



# OpenShift Container Platform 4.7

## 日志记录

OpenShift Logging 安装、使用和发行注记



# OpenShift Container Platform 4.7 日志记录

---

OpenShift Logging 安装、使用和发行注记

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

## 法律通告

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Logging.ent file |.

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

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

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

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

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

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

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

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

All other trademarks are the property of their respective owners.

## 摘要

本文档提供有关安装、配置和使用 OpenShift Logging 的说明，该 Logging 将汇总多个 OpenShift Container Platform 服务的日志。

## 目录

<b>第 1 章 RED HAT OPENSIFT LOGGING 发行注记</b> .....	<b>7</b>
1.1. 使开源包含更多	7
1.2. 支持的版本	7
1.2.1. OpenShift Logging 5.1.0	7
1.2.1.1. 新功能及功能增强	7
1.2.1.2. 弃用和删除的功能	8
1.2.1.2.1. Elasticsearch Curator 已被删除	8
1.2.1.2.2. 使用旧的 Fluentd 和旧 syslog 方法转发日志已被弃用	8
1.2.1.3. 程序错误修复	8
1.2.2. OpenShift Logging 5.0.9	9
1.2.2.1. 程序错误修复	9
1.2.2.2. CVE	9
1.2.3. OpenShift Logging 5.0.8	10
1.2.3.1. 程序错误修复	10
1.2.4. OpenShift Logging 5.0.7	10
1.2.4.1. 程序错误修复	10
1.2.4.2. CVE	10
1.2.5. OpenShift Logging 5.0.6	13
1.2.5.1. 程序错误修复	13
1.2.5.2. CVE	13
1.2.6. OpenShift Logging 5.0.5	15
1.2.6.1. 安全修复	15
1.2.7. OpenShift Logging 5.0.4	15
1.2.7.1. 安全修复	16
1.2.7.2. 程序错误修复	16
1.2.8. OpenShift Logging 5.0.3	16
1.2.8.1. 安全修复	16
1.2.8.2. 程序错误修复	17
1.2.9. OpenShift Logging 5.0.2	18
1.2.9.1. 程序错误修复	18
1.2.10. OpenShift Logging 5.0.1	18
1.2.10.1. 程序错误修复	18
1.2.11. OpenShift Logging 5.0.0	18
1.2.11.1. 新功能及功能增强	19
Cluster Logging 变为 Red Hat OpenShift Logging	19
每个索引最多五个主分片	19
更新了 OpenShift Elasticsearch Operator 名称和成熟度等级	19
OpenShift Elasticsearch Operator 报告 CSV 成功	19
减少 Elasticsearch pod 证书权限警告	19
从警报到解释和故障排除的新链接	19
删除作业的新连接超时	19
最小化滚动索引模板的更新	19
1.2.11.2. 技术预览功能	19
1.2.11.3. 弃用和删除的功能	20
1.2.11.3.1. Elasticsearch Curator 已被弃用	20
1.2.11.3.2. 使用旧的 Fluentd 和旧 syslog 方法转发日志已被弃用	20
1.2.11.4. 程序错误修复	20
1.2.11.5. 已知问题	22
<b>第 2 章 了解 RED HAT OPENSIFT LOGGING</b> .....	<b>23</b>
2.1. 关于部署 OPENSIFT LOGGING	23

2.1.1. 关于 JSON OpenShift Container Platform Logging	23
2.1.2. 关于收集并存储 Kubernetes 事件	23
2.1.3. 关于更新 OpenShift Container Platform Logging	24
2.1.4. 关于查看集群仪表盘	24
2.1.5. 关于 OpenShift Container Platform Logging 故障排除	24
2.1.6. 关于卸载 OpenShift Container Platform Logging	24
2.1.7. 关于导出字段	24
2.1.8. 关于 OpenShift Logging 组件	24
2.1.9. 关于日志记录收集器	25
2.1.10. 关于日志存储	25
2.1.11. 关于日志记录可视化	26
2.1.12. 关于事件路由	26
2.1.13. 关于日志转发	26
<b>第 3 章 安装 OPENSIFT LOGGING</b>	<b>27</b>
3.1. 使用 WEB 控制台安装 OPENSIFT LOGGING	27
3.2. 安装后的任务	32
3.3. 使用 CLI 安装 OPENSIFT LOGGING	32
3.4. 安装后的任务	39
3.4.1. 定义 Kibana 索引模式	39
3.4.2. 启用网络隔离时允许项目间的流量	40
<b>第 4 章 配置日志部署</b>	<b>42</b>
4.1. 集群日志记录自定义资源 (CR)	42
4.1.1. 关于 ClusterLogging 自定义资源	42
4.2. 配置日志记录收集器	43
4.2.1. 不支持的配置	43
4.2.2. 查看日志记录收集器 Pod	43
4.2.3. 配置日志收集器 CPU 和内存限值	44
4.2.4. 日志转发器的高级配置	45
4.2.5. 如果不使用默认的 Elasticsearch 日志存储, 请删除未使用的组件	48
4.3. 配置日志存储	49
4.3.1. 将审计日志转发到日志存储	50
4.3.2. 配置日志保留时间	51
4.3.3. 为日志存储配置 CPU 和内存请求	53
4.3.4. 为日志存储配置复制策略	55
4.3.5. 缩减 Elasticsearch pod	56
4.3.6. 为日志存储配置持久性存储	56
4.3.7. 为 emptyDir 存储配置日志存储	57
4.3.8. 执行 Elasticsearch 集群滚动重启	57
4.3.9. 将日志存储服务公开为路由	60
4.4. 配置日志可视化工具	63
4.4.1. 配置 CPU 和内存限值	63
4.4.2. 为日志可视化器节点扩展冗余性	64
4.5. 配置 OPENSIFT LOGGING 存储	65
4.5.1. OpenShift Logging 和 OpenShift Container Platform 的存储注意事项	65
4.5.2. 其他资源	65
4.6. 为 OPENSIFT LOGGING 组件配置 CPU 和内存限值	65
4.6.1. 配置 CPU 和内存限值	66
4.7. 使用容忍度来控制 OPENSIFT LOGGING POD 放置	67
4.7.1. 使用容忍度来控制日志存储 pod 放置	68
4.7.2. 使用容忍度来控制日志可视化 pod 放置	69
4.7.3. 使用容忍度来控制日志收集器 pod 放置	70

4.7.4. 其他资源	71
4.8. 使用节点选择器移动 OPENSIFT LOGGING 资源	71
4.8.1. 移动 OpenShift Logging 资源	71
4.9. 配置 SYSTEMD-JOURNALD 和 FLUENTD	75
4.9.1. 为 OpenShift Logging 配置 systemd-journald	75
4.10. 维护和支持	78
4.10.1. 不支持的配置	78
4.10.2. 不支持的配置	78
4.10.3. 非受管 Operator 的支持策略	79
<b>第 5 章 查看资源的日志</b>	<b>80</b>
5.1. 查看资源日志	80
<b>第 6 章 使用 KIBANA 查看集群日志</b>	<b>82</b>
6.1. 定义 KIBANA 索引模式	82
6.2. 在 KIBANA 中查看集群日志	83
<b>第 7 章 将日志转发到第三方系统</b>	<b>86</b>
7.1. 关于将日志转发到第三方系统	86
当外部日志聚合器不可用时，Fluentd 日志处理	90
7.2. 支持的日志数据输出类型	90
7.3. 将日志转发到外部 ELASTICSEARCH 实例	90
7.4. 使用 FLUENTD 转发协议转发日志	92
7.5. 使用 SYSLOG 协议转发日志	94
7.5.1. syslog 参数	96
7.5.2. 其他 RFC5424 syslog 参数	98
7.6. 将日志转发到 KAFKA 代理	98
7.7. 从特定项目转发应用程序日志	100
7.8. 从特定 POD 转发应用程序日志	102
7.9. 使用旧的 FLUENTD 方法转发日志	104
7.10. 使用旧的 SYSLOG 方法转发日志	106
<b>第 8 章 启用 JSON 日志记录</b>	<b>110</b>
8.1. 解析 JSON 日志	110
8.2. 为 ELASTICSEARCH 配置 JSON 日志数据	110
8.3. 将 JSON 日志转发到 ELASTICSEARCH 日志存储	113
<b>第 9 章 收集并存储 KUBERNETES 事件</b>	<b>115</b>
9.1. 部署和配置事件路由器	115
<b>第 10 章 更新 OPENSIFT LOGGING</b>	<b>119</b>
10.1. 从 OPENSIFT CONTAINER PLATFORM 4.6 或更早版本的集群日志记录升级到 OPENSIFT LOGGING 5.X	119
10.2. 将 OPENSIFT LOGGING 更新至当前版本	123
<b>第 11 章 查看集群仪表盘</b>	<b>128</b>
11.1. 访问 ELASTICSEARCH 和 OPENSIFT LOGGING 仪表盘	128
11.2. 关于 OPENSIFT LOGGING 仪表盘	128
11.3. LOGGING/ELASTICSEARCH 节点仪表盘上的图表	130
<b>第 12 章 日志故障排除</b>	<b>135</b>
12.1. 查看 OPENSIFT LOGGING 状态	135
12.1.1. 查看 Red Hat OpenShift Logging Operator 的状态	135
12.1.1.1. 情况消息示例	137
12.1.2. 查看 OpenShift Logging 组件的状态	138

12.2. 查看日志存储的状态	140
12.2.1. 查看日志存储的状态	140
12.2.1.1. 情况消息示例	142
12.2.2. 查看日志存储组件的状态	144
12.3. 了解 OPENSIFT LOGGING 警报	147
12.3.1. 查看日志记录收集器警报	148
12.3.2. 关于日志记录收集器警报	148
12.3.3. 关于 Elasticsearch 警报规则	148
12.4. 为红帽支持收集日志记录数据	149
12.4.1. 关于 must-gather 工具	150
12.4.2. 先决条件	150
12.4.3. 收集 OpenShift Logging 数据	150
12.5. 关键警报故障排除	150
12.5.1. Elasticsearch Cluster Health 是红色	151
12.5.2. Elasticsearch Cluster Health 为黄色	153
12.5.3. 已达到 Elasticsearch 节点磁盘低水位线	153
12.5.4. 已达到 Elasticsearch 节点磁盘高水位线	154
12.5.5. 已达到 Elasticsearch 节点磁盘的洪水水位线	155
12.5.6. Elasticsearch JVM 堆使用率是高	157
12.5.7. 聚合日志记录系统 CPU 为高	157
12.5.8. Elasticsearch 进程 CPU 为高	157
12.5.9. Elasticsearch 磁盘空间现为低	157
12.5.10. Elasticsearch FileDescriptor 使用为高	158
<b>第 13 章 卸载 OPENSIFT LOGGING</b>	<b>159</b>
13.1. 从 OPENSIFT CONTAINER PLATFORM 卸载 OPENSIFT LOGGING	159
<b>第 14 章 日志记录字段</b>	<b>161</b>
<b>第 15 章 MESSAGE</b>	<b>162</b>
<b>第 16 章 结构化</b>	<b>163</b>
<b>第 17 章 @TIMESTAMP</b>	<b>164</b>
<b>第 18 章 HOSTNAME</b>	<b>165</b>
<b>第 19 章 IPADDR4</b>	<b>166</b>
<b>第 20 章 IPADDR6</b>	<b>167</b>
<b>第 21 章 LEVEL</b>	<b>168</b>
<b>第 22 章 PID</b>	<b>169</b>
<b>第 23 章 SERVICE</b>	<b>170</b>
<b>第 24 章 TAGS</b>	<b>171</b>
<b>第 25 章 FILE</b>	<b>172</b>
<b>第 26 章 OFFSET</b>	<b>173</b>
<b>第 27 章 KUBERNETES</b>	<b>174</b>
27.1. KUBERNETES.POD_NAME	174
27.2. KUBERNETES.POD_ID	174
27.3. KUBERNETES.NAMESPACE_NAME	174
27.4. KUBERNETES.NAMESPACE_ID	174

---

27.5. KUBERNETES.HOST	174
27.6. KUBERNETES.CONTAINER_NAME	174
27.7. KUBERNETES.ANNOTATIONS	175
27.8. KUBERNETES.LABELS	175
27.9. KUBERNETES.EVENT	175
27.9.1. kubernetes.event.verb	175
27.9.2. kubernetes.event.metadata	175
27.9.2.1. kubernetes.event.metadata.name	175
27.9.2.2. kubernetes.event.metadata.namespace	175
27.9.2.3. kubernetes.event.metadata.selfLink	176
27.9.2.4. kubernetes.event.metadata.uid	176
27.9.2.5. kubernetes.event.metadata.resourceVersion	176
27.9.3. kubernetes.event.involvedObject	176
27.9.3.1. kubernetes.event.involvedObject.kind	176
27.9.3.2. kubernetes.event.involvedObject.namespace	176
27.9.3.3. kubernetes.event.involvedObject.name	177
27.9.3.4. kubernetes.event.involvedObject.uid	177
27.9.3.5. kubernetes.event.involvedObject.apiVersion	177
27.9.3.6. kubernetes.event.involvedObject.resourceVersion	177
27.9.4. kubernetes.event.reason	177
27.9.5. kubernetes.event.source_component	178
27.9.6. kubernetes.event.firstTimestamp	178
27.9.7. kubernetes.event.count	178
27.9.8. kubernetes.event.type	178
<b>第 28 章 OPENSIFT</b> .....	<b>179</b>
28.1. OPENSIFT.LABELS	179



# 第 1 章 RED HAT OPENSIFT LOGGING 发行注记

## 1.1. 使开源包含更多

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始：master、slave、blacklist 和 whitelist。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看 [Red Hat CTO Chris Wright 信息](#)。

## 1.2. 支持的版本

表 1.1. OpenShift Container Platform 版本支持 Red Hat OpenShift Logging (RHOL)

	4.7	4.8	4.9
RHOL 5.1	X	X	
RHOL 5.2	X	X	X
RHOL 5.3		X	X

### 1.2.1. OpenShift Logging 5.1.0

此发行版本包括 [RHSA-2021:2112 OpenShift Logging 程序错误修复 5.1.0](#)。

#### 1.2.1.1. 新功能及功能增强

OpenShift Logging 5.1 现在支持 OpenShift Container Platform 4.7 及之后的版本运行：

- IBM Power 系统
- IBM Z 和 LinuxONE

此版本对以下方面进行了改进

- 作为集群管理员，您可以使用 Kubernetes pod 标签从应用程序收集日志数据并将其发送到特定的日志存储。您可以通过在 **ClusterLogForwarder** 自定义资源 (CR) YAML 文件中配置 **inputs[].application.selector.matchLabels** 元素来收集日志数据。您还可以按命名空间过滤收集的日志数据。（[LOG-883](#)）
- 此发行版本将以下新的 **ElasticsearchNodeDiskWatermarkReached** 警告添加到 OpenShift Elasticsearch Operator (EO)：
  - 已达到 Elasticsearch 节点磁盘低水位线
  - 已达到 Elasticsearch 节点磁盘高水位线
  - 已达到 Elasticsearch 节点磁盘的洪水水位线

当 Elasticsearch 节点预计在未来 6 小时内将到达**磁盘低水位线**、**磁盘高水位线**或**磁盘洪水 Stage 水位线**阈值时，警报会应用过去几个警告。这个警告周期为您提供了在节点达到磁盘水位线阈值前响应的的时间。警告消息也提供故障排除步骤的链接，您可以按照这些链接来帮助缓解问题。EO 应用过去几小时的磁盘空间数据到线性模型来生成这些警告。（[LOG-1100](#)）

- JSON 日志现在可以作为 JSON 对象（而不是带引号的字符串）转发到红帽受管 Elasticsearch 集群或其他支持的第三方系统。另外，您现在可以从 Kibana 中的 JSON 日志消息查询各个字段，从而增加特定日志的可发现性。(LOG-785, LOG-1148)

### 1.2.1.2. 弃用和删除的功能

之前版本中的一些功能已被弃用或删除。

弃用的功能仍然包含在 OpenShift Logging 中，并且仍然被支持。但是，这个功能会在以后的发行版本中被删除，且不建议在新的部署中使用。

#### 1.2.1.2.1. Elasticsearch Curator 已被删除

在这个版本中，Elasticsearch Curator 已被移除，不再被支持。Elasticsearch Curator 可帮助您在 OpenShift Container Platform 4.4 及更早版本中对索引进行策展或管理。配置日志保留时间，而不是使用 Elasticsearch Curator。

#### 1.2.1.2.2. 使用旧的 Fluentd 和旧 syslog 方法转发日志已被弃用

从 OpenShift Container Platform 4.6 升级到现在，使用旧的 Fluentd 和旧 syslog 方法转发日志已被弃用，并将在以后的发行版本中删除。应使用标准的非传统方法。

### 1.2.1.3. 程序错误修复

- 在更新之前，**ClusterLogForwarder** CR 在创建后不会显示 **input[].selector** 元素。在这个版本中，当您在 **ClusterLogForwarder** CR 中指定一个 **selector** 时，它会保留。对于 LOG-883，需要修复此错误，这将启用使用 pod 标签选择器来转发应用日志数据。(LOG-1338)
- 在以前的版本中，集群服务版本 (CSV) 中的更新会意外地为 OpenShift Elasticsearch Operator 容器引入资源和限制。特定情况下，这会导致终止 Elasticsearch Operator pod 的内存不足状况。在这个版本中，通过删除 Operator 容器的 CSV 资源和限制解决了这个问题。Operator 可以被调度且没有问题。(LOG-1254)
- 在更新前，使用串联的证书将日志转发到 Kafka 会失败并显示以下错误消息：  
**state=error: certificate verify failed (unable to get local issuer certificate)**

无法将日志转发到带有中间 CA 签名的证书的 Kafka 代理。这是因为 Fluentd Kafka 插件只能处理对应 secret 的 **ca-bundle.crt** 条目中提供的单个 CA 证书。在这个版本中解决了这个问题，它启用了 Fluentd Kafka 插件来处理对应 secret 的 **ca-bundle.crt** 条目中提供的多个 CA 证书。现在，日志可以被转发到带有中间 CA 签名的证书的 Kafka 代理。(LOG-1218, LOG-1216)

- 在以前的版本中，当存在负载时，Elasticsearch 可能会对一些请求响应 HTTP 500 错误，即使集群没有错误。重试请求会成功。在这个版本中解决了这个问题，方法是更新索引管理 cron 作业使其在遇到临时 HTTP 500 错误时更具弹性。更新的索引管理 cron 作业将在失败前多次重试请求。(LOG-1215)
- 在以前的版本中，如果您没有在集群安装配置中设置 **.proxy** 值，然后在已安装的集群上配置了全局代理，则存在一个程序漏洞，它会阻止 Fluentd 将日志转发到 Elasticsearch。要临时解决这个问题，在代理或集群配置中，将 **no\_proxy** 值设置为 **.svc.cluster.local**，使其跳过内部流量。在这个版本中解决了代理配置问题。如果在安装 OpenShift Container Platform 集群后配置全局代理，Fluentd 将日志转发到 Elasticsearch。(LOG-1187, BZ#1915448)
- 在更新之前，日志记录收集器创建了比必要更多的套接字连接。在这个版本中，日志记录收集器重复使用现有的套接字连接来发送日志。(LOG-1186)
- 在更新前，如果集群管理员尝试从 Elasticsearch 集群中添加或删除存储，OpenShift

Elasticsearch Operator (EO) 会错误地尝试升级 Elasticsearch 集群，显示 **scheduledUpgrade: "True", shardAllocationEnabled: 26ries** 并更改卷。在这个版本中，EO 不会尝试升级 Elasticsearch 集群。

EO 状态显示以下新状态信息，以指示您尝试对已忽略的 Elasticsearch 存储进行不受支持的更改：

- 当您尝试在临时存储结构和持久存储结构之间更改时，**StorageStructureChangeIgnored**。
- 当您尝试更改存储类名称时，**StorageClassNameChangeIgnored**。
- 当您尝试更改存储大小时，**StorageSizeChangeIgnored**。



### 注意

如果您将 **ClusterLogging** 自定义资源 (CR) 配置为从临时切换到持久性存储，EO 会创建一个持久性卷声明 (PVC)，但不创建持久性卷 (PV)。要清除 **StorageStructureChangeIgnored** 状态，您必须恢复对 **ClusterLogging** CR 的更改并删除持久性卷声明 (PVC)。

([LOG-1351](#))

- 在更新前，如果您重新部署了一个完整的 Elasticsearch 集群，它会处于不健康状态，一个非数据节点正在运行，所有其他数据节点关闭。出现这个问题的原因是，新证书阻止 Elasticsearch Operator 缩减 Elasticsearch 集群的非数据节点。在这个版本中，Elasticsearch Operator 可以缩减所有数据和非数据节点，然后再次备份，以便它们加载新证书。Elasticsearch Operator 加载新证书后可以访问新节点。(LOG-1536)

## 1.2.2. OpenShift Logging 5.0.9

此发行版本包括 [RHBA-2021:3705 - 程序错误修复公告](#)。OpenShift Logging 程序错误修复版本(5.0.9)。

### 1.2.2.1. 程序错误修复

此发行版本包括以下程序错误修正：

- 在更新前，一些日志条目没有被识别为 UTF-8 字节，这会导致 Elasticsearch 拒绝消息并阻塞整个缓冲的有效负载。在这个版本中解决了这个问题：被拒绝的载荷丢弃了无效的日志条目并重新提交剩余的条目。(LOG-1574)
- 在更新前，编辑 **ClusterLogging** 自定义资源 (CR) 不会将 **totalLimitSize** 的值应用到 Fluentd **total\_limit\_size** 字段，这会限制缓冲插件实例的大小。因此，Fluentd 应用默认值。在这个版本中，CR 将 **totalLimitSize** 的值应用到 Fluentd **total\_limit\_size** 字段。Fluentd 使用 **total\_limit\_size** 字段的值或默认值（以较少者为准）。(LOG-1736)

### 1.2.2.2. CVE

- [CVE-2020-25648](#)
- [CVE-2021-22922](#)
- [CVE-2021-22923](#)
- [CVE-2021-22924](#)
- [CVE-2021-36222](#)

- [CVE-2021-37576](#)
- [CVE-2021-37750](#)
- [CVE-2021-38201](#)

### 1.2.3. OpenShift Logging 5.0.8

此发行版本包括 [RHBA-2021:3526 - 程序错误修复公告](#)。OpenShift Logging 程序错误修复版本(5.0.8)。

#### 1.2.3.1. 程序错误修复

此发行版本还包括以下程序错误修正：

- 由于发行管道脚本中存在问题，olm **.skipRange** 字段的值在 **5.2.0** 中保持不变，且在 z-stream 号 **0** 增加时没有更新。当前发行版本修复了管道脚本，以在发行号改变时更新此字段的值。  
([LOG-1741](#))

### 1.2.4. OpenShift Logging 5.0.7

此发行版本包括 [RHBA-2021:2884 - 程序错误修复公告](#)。OpenShift Logging 程序错误修复版本(5.0.7)。

#### 1.2.4.1. 程序错误修复

此发行版本还包括以下程序错误修正：

- [LOG-1594](#) - 厂商的 viaq/logerr 依赖项缺少许可证文件

#### 1.2.4.2. CVE

- [CVE-2016-10228](#)
- [CVE-2017-14502](#)
- [CVE-2018-25011](#)
- [CVE-2019-2708](#)
- [CVE-2019-3842](#)
- [CVE-2019-9169](#)
- [CVE-2019-13012](#)
- [CVE-2019-18276](#)
- [CVE-2019-18811](#)
- [CVE-2019-19523](#)
- [CVE-2019-19528](#)
- [CVE-2019-25013](#)
- [CVE-2020-0431](#)

- [CVE-2020-8231](#)
- [CVE-2020-8284](#)
- [CVE-2020-8285](#)
- [CVE-2020-8286](#)
- [CVE-2020-8927](#)
- [CVE-2020-9948](#)
- [CVE-2020-9951](#)
- [CVE-2020-9983](#)
- [CVE-2020-10543](#)
- [CVE-2020-10878](#)
- [CVE-2020-11608](#)
- [CVE-2020-12114](#)
- [CVE-2020-12362](#)
- [CVE-2020-12363](#)
- [CVE-2020-12364](#)
- [CVE-2020-12464](#)
- [CVE-2020-13434](#)
- [CVE-2020-13543](#)
- [CVE-2020-13584](#)
- [CVE-2020-13776](#)
- [CVE-2020-14314](#)
- [CVE-2020-14344](#)
- [CVE-2020-14345](#)
- [CVE-2020-14346](#)
- [CVE-2020-14347](#)
- [CVE-2020-14356](#)
- [CVE-2020-14360](#)
- [CVE-2020-14361](#)
- [CVE-2020-14362](#)

- [CVE-2020-14363](#)
- [CVE-2020-15358](#)
- [CVE-2020-15437](#)
- [CVE-2020-24394](#)
- [CVE-2020-24977](#)
- [CVE-2020-25212](#)
- [CVE-2020-25284](#)
- [CVE-2020-25285](#)
- [CVE-2020-25643](#)
- [CVE-2020-25704](#)
- [CVE-2020-25712](#)
- [CVE-2020-26116](#)
- [CVE-2020-26137](#)
- [CVE-2020-26541](#)
- [CVE-2020-27618](#)
- [CVE-2020-27619](#)
- [CVE-2020-27786](#)
- [CVE-2020-27835](#)
- [CVE-2020-28196](#)
- [CVE-2020-28974](#)
- [CVE-2020-29361](#)
- [CVE-2020-29362](#)
- [CVE-2020-29363](#)
- [CVE-2020-35508](#)
- [CVE-2020-36322](#)
- [CVE-2020-36328](#)
- [CVE-2020-36329](#)
- [CVE-2021-0342](#)
- [CVE-2021-0605](#)

- [CVE-2021-3177](#)
- [CVE-2021-3326](#)
- [CVE-2021-3501](#)
- [CVE-2021-3516](#)
- [CVE-2021-3517](#)
- [CVE-2021-3518](#)
- [CVE-2021-3520](#)
- [CVE-2021-3537](#)
- [CVE-2021-3541](#)
- [CVE-2021-3543](#)
- [CVE-2021-20271](#)
- [CVE-2021-23336](#)
- [CVE-2021-27219](#)
- [CVE-2021-33034](#)

### 1.2.5. OpenShift Logging 5.0.6

此发行版本包括 [RHBA-2021:2655 - 程序错误修复公告](#)。OpenShift Logging 程序错误修复版本 (5.0.6)。

#### 1.2.5.1. 程序错误修复

此发行版本还包括以下程序错误修正：

- LOG-1451 - [1927249] fieldmanager.go:186] [SHOULD NOT HAPPEN] failed to update managedFields...duplicate entries for key [name="POLICY\_MAPPING"] ([LOG-1451](#))
- LOG-1537 - Full Cluster Cert Redeploy is broken when the ES clusters includes non-data nodes([LOG-1537](#))
- LOG-1430 - eventrouter raising "Observed a panic: &runtime.TypeAssertionError" ([LOG-1430](#))
- LOG-1461 - 索引管理作业状态始终为 **Completed**，即使作业日志中出现错误。([LOG-1461](#))
- LOG-1459 - Operator 缺少断开连接的注解 ([LOG-1459](#))
- LOG-1572 - Bug 1981579：修复内置应用行为以收集所有日志 ([LOG-1572](#))

#### 1.2.5.2. CVE

- [CVE-2016-10228](#)
- [CVE-2017-14502](#)
- [CVE-2018-25011](#)

- [CVE-2019-2708](#)
- [CVE-2019-9169](#)
- [CVE-2019-25013](#)
- [CVE-2020-8231](#)
- [CVE-2020-8284](#)
- [CVE-2020-8285](#)
- [CVE-2020-8286](#)
- [CVE-2020-8927](#)
- [CVE-2020-10543](#)
- [CVE-2020-10878](#)
- [CVE-2020-13434](#)
- [CVE-2020-14344](#)
- [CVE-2020-14345](#)
- [CVE-2020-14346](#)
- [CVE-2020-14347](#)
- [CVE-2020-14360](#)
- [CVE-2020-14361](#)
- [CVE-2020-14362](#)
- [CVE-2020-14363](#)
- [CVE-2020-15358](#)
- [CVE-2020-25712](#)
- [CVE-2020-26116](#)
- [CVE-2020-26137](#)
- [CVE-2020-26541](#)
- [CVE-2020-27618](#)
- [CVE-2020-27619](#)
- [CVE-2020-28196](#)
- [CVE-2020-29361](#)
- [CVE-2020-29362](#)

- [CVE-2020-29363](#)
- [CVE-2020-36328](#)
- [CVE-2020-36329](#)
- [CVE-2021-3177](#)
- [CVE-2021-3326](#)
- [CVE-2021-3516](#)
- [CVE-2021-3517](#)
- [CVE-2021-3518](#)
- [CVE-2021-3520](#)
- [CVE-2021-3537](#)
- [CVE-2021-3541](#)
- [CVE-2021-20271](#)
- [CVE-2021-23336](#)
- [CVE-2021-27219](#)
- [CVE-2021-33034](#)

## 1.2.6. OpenShift Logging 5.0.5

此发行版本包括 [RHSA-2021:2374 - 安全公告](#)。中度：OpenShift Logging 程序错误修复版本 (5.0.5)。

### 1.2.6.1. 安全修复

- Gogo/protobuf:plugin/unmarshal/unmarshal.go 缺少某些索引验证。（[CVE-2021-3121](#)）
- glib : 64 位平台上 g\_bytes\_new 功能中的整数溢出，因为隐式多播从 64 位到 32位([CVE-2021-27219](#))

以下问题与上述 CVE 相关：

- BZ#1921650 gogo/protobuf: plugin/unmarshal/unmarshal.go 缺少特定的索引验证 ([BZ#1921650](#))
- LOG-1361 CVE-2021-3121 elasticsearch-operator-container: gogo/protobuf: plugin/unmarshal/unmarshal.go 缺少特定的索引验证 [openshift-logging-5] ([LOG-1361](#))
- LOG-1362 CVE-2021-3121 elasticsearch-proxy-container: gogo/protobuf: plugin/unmarshal/unmarshal.go 缺少特定的索引验证 [openshift-logging-5] ([LOG-1362](#))
- LOG-1363 CVE-2021-3121 logging-eventrouter-container: gogo/protobuf: plugin/unmarshal/unmarshal.go 缺少特定的索引验证 [openshift-logging-5] ([LOG-1363](#))

## 1.2.7. OpenShift Logging 5.0.4

此发行版本包括 [RHSA-2021:2136 - 安全公告](#)。中度：OpenShift Logging 安全和漏洞更新 (5.0.4)。

### 1.2.7.1. 安全修复

- Gogo/protobuf:plugin/unmarshal/unmarshal.go 缺少某些索引验证。 ([CVE-2021-3121](#))

