	集群中节点上的 JVM 堆使用量为 <value>	alert
AggregatedLoggingSystemCPUHigh	集群中节点上的系统 CPU 使用率是 <value>	alert
ElasticsearchProcessCPUHigh	集群中节点上的 ES 进程 CPU 使用率是 <value>	alert

7.5. 配置 KIBANA

OpenShift Container Platform 使用 Kibana 显示由 Fluentd 收集并由 Elasticsearch 索引的日志数据。

您可以扩展 Kibana 来实现冗余性，并为 Kibana 节点配置 CPU 和内存。



注意

除非另有说明，否则在执行这些配置之前，必须将集群日志记录设置为非受管状态。如需更多信息，请参阅更改集群日志记录管理状态。

处于非受管状态的 Operator 不被正式支持，集群管理员需要完全掌控各个组件的配置和升级。

如需了解更多信息，请参阅非受管 Operator 的支持策略

7.5.1. 配置 Kibana CPU 和 内存限值

每个组件规格都允许调整 CPU 和内存限值。

流程

1. 在 **openshift-logging** 项目中编辑集群日志记录自定义资源 (CR)：

```
$ oc edit ClusterLogging instance
```

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

....

spec:
    visualization:
      type: "kibana"
      kibana:
        replicas:
      resources:  1
        limits:
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 1Gi
      proxy:  2
        resources:
          limits:
            memory: 100Mi
          requests:
            cpu: 100m
            memory: 100Mi
```

1. 指定要为每个节点分配的 CPU 和内存限值。
2. 指定要为 Kibana 代理分配的 CPU 和内存限值。

7.5.2. 扩展 Kibana 以实现冗余

您可以扩展 Kibana 部署以实现冗余。

流程

1. 在 **openshift-logging** 项目中编辑集群日志记录自定义资源 (CR)：

```
$ 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) 指定 Kibana 节点的数量。

7.5.3. 使用容忍度（toleration）来控制 Kibana Pod 放置

您可以通过在 Pod 上使用容限（toleration）来控制 Kibana Pod 在哪些节点上运行，并防止其他工作负载使用这些节点。

您可以通过集群日志记录自定义资源 (CR) 将容限应用到 Kibana Pod，并通过节点规格将污点（taint）应用到节点。节点上的污点是一个 **key:value** 对，它指示节点排斥所有不容许该污点的 Pod。通过使用没有在其他 Pod 上使用的特定 **key:value** 对，可以确保仅 Kibana Pod 能够在该节点上运行。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 使用以下命令，将污点添加到要在其上调度 Kibana Pod 的节点：

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

例如：

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

本例在 **node1** 上放置一个键为 **kibana** 且值为 **node** 的污点，污点效果是 **NoExecute**。您必须使用 **NoExecute** 污点设置。**NoExecute** 仅调度与污点匹配的 Pod，并删除不匹配的现有 Pod。

2. 编辑集群日志记录自定义资源 (CR) 的 **visualization** 部分，以配置 Kibana Pod 的容忍度：

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

(1) 指定添加到节点的键。

(2) 指定 **Exists** 运算符，以要求匹配 **key/value/effect** 参数。

(3) 指定 **NoExecute** 效果。

(4) （可选）指定 **tolerationSeconds** 参数，以设置 Pod 在被逐出前可以保持绑定到节点的时长。

此容忍度与 **oc adm taint** 命令创建的污点匹配。具有此容限的 Pod 能够调度到 **node1** 上。

7.5.4. 安装 Kibana Visualize 工具

借助 Kibana 的 **Visualize** 选项卡，您可以创建用于监控容器日志的视觉化和仪表板，让管理员用户（**cluster-admin** 或 **cluster-reader**）能够按照部署、命名空间、Pod 和容器来查看日志。

流程

加载仪表板和其他 Kibana UI 对象：

1. 若有必要，可获取安装 Cluster Logging Operator 时默认创建的 Kibana 路由：

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

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

2. 获取 Elasticsearch Pod 的名称。

```
$ oc get pods -l component=elasticsearch

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

3. 创建此过程需要的用户级配置：

   a. 以您想要添加仪表板的用户身份登录到 Kibana 仪表板。

   ```
   https://kibana-openshift-logging.apps.openshift.com 1
   ```

   1 Kibana 路由所在的 URL。

   b. 如果显示 **Authorize Access** 页面，请选择所有权限，再点 **Allow selected permissions**。

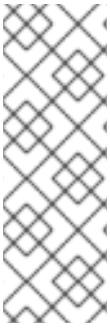
   c. 退出 Kibana 仪表板。

4. 使用任何 Elastiscearch Pod 的名称，从 Pod 所在的项目运行以下命令：

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

例如：

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



注意

Kibana 对象（如视觉化、仪表板等）的元数据以 **.kibana.{user_hash}** 索引格式保存在 Elasticsearch 中。您可以使用 **userhash=$(echo -n $username | sha1sum | awk '{print $1}')** 命令获取 **user_hash**。默认情况下，Kibana **shared_ops** 索引模式允许具有集群管理员角色的所有用户共享索引，并将这个 Kibana 对象元数据保存到 **.kibana** 索引中。

任何自定义仪表板都可以通过使用导入/导出功能，或通过使用 curl 命令将元数据插入 Elasticsearch 索引来为特定用户导入。

7.6. ELASTICSEARCH 数据策展

Elasticsearch Curator 工具在全局范围和/或以项目为基础执行调度的维护操作。Curator 根据其配置执行操作。

Cluster Logging Operator 将安装 Curator 及其配置。您可以使用集群日志记录自定义资源配置 Curator cron 计划，其他配置选项则可在 **openshift-logging** 项目中的 Curator ConfigMap **curator**（融合了 Curator 配置文件 ***curator5.yaml*** 以及 OpenShift Container Platform 自定义配置文件 ***config.yaml***）中找到。

OpenShift Container Platform 在内部使用 ***config.yaml*** 来生成 Curator **action** 文件。

（可选）您可以直接使用此 **action** 文件。通过编辑此文件，您可以使用 Curator 提供的可定期运行的任何操作。但是，仅建议高级用户使用此功能，因为修改文件可能会对集群造成破坏，并可能导致从 Elasticsearch 中删除必要的索引/设置。大多数用户仅需修改 Curator 配置映射，无需编辑 **action** 文件。

7.6.1. 配置 Curator 计划

您可以使用由集群日志记录安装创建的集群日志记录自定义资源来指定 Curator 的调度。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

配置 Curator 调度：

1. 编辑 **openshift-logging** 项目中的集群日志记录自定义资源：

```
$ oc edit clusterlogging instance
```

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

...

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

1 以 cron 格式指定 Curator 的调度。

注意

时区是根据 Curator Pod 运行所在的主机节点设置的。

7.6.2. 配置 Curator 索引删除

您可以配置 Curator，以根据保留设置删除 Elasticsearch 数据。您可以配置在特定项目范围内的设置，也可以配置全局范围的设置。全局设置应用到任何未指定的项目。特定项目范围内的设置会覆盖全局设置。

先决条件

- 必须安装集群日志记录。

流程

删除索引：

1. 编辑 OpenShift Container Platform 自定义 Curator 配置文件：

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

2. 根据需要设置以下参数：

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

可用的参数如下：

表 7.1. 项目选项

变量名称	描述
project_name	项目的实际名称，例如 **myapp-devel**。对于 OpenShift Container Platform **operations** 日志，请使用 **.operations** 作为项目名称。
action	当前只支持 **delete**。
unit	用于删除的期限，可以是 **days**、**weeks** 或 **months**。
value	单位数。

表 7.2. 过滤选项

变量名称	描述
.defaults	使用 **.defaults** 作为 **project_name**，可为尚未指定的项目设置默认值。
.regex	与项目名称匹配的正则表达式列表。

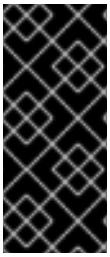
变量名称	描述
pattern	有效且正确转义的正则表达式，用单引号括起。

例如，要将 Curator 配置为：

- 删除 **myapp-dev** 项目中存在时间超过 **1 天**的索引
- 删除 **myapp-qe** 项目中存在时间超过 **1 个星期**的索引
- 删除存在时间超过 **8 个星期**的operations日志
- 删除所有其他项目中存在时间超过 **31 天**的索引
- 删除与 **^project\..+\-dev.*$** 正则表达式匹配且存在时间超过 1 天的索引
- 删除与 **^project\..+\-test.*$** 正则表达式匹配且存在时间超过 2 天的索引

使用：

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

  .operations:
    delete:
      weeks: 8

  myapp-dev:
    delete:
      days: 1

  myapp-qe:
    delete:
      weeks: 1

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



重要

当您将 **months** 用作操作的 **$UNIT** 时，Curator 会从当月的第一天开始计算，而不是当月的当天。例如，如果今天是 4 月 15 日，并且您想要删除目前存在时间已达 2 个月的索引（delete: months: 2），Curator 不会删除日期在 2 月 15 日前的索引，而是会删除日期在 2 月 1 日前的索引。也就是说，它会退回到当前月份的第一天，然后从该日期起返回两个整月。如果您想使 Curator 准确一些，则最好使用 days（例如 **delete: days: 30**）。

7.6.3. Curator 故障排除

您可以参照本节中的信息来调试 Curator。例如，如果 Curator 处于失败状态，但日志消息未提供原因，您可以提高日志级别并触发新任务，而不必等待另一次调度运行 cron 任务。

先决条件

必须安装 Cluster Logging 和 Elasticsearch。

流程

启用 Curator 调试日志并手动触发下一次 Curator 操作

1. 启用 Curator 的调试日志：

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

   指定日志级别：

   - **CRITICAL**：Curator 仅显示严重消息。
   - **ERROR**：Curator 仅显示错误和严重消息。
   - **WARNING**：Curator 仅显示错误、警告和严重消息。
   - **INFO**：Curator 仅显示参考、错误、警告和严重消息。
   - **DEBUG**：除上述所有消息外，Curator 仅显示调试消息。
     默认值为 INFO。

注意

集群日志记录在 OpenShift Container Platform 打包程序脚本（**run.sh** 和 **convert.py**）中使用 OpenShift Container Platform 自定义环境变量 **CURATOR_SCRIPT_LOG_LEVEL**。根据需要，环境变量采用与 **CURATOR_LOG_LEVEL** 相同的值进行脚本调试。

2. 触发下一次 Curator 迭代：

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

3. 使用以下命令来控制 CronJob：

   - 暂停 CronJob：

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

   - 恢复 CronJob：

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

   - 更改 CronJob 调度：

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

1 **schedule** 选项接受 cron 格式的调度。

7.6.4. 在脚本化部署中配置 Curator

如果必须在脚本化部署中配置 Curator，请使用本节中的信息。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。
- 将集群日志记录设置为非受管状态。

流程

在脚本中使用以下代码片段配置 Curator：

- 如果是脚本化部署

  1. 创建并修改配置：

     a. 从 Curator 配置映射中复制 Curator 配置文件和 OpenShift Container Platform 自定义配置文件，并为它们分别创建单独的文件：

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

     b. 编辑 */my/config/curator5.yaml* 和 */my/config/config.yaml* 文件。

  2. 删除现有的 Curator 配置映射，并将编辑后的 YAML 文件添加到新 Curator 配置映射中。

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

     下一次操作将使用此配置。

- 如果使用 **action** 文件：

  1. 创建并修改配置：

     a. 从 Curator 配置映射中复制 Curator 配置文件和 **action** 文件，并为它们分别创建单独的文件：

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

     b. 编辑 */my/config/curator5.yaml* 和 */my/config/actions.yaml* 文件。

  2. 删除现有的 Curator 配置映射，并将编辑后的 YAML 文件添加到新 Curator 配置映射中。

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

下一次操作将使用此配置。

7.6.5. 使用 Curator Action 文件

openshift-logging 项目中的 **Curator** ConfigMap 包含一个 Curator **action** 文件，您可以在其中配置任何要定期运行的 Curator 操作。

不过，使用 **action** 文件时，OpenShift Container Platform 会忽略 **curator** ConfigMap 中的 **config.yaml** 部分，它配置为用于确保不误删重要的内部索引。若要使用 **action** 文件，您应在配置中添加排除规则来保留这些索引。您还必须按照本主题中的步骤，手动添加所有其他模式。

重要

action 和 **config.yaml** 是互斥的配置文件。一旦存在 **action** 文件，OpenShift Container Platform 就会忽略 **config.yaml** 文件。仅建议高级用户使用 **action** 文件，因为使用该文件可能会对集群造成破坏，并可能导致从 Elasticsearch 中删除必要的索引/设置。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。
- 将集群日志记录设置为非受管状态。处于非受管状态的 Operator 不被正式支持，集群管理员需要完全掌控各个组件的配置和升级。

流程

配置 Curator 以删除索引：

1. 编辑 Curator ConfigMap：

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

2. 对 **action** 文件进行以下更改：

