OpenShift Container Platform 4.16

日志记录

在 OpenShift Container Platform 中配置和使用日志

Last Updated: 2024-07-02

OpenShift Container Platform 4.16 日志记录

在 OpenShift Container Platform 中配置和使用日志

法律通告

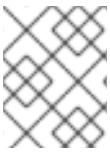
摘要

使用日志记录来收集、视觉化、转发和存储日志数据来排除问题，识别性能瓶颈，并检测 OpenShift Container Platform 中的安全威胁。

目录

第 1 章 发行注记

1.1. LOGGING 5.9



注意

日志记录作为一个可安装的组件提供，它有一个不同于 OpenShift Container Platform 的发布周期。Red Hat OpenShift Container Platform 生命周期政策 概述了发行版本兼容性。

注意

stable 频道只为日志记录的最新版本提供更新。要继续获得之前版本的更新，您必须将订阅频道改为 **stable-x.y**，其中 **x.y** 代表您安装的日志记录的主版本和次版本。例如，**stable-5.7**。

1.1.1. Logging 5.9.3

此发行版本包括 OpenShift Logging 程序错误修复 5.9.3

1.1.1.1. 程序错误修复

- 在此次更新之前，配置 **LokiStack** 时重启 Ingesters 有一个延迟，因为 Loki Operator 会将 write-ahead log **replay_memory_ceiling** 设置为 **1x.demo** 大小。在这个版本中，用于 **replay_memory_ceiling** 的最小值已增加，以避免延迟。(LOG-5614)

- 在此次更新之前，无法监控 Vector 收集器输出缓冲状态。在这个版本中，可以监控和警报 Vector 收集器输出缓冲大小，这提高了可观察性功能，并帮助系统以最佳方式运行。(LOG-5586)

1.1.1.2. CVE

- CVE-2024-2961
- CVE-2024-28182
- CVE-2024-33599
- CVE-2024-33600
- CVE-2024-33601
- CVE-2024-33602

1.1.2. 日志记录 5.9.2

此发行版本包括 OpenShift Logging 程序错误修复 5.9.2

1.1.2.1. 程序错误修复

- 在此次更新之前，因为 **ClusterLogForwarder** CR 中的配置不正确，对 Logging Operator 的更改会导致错误。因此，升级到日志记录会删除 daemonset 收集器。在这个版本中，日志记录 Operator 会重新创建收集器 daemonset，除非出现 **Not authorized to collect** 错误。(LOG-4910)

- 在此次更新之前，因为 Vector 日志收集器中的配置不正确，在有些情况下，轮转的基础架构日志文件会被发送到应用程序索引。在这个版本中，Vector 日志收集器配置可避免收集任何轮转的基础架构日志文件。(LOG-5156)

- 在此次更新之前，日志记录 Operator 不会监控 **grafana-dashboard-cluster-logging** 配置映射的更改。在这个版本中，日志记录 Operator 会监控 **ConfigMap** 对象中的更改，确保系统保持同步并有效地响应配置映射修改。(LOG-5308)

- 在此次更新之前，日志记录 Operator 的指标集合代码中的问题会导致它报告过时的遥测指标。在这个版本中，日志记录 Operator 不会报告过时的遥测指标。(LOG-5426)

- 在此更改前，Fluentd **out_http** 插件会忽略 **no_proxy** 环境变量。在这个版本中，Fluentd 对 ruby 的 **HTTP#start** 方法进行了修改以接受 **no_proxy** 环境变量。(LOG-5466)

1.1.2.2. CVE

- CVE-2022-48554
- CVE-2023-2975
- CVE-2023-3446
- CVE-2023-3817
- CVE-2023-5678
- CVE-2023-6129
- CVE-2023-6237
- CVE-2023-7008
- CVE-2023-45288
- CVE-2024-0727
- CVE-2024-22365
- CVE-2024-25062
- CVE-2024-28834
- CVE-2024-28835

1.1.3. 日志记录 5.9.1

此发行版本包括 OpenShift Logging 程序错误修复 5.9.1

1.1.3.1. 功能增强

- 在此次更新之前，Loki Operator 将 Loki 配置为使用 Amazon Simple Storage Service (S3) 的基于路径的风格访问，它已被弃用。在这个版本中，Loki Operator 默认为 virtual-host 样式，而无需更改其配置。(LOG-5401)

- 在此次更新之前，Loki Operator 不会验证存储 secret 中使用的 Amazon Simple Storage Service (S3) 端点。在这个版本中，验证过程可确保 S3 端点是一个有效的 S3 URL，**LokiStack** 状态更新来指示任何无效的 URL。(LOG-5395)

1.1.3.2. 程序错误修复

- 在此次更新之前，LogQL 解析中的错误会从查询中排除一些行过滤器。在这个版本中，解析会包括所有行过滤器，同时保持原始查询保持不变。(LOG-5268)
- 在此次更新之前，没有定义的 **pruneFilterSpec** 的 prune 过滤器会导致 segfault。在这个版本中，如果修剪过滤器没有定义的 **puneFilterSpec**，会出现验证错误。(LOG-5322)
- 在此次更新之前，没有定义的 **dropTestsSpec** 的 drop 过滤器会导致 segfault。在这个版本中，如果修剪过滤器没有定义的 **puneFilterSpec**，会出现验证错误。(LOG-5323)
- 在此次更新之前，Loki Operator 不会验证存储 secret 中使用的 Amazon Simple Storage Service (S3) 端点 URL 格式。在这个版本中，S3 端点 URL 会经过验证步骤，它反映了 **LokiStack** 的状态。(LOG-5397)
- 在此次更新之前，审计日志记录中格式化的时间戳字段会导致 Red Hat OpenShift Logging Operator 日志中出现 **WARN** 信息。在这个版本中，重新映射转换可确保正确格式化 timestamp 字段。(LOG-4672)
- 在此次更新之前，当验证 **ClusterLogForwarder** 资源名称和命名空间没有对应于正确的错误时，错误消息会抛出。在这个版本中，系统会检查同一命名空间中是否存在具有相同名称的 **ClusterLogForwarder** 资源。如果没有，它对应于正确的错误。(LOG-5062)
- 在此次更新之前，输出配置的验证功能需要一个 TLS URL，即使 Amazon CloudWatch 或 Google Cloud Logging 等服务，在设计不需要 URL。在这个版本中，在没有 URL 的服务的验证逻辑有所改进，错误消息会更为说明。(LOG-5307)
- 在此次更新之前，定义基础架构输入类型不会从集合中排除日志工作负载。在这个版本中，集合排除日志记录服务以避免反馈循环。(LOG-5309)