以下 Jira 包含上述 CVE：

- LOG-1364 CVE-2021-3121 cluster-logging-operator-container: gogo/protobuf:plugin/unmarshal/unmarshal.go 缺少特定的索引验证 [openshift-logging-5]。 ([LOG-1364](#))

### 1.2.7.2. 程序错误修复

此发行版本还包括以下程序错误修正：

- LOG-1328 Port fix to 5.0.z for BZ-1945168. ([LOG-1328](#))

## 1.2.8. OpenShift Logging 5.0.3

此发行版本包括 [RHSA-2021:1515 - 安全公告](#)。重要的 OpenShift Logging 程序错误修复版本 (5.0.3)。

### 1.2.8.1. 安全修复

- jackson-databind: slf4j-ext 类中的任意代码执行 ([CVE-2018-14718](#))
- jackson-databind: blaze-ds-opt 和 blaze-ds-core 类中的任意代码执行 ([CVE-2018-14719](#))
- jackson-databind：在某些 JDK 类中进行过滤/XXE ([CVE-2018-14720](#))
- jackson-databind：axis2-jaxws 类中的服务器端请求伪造 (SSRF) ([CVE-2018-14721](#))
- jackson-databind: axis2-transport-jms 类中的不正确的 polymorphic deserialization ([CVE-2018-19360](#))
- jackson-databind: openjpa 类中的不正确的 polymorphic deserialization ([CVE-2018-19361](#))
- jackson-databind: jboss-common-core 类中的不正确的 polymorphic deserialization ([CVE-2018-19362](#))
- jackson-databind：默认输入错误处理导致执行远程代码 ([CVE-2019-14379](#))
- jackson-databind：com.pastdev.httpcomponents.configuration.JndiConfiguration 中的 serialization gadgets ([CVE-2020-24750](#))
- Jackson-databind:错误地处理与 org.apache.commons.dbcp2.datasources.PerUserPoolDataSource ([CVE-2020-35490](#)) 相关的序列化 gadget 和输入之间的交互
- Jackson-databind:错误处理与 org.apache.commons.dbcp2.datasources.SharedPoolDataSource([CVE-2020-35491](#))相关的序列化 gadget 和输入之间的交互
- Jackson-databind:错误处理与 com.oracle.wls.shaded.org.apache.xalan.lib.sql.JNDIConnectionPool ([CVE-2020-35728](#))

- Jackson-databind:错误处理与  
oadd.org.apache.commons.dbcp.cpsadapter.DriverAdapterCPDS ([CVE-2020-36179](#))
- Jackson-databind:错误地处理与  
org.apache.commons.dbcp2.cpsadapter.DriverAdapterCPDS ([CVE-2020-36180](#))
- Jackson-databind:错误地处理与  
org.apache.tomcat.dbcp.dbcp.cpsadapter.DriverAdapterCPDS ([CVE-2020-36181](#))
- Jackson-databind:错误地处理与  
org.apache.tomcat.dbcp.dbcp2.cpsadapter.DriverAdapterCPDS ([CVE-2020-36182](#))
- Jackson-databind:错误处理序列化 gadgets 和输入与  
org.docx4j.org.apache.xalan.lib.sql.JNDIConnectionPool ([CVE-2020-36183](#))
- Jackson-databind:错误地处理与  
org.apache.tomcat.dbcp.dbcp2.datasources.PerUserPoolDataSource ([CVE-2020-36184](#)) 相关的序列化 gadget 和输入的交互
- Jackson-databind:错误地处理与  
org.apache.tomcat.dbcp.dbcp2.datasources.SharedPoolDataSource ([CVE-2020-36185](#))
- jackson-databind : 与 org.apache.tomcat.dbcp.dbcp.datasources.PerUserPoolDataSource 相关的, 在 serialization gadgets 和 typing 间交互的错误处理 ([CVE-2020-36186](#))
- jackson-databind : 与 org.apache.tomcat.dbcp.dbcp.datasources.SharedPoolDataSource 相关的, 在 serialization gadgets 和 typing 间交互的错误处理 ([CVE-2020-36187](#))
- jackson-databind : 与 com.newrelic.agent.deps.ch.qos.logback.core.db.JNDIConnectionSource 相关的, 在 serialization gadgets 和 typing 间交互的错误处理 ([CVE-2020-36188](#))
- jackson-databind : 与  
com.newrelic.agent.deps.ch.qos.logback.core.db.DriverManagerConnectionSource 相关的, 在 serialization gadgets 和 typing 间交互的错误处理 ([CVE-2020-36189](#))
- jackson-databind : 与 javax.swing 相关的, 在 serialization gadgets 和 typing 间的交互的错误处理 ([CVE-2021-20190](#))
- golang : 在特定包括了 ReverseProxy 的 net/http 服务器中的数据竞争可能会导致 DoS ([CVE-2020-15586](#))
- golang : ReadUvarint 和 ReadVarint 可以从无效的输入中读取没有限制的字节数 ([CVE-2020-16845](#))
- OpenJDK : 不完整的 JAR 签名禁用算法的强制实施 (Librial, 8249906) ([CVE-2021-2163](#))

以下 Jira 包含上述 CVE :

- LOG-1234 CVE-2020-15586 CVE-2020-16845 openshift-eventrouter: various flaws [openshift-4].([LOG-1234](#))
- LOG-1243 CVE-2018-14718 CVE-2018-14719 CVE-2018-14720 CVE-2018-14721 CVE-2018-19360 CVE-2018-19361 CVE-2018-19362 CVE-2019-14379 CVE-2020-35490 CVE-2020-35491 CVE-2020-35728... logging-elasticsearch6-container: various flaws [openshift-logging-5.0].([LOG-1243](#))

### 1.2.8.2. 程序错误修复

此发行版本还包括以下程序错误修正：

- LOG-1224 版本 5.0 - 特定于 ClusterLogForwarder 命名空间的日志转发无法正常工作。 ([LOG-1224](#))
- LOG-1232 5.0 - Bug 1859004 - 有时事件路由器无法收集事件日志。 ([LOG-1232](#))
- LOG-1299 Release 5.0 - 使用证书链将日志转发到 Kafka 失败，错误为 "state=error: certificate verify failed (unable to get local issuer certificate)"。 ([LOG-1299](#))

## 1.2.9. OpenShift Logging 5.0.2

此发行版本包括 [RHBA-2021:1167 - 程序错误修复公告](#)。OpenShift Logging 程序错误修复版本 (5.0.2)。

### 1.2.9.1. 程序错误修复

- 如果您没有在集群安装配置中设置 `.proxy`，然后在已安装的集群上配置了全局代理，则存在一个程序漏洞，阻止 Fluentd 将日志转发到 Elasticsearch。要临时解决这个问题，在 `proxy/cluster` 配置中将 `no_proxy` 设置为 `.svc.cluster.local`，以便跳过内部流量。当前发行版本解决了代理配置问题。现在，如果您在安装 OpenShift 集群后配置全局代理，Fluentd 会将日志转发到 Elasticsearch。 ([LOG-1187](#))
- 在以前的版本中，使用链证书将日志转发到 Kafka 会失败，错误为 "state=error: certificate verify failed (无法获取本地签发者证书)。" 无法将日志转发到带有中间 CA 签名的证书的 Kafka 代理。这是因为 fluentd Kafka 插件只能处理相应 secret 的 `ca-bundle.crt` 条目中提供的单个 CA 证书。当前发行版本解决了这个问题，方法是启用 fluentd Kafka 插件处理对应 secret 的 `ca-bundle.crt` 条目中提供的多个 CA 证书。现在，日志可以被转发到带有中间 CA 签名的证书的 Kafka 代理。 ([LOG-1216](#), [LOG-1218](#))
- 在以前的版本中，集群服务版本 (CSV) 中的更新会意外地为 OpenShift Elasticsearch operator 容器引入资源和限制。在特定情况下，这会导致内存不足条件终止 Elasticsearch operator pod。当前发行版本解决了这个问题，方法是删除 operator 容器的 CSV 资源及限制。现在，Operator 会被调度而没有问题。 ([LOG-1254](#))

## 1.2.10. OpenShift Logging 5.0.1

此发行版本包括 [RHBA-2021:0963 - 程序错误修复公告](#)。OpenShift Logging 程序错误修复版本 (5.0.1)。

### 1.2.10.1. 程序错误修复

- 在以前的版本中，如果启用了旧的日志转发系统，日志不会发送到受管存储。造成这个问题的原因是，生成的日志转发配置无法正确地在当前的日志转发系统和旧的日志转发系统间进行选择。当前发行版本解决了这个问题。如果 `ClusterLogging` CR 定义了一个 `logstore`，日志将发送到受管存储。另外，如果启用了旧的日志转发，无论是否启用了受管存储，日志都会发送到旧的日志转发。 ([LOG-1172](#))
- 在以前的版本中，当存在负载时，Elasticsearch 可能会对一些请求响应 HTTP 500 错误，即使集群没有错误。重试请求会成功。此发行版本解决了这个问题，现在更新了 cron 任务使其在遇到临时 HTTP 500 错误时可以更好地进行处理。现在，它们会在失败前会先进行多次请求。 ([LOG-1215](#))

## 1.2.11. OpenShift Logging 5.0.0

此发行版本包括 [RHBA-2021:0652 - 程序错误修复公告](#)。OpenShift Logging 5.0.0 的勘误公告。

### 1.2.11.1. 新功能及功能增强

此版本对以下方面进行了改进。

#### Cluster Logging 变为 Red Hat OpenShift Logging

在这个版本中，Cluster Logging 变为 Red Hat OpenShift Logging 5.0。

#### 每个索引最多五个主分片

在这个版本中，OpenShift Elasticsearch Operator (EO) 会根据为集群定义的数据节点数量来设置一个或多个索引的主分片数量。

在以前的版本中，EO 会将索引的分片数量设置为数据节点的数量。当 Elasticsearch 中的索引被配置为带有多个副本时，它会为每个主分片创建多个副本，而不是每个索引。因此，因为索引分片，集群中存在更多副本分片，这为集群复制和保持同步造成大量消耗。

#### 更新了 OpenShift Elasticsearch Operator 名称和成熟度等级

此发行版本更新了 OpenShift Elasticsearch Operator 和 Operator 成熟度等级的显示名称。Operator Hub 中更新了新的显示名称并明确指定了 OpenShift Elasticsearch Operator 使用。

#### OpenShift Elasticsearch Operator 报告 CSV 成功

此发行版本添加了报告指标，以表示 OpenShift Elasticsearch Operator 的 **ClusterServiceVersion** (CSV) 对象已成功。在以前的版本中，如果安装或升级 CSV 失败，则无法决定或生成警报。现在，会提供一个警报作为 OpenShift Elasticsearch Operator 的一部分。

#### 减少 Elasticsearch pod 证书权限警告

在以前的版本中，当 Elasticsearch Pod 启动时，它会生成证书权限警告，这会使一些用户对集群进行故障排除。当前的发行版本解决了这些权限问题，以减少这些类型的通知。

#### 从警报到解释和故障排除的新链接

此发行版本添加了来自 Elasticsearch 集群生成的警报的链接至该警报的解释和故障排除步骤页面。

#### 删除作业的新连接超时

当前的发行版本为删除作业添加连接超时，这有助于防止 pod 在查询 Elasticsearch 以删除索引时偶尔挂起。现在，如果底层的 'curl' 调用在超时前没有连接，超时会终止这个调用。

#### 最小化滚动索引模板的更新

在这个版本中，如果 OpenShift Elasticsearch Operator 具有不同的字段值，仅会更新其滚动索引模板。索引模板具有高于索引的优先级。更新模板时，集群会优先将它们分发到索引分片上，这会影响性能。为最小化 Elasticsearch 集群操作，operator 只有在主分片或副本分片数量发生变化时才会更新模板。

### 1.2.11.2. 技术预览功能

这个版本中的一些功能当前还处于技术预览状态。它们并不适用于在生产环境中使用。请参阅红帽门户网站中关于对技术预览功能支持范围的信息：

#### 技术预览功能支持范围

在下表中，功能被标记为以下状态：

- **TP:** 技术预览
- **GA:** 正式发行
- **-:** Not Available

表 1.2. 技术预览

功能	OCP 4.5	OCP 4.6	Logging 5.0
日志转发	TP	GA	GA

### 1.2.11.3. 弃用和删除的功能

之前版本中的一些功能已被弃用或删除。

弃用的功能仍然包含在 OpenShift Logging 中，并且仍然被支持。但是，这个功能会在以后的发行版本中被删除，且不建议在新的部署中使用。

#### 1.2.11.3.1. Elasticsearch Curator 已被弃用

Elasticsearch Curator 已被弃用，并将在以后的发行版本中删除。Elasticsearch Curator 可帮助您在 OpenShift Container Platform 4.4 及更早版本中对索引进行策展或管理。配置日志保留时间，而不是使用 Elasticsearch Curator。

#### 1.2.11.3.2. 使用旧的 Fluentd 和旧 syslog 方法转发日志已被弃用

从 OpenShift Container Platform 4.6 升级到现在，使用旧的 Fluentd 和旧 syslog 方法转发日志已被弃用，并将在以后的发行版本中删除。应使用标准的非传统方法。

### 1.2.11.4. 程序错误修复

- 在以前的版本中，Elasticsearch 会拒绝 HTTP 请求，其标头超过默认的最大标头大小 8 KB。现在，最大标头大小为 128 KB，Elasticsearch 不再拒绝超过最大标头大小的 HTTP 请求。[\(BZ#1845293\)](#)
- 在以前的版本中，节点无法从 **Pending** 状态恢复，因为一个软件漏洞导致无法在 Elasticsearch 自定义资源 (CR) 中正确更新其状态。当前发行版本解决了这个问题，节点可在从 **Pending** 状态中进行恢复。[\(BZ#1887357\)](#)
- 在以前的版本中，当 Cluster Logging Operator (CLO) 将 **clusterlogging** CR 中的 Elasticsearch 节点数量缩减为三个节点时，它会忽略之前创建的具有唯一 ID 的节点。OpenShift Elasticsearch Operator 会拒绝更新，因为它有一个保护机制来防止带有唯一 ID 的节点被删除。现在，当 CLO 缩减节点数量并更新 Elasticsearch CR 时，它会将具有唯一 ID 的节点标记为 count 0，而不是省略它们。因此，用户可以使用 **clusterlogging** CR 将集群缩减到 3 个节点。[\(BZ#1879150\)](#)



#### 注意

在 OpenShift Logging 5.0 及更高版本中，Cluster Logging Operator 称为 Red Hat OpenShift Logging Operator。

- 在以前的版本中，当 **ClusterLogForwarder** 具有错误配置的 secret 时，Fluentd 收集器 Pod 会进入崩溃循环。当前发行版本解决了这个问题。现在，**ClusterLogForwarder** 会验证 secret，并报告其 status 字段中的任何错误。因此，它不会导致 Fluentd 收集器 Pod 出现崩溃。[\(BZ#1888943\)](#)
- 在以前的版本中，如果您将 **clusterlogging** 实例中的 Kibana 资源配置更新为 **resource{}**，则生成的 nil 映射会导致 panic，并将 OpenShift Elasticsearch Operator 的状态改为 **CrashLoopBackOff**。当前发行版本通过初始化映射解决了这个问题。[\(BZ#1889573\)](#)

- 在以前的版本中，当 ClusterLogForwarder 使用相同的 secret 具有多个输出时，fluentd 收集器 Pod 会进入崩溃循环。当前发行版本解决了这个问题。现在，多个输出可以共享一个 secret。  
([BZ#1890072](#))
- 在以前的版本中，如果您删除了 Kibana 路由，Cluster Logging Operator (CLO) 将无法恢复或重新创建它。现在，CLO 会监视路由，如果删除了路由，OpenShift Elasticsearch Operator 可以协调或重新创建路由。  
([BZ#1890825](#))
- 在以前的版本中，Cluster Logging Operator (CLO) 会尝试协调 Elasticsearch 资源，这依赖于红帽提供的 Elastic 自定义资源定义 (CRD)。试图列出未知类型会导致 CLO 退出其协调循环。这是因为 CLO 试图协调其所有受管资源，无论资源是否被定义。当前发行版本解决了这个问题。如果用户定义了受管存储，则 CLO 只会协调 OpenShift Elasticsearch Operator 提供的类型。因此，用户可以通过部署 CLO 来创建仅收集器集群日志记录的部署。  
([BZ#1891738](#))
- 在以前的版本中，因为 RFC 3164 的 LF GA syslog 实现，发送到远程 syslog 的日志与旧行为不兼容。当前发行版本解决了这个问题。AddLogSource 在 "message" 字段中添加有关日志源详情的详细信息。现在，发送到远程 syslog 的日志与旧行为兼容。  
([BZ#1891886](#))
- 在以前的版本中，Elasticsearch 滚动 pod 失败，并显示 **resource\_already\_exists\_exception** 错误。在 Elasticsearch 滚动 API 中，当创建下一个索引时，**\*-write** 别名没有更新来指向它。因此，当下次为该特定索引触发滚动 API 端点时，它会收到资源已存在的错误。当前发行版本解决了这个问题。现在，当在 **indexmanagement cronjobs** 中进行滚动时，如果创建了新索引，它会验证别名指向新的索引。这个行为可防止错误。如果集群已经收到这个错误，cronjob 会修复这个问题，以便后续运行可以正常工作。现在，执行滚动不再会产生异常。  
([BZ#1893992](#))
- 在以前的版本中，Fluent 会在日志堆栈功能正常的情况下停止发送日志。即使端点已可以使用，在一段时间内日志也无法发送到端点。如果最大的 backoff 时间过长且端点下线，会出现这种情况。当前的发行版本解决了这个问题，缩短了 max backoff 时间，因此日志会更早发布。  
([BZ#1894634](#))
- 在以前的版本中，省略 Elasticsearch 节点的存储大小会在 OpenShift Elasticsearch Operator 代码中造成 panic。这个 panic 在日志中被记录为：**Observed a panic: "invalid memory address or nil pointer dereference"**。panic 发生的原因是，存储大小是一个必需的字段，但在代码中没有对其进行检查。当前的发行版本解决了这个问题，在省略存储大小，不会出现 panic。相反，存储被默认为临时存储，并为用户生成一个相关的日志信息。  
([BZ#1899589](#))
- 在以前的版本中，**elasticsearch-rollover** 和 **elasticsearch-delete** pod 会处于 **Invalid JSON:** 或 **ValueError: No JSON object could be decoded** 的错误状态中。产生这个异常的原因是，无效的 JSON 输入没有异常处理程序。当前发行版本解决了这个问题，为无效 JSON 输入提供处理程序。因此，处理器会输出错误消息而不是异常 traceback，而且 **elasticsearch-rollover** 和 **elasticsearch-delete** 作业不会保留在这些错误状态。  
([BZ#1899905](#))
- 在以前的版本中，当将 Fluentd 部署为单机时，即使 **replicas** 值为 **0**，也会创建一个 Kibana Pod。这是因为即使没有 Elasticsearch 节点，Kibana 也会默认为 **1** 个 pod。当前发行版本解决了这个问题。现在，仅在有一个或多个 Elasticsearch 节点时，Kibana 才会被默认为 **1**。  
([BZ#1901424](#))
- 在以前的版本中，如果您删除了 secret，它无法被重新创建。虽然证书存在于 Operator 的一个本地磁盘，但是它们并没有被重新写入，因为它们还没有改变。也就是说，只有在证书有变化时才会被写入。当前发行版本解决了这个问题。如果证书有变化或未找到证书，它会重写 secret。现在，如果删除了 master-certs，它们会被替换。  
([BZ#1901869](#))
- 在以前的版本中，如果集群有多个同名自定义资源，则当 API 组没有完全限定时，资源会按字母顺序进行选择。因此，如果您同时安装了红帽的 OpenShift Elasticsearch Operator 和 OpenShift Elasticsearch Operator，您会在通过 must-gather 报告收集数据时看到失败。当前，通过确保

must-gathers 现在在收集集群自定义资源信息时使用完整的 API 组解决了这个问题。  
([BZ#1897731](#))

- 解决与证书生成相关的问题的早期程序错误修复会出现一个错误。尝试读取证书会导致重新生成证书，因为它们被认为没有存在。然后，这会触发 OpenShift Elasticsearch Operator 对 Elasticsearch 集群执行滚动升级，并可能具有不匹配的证书。此程序漏洞是因为 operator 错误地在工作目录中写入证书造成的。当前发行版本解决了这个问题。现在，operator 会持续将证书读取并写入同一工作目录，证书仅在需要时重新生成。(BZ#1905910)
- 在以前的版本中，查询根端点以检索 Elasticsearch 版本会收到 403 响应。403 响应会破坏之前版本中使用这个端点的所有服务。发生此错误的原因是，非管理员用户没有查询根端点并检索 Elasticsearch 版本所需的 **monitor** 权限。现在，非管理员用户可以查询 Elasticsearch 部署的根端点。(BZ#1906765)
- 在以前的版本中，在某些批量插入的情况下，Elasticsearch 代理在 fluentd 和 Elasticsearch 之间的连接可能会出现超时问题。因此，fluentd 无法发送信息，并在日志中记录一个 **Server returned nothing (no headers, no data)** 错误。当前发行版本解决了这个问题：它将 Elasticsearch 代理中的默认 HTTP 读写超时时间从 5 秒增加到一分钟。它还在 Elasticsearch 代理中提供命令行选项来控制字段中的 HTTP 超时。(BZ#1908707)
- 在以前的版本中，在一些情况下，OpenShift Container Platform 监控仪表板中没有 Red Hat OpenShift Logging/Elasticsearch 仪表板，因为仪表板配置资源指向不同的命名空间所有者，并导致 OpenShift Container Platform 垃圾回收该资源。现在，所有权引用已从 OpenShift Elasticsearch Operator 协调器配置中删除，日志记录仪表板会出现在控制台中。  
(BZ#1910259)
- 在以前的版本中，使用环境变量替换 Kibana 配置文件中值的代码不会考虑注释行。这导致用户无法覆盖 server.maxPayloadBytes 的默认值。当前发行版本解决了这个问题，方法是取消对默认值 server.maxPayloadBytes 的注释。现在，用户可以使用环境变量覆盖值，如相关文档所述。  
(BZ#1918876)
- 在以前的版本中，增大 Kibana 日志级别不会绕过删除无法迁移的索引的说明，这也会导致在包含 Kibana 用户电子邮件地址和 OAuth 令牌的 INFO 级别显示 GET 请求。当前发行版本解决了这个问题，它伪装这些字段，因此 Kibana 日志不会显示它们。(BZ#1925081)

#### 1.2.11.5. 已知问题

- 带有 **ruby-kafka-1.1.0** 和 **fluent-plugin-kafka-0.13.1** gems 的 Fluentd pod 与 Apache Kafka 版本 0.10.1.0 不兼容。

因此，日志转发到 Kafka 会失败并显示消息：**error\_class=Kafka::DeliveryFailed error="Failed to send messages to flux-openshift-v4/1"**

**ruby-kafka-0.7** gem 丢弃了对 Kafka 0.10 的支持，替代了对 Kafka 0.11 的原生支持。**ruby-kafka-1.0.0** gem 添加了对 Kafka 2.3 和 2.4 的支持。当前的 OpenShift Logging 版本测试并支持 Kafka 版本 2.4.1。

要临时解决这个问题，升级到受支持的 Apache Kafka 版本。

([BZ#1907370](#))

## 第 2 章 了解 RED HAT OPENSIFT LOGGING

作为集群管理员，您可以部署 OpenShift Logging 来聚合 OpenShift Container Platform 集群中的所有日志，如节点系统日志、应用程序容器日志和基础架构日志等。OpenShift Logging 汇总整个集群中的这些日志，并将其存储在默认日志存储中。您可以[使用 Kibana web 控制台来可视化日志数据](#)。

OpenShift Logging 聚合了以下类型的日志：

- **application** - 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）。
- **infrastructure** - 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。
- **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。



### 注意

由于内部 OpenShift Container Platform Elasticsearch 日志存储无法为审计日志提供安全存储，所以审计日志默认不会存储在内部 Elasticsearch 实例中。如果要[将审计日志发送到内部日志存储](#)，例如要在 Kibana 中查看审计日志，您必须使用 Log Forwarding API，如[把审计日志转发到日志存储](#)所示。

## 2.1. 关于部署 OPENSIFT LOGGING

OpenShift Container Platform 集群管理员可以使用 OpenShift Container Platform Web 控制台或 CLI 部署 OpenShift Logging 来安装 OpenShift Elasticsearch Operator 和 Operator and Red Hat OpenShift Logging Operator。安装 Operator 后，可创建 **ClusterLogging** 自定义资源 (CR) 以调度 OpenShift Logging pod 和支持 OpenShift Logging 所需的其他资源。Operator 负责部署、升级和维护 OpenShift Logging。

**ClusterLogging** CR 定义包括日志记录堆栈的所有组件在内的完整 OpenShift Logging 环境，以收集、存储和可视化日志。Red Hat OpenShift Logging Operator 监视 OpenShift Logging CR 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。

如需更多信息，请参阅[配置日志收集器](#)。

### 2.1.1. 关于 JSON OpenShift Container Platform Logging

您可以使用 JSON 日志记录配置 Log Forwarding API，将 JSON 字符串解析为结构化对象。您可以执行以下任务：

- 解析 JSON 日志
- 为 Elasticsearch 配置 JSON 日志数据
- 将 JSON 日志转发到 Elasticsearch 日志存储

如需更多信息，请参阅[关于 JSON 日志记录](#)。

### 2.1.2. 关于收集并存储 Kubernetes 事件

OpenShift Container Platform 事件路由器是一个 pod，它监视 Kubernetes 事件，并在 OpenShift Container Platform Logging 中记录它们以收集。您必须手动部署 Event Router。

如需更多信息，[请参阅关于收集和存储 Kubernetes 事件。](#)

### 2.1.3. 关于更新 OpenShift Container Platform Logging

OpenShift Container Platform 允许您更新 OpenShift Container Platform 日志记录。您必须在更新 OpenShift Container Platform Logging 时更新以下 Operator：

- Elasticsearch Operator
- Cluster Logging Operator

如需更多信息，[请参阅关于更新 OpenShift Container Platform Logging。](#)

### 2.1.4. 关于查看集群仪表盘

OpenShift Container Platform Logging 仪表盘包含 chart，在集群级别显示 Elasticsearch 实例的详情。这些图表可帮助您诊断和预测问题。

如需更多信息，[请参阅关于查看集群仪表盘。](#)

### 2.1.5. 关于 OpenShift Container Platform Logging 故障排除

您可以通过执行以下任务排除日志问题：

- 查看日志记录状态
- 查看日志存储的状态
- 了解日志记录警报
- 为红帽支持收集日志记录数据
- 关键警报故障排除

### 2.1.6. 关于卸载 OpenShift Container Platform Logging

您可以通过删除 ClusterLogging 自定义资源(CR)来停止日志聚合。在删除 CR 后，还有其它保留集群日志记录组件，您可以选择性地删除它们。

如需更多信息，[请参阅 卸载 OpenShift Container Platform Logging。](#)

### 2.1.7. 关于导出字段

日志记录系统导出字段。导出的字段出现在日志记录中，可从 Elasticsearch 和 Kibana 搜索。

如需更多信息，[请参阅关于导出字段。](#)

### 2.1.8. 关于 OpenShift Logging 组件

OpenShift Logging（日志记录）组件包括了一个要部署到 OpenShift Container Platform 集群中每个节点的收集器，用于收集所有节点和容器日志并将其写入日志存储。您可以使用集中 web UI 使用汇总的数据创建丰富的视觉化和仪表盘。

OpenShift Logging 的主要组件有：

- collection（收集） - 此组件从集群中收集日志，格式化日志并将其转发到日志存储。当前的实现是 Fluentd。
- log store（日志存储） - 存储日志的位置。默认是 Elasticsearch。您可以使用默认的 Elasticsearch 日志存储，或将日志转发到外部日志存储。默认日志存储经过优化并测试以进行简短存储。
- 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。

在本文中我们可能会互换使用日志存储或 Elasticsearch、可视化或 Kibana、collection 或 Fluentd、收集或 Fluentd。

### 2.1.9. 关于日志记录收集器

OpenShift Container Platform 使用 Fluentd 来收集容器和节点的日志数据。

默认情况下，日志收集器使用以下源：

- 所有系统的日志记录的 journald
- `/var/log/containers/*.log` 用于所有容器日志

如果您配置了日志收集器来收集审计日志，它会从 `/var/log/audit/audit.log` 中获取日志信息。

日志记录收集器是一个守护进程集，它将 pod 部署到每个 OpenShift Container Platform 节点。系统及基础架构日志由来自操作系统、容器运行时和 OpenShift Container Platform 的日志消息生成。应用程序日志由 CRI-O 容器引擎生成。Fluentd 从这些源收集日志，并在内部或外部转发 OpenShift Container Platform 中配置的日志。

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。这些信息不足以区分日志的来源。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[配置日志收集器](#)。

### 2.1.10. 关于日志存储

OpenShift Container Platform 使用 [Elasticsearch \(ES\)](#) 来存储和整理日志数据。另外，您可以使用日志转发功能使用 Fluentd 协议、syslog 协议或 OpenShift Container Platform 日志转发 API 将日志转发到外部日志存储。

OpenShift Logging 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。

Elasticsearch 将日志数据从 Fluentd 整理到数据存储或索引中，然后将每个索引分成多个碎片（称为 *shard*（分片）），分散到 Elasticsearch 集群中的一组 Elasticsearch 节点上。您可以配置 Elasticsearch 来为分片制作备份（称为 *replica*（副本）），Elasticsearch 也会分散到 Elasticsearch 节点上。**ClusterLogging**

自定义资源 (CR) 允许您指定如何复制分片，以提供数据冗余和故障恢复能力。您还可以使用 **ClusterLogging** CR 中的保留策略来指定不同类型的日志的保留的时长。



### 注意

索引模板的主分片数量等于 Elasticsearch 数据节点的数目。

Red Hat OpenShift Logging Operator 和相应的 OpenShift Elasticsearch Operator 确保每个 Elasticsearch 节点都使用带有自身存储卷的唯一部署来进行部署。在需要时，可以使用 **ClusterLogging** 自定义资源 (CR) 来增加 Elasticsearch 节点的数量。有关配置存储的注意事项，请参阅 [Elasticsearch 文档](#)。