```
actions:
1:
  action: delete_indices (1)
  description: >-
    Delete .operations indices older than 30 days.
    Ignore the error if the filter does not
    result in an actionable list of indices (ignore_empty_list).
    See
https://www.elastic.co/guide/en/elasticsearch/client/curator/5.2/ex_delete_indices.html
  options:
    # Swallow curator.exception.NoIndices exception
    ignore_empty_list: True
    # In seconds, default is 300
    timeout_override: ${CURATOR_TIMEOUT}
    # Don't swallow any other exceptions
    continue_if_exception: False
    # Optionally disable action, useful for debugging
    disable_action: False
  # All filters are bound by logical AND
  filters: (2)
  - filtertype: pattern
```

```
    kind: regex
    value: '^\.operations\..*$'
    exclude: False    3
  - filtertype: age
    # Parse timestamp from index name
    source: name
    direction: older
    timestring: '%Y.%m.%d'
    unit: days
    unit_count: 30
    exclude: False
```

1 指定 **delete_indices** 以删除指定的索引。

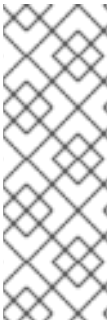
2 使用 **filers** 参数以指定要删除的索引。如需这些参数的信息，请参阅 Elasticsearch Curator 文档。

3 指定 **false** 以允许删除索引。

7.7. 配置日志记录收集器

OpenShift Container Platform 使用 Fluentd 从集群中收集操作和应用程序日志，并借助 Kubernetes Pod 和命名空间元数据丰富这些日志。

您可以配置日志轮转和日志位置，使用外部日志聚合器，以及为日志收集器进行其他配置。



注意

除非另有说明，否则在执行这些配置之前，必须将集群日志记录设置为非受管状态。如需更多信息，请参阅更改集群日志记录管理状态。

处于非受管状态的 Operator 不被正式支持，集群管理员需要完全掌控各个组件的配置和升级。

如需了解更多信息，请参阅非受管 Operator 的支持策略

7.7.1. 查看日志记录收集器 Pod

您可以使用 **oc get pods --all-namespaces -o wide** 命令查看部署了 Fluentd 的节点。

流程

在 **openshift-logging** 项目中运行以下命令：

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

NAME                  READY   STATUS    RESTARTS   AGE     IP           NODE
NOMINATED NODE   READINESS GATES
fluentd-5mr28         1/1     Running   0          4m56s   10.129.2.12  ip-10-0-164-233.ec2.internal
<none>         <none>
fluentd-cnc4c         1/1     Running   0          4m56s   10.128.2.13  ip-10-0-155-142.ec2.internal
<none>         <none>
fluentd-nlp8z         1/1     Running   0          4m56s   10.131.0.13  ip-10-0-138-77.ec2.internal
<none>         <none>
```

```
fluentd-rknlk           1/1     Running   0          4m56s   10.128.0.33   ip-10-0-128-130.ec2.internal   <none>           <none>
fluentd-rsm49           1/1     Running   0          4m56s   10.129.0.37   ip-10-0-163-191.ec2.internal   <none>           <none>
fluentd-wjt8s           1/1     Running   0          4m56s   10.130.0.42   ip-10-0-156-251.ec2.internal   <none>           <none>
```

7.7.2. 配置日志收集器 CPU 和内存限值

日志收集器允许对 CPU 和内存限值进行调整。

流程

1. 在 **openshift-logging** 项目中编辑集群日志记录自定义资源 (CR)：

```
$ oc edit ClusterLogging instance
```

```
$ oc edit ClusterLogging instance

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

....

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

1 根据需要指定 CPU 和内存限值及请求。显示的值是默认值。

7.7.3. 为 Fluentd 配置缓冲数据块限制

如果 Fluentd 日志记录器保存大量日志，Fluentd 会执行文件缓冲来降低内存用量并防止数据丢失。

Fluentd 文件缓冲以*数据块*的形式存储记录。数据块存储在*缓冲区*中。

您可以通过编辑 Fluentd Daemonset 中的环境变量来调整集群中的文件缓冲：

注意

修改 Fluentd Daemonset 中的 **FILE_BUFFER_LIMIT** 或 **BUFFER_SIZE_LIMIT** 参数，您需要把集群日志系统设置为未管理（unmanaged）状态。处于非受管状态的 Operator 不被正式支持，集群管理员需要完全掌控各个组件的配置和升级。

- **BUFFER_SIZE_LIMIT**。此参数决定 Fluentd 创建新块之前每个块文件的最大大小。默认值为 **8M**。此参数设置 Fluentd **chunk_limit_size** 变量。
  高 **BUFFER_SIZE_LIMIT** 可以使每个块文件收集更多记录。但较大的记录需要更长的时间才会被发送到日志存储中。

- **FILE_BUFFER_LIMIT**。这个参数决定每个日志输出的文件缓冲区大小。此值只是基于调度 Fluentd pod 的节点中的可用空间的请求。OpenShift Container Platform 不允许 Fluentd 超过节点容量。默认值为 **256Mi**。
  高 **FILE_BUFFER_LIMIT** 意味着基于输出数量的更高的 **BUFFER_QUEUE_LIMIT**。但是，如果节点的空间面临压力，Fluentd 可能会失败。

  默认情况下，如果所有日志都发送到单个资源，则 **number_of_outputs** 为 **1**，否则每多一个资源就会递增 **1**。如果您使用 Log Forwarding API、Fluentd **Forward** 协议或 syslog 协议将日志转发到外部位置，则可能会有多个输出。

  持久性卷大小必须大于 **FILE_BUFFER_LIMIT** 与输出相乘的结果。

- **BUFFER_QUEUE_LIMIT**。这个参数是允许的最大缓冲块数量。**BUFFER_QUEUE_LIMIT** 参数无法直接调整。OpenShift Container Platform 根据日志记录输出数量、块大小以及可用文件系统空间来计算这个值。默认是 **32** 块。要修改 **BUFFER_QUEUE_LIMIT**，需要修改 **FILE_BUFFER_LIMIT** 的值。**BUFFER_QUEUE_LIMIT** 参数设置 Fluentd **queue_limit_length** 参数。
  OpenShift Container Platform 计算 **BUFFER_QUEUE_LIMIT** 的公式是 **(FILE_BUFFER_LIMIT / (number_of_outputs * BUFFER_SIZE_LIMIT))**。

  使用默认设置的值，**BUFFER_QUEUE_LIMIT** 的值是 32：

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

OpenShift Container Platform 使用 Fluentd **file** 缓冲插件来配置存储块的方式。您可以使用以下命令查看缓冲区文件的位置：

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

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

(1) Fluentd **file** 缓冲插件。不要更改这个值。

(2) 保存缓冲块的路径。

先决条件

- 将集群日志记录设置为非受管状态。处于非受管状态的 Operator 不被正式支持，集群管理员需要完全掌控各个组件的配置和升级。

流程

配置缓冲区块限制：

1. 编辑 **fluentd** Daemonset 中的以下任一参数。

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

(1) 指定每个输出的 Fluentd 文件缓冲大小。

(2) 指定每个 Fluentd 缓冲块的最大大小。

7.7.4. 使用环境变量配置日志记录收集器

您可以使用环境变量来修改 Fluentd 日志收集器的配置。

如需可用环境变量的列表，请参见 Github 中的 Fluentd README。

前提条件

- 将集群日志记录设置为非受管状态。处于非受管状态的 Operator 不被正式支持，集群管理员需要完全掌控各个组件的配置和升级。

流程

根据需要设置任何 Fluentd 环境变量：

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

例如：

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

7.7.5. 关于日志记录收集器警报

以下警报由日志记录收集器生成，可以在 Prometheus UI 的 **Alerts** 选项卡上查看。

所有日志记录收集器警报都列在 OpenShift Container Platform Web 控制台的 **Monitoring → Alerts** 页面中。警报处于以下状态之一：

- **Firing**：在超时期限内警报条件为 true。点击在触发警报末尾的 **Options** 菜单，以查看更多信息或使警告静音。
- **Pending**：警报条件当前为 true，但尚未达到超时时间。
- **Not Firing**：当前未触发警报。

表 7.3. Fluentd Prometheus 警报

警报	消息	描述	重要性
FluentdErrorsHigh	**In the last minute, <value> errors reported by fluentd <instance>.**	Fluentd 报告的问题数量超过了指定数量（默认值为 10）。	Critical
FluentdNodeDown	**Prometheus could not scrape fluentd <instance> for more than 10m.**	Fluentd 报告 Prometheus 可能无法抓取特定的 Fluentd 实例。	Critical
FluentdQueueLengthBurst	**In the last minute, fluentd <instance> buffer queue length increased more than 32.Current value is <value>.**	Fluentd 报告自己已不堪重负。	Warning
FluentdQueueLengthIncreasing	**In the last 12h, fluentd <instance> buffer queue length constantly increased more than 1.Current value is <value>.**	Fluentd 报告队列使用方法问题。	Critical

7.8. 收集并存储 KUBERNETES 事件

OpenShift Container Platform 事件路由器是一个 Pod，它监视 Kubernetes 事件，并通过集群日志记录记录它们以收集。您必须手动部署 Event Router。

Event Router 从所有项目收集事件，并将其写入 **STDOUT**。Fluentd 收集这些事件并将其转发到 OpenShift Container Platform Elasticsearch 实例。Elasticsearch 将事件索引到 **infra** 索引。

重要

事件路由器为 Fluentd 增加额外的负载，并可能会影响其他可以被处理的日志消息数量。

7.8.1. 部署和配置事件路由器

使用以下步骤将事件路由器部署到集群中。您应该始终将 Event Router 部署到 **openshift-logging** 项目，以确保其从集群中收集事件。

以下 Template 对象创建事件路由器所需的服务帐户、集群角色和集群角色绑定。模板还会配置和部署 Event Router Pod。您可以使用此模板而无需更改，或更改 Deployment 对象 CPU 和内存请求。

先决条件

- 需要适当的权限，以便能创建服务帐户和更新集群角色绑定。例如，您可以使用具有 **cluster-admin** 角色的用户来运行以下模板。
- 必须安装集群日志记录。

流程

1. 为事件路由器创建模板：

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

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

1. 在 **openshift-logging** 项目中为事件路由器创建一个服务帐户。
2. 创建用于监控集群中事件的 ClusterRole。
3. 创建一个 ClusterRoleBinding 将 ClusterRole 绑定到 ServiceAccount。
4. 在 **openshift-logging** 项目中创建一个 ConfigMap 来生成所需的 **config.json** 文件。
5. 在 **openshift-logging** 项目中创建一个 Deployment 来生成并配置 Event Router Pod。
6. 指定分配给事件路由器 Pod 的最小内存量。默认值为**128Mi**。

7 指定分配给事件路由器 Pod 的最小 CPU 量。默认值为**100m**。

8 指定要在其中安装对象的 **openshift-logging** 项目。

2. 使用以下命令来处理和应用模板：

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

例如：

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

输出示例

```
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. 验证 **openshift-logging** 项目中安装的 Event Router:

   a. 查看新的事件路由器 Pod:

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

   输出示例

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

   b. 查看事件路由器收集的事件：

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

   例如：

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

   输出示例

   ```
   {"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"}}
```

您还可以使用 Elasticsearch **infra** index 创建索引模式来使用 Kibana 来查看事件。

7.9. 使用容忍度来控制集群日志记录 POD 放置

您可以使用污点和容忍度来确保集群日志记录 Pod 在特定节点上运行，并确保其他工作负载不在这些节点上运行。

污点和容忍度是简单的 **key:value** 对。节点上的污点指示节点排斥所有不容许该污点的 Pod。

key 是最长为 253 个字符的任意字符串， **value** 则是最长为 63 个字符的任意字符串。字符串必须以字母或数字开头，并且可以包含字母、数字、连字符、句点和下划线。

具有容忍度的集群日志记录 CR 的示例

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 1
      tolerations: 1
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
      resources:
        limits:
          memory: 8Gi
        requests:
          cpu: 100m
          memory: 1Gi
      storage: {}
      redundancyPolicy: "ZeroRedundancy"
  visualization:
    type: "kibana"
    kibana:
      tolerations: 2
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
      resources:
        limits:
          memory: 2Gi
        requests:
          cpu: 100m
```

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

1 此容忍度添加到 Elasticsearch Pod。

2 此容忍度添加到 Kibana Pod。

3 此容忍度添加到 Curator Pod。

4 此容忍度添加到日志记录收集器 Pod。

7.9.1. 使用容忍度来控制 Elasticsearch Pod 放置

您可以通过在 Pod 上使用容忍度来控制 Elasticsearch Pod 在哪些节点上运行，并防止其他工作负载使用这些节点。

您可以通过集群日志记录自定义资源 (CR) 将容忍度应用到 Elasticsearch Pod，并通过节点规格将污点应用到节点。节点上的污点是一个 **key:value** 对，它指示节点排斥所有不容许该污点的 Pod。通过使用不在其他 Pod 上的特定 **key:value** 对，可以确保仅 Elasticseach Pod 能够在该节点上运行。

默认情况下，Elasticsearch Pod 具有以下容忍度：