1.1.3.3. CVE

没有 CVE。

1.1.4. Logging 5.9.0

此发行版本包括 OpenShift Logging 程序错误修复 5.9.0

1.1.4.1. 删除通知

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。来自之前日志记录版本的 OpenShift Elasticsearch Operator 实例被支持，直到日志记录版本的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

1.1.4.2. 弃用通知

- 在 Logging 5.9，Fluentd，Elasticsearch、Fluentd 和 Kibana 已被弃用，计划在 Logging 6.0 中删除，该 6.0 应该与以后的 OpenShift Container Platform 版本一起提供。红帽将在当前发行生命周期中对这些组件提供关键及以上的 CVE 程序错误修复和支持，但这些组件将不再获得功能增

强。由 Red Hat OpenShift Logging Operator 和 LokiStack 提供的基于 Vector 的收集器和 Loki Operator 提供的 LokiStack 是日志集合和存储的首选 Operator。我们鼓励所有用户使用 Vector 和 Loki 日志堆栈，因为这个组合会持续进一步进行改进。

- 在 Logging 5.9 中，Splunk 输出类型的 **Fields** 选项没有实现，并现已弃用。它将在以后的发行版本中被删除。

1.1.4.3. 功能增强

1.1.4.3.1. 日志集合

- 此功能增强增加了使用工作负载的元数据根据其内容 **drop** 或 **prune** 日志的功能。另外，它允许收集基础架构日志（如日志或容器日志）和审计日志（如 **kube api** 或 **ovn** 日志）仅收集单个源。(LOG-2155)

- 此增强引入了一个新的远程日志接收器 (syslog 接收器)。您可以将其配置为通过网络公开端口，允许外部系统使用兼容工具（如 rsyslog）发送 syslog 日志。(LOG-3527)

- 在这个版本中，**ClusterLogForwarder** API 支持转发到 Azure Monitor 日志，为用户提供更好的监控功能。此功能可帮助用户维护最佳的系统性能，并简化 Azure Monitor 中的日志分析过程，从而加快问题解决并提高操作效率。(LOG-4605)

- 此功能增强通过将收集器部署为带有两个副本的部署来提高收集器资源利用率。当 **ClusterLogForwarder** 自定义资源 (CR) 中定义的唯一输入源是接收器输入而不是在所有节点上使用守护进程集时，会出现这种情况。此外，以这种方式部署的收集器不会挂载主机文件系统。要使用此增强，您需要使用 **logging.openshift.io/dev-preview-enable-collector-as-deployment** 注解来注解 **ClusterLogForwarder** CR。(LOG-4779)

- 此功能增强引入了在所有支持的输出中自定义租户配置的功能，以逻辑方式促进日志记录的组织。但是，它不允许自定义租户配置来记录受管存储。(LOG-4843)

- 在这个版本中，**ClusterLogForwarder** CR 使用一个或多个基础架构命名空间（如 **default**、**openshift*** 或 **kube***）指定应用程序输入，现在需要一个具有 **collect-infrastructure-logs** 角色的服务帐户。(LOG-4943)

- 此功能增强引入了调整一些输出设置（如压缩、重试持续时间和最大有效负载）的功能，以匹配接收器的特性。此外，此功能还包括一种交付模式，允许管理员在吞吐量和日志持久性之间进行选择。例如，**AtLeastOnce** 选项配置收集日志的最小磁盘缓冲区，以便收集器重启后可以发送这些日志。(LOG-5026)

- 此功能增强添加了三个新的 Prometheus 警报，警告用户有关 Elasticsearch、Fluentd 和 Kibana 弃用的信息。(LOG-5055)

1.1.4.3.2. 日志存储

- LokiStack 中的这个增强通过使用新的 V13 对象存储格式并默认启用自动流分片功能，改进了对 OTEL 的支持。这也可让收集器为将来的改进和配置准备。(LOG-4538)

- 此功能增强引进了对使用 Azure 和 AWS 日志存储的短期令牌工作负载身份联合的支持，启用了 STS 的 OpenShift Container Platform 4.14 及更新的版本。本地存储需要在 LokiStack CR 的 **spec.storage.secret** 下添加 **CredentialMode: static** 注解。(LOG-4540)

- 在这个版本中，Azure 存储 secret 的验证被扩展，为某些错误状况提供早期警告。(LOG-4571)

- 在这个版本中，Loki 添加了对 GCP 工作负载身份联邦机制的上游和下游支持。这允许对对应对象存储服务进行身份验证和授权访问。(LOG-4754)