### 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。管理员可以获取所有日志，开发人员只能访问自己项目中的日志。

如需更多信息，请参阅 [配置日志存储](#)。

## 2.1.11. 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、其他视觉化方式，来查询、发现和视觉化您的 Elasticsearch 数据。

如需更多信息，请参阅 [配置日志可视化工具](#)。

## 2.1.12. 关于事件路由

Event Router 是一个 pod，它监视 OpenShift Container Platform 事件，以便通过 OpenShift Logging 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。

如需更多信息，请参阅 [收集并存储 Kubernetes 事件](#)。

## 2.1.13. 关于日志转发

默认情况下，OpenShift Logging 将日志发送到 **ClusterLogging** 自定义资源 (CR) 中定义的默认内部 Elasticsearch 日志存储。如果要将日志转发到其他日志聚合器，您可以使用日志转发功能将日志发送到集群内部或外部的特定端点。

如需更多信息，请参阅 [将日志转发到第三方系统](#)。

## 第 3 章 安装 OPENSIFT LOGGING

您可以通过部署 OpenShift Elasticsearch 和 Red Hat OpenShift Logging Operator 来安装 OpenShift Logging。OpenShift Elasticsearch Operator 会创建和管理 OpenShift Logging 使用的 Elasticsearch 集群。Red Hat OpenShift Logging Operator 会创建和管理日志记录堆栈的组件。

将 OpenShift Logging 部署到 OpenShift Container Platform 的过程涉及以下任务：

- 参阅 [OpenShift Logging 存储注意事项](#)。
- 使用 OpenShift Container Platform [Web 控制台](#) 或 [CLI](#) 安装 OpenShift Elasticsearch Operator 和 Red Hat OpenShift Logging Operator。

### 3.1. 使用 WEB 控制台安装 OPENSIFT LOGGING

您可以使用 OpenShift Container Platform Web 控制台安装 OpenShift Elasticsearch 和 Red Hat OpenShift Logging Operator。



#### 注意

如果您不希望使用默认的 Elasticsearch 日志存储，您可以从 **ClusterLogging** 自定义资源 (CR) 中删除内部 Elasticsearch **logStore** 和 Kibana **visualization** 组件。删除这些组件是可选的，但会保存资源。如需更多信息，请参阅[在没有使用默认的 Elasticsearch 日志存储时删除未使用的组件](#)。

#### 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。



#### 注意

如果将本地卷用于持久性存储，请不要使用原始块卷，这在 **LocalVolume** 对象中的 **volumeMode: block** 描述。Elasticsearch 无法使用原始块卷。

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，在集群中添加更多 Elasticsearch 节点，而不是增加现有节点上的内存。

#### 流程

使用 OpenShift Container Platform Web 控制台安装 OpenShift Elasticsearch Operator 和 Red Hat OpenShift Logging Operator：

1. 安装 OpenShift Elasticsearch Operator:
  - a. 在 OpenShift Container Platform Web 控制台中，点击 **Operators** → **OperatorHub**。
  - b. 从可用的 Operator 列表中选择 **OpenShift Elasticsearch Operator**，然后点 **Install**。
  - c. 确定在 **Installation Mode** 下选择了 **All namespaces on the cluster**。
  - d. 确定在 **Installed Namespace** 下选择了 **openshift-operators-redhat**。

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

- e. 选择 **Enable operator recommended cluster monitoring on this namespace**  
这个选项在 Namespace 对象中设置 **openshift.io/cluster-monitoring: "true"** 标识。您必须设置这个选项，以确保集群监控提取 **openshift-operators-redhat** 命名空间。
  - f. 选择 **stable-5.x** 作为 **更新频道**。
  - g. 选择一个**批准策略**。
    - **Automatic** 策略允许 Operator Lifecycle Manager (OLM) 在有新版本可用时自动更新 Operator。
    - **Manual** 策略需要拥有适当凭证的用户批准 Operator 更新。
  - h. 点击 **Install**。
  - i. 通过切换到 **Operators → Installed Operators** 页来验证 OpenShift Elasticsearch Operator 已被安装。
  - j. 确定 **OpenShift Elasticsearch Operator** 在所有项目中被列出，请 **Status** 为 **Succeeded**。
2. 安装 Red Hat OpenShift Logging Operator :
- a. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
  - b. 从可用的 Operator 列表中选择 **Red Hat OpenShift Logging**，然后点 **Install**。
  - c. 确定在 **Installation Mode** 下选择了 **A specific namespace on the cluster**
  - d. 确定在 **Installed Namespace** 下的 **Operator recommended namespace** 是 **openshift-logging**。
  - e. 选择 **Enable operator recommended cluster monitoring on this namespace**  
这个选项在 Namespace 对象中设置 **openshift.io/cluster-monitoring: "true"** 标识。您必须选择这个选项，以确保集群监控提取 **openshift-logging** 命名空间。
  - f. 选择 **stable-5.x** 作为 **更新频道**。
  - g. 选择一个**批准策略**。
    - **Automatic** 策略允许 Operator Lifecycle Manager (OLM) 在有新版本可用时自动更新 Operator。
    - **Manual** 策略需要拥有适当凭证的用户批准 Operator 更新。
  - h. 点击 **Install**。
  - i. 通过切换到 **Operators → Installed Operators** 页来验证 Red Hat OpenShift Logging Operator 已被安装。
  - j. 确保 **openshift-logging** 项目中列出的 **Red Hat OpenShift Logging** 的 **Status** 为 **InstallSucceeded**。  
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** 项目中报告问题的 pod 的日志。

### 3. 创建 OpenShift Logging 实例：

- 切换到 **Administration → Custom Resource Definitions** 页面。
- 在 **Custom Resource Definitions** 页面上，点 **ClusterLogging**。
- 在 **Custom Resource Definition details** 页中，从 **Actions** 菜单中选择 **View Instances**。
- 在 **ClusterLoggings** 页中，点 **Create ClusterLogging**。  
您可能需要刷新页面来加载数据。
- 将 YAML 项中的代码替换为以下内容：



#### 注意

此默认 OpenShift Logging 配置应该可以支持各种环境。参阅有关调优和配置 OpenShift Logging 组件的主题，以了解有关可对 OpenShift Logging 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
    retentionPolicy: 4
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3 5
      storage:
        storageClassName: "<storage_class_name>" 6
        size: 200G
      resources: 7
        limits:
          memory: "16Gi"
        requests:
          memory: "16Gi"
    proxy: 8
      resources:
        limits:
          memory: 256Mi

```

```
requests:
  memory: 256Mi
  redundancyPolicy: "SingleRedundancy"
visualization:
  type: "kibana" 9
  kibana:
    replicas: 1
collection:
  logs:
    type: "fluentd" 10
    fluentd: {}
```

- 1 名称必须是 **instance**。
- 2 OpenShift Logging 管理状态。在一些数情况下，如果更改了 OpenShift Logging 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 OpenShift Logging 重新变为受管状态为止。
- 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。
- 4 指定 Elasticsearch 应该保留每个日志源的时间长度。输入一个整数和时间单位：周(w)、小时(h/H)、分钟(m)和秒。例如，**7d** 代表 7 天。时间超过 **maxAge** 的旧日志会被删除。您必须为每个日志源指定一个保留策略，否则不会为该源创建 Elasticsearch 索引。
- 5 指定 Elasticsearch 节点的数量。请参阅此列表后面的备注。
- 6 为 Elasticsearch 存储输入现有存储类的名称。为获得最佳性能，请指定分配块存储的存储类。如果没有指定存储类，OpenShift Logging 将使用临时存储。
- 7 根据需要指定 Elasticsearch 的 CPU 和内存请求。如果这些值留白，则 OpenShift Elasticsearch Operator 会设置默认值，它们应足以满足大多数部署的需要。内存请求的默认值为 **16Gi**，CPU 请求为 **1**。
- 8 根据需要指定 Elasticsearch 代理的 CPU 和内存请求。如果这些值留白，则 OpenShift Elasticsearch Operator 会设置默认值，它们应足以满足大多数部署的需要。内存请求的默认值为 **256Mi**，CPU 请求的默认值为 **100m**。
- 9 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[配置日志可视化工具](#)。
- 10 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 control plane 节点（也称为 master 节点）的最大数量是三个。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。control plane 节点执行集群范围的操作，如创建或删除索引、分片分配和跟踪节点。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
```

## 输出示例

```
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**。这会创建 OpenShift Logging 组件、**Elasticsearch** 自定义资源和组件以及 Kibana 接口。
4. 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 OpenShift Logging、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

## 其他资源

- [安装来自 OperatorHub 的 Operator](#)

## 3.2. 安装后的任务

如果计划使用 Kibana，必须 [手动创建 Kibana 索引模式和视觉化](#)，以便在 Kibana 中探索和视觉化数据。

如果您的集群网络供应商强制实施网络隔离，[允许包含 OpenShift Logging Operator 的项目之间的网络流量](#)。

## 3.3. 使用 CLI 安装 OPENSIFT LOGGING

您可以使用 OpenShift Container Platform CLI 安装 OpenShift Elasticsearch 和 Red Hat OpenShift Logging 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。



### 注意

如果将本地卷用于持久性存储，请不要使用原始块卷，这在 **LocalVolume** 对象中的 **volumeMode: block** 描述。Elasticsearch 无法使用原始块卷。

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，在集群中添加更多 Elasticsearch 节点，而不是增加现有节点上的内存。

## 流程

使用 CLI 安装 OpenShift Elasticsearch Operator 和 Red Hat OpenShift Logging Operator :

1. 为 OpenShift Elasticsearch Operator 创建命名空间。
  - a. 为 OpenShift 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** 命名空间可能包含社区 Operator，这些 Operator 不被信

任，并可能会发布与 OpenShift Container Platform 指标相同的名称，从而导致冲突。

- 2 字符串。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. 为 Red Hat OpenShift Logging Operator 创建命名空间：

a. 为 Red Hat OpenShift Logging Operator 创建一个命名空间对象 YAML 文件（例如，**olo-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 olo-namespace.yaml
```

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

a. 为 OpenShift 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. 创建一个 Subscription 对象 YAML 文件（例如 **eo-sub.yaml**）来订阅 OpenShift Elasticsearch 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" ❶
spec:
  channel: "stable-5.1" ❷
  installPlanApproval: "Automatic"
  source: "redhat-operators" ❸
  sourceNamespace: "openshift-marketplace"
  name: "elasticsearch-operator"
```

- ❶ 您必须指定 **openshift-operators-redhat** 命名空间。
- ❷ 指定 **5.0**、**stable** 或 **stable-5.<x>** 作为频道。请参见以下注释。
- ❸ 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。



### 注意

指定 **stable** 安装最新稳定版本的当前版本。使用带有 **installPlanApproval: "Automatic"** 的 **stable** 会自动将 Operator 升级到最新的稳定主版本和次发行版本。

指定 **stable-5.<x>** 会安装特定主版本的当前次版本。使用带有 **installPlanApproval: "Automatic"** 的 **stable-5.<x>** 会在您使用 **x** 指定的主版本中自动将 Operator 升级到最新的稳定次版本。

- d. 创建订阅对象：

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

例如：

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

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

- e. 验证 Operator 安装：

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

### 输出示例

```

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

```

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

#### 4. 通过创建以下对象来安装 Red Hat OpenShift Logging Operator :

- a. 为 Red Hat OpenShift Logging Operator 创建 Operator Group 对象 YAML 文件（如 **olo-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 olo-og.yaml
```

- c. 创建一个订阅对象 YAML 文件（例如 **olo-sub.yaml**）来订阅 Red Hat OpenShift Logging Operator 的命名空间。

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

- 1** 您必须指定 **openshift-logging** 命名空间。
- 2** 指定 **5.0**、**stable** 或 **stable-5.<x>** 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。

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

例如：

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

Red Hat OpenShift Logging Operator 已安装到 **openshift-logging** 命名空间中。

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

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

输出示例

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

5. 创建 OpenShift Logging 实例：

- a. 为 Red Hat OpenShift Logging Operator 创建实例对象 YAML 文件（如 **olo-instance.yaml**）：



## 注意

此默认 OpenShift Logging 配置应该可以支持各种环境。参阅有关调优和配置 OpenShift Logging 组件的主题，以了解有关可对 OpenShift Logging 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
    retentionPolicy: 4
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3 5
      storage:
        storageClassName: "<storage-class-name>" 6
        size: 200G
      resources: 7
        limits:
          memory: "16Gi"
        requests:
          memory: "16Gi"
      proxy: 8
        resources:
          limits:
            memory: 256Mi
          requests:
            memory: 256Mi
        redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" 9
    kibana:
      replicas: 1
  collection:
    logs:
      type: "fluentd" 10
      fluentd: {}
  
```

- 1 名称必须是 **instance**。
- 2 OpenShift Logging 管理状态。在一些数情况下，如果更改了 OpenShift Logging 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 OpenShift Logging 重新变为受管状态为止。将部署重新置于受管状态可能会使您所做的任何修改被恢复。

- 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。
- 4 指定 Elasticsearch 应该保留每个日志源的时间长度。输入一个整数和时间单位：周 (w)、小时(h/H)、分钟(m)和秒。例如，**7d** 代表 7 天。时间超过 **maxAge** 的旧日志会被删除。您必须为每个日志源指定一个保留策略，否则不会为该源创建 Elasticsearch 索引。
- 5 指定 Elasticsearch 节点的数量。请参阅此列表后面的备注。
- 6 为 Elasticsearch 存储输入现有存储类的名称。为获得最佳性能，请指定分配块存储的存储类。如果没有指定存储类，OpenShift Container Platform 只会使用临时存储部署 OpenShift Logging。
- 7 根据需要指定 Elasticsearch 的 CPU 和内存请求。如果这些值留空，OpenShift Elasticsearch Operator 会设置默认值，足以满足大多数部署的需要。内存请求的默认值为 **16Gi**，CPU 请求为 **1**。
- 8 根据需要指定 Elasticsearch 代理的 CPU 和内存请求。如果这些值留白，则 OpenShift Elasticsearch Operator 会设置默认值，它们应足以满足大多数部署的需要。内存请求的默认值为 **256Mi**，CPU 请求的默认值为 **100m**。
- 9 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 Pod 配置 CPU 和内存。如需更多信息，请参阅[配置日志可视化工具](#)。
- 10 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 control plane 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。control plane 节点执行集群范围的操作，如创建或删除索引、分片分配和跟踪节点。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
```

### 输出示例

```
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 olo-instance.yaml
```

这会创建 OpenShift Logging 组件、**Elasticsearch** 自定义资源和组件以及 Kibana 接口。

- 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 OpenShift Logging、Elasticsearch、Fluentd 和 Kibana 的 pod，类似于以下列表：

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

### 输出示例

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

## 3.4. 安装后的任务

如果计划使用 Kibana，必须 [手动创建 Kibana 索引模式和可视化](#)，以便在 Kibana 中探索和可视化数据。

如果您的集群网络供应商强制实施网络隔离，[允许包含 OpenShift Logging Operator 的项目之间的网络流量](#)。

### 3.4.1. 定义 Kibana 索引模式

索引模式定义了您要视觉化的 Elasticsearch 索引。要在 Kibana 中探索和可视化数据，您必须创建索引模式。

#### 先决条件

- 用户必须具有 **cluster-admin** 角色、**cluster-reader** 角色或这两个角色，才能在 Kibana 中查看 **infra** 和 **audit** 索引。默认 **kubeadmin** 用户具有查看这些索引的权限。  
如果可以查看 **default**、**kube-** 和 **openshift-** 项目中的 pod 和日志，则应该可以访问这些索引。  
您可以使用以下命令检查当前用户是否有适当的权限：

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

### 输出示例

yes



### 注意

默认情况下，审计日志不会存储在 OpenShift Container Platform 内部 Elasticsearch 实例中。要在 Kibana 中查看审计日志，您必须使用 Log Forward API 配置使用审计日志的 **default** 输出的管道。

- 在创建索引模式前，Elasticsearch 文档必须被索引。这会自动完成，但在一个新的或更新的集群中可能需要几分钟。

## 流程

在 Kibana 中定义索引模式并创建可视化：

1. 在 OpenShift Container Platform 控制台中点击 Application Launcher  并选择 **Logging**。
2. 点 **Management** → **Index Patterns** → **Create index pattern** 创建 Kibana 索引模式
  - 首次登录 Kibana 时，每个用户必须手动创建索引模式才能查看其项目的日志。用户必须创建一个名为 **app** 的索引模式，并使用 **@timestamp** 时间字段查看其容器日志。
  - 每个 admin 用户在首次登录 Kibana 时，必须使用 **@timestamp** 时间字段为 **app**、**infra** 和 **audit** 索引创建索引模式。
3. 从新的索引模式创建 Kibana 可视化。

### 3.4.2. 启用网络隔离时允许项目间的流量

集群网络供应商可能会强制实施网络隔离。如果是这样，您必须允许包含 OpenShift Logging 部署的 Operator 的项目间的网络流量。

网络隔离会阻止位于不同项目中的 pod 或服务之间的网络流量。OpenShift Logging 在 **openshift-operators-redhat** 项目中安装 *OpenShift Elasticsearch Operator*，在 **openshift-logging** 项目中安装 *Red Hat OpenShift Logging Operator*。因此，您必须允许这两个项目之间的流量。

OpenShift Container Platform 为默认 Container Network Interface (CNI) 网络供应商 (OpenShift SDN 和 OVN-Kubernetes) 提供两个支持的选择。这两个提供程序实施各种网络隔离策略。

OpenShift SDN 有三种模式：

#### 网络策略

这是默认的模式。如果没有定义策略，它将允许所有流量。但是，如果用户定义了策略，它们通常先拒绝所有流量，然后再添加例外。此过程可能会破坏在不同项目中运行的应用。因此，显式配置策略以允许从一个与日志记录相关的项目出口到另一个项目的流量。

#### 多租户

这个模式强制实施网络隔离。您必须加入两个与日志记录相关的项目，以允许它们之间的流量。

#### subnet

此模式允许所有流量。它不强制实施网络隔离。不需要操作。

OVN-Kubernetes 始终使用**网络策略**。因此，与 OpenShift SDN 一样，您必须配置策略，以允许流量从一个与日志相关的项目出口到另一个项目。

## 流程

- 如果您以多租户（multitenant）模式使用 OpenShift SDN，请加入这两个项目。例如：

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

- 否则，对于网络策略模式的 OpenShift SDN 以及 OVN-Kubernetes，请执行以下操作：

- a. 在 **openshift-operators-redhat** 命名空间中设置标签。例如：

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

- b. 在 **openshift-logging** 命名空间中创建一个网络策略对象，允许从 **openshift-operators-redhat**、**openshift-monitoring** 和 **openshift-ingress** 项目中的 ingress 到 openshift-logging 项目。例如：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-monitoring-ingress-operators-redhat
spec:
  ingress:
    - from:
      - podSelector: {}
      - from:
        - namespaceSelector:
            matchLabels:
              project: "openshift-operators-redhat"
      - from:
        - namespaceSelector:
            matchLabels:
              name: "openshift-monitoring"
      - from:
        - namespaceSelector:
            matchLabels:
              network.openshift.io/policy-group: ingress
  podSelector: {}
  policyTypes:
    - Ingress
```

## 其他资源

- [关于网络策略](#)
- [关于 OpenShift SDN 默认 CNI 网络供应商](#)
- [关于 OVN-Kubernetes 默认 Container Network Interface \(CNI\) 网络供应商](#)

## 第 4 章 配置日志部署

### 4.1. 集群日志记录自定义资源 (CR)

要配置 OpenShift Logging，您需要自定义 **ClusterLogging** 自定义资源 (CR)。

#### 4.1.1. 关于 ClusterLogging 自定义资源

要更改 OpenShift Logging 环境，请创建并修改 **ClusterLogging** 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。

以下示例显示了 OpenShift Logging 的典型自定义资源。

#### ClusterLogging 自定义资源 (CR) 示例

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

```
collection: 6
logs:
  type: "fluentd"
  fluentd:
    resources:
      limits:
        memory: 736Mi
      requests:
        cpu: 100m
        memory: 736Mi
```

- 1 名称必须是 **instance**。
- 2 CR 必须安装到 **openshift-logging** 命名空间。
- 3 Red Hat OpenShift Logging Operator 管理状态。当设置为 **非受管状态 (unmanaged)** 时，Operator 处于不被支持的状态且不会获取更新。
- 4 日志存储的设置，包括保留策略、节点数、资源请求和限值以及存储类。
- 5 视觉化工具的设置，包括资源请求和限值，以及 pod 副本数。
- 6 日志收集器的设置，包括资源请求和限值。

## 4.2. 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，并使用 Kubernetes pod 和项目元数据丰富这些日志。

您可以为日志收集器配置 CPU 和内存限值，并将 [日志收集器 Pod](#) 移到特定的节点。所有支持的对日志收集器的修改，均可通过 **ClusterLogging** 自定义资源 (CR) 中的 **spec.collection.log.fluentd** 小节来执行。

### 4.2.1. 不支持的配置

配置 OpenShift Logging 的支持方法是使用本文档中介绍的选项进行配置。请勿使用其他配置，因为不受支持。各个 OpenShift Container Platform 发行版本的配置范例可能会有所变化，只有掌握了所有可能的配置，才能稳妥应对这样的配置变化。如果使用本文档中描述的配置以外的配置，您的更改可能会丢失，因为 OpenShift Elasticsearch Operator 和 Red Hat OpenShift Logging Operator 会调节差异。按照设计，Operator 会默认将一切还原到定义的状态。



#### 注意

如果 *必须* 执行 OpenShift Container Platform 文档中没有描述的配置，您 *必须* 将 Red Hat OpenShift Logging Operator 或 OpenShift Elasticsearch Operator 设置为 **Unmanaged**。一个不受管理的 OpenShift Logging 环境 *不被支持*，且不会接收更新，直到 OpenShift Logging 返回到 **Managed**。

### 4.2.2. 查看日志记录收集器 Pod

您可以查看 Fluentd 日志记录收集器 Pod 以及它们正在运行的对应节点。Fluentd 日志记录收集器 Pod 仅在 **openshift-logging** 项目中运行。

## 流程

- 在 **openshift-logging** 项目中运行以下命令来查看 Fluentd 日志记录收集器 Pod 及其详情：

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

## 输出示例

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

### 4.2.3. 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. 编辑 **openshift-logging** 项目中的 **ClusterLogging** 自定义资源 (CR) ：

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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
...
spec:
  collection:
    logs:
      fluentd:
        resources:
          limits: 1
            memory: 736Mi
          requests:
            cpu: 100m
            memory: 736Mi
```

- 1 根据需要指定 CPU 和内存限值及请求。显示的值是默认值。

#### 4.2.4. 日志转发器的高级配置

OpenShift Logging 包含多个 Fluentd 参数，可用于调整 Fluentd 日志转发器的性能。通过这些参数，可以更改以下 Fluentd 行为：

- Fluentd 块和块缓冲的大小
- Fluentd 块清除行为
- Fluentd 块转发重试行为

Fluentd 在名为 *chunk*（块）的单个 blob 中收集日志数据。当 Fluentd 创建一个块时，块被视为处于 *stage*，在这个阶段，数据会被填充到块中。当块已满时，Fluentd 会将块移到 *queue*，在块被清除或将其写入其目的地前，数据会被保存在这里。有一些原因会导致 Fluentd 清除块，如网络问题或目的地的容量问题。如果无法清除块，Fluentd 会按照配置重试清除操作（flushing）。

在 OpenShift Container Platform 中，Fluentd 会使用 *exponential backoff* 方法来重试清理（flushing）操作，Fluentd 会加倍尝试重试清理操作之间的等待时间，这有助于减少到目的地的连接请求。您可以禁用 *exponential backoff* 的方法，并使用 *定期*重试的方法。它可在指定的时间间隔里重试 flush 块。默认情况下，Fluentd 会无限期重试块清除。在 OpenShift Container Platform 中，您无法更改无限期重试行为。

这些参数可帮助您权衡延迟和吞吐量之间的利弊。

- 要优化 Fluentd 的吞吐量，您可以使用这些参数通过配置较大的缓冲和队列、延迟清除以及设置重试间隔间的更多时间来减少网络数据包的数量。请注意，大型缓冲区需要在节点文件系统有更多空间。
- 要优化低延迟，您可以使用参数尽快发送数据，避免批量的构建，具有较短的队列和缓冲，并使用更频繁的清理和重试。

您可以使用 **ClusterLogging** 自定义资源（CR）中的以下参数配置 chunking 和 flushing 行为。然后这些参数会自动添加到 Fluentd 配置映射中，供 Fluentd 使用。



#### 注意

这些参数：

- 与大多数用户无关。默认设置应该就可以提供良好的一般性能。
- 只适用于对 Fluentd 配置和性能有详细了解的高级用户。
- 仅用于性能调整。它们对日志的功能性没有影响。

表 4.1. 高级 Fluentd 配置参数

参数	描述	默认
<b>chunkLimitSize</b>	每个块的最大值。当数据达到这个大小时，Fluentd 会停止将数据写入一个块。然后，Fluentd 将块发送到队列并打开一个新的块。	<b>8m</b>

参数	描述	默认
<b>totalLimitSize</b>	缓冲区的最大大小，即阶段（stage）和队列（stage）的总大小。如果缓冲区的大小超过这个值，Fluentd 会停止将数据添加到块，并显示错误失败。所有不在块中的数据都丢失。	<b>8G</b>
<b>flushInterval</b>	块清除之间的间隔。您可以使用 <b>s</b> （秒）、 <b>m</b> （分钟）、 <b>h</b> （小时）或 <b>d</b> （天）。	<b>1s</b>
<b>flushMode</b>	执行清除的方法： <ul style="list-style-type: none"> <li>● <b>lazy</b>: 基于 <b>timekey</b> 参数对块进行清理。您无法修改 <b>timekey</b> 参数。</li> <li>● <b>interval</b> : 基于 <b>flushInterval</b> 参数清理块。</li> <li>● <b>Immediate</b>: 在将数据添加到一个块后马上清理块。</li> </ul>	<b>interval</b>
<b>flushThreadCount</b>	执行块清除（flushing）的线程数量。增加线程数量可提高冲刷吞吐量，这会隐藏网络延迟的情况。	<b>2</b>
<b>overflowAction</b>	当队列满时块的行为： <ul style="list-style-type: none"> <li>● <b>throw_exception</b> : 发出一个异常并在日志中显示。</li> <li>● <b>block</b> : 停止对数据进行块除了，直到缓冲区已用完的问题被解决为止。</li> <li>● <b>drop_oldest_chunk</b> : 删除旧的块以接受新传入的块。旧块的价值比新块要小。</li> </ul>	<b>block</b>
<b>retryMaxInterval</b>	<b>exponential_backoff</b> 重试方法的最大时间（以秒为单位）。	<b>300s</b>

参数	描述	默认
<b>retryType</b>	flushing 失败时重试的方法： <ul style="list-style-type: none"> <li>● <b>exponential_backoff</b>：增加每次重新清理操作的间隔时间。Fluentd 会加倍到下一次重试需要等待的时间，直到达到 <b>retry_max_interval</b> 参数指定的值。</li> <li>● <b>periodic</b>：基于 <b>retryWait</b> 参数，定期重试清理操作。</li> </ul>	<b>exponential_backoff</b>
<b>retryWait</b>	下一次块清除前的时间（以秒为单位）。	<b>1s</b>

如需有关 Fluentd 块生命周期的更多信息，请参阅 [Fluentd 文档](#) 中的缓冲插件。

## 流程

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

```
$ oc edit ClusterLogging instance
```

2. 添加或修改以下任何参数：

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

① 请指定每个块在排队进行清除前的最大大小。

② 指定块清除之间间隔。

- 3 指定执行块清除的方法：**lazy**、**interval** 或 **immediate**。
- 4 指定用于块清除的线程数量。
- 5 指定当队列满时的块行为：**throw\_exception**、**block** 或 **drop\_oldest\_chunk**
- 6 指定使用 **exponential\_backoff** 块清理方法时的最大间隔时间（以秒为单位）。
- 7 指定当块清除失败时重试的类型：**exponential\_backoff** 或 **periodic**。
- 8 指定下一次块清除前的时间（以秒为单位）。
- 9 指定块缓冲区的最大大小。

3. 验证 Fluentd Pod 是否已重新部署：

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

4. 检查 **fluentd** 配置映射中的新值：

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

#### fluentd.conf 示例

```
<buffer>
@type file
path '/var/lib/fluentd/default'
flush_mode interval
flush_interval 5s
flush_thread_count 3
retry_type periodic
retry_wait 1s
retry_max_interval 300s
retry_timeout 60m
queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '32'}"
total_limit_size 32m
chunk_limit_size 8m
overflow_action throw_exception
</buffer>
```

#### 4.2.5. 如果不使用默认的 Elasticsearch 日志存储，请删除未使用的组件

作为管理员，在非常罕见的情况下，当您将日志转发到第三方日志存储且不使用默认的 Elasticsearch 存储时，您可以从日志集群中移除几个未使用的组件。

换句话说，如果没有使用默认的 Elasticsearch 日志存储，您可以从 **ClusterLogging** 自定义资源 (CR) 中删除内部 Elasticsearch **logStore** 和 Kibana **visualization** 组件。删除这些组件是可选的，但会保存资源。

#### 先决条件

- 验证您的日志转发程序没有将日志数据发送到默认的内部 Elasticsearch 集群。检查您用来配置日志转发的 **ClusterLogForwarder** CR YAML 文件。验证它没有指定 **default** 的 **outputRefs** 元素。例如：

```
outputRefs:
- default
```



### 警告

假定 **ClusterLogForwarder** CR 将日志数据转发到内部 Elasticsearch 集群，并从 **ClusterLogging** CR 中删除 **logStore** 组件。在这种情况下，内部 Elasticsearch 集群将不存在来存储日志数据。这会导致数据丢失。

## 流程

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

```
$ oc edit ClusterLogging instance
```

2. 如果存在，请从 **ClusterLogging** CR 中删除 **logStore** 和 **visualization** 小节。
3. 保留 **ClusterLogging** CR 的 **collection** 小节。结果应类似以下示例 :

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

4. 验证 Fluentd Pod 是否已重新部署 :

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

## 其他资源

- [将日志转发到第三方系统](#)

## 4.3. 配置日志存储

OpenShift Container Platform 使用 Elasticsearch 6 (ES) 来存储和整理日志数据。

您可以修改日志存储，包括：

- Elasticsearch 集群的存储
- 在集群中的数据节点间复制分片，从完整复制到不复制

- 外部访问 Elasticsearch 数据

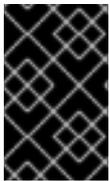
Elasticsearch 是内存密集型应用程序。每个 Elasticsearch 节点都需要至少 16G 内存来满足内存请求和限制的需要，除非 **ClusterLogging** 自定义资源中另有指定。最初的 OpenShift Container Platform 节点组可能不足以支持 Elasticsearch 集群。您必须在 OpenShift Container Platform 集群中添加额外的节点才能使用推荐或更高的内存运行，每个 Elasticsearch 节点最多可使用 64G 个内存。