```
tolerations:
- effect: "NoExecute"
```

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

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 使用以下命令，将污点添加到要在其上调度集群日志记录 Pod 的节点：

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

例如：

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

本例在 **node1** 上放置一个键为 **elasticsearch** 且值为 **node** 的污点，污点效果是 **NoExecute**。具有 **NoExecute** 效果的节点仅调度与污点匹配的 Pod，并删除不匹配的现有 Pod。

2. 编辑集群日志记录自定义资源 (CR) 的 **logstore** 部分，以配置 Elasticsearch Pod 的容忍度：

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

1. 指定添加到节点的键。
2. 指定 **Exists** operator 需要节点上有一个带有键为 **elasticsearch** 的污点。
3. 指定 **NoExecute** 效果。
4. （可选）指定 **tolerationSeconds** 参数，以设置 Pod 在被逐出前可以保持绑定到节点的时长。

此容忍度与 **oc adm taint** 命令创建的污点匹配。具有此容忍度的 Pod 可以调度到 **node1** 上。

7.9.2. 使用容忍度（toleration）来控制 Kibana Pod 放置

您可以通过在 Pod 上使用容限（toleration）来控制 Kibana Pod 在哪些节点上运行，并防止其他工作负载使用这些节点。

您可以通过集群日志记录自定义资源 (CR) 将容限应用到 Kibana Pod，并通过节点规格将污点（taint）应用到节点。节点上的污点是一个 **key:value** 对，它指示节点排斥所有不容许该污点的 Pod。通过使用没有在其他 Pod 上使用的特定 **key:value** 对，可以确保仅 Kibana Pod 能够在该节点上运行。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 使用以下命令，将污点添加到要在其上调度 Kibana Pod 的节点：

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

例如：

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

本例在 **node1** 上放置一个键为 **kibana** 且值为 **node** 的污点，污点效果是 **NoExecute**。您必须使用 **NoExecute** 污点设置。**NoExecute** 仅调度与污点匹配的 Pod，并删除不匹配的现有 Pod。

2. 编辑集群日志记录自定义资源 (CR) 的 **visualization** 部分，以配置 Kibana Pod 的容忍度：

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

(1) 指定添加到节点的键。

(2) 指定 **Exists** 运算符，以要求匹配 **key/value/effect** 参数。

(3) 指定 **NoExecute** 效果。

(4) （可选）指定 **tolerationSeconds** 参数，以设置 Pod 在被逐出前可以保持绑定到节点的时长。

此容忍度与 **oc adm taint** 命令创建的污点匹配。具有此容限的 Pod 能够调度到 **node1** 上。

7.9.3. 使用容忍度来控制 Curator Pod 放置

您可以通过在 Pod 上使用容忍度来控制 Curator Pod 在哪些节点上运行，并防止其他工作负载使用这些节点。

您可以通过集群日志记录自定义资源 (CR) 将容忍度应用到 Curator Pod，并通过节点规格将污点应用到节点。节点上的污点是一个 **key:value** 对，它指示节点排斥所有不容许该污点的 Pod。通过使用没有在其他 Pod 上使用的特定 **key:value** 对，可以确保仅 Curator Pod 能够在该节点上运行。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 使用以下命令，将污点添加到要在其上调度 Curator Pod 的节点：

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

例如：

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

本例在 **node1** 上放置一个键为 **curator** 且值为 **node** 的污点，污点效果是 **NoExecute**。您必须使用 **NoExecute** 污点设置。**NoExecute** 仅调度与污点匹配的 Pod，并删除不匹配的现有 Pod。

2. 编辑集群日志记录自定义资源 (CR) 的 **curation** 部分，以配置 Curator Pod 的容忍度：

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

(1) 指定添加到节点的键。

(2) 指定 **Exists** 运算符，以要求匹配 **key/value/effect** 参数。

(3) 指定 **NoExecute** 效果。

(4) （可选）指定 **tolerationSeconds** 参数，以设置 Pod 在被逐出前可以保持绑定到节点的时长。

此容忍度与 **oc adm taint** 命令创建的污点匹配。具有此容限的 Pod 能够调度到 **node1** 上。

7.9.4. 使用容忍度来控制日志收集器 Pod 放置

您可以通过在 Pod 上使用容忍度来确保日志记录收集器 Pod 在哪些节点上运行，并防止其他工作负载使用这些节点。

您可以通过集群日志记录自定义资源 (CR) 将容忍度应用到日志记录收集器 Pod，并通过节点规格将污点应用到节点。您可以使用污点和容限来确保 Pod 不会因为内存和 CPU 问题而被驱除。

默认情况下，日志记录收集器 Pod 具有以下容忍度：

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

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 使用以下命令，将污点添加到要在其上调度日志记录收集器 Pod 的节点：

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

例如：

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

本例在 **node1** 上放置一个键为 **collector** 且值为 **node** 的污点，污点效果是 **NoExecute**。您必须使用 **NoExecute** 污点设置。**NoExecute** 仅调度与污点匹配的 Pod，并删除不匹配的现有 Pod。

2. 编辑集群日志记录自定义资源 (CR) 的 **collection** 部分，以配置日志记录收集器 Pod 的容忍度：

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

1. 指定添加到节点的键。
2. 指定 **Exists** 运算符，以要求匹配 **key/value/effect** 参数。
3. 指定 **NoExecute** 效果。
4. （可选）指定 **tolerationSeconds** 参数，以设置 Pod 在被逐出前可以保持绑定到节点的时长。

此容忍度与 **oc adm taint** 命令创建的污点匹配。具有此容限的 Pod 能够调度到 **node1** 上。

7.9.5. 其他资源

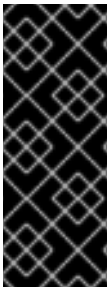
如需有关污点和容忍度的更多信息，请参见使用节点污点控制 Pod 位置。

7.10. 将日志转发到第三方系统

默认情况下，OpenShift Container Platform 集群日志将日志发送到集群日志记录自定义资源中定义的默认内部 Elasticsearch 日志存储。

- **使用 Fluentd 转发协议发送日志**。您可以创建一个 Configmap，使用 Fluentd 转发 协议 将日志安全发送到可以接受 Fluent 转发 协议的外部日志聚合器。
- **使用 syslog 发送日志**。您可以创建一个 Configmap 来使用 syslog 协议 将日志发送到外部 syslog (RFC 3164) 服务器。

另外，您还可以使用 Log Forwarding API（当前为技术预览）。Log Forwarding API 比 Fluentd 协议和 syslog 更容易配置，可以提供将日志发送到内部 Elasticsearch 日志存储和外部 Fluentd 日志聚合解决方案的配置。



重要

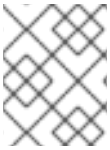
Log Forwarding API 只是一个技术预览功能。技术预览功能不被红帽产品服务等级协议 (SLA) 支持，且可能在功能方面有缺陷。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参阅 https://access.redhat.com/support/offerings/techpreview/。

使用 ConfigMap 转发日志的方法已弃用，并将在以后的版本中由 Log Forwarding API 替换。

7.10.1. 使用 Fluentd 转发协议转发日志

您可以使用 Fluentd 的**转发（forward）**协议，将 Fluentd 配置为将其日志的副本发送到外部日志聚合器，而不是发送到默认的 Elasticsearch 存储。在 OpenShift Container Platform 集群中，您可以使用 Fluentd **转发** 协议将日志发送到配置为接受该协议的服务器。您需要配置外部日志聚合器，以接收 OpenShift Container Platform 中的日志。



注意

此转发日志的方法在 OpenShift Container Platform 中已弃用，并将在以后的版本中由 Log Forwarding API 替代。

要将 OpenShift Container Platform 配置为使用 Fluentd 转发协议来发送日志，在 **openshift-logging** 命名空间内创建一个名为 **secure-forward** 的 ConfigMap 来指向一个外部的日志聚合系统。

重要

从 OpenShift Container Platform 4.3 开始，使用 Fluentd 转发协议的过程发生了变化。现在，需要创建一个 ConfigMap，如下所述。

另外，您还可以将配置所需的证书添加到名为 **secure-forward** 的 secret 中，该 secret 将被挂载到 Fluentd Pod 中。

secure-forward.conf 示例

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

  tls_cert_path '/etc/ocp-forward/ca-bundle.crt'
  <buffer>
    @type file
    path '/var/lib/fluentd/secureforwardlegacy'
    queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '1024' }"
    chunk_limit_size "#{ENV['BUFFER_SIZE_LIMIT'] || '1m' }"
    flush_interval "#{ENV['FORWARD_FLUSH_INTERVAL'] || '5s'}"
    flush_at_shutdown "#{ENV['FLUSH_AT_SHUTDOWN'] || 'false'}"
    flush_thread_count "#{ENV['FLUSH_THREAD_COUNT'] || 2}"
```

```
    retry_max_interval "#{ENV['FORWARD_RETRY_WAIT'] || '300'}"
    retry_forever true
    # the systemd journald 0.0.8 input plugin will just throw away records if the buffer
    # queue limit is hit - 'block' will halt further reads and keep retrying to flush the
    # buffer to the remote - default is 'exception' because in_tail handles that case
    overflow_action "#{ENV['BUFFER_QUEUE_FULL_ACTION'] || 'exception'}"
  </buffer>
  <server>
    host fluent-receiver.openshift-logging.svc  # or IP
    port 24224
  </server>
</store>
```

基于配置的 secure-forward ConfigMap 示例

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

流程

配置 OpenShift Container Platform，以使用 Fluentd 转发协议转发日志：

1. 为 **forward** 参数创建一个名为 **secure-forward.conf** 的配置文件：

   a. 配置 secret 和 TLS 信息：

```
<store>
 @type forward

 self_hostname ${hostname} 1
 shared_key <SECRET_STRING> 2

 transport tls 3

 tls_verify_hostname true 4
 tls_cert_path <path_to_file> 5
```

   1. 指定自动生成的证书通用名称 (CN) 的默认值。
   2. 输入在节点间共享的密钥
   3. 指定 **tls** 启用 TLS 验证。
   4. 设置为 **true** 以验证服务器认证主机名。设置为 **false** 以忽略服务器 cert 主机名。
   5. 指定到私有 CA 证书文件的路径 **/etc/ocp-forward/ca_cert.pem**。

   要使用 mTLS，请参阅 Fluentd 文档 来获取有关客户端证书和密钥参数及其他设置的信息。

   b. 配置外部 Fluentd 服务器的名称、主机和端口。

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

   1. 可选：为这个服务器输入一个名称。
   2. 指定服务器的主机名或 IP。
   3. 指定服务器的主机标识。
   4. 指定服务器的端口。
   5. （可选）添加额外的服务器。如果您指定了两个或者两个以上的服务器，**forward** 会以轮循（round-robin）顺序使用这些服务器节点。

例如：

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

2. 在 **openshift-logging** 命名空间中创建一个名为 **secure-forward** 的 ConfigMap：

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

3. 可选：导入服务器所需的任何 secret：

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

例如：

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

4. 刷新 **fluentd** Pod 以应用 **secure-forward** secret 和 **secure-forward** ConfigMap：

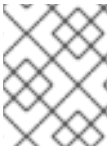
```
$ oc delete pod --selector logging-infra=fluentd
```

5. 配置外部的日志聚会系统以安全地接受来自 OpenShift Container Platform 的信息。

7.10.2. 使用 syslog 协议转发日志

您可以使用 **syslog** 协议将日志副本发送到外部 syslog 服务器，而不是默认的 Elasticsearch 日志存储。注意以下有关 **syslog** 协议的信息：

- 使用 syslog 协议 (RFC 3164)，而不是 RFC 5424
- 不支持 TLS，因此不提供安全功能
- 不提供 Kubernetes 元数据、systemd 数据或其他元数据



注意

此转发日志的方法在 OpenShift Container Platform 中已弃用，并将在以后的版本中由 Log Forwarding API 替代。

syslog 协议有两个版本：

- **out_syslog**：非缓冲的实现（通过 UDP 进行沟通）不会缓冲数据并立即写入结果。
- **out_syslog**_buffered：通过 TCP 进行沟通的缓冲实现会将数据缓冲到块中。

要配置日志转发来使用 **syslog** 协议，创建名为 **syslog.conf** 的配置文件，并提供转发日志所需的信息。然后，使用该文件在 **openshift-logging** 命名空间中创建名为 **syslog** 的 ConfigMap，OpenShift Container Platform 在转发日志时使用该文件。您需要配置 syslog 服务器以接收 OpenShift Container Platform 的日志。

重要

从 OpenShift Container Platform 4.3 开始，使用 **syslog** 协议的过程发生了变化。现在，需要创建一个 ConfigMap，如下所述。