1.1.4.4. 程序错误修复

- 在此次更新之前，日志记录 must-gather 无法收集启用了 FIPS 的集群中的任何日志。在这个版本中，**cluster-logging-rhel9-operator** 中提供了一个新的 **oc** 客户端，must-gather 可以在 FIPS 集群中正常工作。(LOG-4403)
- 在此次更新之前，LokiStack 规则器 pod 无法将 IPv6 pod IP 格式化为用于跨 pod 通信的 HTTP URL。此问题导致通过与 Prometheus 兼容的 API 查询规则和警报失败。在这个版本中，LokiStack 规则器 pod 将 IPv6 pod IP 封装在方括号中，从而解决了这个问题。现在，通过与 Prometheus 兼容的 API 查询规则和警报的工作方式就像在 IPv4 环境中一样。(LOG-4709)
- 在这个版本中，日志记录 must-gather 的 YAML 内容在一行中导出，使其不可读取。在这个版本中，YAML 空格会被保留，确保该文件被正确格式化。(LOG-4792)
- 在此次更新之前，当启用 **ClusterLogForwarder** CR 时，当 **ClusterLogging.Spec.Collection** 为 nil 时，Red Hat OpenShift Logging Operator 可能会进入 nil pointer 异常。在这个版本中，这个问题已在 Red Hat OpenShift Logging Operator 中解决。(LOG-5006)
- 在此次更新之前，在特定的基础情形中，替换 **ClusterLogForwarder** CR status 字段会导致 **resourceVersion** 在 **Status** 条件中更改时间戳而持续更新。这个条件会导致一个无限的协调循环。在这个版本中，所有状态条件都会同步，以便在条件保持不变时时间戳保持不变。(LOG-5007)
- 在此次更新之前，内部缓冲行为需要 **drop_newest** 来解决收集器高内存消耗，从而导致大量日志丢失。在这个版本中，行为恢复为使用收集器默认值。(LOG-5123)
- 在此次更新之前，**openshift-operators-redhat** 命名空间中的 Loki Operator **ServiceMonitor** 使用静态令牌和 CA 文件进行身份验证，从而导致 **ServiceMonitor** 配置上的 User Workload Monitoring spec 中 Prometheus Operator 出现错误。在这个版本中，**openshift-operators-redhat** 命名空间中的 Loki Operator **ServiceMonitor** 现在通过 **LocalReference** 对象引用服务帐户令牌 secret。这种方法允许 Prometheus Operator 中的 User Workload Monitoring spec 成功处理 Loki Operator **ServiceMonitor**，使 Prometheus 能够提取 Loki Operator 指标。(LOG-5165)
- 在此次更新之前，Loki Operator **ServiceMonitor** 的配置可能与许多 Kubernetes 服务匹配，从而导致 Loki Operator 指标被多次收集。在这个版本中，**ServiceMonitor** 的配置只与专用指标服务匹配。(LOG-5212)

1.1.4.5. 已知问题

无。

1.1.4.6. CVE

- CVE-2023-5363
- CVE-2023-5981
- CVE-2023-46218
- CVE-2024-0553
- CVE-2023-0567

1.2. LOGGING 5.8

注意

日志记录作为一个可安装的组件提供，它有一个不同于 OpenShift Container Platform 的发布周期。Red Hat OpenShift Container Platform 生命周期政策 概述了发行版本兼容性。

注意

stable 频道只为日志记录的最新版本提供更新。要继续获得之前版本的更新，您必须将订阅频道改为 **stable-x.y**，其中 **x.y** 代表您安装的日志记录的主版本和次版本。例如，**stable-5.7**。

1.2.1. Logging 5.8.4

此发行版本包括 OpenShift Logging 程序错误修复 5.8.4 。

1.2.1.1. 程序错误修复

- 在此次更新之前，开发人员控制台日志没有考虑当前命名空间，从而导致在没有集群范围日志访问权限的情况下对用户进行查询。在这个版本中，所有支持的 OCP 版本都确保命名空间包含正确。(LOG-4905)

- 在此次更新之前，Cluster Logging Operator 仅在默认日志输出为 LokiStack 时部署支持 LokiStack 部署的 **ClusterRole**。在这个版本中，角色被分成两个组：read 和 write。写入角色根据默认日志存储的设置部署，就像以前使用的所有角色一样。read 角色根据日志记录控制台插件是否活跃部署。(LOG-4987)

- 在此次更新之前，多个定义相同输入接收器名称的 **ClusterLogForwarders** 因在一个服务上更改 **ownerReferences** 而被无限协调。在这个版本中，每个接收器输入都有自己的服务，其约定为 **<CLF.Name>-<input.Name>**。(LOG-5009)

- 在此次更新之前，当在没有 secret 的情况下将日志转发到 cloudwatch 时，**ClusterLogForwarder** 不会报告错误。在这个版本中，当在没有 secret 时将日志转发到 cloudwatch 时会出现以下错误：**secret must be provided for cloudwatch output**。(LOG-5021)

- 在此次更新之前，**log_forwarder_input_info** 包括 **application**, **infrastructure**, 和 **audit** 输入指标点。在这个版本中，**http** 也添加为指标点。(LOG-5043)

1.2.1.2. CVE

- CVE-2021-35937
- CVE-2021-35938
- CVE-2021-35939
- CVE-2022-3545
- CVE-2022-24963
- CVE-2022-36402
- CVE-2022-41858
- CVE-2023-2166

- CVE-2023-2176
- CVE-2023-3777
- CVE-2023-3812
- CVE-2023-4015
- CVE-2023-4622
- CVE-2023-4623
- CVE-2023-5178
- CVE-2023-5363
- CVE-2023-5388
- CVE-2023-5633
- CVE-2023-6679
- CVE-2023-7104
- CVE-2023-27043
- CVE-2023-38409
- CVE-2023-40283
- CVE-2023-42753
- CVE-2023-43804
- CVE-2023-45803
- CVE-2023-46813
- CVE-2024-20918
- CVE-2024-20919
- CVE-2024-20921
- CVE-2024-20926
- CVE-2024-20945
- CVE-2024-20952