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

### 4.3.1. 将审计日志转发到日志存储

由于内部 OpenShift Container Platform Elasticsearch 日志存储不为审计日志提供安全存储，所以默认审计日志不会存储在内部 Elasticsearch 实例中。

如果要将审计日志发送到内部日志存储，例如要在 Kibana 中查看审计日志，则必须使用 Log Forward API。



#### 重要

内部 OpenShift Container Platform Elasticsearch 日志存储不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 Logging 不遵循这些规范。

#### 流程

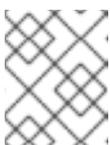
使用 Log Forward API 将审计日志转发到内部 Elasticsearch 实例：

1. 创建 **ClusterLogForwarder** CR YAML 文件或编辑现有的 CR：

- 创建 CR 以将所有日志类型发送到内部 Elasticsearch 实例。您可以在不进行任何更改的情况下使用以下示例：

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

- ① 管道（pipeline）定义使用指定输出转发的日志类型。默认输出将日志转发到内部 Elasticsearch 实例。



#### 注意

您必须在管道中指定所有三种类型的日志：应用程序、基础架构和审核。如果没有指定日志类型，这些日志将不会被存储并丢失。

- 如果您有一个现有的 **ClusterLogForwarder** CR，请将管道添加到审计日志的默认输出中。您不需要定义默认输出。例如：

```

apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
    - name: elasticsearch-insecure
      type: "elasticsearch"
      url: http://elasticsearch-insecure.messaging.svc.cluster.local
      insecure: true
    - name: elasticsearch-secure
      type: "elasticsearch"
      url: https://elasticsearch-secure.messaging.svc.cluster.local
      secret:
        name: es-audit
    - name: secureforward-offcluster
      type: "fluentdForward"
      url: https://secureforward.offcluster.com:24224
      secret:
        name: secureforward
  pipelines:
    - name: container-logs
      inputRefs:
        - application
      outputRefs:
        - secureforward-offcluster
    - name: infra-logs
      inputRefs:
        - infrastructure
      outputRefs:
        - elasticsearch-insecure
    - name: audit-logs
      inputRefs:
        - audit
      outputRefs:
        - elasticsearch-secure
        - default 1

```

- 1** 此管道除外部实例外，还会将审计日志发送到内部 Elasticsearch 实例。

## 其他资源

- 有关 Log Forwarding API 的更多信息，请参阅 [使用 Log Forwarding API 转发日志](#)。

### 4.3.2. 配置日志保留时间

您可以配置 *保留策略*，指定默认 Elasticsearch 日志存储保留三个日志源的索引的时长：基础架构日志、应用程序日志和审计日志。

要配置保留策略，您需要为 **ClusterLogging** 自定义资源 (CR) 中的每个日志源设置 **maxAge** 参数。CR 将这些值应用到 Elasticsearch 滚动调度，它决定 Elasticsearch 何时删除滚动索引。

如果索引与以下条件之一匹配，Elasticsearch 会滚动索引，移动当前的索引并创建新索引：

- 索引早于 **Elasticsearch** CR 中的 **rollover.maxAge** 值。
- 索引大小超过主分片数乘以 40GB 的值。
- 索引的 doc 数大于主分片数乘以 40960 KB 的值。

Elasticsearch 会根据您配置的保留策略删除滚动索引。如果您没有为任何日志源创建保留策略，则默认在 7 天后删除日志。

### 先决条件

- 必须安装 OpenShift Logging 和 OpenShift Elasticsearch Operator。

### 流程

配置日志保留时间：

1. 编辑 **ClusterLogging** CR，以添加或修改 **reservedPolicy** 参数：

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

- 1 指定 Elasticsearch 应该保留每个日志源的时间。输入一个整数和时间单位：周(w)、小时(h/H)、分钟(m)和秒。例如，**1d** 代表一天。时间超过 **maxAge** 的旧日志会被删除。默认情况下，日志会保留 7 天。

2. 您可以验证 **Elasticsearch** 自定义资源 (CR) 中的设置。例如，Red Hat OpenShift Logging Operator 更新了以下 **Elasticsearch** CR 以配置保留策略，包括设置以每八小时滚动基础架构日志的活跃索引，并在滚动后 7 天删除滚动的索引。OpenShift Container Platform 每 15 分钟检查一次，以确定是否需要滚动索引。

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

```
spec:
...
indexManagement:
  policies: ❶
    - name: infra-policy
      phases:
        delete:
          minAge: 7d ❷
          hot:
            actions:
              rollover:
                maxAge: 8h ❸
            pollInterval: 15m ❹
...

```

- ❶ 对于每个日志源，保留策略代表何时删除和滚动该源的日志。
- ❷ 什么时候 OpenShift Container Platform 删除滚动索引。此设置是在 **ClusterLogging** CR 中设置的 **maxAge**。
- ❸ 当滚动索引时，OpenShift Container Platform 需要考虑的索引年龄。此值由 **ClusterLogging** CR 中的 **maxAge** 决定。
- ❹ OpenShift Container Platform 什么时候检查应该检查滚动索引。这是默认设置，不可更改。



### 注意

不支持修改 **Elasticsearch** CR。对保留策略的所有更改都必须在 **ClusterLogging** CR 中进行。

OpenShift Elasticsearch Operator 部署 cron job，以使用定义的策略为每个映射滚动索引,并使用 **pollInterval** 调度。

```
$ oc get cronjob
```

### 输出示例

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

### 4.3.3. 为日志存储配置 CPU 和内存请求

每个组件规格都允许调整 CPU 和内存请求。您不应该手动调整这些值，因为 OpenShift Elasticsearch Operator 会设置适当的值以满足环境的要求。



### 注意

在大型集群中，Elasticsearch 代理容器的默认内存限值可能不足，从而导致代理容器被 OOMKilled。如果您遇到这个问题，请提高 Elasticsearch 代理的内存请求和限值。

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 OpenShift Logging 和 Elasticsearch。

## 流程

1. 编辑 `openshift-logging` 项目中的 `ClusterLogging` 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: 1
      resources:
        limits: 2
          memory: "32Gi"
        requests: 3
          cpu: "1"
          memory: "16Gi"
      proxy: 4
        resources:
          limits:
            memory: 100Mi
          requests:
            memory: 100Mi
```

- 1** 根据需要指定 Elasticsearch 的 CPU 和内存请求。如果这些值留白，则 OpenShift Elasticsearch Operator 会设置默认值，它们应足以满足大多数部署的需要。内存请求的默认值为 **16Gi**，CPU 请求为 **1**。
- 2** pod 可以使用的最大资源量。
- 3** 调度 pod 所需的最少资源。
- 4** 根据需要指定 Elasticsearch 代理的 CPU 和内存请求。如果这些值留空，OpenShift Elasticsearch Operator 会设置默认值，足以满足大多数部署的需要。内存请求的默认值为 **256Mi**，CPU 请求的默认值为 **100m**。

在调整 Elasticsearch 内存量时，**请求和限值** 应使用相同的值。

例如：

```
resources:
```

```
limits: 1
  memory: "32Gi"
requests: 2
  cpu: "8"
  memory: "32Gi"
```

- 1 资源的最大数量。
- 2 要求的最小数量。

Kubernetes 一般遵循节点配置，不允许 Elasticsearch 使用指定的限值。为 **请求 (request)** 和 **限值 (limit)** 设置相同的值可确保 Elasticsearch 可以使用您想要的内存，假设节点具有可用内存。

#### 4.3.4. 为日志存储配置复制策略

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

- 必须安装 OpenShift Logging 和 Elasticsearch。

##### 流程

1. 编辑 **openshift-logging** 项目中的 **ClusterLogging** 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。

#### 4.3.5. 缩减 Elasticsearch pod

减少集群中的 Elasticsearch pod 数量可能会导致数据丢失或 Elasticsearch 性能下降。

如果缩减，应该一次缩减一个 pod，并允许集群重新平衡分片和副本。Elasticsearch 健康状态返回绿色后，您可以根据另一个 pod 进行缩减。



### 注意

如果 Elasticsearch 集群设置为 **ZeroRedundancy**，则不应缩减 Elasticsearch pod。

#### 4.3.6. 为日志存储配置持久性存储

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 OpenShift Logging 和 Elasticsearch。

### 流程

1. 编辑 **ClusterLogging** CR，将集群中的每个数据节点指定为绑定到持久性卷声明。

```
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。

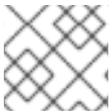


### 注意

如果将本地卷用于持久性存储，请不要使用原始块卷，这在 **LocalVolume** 对象中的 **volumeMode: block** 描述。Elasticsearch 无法使用原始块卷。

#### 4.3.7. 为 emptyDir 存储配置日志存储

您可以将 emptyDir 与日志存储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日志存储，数据将会丢失。

#### 先决条件

- 必须安装 OpenShift Logging 和 Elasticsearch。

#### 流程

1. 编辑 **ClusterLogging** CR 以指定 emptyDir:

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

#### 4.3.8. 执行 Elasticsearch 集群滚动重启

在更改 **elasticsearch** 配置映射或任何 **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。

#### 先决条件

- 必须安装 OpenShift Logging 和 Elasticsearch。

#### 流程

执行集群滚动重启：

1. 进入 **openshift-logging** 项目：

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

- 获取 Elasticsearch Pod 的名称：

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

- 缩减 Fluentd pod，以便它们停止向 Elasticsearch 发送新日志：

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

- 使用 OpenShift Container Platform `es_util` 工具执行分片同步刷新，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 -- es_util --query="_flush/synced" -XPOST
```

例如：

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

### 输出示例

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

- 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 -d '{ "persistent": { "cluster.routing.allocation.enable" : "primaries" } }'
```

例如：

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

### 输出示例

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

- 完成后，会在每个部署中都有一个 ES 集群：

- 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
```

## 输出示例

```
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-
```

## 输出示例

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. 部署完成后，重置 Pod 以禁止推出部署：

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

例如：

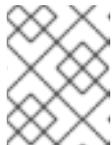
```
$ oc rollout pause deployment/elasticsearch-cdm-0-1
```

## 输出示例

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

- c. 检查 Elasticsearch 集群是否处于 **green** 或 **yellow** 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" : "yellow", 1
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 8,
  "active_shards" : 16,
  "relocating_shards" : 0,
```

```

"initializing_shards" : 0,
"unassigned_shards" : 1,
"delayed_unassigned_shards" : 0,
"number_of_pending_tasks" : 0,
"number_of_in_flight_fetch" : 0,
"task_max_waiting_in_queue_millis" : 0,
"active_shards_percent_as_number" : 100.0
}

```

**1** 在继续操作前，请确保此参数值为 **green** 或者 **yellow**。

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

8. 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 -d '{"persistent": {"cluster.routing.allocation.enable" : "all" }
}'

```

例如：

```

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

```

#### 输出示例

```

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

```

9. 扩展 Fluentd Pod，以便它们向 Elasticsearch 发送新日志。

```

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

```

### 4.3.9. 将日志存储服务公开为路由

默认情况下，无法从日志记录集群外部访问部署了 OpenShift Logging 的日志存储。您可以启用一个 re-encryption termination 模式的路由，以实现外部对日志存储服务的访问来获取数据。

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 证书以从外部访问日志存储。然后，使用包含以下内容的 cURL 请求访问托管日志存储服务的节点：

- **Authorization: Bearer \${token}**
- Elasticsearch 重新加密路由和 [Elasticsearch API 请求](#)。

在内部，可以使用日志存储集群 IP 访问日志存储服务。您可以使用以下命令之一获取它：

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

### 输出示例

```
172.30.183.229
```

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

### 输出示例

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

您可以使用类似如下的命令检查集群 IP 地址：

```
$ 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"
```

### 输出示例

```
% 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 OpenShift 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

对外部公开日志存储：

1. 进入 **openshift-logging** 项目：

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

2. 从日志存储提取 CA 证书并写入 **admin-ca** 文件：

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

## 输出示例

```
admin-ca
```

3. 以 YAML 文件形式创建日志存储服务的路由：

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 添加日志存储 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，将日志存储 CA 证书添加到您在上一步中创建的路由 YAML 中：

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

c. 创建路由：

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

## 输出示例

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

4. 检查是否公开了 Elasticsearch 服务：

a. 获取此服务帐户的令牌，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}"
```

其响应类似于如下：

## 输出示例

```
{
  "name" : "elasticsearch-cdm-i40ktba0-1",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "0eY-tJzcR3K0dpgeMJo-MQ",
  "version" : {
    "number" : "6.8.1",
    "build_flavor" : "oss",
    "build_type" : "zip",
    "build_hash" : "Unknown",
    "build_date" : "Unknown",
    "build_snapshot" : true,
    "lucene_version" : "7.7.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "<tagline>" : "<for search>"
}
```

## 4.4. 配置日志可视化工具

OpenShift Container Platform 使用 Kibana 显示 OpenShift Logging 收集的日志数据。

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

### 4.4.1. 配置 CPU 和内存限值

OpenShift Logging 组件允许对 CPU 和内存限值进行调整。

#### 流程

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

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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
...
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources: ①
      limits:
        memory: 16Gi
      requests:
```

```

    cpu: 200m
    memory: 16Gi
  storage:
    storageClassName: "gp2"
    size: "200G"
    redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources: ❷
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    proxy:
      resources: ❸
      limits:
        memory: 100Mi
      requests:
        cpu: 100m
        memory: 100Mi
    replicas: 2
  collection:
    logs:
      type: "fluentd"
      fluentd:
        resources: ❹
        limits:
          memory: 736Mi
        requests:
          cpu: 200m
          memory: 736Mi

```

- ❶ 根据需要指定日志存储的 CPU 和内存限值及请求。对于 Elasticsearch，您必须调整请求值和限制值。
- ❷ ❸ 根据需要为日志 visualizer 指定 CPU 和内存限值和请求。
- ❹ 根据需要指定日志收集器的 CPU 和内存限值及请求。

#### 4.4.2. 为日志可视化器节点扩展冗余性

您可以扩展托管日志视觉化器的 pod 以增加它的冗余性。

##### 流程

1. 编辑 **openshift-logging** 项目中的 **ClusterLogging** 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。

## 4.5. 配置 OPENSIFT LOGGING 存储

Elasticsearch 是内存密集型应用程序。默认 OpenShift Logging 安装为内存请求和内存限值部署 16G 内存。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，但在生产环境中不建议这样做。

### 4.5.1. OpenShift Logging 和 OpenShift Container Platform 的存储注意事项

每个 Elasticsearch 部署配置都需要一个持久性卷。在 OpenShift Container Platform 中，这可以使用 PVC 来实现。



#### 注意

如果将本地卷用于持久性存储，请不要使用原始块卷，这在 **LocalVolume** 对象中的 **volumeMode: block** 描述。Elasticsearch 无法使用原始块卷。

OpenShift Elasticsearch Operator 使用 Elasticsearch 资源名称为 PVC 命名。

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

Elasticsearch 需要足够内存来执行大型合并操作。如果没有足够的内存，它将会变得无响应。要避免这个问题，请评估应用程序日志数据的数量，并分配大约两倍的可用存储容量。

默认情况下，当存储容量为 85% 满时，Elasticsearch 会停止向节点分配新数据。90% 时，Elasticsearch 会在可能的情况下将现有分片重新定位到其他节点。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。您可以修改这些默认值。虽然警报使用相同的默认值，但无法在警报中更改这些值。

### 4.5.2. 其他资源

- [为日志存储配置持久性存储](#)

## 4.6. 为 OPENSIFT LOGGING 组件配置 CPU 和内存限值

您可以根据需要配置每个 OpenShift Logging 组件的 CPU 和内存限值。

#### 4.6.1. 配置 CPU 和内存限值

OpenShift Logging 组件允许对 CPU 和内存限值进行调整。

##### 流程

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

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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
...
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources: ①
      limits:
        memory: 16Gi
      requests:
        cpu: 200m
        memory: 16Gi
    storage:
      storageClassName: "gp2"
      size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources: ②
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    proxy:
      resources: ③
      limits:
        memory: 100Mi
      requests:
        cpu: 100m
        memory: 100Mi
    replicas: 2
  collection:
```

```

logs:
  type: "fluentd"
  fluentd:
    resources: 4
    limits:
      memory: 736Mi
    requests:
      cpu: 200m
      memory: 736Mi

```

- 1 根据需要指定日志存储的 CPU 和内存限值及请求。对于 Elasticsearch，您必须调整请求值和限制值。
- 2 3 根据需要为日志 visualizer 指定 CPU 和内存限值及请求。
- 4 根据需要指定日志收集器的 CPU 和内存限值及请求。

## 4.7. 使用容忍度来控制 OPENSIFT LOGGING POD 放置

您可以使用污点和容忍来确保 OpenShift Logging 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。

### 具有容忍的 OpenShift Logging 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
      tolerations: 1
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
    resources:
      limits:
        memory: 16Gi
      requests:
        cpu: 200m
        memory: 16Gi

```

```

    storage: {}
    redundancyPolicy: "ZeroRedundancy"
  visualization:
    type: "kibana"
    kibana:
      tolerations: ❷
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
    resources:
      limits:
        memory: 2Gi
      requests:
        cpu: 100m
        memory: 1Gi
    replicas: 1
  collection:
    logs:
      type: "fluentd"
      fluentd:
        tolerations: ❸
        - key: "logging"
          operator: "Exists"
          effect: "NoExecute"
          tolerationSeconds: 6000
    resources:
      limits:
        memory: 2Gi
      requests:
        cpu: 100m
        memory: 1Gi

```

- ❶ 此容忍度添加到 Elasticsearch Pod。
- ❷ 此容忍度添加到 Kibana Pod。
- ❸ 此容忍度添加到日志记录收集器 Pod。

#### 4.7.1. 使用容忍度来控制日志存储 pod 放置

您可以通过在 pod 上使用容忍度来控制日志存储 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 **ClusterLogging** 自定义资源（CR）将容忍度应用到日志存储 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，可以确保仅日志存储 pod 能够在该节点上运行。

默认情况下，日志存储 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 OpenShift Logging 和 Elasticsearch。

## 流程

1. 使用以下命令，将污点添加到要在其上调度 OpenShift Logging 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 **ClusterLogging** 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间。

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。

### 4.7.2. 使用容忍度来控制日志可视化 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 **ClusterLogging** 自定义资源（CR）将容忍度应用到日志可视化 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 Kibana Pod 能够在该节点上运行。

## 先决条件

- 必须安装 OpenShift 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可视化 pod：

```
$ 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 **ClusterLogging** 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。具有此容忍度的 pod 可以调度到 **node1** 上。

### 4.7.3. 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 **ClusterLogging** 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忍度来确保 pod 不会因为内存和 CPU 问题而被驱逐。

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 OpenShift 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 **ClusterLogging** 自定义资源 (CR) 的 **collection** 小节，以配置日志记录收集器 Pod 的容忍度：

```
collection:
  logs:
    type: "fluentd"
    fluentd:
      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。具有此容忍度的 pod 可以调度到 **node1** 上。

### 4.7.4. 其他资源

- [使用节点污点控制 pod 放置。](#)

## 4.8. 使用节点选择器移动 OPENSIFT LOGGING 资源

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

### 4.8.1. 移动 OpenShift Logging 资源

您可以配置 Cluster Logging Operator，将 OpenShift Logging 组件（如 Elasticsearch 和 Kibana）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。

### 先决条件

- 必须安装 OpenShift 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 **ClusterLogging** 自定义资源 (CR) :

```
$ oc edit ClusterLogging instance

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

...

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

...
```

- 1 2 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

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
```

### 输出示例

```
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
```

### 输出示例

```
NAME                                STATUS ROLES    AGE  VERSION
ip-10-0-133-216.us-east-2.compute.internal Ready  master    60m  v1.20.0
ip-10-0-139-146.us-east-2.compute.internal Ready  master    60m  v1.20.0
ip-10-0-139-192.us-east-2.compute.internal Ready  worker    51m  v1.20.0
ip-10-0-139-241.us-east-2.compute.internal Ready  worker    51m  v1.20.0
ip-10-0-147-79.us-east-2.compute.internal Ready  worker    51m  v1.20.0
ip-10-0-152-241.us-east-2.compute.internal Ready  master    60m  v1.20.0
ip-10-0-139-48.us-east-2.compute.internal Ready  infra     51m  v1.20.0
```

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
```

### 输出示例

```
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，编辑 **ClusterLogging** CR 以添加节点选择器：

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

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

- ❶ 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
```

### 输出示例

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

- 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
```

### 输出示例

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>	<none>			

- 片刻后，原始 Kibana Pod 将被删除。

```
$ oc get pods
```

### 输出示例

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

## 4.9. 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 **RateLimitIntervalSec=30s** 和 **RateLimitBurst=10000**（如有必要甚至更高）以防止日志丢失条目。

### 4.9.1. 为 OpenShift Logging 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 OpenShift Logging 在不丢弃日志的情况下不使用过量资源。

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
RateLimitIntervalSec=30s
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 **RateLimitIntervalSec=30s** 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. 运行以下命令，将 **journal.conf** 文件转换为 base64，并将其存储在名为 **jrnl\_cnf** 的变量中：

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

3. 创建一个 **MachineConfig** 对象，其中包含上一步中创建的 **jrnl\_cnf** 变量。以下示例命令为 worker 创建 **MachineConfig** 对象：

```
$ cat << EOF > ./40-worker-custom-journald.yaml 1
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
```

```

machineconfiguration.openshift.io/role: worker ❷
name: 40-worker-custom-journald ❸
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 3.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,${jrnل_cnf} ❹
          verification: {}
        filesystem: root
        mode: 0644 ❺
        path: /etc/systemd/journald.conf.d/custom.conf
      osImageURL: ""
EOF

```

- ❶ 可选：对于 control plane（也称为 master）节点，您可以提供文件名为 **40-master-custom-journald.yaml**。
- ❷ 可选：对于 control plane（也称为 master）节点，将角色提供为 **master**。
- ❸ 可选：对于 control plane（也称为 master）节点，您可以提供名称为 **40-master-custom-journald**。
- ❹ 可选：要在 **journald.conf** 文件中包含参数的静态副本，请将 **\${jrnل\_cnf}** 替换为 **echo \$jrnل\_cnf** 命令的输出。
- ❺ 为 **journal.conf** 文件设置权限。建议把选项设置为 **0644**。

4. 运行以下命令来创建机器配置：

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

控制器检测到新的 **MachineConfig** 对象，并生成新的 **rendered-worker-<hash>** 版本。

5. 运行以下命令，监控新配置在每个节点中的推出状态：

```
$ oc describe machineconfigpool/<node> ❶
```

- ❶ 将节点指定为 **master** 或 **worker**。

### worker 的输出示例

```

Name:      worker
Namespace:
Labels:    machineconfiguration.openshift.io/mco-built-in=

```

```

Annotations: <none>
API Version: machineconfiguration.openshift.io/v1
Kind:      MachineConfigPool
...
Conditions:
  Message:
  Reason:      All nodes are updating to rendered-worker-
913514517bcea7c93bd446f4830bc64e

```

## 4.10. 维护和支持

### 4.10.1. 不支持的配置

配置 OpenShift Logging 的支持方法是使用本文档中介绍的选项进行配置。请勿使用其他配置，因为不受支持。各个 OpenShift Container Platform 发行版本的配置范例可能会有所变化，只有掌握了所有可能的配置，才能稳妥应对这样的配置变化。如果使用本文档中描述的配置以外的配置，您的更改可能会丢失，因为 OpenShift Elasticsearch Operator 和 Red Hat OpenShift Logging Operator 会调节差异。按照设计，Operator 会默认将一切还原到定义的状态。



#### 注意

如果 *必须* 执行 OpenShift Container Platform 文档中没有描述的配置，您 *必须* 将 Red Hat OpenShift Logging Operator 或 OpenShift Elasticsearch Operator 设置为 **Unmanaged**。一个不受管理的 OpenShift Logging 环境 *不被支持*，且不会接收更新，直到 OpenShift Logging 返回到 **Managed**。

### 4.10.2. 不支持的配置

您必须将 Red Hat OpenShift Logging Operator 设置为非受管状态，才能修改以下组件：

- **Elasticsearch** CR
- Kibana 部署
- **fluent.conf** 文件
- Fluentd 守护进程集

您必须将 OpenShift Elasticsearch Operator 设置为非受管状态才能修改以下组件：

- Elasticsearch 部署文件。

明确不支持的情形包括：

- **配置默认日志轮转。** 您无法修改默认的日志轮转配置。
- **配置所收集日志的位置。** 您无法更改日志收集器输出文件的位置，默认为 `/var/log/fluentd/fluentd.log`。
- **日志收集节流。** 您不能减慢日志收集器读取日志的速度。
- **使用环境变量配置日志记录收集器。** 您不能使用环境变量来修改日志收集器。

- **配置日志收集器规范日志的方式。**您无法修改默认日志规范化。

### 4.10.3. 非受管 Operator 的支持策略

Operator 的 **管理状态** 决定了一个 Operator 是否按设计积极管理集群中其相关组件的资源。如果 Operator 设置为 **非受管 (unmanaged)** 状态，它不会响应配置更改，也不会收到更新。

虽然它可以在非生产环境集群或调试过程中使用，但处于非受管状态的 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。

可使用以下方法将 Operator 设置为非受管状态：

- **独立 Operator 配置**

独立 Operator 的配置中具有 **managementState** 参数。这可以通过不同的方法来访问，具体取决于 Operator。例如，Red Hat OpenShift Logging Operator 通过修改它管理的自定义资源 (CR) 来达到此目的，而 Cluster Samples Operator 使用了集群范围配置资源。

将 **managementState** 参数更改为 **Unmanaged** 意味着 Operator 不会主动管理它的资源，也不会执行与相关组件相关的操作。一些 Operator 可能不支持此管理状态，因为它可能会损坏集群，需要手动恢复。



#### 警告

将独立 Operator 更改为**非受管**状态会导致不支持该特定组件和功能。报告的问题必须在 **受管 (Managed)** 状态中可以重复出现才能继续获得支持。

- **Cluster Version Operator (CVO) 覆盖**

可将 **spec.overrides** 参数添加到 CVO 配置中，以便管理员提供对组件的 CVO 行为覆盖的列表。将一个组件的 **spec.overrides[].unmanaged** 参数设置为 **true** 会阻止集群升级并在设置 CVO 覆盖后提醒管理员：

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



#### 警告

设置 CVO 覆盖会使整个集群处于不受支持状态。在删除所有覆盖后，必须可以重现报告的问题方可获得支持。

## 第 5 章 查看资源的日志

您可以使用 OpenShift CLI (oc) 和 Web 控制台查看各种资源的日志，如构建、部署和 pod。



### 注意

资源日志是一个默认功能，可提供有限的日志查看功能。为增强日志检索和查看体验，建议您安装 [OpenShift Logging](#)。OpenShift Logging 将 OpenShift Container Platform 集群中的所有日志（如节点系统审计日志、应用程序容器日志和基础架构日志）聚合到专用日志存储中。然后，您可以通过 [Kibana 接口](#) 查询、发现和可视化日志数据。资源日志无法访问 OpenShift Logging 日志存储。

### 5.1. 查看资源日志

您可以在 OpenShift CLI (oc) 和 Web 控制台中查看各种资源的日志。日志从日志的尾部或末尾读取。

#### 先决条件

- 访问 OpenShift CLI (oc) 。

#### 流程 (UI)

1. 在 OpenShift Container Platform 控制台中，导航到 **Workloads** → **Pods**，或通过您要调查的资源导航到 pod。



### 注意

有些资源（如构建）没有直接查询的 pod。在这种情况下，您可以在资源的 **Details** 页面中找到 **Logs** 链接。

2. 从下拉菜单中选择一个项目。
3. 点您要调查的 pod 的名称。
4. 点击 **Logs**。

#### 流程 (CLI)

- 查看特定 pod 的日志：

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

其中：

**-f**

可选：指定输出是否遵循要写到日志中的内容。

**<pod\_name>**

指定 pod 的名称。

**<container\_name>**

可选：指定容器的名称。当 pod 具有多个容器时，您必须指定容器名称。

例如：

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

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

输出的日志文件内容。

- 查看特定资源的日志：

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

- 1** 指定资源类型和名称。

例如：

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

输出的日志文件内容。

## 第 6 章 使用 KIBANA 查看集群日志

OpenShift Logging 包括一个 Web 控制台，用于可视化收集的日志数据。目前，OpenShift Container Platform 部署 Kibana 控制台以进行可视化。

通过日志可视化工具，您可以使用以下数据进行以下操作：

- 使用 **Discover** 标签页搜索并浏览数据。
- 使用 **Visualize** 标签页对数据进行图表显示。
- 使用 **Dashboard** 标签页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

默认情况下，审计日志不会存储在 OpenShift Container Platform 内部 Elasticsearch 实例中。要在 Kibana 中查看审计日志，您必须使用 [Log Forwarding API](#) 配置使用审计日志的 **default** 输出的管道。

### 6.1. 定义 KIBANA 索引模式

索引模式定义了您要视觉化的 Elasticsearch 索引。要在 Kibana 中探索和可视化数据，您必须创建索引模式。

#### 先决条件

- 用户必须具有 **cluster-admin** 角色、**cluster-reader** 角色或这两个角色，才能在 Kibana 中查看 **infra** 和 **audit** 索引。默认 **kubeadmin** 用户具有查看这些索引的权限。如果可以查看 **default**、**kube-** 和 **openshift-** 项目中的 pod 和日志，则应该可以访问这些索引。您可以使用以下命令检查当前用户是否有适当的权限：

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

#### 输出示例