您可以通过在配置文件中指定单独的**<store>** 小节来把日志转发到多个 syslog 服务器。

syslog.conf示例

```
<store>
@type syslog_buffered 1
remote_syslog rsyslogserver.openshift-logging.svc.cluster.local 2
port 514 3
hostname fluentd-4nzfz 4
remove_tag_prefix tag 5
tag_key ident,systemd.u.SYSLOG_IDENTIFIER 6
facility local0 7
severity info 8
use_record true 9
payload_key message 10
</store>
```

1. **syslog** 协议：**syslog** 或 **syslog_buffered**.
2. Syslog 服务器的完全限定域名 (FQDN) 或 IP 地址。
3. 要连接的端口号。默认值为 **514**。
4. Syslog 服务器的名称。
5. 从标签中删除前缀，默认为 ''（空白）。默认为false。
6. 设置 syslog 键的字段。
7. Syslog 日志工具或源。
8. Syslog 日志严重性。
9. 决定是否使用记录中的严重性和工具（如果可用）。
10. 设置 syslog 消息有效负载的键值。默认为false。

基于示例 syslog.conf 的示例 syslog ConfigMap

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

流程

配置 OpenShift Container Platform 使用 **syslog** 协议转发日志：

1. 创建名为 **syslog.conf** 的配置文件，该文件在 **<store>** 部分中包括以下参数 ：

   a. 指定 **syslog** 协议类型：

   ```
   @type syslog_buffered (1)
   ```

   (1) 指定要使用的协议，可以是: **syslog** 或 **syslog_buffered**。

   b. 配置外部 Fluentd 服务器的名称、主机和端口。

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

   (1) 指定 syslog 服务器的 FQDN 或 IP 地址。

   (2) 指定接收方的端口。

   (3) 为这个 syslog 服务器指定名称。

   例如：

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

   c. 根据需要配置其他 syslog 变量：

   ```
   remove_tag_prefix (1)
   ```

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

1. 添加此参数从 syslog 前缀中删除 **tag** 字段。
2. 指定设置 syslog 键的字段。
3. 指定 syslog 日志工具或源。如需值，请参阅 RTF 3164。
4. 指定 syslog 日志的严重性。如需值，请参阅链接:RTF 3164。
5. **如果可用，请指定 true** 来使用记录中的严重性和工具。如果为 **true**$则输出内容中包含 **container _name、namespace _name 和 pod_name** 。
6. 指定设置 syslog 消息有效负载的键。默认为false。

例如：

```
facility local0
severity info
```

其响应类似于如下：

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

2. 在 **openshift-logging** 命名空间中创建一个名为 **secure-forward** 的 ConfigMap：

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

Cluster Logging Operator 会重新部署 Fluentd Pod。如果 Pod 没有重新部署，您可以删除 Fluentd Pod 来强制重新部署。

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

7.10.3. 使用 Log Forwarding API 转发日志

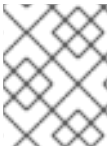
通过 Log Forwarding API，管理员可以配置自定义管道，将容器和节点日志发送到集群内部或外部的特定端点。您可以根据类型，将日志发送到 OpenShift Container Platform 内部的 Elasticsearch 日志存储，和不受 OpenShift Container Platform 集群日志记录管理的系统，如现有的日志记录服务、外部 Elasticsearch 集群、外部日志聚合解决方案或安全信息和事件管理 (SIEM) 系统。

重要

请注意，Log Fowarding API 目前还只是一个技术预览。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关详细信息，请参阅红帽技术预览功能支持范围 。

您可以向不同的系统发送不同类型的日志，以便能够控制机构中的不同人可以访问不同日志类别。可选的 TLS 支持确保了您可以使用您的机构所需的安全通讯方式发送日志。

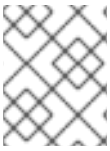


注意

使用 Log Fowarding API 是可选的。如果您只希望将日志转发到内部 OpenShift Container Platform Elasticsearch 实例，请不要配置 Log Forwarding API。

7.10.3.1. 了解 Log Forwarding API

使用 Log Forwarding API 转发集群日志需要结合 *outputs* 和 *pipelines*，将日志发送到 OpenShift Container Platform 集群内部和外部的特定端点。



注意

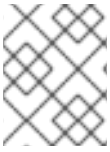
如果只使用默认的内部 OpenShift Container Platform Elasticsearch 存储，不要配置 Log Forwarding 功能。

默认情况下，Cluster Logging Operator 将日志发送到默认的内部 Elasticsearch 存储（由 Cluster Logging Custom Resource 定义）。要使用 Log Forwarding 功能，您需要创建一个自定义 **logforwarding** 配置文件，将日志发送到您指定的目的地。

*输出（output）*是日志数据的目的地，*管道（pipeline）*定义了从一个源到一个或多个输出的简单路由。

输出可以是：

- **elasticsearch** 将日志转发到外部 Elasticsearch v5.x 集群，和/或内部 OpenShift Container Platform Elasticsearch 实例。
- **forward** 转发日志到外部日志聚合解决方案。这个选项使用 Fluentd 转发协议。



注意

如果启用了使用 CIDR 注解的集群范围代理，端点必须是服务器名称或 FQDN，而不是 IP 地址。

*管道（pipeline）*将数据源与输出相关联。数据源是以下之一：

- **logs.app** - 由集群中运行的用户应用程序生成的容器日志（基础架构容器应用程序除外）。
- **logs.infra** - 在集群和 OpenShift Container Platform 节点上运行的基础架构组件生成的日志，如 journal 日志。基础架构组件是在 **openshift***、**kube*** 或 **default** 项目中运行的 pod。
- **logs.audit** - 由节点审计系统 (auditd) 生成的日志，这些日志保存在 **/var/log/audit/audit.log** 文件中，以及 Kubernetes apiserver 和 OpenShift apiserver 的审计日志。

请注意：

- 内部 OpenShift Container Platform Elasticsearch 实例不会为审计日志提供安全存储。您需要自己确保转发审计日志的系统符合您所在机构及政府的相关要求，并具有适当的安全性。OpenShift Container Platform 集群日志记录本身并不会遵循这些规范。
- 通过使用 secret，输出支持 TLS 通讯。secret 必须具有以下密钥: **tls.crt**、**tls.key** 和 **ca-Bundler.crt**，指向它们所代表的相应证书。在使用安全的方式进行转发时，secret 必须具有密钥 **shared_key**。
- 您需要负责创建和维护外部目的地可能需要的额外配置，如密钥和 secret、服务帐户、端口打开或全局代理服务器配置。

以下示例创建了三个输出：

- 内部 OpenShift Container Platform Elasticsearch 实例，
- 无安全保护的外部管理的 Elasticsearch 实例，
- 使用转发协议的具有安全保护的外部日志聚合程序。

三个管道发送：

- 应用程序日志记录发送到内部 OpenShift Container Platform Elasticsearch，
- 基础架构日志发送到外部 Elasticsearch 实例，
- 通过 **forward** 协议将审计日志记录到安全设备。

日志转发输出和管道示例

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

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

1. 日志转发 CR 的名称必须是 **instance**。
2. 禁用默认的日志转发行为的参数。
3. 输出配置。
4. 描述输出的名称。
5. 输出的类型可以是 **elasticsearch** 或 **forward**。
6. 输入端点，可以是服务器名称，也可以是 FQDN 或 IP 地址。如果启用了使用 CIDR 注解的集群范围代理，端点必须是服务器名称或 FQDN，而不是 IP 地址。对于内部 OpenShift Container Platform Elasticsearch 实例，指定 **elasticsearch.openshift-logging.svc:9200**。
7. TLS 通信端点所需的 secret 的名称（可选）。secret 必须存在于 **openshift-logging** 项目中。
8. 如果端点不使用 secret（可选设置），则会造成不安全的通信。
9. 管道配置。
10. 描述管道的名称。
11. 源类型：**logs.app**、**logs.infra** 或 **logs.audit**。
12. 在 CR 中配置的一个或多个输出的名称。

7.10.3.2. 启用 Log Forwarding API

您必须先启用 Log Forwarding API，然后才能使用 API 转发日志。

流程

启用 Log Forwarding API：

1. 在 **openshift-logging** 项目中编辑集群日志记录自定义资源 (CR)：

```
$ oc edit ClusterLogging instance
```

2. 添加 **clusterlogging.openshift.io/logforwardingtechpreview** 注解并设置为 **enabled**：

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

```
metadata:
  annotations:
    clusterlogging.openshift.io/logforwardingtechpreview: enabled (1)
  name: "instance"
  namespace: "openshift-logging"
spec:

...

  collection: (2)
    logs:
      type: "fluentd"
      fluentd: {}
```

1. 启用和禁用 Log Forwarding API。设置为 **enabled** 来使用日志转发功能。如果只使用 OpenShift Container Platform Elasticsearch 实例，则将其设置为 disabled 或不添加注解。

2. 在 Cluster Logging CR 的 **spec.collection** 部分中必须定义使用 Fluentd。

7.10.3.3. 使用 Log Forwarding API 配置日志转发

要配置日志转发，请编辑集群日志记录自定义资源 (CR) 来添加 **clusterlogging.openshift.io/logforwardingtechpreview: enabled** 注解并创建一个日志转发自定义资源来指定输出、管道并启用日志转发。

如果启用日志转发，您应该为三种源类型都定义一个管道：**logs.app**、**logs.infra** 和 **logs.audit**。来自任何未定义源类型的日志将会被丢弃。例如：如果您为 **logs.app** 和 **log-audit** 类型指定了管道，但没有为 **logs.infra** 类型指定管道，则 **logs.infra** 的日志会被丢弃。

流程

使用 API 进行日志转发：

1. 创建一个类似以下示例的 Log Forwarding CR YAML 文件：

```
apiVersion: "logging.openshift.io/v1alpha1"
kind: "LogForwarding"
metadata:
  name: instance (1)
  namespace: openshift-logging (2)
spec:
  disableDefaultForwarding: true (3)
  outputs: (4)
   - name: elasticsearch
     type: "elasticsearch"
     endpoint: elasticsearch.openshift-logging.svc:9200
     secret:
       name: elasticsearch
   - name: elasticsearch-insecure
     type: "elasticsearch"
     endpoint: elasticsearch-insecure.svc.messaging.cluster.local
     insecure: true
   - name: secureforward-offcluster
     type: "forward"
```

```
    endpoint: https://secureforward.offcluster.com:24224
    secret:
      name: secureforward
  pipelines: 5
  - name: container-logs
    inputSource: logs.app
    outputRefs:
    - elasticsearch
    - secureforward-offcluster
  - name: infra-logs
    inputSource: logs.infra
    outputRefs:
    - elasticsearch-insecure
  - name: audit-logs
    inputSource: logs.audit
    outputRefs:
    - secureforward-offcluster
```

1 日志转发 CR 的名称必须是 **instance**。

2 日志转发 CR 的命名空间必须是 **openshift-logging**。

3 设置为 **true** 可禁用默认的日志转发行为。

4 添加一个或多个端点：

- 指定输出的类型，可以是 **elasticseach** 或 **forward**。
- 输入输出的名称。
- 输入端点，可以是服务器名称，也可以是 FQDN 或 IP 地址。如果启用了使用 CIDR 注解的集群范围代理，端点必须是服务器名称或 FQDN，而不是 IP 地址。对于内部 OpenShift Container Platform Elasticsearch 实例，指定 **elasticsearch.openshift-logging.svc:9200**。
- 可选：为 TLS 通信输入端点所需的 secret 名称。secret 必须存在于 **openshift-logging** 项目中。
- 如果端点不使用 secret，则指定 **insecure: true**，这会造成不安全的通信。

5 添加一个或多个管道：

- 为管道输入一个名称
- 指定源类型: **logs.app**、**logs.infra**或 **logs.audit**。
- 指定在 CR 中配置的一个或多个输出的名称。

注意

如果设置了 **disableDefaultForwarding: true**，必须为所有三种类型的日志(应用程序、基础架构和审核）配置管道和输出。如果您没有为其中的一个日志类型指定管道和输出，则相关日志将不会被存储并将丢失。

2. 创建 CR 对象。

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

7.10.3.3.1. 日志转发自定义资源示例

典型的日志转发配置类似以下示例。

以下日志转发自定义资源会将所有日志发送到安全的外部 Elasticsearch 日志存储：

转发到 Elasticsearch 日志存储的自定义资源示例

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

以下日志转发自定义资源使用 Fluentd **forward** 协议将所有日志发送到安全的 Fluentd 实例。

使用 forward 协议的自定义资源样本

```
apiVersion: logging.openshift.io/v1alpha1
kind: LogForwarding
metadata:
  name: instance
  namespace: openshift-logging
spec:
  disableDefaultForwarding: true
  outputs:
    - name: fluentd-created-by-user
      type: forward
      endpoint: 'fluentdserver.openshift-logging.svc:24224'
      secret:
        name: fluentdserver
```

```
pipelines:
 - name: app-pipeline
   inputSource: logs.app
   outputRefs:
     - fluentd-created-by-user
 - name: infra-pipeline
   inputSource: logs.infra
   outputRefs:
     - fluentd-created-by-user
 - name: clo-default-audit-pipeline
   inputSource: logs.audit
   outputRefs:
     - fluentd-created-by-user
```