1.2.2. Logging 5.8.3

此发行版本包括 Logging 程序错误修复 5.8.3 和 Logging 安全修复 5.8.3

1.2.2.1. 程序错误修复

- 在此次更新之前，当配置为读取自定义 S3 证书颁发机构时，Loki Operator 不会在 ConfigMap 的名称或内容改变时自动更新配置。在这个版本中，Loki Operator 会监视 ConfigMap 的更改，并自动更新生成的配置。(LOG-4969)

- 在此次更新之前，在没有有效 URL 的情况下配置的 Loki 输出会导致收集器 Pod 崩溃。在这个版本中，输出会根据 URL 验证进行相应的处理，从而解决了这个问题。(LOG-4822)

- 在此次更新之前，Cluster Logging Operator 会为没有指定 secret 来使用服务帐户 bearer 令牌的输出生成收集器配置字段。在这个版本中，输出不需要身份验证，从而解决了这个问题。(LOG-4962)

- 在此次更新之前，输出的 **tls.insecureSkipVerify** 字段没有设置为 **true**，且没有定义 secret。在这个版本中，不再需要一个 secret 来设置这个值。(LOG-4963)

- 在此次更新之前，输出配置允许使用 TLS 身份验证的不安全(HTTP) URL 的组合。在这个版本中，为 TLS 身份验证配置的输出需要一个安全(HTTPS) URL。(LOG-4893)

1.2.2.2. CVE

- CVE-2021-35937
- CVE-2021-35938
- CVE-2021-35939
- CVE-2023-7104
- CVE-2023-27043
- CVE-2023-48795
- CVE-2023-51385
- CVE-2024-0553

1.2.3. Logging 5.8.2

此发行版本包括 OpenShift Logging 程序错误修复 5.8.2。

1.2.3.1. 程序错误修复

- 在此次更新之前，LokiStack 规则器 pod 不会将 IPv6 pod IP 格式化为用于跨 pod 通信的 HTTP URL，从而导致通过 Prometheus 兼容的 API 查询规则和警报失败。在这个版本中，LokiStack 规则器 pod 将 IPv6 pod IP 封装在方括号中，从而解决了这个问题。(LOG-4890)

- 在此次更新之前，开发人员控制台日志没有考虑当前命名空间，从而导致在没有集群范围日志访问权限的情况下对用户进行查询。在这个版本中，命名空间包含已被修正，从而解决了这个问题。(LOG-4947)

- 在此次更新之前，OpenShift Container Platform Web 控制台的日志记录视图插件不允许自定义节点放置和容限。在这个版本中，定义自定义节点放置和容限被添加到 OpenShift Container Platform Web 控制台的日志记录视图插件中。(LOG-4912)

1.2.3.2. CVE

- CVE-2022-44638

- CVE-2023-1192
- CVE-2023-5345
- CVE-2023-20569
- CVE-2023-26159
- CVE-2023-39615
- CVE-2023-45871

1.2.4. Logging 5.8.1

此发行版本包括 OpenShift Logging 程序错误修复 5.8.1 和 OpenShift Logging 程序错误修复 5.8.1 Kibana。

1.2.4.1. 功能增强

1.2.4.1.1. 日志集合

- 在这个版本中，在将 Vector 配置为收集器时，您可以在 Red Hat OpenShift Logging Operator 中添加逻辑，以使用 secret 中指定的令牌来代替与服务帐户关联的令牌。(LOG-4780)
- 在这个版本中，**BoltDB Shipper** Loki 仪表板被重命名为 **Index** 仪表板。(LOG-4828)

1.2.4.2. 程序错误修复

- 在此次更新之前，**ClusterLogForwarder** 在启用 JSON 日志解析后会创建空索引，即使没有满足滚动条件。在这个版本中，**ClusterLogForwarder** 会在 **write-index** 为空时跳过滚动。(LOG-4452)
- 在此次更新之前，Vector 会错误地设置了**默认**日志级别。在这个版本中，通过改进正则表达式（**regexp**）进行日志级别检测来设置正确的日志级别。(LOG-4480)
- 在此次更新之前，在创建索引模式的过程中，每个日志输出的初始索引中缺少默认别名。因此，Kibana 用户无法使用 OpenShift Elasticsearch Operator 创建索引模式。在这个版本中，在 OpenShift Elasticsearch Operator 中添加缺少的别名，从而解决了这个问题。Kibana 用户现在可以创建包含 **{app,infra,audit}-000001** 索引的索引模式。(LOG-4683)
- 在此次更新之前，Fluentd 收集器 Pod 处于 **CrashLoopBackOff** 状态，因为 IPv6 集群上的 Prometheus 服务器绑定。在这个版本中，收集器可以在 IPv6 集群中正常工作。(LOG-4706)
- 在此次更新之前，Red Hat OpenShift Logging Operator 会在 **ClusterLogForwarder** 中有更改时进行大量协调。在这个版本中，Red Hat OpenShift Logging Operator 会忽略触发协调的收集器 daemonset 的状态更改。(LOG-4741)
- 在此次更新之前，Vector 日志收集器 pod 在 IBM Power 机器上处于 **CrashLoopBackOff** 状态。在这个版本中，Vector 日志收集器 Pod 在 IBM Power 架构机器上成功启动。(LOG-4768)
- 在此次更新之前，使用旧的转发器转发到内部 LokiStack 会导致使用 Fluentd 收集器 Pod 生成 SSL 证书错误。在这个版本中，日志收集器服务帐户默认使用关联的令牌和 **ca.crt** 进行身份验证。(LOG-4791)