```
yes
```



### 注意

默认情况下，审计日志不会存储在 OpenShift Container Platform 内部 Elasticsearch 实例中。要在 Kibana 中查看审计日志，您必须使用 Log Forward API 配置使用审计日志的 **default** 输出的管道。

- 在创建索引模式前，Elasticsearch 文档必须被索引。这会自动完成，但在一个新的或更新的集群中可能需要几分钟。

#### 流程

在 Kibana 中定义索引模式并创建可视化：

1. 在 OpenShift Container Platform 控制台中点击 Application Launcher  并选择 **Logging**。

## 2. 点 Management → Index Patterns → Create index pattern 创建 Kibana 索引模式

- 首次登录 Kibana 时，每个用户必须手动创建索引模式才能查看其项目的日志。用户必须创建一个名为 **app** 的索引模式，并使用 **@timestamp** 时间字段查看其容器日志。
- 每个 admin 用户在首次登录 Kibana 时，必须使用 **@timestamp** 时间字段为 **app**、**infra** 和 **audit** 索引创建索引模式。

## 3. 从新的索引模式创建 Kibana 视觉化。

## 6.2. 在 KIBANA 中查看集群日志

您可以在 Kibana web 控制台中查看集群日志。在 Kibana 中查看和视觉化您的数据的方法，它们超出了本文档的范围。如需更多信息，请参阅 [Kibana 文档](#)。

### 先决条件

- 必须安装 OpenShift Logging 和 Elasticsearch。
- Kibana 索引模式必须存在。
- 用户必须具有 **cluster-admin** 角色、**cluster-reader** 角色或这两个角色，才能在 Kibana 中查看 **infra** 和 **audit** 索引。默认 **kubeadmin** 用户具有查看这些索引的权限。如果可以查看 **default**、**kube-** 和 **openshift-** 项目中的 pod 和日志，则应该可以访问这些索引。您可以使用以下命令检查当前用户是否有适当的权限：

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

### 输出示例

```
yes
```



### 注意

默认情况下，审计日志不会存储在 OpenShift Container Platform 内部 Elasticsearch 实例中。要在 Kibana 中查看审计日志，您必须使用 Log Forward API 配置使用审计日志的 **default** 输出的管道。

### 流程

在 Kibana 中查看日志：

1. 在 OpenShift Container Platform 控制台中点击 Application Launcher  并选择 **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。
3. 在 Kibana 中，点 **Discover**。
4. 从左上角的下拉菜单中选择您创建的索引模式：**app**、**audit** 或 **infra**。日志数据显示为时间戳文档。
5. 展开一个时间戳的文档。

## 6. 点 JSON 选项卡显示该文件的日志条目。

## 例 6.1. Kibana 中的基础架构日志条目示例

```

{
  "_index": "infra-000001",
  "_type": "_doc",
  "_id": "YmJmYTBINDkZTRmLTliMGQtMjE3NmFiOGUyOWM3",
  "_version": 1,
  "_score": null,
  "_source": {
    "docker": {
      "container_id": "f85fa55bbef7bb783f041066be1e7c267a6b88c4603dfce213e32c1"
    },
    "kubernetes": {
      "container_name": "registry-server",
      "namespace_name": "openshift-marketplace",
      "pod_name": "redhat-marketplace-n64gc",
      "container_image": "registry.redhat.io/redhat/redhat-marketplace-index:v4.7",
      "container_image_id": "registry.redhat.io/redhat/redhat-marketplace-
index@sha256:65fc0c45aabb95809e376feb065771ecda9e5e59cc8b3024c4545c168f",
      "pod_id": "8f594ea2-c866-4b5c-a1c8-a50756704b2a",
      "host": "ip-10-0-182-28.us-east-2.compute.internal",
      "master_url": "https://kubernetes.default.svc",
      "namespace_id": "3abab127-7669-4eb3-b9ef-44c04ad68d38",
      "namespace_labels": {
        "openshift_io/cluster-monitoring": "true"
      },
      "flat_labels": [
        "catalogsource_operators_coreos_com/update=redhat-marketplace"
      ]
    },
    "message": "time=\"2020-09-23T20:47:03Z\" level=info msg=\"serving registry\"
database=/database/index.db port=50051",
    "level": "unknown",
    "hostname": "ip-10-0-182-28.internal",
    "pipeline_metadata": {
      "collector": {
        "ipaddr4": "10.0.182.28",
        "inputname": "fluent-plugin-systemd",
        "name": "fluentd",
        "received_at": "2020-09-23T20:47:15.007583+00:00",
        "version": "1.7.4 1.6.0"
      }
    },
    "@timestamp": "2020-09-23T20:47:03.422465+00:00",
    "viaq_msg_id": "YmJmYTBINDkZTRmLTliMGQtMjE3NmFiOGUyOWM3",
    "openshift": {
      "labels": {
        "logging": "infra"
      }
    }
  },
  "fields": {
    "@timestamp": [
      "2020-09-23T20:47:03.422Z"
    ]
  }
}

```

```
"pipeline_metadata.collector.received_at": [  
  "2020-09-23T20:47:15.007Z"  
]  
},  
"sort": [  
  1600894023422  
]  
}
```

## 第 7 章 将日志转发到第三方系统

默认情况下，OpenShift Logging 将容器和基础架构日志发送到 **ClusterLogging** 自定义资源中定义的默认内部 Elasticsearch 日志存储。但是，它不会将审计日志发送到内部存储，因为它不提供安全存储。如果此默认配置满足您的需要，则不需要配置 Cluster Log Forwarder。

要将日志发送到其他日志聚合器，请使用 OpenShift Container Platform Cluster Log Forwarder。通过在这个 API，您可以将容器、基础架构和审计日志发送到集群内部或外部的特定端点。另外，您可以向不同的系统发送不同类型的日志，这样不同个人就可以访问不同系统。您还可以根据机构的要求，启用传输层安全 (TLS) 支持来安全地发送日志。



### 注意

要将审计日志发送到内部日志存储，请使用 Cluster Log Forwarder，如将[审计日志转发到日志存储](#)中所述。

当外部转发日志时，Red Hat OpenShift Logging Operator 会创建或修改 Fluentd 配置映射来使用所需的协议发送日志。您需要在外部日志聚合器上配置协议。

另外，您可以创建一个配置映射来使用 [Fluentd forward](#) 协议 或 [syslog](#) 协议 将日志发送到外部系统。但是，这些转发日志的方法在 OpenShift Container Platform 中已弃用，并将在以后的版本中删除。



### 重要

您不能在同一集群中使用配置映射方法和 Cluster Log Forwarder。

### 7.1. 关于将日志转发到第三方系统

将集群日志转发到外部第三方系统需要结合 **ClusterLogForwarder** 自定义资源 (CR) 中指定的 *输出* 和 *管道*，将日志发送到 OpenShift Container Platform 集群内部和外部的特定端点。您还可以使用 *输入* 将与特定项目关联的应用程序日志转发到端点。

- *输出* 是您定义的日志数据的目的地，或您希望发送日志的位置。输出可以是以下类型之一：
  - **elasticsearch**。一个外部 Elasticsearch 6（所有发行版本）实例。**elasticsearch** 输出可以使用 TLS 连接。
  - **fluentdForward**。一个支持 Fluentd 的外部日志聚合解决方案。这个选项使用 Fluentd 转发协议。**fluentdForward** 输出可以使用 TCP 或 TLS 连接，并通过在 `secret` 中提供一个 `shared_key` 字段来支持共享密钥身份验证。共享密钥身份验证可在使用或不使用 TLS 的情况下使用。
  - **syslog**。支持 syslog [RFC3164](#) 或 [RFC5424](#) 协议的外部日志聚合解决方案。**syslog** 输出可以使用 UDP、TCP 或 TLS 连接。
  - **kafka**。Kafka 代理。**kafka** 输出可以使用 TCP 或 TLS 连接。
  - **default**。内部 OpenShift Container Platform Elasticsearch 实例。您不需要配置默认输出。如果配置 **default** 输出，您会收到出错信息，因为 Red Hat OpenShift Logging Operator 保留了 **default** 输出。

如果输出 URL 方案需要 TLS (HTTPS、TLS 或 UDPS) ,则需要启用 TLS 服务器端的身份验证。为了同时启用客户端身份验证，输出必须在 **openshift-logging** 项目中命名一个 `secret`。secret 必须具有代表指向以下整数的键：`tls.crt`、`tls.key` 和 `ca-bundle.crt`。

- 管道(*pipeline*)定义从一个日志类型到一个或多个输出, 或您要发送的日志的简单路由。日志类型是以下之一:
  - **application**.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)。
  - **infrastructure**.在 **openshift\***、**kube\*** 或 **default** 项目中运行的容器日志, 以及来源于节点文件系统的 **journal** 日志。
  - **audit**.由 **auditd**、节点审计系统以及 Kubernetes API 服务器和 OpenShift API 服务器的审计日志生成的日志。

您可以使用管道中的 **key:value** 对为出站日志消息添加标签。例如,您可以在转发给其他数据中心的消息中添加一个标签,或者根据类型为日志添加标签。添加到对象的标签也会通过日志消息转发。

- *input* 会将与特定项目关联的应用程序日志转发到管道。

在管道中, 您要定义使用 **inputRef** 参数转发哪些日志类型, 以及将日志转发到使用 **outputRef** 参数的位置。

请注意:

- 如果 **ClusterLogForwarder** CR 对象存在, 日志不会转发到默认的 Elasticsearch 实例, 除非有带有 **default** 输出的管道。
- 默认情况下, OpenShift Logging 将容器和基础架构日志发送到 **ClusterLogging** 自定义资源中定义的默认内部 Elasticsearch 日志存储。但是, 它不会将审计日志发送到内部存储, 因为它不提供安全存储。如果此默认配置满足您的需要, 则不需要配置 Log Forwarding API。
- 如果您没有为日志类型定义管道, 则将丢弃未定义类型的日志。例如, 如果您为 **application** 和 **audit** 类型指定管道, 但没有为 **infrastructure** 类型指定管道, 则 **infrastructure** 日志会丢弃。
- 您可以使用 **ClusterLogForwarder** 自定义资源 (CR) 中的多种输出类型将日志发送到支持不同协议的服务器。
- 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 Logging 不遵循这些规范。
- 您需要创建并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。

以下示例将审计日志转发到安全的外部 Elasticsearch 实例, 基础架构日志发送到不安全的外部 Elasticsearch 实例, 应用程序日志发送到 Kafka 代理, 以及 **my-apps-logs** 项目中的应用程序日志发送到内部 Elasticsearch 实例。

## 日志转发输出和管道示例

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance ①
  namespace: openshift-logging ②
spec:
  outputs:
    - name: elasticsearch-secure ③
      type: "elasticsearch"
```

```

url: https://elasticsearch.secure.com:9200
secret:
  name: elasticsearch
- name: elasticsearch-insecure 4
  type: "elasticsearch"
  url: http://elasticsearch.insecure.com:9200
- name: kafka-app 5
  type: "kafka"
  url: tls://kafka.secure.com:9093/app-topic
inputs: 6
- name: my-app-logs
  application:
    namespaces:
      - my-project
pipelines:
- name: audit-logs 7
  inputRefs:
    - audit
  outputRefs:
    - elasticsearch-secure
    - default
  parse: json 8
  labels:
    secure: "true" 9
    datacenter: "east"
- name: infrastructure-logs 10
  inputRefs:
    - infrastructure
  outputRefs:
    - elasticsearch-insecure
  labels:
    datacenter: "west"
- name: my-app 11
  inputRefs:
    - my-app-logs
  outputRefs:
    - default
- inputRefs: 12
  - application
  outputRefs:
    - kafka-app
  labels:
    datacenter: "south"

```

- 1 **ClusterLogForwarder** CR 的名称必须是 **instance**。
- 2 **ClusterLogForwarder** CR 的命名空间必须是 **openshift-logging**。
- 3 使用带有安全 URL 的 **secret** 来配置安全 Elasticsearch 输出。
  - 描述输出的名称。
  - 输出类型：**elasticsearch**。
  - Elasticsearch 实例的安全 URL 和端口作为有效的绝对 URL，包括前缀。

- 用于 TLS 通信的端点所需的 secret。secret 必须存在于 **openshift-logging** 项目中。

4 配置不安全的 Elasticsearch 输出：

- 描述输出的名称。
- 输出类型：**elasticsearch**。
- Elasticsearch 实例的不安全 URL 和端口作为有效的绝对 URL，包括前缀。

5 使用客户端验证的 TLS 通信通过安全 URL 配置 Kafka 输出

- 描述输出的名称。
- 输出的类型：**kafka**。
- 将 Kafka 代理的 URL 和端口指定为一个有效的绝对 URL，包括前缀。

6 用于过滤 **my-project** 命名空间中的应用程序日志的输入配置。

7 用于将审计日志发送到安全的外部 Elasticsearch 实例的管道配置：

- 可选。描述管道的名称。
- **inputRefs** 是日志类型，在这个示例中是 **audit**。
- **outputRefs** 是输出使用的名称，在本例中，**elasticsearch-secure** 可以转发到安全的 Elasticsearch 实例，**default** 转发到内部 Elasticsearch 实例。
- 可选：添加到日志的标签。

8 可选：转发结构化的 JSON 日志条目作为 JSON 对象，在 **structured** 项。日志条目必须包含有效的结构化 JSON；否则，OpenShift Logging 会删除 **structured** 字段，并将日志条目发送到默认索引 **app-00000x**。

9 可选：字符串。要添加到日志中的一个或多个标签。对值加引号（如 "true"），以便它们被识别为字符串值，而不是作为布尔值。

10 管道配置，将基础架构日志发送到不安全的外部 Elasticsearch 实例。

11 管道配置，用于将日志从 **my-project** 项目发送到内部 Elasticsearch 实例。

- 可选。描述管道的名称。
- **inputRefs** 是一个特定的输入：**my-app-logs**。
- **outputRefs** 是 **default**。
- 可选：字符串。要添加到日志中的一个或多个标签。

12 将日志发送到 Kafka 代理的管道配置，不带有管道名称：

- **inputRefs** 是日志类型，在这个示例中是 **application**。
- **outputRefs** 是要使用的输出名称。
- 可选：字符串。要添加到日志中的一个或多个标签。

## 当外部日志聚合器不可用时，Fluentd 日志处理

如果外部日志记录聚合器不可用且无法接收日志，Fluentd 会继续收集日志并将其存储在缓冲中。当日志聚合器可用时，日志转发会恢复，包括缓冲的日志。如果缓冲区已满，Fluentd 会停止收集日志。OpenShift Container Platform 轮转日志并删除日志。您无法调整缓冲区大小，或者将持久性卷声明（PVC）添加到 Fluentd 守护进程集或 Pod 中。

## 7.2. 支持的日志数据输出类型

Red Hat OpenShift Logging 5.0 提供以下输出类型和协议，用于将日志数据发送到目标日志收集器。

红帽会测试下表中显示的每个组合。然而，您应该能够将日志数据发送到最接近这些协议的更大范围的目标日志收集器。

输出类型	协议	测试使用
fluentdForward	fluentd forward v1	fluentd 1.7.4 logstash 7.10.1
elasticsearch	elasticsearch	Elasticsearch 6.8.1 Elasticsearch 7.10.1
syslog	RFC-3164、RFC-5424	rsyslog 8.37.0-9.el7
kafka	kafka 0.11	kafka 2.4.1



### 注意

在以前的版本中，syslog 输出只支持 RFC-3164。当前的 syslog 输出添加了对 RFC-5424 的支持。

## 7.3. 将日志转发到外部 ELASTICSEARCH 实例

除了内部 OpenShift Container Platform Elasticsearch 实例外，您还可以将日志转发到外部 Elasticsearch 实例。您需要配置外部日志聚合器，以接收来自 OpenShift Container Platform 的日志数据。

要配置日志转发到外部 Elasticsearch 实例，请创建一个 **ClusterLogForwarder** 自定义资源（CR），其中包含输出到该实例的输出以及使用输出的管道。外部 Elasticsearch 输出可以使用 HTTP（不安全）或 HTTPS（安全 HTTP）连接。

要将日志转发到外部和内部 Elasticsearch 实例，请将输出和管道创建到外部实例，以及一个使用 **default** 输出将日志转发到内部实例的管道。您不需要创建 **default** 输出。如果配置 **default** 输出，您会收到出错信息，因为 Red Hat OpenShift Logging Operator 保留了 **default** 输出。



### 注意

如果您只想将日志转发到内部 OpenShift Container Platform Elasticsearch 实例，则不需要创建一个 **ClusterLogForwarder** CR。

### 先决条件

- 您必须有配置为使用指定协议或格式接收日志数据的日志服务器。

## 流程

1. 创建一个类似如下的 **ClusterLogForwarder** CR YAML 文件：

```

apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
    - name: elasticsearch-insecure 3
      type: "elasticsearch" 4
      url: http://elasticsearch.insecure.com:9200 5
    - name: elasticsearch-secure
      type: "elasticsearch"
      url: https://elasticsearch.secure.com:9200
      secret:
        name: es-secret 6
  pipelines:
    - name: application-logs 7
      inputRefs: 8
      - application
      - audit
      outputRefs:
      - elasticsearch-secure 9
      - default 10
      parse: json 11
      labels:
        myLabel: "myValue" 12
    - name: infrastructure-audit-logs 13
      inputRefs:
      - infrastructure
      outputRefs:
      - elasticsearch-insecure
      labels:
        logs: "audit-infra"

```

- 1 **ClusterLogForwarder** CR 的名称必须是 **instance**。
- 2 **ClusterLogForwarder** CR 的命名空间必须是 **openshift-logging**。
- 3 指定输出的名称。
- 4 指定 **elasticsearch** 类型。
- 5 指定外部 Elasticsearch 实例的 URL 和端口作为有效的绝对 URL。您可以使用 **http**（不安全）或 **https**（安全 HTTP）协议。如果启用了使用 CIDR 注解的集群范围代理，输出必须是服务器名称或 FQDN，而不是 IP 地址。
- 6

如果使用 **https** 前缀，则必须指定 TLS 通信端点所需的 **secret** 名称。secret 必须存在于 **openshift-logging** 项目中，且必须具有指向它们所代表的相应证书的 **tls.crt**、**tls.key** 和

- 7 可选：指定管道的名称。
- 8 指定使用该管道转发哪些日志类型：**application**、**infrastructure** 或 **audit**。
- 9 指定要与该管道搭配使用的输出来转发日志。
- 10 可选：指定将日志发送到内部 Elasticsearch 实例的 **default** 输出。
- 11 可选：转发结构化的 JSON 日志条目作为 JSON 对象，在 **structured** 项。日志条目必须包含有效的结构化 JSON；否则，OpenShift Logging 会删除 **structured** 字段，并将日志条目发送到默认索引 **app-00000x**。
- 12 可选：字符串。要添加到日志中的一个或多个标签。
- 13 可选：配置多个输出，将日志转发到任何受支持类型的其他外部日志聚合器：
  - 可选。描述管道的名称。
  - **inputRefs** 是使用管道转发的日志类型：**application**、**infrastructure** 或 **audit**。
  - **outputRefs** 是要使用的输出名称。
  - 可选：字符串。要添加到日志中的一个或多个标签。

## 2. 创建 CR 对象。

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

Red Hat OpenShift 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
```

## 7.4. 使用 FLUENTD 转发协议转发日志

您可以使用 Fluentd **forward** 协议将日志副本发送到配置为接受协议的外部日志聚合器，而非默认的 Elasticsearch 日志存储。您需要配置外部日志聚合器以接收来自 OpenShift Container Platform 的日志。

要使用 **forward** 协议配置日志转发，请创建一个 **ClusterLogForwarder** 自定义资源（CR），并将一个或多个输出输出到使用这些输出的 Fluentd 服务器和管道。Fluentd 输出可以使用 TCP（不安全）或 TLS（安全 TCP）连接。



### 注意

另外，您可以通过配置映射来使用 **forward** 协议转发日志。但是，此方法在 OpenShift Container Platform 中已弃用，并将在以后的发行版本中删除。

### 先决条件

- 您必须有配置为使用指定协议或格式接收日志数据的日志服务器。

## 流程

1. 创建一个类似如下的 **ClusterLogForwarder** CR YAML 文件：

```

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

```

- ① **ClusterLogForwarder** CR 的名称必须是 **instance**。
- ② **ClusterLogForwarder** CR 的命名空间必须是 **openshift-logging**。
- ③ 指定输出的名称。
- ④ 指定 **fluentdForward** 类型。
- ⑤ 指定外部 Fluentd 实例的 URL 和端口作为有效的绝对 URL。您可以使用 **tcp**（不安全）或者 **tls**（安全 TCP）协议。如果启用了使用 CIDR 注解的集群范围代理，输出必须是服务器名称或 FQDN，而不是 IP 地址。
- ⑥ 如果使用 **tls** 前缀，您必须为 TLS 通信指定端点所需的 secret 名称。secret 必须存在于 **openshift-logging** 项目中，且必须具有指向它们所代表的相应证书的 **tls.crt**、**tls.key** 和 **ca-bundle.crt** 的键。
- ⑦ 可选。为管道指定一个名称。

- 8 指定使用该管道转发哪些日志类型：**application**、**infrastructure** 或 **audit**。
- 9 指定要与该管道搭配使用的输出来转发日志。
- 10 可选。指定将日志转发到内部 Elasticsearch 实例的默认输出。
- 11 可选：转发结构化的 JSON 日志条目作为 JSON 对象，在 **structured** 项。日志条目必须包含有效的结构化 JSON；否则，OpenShift Logging 会删除 **structured** 字段，并将日志条目发送到默认索引 **app-00000x**。
- 12 可选：字符串。要添加到日志中的一个或多个标签。
- 13 可选：配置多个输出，将日志转发到任何受支持类型的其他外部日志聚合器：
  - 可选。描述管道的名称。
  - **inputRefs** 是使用管道转发的日志类型：**application**、**infrastructure** 或 **audit**。
  - **outputRefs** 是要使用的输出名称。
  - 可选：字符串。要添加到日志中的一个或多个标签。

## 2. 创建 CR 对象。

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

Red Hat OpenShift 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
```

## 7.5. 使用 SYSLOG 协议转发日志

您可以使用 [syslog RFC3164](#) 或 [RFC5424](#) 协议将日志副本发送到配置为接受该协议的外部日志聚合器（替代默认的 Elasticsearch 日志存储或作为它的补充）。您需要配置外部日志聚合器（如 syslog 服务器）来接收来自 OpenShift Container Platform 的日志。

要使用 **syslog** 协议配置日志转，请创建一个 **ClusterLogForwarder** 自定义资源（CR），并将一个或多个输出输出到使用这些输出的 syslog 服务器和管道。syslog 输出可以使用 UDP、TCP 或 TLS 连接。



### 注意

另外，您可以使用配置映射使用 **syslog** RFC3164 协议转发日志。但是，此方法在 OpenShift Container Platform 中已弃用，并将在以后的发行版本中删除。

### 先决条件

- 您必须有配置为使用指定协议或格式接收日志数据的日志服务器。

### 流程

1. 创建一个类似如下的 **ClusterLogForwarder** CR YAML 文件：

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
    - name: rsyslog-east 3
      type: syslog 4
      syslog: 5
        facility: local0
        rfc: RFC3164
        payloadKey: message
        severity: informational
      url: 'tls://rsyslogserver.east.example.com:514' 6
      secret: 7
        name: syslog-secret
    - name: rsyslog-west
      type: syslog
      syslog:
        appName: myapp
        facility: user
        msgID: mymsg
        proclD: myproc
        rfc: RFC5424
        severity: debug
      url: 'udp://rsyslogserver.west.example.com:514'
  pipelines:
    - name: syslog-east 8
      inputRefs: 9
        - audit
        - application
      outputRefs: 10
        - rsyslog-east
        - default 11
      parse: json 12
      labels:
        secure: "true" 13
        syslog: "east"
    - name: syslog-west 14
      inputRefs:
        - infrastructure
      outputRefs:
        - rsyslog-west
        - default
      labels:
        syslog: "west"

```

- 1 **ClusterLogForwarder** CR 的名称必须是 **instance**。
- 2 **ClusterLogForwarder** CR 的命名空间必须是 **openshift-logging**。
- 3 指定输出的名称。

- 4 指定 **syslog** 类型。
- 5 可选。指定以下列出的 **syslog** 参数。
- 6 指定外部 **syslog** 实例的 URL 和端口。您可以使用 **udp**（不安全）、**tcp**（不安全）或者 **tls**（安全 TCP）协议。如果启用了使用 CIDR 注解的集群范围代理，输出必须是服务器名称或 FQDN，而不是 IP 地址。
- 7 如果使用 **tls** 前缀，您必须为 TLS 通信指定端点所需的 **secret** 名称。**secret** 必须存在于 **openshift-logging** 项目中，且必须具有指向它们所代表的相应证书的 **tls.crt**、**tls.key** 和 **ca-bundle.crt** 的键。
- 8 可选：指定管道的名称。
- 9 指定使用该管道转发哪些日志类型：**application**、**infrastructure** 或 **audit**。
- 10 指定要与该管道搭配使用的输出来转发日志。
- 11 可选：指定将日志转发到内部 Elasticsearch 实例的 **default** 输出。
- 12 可选：转发结构化的 JSON 日志条目作为 JSON 对象，在 **structured** 项。日志条目必须包含有效的结构化 JSON；否则，OpenShift Logging 会删除 **structured** 字段，并将日志条目发送到默认索引 **app-00000x**。
- 13 可选：字符串。要添加到日志中的一个或多个标签。对值加引号（如 "true"），以便它们被识别为字符串值，而不是作为布尔值。
- 14 可选：配置多个输出，将日志转发到任何受支持类型的其他外部日志聚合器：
  - 可选。描述管道的名称。
  - **inputRefs** 是使用管道转发的日志类型：**application**、**infrastructure** 或 **audit**。
  - **outputRefs** 是要使用的输出名称。
  - 可选：字符串。要添加到日志中的一个或多个标签。

## 2. 创建 CR 对象。

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

Red Hat OpenShift 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
```

### 7.5.1. syslog 参数

您可以为 **syslog** 输出配置以下内容。如需更多信息,请参阅 [syslog RFC3164](#) 或 [RFC5424 RFC](#)。

- **facility**: **syslog facility**. 该值可以是十进制整数，也可以是区分大小写的关键字：
  - **0** 或 **kern** 用于内核信息
  - **1** 或 **user** 代表用户级信息（默认）。

- **2** 或 **mail** 用于邮件系统。
  - **3** 或 **daemon** 用于系统守护进程
  - **4** 或 **auth** 用于安全/身份验证信息
  - **5** 或 **syslog** 用于 syslogd 内部生成的信息
  - **6** 或 **lpr** 用于行打印机子系统
  - **7** 或 **news** 用于网络新闻子系统
  - **8** 或 **uucp** 用于 UUCP 子系统
  - **9** 或 **cron** 用于 clock 守护进程
  - **10** 或 **authpriv** 用于安全身份验证信息
  - **11** 或 **ftp** 用于 FTP 守护进程
  - **12** 或 **ntp** 用于 NTP 子系统
  - **13** 或 **security** 用于 syslog audit 日志
  - **14** 或 **console** 用于 syslog alert 日志
  - **15** 或 **solaris-cron** 用于 scheduling 守护进程
  - **16-23** 或 **local0 - local7** 用于本地使用的工具
- 可选。 **payloadKey** : 用作 syslog 消息有效负载的记录字段。



### 注意

配置 **payloadKey** 参数可防止将其他参数转发到 syslog。

- RFC : 用于使用 syslog 发送日志的 RFC。默认为 RFC5424。
- severity : 设置传出的 syslog 记录的 **syslog** 的 **严重性**。该值可以是十进制整数，也可以是区分大小写的关键字：
  - **0** 或 **Emergency** 用于代表系统不可用的信息
  - **1** 或 **Alert** 用于代表立即执行操作的信息
  - **2** 或 **Critical** 用于代表关键状况的信息
  - **3** 或 **Error** 用于代表错误状况的信息
  - **4** 或 **Warning** 用于代表警告条件的信息
  - **5** 或 **Notice** 用于代表正常但存在重要条件的信息
  - **6** 或 **Informational** 用于代表提示信息的信息
  - **7** 或 **Debug** 用于代表调试级别的信息（默认）

- tag : Tag 指定记录字段，用作 syslog 消息上的标签。
- trimPrefix : 从标签中删除指定的前缀。

## 7.5.2. 其他 RFC5424 syslog 参数

以下参数适用于 RFC5424:

- appName: APP-NAME 是一个自由文本字符串，用于标识发送日志的应用程序。必须为 **RFC5424** 指定。
- msgID: MSGID 是一个用于标识消息类型的自由文本字符串。必须为 **RFC5424** 指定。
- PROCID: PROCID 是一个自由文本字符串。该值的变化表示 syslog 报告不连续。必须为 **RFC5424** 指定。

## 7.6. 将日志转发到 KAFKA 代理

除了默认的 Elasticsearch 存储外，您还可以将日志转发到外部 Kafka 代理。

要配置日志转发到外部 Kafka 实例，请创建一个 **ClusterLogForwarder** 自定义资源 (CR)，包括输出到该实例的输出以及使用输出的管道。您可以在输出中包括特定的 Kafka 主题，也可以使用默认值。Kafka 输出可以使用 TCP (不安全) 或者 TLS (安全 TCP) 连接。

### 流程

1. 创建一个类似如下的 **ClusterLogForwarder** CR YAML 文件：

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance ①
  namespace: openshift-logging ②
spec:
  outputs:
    - name: app-logs ③
      type: kafka ④
      url: tls://kafka.example.devlab.com:9093/app-topic ⑤
      secret:
        name: kafka-secret ⑥
    - name: infra-logs
      type: kafka
      url: tcp://kafka.devlab2.example.com:9093/infra-topic ⑦
    - name: audit-logs
      type: kafka
      url: tls://kafka.qelab.example.com:9093/audit-topic
      secret:
        name: kafka-secret-qe
  pipelines:
    - name: app-topic ⑧
      inputRefs: ⑨
      - application
      outputRefs: ⑩
      - app-logs

```

```

parse: json 11
labels:
  logType: "application" 12
- name: infra-topic 13
  inputRefs:
  - infrastructure
  outputRefs:
  - infra-logs
  labels:
    logType: "infra"
- name: audit-topic
  inputRefs:
  - audit
  outputRefs:
  - audit-logs
  - default 14
  labels:
    logType: "audit"

```

- 1** **ClusterLogForwarder** CR 的名称必须是 **instance**。
- 2** **ClusterLogForwarder** CR 的命名空间必须是 **openshift-logging**。
- 3** 指定输出的名称。
- 4** 指定 **kafka** 类型。
- 5** 将 Kafka 代理的 URL 和端口指定为一个有效的绝对 URL，也可以同时指定特定标题。您可以使用 **tcp**（不安全）或者 **tls**（安全 TCP）协议。如果启用了使用 CIDR 注解的集群范围代理，输出必须是服务器名称或 FQDN，而不是 IP 地址。
- 6** 如果使用 **tls** 前缀，您必须为 TLS 通信指定端点所需的 secret 名称。secret 必须存在于 **openshift-logging** 项目中，且必须具有指向它们所代表的相应证书的 **tls.crt**、**tls.key** 和 **ca-bundle.crt** 的键。
- 7** 可选：要发送不安全的输出，在 URL 前面使用 **tcp** 前缀。另外，省略此输出中的 **secret** 键及其 **name**。
- 8** 可选：指定管道的名称。
- 9** 指定使用该管道转发哪些日志类型：**application**、**infrastructure** 或 **audit**。
- 10** 指定要与该管道搭配使用的输出来转发日志。
- 11** 可选：转发结构化的 JSON 日志条目作为 JSON 对象，在 **structured** 项。日志条目必须包含有效的结构化 JSON；否则，OpenShift Logging 会删除 **structured** 字段，并将日志条目发送到默认索引 **app-00000x**。
- 12** 可选：字符串。要添加到日志中的一个或多个标签。
- 13** 可选：配置多个输出，将日志转发到任何受支持类型的其他外部日志聚合器：
  - 可选。描述管道的名称。
  - **inputRefs** 是使用管道转发的日志类型：**application**、**infrastructure** 或 **audit**。
  - **outputRefs** 是要使用的输出名称。

- 可选：字符串。要添加到日志中的一个或多个标签。

14 可选：指定 **default** 将日志转发到内部 Elasticsearch 实例。

2. 可选：要将单个输出转发到多个 Kafka 代理，请指定 Kafka 代理数组，如下例所示：