7.10.3.4. 禁用 Log Forwarding API

要禁用 Log Forwarding API，并停止将日志转发到特定端点，从 Cluster Logging CR 中删除 **metadata.annotations.clusterlogging.openshift.io/logforwardingtechpreview:enabled** 参数，并删除 Log Forwarding CR。容器和节点日志将转发到内部 OpenShift Container Platform Elasticsearch 实例。

注意

设置 **disableDefaultForwarding=false** 可防止集群日志记录发送到指定的端点**以及**默认的 OpenShift Container Platform Elasticsearch 实例。

流程

禁用 Log Forwarding API:

1. 在 **openshift-logging** 项目中编辑集群日志记录自定义资源 (CR)：

```
$ oc edit ClusterLogging instance
```

2. 删除 **clusterlogging.openshift.io/logforwardingtechpreview** 注解：

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

(1) 删除此注解。

3. 创建日志转发自定义资源：

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

7.11. 配置 SYSTEMD-JOURNALD 和 FLUENTD

Fluentd 需要从日志 (journal) 中读取数据。因为日志默认设置非常低，它可能无法跟上系统服务的日志记录率，所以日志条目可能会丢失。

我们推荐设置 **RateLimitInterval=1s** 和 **RateLimitBurst=10000** （如有必要甚至更高）以防止日志丢失条目。

7.11.1. 为集群日志记录配置 systemd-journald

随着项目的扩展，默认的日志记录环境可能需要进行一些调整。

例如，如果有缺少日志数据的情况，则可能需要提高 journald 的速率限制。您可以调整在指定时间段内保留的消息数量，以确保集群日志记录在不丢弃日志的情况下不会使用过量资源。

您还可以确定是否压缩日志、日志需要保留的时间、如何存储日志，以及其他设置。

流程

1. 使用所需设置创建 **journald.conf** 文件：

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

(1) 指定是否要在将日志写入文件系统前压缩日志。指定 **yes** 来压缩消息，或指定 **no** 不压缩信息。默认为 **yes**。

(2) 配置是否转发日志信息。每个默认值为 **no**。指定：

- **ForwardToConsole** 将日志转发到系统控制台。
- **ForwardToKsmg** 将日志转发到内核日志缓冲。
- **ForwardToSyslog** 将日志转发到 syslog 守护进程。
- **ForwardToWall** 将信息作为墙信息转发给所有登录的用户。

(3) 指定存储日志条目的最长时间。输入秒数。或包括一个单位："year"、"month"、"week"、"day"、"h" 或 "m"。输入 **0** 来禁用。默认值为 **1month**。

(4) 配置速率限制。在 **RateLimitIntervalSec** 定义的时间段内，如果接收的日志数量超过了 **RateLimitBurst** 指定的值，则以后的所有信息都会被丢弃，直到该时间段结束。建议您设置 **RateLimitInterval=1s** 和 **RateLimitBurst=10000**，它们是默认值。

(5) 指定日志的存储方式。默认为 **persistent**：

- **volatile** 在 **/var/log/journal/** 中存储内存中的日志数据。

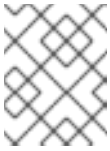
- **persistent** 把日志保存到磁盘的 **/var/log/journal/**。如果这个目录步存在，systemd 将会创建这个目录。
- **auto** 如果目录存在，把日志保存在 **/var/log/journal/** 中。如果不存在，systemd 会临时将日志保存在 **/run/systemd/journal** 中。
- **none** 不存储日志。systemd 丢弃所有日志。

6 指定在将 **ERR**, **WARNING**, **NOTICE**, **INFO** 和 **DEBUG** 日志同步到磁盘上前等待的超时时间。systemd 在接收到 **CRIT**, **ALERT** 或 **EMERG** 日志后会立即进行同步。默认值为 **1s**。

7 指定日志可以使用的最大值。默认值为 **8g**。

8 指定 systemd 必须保留多少磁盘空间。默认值为 **20%**。

9 指定保存在 **/var/log/journal** 中的独立日志文件的最大大小。默认值为 **10M**。



注意

如果删除速率限制，您可能会看到系统日志记录守护进程的 CPU 使用率增加，因为它需要处理在以前可以被限制掉的信息。

如需了解更多关于 systemd 设置的信息，请参阅 https://www.freedesktop.org/software/systemd/man/journald.conf.html。该页面中列出的默认设置可能不适用于 OpenShift Container Platform。

2. 将 **journal.conf** 文件转换为 base64：

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

3. 为 master 或 worker 创建新的 MachineConfig，并添加 **journal.conf** 参数：例如：

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

1 为 **journal.conf** 文件设置权限 。建议把选项设置为 **0644** 。

2 指定到 base64 编码的 **journal.conf** 文件的路径 。

4. 创建 MachineConfig：

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

控制器会检测新的 MachineConfig，并生成新的 **rendered-worker-<hash>** 版本。

5. 监控新配置在每个节点中的应用状态：

```
$ oc describe machineconfigpool/worker


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

...

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

第 8 章 查看 ELASTICSEARCH 状态

您可以查看 Elasticsearch Operator 的状态，以及多个 Elasticsearch 组件的状态。

8.1. 查看 ELASTICSEARCH 状态

您可以查看 Elasticsearch 集群的状态。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 进入 **openshift-logging** 项目。

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

2. 查看 Elasticsearch 集群状态：

   a. 获取 Elasticsearch 实例的名称：

```
$ oc get Elasticsearch

NAME            AGE
elasticsearch   5h9m
```

   b. 获取 Elasticsearch 状态：

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

   例如：

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

   输出中包含类似于如下的信息：

```
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. 在输出中，集群状态字段显示在 **status** 小节中。

2. Elasticsearch 集群的状态：

   - 活跃的主分片的数量。
   - 活跃分片的数量。
   - 正在初始化的分片的数量。
   - Elasticsearch 数据节点的数量。
   - Elasticsearch 节点的总数。
   - 待处理的任务数量。
   - Elasticsearch 状态：**green**、**red** 或 **yellow**。
   - 未分配分片的数量。

3. 任何状态条件（若存在）。Elasticsearch 集群状态代表了当无法放置容器时来自于调度程序的原因。显示与以下情况有关的所有事件：

   - 容器正在等待 Elasticsearch 和代理容器。
   - 容器因 Elasticsearch 和代理容器而终止。
   - Pod 不可调度。此外还显示适用于多个问题的情况，具体请参阅**情况消息示例**。

4. 集群中的 Elasticsearch 节点，以及 **upgradeStatus**。
5. 集群中的 Elasticsearch 客户端、数据和 master 节点，列在 failed、**notReady** 或 **ready** 状态下。

8.1.1. 情况消息示例

以下是来自 Elasticsearch 实例的 **Status** 部分的一些情况消息的示例。

此状态消息表示节点已超过配置的低水位线，并且没有分片将分配给此节点。

```
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: {}
```

此状态消息表示节点已超过配置的高水位线，并且分片将重新定位到其他节点。

```
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: {}
```

此状态消息表示 CR 中的 Elasticsearch 节点选择器与集群中的任何节点都不匹配：

```
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
```

此状态消息表示 Elasticsearch CR 使用了不存在的 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
```

此状态消息表示 Elasticsearch 集群没有足够的节点来支持 Elasticsearch 冗余策略。

```
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
```

此状态消息表示集群中包含太多 master 节点：

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

8.2. 查看 ELASTICSEARCH 组件状态

您可以查看多个 Elasticsearch 组件的状态。

Elasticsearch 索引

您可以查看 Elasticsearch 索引的状态。

1. 获取 Elasticsearch Pod 的名称：

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

pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. 获取索引的状态：

```
$ oc exec elasticsearch-cdm-1godmszn-1-6f8495-vp4lw -- indices

Defaulting container name to elasticsearch.
Use 'oc describe pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw -n openshift-logging'
to see all of the containers in this pod.
Wed Apr 10 05:42:12 UTC 2019
```

```
health status index                                uuid                   pri rep docs.count
docs.deleted store.size pri.store.size
red    open   .kibana.647a750f1787408bf50088234ec0edd5a6a9b2ac
N7iCbRjSSc2bGhn8Cpc7Jg   2   1
green  open   .operations.2019.04.10               GTewEJEzQjaus9QjvBBnGg   3   1
2176114            0       3929          1956
green  open   .operations.2019.04.11               ausZHoKxTNOoBvv9RlXfrw   3   1
1494624            0       2947          1475
green  open   .kibana                             9Fltn1D0QHSnFMXpphZ--Q   1   1          1
0          0            0
green  open   .searchguard                        chOwDnQlSsqhfSPcot1Yiw   1   1
5          1          0            0
```

Elasticsearch Pod

您可以查看 Elasticsearch Pod 的状态。

1. 获取 Pod 的名称：

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

pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. 获取 Pod 的状态：

```
oc describe pod elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
```

输出中包括以下状态信息：

```
....
Status:             Running

....

Containers:
  elasticsearch:
    Container ID:   cri-o://b7d44e0a9ea486e27f47763f5bb4c39dfd2
    State:          Running
      Started:      Mon, 08 Apr 2019 10:17:56 -0400
    Ready:          True
    Restart Count:  0
    Readiness:  exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
period=5s #success=1 #failure=3

....

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

```
....

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

....

Events:         <none>
```

Elasticsearch 部署配置

您可以查看 Elasticsearch 部署配置的状态。

1. 获取部署配置的名称：

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

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

2. 获取部署配置状态：

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

输出中包括以下状态信息：

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

....

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

....

Events:         <none>
```

Elasticsearch ReplicaSet

您可以查看 Elasticsearch ReplicaSet 的状态。

1. 获取副本集的名称：

```
$ 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. 获取副本集的状态：

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

输出中包括以下状态信息：

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

....

Events:         <none>
```

第 9 章 查看集群日志记录状态

您可以查看 Cluster Logging Operator 的状态以及多个集群日志记录组件的状态。

9.1. 查看 CLUSTER LOGGING OPERATOR 的状态

您可以查看 Cluster Logging Operator 的状态。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 进入 **openshift-logging** 项目。

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

2. 查看集群日志记录状态：

   a. 获取集群日志记录状态：

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

   输出中包含类似于如下的信息：

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

....

status: 1
  collection:
    logs:
      fluentdStatus:
        daemonSet: fluentd 2
        nodes:
          fluentd-2rhqp: ip-10-0-169-13.ec2.internal
          fluentd-6fgjh: ip-10-0-165-244.ec2.internal
          fluentd-6l2ff: ip-10-0-128-218.ec2.internal
          fluentd-54nx5: ip-10-0-139-30.ec2.internal
          fluentd-flpnn: ip-10-0-147-228.ec2.internal
          fluentd-n2frh: ip-10-0-157-45.ec2.internal
        pods:
          failed: []
          notReady: []
          ready:
          - fluentd-2rhqp
          - fluentd-54nx5
          - fluentd-6fgjh
          - fluentd-6l2ff
          - fluentd-flpnn
          - fluentd-n2frh
  curation: 3
```

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

1. 在输出中，集群状态字段显示在 **status** 小节中。
2. Fluentd Pod 的相关信息。
3. Curator Pod 的相关信息。
4. Elasticsearch Pod 的相关信息，包括 Elasticsearch 集群健康状态 **green**、**yellow** 或 **red**

red。

5 Kibana Pod 的相关信息。

9.1.1. 情况消息示例

以下示例是来自集群日志记录实例的 **Status.Nodes** 部分的一些情况消息。

类似于以下内容的状态消息表示节点已超过配置的低水位线，并且没有分片将分配给此节点：

```
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: {}
```

类似于以下内容的状态消息表示节点已超过配置的高水位线，并且分片将重新定位到其他节点：

```
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: {}
```

类似于以下内容的状态消息表示 CR 中的 Elasticsearch 节点选择器与集群中的任何节点都不匹配：

```
  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:
```

类似于以下内容的状态消息表示请求的 PVC 无法绑定到 PV：

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

类似于以下内容的状态消息表示无法调度 Curator Pod，因为节点选择器与任何节点都不匹配：

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

类似于以下内容的状态消息表示无法调度 Fluentd Pod，因为节点选择器与任何节点都不匹配：

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

9.2. 查看集群日志记录组件的状态

您可以查看多个集群日志记录组件的状态。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

1. 进入 **openshift-logging** 项目。

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

2. 查看集群日志记录部署的状态：

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

   输出中包括以下状态信息：

   ```
   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. 查看集群日志记录 ReplicaSet 的状态：

a. 获取 ReplicaSet 的名称：

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

b. 获取 ReplicaSet 的状态：

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

输出中包括以下状态信息：