- 在此次更新之前，使用旧的转发器转发到内部 LokiStack 会导致使用 Vector 收集器 Pod 生成 SSL 证书错误。在这个版本中，日志收集器服务帐户默认使用关联的令牌和 **ca.crt** 进行身份验证。(LOG-4852)

- 在此次更新之前，在为占位符评估一个或多个主机后，不会正确解析 IPv6 地址。在这个版本中，IPv6 地址会被正确解析。(LOG-4811)

- 在此次更新之前，需要创建一个 **ClusterRoleBinding** 来为 HTTP 接收器输入收集审计权限。在这个版本中，不需要创建 **ClusterRoleBinding**，因为端点已经依赖于集群证书颁发机构。(LOG-4815)

- 在此次更新之前，Loki Operator 不会将自定义 CA 捆绑包挂载到规则 pod。因此，在评估警报或记录规则的过程中，对象存储访问会失败。在这个版本中，Loki Operator 将自定义 CA 捆绑包挂载到所有规则 pod。规则器 pod 可以从对象存储下载日志，以评估警报或记录规则。(LOG-4836)

- 在此次更新之前，在删除 **ClusterLogForwarder** 中的 **inputs.receiver** 部分时，HTTP 输入服务及其关联的 secret 不会被删除。在这个版本中，如果需要，HTTP 输入资源会被删除。(LOG-4612)

- 在此次更新之前，**ClusterLogForwarder** 指示状态中的验证错误，但输出和管道状态无法准确反映特定的问题。在这个版本中，管道状态会在错误的输出、输入或过滤器时正确显示验证失败的原因。(LOG-4821)

- 在此次更新之前，更改使用控制的 **LogQL** 查询（如时间范围或严重性）更改了标签 matcher operator 定义它，就像正则表达式一样。在这个版本中，正则表达式运算符在更新查询时保持不变。(LOG-4841)

1.2.4.3. CVE

- CVE-2007-4559
- CVE-2021-3468
- CVE-2021-3502
- CVE-2021-3826
- CVE-2021-43618
- CVE-2022-3523
- CVE-2022-3565
- CVE-2022-3594
- CVE-2022-4285
- CVE-2022-38457
- CVE-2022-40133
- CVE-2022-40982
- CVE-2022-41862
- CVE-2022-42895

- CVE-2023-0597
- CVE-2023-1073
- CVE-2023-1074
- CVE-2023-1075
- CVE-2023-1076
- CVE-2023-1079
- CVE-2023-1206
- CVE-2023-1249
- CVE-2023-1252
- CVE-2023-1652
- CVE-2023-1855
- CVE-2023-1981
- CVE-2023-1989
- CVE-2023-2731
- CVE-2023-3138
- CVE-2023-3141
- CVE-2023-3161
- CVE-2023-3212
- CVE-2023-3268
- CVE-2023-3316
- CVE-2023-3358
- CVE-2023-3576
- CVE-2023-3609
- CVE-2023-3772
- CVE-2023-3773
- CVE-2023-4016
- CVE-2023-4128
- CVE-2023-4155
- CVE-2023-4194

- CVE-2023-4206
- CVE-2023-4207
- CVE-2023-4208
- CVE-2023-4273
- CVE-2023-4641
- CVE-2023-22745
- CVE-2023-26545
- CVE-2023-26965
- CVE-2023-26966
- CVE-2023-27522
- CVE-2023-29491
- CVE-2023-29499
- CVE-2023-30456
- CVE-2023-31486
- CVE-2023-32324
- CVE-2023-32573
- CVE-2023-32611
- CVE-2023-32665
- CVE-2023-33203
- CVE-2023-33285
- CVE-2023-33951
- CVE-2023-33952
- CVE-2023-34241
- CVE-2023-34410
- CVE-2023-35825
- CVE-2023-36054
- CVE-2023-37369
- CVE-2023-38197
- CVE-2023-38545

- CVE-2023-38546
- CVE-2023-39191
- CVE-2023-39975
- CVE-2023-44487

1.2.5. Logging 5.8.0

此发行版本包括 OpenShift Logging 程序错误修复 5.8.0 和 OpenShift Logging 程序错误修复 5.8.0 Kibana。

1.2.5.1. 弃用通知

在 Logging 5.8 中，Elasticsearch、Fluentd 和 Kibana 已被弃用，计划在 Logging 6.0 中删除，该 6.0 应该与以后的 OpenShift Container Platform 版本一起提供。红帽将在当前发行生命周期中对这些组件提供关键及以上的 CVE 程序错误修复和支持，但这些组件将不再获得功能增强。由 Red Hat OpenShift Logging Operator 和 LokiStack 提供的基于 Vector 的收集器和 Loki Operator 提供的 LokiStack 是日志集合和存储的首选 Operator。我们鼓励所有用户使用 Vector 和 Loki 日志堆栈，因为这个组合会持续进一步进行改进。

1.2.5.2. 功能增强

1.2.5.2.1. 日志集合

- 在这个版本中，LogFileMetricExporter 不再默认和收集器一起部署。您需要手动创建一个 **LogFileMetricExporter** 自定义资源 (CR)，从运行容器生成的日志中生成指标。如果没有创建 **LogFileMetricExporter** CR，您可能会在 OpenShift Container Platform Web 控制台仪表板中看到 **Produced Logs** 的 **No datapoints found** 信息。(LOG-3819)
- 在这个版本中，您可以在任何命名空间中部署多个、隔离和 RBAC 保护的 **ClusterLogForwarder** 自定义资源(CR)实例。这允许独立组将所需的日志转发到任何目的地，同时将其配置与其他收集器部署隔离。(LOG-1343)