```
...
spec:
  outputs:
  - name: app-logs
    type: kafka
    secret:
      name: kafka-secret-dev
    kafka: ①
      brokers: ②
        - tls://kafka-broker1.example.com:9093/
        - tls://kafka-broker2.example.com:9093/
      topic: app-topic ③
...

```

① 指定一个带有 **brokers** 和 **topic** 键的 **kafka** 键。

② 使用 **brokers** 键指定一个或多个代理的数组。

③ 使用 **topic** 键指定要接收日志的目标主题。

3. 创建 CR 对象。

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

Red Hat OpenShift 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
```

## 7.7. 从特定项目转发应用程序日志

您可以使用 Cluster Log Forwarder 将应用日志的副本从特定项目发送到外部日志聚合器。除了使用默认的 Elasticsearch 日志存储外，您还可以进行此操作。您还必须配置外部日志聚合器，以接收来自 OpenShift Container Platform 的日志数据。

要从项目中配置转发应用程序日志，创建一个 **ClusterLogForwarder** 自定义资源（CR），其中至少从一个项目中输入，为其他日志聚合器提供可选输出，以及使用这些输入和输出的管道。

### 先决条件

- 您必须有配置为使用指定协议或格式接收日志数据的日志服务器。

### 流程

1. 创建一个类似如下的 **ClusterLogForwarder** CR YAML 文件：

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance ❶
  namespace: openshift-logging ❷
spec:
  outputs:
    - name: fluentd-server-secure ❸
      type: fluentdForward ❹
      url: 'tls://fluentdserver.security.example.com:24224' ❺
      secret: ❻
        name: fluentd-secret
    - name: fluentd-server-insecure
      type: fluentdForward
      url: 'tcp://fluentdserver.home.example.com:24224'
  inputs: ❼
    - name: my-app-logs
      application:
        namespaces:
          - my-project
  pipelines:
    - name: forward-to-fluentd-insecure ❽
      inputRefs: ❾
        - my-app-logs
      outputRefs: ❿
        - fluentd-server-insecure
      parse: json ⓫
      labels:
        project: "my-project" ⓬
    - name: forward-to-fluentd-secure ⓭
      inputRefs:
        - application
        - audit
        - infrastructure
      outputRefs:
        - fluentd-server-secure
        - default
      labels:
        clusterId: "C1234"

```

- ❶ **ClusterLogForwarder** CR 的名称必须是 **instance**。
- ❷ **ClusterLogForwarder** CR 的命名空间必须是 **openshift-logging**。
- ❸ 指定输出的名称。
- ❹ 指定输出类型：**elasticsearch**、**fluentdForward**、**syslog** 或 **kafka**。
- ❺ 将外部日志聚合器的 URL 和端口指定为有效的绝对 URL。如果启用了使用 CIDR 注解的集群范围代理，输出必须是服务器名称或 FQDN，而不是 IP 地址。
- ❻ 如果使用 **tls** 前缀，您必须为 TLS 通信指定端点所需的 secret 名称。secret 必须存在于 **openshift-logging** 项目中，并具有每个指向它们所代表证书的 **tls.crt**、**tls.key** 和 **ca-bundle.crt** 密钥。

- 7 用于过滤指定项目的应用程序日志的输入配置。
- 8 管道配置，使用输入将项目应用程序日志发送到外部 Fluentd 实例。
- 9 **my-app-logs** 输入。
- 10 要使用的输出名称。
- 11 可选：转发结构化的 JSON 日志条目作为 JSON 对象，在 **structured** 项。日志条目必须包含有效的结构化 JSON；否则，OpenShift Logging 会删除 **structured** 字段，并将日志条目发送到默认索引 **app-00000x**。
- 12 可选：字符串。要添加到日志中的一个或多个标签。
- 13 管道配置，将日志发送到其他日志聚合器。
  - 可选：指定管道的名称。
  - 指定使用该管道转发哪些日志类型：**application**、**infrastructure** 或 **audit**。
  - 指定要与该管道搭配使用的输出来转发日志。
  - 可选：指定将日志转发到内部 Elasticsearch 实例的 **default** 输出。
  - 可选：字符串。要添加到日志中的一个或多个标签。

## 2. 创建 CR 对象。

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

## 7.8. 从特定 POD 转发应用程序日志

作为集群管理员，您可以使用 Kubernetes pod 标签从特定 pod 收集日志数据并将其转发到日志收集器。

假设您的应用由容器集组成，并与不同命名空间中的其他容器集一起运行。如果这些 pod 具有标识应用程序标签，您可以收集和输出其日志数据到特定的日志收集器。

要指定 pod 标签，请使用一个或多个 **matchLabels** 键值对。如果指定了多个键值对，pod 必须与要选择的所有值匹配。

### 流程

1. 创建 **ClusterLogForwarder** 自定义资源 (CR) YAML 文件。
2. 在 YAML 文件中，使用 **inputs[].name.application.selector.matchLabels** 下的简单基于平等的选择器来指定 pod 标签，如下例所示。

### ClusterLogForwarder CR YAML 文件示例

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

```

spec:
  pipelines:
    - inputRefs: [ myAppLogData ] ❸
      outputRefs: [ default ] ❹
      parse: json ❺
  inputs: ❻
    - name: myAppLogData
      application:
        selector:
          matchLabels: ❷
            environment: production
            app: nginx
          namespaces: ❸
            - app1
            - app2
  outputs: ❾
    - default
    ...

```

- ❶ **ClusterLogForwarder** CR 的名称必须是 **instance**。
- ❷ **ClusterLogForwarder** CR 的命名空间必须是 **openshift-logging**。
- ❸ 指定来自 **inputs[].name** 的一个或多个以逗号分隔的值。
- ❹ 指定来自 **outputs[]** 的一个或多个以逗号分隔的值。
- ❺ 可选：转发结构化的 JSON 日志条目作为 JSON 对象，在 **structured** 项。日志条目必须包含有效的结构化 JSON；否则，OpenShift Logging 会删除 **structured** 字段，并将日志条目发送到默认索引 **app-00000x**。
- ❻ 为具有一组唯一 pod 标签的每个应用程序定义唯一的 **inputs[].name**。
- ❼ 指定您要收集的日志数据的 pod 标签的键值对。您必须指定一个键和值，而不仅仅是一个键。要被选择，pod 必须与所有键值对匹配。
- ❽ 可选：指定一个或多个命名空间。
- ❾ 指定要将日志数据转发到的一个或多个输出。此处显示的可选默认输出将日志数据发送到内部 Elasticsearch 实例。

3. 可选：要将日志数据收集限制为特定的命名空间，请使用 **inputs[].name.application.namespaces**，如上例中所示。
4. 可选：您可以从具有不同 pod 标签的额外应用程序向同一管道发送日志数据。
  - a. 对于 pod 标签的每个唯一组合，创建一个类似于显示的 **inputs[].name** 部分。
  - b. 更新选择器 (**selectors**) 以匹配此应用的容器集标签。
  - c. 将新的 **inputs[].name** 值添加到 **inputRefs**。例如：

```

- inputRefs: [ myAppLogData, myOtherAppLogData ]

```

5. 创建 CR 对象。

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

## 其他资源

- 如需有关 Kubernetes 中 **matchLabels** 的更多信息，请参阅[支持基于集合的要求的资源](#)。

## 7.9. 使用旧的 FLUENTD 方法转发日志

您可以通过创建配置文件和配置映射，使用 Fluentd **forward** 协议将日志发送到 OpenShift Container Platform 集群外的目的地。您需要配置外部日志聚合器，以接收来自 OpenShift Container Platform 的日志数据。



### 重要

此转发日志的方法在 OpenShift Container Platform 中已弃用，并将在以后的发行版本中删除。

要使用 Fluentd **forward** 协议发送日志，请创建一个名为 **secure-forward.conf** 的配置文件，指向外部日志聚合器。然后，使用该文件在 **openshift-logging** 项目中创建一个名为 **secure-forward** 的配置映射，OpenShift Container Platform 在转发日志时使用它。

## 先决条件

- 您必须有配置为使用指定协议或格式接收日志数据的日志服务器。

## Fluentd 配置文件示例

```
<store>
  @type forward
  <security>
    self_hostname ${hostname}
    shared_key "fluent-receiver"
  </security>
  transport tls
  tls_verify_hostname false
  tls_cert_path '/etc/ocp-forward/ca-bundle.crt'
  <buffer>
    @type file
    path '/var/lib/fluentd/secureforwardlegacy'
    queued_chunks_limit_size "1024"
    chunk_limit_size "1m"
    flush_interval "5s"
    flush_at_shutdown "false"
    flush_thread_count "2"
    retry_max_interval "300"
    retry_forever true
    overflow_action "#{ENV['BUFFER_QUEUE_FULL_ACTION']} || 'throw_exception'}"
  </buffer>
  <server>
    host fluent-receiver.example.com
    port 24224
  </server>
</store>
```

## 流程

配置 OpenShift Container Platform 使用旧的 Fluentd 方法转发日志：

1. 创建名为 **secure-forward** 的配置文件，并在 **<store>** 部分中指定类似如下的参数：

```
<store>
  @type forward
  <security>
    self_hostname ${hostname}
    shared_key <key> 1
  </security>
  transport tls 2
  tls_verify_hostname <value> 3
  tls_cert_path <path_to_file> 4
  <buffer> 5
    @type file
    path '/var/lib/fluentd/secureforwardlegacy'
    queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '1024'}"
    chunk_limit_size "#{ENV['BUFFER_SIZE_LIMIT'] || '1m'}"
    flush_interval "#{ENV['FORWARD_FLUSH_INTERVAL'] || '5s'}"
    flush_at_shutdown "#{ENV['FLUSH_AT_SHUTDOWN'] || 'false'}"
    flush_thread_count "#{ENV['FLUSH_THREAD_COUNT'] || '2'}"
    retry_max_interval "#{ENV['FORWARD_RETRY_WAIT'] || '300'}"
    retry_forever true
  </buffer>
  <server>
    name 6
    host 7
    hostlabel 8
    port 9
  </server>
  <server> 10
    name
    host
  </server>
```

- 1 输入节点间共享的密钥。
- 2 指定 **tls** 启用 TLS 验证。
- 3 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。
- 4 将私有 CA 证书文件的路径指定为 **/etc/ocp-forward/ca\_cert.pem**。
- 5 根据需要指定 [Fluentd 缓冲参数](#)。
- 6 可选：为这个服务器输入一个名称。
- 7 指定服务器的主机名或 IP。
- 8 指定服务器的主机标识。
- 9 指定服务器的端口。

- 10 (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。

要使用 Mutual TLS (mTLS) 身份验证，请参阅 [Fluentd 文档](#) 来获取有关客户端证书、密钥参数和其他设置的信息。

2. 在 **openshift-logging** 项目中创建名为 **secure-forward** 的配置映射：

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

Red Hat OpenShift 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. 使用旧的 SYSLOG 方法转发日志

您可以通过创建配置文件和配置映射，使用 **syslog** RFC3164 协议将日志发送到 OpenShift Container Platform 集群外的目的地。您需要配置外部日志聚合器（如 syslog 服务器）来接收来自 OpenShift Container Platform 的日志。



### 重要

此转发日志的方法在 OpenShift Container Platform 中已弃用，并将在以后的发行版本中删除。

**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\_uffered**：缓冲的实现，它通过 TCP 进行通讯并将数据缓冲到块中。

要使用 **syslog** 协议发送日志，请创建一个名为 **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** 项目中创建一个名为 **syslog** 的配置映射，OpenShift Container Platform 在转发日志时使用该文件。

### 先决条件

- 您必须有配置为使用指定协议或格式接收日志数据的日志服务器。

### syslog 配置文件示例

```
<store>
@type syslog_buffered
remote_syslog rsyslogserver.example.com
port 514
hostname ${hostname}
remove_tag_prefix tag
facility local0
severity info
use_record true
```

```
payload_key message
rfc 3164
</store>
```

您可以配置以下 **syslog** 参数。如需更多信息，请参阅 [syslog RFC3164](#)。

- facility: [syslog facility](#)。该值可以是十进制整数，也可以是区分大小写的关键字：
  - **0** 或 **kern** 用于内核信息
  - **1** 或 **user** 代表用户级信息（默认）。
  - **2** 或 **mail** 用于邮件系统。
  - **3** 或 **daemon** 用于系统守护进程
  - **4** 或 **auth** 用于安全/身份验证相关的信息
  - **5** 或 **syslog** 用于 syslogd 内部生成的信息
  - **6** 或 **lpr** 用于行打印机子系统
  - **7** 或 **news** 用于网络新闻子系统
  - **8** 或 **uucp** 用于 UUCP 子系统
  - **9** 或 **cron** 用于 clock 守护进程
  - **10** 或 **authpriv** 用于安全身份验证信息
  - **11** 或 **ftp** 用于 FTP 守护进程
  - **12** 或 **ntp** 用于 NTP 子系统
  - **13** 或 **security** 用于 syslog audit 日志
  - **14** 或 **console** 用于 syslog alert 日志
  - **15** 或 **solaris-cron** 用于 scheduling 守护进程
  - **16-23** 或 **local0 - local7** 用于本地使用的工具
- payloadKey：用作 syslog 消息有效负载的记录字段。
- RFC：用于使用 syslog 发送日志的 RFC。
- severity：设置传出的 syslog 记录的[syslog 的严重性](#)。该值可以是十进制整数，也可以是区分大小写的关键字：
  - **0** 或 **Emergency** 用于代表系统不可用的信息
  - **1** 或 **Alert** 用于代表立即执行操作的信息
  - **2** 或 **Critical** 用于代表关键状况的信息
  - **3** 或 **Error** 用于代表错误状况的信息
  - **4** 或 **Warning** 用于代表警告条件的信息

- **5** 或 **Notice** 用于代表正常但存在重要条件的信息
- **6** 或 **Informational** 用于代表提示信息的信息
- **7** 或 **Debug** 用于代表调试级别的信息（默认）
- tag : 用作 syslog 消息标签的记录字段。
- trimPrefix : 从标签中删除的前缀。

## 流程

配置 OpenShift Container Platform 使用旧配置方法转发日志：

1. 创建名为 **syslog.conf** 的配置文件，并在 **<store>** 部分中指定类似如下的参数：

```
<store>
@type <type> 1
remote_syslog <syslog-server> 2
port 514 3
hostname ${hostname}
remove_tag_prefix <prefix> 4
facility <value>
severity <value>
use_record <value>
payload_key message
rfc 3164 5
</store>
```

- 1 指定要使用的协议，可以是: **syslog** 或 **syslog\_buffered**。
- 2 指定 syslog 服务器的 FQDN 或 IP 地址。
- 3 指定接收方的端口。
- 4 可选：指定适当的 syslog 参数，例如：
  - 从 syslog 前缀中删除指定的 **tag** 字段的参数。
  - 参数将指定的字段设置为 syslog 键。
  - 指定 syslog 日志工具或源的参数。
  - 指定 syslog 日志严重性参数。
  - 参数来使用记录中的严重性和工具（如果可用）。如果为 **true** 则输出内容中包含 **container\_name**、**namespace\_name** 和 **pod\_name**。
  - 参数来指定设置 syslog 消息有效负载的键。默认为 **message**。
- 5 如果使用旧的 syslog 方法，必须把 **rfc** 值指定为 **3164**。

2. 在 **openshift-logging** 项目中创建名为 **syslog** 的配置映射：

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

Red Hat OpenShift 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
```

## 第 8 章 启用 JSON 日志记录

您可以配置 Log Forwarding API，将 JSON 字符串解析为结构化对象。

### 8.1. 解析 JSON 日志

包含 JSON 日志的日志通常以 **message** 字段中的字符串表示。这使得用户难以查询 JSON 文档中的特定字段。OpenShift Logging 的 Log Forwarding API 可让您将 JSON 日志解析到结构化对象，并将其转发到 OpenShift Logging 管理的 Elasticsearch 或 Log Forwarding API 支持的任何其他第三方系统。

为了说明其工作原理，假定您有以下结构化 JSON 日志条目：

#### 结构化 JSON 日志条目示例

```
{"level":"info","name":"fred","home":"bedrock"}
```

通常，**ClusterLogForwarder** 自定义资源 (CR) 会在 **message** 字段中转发该日志条目。**message** 字段包含与 JSON 日志条目等效的 JSON-quoted 字符串，如下例中所示：

#### message 字段示例

```
{"message": "{\"level\":\"info\",\"name\":\"fred\",\"home\":\"bedrock\"\",  
  \"more fields...\"}
```

要启用解析 JSON 日志，您需要将 **parse: json** 添加到 **ClusterLogForwarder** CR 的管道中，如下例所示。

#### 显示 parse: json 的片段示例

```
pipelines:  
- inputRefs: [ application ]  
  outputRefs: myFluentd  
  parse: json
```

当使用 **parse: json** 来启用 JSON 日志解析时，CR 会复制 **structured** 项中的 JSON 结构化日志条目，如下例所示。这不会修改原始的 **message** 字段。

#### 包含结构化 JSON 日志条目的 structured 输出示例

```
{"structured": { "level": "info", "name": "fred", "home": "bedrock" },  
  "more fields..."}
```



#### 重要

如果日志条目不包含有效的结构化 JSON，则将缺少 **structured** 字段。

要启用为特定日志记录平台解析 JSON 日志，请参阅[将日志转发到第三方系统](#)。

### 8.2. 为 ELASTICSEARCH 配置 JSON 日志数据

如果您的 JSON 日志遵循多个模式，在单个索引中存储它们可能会导致类型冲突和卡性问题。要避免这种情况，您必须配置 **ClusterLogForwarder** 自定义资源 (CR)，将每个 schema 分组到单个输出定义中。这样，每个架构被转发到单独的索引。



### 重要

如果您将 JSON 日志转发到 OpenShift Logging 管理的默认 Elasticsearch 实例，它会根据您的配置生成新的索引。为避免与索引数量过多相关的性能问题，请考虑通过标准化到常见模式来保持可能的模式数量较低。

## 结构类型

您可以使用 **ClusterLogForwarder** CR 中的以下结构类型来为 Elasticsearch 日志存储构建索引名称：

- **structTypeKey**（字符串，可选）是消息字段的名称。如果存在该字段的值，则用于构造索引名称。
  - **kubernetes.labels.<key>** 是 Kubernetes pod 标签，其值用于构造索引名称。
  - **openshift.labels.<key>** 是 **ClusterLogForwarder** CR 中的 **pipeline.label.<key>** 元素，其值用于构造索引名称。
  - **kubernetes.container\_name** 使用容器名称来构造索引名称。
- **structTypeName**：（字符串，可选）如果未设置 **structTypeKey**，或者其键不存在，OpenShift Logging 将 **structTypeName** 的值用作结构化类型。当同时使用 **structuredTypeKey** 和 **structuredTypeName** 时，如果 JSON 日志数据中缺少 **structuredTypeKey** 中的键，则 **structTypeName** 提供一个回退索引名称。



### 注意

虽然您可以将 **structuredTypeKey** 的值设置为 "Log Record Fields" 主题中显示的任何字段，但最有用的字段将显示在前面的结构类型列表中。

## structuredTypeKey: kubernetes.labels.<key> 示例

假设如下：

- 集群正在运行以两种不同格式生成 JSON 日志的应用 pod，即 "apache" 和 "google"。
- 用户使用 **logFormat=apache** 和 **logFormat=google** 标记这些应用 pod。
- 您可以在 **ClusterLogForwarder** CR YAML 文件中使用以下代码片段。

```
outputDefaults:
- elasticsearch:
  structuredTypeKey: kubernetes.labels.logFormat 1
  structuredTypeName: nologformat
pipelines:
- inputRefs: <application>
  outputRefs: default
  parse: json 2
```

- 1** 使用 Kubernetes **logFormat** 标签形成的键值对值。

## 2 启用解析 JSON 日志。

在这种情况下，以下结构化日志记录进入 **app-apache-write** 索引：

```
{
  "structured":{"name":"fred","home":"bedrock"},
  "kubernetes":{"labels":{"logFormat": "apache", ...}}
}
```

以下结构化日志记录进入 **app-google-write** 索引中：

```
{
  "structured":{"name":"wilma","home":"bedrock"},
  "kubernetes":{"labels":{"logFormat": "google", ...}}
}
```

### structuredTypeKey: openshift.labels.<key> 示例

假设您在 **ClusterLogForwarder** CR YAML 文件中使用了以下代码片段：

```
outputDefaults:
- elasticsearch:
  structuredTypeKey: openshift.labels.myLabel 1
  structuredTypeName: nologformat
pipelines:
- name: application-logs
  inputRefs:
  - application
  - audit
  outputRefs:
  - elasticsearch-secure
  - default
  parse: json
  labels:
    myLabel: myValue 2
```

1 使用由 OpenShift **myLabel** 标签组成的键值对的值。

2 **myLabel** 元素将字符串值 **myValue** 提供给结构化日志消息。

在这种情况下，以下结构化日志记录进入 **app-myValue-write** 索引中：

```
{
  "structured":{"name":"fred","home":"bedrock"},
  "openshift":{"labels":{"myLabel": "myValue", ...}}
}
```

### 其他注意事项

- 结构化记录的 Elasticsearch 索引通过将 "app-" 添加到结构化类型并附加 "-write" 来形成。
- 非结构化记录不会发送到结构化索引。在应用、基础架构或审计索引中，它们按照常态进行索引。

- 如果没有非空的结构化类型，则转发一个没有 **structured** 项的 *unstructured* 记录。

不要过载有太多索引的 Elasticsearch。仅对不同的日志格式使用不同的结构化类型，而不用为每个应用程序或命名空间都使用不同的结构化类型。例如，大多数 Apache 应用使用相同的 JSON 日志格式和结构化类型，如 **LogApache**。

### 8.3. 将 JSON 日志转发到 ELASTICSEARCH 日志存储

对于 Elasticsearch 日志存储，如果您的 JSON 日志条目遵循不同的模式，请将 **ClusterLogForwarder** 自定义资源 (CR) 配置为将每个 JSON 模式分组到单个输出定义中。这样，Elasticsearch 会为每个 schema 使用一个单独的索引。



#### 重要

因为将不同的模式转发到同一索引可能会导致类型冲突和卡化问题，所以您必须在将数据转发到 Elasticsearch 存储前执行此配置。

为避免与索引数量过多相关的性能问题，请考虑通过标准化到常见模式来保持可能的模式数量较低。

#### 流程

1. 将以下代码片段添加到 **ClusterLogForwarder** CR YAML 文件中。

```
outputDefaults:
- elasticsearch:
  structuredTypeKey: <log record field>
  structuredTypeName: <name>
pipelines:
- inputRefs:
  - application
  outputRefs: default
  parse: json
```

2. 可选：使用 **structTypeKey** 指定其中一个日志记录字段，如前面的为 [Elasticsearch 配置 JSON 日志数据](#) 所述。否则，删除此行。
3. 可选：使用 **structTypeName** 指定 **<name>**，如前面的为 [Elasticsearch 配置 JSON 日志数据](#) 所述。否则，删除此行。



#### 重要

要解析 JSON 日志，您必须设置 **structuredTypeKey** 或 **structuredTypeName**，或者同时设置 **structuredTypeKey** 和 **structuredTypeName**。

4. 对于 **inputRefs**，指定应使用该管道转发哪些日志类型，如 **application**、**infrastructure** 或 **audit**。
5. 将 **parse: json** 元素添加到管道。
6. 创建 CR 对象。

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

Red Hat OpenShift Logging Operator 会重新部署 Fluentd Pod。但是，如果没有重新部署，请删除 Fluentd Pod 以强制重新部署。

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

### 其他资源

- [将日志转发到第三方系统](#)

## 第 9 章 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，并将 OpenShift Logging 收集的数据记录到日志中。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。

### 9.1. 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，或更改部署对象 CPU 和内存请求。

#### 先决条件

- 需要适当的权限，以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 OpenShift Logging。

#### 流程

1. 为事件路由器创建模板：

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to OpenShift Logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
- kind: ServiceAccount 1
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
- kind: ClusterRole 2
  apiVersion: rbac.authorization.k8s.io/v1
  metadata:
    name: event-reader
  rules:
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "watch", "list"]
- kind: ClusterRoleBinding 3
  apiVersion: rbac.authorization.k8s.io/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 6
  displayName: Image
  value: "registry.redhat.io/openshift-logging/eventrouter-rhel8:v0.3"
- name: CPU 7
  displayName: CPU
  value: "100m"
- name: MEMORY 8
  displayName: Memory
  value: "128Mi"
- name: NAMESPACE
  displayName: Namespace
  value: "openshift-logging" 9

```

- 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 绑定到服务帐户。
- 4 在 **openshift-logging** 项目中创建一个配置映射来生成所需的 **config.json** 文件。
- 5 在 **openshift-logging** 项目中创建一个部署，以生成并配置 Event Router pod。
- 6 指定镜像，由标签标识，如 **v0.3**。
- 7 指定分配给事件路由器 pod 的最小内存量。默认值为**128Mi**。
- 8 指定分配给事件路由器 pod 的最小 CPU 量。默认值为**100m**。
- 9 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程序模板：

```
$ oc process -f <templatefile> | oc apply -n openshift-logging -f -
```

例如：

```
$ oc process -f eventrouter.yaml | oc apply -n openshift-logging -f -
```

### 输出示例

```

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

## 第 10 章 更新 OPENSIFT LOGGING

表 10.1. OpenShift Container Platform 版本支持 Red Hat OpenShift Logging (RHOL)

	4.7	4.8	4.9
RHOL 5.1	X	X	
RHOL 5.2	X	X	X
RHOL 5.3		X	X

要从 OpenShift Container Platform 4.6 及更早版本的集群日志记录升级到 OpenShift Logging 5.x，您需要将 OpenShift Container Platform 集群更新至 4.7 或 4.8。然后，您更新以下 operator：

- 从 Elasticsearch Operator 4.x 到 OpenShift Elasticsearch Operator 5.x
- 从 Cluster Logging Operator 4.x 到 Red Hat OpenShift Logging Operator 5.x

要从 OpenShift Logging 的旧版本升级到当前版本，您需要将 OpenShift Elasticsearch Operator 和 Red Hat OpenShift Logging Operator 更新至当前版本。

### 10.1. 从 OPENSIFT CONTAINER PLATFORM 4.6 或更早版本的集群日志记录升级到 OPENSIFT LOGGING 5.X

OpenShift Container Platform 4.7 进行了以下名称更改：

- *集群日志记录功能成为 Red Hat OpenShift Logging 5.x 产品。*
- *Cluster Logging Operator 成为 Red Hat OpenShift Logging Operator。*
- *Elasticsearch Operator 成为 OpenShift Elasticsearch Operator。*

要从 OpenShift Container Platform 4.6 及更早版本的集群日志记录升级到 OpenShift Logging 5.x，您需要将 OpenShift Container Platform 集群更新至 4.7 或 4.8。然后，您更新以下 operator：

- 从 Elasticsearch Operator 4.x 到 OpenShift Elasticsearch Operator 5.x
- 从 Cluster Logging Operator 4.x 到 Red Hat OpenShift Logging Operator 5.x



#### 重要

在更新 Red Hat OpenShift Logging Operator 前，您必须更新 OpenShift Elasticsearch Operator。您还必须将两个 Operator 更新至同一版本。

如果您以错误的顺序更新 Operator，则 Kibana 不会更新，并且不会创建 Kibana 自定义资源 (CR)。要临时解决这个问题，请删除 Red Hat OpenShift Logging Operator pod。当 Red Hat OpenShift Logging Operator pod 重新部署时，它会创建 Kibana CR 和 Kibana 再次可用。

#### 先决条件

- OpenShift Container Platform 版本为 4.7 或更高版本。

- OpenShift Logging 处于健康状态 :
  - 所有 pod 都为 **Ready** 状态。
  - Elasticsearch 集群处于健康状态。
- 您的 Elasticsearch 和 Kibana 数据已被备份。

## 流程

1. 更新 OpenShift Elasticsearch Operator:
  - a. 在 Web 控制台中，点 **Operators** → **Installed Operators**。
  - b. 选择 **openshift-operators-redhat** 项目。
  - c. 点 **OpenShift Elasticsearch Operator**。
  - d. 点 **Subscription** → **Channel**。
  - e. 在 **Change Subscription Update Channel** 窗口中，选择 **5.0** 或 **stable-5.1** 并点 **Save**。
  - f. 等待几秒钟，然后点 **Operators** → **Installed Operators**。  
验证 OpenShift Elasticsearch Operator 版本是否为 5.x.x。  
  
等待 **Status** 的值变为 **Succeeded**。
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 **5.0** 或 **stable-5.1** 并点 **Save**。
  - f. 等待几秒钟，然后点 **Operators** → **Installed Operators**。  
验证 Red Hat OpenShift Logging Operator 版本是否为 5.0.x 或 5.1.x。  
  
等待 **Status** 的值变为 **Succeeded**。
3. 检查日志记录组件 :
  - a. 确保所有 Elasticsearch pod 都处于 **Ready** 状态 :

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

### 输出示例

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

- b. 确保 Elasticsearch 集群健康：

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

```
{
  "cluster_name" : "elasticsearch",
  "status" : "green",
}
```

- c. 确保创建了 Elasticsearch cron 任务：

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

```
$ oc get cronjob
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
elasticsearch-im-app	*/15 * * * *	False	0	<none>	56s
elasticsearch-im-audit	*/15 * * * *	False	0	<none>	56s
elasticsearch-im-infra	*/15 * * * *	False	0	<none>	56s

- d. 验证日志存储是否已更新至 5.0 或 5.1，并且索引是绿色的：