```
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----
```

第 10 章 使用节点选择器移动集群日志记录资源

您可以使用节点选择器，将 Elasticsearch、Kibana 和 Curator Pod 部署到不同的节点上。

10.1. 移动集群日志记录资源

您可以配置 Cluster Logging Operator，以将任何或所有 Cluster Logging 组件、Elasticsearch、Kibana 和 Curator 的 Pod 部署到不同的节点上。您无法将 Cluster Logging Operator Pod 从其安装位置移走。

例如，您可以因为 CPU、内存和磁盘要求较高而将 Elasticsearch Pod 移到一个单独的节点上。

注意

您应该将 MachineSet 设置为至少使用 6 个副本。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。默认情况下没有安装这些功能。

流程

1. 编辑 **openshift-logging** 项目中的集群日志记录自定义资源：

```
$ oc edit ClusterLogging instance
```

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

....

spec:
  collection:
    logs:
      fluentd:
        resources: null
      type: fluentd
  curation:
    curator:
      nodeSelector: 1
        node-role.kubernetes.io/infra: ''
      resources: null
      schedule: 30 3 * * *
    type: curator
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 2
        node-role.kubernetes.io/infra: ''
      redundancyPolicy: SingleRedundancy
      resources:
        limits:
          cpu: 500m
          memory: 16Gi
        requests:
```

```
        cpu: 500m
        memory: 16Gi
    storage: {}
  type: elasticsearch
managementState: Managed
visualization:
  kibana:
    nodeSelector: 3
      node-role.kubernetes.io/infra: '' 4
    proxy:
      resources: null
    replicas: 1
    resources: null
  type: kibana

....
```

1 2 3 4 添加 **nodeSelector** 参数，并设为适用于您想要移动的组件的值。您可以根据为节点指定的值，按所示格式使用 **nodeSelector** 或使用 **<key>: <value>** 对。

验证步骤

要验证组件是否已移动，您可以使用 **oc get pod -o wide** 命令。

例如：

- 您需要移动来自 **ip-10-0-147-79.us-east-2.compute.internal** 节点上的 Kibana pod：

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

- 您需要将 Kibana Pod 移到 **ip-10-0-139-48.us-east-2.compute.internal** 节点，该节点是一个专用的基础架构节点：

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

请注意，该节点具有 **node-role.kubernetes.io/infra: ''** label:

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

kind: Node
apiVersion: v1
metadata:
```

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

- 要移动 Kibana Pod，请编辑集群日志记录 CR 以添加节点选择器：

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

....

spec:

....

  visualization:
    kibana:
      nodeSelector: 1
        node-role.kubernetes.io/infra: '' 2
      proxy:
        resources: null
      replicas: 1
      resources: null
    type: kibana
```

1 2 添加节点选择器以匹配节点规格中的 label。

- 保存 CR 后，当前 Kibana Pod 将被终止，新的 Pod 会被部署：

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

- 新 pod 位于 **ip-10-0-139-48.us-east-2.compute.internal** 节点上：

```
$ oc get pod kibana-7d85dcffc8-bfpfp -o wide
NAME                     READY   STATUS        RESTARTS   AGE   IP          NODE
NOMINATED NODE   READINESS GATES
```

```
kibana-7d85dcffc8-bfpfp   2/2     Running     0          43s   10.131.0.22   ip-10-0-139-48.us-
east-2.compute.internal   <none>          <none>
```

- 片刻后，原始 Kibana Pod 将被删除。

```
$ oc get pods
NAME                                            READY   STATUS    RESTARTS   AGE
cluster-logging-operator-84d98649c4-zb9g7       1/1     Running   0          30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj   2/2     Running   0          29m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78    2/2     Running   0          29m
fluentd-42dzz                                   1/1     Running   0          29m
fluentd-d74rq                                   1/1     Running   0          29m
fluentd-m5vr9                                   1/1     Running   0          29m
fluentd-nkxl7                                   1/1     Running   0          29m
fluentd-pdvqb                                   1/1     Running   0          29m
fluentd-tflh6                                   1/1     Running   0          29m
kibana-7d85dcffc8-bfpfp                         2/2     Running   0          62s
```

第 11 章 手动滚动部署 ELASTICSEARCH

OpenShift Container Platform 支持 Elasticsearch 集群滚动重启。滚动重启可以在不需要 Elasticsearch 集群下线的情况下，将适用的更改应用到 Elasticsearch 集群（这需要已配置了三个 master）。Elasticsearch 集群保持联机和可运行状态，节点则逐一脱机。

11.1. 执行 ELASTICSEARCH 集群滚动重启

在更改 **elasticsearch** ConfigMap 或任何 **elasticsearch-*** 部署配置时，执行滚动重启。

此外，如果运行 Elasticsearch Pod 的节点需要重启，则建议滚动重启。

前提条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

执行集群滚动重启：

1. 进入 **openshift-logging** 项目：

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

2. 使用以下命令，从 Elasticsearch 提取 CA 证书并写入到 ***admin-ca*** 文件：

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

admin-ca
```

3. 执行分片同步刷新，确保在关机之前没有等待写入磁盘的待定操作：

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

例如：

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

4. 使用 OpenShift Container Platform **es_util** 工具防止在有意关闭节点时进行分片平衡：

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

例如：

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query=_cluster/settings?pretty=true -XPUT 'https://localhost:9200/_cluster/settings' -d '{
"transient": { "cluster.routing.allocation.enable" : "none" } }'
```

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

5. 完成后，会在每个部署中都有一个 ES 集群：

   a. 默认情况下，OpenShift Container Platform Elasticsearch 集群会阻止向其节点推出部署。使用以下命令来允许推出部署并允许 Pod 获取更改：

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

   例如：

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

   部署了一个新 Pod。当 Pod 具有就绪的容器后，就能继续进行下一部署。

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

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

   b. 完成后，重置 Pod 以禁止推出部署：

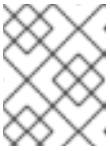
   ```
   $ oc rollout pause deployment/<deployment-name>
   ```

   例如：

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

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

   c. 检查 Elasticsearch 集群是否处于 **green** 状态：

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



注意

如果您对先前命令中使用的 Elasticsearch Pod 执行了推出部署，该 Pod 将不再存在，并且此处需要使用新的 Pod 名称。

例如：

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

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

(1) 在继续操作之前，请确保此参数为 **green**。

6. 如果更改了 Elasticsearch 配置映射，请对每个 Elasticsearch Pod 重复这些步骤。

7. 推出集群的所有部署后，重新启用分片平衡：

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

例如：

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

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

```
    }
  }
}
```

第 12 章 KIBANA 故障排除

将 Kibana 控制台与 OpenShift Container Platform 搭配使用可能导致一些问题。这些问题容易解决，但在出现问题时没有提供有用的错误消息。如果在 OpenShift Container Platform 中部署 Kibana 时遇到任何问题，请检查以下故障排除章节。

12.1. KUBERNETES 登录循环故障排除

Kibana 控制台上的 OAuth2 代理必须与 master 主机的 OAuth2 服务器共享一个 secret。如果两台服务器上的 secret 不相同，则可能导致登录循环，造成您不断重定向到 Kibana 登录页面。

流程

解决此问题的步骤：

1. 运行以下命令以删除当前的 OAuthClient：

```
$ oc delete oauthclient/kibana-proxy
```

12.2. 排查查看 KIBANA 控制台时的 KUBERNETES 密码错误

尝试访问 Kibana 控制台时，您可能会收到浏览器错误：

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

这可能是由 OAuth2 客户端与服务器之间不匹配造成的。客户端的返回地址必须在白名单中，这样服务器才能在登录后安全地重定向回此地址。

通过替换 OAuthClient 条目来解决此问题。

流程

替换 OAuthClient 条目：

1. 运行以下命令以删除当前的 OAuthClient：

```
$ oc delete oauthclient/kibana-proxy
```

如果问题仍然存在，请检查您是否通过 OAuth 客户端中列出的 URL 来访问 Kibana。通过转发端口（例如 1443）而非标准的 443 HTTPS 端口访问 URL 可能会造成此问题。您可以通过编辑 OAuth 客户端来调整服务器白名单：

```
$ oc edit oauthclient/kibana-proxy
```

12.3. 排查查看 KIBANA 控制台时的 KUBERNETES 503 错误

如果在查看 Kibana 控制台时收到代理错误，原因可能是以下两个问题之一：

- Kibana 可能无法识别 Pod。如果 Elasticsearch 启动缓慢，则 Kibana 可能会在尝试访问它时超时。检查相关服务是否具有任何端点：

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

如果有任何 Kibana Pod 处于活动状态，则会列出端点。如果没有，请检查 Kibana Pod 和部署的状态。您可能需要缩减部署，然后再次备份。

- 用于访问 Kibana 服务的路由已被屏蔽。如果您在一个项目中执行测试部署，然后在另一项目中进行部署，但没有彻底删除第一个部署，可能会发生这种情况。当多个路由发送到同一目的地时，默认路由器将仅路由到最先创建的路由。检查有问题的路由，以查看是否在多个位置定义了该路由：

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

第 13 章 导出字段

这些字段由日志记录系统导出，可从 Elasticsearch 和 Kibana 搜索。在搜索时，请使用完整的带句点字段名称。例如，对于 Elasticsearch **/_search URL**，若要查找 Kubernetes Pod 名称，请使用 **/_search/q=kubernetes.pod_name:name-of-my-pod**。

以下小节中描述的字段可能不出现在您的日志记录存储中。并非所有这些字段都会出现在每条记录中。这些字段划分为如下类别：

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

13.1. 默认导出的字段

这些默认字段由日志记录系统导出，Elasticsearch 和 Kibana 可从对它们进行搜索。默认字段是顶级字段和 **collectd*** 字段

顶级字段
顶级字段对于每个应用程序都是通用的，并且可能会出现在每条记录中。对于 Elasticsearch 模板，顶级字段在模板的映射部分中填充 **default** 的实际映射。

参数	描述
@timestamp	此 UTC 值标记日志有效负载的创建时间，如果创建时间未知，则标记首次收集日志有效负载的时间。这是日志处理管道尽力确定何时生成日志有效负载的方法。添加 **@** 前缀惯例，可注明字段保留给特定用途。使用 Elasticsearch 时，大多数工具默认查找 **@timestamp**。例如，格式是 2015-01-24 14:06:05.071000。
geoip	这是机器的 geo-ip。
hostname	**hostname** 是生成原始有效负载的实体的完全限定域名 (FQDN)。此字段尝试生成此上下文。有时候，生成它的实体知道其上下文。但在其他时候，实体本身具有受限的命名空间，为收集器或规范化程序所知。
ipaddr4	源服务器的 IPv4 地址，可以是数组。
ipaddr6	源服务器的 IPv6 地址（若有）。

参数	描述
level	由 python 的日志记录模块 rsyslog（severitytext 属性）提供的日志记录级别。**misc/sys/syslog.h** 列出了可能的值，另外还有 **trace** 和 **unknown**。例如，“alert crit debug emerg err info notice trace unknown warning”。注意 **trace** 不在 **syslog.h** 列表中，但许多应用程序都用到它。 .只有在日志记录系统获取了不理解的值时，才应使用 **unknown**。另外，还需要注意它是最高级别。**trace** 可以视为比 **debug** 的级别更高级且产生更详细的信息。**error** 已弃用，请使用 **err**。将 **panic** 转换为 **emerg**。将 **warn** 转换为 **warning**。 通常可以使用 misc/sys/syslog.h 上列出的优先级值来映射 **syslog/journal PRIORITY** 中的数值。 其他日志记录系统的日志级别和优先级应映射到最接近的匹配项。请参阅python 日志记录中的示例。
message	典型的日志条目消息或有效负载。可以从中剥离由收集器或规范化程序提取的元数据，这些元数据采用 UTF-8 编码。
pid	这是日志记录实体（若有）的进程 ID。
service	与日志记录实体（若有）关联的服务的名称。例如，**syslog APP-NAME** 属性映射到 service 字段。
tags	（可选）由 Operator 定义的标签的列表，这些标签由收集器或规范化程序放置在每个日志上。有效负载可以是含有空格分隔字符串令牌的字符串，也可以是字符串令牌的 JSON 列表。
file	文件的可选路径，该文件包含对文件路径的收集器 **TODO** 分析器而言是本地的日志条目。
offset	偏移值可以表示文件中日志行开头的字节数（从零或一算起），或者表示日志行号（从零或一算起），只要这些值在单个日志的上下文中严格单调递增。允许对这些值换行，以表示日志文件的新版本（轮转）。
namespace_name	将这条记录与共享其名称的 **namespace** 关联。这个值不会存储下来，而是用于将记录与适当的 **namespace** 相关联，以进行访问控制和视觉化。通常，这个值在标签中给出，但如果协议不支持发送标签，则可以使用此字段。如果存在此字段，它将覆盖标签（tag）或 **kubernetes.namespace_name** 中指定的**namespace**。
namespace_uuid	与 **namespace_name** 关联的 **uuid**。这个值不会存储下来，而是用于将记录与适当的命名空间相关联，以进行访问控制和视觉化。如果存在此字段，它将覆盖 **kubernetes.namespace_uuid** 给出的 **uuid**。这也将会导致针对此日志记录跳过 Kubernetes 元数据查找。

collectd 字段
以下字段代表命名空间指标元数据。

参数	描述
collectd.interval	类型：浮点数 **collectd** 间隔。
collectd.plugin	类型：字符串 **collectd** 插件。
collectd.plugin_instance	类型：字符串 **collectd** 插件实例。
collectd.type_instance	类型：字符串 **collectd 类型实例。**
collectd.type	类型：字符串 **collectd** 类型。
collectd.dstypes	类型：字符串 **collectd** dstypes。

collectd.processes 字段。
以下字段对应于 **collectd** 进程插件。

参数	描述
collectd.processes.ps_state	类型：整数。**collectd ps_state** 型进程插件。

collectd.processes.ps_disk_ops 字段
collectd ps_disk_ops 型进程插件。

参数	描述
collectd.processes.ps_disk_ops.read	类型：浮点数 **TODO**
collectd.processes.ps_disk_ops.write	类型：浮点数 **TODO**