重要

要在 **openshift-logging** 命名空间以外的额外命名空间中支持多集群日志转发功能，您必须更新 Red Hat OpenShift Logging Operator 以监视所有命名空间。在新的 Red Hat OpenShift Logging Operator 版本 5.8 版本中默认支持此功能。

- 在这个版本中，您可以使用流控制或速率限制机制来限制可以通过丢弃超额日志记录来收集或转发日志数据的卷。输入限制防止性能不佳的容器过载 Logging，输出限制则限制在提供给给定数据存储的日志的速度上造成不佳。(LOG-884)
- 在这个版本中，您可以将日志收集器配置为查找 HTTP 连接，并接收日志作为 HTTP 服务器，也称为 Webhook。(LOG-4562)
- 在这个版本中，您可以配置审计策略来控制日志收集器转发哪些 Kubernetes 和 OpenShift API 服务器事件。(LOG-3982)

1.2.5.2.2. 日志存储

- 在这个版本中，Loki 管理员可以更精细地控制谁可以通过基于命名空间授予对日志的访问权限。(LOG-3841)
- 在这个版本中，Loki Operator 在 LokiStack 部署中引入了 **PodDisruptionBudget** 配置，通过保持 ingestion 和查询路径可用来确保 OpenShift Container Platform 集群重启期间正常操作。(LOG-3839)
- 在这个版本中，现有 LokiStack 安装的可靠性通过应用一组默认的 Affinity 和 Anti-Affinity 策略来无缝改进。(LOG-3840)
- 在这个版本中，您可以作为 LokiStack 中的管理员管理区感知数据复制，以便在区失败时增强可靠性。(LOG-3266)
- 在这个版本中，增加了一个新的受支持的小规模的 LokiStack 大小（1x.extra-small），它适用于托管少量工作负载并有较小的 ingestion 卷（最多为 100GB/天）的 OpenShift Container Platform 集群。(LOG-4329)
- 在这个版本中，Loki 管理员可以访问官方 Loki 仪表板，以检查存储性能和每个组件的健康状态。(LOG-4327)

1.2.5.2.3. 日志控制台

- 在这个版本中，当 Elasticsearch 是默认的日志存储时，您可以启用日志记录控制台插件。(LOG-3856)
- 在这个版本中，OpenShift Container Platform 应用程序所有者可以为 OpenShift Container Platform 版本 4.14 及之后的版本在 OpenShift Container Platform Web 控制台 **Developer** 视角中接收基于应用程序日志的警报的通知。(LOG-3548)

1.2.5.3. 已知问题

- 目前，在升级到 Red Hat OpenShift Logging Operator 版本 5.8 后，Splunk 日志转发可能无法正常工作。这个问题是由从 OpenSSL 版本 1.1.1 转换到 3.0.7 版本造成的。在较新的 OpenSSL 版本中，有一个默认行为更改，如果没有公开 RFC 5746 扩展，则拒绝到 TLS 1.2 端点的连接。作为临时解决方案，请在 Splunk HEC (HTTP Event Collector) 端点前对 TLS 终止负载均衡器启用 TLS 1.3 支持。Splunk 是一个第三方系统，它应该从 Splunk 端配置。
- 目前，在处理 HTTP/2 协议中的多路流中存在一个缺陷，您可以在其中重复向新的多路流发出请求，并立即发送 **RST_STREAM** 框架来取消它。这会为服务器设置和缩减流造成额外的工作，从而导致因为服务器资源消耗而拒绝服务。当前没有解决此问题的方法。(LOG-4609)
- 目前，当使用 FluentD 作为收集器时，收集器 Pod 无法在 OpenShift Container Platform IPv6-enabled 集群中启动。pod 日志会生成 **fluentd pod [error]: unexpected error error_class=SocketError error="getaddrinfo: Name or service not known** 错误。当前没有解决此问题的方法。(LOG-4706)
- 目前，启用了 IPv6 的集群上没有日志警报。当前没有解决此问题的方法。(LOG-4709)
- 目前，**must-gather** 无法在启用了 FIPS 的集群中收集任何日志，因为 **cluster-logging-rhel9-operator** 中没有所需的 OpenSSL 库。当前没有解决此问题的方法。(LOG-4403)
- 目前，当在启用了 FIPS 的集群上部署 logging 版本 5.8 时，收集器 Pod 无法启动，并处于 **CrashLoopBackOff** 状态，同时将 FluentD 用作收集器。当前没有解决此问题的方法。(LOG-3933)

1.2.5.4. CVE

- CVE-2023-40217

1.3. LOGGING 5.7

注意

日志记录作为一个可安装的组件提供，它有一个不同于 OpenShift Container Platform 的发布周期。Red Hat OpenShift Container Platform 生命周期政策 概述了发行版本兼容性。

注意

stable 频道只为日志记录的最新版本提供更新。要继续获得之前版本的更新，您必须将订阅频道改为 **stable-x.y**，其中 **x.y** 代表您安装的日志记录的主版本和次版本。例如，**stable-5.7**。

1.3.1. Logging 5.7.8

此发行版本包括 OpenShift Logging 程序错误修复 5.7.8。

1.3.1.1. 程序错误修复

- 在此次更新之前，当与 **ClusterLogForwarder** 自定义资源(CR)中的 **outputRefs** 和 **inputRefs** 参数使用相同的名称时，则会有潜在的冲突。因此，收集器 Pod 在 **CrashLoopBackOff** 状态中输入。在这个版本中，输出标签包含 **OUTPUT_** 前缀，以确保输出标签和管道名称之间的区别。(LOG-4383)