```
$ oc exec -c elasticsearch <any_es_pod_in_the_cluster> -- indices
```

验证输出是否包含 **app-00000x**、**infra-00000x**、**audit-00000x**、**.security** 索引。

#### 例 10.1. 带有绿色状态索引的输出示例

```
Tue Jun 30 14:30:54 UTC 2020
health status index                                uuid          pri rep
docs.count docs.deleted store.size pri.store.size
green open  infra-000008
bnBvUFEXTWi92z3zWAzieQ 3 1    222195      0    289      144
green open  infra-000004
rtDSzoqsSl6saisSK7Au1Q 3 1    226717      0    297      148
green open  infra-000012
RSf_kUwDSR2xEuKRZMPqZQ 3 1    227623      0    295      147
green open  .kibana_7
1SJdCqlZTPWIIAaOUd78yg 1 1     4           0    0         0
green open  infra-000010
iXwL3bnqTuGEABbUDA6OVw 3 1    248368      0    317      158
green open  infra-000009
YN9EsULWSNaxWeeNvOs0RA 3 1    258799      0    337      168
green open  infra-000014
YP0U6R7FQ_GVQQZ6Yh9lg 3 1    223788      0    292      146
green open  infra-000015
JRBbAbEmSMqK5X40df9HbQ 3 1    224371      0    291      145
green open  .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA 3 1     9           0    0         0
green open  infra-000007
llkkAVSzS0mosWTSAJM_hg 3 1    228584      0    296      148
green open  infra-000005
```

```

d9BoGQdiQASsS3BBFm2iRA 3 1 227987 0 297 148
green open infra-000003
goREK1QUKIQPAIVkWVaQ 3 1 226719 0 295 147
green open .security
zeT65uOuRTKZMjg_bbUc1g 1 1 5 0 0 0
green open .kibana-377444158_kubeadmin wvMhDwJkR-
mRZQO84K0gUQ 3 1 1 0 0 0
green open infra-000006 5H-
KBSXGQKiO7hdapDE23g 3 1 226676 0 295 147
green open infra-000001 eH53BQ-
bSxSWR5xYZB6IVg 3 1 341800 0 443 220
green open .kibana-6
RVp7TemSSemGJcsSUmf3A 1 1 4 0 0 0
green open infra-000011
J7XWBauWSTe0jnzX02fU6A 3 1 226100 0 293 146
green open app-000001
axSAFfONQDmKwatkjPXdtw 3 1 103186 0 126 57
green open infra-000016
m9c1iRLtStWSF1GopaRyCg 3 1 13685 0 19 9
green open infra-000002 Hz6WvINtTvKcQzw-
ewmbYg 3 1 228994 0 296 148
green open infra-000013 KR9mMFUpQl-
jraYtanyIGw 3 1 228166 0 298 148
green open audit-000001
eERqLdLmQOiQDFES1LBATQ 3 1 0 0 0 0

```

- e. 验证日志收集器是否已更新至 5.0 或 5.1 :

```
$ oc get ds fluentd -o json | grep fluentd-init
```

验证输出是否包含 **fluentd-init** 容器 :

```
"containerName": "fluentd-init"
```

- f. 使用 Kibana CRD 验证日志可视化工具是否已更新至 5.0 或 5.1 :

```
$ oc get kibana kibana -o json
```

验证输出是否包含具有 **ready** 状态的 Kibana Pod :

#### 例 10.2. 带有就绪 Kibana pod 的输出示例

```

[
  {
    "clusterCondition": {
      "kibana-5fdd766ffd-nb2jj": [
        {
          "lastTransitionTime": "2020-06-30T14:11:07Z",
          "reason": "ContainerCreating",
          "status": "True",
          "type": ""
        }
      ],
    }
  }
]

```

```

    "lastTransitionTime": "2020-06-30T14:11:07Z",
    "reason": "ContainerCreating",
    "status": "True",
    "type": ""
  }
]
},
"deployment": "kibana",
"pods": {
  "failed": [],
  "notReady": []
  "ready": []
},
"replicaSets": [
  "kibana-5fdd766ffd"
],
"replicas": 1
}
]

```

## 10.2. 将 OPENSIFT LOGGING 更新至当前版本

要将 OpenShift Logging 从 5.x 更新至当前版本，您需要更改 OpenShift Elasticsearch Operator 和 Red Hat OpenShift Logging Operator 的订阅。



### 重要

在更新 Red Hat OpenShift Logging Operator 前，您必须更新 OpenShift Elasticsearch Operator。您还必须将两个 Operator 更新至同一版本。

如果您以错误的顺序更新 Operator，则 Kibana 不会更新，并且不会创建 Kibana 自定义资源 (CR)。要临时解决这个问题，请删除 Red Hat OpenShift Logging Operator pod。当 Red Hat OpenShift Logging Operator pod 重新部署时，它会创建 Kibana CR 和 Kibana 再次可用。

### 先决条件

- OpenShift Container Platform 版本为 4.7 或更高版本。
- OpenShift Logging 处于健康状态：
  - 所有 pod 都为 **Ready** 状态。
  - Elasticsearch 集群处于健康状态。
- 您的 Elasticsearch 和 Kibana 数据已被备份。

### 流程

1. 更新 OpenShift Elasticsearch Operator:
  - a. 在 Web 控制台中，点 **Operators** → **Installed Operators**。
  - b. 选择 **openshift-operators-redhat** 项目。

- c. 点 **OpenShift Elasticsearch Operator**。
  - d. 点 **Subscription → Channel**。
  - e. 在 **Change Subscription Update Channel**窗口中，选择 **stable-5.x** 并点 **Save**。
  - f. 等待几秒钟，然后点 **Operators → Installed Operators**。  
验证 OpenShift Elasticsearch Operator 版本是否为 5.x.x。  
  
等待 **Status** 的值变为 **Succeeded**。
2. 更新 Red Hat OpenShift Logging Operator :
    - a. 在 Web 控制台中，点 **Operators → Installed Operators**。
    - b. 选择 **openshift-logging** 项目。
    - c. 点 **Red Hat OpenShift Logging Operator**。
    - d. 点 **Subscription → Channel**。
    - e. 在 **Change Subscription Update Channel**窗口中，选择 **stable-5.x** 并点 **Save**。
    - f. 等待几秒钟，然后点 **Operators → Installed Operators**。  
验证 Red Hat OpenShift Logging Operator 版本是否为 5.x.x。  
  
等待 **Status** 的值变为 **Succeeded**。

3. 检查日志记录组件 :

- a. 确保所有 Elasticsearch pod 都处于 **Ready** 状态 :

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

#### 输出示例

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

- b. 确保 Elasticsearch 集群健康 :

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

```
{
  "cluster_name" : "elasticsearch",
  "status" : "green",
}
```

- c. 确保创建了 Elasticsearch cron 任务 :

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

```
$ oc get cronjob
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
elasticsearch-im-app	*/15 * * * *	False	0	<none>	56s
elasticsearch-im-audit	*/15 * * * *	False	0	<none>	56s
elasticsearch-im-infra	*/15 * * * *	False	0	<none>	56s

d. 检查日志存储是否已更新至 5.x, 并且索引是绿色的 :

```
$ oc exec -c elasticsearch <any_es_pod_in_the_cluster> -- indices
```

验证输出是否包含 **app-00000x**、**infra-00000x**、**audit-00000x**、**.security** 索引。

### 例 10.3. 带有绿色状态索引的输出示例

```
Tue Jun 30 14:30:54 UTC 2020
health status index                                uuid                                pri rep
docs.count docs.deleted store.size pri.store.size
green open  infra-000008
bnBvUFEXTWi92z3zWAzieQ 3 1    222195    0    289    144
green open  infra-000004
rtDSzoqsSI6saisSK7Au1Q 3 1    226717    0    297    148
green open  infra-000012
RSf_kUwDSR2xEuKRZMPqZQ 3 1    227623    0    295    147
green open  .kibana_7
1SJdCqIZTPWIIAaOUd78yg 1 1     4         0    0       0
green open  infra-000010
iXwL3bnqTuGEABbUDa6OVw 3 1    248368    0    317    158
green open  infra-000009
YN9EsULWSNaxWeeNvOs0RA 3 1    258799    0    337    168
green open  infra-000014
YP0U6R7FQ_GVQVQZ6Yh9lg 3 1    223788    0    292    146
green open  infra-000015
JRBbAbEmSMqK5X40df9HbQ 3 1    224371    0    291    145
green open  .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA 3 1     9         0    0       0
green open  infra-000007
llkkAVSzMosWTSAJM_hg 3 1    228584    0    296    148
green open  infra-000005
d9BoGQdiQASsS3BBFm2iRA 3 1    227987    0    297    148
green open  infra-000003
goREK1QUKIQPAIVkWVaQ 3 1    226719    0    295    147
green open  .security
zeT65uOuRTKZMjg_bbUc1g 1 1     5         0    0       0
green open  .kibana-377444158_kubeadmin          wwMhDwJkR-
mRZQO84K0gUQ 3 1     1         0    0       0
green open  infra-000006
KBSXGQKiO7hdapDE23g 3 1    226676    0    295    147
green open  infra-000001
bSxSWR5xYZB6lVg 3 1    341800    0    443    220
green open  .kibana-6
RVp7TemSSemGJcsSUmuF3A 1 1     4         0    0       0
green open  infra-000011
J7XWBauWSTe0jnzX02fU6A 3 1    226100    0    293    146
```

```

green open app-000001
axSAFfONQDmKwatkjPXdtw 3 1 103186 0 126 57
green open infra-000016
m9c1iRLiStWSF1GopaRyCg 3 1 13685 0 19 9
green open infra-000002
ewmbYg 3 1 228994 0 296 148 Hz6WvINtTvKcQzw-
green open infra-000013
jraYtanyIGw 3 1 228166 0 298 148 KR9mMFUpQI-
green open audit-000001
eERqLdLmQOiQDFES1LBATQ 3 1 0 0 0 0

```

- e. 验证日志收集器是否已更新至 5.x:

```
$ oc get ds fluentd -o json | grep fluentd-init
```

验证输出是否包含 **fluentd-init** 容器：

```
"containerName": "fluentd-init"
```

- f. 使用 Kibana CRD 验证日志可视化工具是否已更新至 5.x：

```
$ oc get kibana kibana -o json
```

验证输出是否包含具有 **ready** 状态的 Kibana Pod：

#### 例 10.4. 带有就绪 Kibana pod 的输出示例

```

[
  {
    "clusterCondition": {
      "kibana-5fdd766ffd-nb2jj": [
        {
          "lastTransitionTime": "2020-06-30T14:11:07Z",
          "reason": "ContainerCreating",
          "status": "True",
          "type": ""
        },
        {
          "lastTransitionTime": "2020-06-30T14:11:07Z",
          "reason": "ContainerCreating",
          "status": "True",
          "type": ""
        }
      ]
    },
    "deployment": "kibana",
    "pods": {
      "failed": [],
      "notReady": [],
      "ready": []
    },
    "replicaSets": [
      "kibana-5fdd766ffd"
    ]
  }
]

```

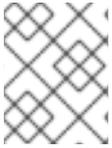
```
    ],  
    "replicas": 1  
  }  
]
```

## 第 11 章 查看集群仪表板

OpenShift Container Platform web 控制台中的 **Logging/Elasticsearch Nodes** 和 **OpenShift Logging** 仪表板显示有关 Elasticsearch 实例以及用于防止和诊断问题的单个 Elasticsearch 节点的详细信息。

**OpenShift Logging** 仪表板包含 chart，在集群级别显示 Elasticsearch 实例的详情，包括集群资源、垃圾回收、集群中的分片和 Fluentd 统计。

**Logging/Elasticsearch Nodes** 仪表板包含 charts，显示 Elasticsearch 实例的详情，很多在节点级别，包括索引、分片、资源等详情。



### 注意

如需更多详细数据，请点击仪表板中的 **Grafana UI** 链接来启动 Grafana 仪表板。Grafana 由 [OpenShift 集群监控](#) 提供。

### 11.1. 访问 ELASTISEARCH 和 OPENSIFT LOGGING 仪表板

您可以在 OpenShift Container Platform web 控制台中查看 **Logging/Elasticsearch Nodes** 和 **OpenShift Logging** 仪表板。

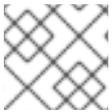
#### 流程

启动仪表板：

1. 在 OpenShift Container Platform web 控制台中点 **Monitoring → Dashboards**。
2. 在 **Dashboards** 页面中，从 **Dashboard** 菜单中选择 **Logging/Elasticsearch Nodes** 或 **OpenShift Logging**。  
对于 **Logging/Elasticsearch Nodes** 仪表板，可以选择您要查看的 Elasticsearch 节点并设置数据解析。

此时会显示正确的仪表板，显示多个数据图表。

3. (可选) 从 **Time Range** 和 **Refresh Interval** 菜单中选择不同时间范围来显示或刷新数据。



### 注意

如需了解更多详细数据，请点 **Grafana UI** 链接来启动 Grafana 仪表板。

有关仪表板图表的信息，请参阅 [关于 OpenShift Logging 仪表板](#) 和 [关于 Logging/Elasticsearch Nodes 仪表板](#)。

### 11.2. 关于 OPENSIFT LOGGING 仪表板

**OpenShift Logging** 仪表板包含 chart，可在集群级别显示 Elasticsearch 实例的详情，用于诊断和预期问题。

表 11.1. OpenShift Logging chart

指标	描述
----	----

指标	描述
Elastic 集群状态	当前的 Elasticsearch 状态： <ul style="list-style-type: none"> <li>● ONLINE - 表示 Elasticsearch 实例在线。</li> <li>● OFFLINE - 表示 Elasticsearch 实例离线。</li> </ul>
弹性节点	Elasticsearch 实例中的 Elasticsearch 节点总数。
Elastic 分片	Elasticsearch 实例中的 Elasticsearch 分片的总数。
Elastic 文档	Elasticsearch 实例中的 Elasticsearch 文档总数。
磁盘上的总索引大小	正在用于 Elasticsearch 索引的总磁盘空间。
Elastic 待处理的任務	Elasticsearch 尚未完成的更改总数，如索引创建、索引映射、分片分配或分片失败。
Elastic JVM GC 时间	JVM 在集群中执行 Elasticsearch 垃圾回收操作所需的时间。
Elastic JVM GC 率	JVM 每秒执行垃圾操作的次数总数。
Elastic Query/Fetch Latency Sum	<ul style="list-style-type: none"> <li>● Query latency: Elasticsearch 搜索查询执行的平均时间。</li> <li>● 获取延迟：每个 Elasticsearch 搜索查询的平均时间获取数据。</li> </ul> <p>获取延迟的时间通常比查询延迟要短。如果抓取延迟持续增加，则代表磁盘、数据配置速度较慢，或者带有许多结果的大量请求。</p>
Elastic 查询率	每个 Elasticsearch 节点每秒对 Elasticsearch 实例执行的查询总数。
CPU	Elasticsearch、Fluentd 和 Kibana 使用的 CPU 数量，显示了各个组件的 CPU 数量。
已使用的 Elastic JVM Heap	使用的 JVM 内存量。在一个健康的集群中，图形显示由 JVM 垃圾回收所释放的内存。
Elasticsearch 磁盘使用量	Elasticsearch 实例用于每个 Elasticsearch 节点的总磁盘空间。
使用中的文件描述符	Elasticsearch、Fluentd 和 Kibana 使用的文件描述符总数。

指标	描述
Fluentd emit 数量	Fluentd 默认输出每秒的 Fluentd 消息总数，以及默认输出的重试计数。
Fluentd 缓冲可用性	用于块的 Fluentd 缓冲的百分比。完整缓冲可能表示 Fluentd 无法处理收到的日志数量。
Elastic rx 字节	Elasticsearch 提供的 FluentD、Elasticsearch 节点和其它源的字节总数。
Elastic Index Failure Rate	Elasticsearch 索引失败的每秒总次数。高速率表示索引时出现问题。
Fluentd 输出错误率	FluentD 无法输出日志的每秒总次数。

### 11.3. LOGGING/ELASTICSEARCH 节点仪表板上的图表

**Logging/Elasticsearch Nodes** 仪表板包含 charts，显示 Elasticsearch 实例的详情（很多在节点级别），以进行进一步诊断。

#### Elasticsearch 状态

**Logging/Elasticsearch Nodes** 仪表板包含有关 Elasticsearch 实例状态的以下图表。

表 11.2. Elasticsearch 状态字段

指标	描述
集群状态	<p>在所选时间段内的集群健康状态，使用 Elasticsearch 绿色、黄色和红色代表：</p> <ul style="list-style-type: none"> <li>● 0 - 表示 Elasticsearch 实例处于绿色状态，这意味着分配了所有分片。</li> <li>● 1 - 表示 Elasticsearch 实例处于黄色状态，这意味着至少一个分片的副本分片不会被分配。</li> <li>● 2 - 表示 Elasticsearch 实例处于红色状态，这意味着至少不分配一个主分片及其副本。</li> </ul>
集群节点	集群中的 Elasticsearch 节点总数。
集群数据节点	集群中的 Elasticsearch 数据节点数量。
集群待定任务	集群状态更改的数量，这些更改尚未完成，并在集群队列中等待，例如索引创建、索引删除或分片分配。增长的倾向表示集群无法跟上变化。

## Elasticsearch 集群索引分片状态

每个 Elasticsearch 索引都是一个或多个分片的逻辑组，它们是持久化数据的基本单元。索引分片有两种类型：主分片和副本分片。当将文档索引为索引时，会将其保存在其主分片中，并复制到该分片的每个副本中。当索引被创建时，主分片的数量会被指定，在索引生命周期内这个数量不能改变。您可以随时更改副本分片的数量。

索引分片可能处于几个状态，具体取决于其生命周期阶段或集群中发生的事件。当分片能够执行搜索和索引请求时，分片就是活跃的。如果分片无法执行这些请求，分片就不是活跃的。如果分片正在初始化、重新分配、取消分配等等，分片可能不是活跃的。

索引分片由多个较小的内部块组成，称为索引片段，它们是数据的物理表示。索引片段是一个相对较小的不可变 Lucene 索引，它是 Lucene 提交新索引数据时生成的。Lucene 是 Elasticsearch 使用的搜索库，将索引片段合并到后台里的较大片段，从而使片段总数较低。如果合并片段的过程比生成新网段的速度慢，则可能表明问题。

当 Lucene 执行数据操作（如搜索操作）时，Lucene 会根据相关索引中的索引片段执行操作。为此，每个片段都包含在内存中载入并映射的特定数据结构。索引映射会对片段数据结构使用的内存有重大影响。

Logging/Elasticsearch Nodes 仪表板包含有关 Elasticsearch 索引分片的以下图表。

表 11.3. Elasticsearch 集群分片状态 chart

指标	描述
集群活跃分片	集群中活跃的主分片的数量和分片（包括副本）的总数。如果分片数量增加，集群性能就可以启动它。
集群初始化分片	集群中的非活跃分片数量。非活跃分片是正在初始化、被重新分配到不同节点或未分配的分片。集群通常具有非活跃分片（non-active 分片）的短时间。较长时间的非活跃分片数量增加可能代表有问题。
集群重新定位分片	Elasticsearch 重新定位到新节点的分片数量。Elasticsearch 由于多个原因重新定位节点，如在一个节点上或向集群中添加新节点时使用高内存。
集群未分配分片	未分配分片的数量。由于添加新索引或节点失败等原因，Elasticsearch 分片可能没有被分配。

## Elasticsearch 节点指标

每个 Elasticsearch 节点都有有限的资源，可用于处理任务。当所有资源都被已被使用，Elasticsearch 尝试执行新任务时，Elasticsearch 会将任务放入队列等待出现可用的资源。

Logging/Elasticsearch Nodes 仪表板包含以下有关所选节点的资源使用情况，以及 Elasticsearch 队列中等待的任务数量的图表。

表 11.4. Elasticsearch 节点指标图表

指标	描述
----	----

指标	描述
ThreadPool 任务	按任务类型显示的独立队列中等待的任务数量。在任何队列中的长期任务可能意味着节点资源短缺或其他问题。
CPU 用量	所选 Elasticsearch 节点使用的 CPU 量作为分配给主机容器的 CPU 总量的百分比。
内存用量	所选 Elasticsearch 节点使用的内存量。
磁盘用量	所选 Elasticsearch 节点上用于索引数据和元数据的总磁盘空间。
文档索引率	文档在所选 Elasticsearch 节点上索引的频率。
索引延迟	在所选 Elasticsearch 节点上索引文档所需时间。索引延迟会受到很多因素的影响，如 JVM Heap 内存和整个负载。延迟增加代表实例中资源容量不足。
搜索率	在所选 Elasticsearch 节点上运行的搜索请求数量。
搜索延迟	在所选 Elasticsearch 节点上完成搜索请求的时间。搜索延迟可能会受到很多因素的影响。延迟增加代表实例中资源容量不足。
文档计数（包括副本）	存储在所选 Elasticsearch 节点上的 Elasticsearch 文档数量，包括存储在主分片和节点上分配的副本分片中的文档。
文档删除速率	要从分配给所选 Elasticsearch 节点的任何索引分片中删除 Elasticsearch 文档的数量。
文档合并率	分配给所选 Elasticsearch 节点的任何索引分片中合并的 Elasticsearch 文档数量。

### Elasticsearch 节点 fielddata

*Fielddata* 是一个 Elasticsearch 数据结构，它以索引形式保存术语列表，并保存在 JVM 堆中。因为 *fielddata* 构建非常昂贵，所以 Elasticsearch 会缓存 *fielddata* 结构。当底层索引分段被删除或合并时，或者没有足够 JVM HEAP 内存用于所有 *fielddata* 缓存时，Elasticsearch 可以驱除 *fielddata* 缓存。

Logging/Elasticsearch Nodes 仪表板包含有关 Elasticsearch 字段数据的以下图表。

表 11.5. Elasticsearch 节点字段数据图表

指标	描述
----	----

指标	描述
Fielddata 内存大小	用于所选 Elasticsearch 节点上的 fielddata 缓存的 JVM 堆数量。
Fielddata 驱除	从所选 Elasticsearch 节点中删除的 fielddata 结构数量。

### Elasticsearch 节点查询缓存

如果索引中存储的数据没有改变，搜索查询结果会在节点级别的查询缓存中缓存，以便 Elasticsearch 重复使用。

Logging/Elasticsearch Nodes 仪表板包含有关 Elasticsearch 节点查询缓存的以下图表。

表 11.6. Elasticsearch 节点查询图表

指标	描述
查询缓存大小	用于查询缓存的内存总量，用于分配给所选 Elasticsearch 节点的所有分片。
查询缓存驱除	所选 Elasticsearch 节点上的查询缓存驱除数量。
查询缓存点击	所选 Elasticsearch 节点上的查询缓存数量。
查询缓存丢失	所选 Elasticsearch 节点上丢失的查询缓存数。

### Elasticsearch 索引节流

在索引文档时，Elasticsearch 将文档存储在索引片段中，这些部分是数据的物理表示。同时，Elasticsearch 会定期将较小的片段合并到较大的片段中，以优化资源使用。如果索引速度更快，那么合并过程就无法迅速完成，从而导致搜索和性能出现问题。为了防止这种情况，Elasticsearch 节流（throttles）的索引通常是通过减少分配给索引到单个线程的线程数量来实现的。

Logging/Elasticsearch Nodes 仪表板包含有关 Elasticsearch 索引节流的以下图表。

表 11.7. 索引节流图表

指标	描述
索引节流	Elasticsearch 在所选 Elasticsearch 节点上节流索引操作的时间。
合并节流	Elasticsearch 在所选 Elasticsearch 节点上节流部署片段合并操作的时间。

### 节点 JVM 堆统计

Logging/Elasticsearch Nodes 仪表板包含以下有关 JVM Heap 操作的图表。

表 11.8. JVM Heap 统计图表

指标	描述
使用的堆	所选 Elasticsearch 节点上分配的 JVM 堆空间量。
GC 计数	在所选 Elasticsearch 节点上运行的垃圾回收操作数量，包括旧垃圾回收量。
GC 时间	JVM 在所选 Elasticsearch 节点上运行垃圾回收操作的时间、旧的垃圾回收时间。

## 第 12 章 日志故障排除

### 12.1. 查看 OPENSIFT LOGGING 状态

您可以查看 Red Hat OpenShift Logging Operator 的状态以及多个 OpenShift Logging 组件的状态。

#### 12.1.1. 查看 Red Hat OpenShift Logging Operator 的状态

您可以查看 Red Hat OpenShift Logging Operator 的状态。

##### 先决条件

- 必须安装 OpenShift Logging 和 Elasticsearch。

##### 流程

1. 进入 **openshift-logging** 项目。

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

2. 查看 OpenShift Logging 状态：

- a. 获取 OpenShift Logging 状态：

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

##### 输出示例

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

```
- fluentd-flpnn
- fluentd-n2frh
logstore: ③
elasticsearchStatus:
- ShardAllocationEnabled: all
cluster:
  activePrimaryShards: 5
  activeShards: 5
  initializingShards: 0
  numDataNodes: 1
  numNodes: 1
  pendingTasks: 0
  relocatingShards: 0
  status: green
  unassignedShards: 0
clusterName: elasticsearch
nodeConditions:
  elasticsearch-cdm-mkkdys93-1:
nodeCount: 1
pods:
  client:
    failed:
    notReady:
    ready:
    - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
  data:
    failed:
    notReady:
    ready:
    - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
  master:
    failed:
    notReady:
    ready:
    - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
visualization: ④
kibanaStatus:
- deployment: kibana
pods:
  failed: []
  notReady: []
  ready:
  - kibana-7fb4fd4cc9-f2nls
replicaSets:
- kibana-7fb4fd4cc9
replicas: 1
```

- ① 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 Elasticsearch Pod 的相关信息，包括 Elasticsearch 集群健康状态 **green**、**yellow** 或 **red**。
- ④ Kibana Pod 的相关信息。

### 12.1.1.1. 情况消息示例

以下是来自 OpenShift Logging 实例的 **Status.Nodes** 部分的一些情况消息示例。

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：

#### 输出示例

```
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：

#### 输出示例

```
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：

#### 输出示例

```
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：

### 输出示例

```

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 Fluentd Pod，因为节点选择器与任何节点都不匹配：

### 输出示例

```

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

```

## 12.1.2. 查看 OpenShift Logging 组件的状态

您可以查看多个 OpenShift Logging 组件的状态。

### 先决条件

- 必须安装 OpenShift Logging 和 Elasticsearch。

### 流程

1. 进入 **openshift-logging** 项目。

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

2. 查看 OpenShift Logging 环境的状态：

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

### 输出示例

```
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 OpenShift Logging 副本集的状态：

- a. 获取副本集的名称：

### 输出示例

```
$ oc get replicaset
```

### 输出示例

```
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. 获取副本集的状态：

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

### 输出示例

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

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

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

## 12.2. 查看日志存储的状态

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

### 12.2.1. 查看日志存储的状态

您可以查看日志存储的状态。

#### 先决条件

- 必须安装 OpenShift Logging 和 Elasticsearch。

#### 流程

1. 进入 **openshift-logging** 项目。

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

2. 查看状态：

a. 获取日志存储实例的名称：

```
$ oc get Elasticsearch
```

### 输出示例

```
NAME          AGE
elasticsearch 5h9m
```

b. 获取日志存储状态：

```
$ 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：

### 输出示例

```
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 日志存储的状态：
  - 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。
  - 保存数据节点的日志数量。
  - 日志存储节点的总数。
  - 待处理的任务数量。
  - 日志存储状态：**green**、**red**、**yellow**。
  - 未分配分片的数量。
- 3 任何状态条件（若存在）。日志存储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：
  - 容器正在等待日志存储和代理容器。
  - 日志存储和代理容器的容器终止。
  - 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 集群中的日志存储节点，带有 **upgradeStatus**。
- 5 集群中的日志存储客户端、数据和 master 节点，列在 **failed**、**notReady** 或 **ready** 状态下。

### 12.2.1.1. 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。

以下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: {}
```

以下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: {}

```

以下状态消息表示 CR 中的日志存储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

```

以下状态消息表示日志存储 CR 使用了不存在的持久性卷声明 (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

```

以下状态消息表示日志存储集群没有足够的节点来支持冗余策略。

```

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有太多 control plane 节点（也称为 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
```

以下状态消息表示 Elasticsearch 存储不支持您尝试进行的更改。

例如：

```
status:
clusterHealth: green
conditions:
- lastTransitionTime: "2021-05-07T01:05:13Z"
message: Changing the storage structure for a custom resource is not supported
reason: StorageStructureChangelgnored
status: 'True'
type: StorageStructureChangelgnored
```

**reason** 和 **type** 类型字段指定不受支持的更改类型：

#### **StorageClassNameChangelgnored**

不支持更改存储类名称。

#### **StorageSizeChangelgnored**

不支持更改存储大小。

#### **StorageStructureChangelgnored**

不支持在临时存储结构和持久性存储结构间更改。



#### **重要**

如果您将 **ClusterLogging** 自定义资源 (CR) 配置为从临时切换到持久性存储，OpenShift Elasticsearch Operator 会创建一个持久性卷声明 (PVC)，但不创建持久性卷 (PV)。要清除 **StorageStructureChangelgnored** 状态，您必须恢复对 **ClusterLogging** CR 的更改并删除 PVC。

### 12.2.2. 查看日志存储组件的状态

您可以查看多个日志存储组件的状态。

#### **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
```

#### **输出示例**

```
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-4vjor49p-2-6d4d7db474-q2w7z -- indices
```

### 输出示例

```
Defaulting container name to elasticsearch.
Use 'oc describe pod/elasticsearch-cdm-4vjor49p-2-6d4d7db474-q2w7z -n openshift-logging' to see all of the containers in this pod.

green open infra-000002                                S4QANnf1QP6NgCegfnrnBQ
3 1 119926 0 157 78
green open audit-000001                                8_EQx77iQCSTzFOXtxRqFw
3 1 0 0 0 0
green open .security                                    iDjSCH7aSUGhldq0LheLBQ 1
1 5 0 0 0
green open .kibana_-377444158_kubeadmin                 yBywZ9GfSrKebz5gWBZbjw 3 1 1 0 0 0
green open infra-000001                                z6Dpe__ORgiopEpW6YI44A
3 1 871000 0 874 436
green open app-000001                                   hlrazQCeSISewG3c2VlvsQ
3 1 2453 0 3 1
green open .kibana_1                                    JCitcBMSQxKOvlq6iQW6wg
1 1 0 0 0 0
green open .kibana_-1595131456_user1                   glYFIEGRRRe-
ka0W3okS-mQ 3 1 1 0 0 0
```

### 日志存储 pod

您可以查看托管日志存储的 pod 的状态。

1. 获取 Pod 的名称：

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

### 输出示例

```
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：

### 输出示例

```
....
Status:      Running

....

Containers:
  elasticsearch:
```

```

Container ID: cri-o://b7d44e0a9ea486e27f47763f5bb4c39dfd2
State:      Running
  Started:   Mon, 08 Jun 2020 10:17:56 -0400
  Ready:     True
  Restart Count: 0
  Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
             period=5s #success=1 #failure=3

....

proxy:
  Container ID: cri-
  o://3f77032abaddbb1652c116278652908dc01860320b8a4e741d06894b2f8f9aa1
  State:     Running
  Started:   Mon, 08 Jun 2020 10:18:38 -0400
  Ready:     True
  Restart Count: 0

....

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

....

Events:      <none>

```

## 日志存储 pod 部署配置

您可以查看日志存储部署配置的状态。

1. 获取部署配置的名称：

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

### 输出示例

```

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：

### 输出示例

```

....
Containers:

```

```

    elasticsearch:
      Image: registry.redhat.io/openshift-logging/elasticsearch6-rhel8
      Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
      period=5s #success=1 #failure=3
    ....