参数	描述
collectd.processes.ps_vm	类型：整数 **collectd ps_vm** 型进程插件。
collectd.processes.ps_rss	类型：整数 **collectd ps_rss** 型进程插件。
collectd.processes.ps_data	类型：整数 **collectd ps_data** 型进程插件。
collectd.processes.ps_code	类型：整数 **collectd ps_code** 型进程插件。
collectd.processes.ps_stacksize	类型：整数 **collectd ps_stacksize** 型进程插件。

collectd.processes.ps_cputime 字段
collectd ps_cputime 型进程插件。

参数	描述
collectd.processes.ps_cputime.user	类型：浮点数 **TODO**
collectd.processes.ps_cputime.syst	类型：浮点数 **TODO**

collectd.processes.ps_count 字段
collectd ps_count 型进程插件。

参数	描述
collectd.processes.ps_count.processes	类型：整数 **TODO**
collectd.processes.ps_count.threads	类型：整数 **TODO**

collectd.processes.ps_pagefaults 字段

collectd ps_pagefaults 型进程插件。

参数	描述
collectd.processes.ps_pagefaults.majflt	类型：浮点数 **TODO**
collectd.processes.ps_pagefaults.minflt	类型：浮点数 **TODO**

collectd.processes.ps_disk_octets 字段
collectd ps_disk_octets 型进程插件。

参数	描述
collectd.processes.ps_disk_octets.read	类型：浮点数 **TODO**
collectd.processes.ps_disk_octets.write	类型：浮点数 **TODO**
collectd.processes.fork_rate	类型：浮点数 **collectd fork_rate** 型进程插件。

collectd.disk 字段
对应于 **collectd** 磁盘插件。

collectd.disk.disk_merged 字段
collectd disk_merged 型磁盘插件。

参数	描述
collectd.disk.disk_merged.read	类型：浮点数 **TODO**
collectd.disk.disk_merged.write	类型：浮点数 **TODO**

collectd.disk.disk_octets 字段
collectd disk_octets 型磁盘插件。

参数	描述
collectd.disk.disk_octets.read	类型：浮点数 **TODO**
collectd.disk.disk_octets.write	类型：浮点数 **TODO**

collectd.disk.disk_time 字段
collectd disk_time 型磁盘插件。

参数	描述
collectd.disk.disk_time.read	类型：浮点数 **TODO**
collectd.disk.disk_time.write	类型：浮点数 **TODO**

collectd.disk.disk_ops 字段
collectd disk_ops 型磁盘插件。

参数	描述
collectd.disk.disk_ops.read	类型：浮点数 **TODO**
collectd.disk.disk_ops.write	类型：浮点数 **TODO**
collectd.disk.pending_operations	类型：整数 **collectd pending_operations** 型磁盘插件。

collectd.disk.disk_io_time 字段
collectd disk_io_time 型磁盘插件。

参数	描述
collectd.disk.disk_io_time.io_time	类型：浮点数 **TODO**

参数	描述
collectd.disk.disk_io_time .weighted_io_time	类型：浮点数 **TODO**

collectd.interface 字段
对应于 **collectd** 接口插件。

collectd.interface.if_octets 字段
collectd if_octets 型接口插件。

参数	描述
collectd.interface.if_octets.rx	类型：浮点数 **TODO**
collectd.interface.if_octets.tx	类型：浮点数 **TODO**

collectd.interface.if_packets 字段
collectd if_packets 型接口插件。

参数	描述
collectd.interface.if_packets.rx	类型：浮点数 **TODO**
collectd.interface.if_packets.tx	类型：浮点数 **TODO**

collectd.interface.if_errors 字段
collectd if_errors 型接口插件。

参数	描述
collectd.interface.if_errors.rx	类型：浮点数 **TODO**
collectd.interface.if_errors.tx	类型：浮点数 **TODO**

collectd.interface.if_dropped 字段
collectd if_dropped 型接口插件。

参数	描述
collectd.interface.if_dropped.rx	类型：浮点数 **TODO**
collectd.interface.if_dropped.tx	类型：浮点数 **TODO**

collectd.virt 字段
对应于 **collectd** virt 插件。

collectd.virt.if_octets 字段
collectd if_octets 型 virt 插件。

参数	描述
collectd.virt.if_octets.rx	类型：浮点数 **TODO**
collectd.virt.if_octets.tx	类型：浮点数 **TODO**

collectd.virt.if_packets 字段
collectd if_packets 型 virt 插件。

参数	描述
collectd.virt.if_packets.rx	类型：浮点数 **TODO**
collectd.virt.if_packets.tx	类型：浮点数 **TODO**

collectd.virt.if_errors 字段
collectd if_errors 型 virt 插件。

参数	描述

参数	描述
collectd.virt.if_errors.rx	类型：浮点数 **TODO**
collectd.virt.if_errors.tx	类型：浮点数 **TODO**

collectd.virt.if_dropped 字段
collectd if_dropped 型 virt 插件。

参数	描述
collectd.virt.if_dropped.rx	类型：浮点数 **TODO**
collectd.virt.if_dropped.tx	类型：浮点数 **TODO**

collectd.virt.disk_ops 字段
collectd disk_ops 型 virt 插件。

参数	描述
collectd.virt.disk_ops.read	类型：浮点数 **TODO**
collectd.virt.disk_ops.write	类型：浮点数 **TODO**

collectd.virt.disk_octets 字段
collectd disk_octets 型 virt 插件。

参数	描述
collectd.virt.disk_octets.read	类型：浮点数 **TODO**
collectd.virt.disk_octets.write	类型：浮点数 **TODO**

参数	描述
collectd.virt.memory	类型：浮点数 **collectd** 内存型 virt 插件。
collectd.virt.virt_vcpu	类型：浮点数 **collectd virt_vcpu** 型 virt 插件。
collectd.virt.virt_cpu_total	类型：浮点数 **collectd virt_cpu_total** 型 virt 插件。

collectd.CPU 字段
对应于 **collectd** CPU 插件。

参数	描述
collectd.CPU.percent	类型：浮点数 **collectd** 百分比型 CPU 插件。

collectd.df 字段
对应于 **collectd df** 插件。

参数	描述
collectd.df.df_complex	类型：浮点数 **collectd df_complex** 型 **df** 插件。
collectd.df.percent_bytes	类型：浮点数 **collectd percent_bytes** 型 **df** 插件。

collectd.entropy 字段
对应于 **collectd** 熵插件。

参数	描述
collectd.entropy.entropy	类型：整数 **collectd** 熵型熵插件。

collectd.memory 字段
对应于 **collectd** 内存插件。

参数	描述
collectd.memory.memory	类型：浮点数 **collectd** 内存型内存插件。
collectd.memory.percent	类型：浮点数 **collectd** 百分比型内存插件。

collectd.swap 字段
对应于 **collectd** 交换插件。

参数	描述
collectd.swap.swap	类型：整数 **collectd** 交换型交换插件。
collectd.swap.swap_io	类型：整数 **collectd swap_io** 型交换插件。

collectd.load 字段
对应于 **collectd** 负载插件。

collectd.load.load 字段
collectd 负载型负载插件。

参数	描述
collectd.load.load.shortterm	类型：浮点数 **TODO**
collectd.load.load.midterm	类型：浮点数 **TODO**
collectd.load.load.longterm	类型：浮点数 **TODO**

collectd.aggregation 字段
对应于 **collectd** 聚合插件。

参数	描述
collectd.aggregation.percent	类型：浮点数 **TODO**

collectd.statsd 字段
对应于 **collectd statsd** 插件。

参数	描述
collectd.statsd.host_cpu	类型：整数 **collectd** CPU 型 **statsd** 插件。
collectd.statsd.host_elapsed_time	类型：整数 **collectd elapsed_time** 型 **statsd** 插件。
collectd.statsd.host_memory	类型：整数 **collectd** 内存型 **statsd** 插件。
collectd.statsd.host_nic_speed	类型：整数 **collectd nic_speed** 型 **statsd** 插件。
collectd.statsd.host_nic_rx	类型：整数 **collectd nic_rx** 型 **statsd** 插件。
collectd.statsd.host_nic_tx	类型：整数 **collectd nic_tx** 型 **statsd** 插件。
collectd.statsd.host_nic_rx_dropped	类型：整数 **collectd nic_rx_dropped** 型 **statsd** 插件。
collectd.statsd.host_nic_tx_dropped	类型：整数 **collectd nic_tx_dropped** 型 **statsd** 插件。
collectd.statsd.host_nic_rx_errors	类型：整数 **collectd nic_rx_errors** 型 **statsd** 插件。
collectd.statsd.host_nic_tx_errors	类型：整数 **collectd nic_tx_errors** 型 **statsd** 插件。

参数	描述
collectd.statsd.host_storage	类型：整数 **collectd** 存储型 **statsd** 插件。
collectd.statsd.host_swap	类型：整数 **collectd** 交换型 **statsd** 插件。
collectd.statsd.host_vdsm	类型：整数 **collectd** VDSM 型 **statsd** 插件。
collectd.statsd.host_vms	类型：整数 **collectd** VMS 型 **statsd** 插件。
collectd.statsd.vm_nic_tx_dropped	类型：整数 **collectd nic_tx_dropped** 型 **statsd** 插件。
collectd.statsd.vm_nic_rx_bytes	类型：整数 **collectd nic_rx_bytes** 型 **statsd** 插件。
collectd.statsd.vm_nic_tx_bytes	类型：整数 **collectd nic_tx_bytes** 型 **statsd** 插件。
collectd.statsd.vm_balloon_min	类型：整数 **collectd balloon_min** 型 **statsd** 插件。
collectd.statsd.vm_balloon_max	类型：整数 **collectd balloon_max** 型 **statsd** 插件。
collectd.statsd.vm_balloon_target	类型：整数 **collectd balloon_target** 型 **statsd** 插件。
collectd.statsd.vm_balloon_cur	类型：整数 **collectd balloon_cur** 型 **statsd** 插件。
collectd.statsd.vm_cpu_sys	类型：整数 **collectd cpu_sys** 型 **statsd** 插件。

参数	描述
collectd.statsd.vm_cpu_usage	类型：整数 **collectd cpu_usage** 型 **statsd** 插件。
collectd.statsd.vm_disk_read_ops	类型：整数 **collectd disk_read_ops** 型 **statsd** 插件。
collectd.statsd.vm_disk_write_ops	类型：整数 **statsd** 插件的 **collectd disk_write_ops** 类型。
collectd.statsd.vm_disk_flush_latency	类型：整数 **collectd disk_flush_latency** 型 **statsd** 插件。
collectd.statsd.vm_disk_apparent_size	类型：整数 **collectd disk_apparent_size** 型 **statsd** 插件。
collectd.statsd.vm_disk_write_bytes	类型：整数 **collectd disk_write_bytes** 型 **statsd** 插件。
collectd.statsd.vm_disk_write_rate	类型：整数 **collectd disk_write_rate** 型 **statsd** 插件。
collectd.statsd.vm_disk_true_size	类型：整数 **collectd disk_true_size** 型 **statsd** 插件。
collectd.statsd.vm_disk_read_rate	类型：整数 **collectd disk_read_rate** 型 **statsd** 插件。
collectd.statsd.vm_disk_write_latency	类型：整数 **collectd disk_write_latency** 型 **statsd** 插件。
collectd.statsd.vm_disk_read_latency	类型：整数 **collectd disk_read_latency** 型 **statsd** 插件。
collectd.statsd.vm_disk_read_bytes	类型：整数 **collectd disk_read_bytes** 型 **statsd** 插件。