- 在此次更新之前，在配置 JSON 日志解析器时，如果您没有为 Cluster Logging Operator 设置 **structuredTypeKey** 或 **structuredTypeName** 参数，则不会显示有关无效配置的警报。在这个版本中，Cluster Logging Operator 会告知您配置问题。(LOG-4441)

- 在此次更新之前，如果为 Splunk 输出指定的 secret 中缺少 **hecToken** 键或不正确，验证会失败，因为向量在没有令牌的情况下将日志转发到 Splunk。在这个版本中，如果 **hecToken** 键缺失或不正确，验证会失败，并显示 **A non-empty hecToken entry is required** 错误消息。(LOG-4580)

- 在此次更新之前，使用 web 控制台从**自定义时间范围**中选择日志的日期会导致出现错误。在这个版本中，您可以在 web 控制台中成功从时间范围模型中选择日期。(LOG-4684)

1.3.1.2. CVE

- CVE-2023-40217

- CVE-2023-44487

1.3.2. Logging 5.7.7

此发行版本包括 OpenShift Logging 程序错误修复 5.7.7。

1.3.2.1. 程序错误修复

- 在此次更新之前，FluentD 规范化由 EventRouter 发送的日志与 Vector 不同。在这个版本中，Vector 以一致的格式生成日志记录。(LOG-4178)

- 在此次更新之前，在由 Cluster Logging Operator 创建的指标仪表板中，查询中存在一个错误，用于 **FluentD Buffer Availability** 图，因为它显示最小缓冲区用量。在这个版本中，图形显示最大缓冲区使用量，现在被重命名为 **FluentD Buffer Usage**。(LOG-4555)

- 在此次更新之前，在 IPv6 或双栈 OpenShift Container Platform 集群上部署 LokiStack 会导致 LokiStack memberlist 注册失败。因此，经销商 Pod 会进入崩溃循环。在这个版本中，管理员可以通过将 **lokistack.spec.hashRing.memberlist.enableIPv6:** 值设置为 **true** 来启用 IPv6，这会解决这个问题。(LOG-4569)

- 在此次更新之前，日志收集器依赖于默认配置设置来读取容器日志行。因此，日志收集器无法有效地读取轮转的文件。在这个版本中，读取字节数会增加，允许日志收集器高效地处理轮转文件。(LOG-4575)

- 在此次更新之前，事件路由器中未使用的指标会导致容器因为过量内存用量而失败。在这个版本中，通过删除未使用的指标来减少事件路由器的内存用量。(LOG-4686)

1.3.2.2. CVE

- CVE-2023-0800
- CVE-2023-0801
- CVE-2023-0802
- CVE-2023-0803
- CVE-2023-0804
- CVE-2023-2002
- CVE-2023-3090
- CVE-2023-3390
- CVE-2023-3776
- CVE-2023-4004
- CVE-2023-4527
- CVE-2023-4806
- CVE-2023-4813
- CVE-2023-4863
- CVE-2023-4911
- CVE-2023-5129
- CVE-2023-20593
- CVE-2023-29491
- CVE-2023-30630
- CVE-2023-35001

- CVE-2023-35788

1.3.3. Logging 5.7.6

此发行版本包括 OpenShift Logging 程序错误修复 5.7.6。

1.3.3.1. 程序错误修复

- 在此次更新之前，收集器依赖于默认配置设置来读取容器日志行。因此，收集器无法有效地读取轮转的文件。在这个版本中，读取字节数会增加，允许收集器高效地处理轮转文件。(LOG-4501)
- 在此次更新之前，当用户使用预定义的过滤器粘贴 URL 时，一些过滤器没有反映。在这个版本中，UI 反映了 URL 中的所有过滤器。(LOG-4459)
- 在此次更新之前，使用自定义标签转发到 Loki 会在从 Fluentd 切换到 Vector 时生成错误。在这个版本中，Vector 配置清理标签与 Fluentd 相同，以确保收集器启动并正确处理标签。(LOG-4460)
- 在此次更新之前，Observability Logs 控制台搜索字段不接受它应该转义的特殊字符。在这个版本中，它会在查询中正确转义特殊字符。(LOG-4456)
- 在此次更新之前，在将日志发送到 Splunk 会出现以下镜像信息：**Timestamp was not found.**在这个版本中，更改会覆盖用于检索 Timestamp 的日志字段的名称，并将其发送到 Splunk，而不发出警告。(LOG-4413)
- 在此次更新之前，向量的 CPU 和内存用量随着时间增加。在这个版本中，Vector 配置包含 **expire_metrics_secs=60** 设置来限制指标的生命周期，并上限相关的 CPU 用量和内存占用量。(LOG-4171)
- 在此次更新之前，LokiStack 网关会广泛缓存授权请求。因此，这会导致错误的授权结果。在这个版本中，Loki 网关缓存以更精细的方式缓存来解决这个问题。(LOG-4393)
- 在此次更新之前，Fluentd 运行时镜像包含在运行时不需要的构建程序工具。在这个版本中，构建器工具已被删除，从而解决了这个问题。(LOG-4467)

1.3.3.2. CVE

- CVE-2023-3899
- CVE-2023-4456
- CVE-2023-32360
- CVE-2023-34969

1.3.4. Logging 5.7.4

此发行版本包括 OpenShift Logging 程序错误修复 5.7.4。

1.3.4.1. 程序错误修复

- 在此次更新之前，当将日志转发到 CloudWatch 时，**namespaceUUID** 值不会被附加到 **logGroupName** 字段中。在这个版本中，包含 **namespaceUUID** 值，因此 CloudWatch 中的 **logGroupName** 显示为 **logGroupName: vectorcw.b443fb9e-bd4c-4b6a-b9d3-c0097f9ed286**。(LOG-2701)