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

Events:          <none>

```

### 日志存储副本集

您可以查看日志存储副本集的状态。

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：

#### 输出示例

```

....
Containers:
  elasticsearch:
    Image: registry.redhat.io/openshift-logging/elasticsearch6-
    rhel8@sha256:4265742c7cdd85359140e2d7d703e4311b6497eec7676957f455d6908e7b1
    c25
    Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
    period=5s #success=1 #failure=3
  ....

Events:          <none>

```

### 12.3. 了解 OPENSIFT LOGGING 警报

所有日志记录收集器警报都列在 OpenShift Container Platform Web 控制台的 Alerting UI 中。

### 12.3.1. 查看日志记录收集器警报

警报显示在 OpenShift Container Platform web 控制台中,在 Alerting UI 的 **Alerts** 选项卡中显示。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。

#### 流程

查看 OpenShift Logging 和其他 OpenShift Container Platform 警报：

1. 在 OpenShift Container Platform 控制台中点 **Monitoring** → **Alerting**。
2. 点击 **Alerts** 标签页。根据所选择的过滤器，列出警报。

#### 其他资源

- 如需有关 Alerting UI 的更多信息，请参阅[管理警报](#)。

### 12.3.2. 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。您可以在 OpenShift Container Platform web 控制台的 Alerting UI 的 **Alerts** 页面中查看这些警报。

表 12.1. Fluentd Prometheus 警报

警报	消息	描述	重要性
<b>FluentDHighErrorRate</b>	<b>&lt;value&gt; of records have resulted in an error by fluentd &lt;instance&gt;.</b>	FluentD 输出错误数量很高，在前 15 分钟中默认超过 10。	Warning
<b>FluentdNodeDown</b>	<b>Prometheus could not scrape fluentd &lt;instance&gt; for more than 10m.</b>	Fluentd 报告 Prometheus 可能无法抓取特定的 Fluentd 实例。	Critical
<b>FluentdQueueLengthIncreasing</b>	<b>In the last 12h, fluentd &lt;instance&gt; buffer queue length constantly increased more than 1.Current value is &lt;value&gt;.</b>	Fluentd 报告队列大小正在增加。	Critical
<b>FluentDVeryHighErrorRate</b>	<b>&lt;value&gt; of records have resulted in an error by fluentd &lt;instance&gt;.</b>	FluentD 输出错误的数量非常大，在之前的 15 分钟中，默认情况下超过 25 个。	Critical

### 12.3.3. 关于 Elasticsearch 警报规则

您可以在 Prometheus 中查看这些警报规则。

表 12.2. 警报规则

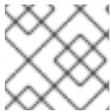
警报	描述	重要性
<b>ElasticsearchClusterNotHealthy</b>	集群健康状态处于 RED 至少 2 分钟。集群不接受写操作，分片可能缺失，或者 master 节点尚未选定。	Critical
<b>ElasticsearchClusterNotHealthy</b>	集群健康状态为 YELLOW 至少 20 分钟。某些分片副本尚未分配。	警告
<b>ElasticsearchDiskSpaceRunningLow</b>	集群预期在以后的 6 小时内处于磁盘空间之外。	Critical
<b>ElasticsearchHighFileDescriptorUsage</b>	在下一个小时内，集群预计会在下一个小时内消耗掉所有文件描述符。	警告
<b>ElasticsearchJVMHeapUseHigh</b>	指定节点上的 JVM 堆使用率很高。	警报
<b>ElasticsearchNodeDiskWatermarkReached</b>	由于可用磁盘空间较低，指定节点达到低水位线。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。	info
<b>ElasticsearchNodeDiskWatermarkReached</b>	由于可用磁盘空间较低，指定节点达到高水位线。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。	警告
<b>ElasticsearchNodeDiskWatermarkReached</b>	由于可用磁盘空间不足，指定节点达到洪水水位线。每个在这个节点上分配了分片的索引都会强制使用只读块。当磁盘使用低于高水位线时，索引块必须手动发布。	Critical
<b>ElasticsearchJVMHeapUseHigh</b>	指定节点上的 JVM 堆使用率太高。	警报
<b>ElasticsearchWriteRequestsRejectionJumps</b>	Elasticsearch 在指定节点上的写入增加。此节点可能无法跟上索引速度。	警告
<b>AggregatedLoggingSystemCPUHigh</b>	该系统在指定节点上使用的 CPU 太高。	警报
<b>ElasticsearchProcessCPUHigh</b>	Elasticsearch 在指定节点上使用的 CPU 太高。	警报

## 12.4. 为红帽支持收集日志记录数据

在提交问题单时同时提供您的集群信息，可以帮助红帽支持为您进行排除故障。

您可使用 [must-gather 工具](#) 来收集有关项目级别资源、集群级别资源和每个 OpenShift Logging 组件的诊断信息。

为了获得快速支持，请提供 OpenShift Container Platform 和 OpenShift Logging 的诊断信息。



### 注意

不要使用 `hack/logging-dump.sh` 脚本。这个脚本不再被支持且不收集数据。

## 12.4.1. 关于 `must-gather` 工具

`oc adm must-gather` CLI 命令会收集最有助于解决问题的集群信息。

对于 OpenShift Logging 环境，`must-gather` 会收集以下信息：

- 项目级别资源，包括 Pod、配置映射、服务帐户、角色、角色绑定和事件
- 集群级资源，包括集群级别的节点、角色和角色绑定
- `openshift-logging` 和 `openshift-operators-redhat` 命名空间中的 OpenShift Logging 资源，包括日志收集器的健康状况、日志存储和日志可视化工具

在运行 `oc adm must-gather` 时，集群上会创建一个新 pod。在该 pod 上收集数据，并保存至以 `must-gather.local` 开头的一个新目录中。此目录在当前工作目录中创建。

## 12.4.2. 先决条件

- 必须安装 OpenShift Logging 和 Elasticsearch。

## 12.4.3. 收集 OpenShift Logging 数据

您可以使用 `oc adm must-gather` CLI 命令来收集有关 OpenShift Logging 环境的信息。

### 流程

使用 `must-gather` 来收集 OpenShift Logging 信息：

1. 进入要存储 `must-gather` 信息的目录。
2. 针对 OpenShift Logging 镜像运行 `oc adm must-gather` 命令：

```
$ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator -o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-operator")].image}')
```

`must-gather` 工具会创建一个以当前目录中 `must-gather.local` 开头的新目录。例如：`must-gather.local.4157245944708210408`。

3. 从刚刚创建的 `must-gather` 目录创建一个压缩文件。例如，在使用 Linux 操作系统的计算机上运行以下命令：

```
$ tar -cvaf must-gather.tar.gz must-gather.local.4157245944708210408
```

4. 在[红帽客户门户](#)中为您的问题单附上压缩文件。

## 12.5. 关键警报故障排除

## 12.5.1. Elasticsearch Cluster Health 是红色

至少一个主分片及其副本没有分配给节点。

### 故障排除

1. 检查 Elasticsearch 集群健康状态，并验证集群的 **status** 是否为红色。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- health
```

2. 列出已加入集群的节点。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/nodes?v
```

3. 列出 Elasticsearch Pod，并将它们与上一步中命令输出中的节点进行比较。

```
oc -n openshift-logging get pods -l component=elasticsearch
```

4. 如果某些 Elasticsearch 节点没有加入集群，请执行以下步骤。

- a. 确认 Elasticsearch 已选定 control plane 节点。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/master?v
```

- b. 检查所选 control plane 节点的 pod 日志问题。

```
oc logs <elasticsearch_master_pod_name> -c elasticsearch -n openshift-logging
```

- c. 检查尚未加入集群的节点日志。

```
oc logs <elasticsearch_node_name> -c elasticsearch -n openshift-logging
```

5. 如果所有节点都加入集群，请执行以下步骤，检查集群是否正在进行恢复。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/recovery?active_only=true
```

如果没有命令输出，恢复过程可能会因为待处理的任务而延迟或停止。

6. 检查是否有待处理的任务。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- health |grep
number_of_pending_tasks
```

7. 如果有待处理的任务，请监控其状态。

如果它们的状态发生变化，并且表示集群正在恢复，请继续等待。恢复时间因集群大小和其它因素而异。

否则，如果待处理任务的状态没有改变，这表示恢复已停止。

8. 如果恢复似乎已停止，请检查 **cluster.routing.allocation.enable** 是否设置为 **none**。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/settings?pretty
```

9. 如果 **cluster.routing.allocation.enable** 设为 **none**，请将它设置为 **all**。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/settings?pretty -X PUT -d '{"persistent":
{"cluster.routing.allocation.enable":"all"}}'
```

10. 检查哪些索引仍然是红色的。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/indices?v
```

11. 如果有任何索引仍然是红色的，请尝试通过执行以下步骤清除它们。

- a. 清除缓存。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_cache/clear?pretty
```

- b. 增加最大分配重试数。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_settings?pretty -X PUT -d
'{"index.allocation.max_retries":10}'
```

- c. 删除所有滚动项目。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_search/scroll/_all -X DELETE
```

- d. 增加超时时间。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_settings?pretty -X PUT -d
'{"index.unassigned.node_left.delayed_timeout":"10m"}'
```

12. 如果前面的步骤没有清除红色索引，请单独删除索引。

- a. 标识红色索引名称。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/indices?v
```

- b. 删除红色索引。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_red_index_name> -X DELETE
```

13. 如果没有红色索引且集群状态为红色，请在数据节点上检查是否有连续重量处理负载。

- a. 检查 Elasticsearch JVM 堆使用量是否很高。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_nodes/stats?pretty
```

在命令输出中，检查 `node_name.jvm.mem.heap_used_percent` 字段，以确定 JVM Heap 使用量。

- b. 检查高 CPU 使用率。

### 其他资源

- 在 Elasticsearch 中搜索 "Free up or increase disk space"，[Fix a red or yellow cluster status](#)。

## 12.5.2. Elasticsearch Cluster Health 为黄色

至少一个主分片的副本分片没有分配给节点。

### 故障排除

1. 通过调整 `ClusterLogging` CR 中的 `nodeCount` 来增加节点数。

### 其他资源

- [集群日志记录自定义资源 \(CR\)](#)
- [为日志存储配置持久性存储](#)
- 在 Elasticsearch 中搜索 "Free up or increase disk space"，[Fix a red or yellow cluster status](#)。

## 12.5.3. 已达到 Elasticsearch 节点磁盘低水位线

Elasticsearch 不将分片分配给达到低水位线的节点。

### 故障排除

1. 识别要在其上部署 Elasticsearch 的节点。

```
oc -n openshift-logging get po -o wide
```

2. 检查是否有未分配的分片。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/health?pretty | grep unassigned_shards
```

3. 如果存在未分配的分片，请检查每个节点上的磁盘空间。

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

4. 检查 `nodes.node_name.fs` 字段，以确定该节点上的可用磁盘空间。  
如果使用的磁盘百分比超过 85%，则节点已超过低水位线，并且分片无法再分配给此节点。
5. 尝试增加所有节点上的磁盘空间。

6. 如果无法增加磁盘空间，请尝试向集群添加新数据节点。
7. 如果添加新数据节点有问题，请减少集群冗余策略总数。
  - a. 检查当前的 **redundancyPolicy**。

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```



### 注意

如果使用 **ClusterLogging** CR，请输入：

```
oc -n openshift-logging get cl -o jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

- b. 如果集群 **redundancyPolicy** 大于 **SingleRedundancy**，将其设置为 **SingleRedundancy** 并保存这个更改。
8. 如果前面的步骤没有解决这个问题，请删除旧的索引。

- a. 检查 Elasticsearch 上所有索引的状态。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

- b. 确定可以删除的旧索引。

- c. 删除索引。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util -- query=<elasticsearch_index_name> -X DELETE
```

## 其他资源

- 在 [About the Cluster Logging custom resource](#) 的 "Sample **ClusterLogging** custom resource (CR)" 中搜索 "redundancyPolicy"

### 12.5.4. 已达到 Elasticsearch 节点磁盘高水位线

Elasticsearch 会尝试从**已达到高水位线**的节点中重新定位分片。

## 故障排除

1. 识别要在其上部署 Elasticsearch 的节点。

```
oc -n openshift-logging get po -o wide
```

2. 检查每个节点上的磁盘空间。

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

3. 检查集群是否重新平衡。

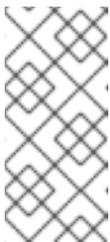
```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/health?pretty | grep relocating_shards
```

如果命令输出显示重新定位分片，则代表超过了高水位线（High Watermark）。High Watermark 的默认值为 90%。

分片重新定位到磁盘使用率较低且未超过任何水位线阈值的节点。

4. 若要将分片分配到特定节点，请释放一些空间。
5. 尝试增加所有节点上的磁盘空间。
6. 如果无法增加磁盘空间，请尝试向集群添加新数据节点。
7. 如果添加新数据节点有问题，请减少集群冗余策略总数。
  - a. 检查当前的 **redundancyPolicy**。

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```



### 注意

如果使用 **ClusterLogging** CR，请输入：

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

- b. 如果集群 **redundancyPolicy** 大于 **SingleRedundancy**，将其设置为 **SingleRedundancy** 并保存这个更改。
8. 如果前面的步骤没有解决这个问题，请删除旧的索引。
    - a. 检查 Elasticsearch 上所有索引的状态。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

- b. 确定可以删除的旧索引。
- c. 删除索引。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

### 其他资源

- 在 [About the Cluster Logging custom resource](#) 的 "Sample **ClusterLogging** custom resource (CR)" 中搜索 "redundancyPolicy"

### 12.5.5. 已达到 Elasticsearch 节点磁盘的洪水水位线

Elasticsearch 在每个具有这两个条件的索引中强制使用只读索引块：

- 为节点分配一个或多个分片。
- 一个个或多个磁盘超过 [flood stage](#)。

## 故障排除

1. 检查 Elasticsearch 节点的磁盘空间。

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

检查 **nodes.node\_name.fs** 字段，以确定该节点上的可用磁盘空间。

2. 如果使用的磁盘百分比超过 95%，这表明该节点已跨过洪水水位线。对于在此特定节点上分配的分片，写入被阻止。
3. 尝试增加所有节点上的磁盘空间。
4. 如果无法增加磁盘空间，请尝试向集群添加新数据节点。
5. 如果添加新数据节点有问题，请减少集群冗余策略总数。
  - a. 检查当前的 **redundancyPolicy**。

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```



### 注意

如果使用 **ClusterLogging** CR，请输入：

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

- b. 如果集群 **redundancyPolicy** 大于 **SingleRedundancy**，将其设置为 **SingleRedundancy** 并保存这个更改。
6. 如果前面的步骤没有解决这个问题，请删除旧的索引。
    - a. 检查 Elasticsearch 上所有索引的状态。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

- b. 确定可以删除的旧索引。
- c. 删除索引。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

7. 继续释放和监控磁盘空间，直到使用的磁盘空间下降到 90% 以下。然后，取消阻塞写入此特定节点。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_all/_settings?pretty -X PUT -d '{"index.blocks.read_only_allow_delete": null}'
```

## 其他资源

- 在 [About the Cluster Logging custom resource](#) 的 "Sample **ClusterLogging** custom resource (CR)" 中搜索 "redundancyPolicy"

### 12.5.6. Elasticsearch JVM 堆使用率是高

使用的 Elasticsearch 节点 JVM Heap 内存超过 75%。

#### 故障排除

考虑[增大堆大小](#)。

### 12.5.7. 聚合日志记录系统 CPU 为高

节点上的系统 CPU 使用率高。

#### 故障排除

检查集群节点的 CPU。考虑向节点分配更多 CPU 资源。

### 12.5.8. Elasticsearch 进程 CPU 为高

节点上的 Elasticsearch 进程 CPU 使用率很高。

#### 故障排除

检查集群节点的 CPU。考虑向节点分配更多 CPU 资源。

### 12.5.9. Elasticsearch 磁盘空间现为低

根据当前的磁盘使用情况，Elasticsearch 集群预计在以后的 6 小时内会耗尽磁盘空间。

#### 故障排除

1. 获取 Elasticsearch 节点的磁盘空间。

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

2. 在命令输出中，检查 **nodes.node\_name.fs** 字段以确定该节点上的可用磁盘空间。
3. 尝试增加所有节点上的磁盘空间。
4. 如果无法增加磁盘空间，请尝试向集群添加新数据节点。
5. 如果添加新数据节点有问题，请减少集群冗余策略总数。
  - a. 检查当前的 **redundancyPolicy**。

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```



### 注意

如果使用 **ClusterLogging** CR，请输入：

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

- b. 如果集群 **redundancyPolicy** 大于 **SingleRedundancy**，将其设置为 **SingleRedundancy** 并保存这个更改。
6. 如果前面的步骤没有解决这个问题，请删除旧的索引。
    - a. 检查 Elasticsearch 上所有索引的状态。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

- b. 确定可以删除的旧索引。
- c. 删除索引。

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

### 其他资源

- 在 [About the Cluster Logging custom resource](#) 的 "Sample **ClusterLogging** custom resource (CR)" 中搜索 "redundancyPolicy"
- 在 [关于 Elasticsearch 警报规则](#) 中搜索 "ElasticsearchDiskSpaceRunningLow"。
- 在 Elasticsearch 中搜索 "Free up or increase disk space"，[Fix a red or yellow cluster status](#)。

### 12.5.10. Elasticsearch FileDescriptor 使用为高

根据当前的使用趋势，预计节点上的文件描述符数量不足。

### 故障排除

检查并根据需要为每个节点配置 **max\_file\_descriptors** 的值，如 Elasticsearch [文件描述符](#) 主题中所述。

### 其他资源

- 在 [关于 Elasticsearch 警报规则](#) 中搜索 "ElasticsearchHighFileDescriptorUsage"。
- 在 [OpenShift Logging 仪表盘](#) 中搜索 "使用中的文件描述符"。

## 第 13 章 卸载 OPENSIFT LOGGING

您可以从 OpenShift Container Platform 集群中删除 OpenShift Logging。

### 13.1. 从 OPENSIFT CONTAINER PLATFORM 卸载 OPENSIFT LOGGING

您可以通过删除 **ClusterLogging** 自定义资源(CR)来停止日志聚合。在删除 CR 后，还有其他 OpenShift Logging 组件保留下来，您可以选择将其删除。

删除 **ClusterLogging** CR 不会删除持久性卷声明 (PVC)。要保留或删除剩余的 PVC、持久性卷 (PV) 和相关数据，您必须执行进一步操作。

#### 先决条件

- 必须安装 OpenShift Logging 和 Elasticsearch。

#### 流程

删除 OpenShift Logging:

1. 使用 OpenShift Container Platform Web 控制台删除 **ClusterLogging** CR:

- 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 Details** 页面中点 **Instances**。

- d. 单击实例旁的 Options 菜单 ，然后选择 **Delete ClusterLogging**。

2. 可选：删除自定义资源定义(CRD):

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

- b. 单击 **ClusterLogForwarder** 旁边的 Options 菜单 ，然后选择 **Delete Custom Resource Definition**。

- c. 单击 **ClusterLogging** 旁边的 Options 菜单 ，然后选择 **Delete Custom Resource Definition**。

- d. 单击 **Elasticsearch** 旁边的 Options 菜单 ，然后选择 **Delete Custom Resource Definition**。

3. 可选：删除 Red Hat OpenShift Logging Operator 和 OpenShift Elasticsearch Operator :

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

- b. 点 Red Hat OpenShift Logging Operator  旁边的 Options 菜单并选择 **Uninstall Operator**。
- c. 点击 OpenShift Elasticsearch Operator  旁边的 Options 菜单并选择 **Uninstall Operator**。
4. 可选：删除 OpenShift Logging 和 Elasticsearch 项目。
  - a. 切换到 **Home → Projects** 页面。
  - b. 点击 **openshift-logging** 项目  旁的 Options 菜单,然后选择 **Delete Project**。
  - c. 在对话框中输入 **openshift-logging** 并点 **Delete** 来确认删除。
  - d. 点击 **openshift-operators-redhat** 项目  旁的 Options 菜单并选择 **Delete Project**。



### 重要

如果在此命名空间中安装了其他全局 Operator，请不要删除 **openshift-operators-redhat** 项目。

- e. 通过在对话框中输入 **openshift-operators-redhat** 并点 **Delete** 来确认删除。
5. 要保留 PVC 以便与其他 pod 重复使用，保留标签或 PVC 名称，以便重新声明 PVC。
6. 可选：如果您不想保留 PVC，可以删除它们。



### 警告

释放或删除 PVC 可能会导致 PV 删除并导致数据丢失。

- a. 切换到 **Storage → Persistent Volume Claims** 页面。
- b. 点击每个 PVC  旁边的 Options 菜单，然后选择 **Delete Persistent Volume Claim**。
- c. 如果要恢复存储空间，可以删除 PV。

### 其他资源

- [手动重新声明持久性卷](#)

## 第 14 章 日志记录字段

以下字段可出现在 OpenShift Logging 导出的日志记录中。虽然日志记录通常格式为 JSON 对象，但相同的数据模型可以应用到其他编码。

要从 Elasticsearch 和 Kibana 搜索这些字段，在搜索时使用完整的点号字段名称。例如，使用 Elasticsearch `/_search` URL，若要查找 Kubernetes pod 名称，请使用 `/_search/q=kubernetes.pod_name:name-of-my-pod`。

顶级字段可以出现在每条记录中。

## 第 15 章 MESSAGE

原始日志条目文本 UTF-8 编码。如果存在非空的 **structured** 字段，则此字段可能不存在或为空。请参见关于结构化的描述，了解更多。

数据类型	text
示例值	<b>HAPPY</b>

## 第 16 章 结构化

原始日志条目作为结构化对象.如果转发器配置为解析结构化 JSON 日志,则可能存在此字段。如果原始日志条目是有效的结构化日志,此字段将包含等同的 JSON 结构。否则此字段为空或不存在,**message** 字段将包含原始日志消息。**structured** 字段可以包含日志消息中包含的任何子字段,此处没有定义任何限制。

数据类型	group
示例值	map[message:starting fluentd worker pid=21631 ppid=21618 worker=0 pid:21631 ppid:21618 worker:0]

## 第 17 章 @TIMESTAMP

一个 UTC 值，用于标记日志有效负载创建的时间，如果创建时间未知，则标记首次收集日志有效负载的时间。"@前缀表示为特定用途保留的字段。默认情况下，大多数工具都通过 Elasticsearch 来查找"@timestamp"。

数据类型	date
示例值	<b>2015-01-24 14:06:05.071000000 Z</b>

## 第 18 章 HOSTNAME

此日志消息的来源主机的名称。在 Kubernetes 集群中，这与 `kubernetes.host` 相同。

数据类型	关键字
------	-----

## 第 19 章 IPADDR4

源服务器的 IPv4 地址。可以是一个数组。

数据类型	ip
------	----

## 第 20 章 IPADDR6

源服务器的 IPv6 地址（如果可用）。可以是一个数组。

数据类型	ip
------	----

## 第 21 章 LEVEL

来自各种来源的日志记录级别，包括 **rsyslog(severitytext property)**、一个 Python 日志记录模块等。

以下值来自 **syslog.h**，并在前面加上它们的 **等效数字**：

- **0 = emerg**，系统不可用。
- **1 = alert**，必须立即执行操作。
- **2 = crit**，关键条件。
- **3 = err**，错误条件。
- **4 = warn**，警告条件。
- **5 = notice**，正常但有严重情况。
- **6 = info**，信息。
- **7 = debug**，debug 级信息。

以下两个值不是 **syslog.h** 的一部分，但被广泛使用：

- **8 = trace**，trace 级的信息，它比 **debug** 信息更详细。
- **9 = unknown**，日志记录系统获得一个无法被识别的值。

在前面的列表中，将其他日志记录系统的日志级别或优先级映射到其最接近的匹配项。例如，在 [python logging](#) 中，您可以使用 **CRITICAL** 匹配 **crit**，使用 **ERROR** 匹配 **err**，以此类推。

数据类型	关键字
示例值	<b>info</b>

## 第 22 章 PID

日志记录实体的进程 ID（若有）。

数据类型	关键字
------	-----

## 第 23 章 SERVICE

与日志记录实体（若有）关联的服务的名称。例如，syslog 的 **APP-NAME** 和 rsyslog 的 **programname** 属性映射到 service 字段。

数据类型	关键字
------	-----

## 第 24 章 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。有效负载可以是带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。

数据类型	text
------	------

## 第 25 章 FILE

收集器从中读取此日志条目的日志文件路径。通常，这是集群节点的 `/var/log` 文件系统中的路径。

数据类型	text
------	------

## 第 26 章 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）。

数据类型	long
------	------

## 第 27 章 KUBERNETES

特定于 Kubernetes 元数据的命名空间

数据类型	group
------	-------

### 27.1. KUBERNETES.POD\_NAME

pod 的名称

数据类型	关键字
------	-----

### 27.2. KUBERNETES.POD\_ID

pod 的 Kubernetes ID

数据类型	关键字
------	-----

### 27.3. KUBERNETES.NAMESPACE\_NAME

Kubernetes 中命名空间的名称

数据类型	关键字
------	-----

### 27.4. KUBERNETES.NAMESPACE\_ID

Kubernetes 中命名空间的 ID

数据类型	关键字
------	-----

### 27.5. KUBERNETES.HOST

Kubernetes 节点名称

数据类型	关键字
------	-----

### 27.6. KUBERNETES.CONTAINER\_NAME

Kubernetes 中容器的名称

数据类型	关键字
------	-----

## 27.7. KUBERNETES.ANNOTATIONS

与 Kubernetes 对象关联的注解

数据类型	group
------	-------

## 27.8. KUBERNETES.LABELS

原始 Kubernetes Pod 上存在的标签

数据类型	group
------	-------

## 27.9. KUBERNETES.EVENT

从 Kubernetes 主机 API 获取的 Kubernetes 事件。此事件描述大致跟随 [Event v1 core](#) 中的类型事件。

数据类型	group
------	-------

### 27.9.1. kubernetes.event.verb

事件类型，**ADDED**、**MODIFIED** 或 **DELETED**

数据类型	关键字
示例值	<b>ADDED</b>

### 27.9.2. kubernetes.event.metadata

与事件创建位置和时间相关的信息

数据类型	group
------	-------

#### 27.9.2.1. kubernetes.event.metadata.name

触发事件创建的对象名称

数据类型	关键字
示例值	<b>java-mainclass-1.14d888a4cfc24890</b>

#### 27.9.2.2. kubernetes.event.metadata.namespace

最初发生事件的命名空间的名称。请注意，它与 `kubernetes.namespace_name` 不同，后者是部署 `eventrouter` 应用程序的命名空间。

数据类型	关键字
示例值	<b>default</b>

### 27.9.2.3. kubernetes.event.metadata.selfLink

到事件的链接

数据类型	关键字
示例值	<b>/api/v1/namespaces/javaj/events/java-mainclass-1.14d888a4cfc24890</b>

### 27.9.2.4. kubernetes.event.metadata.uid

事件的唯一 ID

数据类型	关键字
示例值	<b>d828ac69-7b58-11e7-9cf5-5254002f560c</b>

### 27.9.2.5. kubernetes.event.metadata.resourceVersion

标识服务器内部版本的事件的字符串。客户端可以使用此字符串来确定对象何时更改。

数据类型	整数
示例值	<b>311987</b>

## 27.9.3. kubernetes.event.involvedObject

事件所针对的对象。

数据类型	group
------	-------

### 27.9.3.1. kubernetes.event.involvedObject.kind

对象的类型

数据类型	关键字
示例值	<b>ReplicationController</b>

### 27.9.3.2. kubernetes.event.involvedObject.namespace

相关对象的命名空间名称。请注意，它可能与 `kubernetes.namespace_name` 不同，后者是部署 `eventrouter` 应用程序的命名空间。

数据类型	关键字
示例值	<b>default</b>

### 27.9.3.3. kubernetes.event.involvedObject.name

触发事件的对象名称

数据类型	关键字
示例值	<b>java-mainclass-1</b>

### 27.9.3.4. kubernetes.event.involvedObject.uid

对象的唯一 ID

数据类型	关键字
示例值	<b>e6bff941-76a8-11e7-8193-5254002f560c</b>

### 27.9.3.5. kubernetes.event.involvedObject.apiVersion

kubernetes master API 的版本

数据类型	关键字
示例值	<b>v1</b>

### 27.9.3.6. kubernetes.event.involvedObject.resourceVersion

标识触发该事件的 pod 的内部版本的字符串。客户端可以使用此字符串来确定对象何时更改。

数据类型	关键字
示例值	<b>308882</b>

## 27.9.4. kubernetes.event.reason

简短的机器可读字符串，给出生成此事件的原因

数据类型	关键字
------	-----

示例值	<b>SuccessfulCreate</b>
-----	-------------------------

### 27.9.5. kubernetes.event.source\_component

报告此事件的组件

数据类型	关键字
示例值	<b>replication-controller</b>

### 27.9.6. kubernetes.event.firstTimestamp

事件首次记录的时间

数据类型	date
示例值	<b>2017-08-07 10:11:57.000000000 Z</b>

### 27.9.7. kubernetes.event.count

发生此事件的次数

数据类型	整数
示例值	<b>1</b>

### 27.9.8. kubernetes.event.type

事件类型，**Normal** 或 **Warning**。以后可能会添加新类型。

数据类型	关键字
示例值	<b>Normal</b>

## 第 28 章 OPENSIFT

openshift-logging 特定元数据的命名空间

数据类型	group
------	-------

### 28.1. OPENSIFT.LABELS

由 Cluster Log Forwarder 配置添加的标签

数据类型	group
------	-------