参数	描述
collectd.statsd.vm_nic_rx_dropped	类型：整数 **collectd nic_rx_dropped** 型 **statsd** 插件。
collectd.statsd.vm_cpu_user	类型：整数 **collectd cpu_user** 型 **statsd** 插件。
collectd.statsd.vm_nic_rx_errors	类型：整数 **collectd nic_rx_errors** 型 **statsd** 插件。
collectd.statsd.vm_nic_tx_errors	类型：整数 **collectd nic_tx_errors** 型 **statsd** 插件。
collectd.statsd.vm_nic_speed	类型：整数 **collectd nic_speed** 型 **statsd** 插件。

collectd.postgresql 字段
对应于 **collectd postgresql** 插件。

参数	描述
collectd.postgresql.pg_n_tup_g	类型：整数 **collectd pg_n_tup_g** 型 postgresql 插件。
collectd.postgresql.pg_n_tup_c	类型：整数 **collectd pg_n_tup_c** 型 postgresql 插件。
collectd.postgresql.pg_numbackends	类型：整数 **collectd pg_numbackends** 型 postgresql 插件。
collectd.postgresql.pg_xact	类型：整数 **collectd pg_xact** 型 postgresql 插件。
collectd.postgresql.pg_db_size	类型：整数 **collectd pg_db_size** 型 postgresql 插件。
collectd.postgresql.pg_blks	类型：整数 **collectd pg_blks** 型 postgresql 插件。

13.2. SYSTEMD 导出字段

以下 **systemd** 字段由 OpenShift Container Platform 集群日志记录导出，可从 Elasticsearch 和 Kibana 搜索。

包括特定于 **systemd** 系统日志的常用字段。应用程序可能会向系统日志写入自己的字段。这些字段在 **systemd.u** 命名空间下提供。**RESULT** 和 **UNIT** 就是这样的两个字段。

systemd.k 字段
下表包含 **systemd** 内核相关的元数据。

参数	描述
systemd.k.KERNEL_DEVICE	**systemd.k.KERNEL_DEVICE** 是内核设备名称。
systemd.k.KERNEL_SUBSYSTEM	**systemd.k.KERNEL_SUBSYSTEM** 是内核子系统名称。
systemd.k.UDEV_DEVLINK	**systemd.k.UDEV_DEVLINK** 包含指向节点的额外符号链接名称。
systemd.k.UDEV_DEVNODE	**systemd.k.UDEV_DEVNODE** 是设备的节点路径。
systemd.k.UDEV_SYSNAME	**systemd.k.UDEV_SYSNAME** 是内核设备名称。

systemd.t 字段
systemd.t 字段是可信的系统日志字段，由系统日志隐式添加，无法通过客户端代码更改。

参数	描述
systemd.t.AUDIT_LOGINUID	**systemd.t.AUDIT_LOGINUID** 是系统日志录入进程的用户 ID。
systemd.t.BOOT_ID	**systemd.t.BOOT_ID** 是内核启动 ID。
systemd.t.AUDIT_SESSION	**systemd.t.AUDIT_SESSION** 是系统日志录入进程的会话。
systemd.t.CAP_EFFECTIVE	**systemd.t.CAP_EFFECTIVE** 代表系统日志录入进程的功能。
systemd.t.CMDLINE	**systemd.t.CMDLINE** 是系统日志录入进程的命令行。
systemd.t.COMM	**systemd.t.COMM** 是系统日志录入进程的名称。
systemd.t.EXE	**systemd.t.EXE** 是系统日志录入进程的可执行路径。

参数	描述
systemd.t.GID	systemd.t.GID 是系统日志录入进程的组 ID。
systemd.t.HOSTNAME	systemd.t.HOSTNAME 是主机的名称。
systemd.t.MACHINE_ID	systemd.t.MACHINE_ID 是主机的机器 ID。
systemd.t.PID	systemd.t.PID 是系统日志录入进程的进程 ID。
systemd.t.SELINUX_CONTEXT	systemd.t.SELINUX_CONTEXT 是系统日志录入进程的安全性上下文或标签。
systemd.t.SOURCE_REALTIME_TIMESTAMP	systemd.t.SOURCE_REALTIME_TIMESTAMP 是消息的最早、最可靠的时间戳。这会转换为 RFC 3339 NS 格式。
systemd.t.SYSTEMD_CGROUP	systemd.t.SYSTEMD_CGROUP 是 **systemd** 控制组路径。
systemd.t.SYSTEMD_OWNER_UID	systemd.t.SYSTEMD_OWNER_UID 是会话的所有者 ID。
systemd.t.SYSTEMD_SESSION	systemd.t.SYSTEMD_SESSION（若适用）是 **systemd** 会话 ID。
systemd.t.SYSTEMD_SLICE	systemd.t.SYSTEMD_SLICE 是系统日志录入进程的切片单元。
systemd.t.SYSTEMD_UNIT	systemd.t.SYSTEMD_UNIT 是会话的单元名称。
systemd.t.SYSTEMD_USER_UNIT	systemd.t.SYSTEMD_USER_UNIT（若适用）是会话的用户单元名称。
systemd.t.TRANSPORT	systemd.t.TRANSPORT 是系统日志服务的录入方法。这包括 **audit**、**driver**、**syslog**、**journal**、**stdout** 和 **kernel**。
systemd.t.UID	systemd.t.UID 是系统日志录入进程的用户 ID。
systemd.t.SYSLOG_FACILITY	systemd.t.SYSLOG_FACILITY 字段含有 **syslog** 的工具，格式为十进制字符串。
systemd.t.SYSLOG_IDENTIFIER	systemd.t.systemd.t.SYSLOG_IDENTIFIER 是 **syslog** 的标识符。
systemd.t.SYSLOG_PID	SYSLOG_PID 是 **syslog** 的客户端进程 ID。

systemd.u 字段

systemd.u Fields 直接从客户端传递，并存入在系统日志中。

参数	描述
systemd.u.CODE_FILE	**systemd.u.CODE_FILE** 是含有源的文件名的代码位置。
systemd.u.CODE_FUNCTION	**systemd.u.CODE_FUNCTION** 是含有源的功能的代码位置。
systemd.u.CODE_LINE	**systemd.u.CODE_LINE** 是含有源的行号的代码位置。
systemd.u.ERRNO	**systemd.u.ERRNO**（若存在）是低级错误编号，是采用数值格式的十进制字符串。
systemd.u.MESSAGE_ID	**systemd.u.MESSAGE_ID** 是用于识别消息类型的消息标识符 ID。
systemd.u.RESULT	仅供私下使用。
systemd.u.UNIT	仅供私下使用。

13.3. KUBERNETES 导出字段

以下 Kubernetes 字段由 OpenShift Container Platform 集群日志记录导出，可从 Elasticsearch 和 Kibana 搜索。

特定于 Kubernetes 元数据的命名空间。**kubernetes.pod_name** 是 Pod 的名称。

kubernetes.labels 字段
附加至 OpenShift 对象的标签是 **kubernetes.labels**。每个标签名都是标签字段的子字段。每个标签名称都会进行去句点处理，即名称中的句点都被替换成下划线。

参数	描述
kubernetes.pod_id	Pod 的 Kubernetes ID。
kubernetes.namespace_name	Kubernetes 中命名空间的名称。
kubernetes.namespace_id	Kubernetes 中命名空间的 ID。
kubernetes.host	Kubernetes 节点名称。
kubernetes.container_name	Kubernetes 中容器的名称。
kubernetes.labels.deployment	与 Kubernetes 对象关联的部署。

参数	描述
kubernetes.labels.deploymentconfig	与 Kubernetes 对象关联的部署配置。
kubernetes.labels.component	与 Kubernetes 对象关联的组件。
kubernetes.labels.provider	与 Kubernetes 对象关联的提供程序。

kubernetes.annotations 字段
与 OpenShift 对象关联的注解是 **kubernetes.annotations** 字段。

13.4. 容器导出字段

以下 Docker 字段由 OpenShift Container Platform 集群日志记录导出，可被 Elasticsearch 和 Kibana 搜索。特定于 docker 容器的元数据的命名空间。docker.container_id 是 Docker 容器 ID。

pipeline_metadata.collector 字段
本节包含与收集器相关的元数据。

参数	描述
pipeline_metadata.collector.hostname	收集器的 FQDN。可能与日志的实际发出者的 FQDN 不同。
pipeline_metadata.collector.name	收集器的名称。
pipeline_metadata.collector.version	收集器的版本。
pipeline_metadata.collector.ipaddr4	收集器服务器的 IPv4 地址，可以是一个数组。
pipeline_metadata.collector.ipaddr6	收集器服务器的 IPv6 地址，可以是一个数组。
pipeline_metadata.collector.inputname	收集器接收日志消息的方式，可以是 TCP/UDP 或 imjournal/imfile。
pipeline_metadata.collector.received_at	收集器收到消息的时间。
pipeline_metadata.collector.original_raw_message	原始的未解析日志消息，由收集器收集或者尽可能与源接近。

pipeline_metadata.normalizer 字段
本节包含与规范化程序相关的元数据。

参数	描述
pipeline_metadata.normalizer.hostname	规范化程序的 FQDN。
pipeline_metadata.normalizer.name	规范化程序的名称。
pipeline_metadata.normalizer.version	规范化程序的版本。
pipeline_metadata.normalizer.ipaddr4	规范化程序服务器的 IPv4 地址，可以是数组。
pipeline_metadata.normalizer.ipaddr6	规范化程序服务器的 IPv6 地址，可以是数组。
pipeline_metadata.normalizer.inputname	规范化程序接收日志消息的方式，可以是 TCP/UDP。
pipeline_metadata.normalizer.received_at	规范化程序收到消息的时间。
pipeline_metadata.normalizer.original_raw_message	原始的未解析日志消息，与规范化程序收到的一致。
pipeline_metadata.trace	该字段记录消息的轨迹。每个收集器和规范化程序都会附加关于自身的信息，以及处理消息的日期和时间。

13.5. OVIRT 导出字段

以下 oVirt 字段由 OpenShift Container Platform 集群日志记录导出，可从 Elasticsearch 和 Kibana 搜索。

oVirt 元数据的命名空间。

参数	描述
ovirt.entity	数据源、主机、VMS 和引擎的类型。
ovirt.host_id	oVirt 主机 UUID。

ovirt.engine 字段
oVirt 引擎相关元数据的命名空间。oVirt 引擎的 FQDN 是 **ovirt.engine.fqdn**

13.6. AUSHAPE 导出字段

以下 Aushape 字段由 OpenShift Container Platform 集群日志记录导出，可从 Elasticsearch 和 Kibana 搜索。

通过 Aushape 转换的审计事件。如需更多信息，请参阅 Aushape。

参数	描述
aushape.serial	审计事件序列号。
aushape.node	发生审计事件的主机的名称。
aushape.error	在转换事件时 Aushape 遇到的错误。
aushape.trimmed	相对于事件对象的 JSONPath 表达式的数组，用于指定因为事件大小限制而删除了内容的对象或数组。空字符串表示事件删除了内容，空数组表示裁剪是由未指定的对象和数组造成的。
aushape.text	代表原始审计事件的一组日志记录字符串。

aushape.data 字段
与 Aushape 相关的已解析审计事件数据。

参数	描述
aushape.data.avc	类型：嵌套
aushape.data.execve	类型：字符串
aushape.data.netfilter_cfg	类型：嵌套
aushape.data.obj_pid	类型：嵌套
aushape.data.path	类型：嵌套

13.7. TLOG 导出字段

以下 Tlog 字段由 OpenShift Container Platform 集群日志记录系统导出，可从 Elasticsearch 和 Kibana 搜索。

记录消息的 Tlog 终端 I/O。如需更多信息，请参阅 Tlog。

参数	描述
tlog.ver	消息格式版本号。

参数	描述
tlog.user	记录的用户名。
tlog.term	终端类型名称。
tlog.session	所记录会话的审计会话 ID。
tlog.id	会话内消息的 ID。
tlog.pos	消息在会话内的位置，以毫秒为单位。
tlog.timing	此消息的事件的时间分布。
tlog.in_txt	清理掉无效字符的输入文本。
tlog.in_bin	清理掉的无效输入字符，以字节数为单位。
tlog.out_txt	清理掉无效字符的输出文本。
tlog.out_bin	清理掉的无效输出字符，以字节数为单位。

第 14 章 卸载集群日志记录

您可以从 OpenShift Container Platform 集群中移除集群日志记录。

14.1. 从 OPENSHIFT CONTAINER PLATFORM 卸载集群日志记录

您可以从集群中移除集群日志记录。

先决条件

- 必须安装 Cluster Logging 和 Elasticsearch。

流程

移除集群日志记录：

1. 使用以下命令，移除部署期间生成的一切内容。

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

2. 使用以下命令，删除 Operator 实例已被删除后保留的持久性卷声明（PVC）：

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