- 在此次更新之前，当通过 HTTP 将日志转发到非集群目的地时，向量收集器无法向集群范围的 HTTP 代理进行身份验证，即使代理 URL 中提供了正确的凭证。在这个版本中，Vector 日志收集器可以向集群范围的 HTTP 代理进行身份验证。(LOG-3381)

- 在此次更新之前，如果 Fluentd 收集器配置了 Splunk 作为输出，Operator 将失败，因为不支持此配置。在这个版本中，配置验证会拒绝不支持的输出，从而解决了这个问题。(LOG-4237)

- 在此次更新之前，当 Vector 收集器在 AWS Cloudwatch 日志的 TLS 配置中更新了 **enabled = true** 值时，GCP Stackdriver 会导致配置错误。在这个版本中，为这些输出删除 **enabled = true** 值，从而解决了这个问题。(LOG-4242)

- 在此次更新之前，向量收集器偶尔会在日志中出现以下错误信息：**thread 'vector-worker' panicked at 'all branch are disabled, no else branch', src/kubernetes/reflector.rs:26:9**。在这个版本中，这个错误已解决。(LOG-4275)

- 在此次更新之前，如果 Operator 配置了该租户的额外选项，Loki Operator 中的问题会导致应用程序租户的 **alert-manager** 配置消失。在这个版本中，生成的 Loki 配置包含自定义和自动生成的配置。(LOG-4361)

- 在此次更新之前，当使用多个角色使用带有 AWS Cloudwatch 转发的 STS 进行身份验证时，最近更新会导致凭证不是唯一的。在这个版本中，STS 角色和静态凭证的多个组合可以再次用于与 AWS Cloudwatch 进行身份验证。(LOG-4368)

- 在此次更新之前，Loki 为活跃流过滤标签值，但没有删除重复，使 Grafana 的标签浏览器不可用。在这个版本中，Loki 会过滤活跃流的重复标签值，从而解决了这个问题。(LOG-4389)

- 升级到 OpenShift Logging 5.7 后，在 **ClusterLogForwarder** 自定义资源 (CR) 中没有指定 **name** 字段的管道将停止工作。在这个版本中，这个错误已解决。(LOG-4120)

1.3.4.2. CVE

- CVE-2022-25883

- CVE-2023-22796

1.3.5. Logging 5.7.3

此发行版本包括 OpenShift Logging 程序错误修复 5.7.3。

1.3.5.1. 程序错误修复

- 在此次更新之前，当在 OpenShift Container Platform Web 控制台中查看日志时，缓存的文件会导致数据无法刷新。在这个版本中，bootstrap 文件不会被缓存，从而解决了这个问题。(LOG-4100)

- 在此次更新之前，Loki Operator 会重置错误，导致识别配置问题很难排除故障。在这个版本中，错误会保留，直到配置错误解决为止。(LOG-4156)

- 在此次更新之前，在更改 **RulerConfig** 自定义资源(CR) 后，LokiStack 规则器不会重启。在这个版本中，Loki Operator 在更新 **RulerConfig** CR 后重启规则 pod。(LOG-4161)

- 在此次更新之前，当输入匹配标签值包含 **ClusterLogForwarder** 中的 / 字符时，向量收集器意外终止。在这个版本中，通过引用 match 标签解决了这个问题，使收集器能够启动和收集日志。(LOG-4176)

- 在此次更新之前，当 **LokiStack** CR 定义租户限制而不是全局限制时，Loki Operator 意外终止。在这个版本中，Loki Operator 可以在没有全局限制的情况下处理 **LokiStack** CR，从而解决了这个问题。(LOG-4198)
- 在此次更新之前，当提供的私钥受密码保护时，Fluentd 不会将日志发送到 Elasticsearch 集群。在这个版本中，Fluentd 在与 Elasticsearch 建立连接时可以正确地处理受密语保护的私钥。(LOG-4258)
- 在此次更新之前，具有超过 8,000 个命名空间的集群会导致 Elasticsearch 拒绝查询，因为命名空间列表大于 **http.max_header_size** 设置。在这个版本中，标头大小的默认值有所增加，从而解决了这个问题。(LOG-4277)
- 在此次更新之前，**ClusterLogForwarder** CR 中包含 / 字符的标签值会导致收集器意外终止。在这个版本中，斜杠被下划线替代，从而解决了这个问题。(LOG-4095)
- 在此次更新之前，Cluster Logging Operator 会在设置为非受管状态时意外终止。在这个版本中，在启动 **ClusterLogForwarder** CR 的协调前，确保 **ClusterLogging** 资源处于正确的管理状态，从而解决了这个问题。(LOG-4177)
- 在此更新之前，当在 OpenShift Container Platform Web 控制台中查看日志时，通过拖放到直方图中的时间范围无法用于 pod 详情中的聚合日志视图。在这个版本中，可以通过拖动到这个视图中的直方图来选择时间范围。(LOG-4108)
- 在此次更新之前，当在 OpenShift Container Platform Web 控制台中查看日志时，查询会超过 30 秒超时。在这个版本中，超时值可以在 configmap/logging-view-plugin 中配置。(LOG-3498)
- 在此次更新之前，当在 OpenShift Container Platform Web 控制台中查看日志时，**点更多数据可用**选项仅在第一次点击时加载更多日志条目。在这个版本中，每次点击时会加载更多条目。(OU-188)
- 在此次更新之前，当在 OpenShift Container Platform Web 控制台中查看日志时，点 **streaming** 选项只显示 **流传输** 日志消息，而无需显示实际日志。在这个版本中，消息和日志流都会正确显示。(OU-166)

1.3.5.2. CVE

- CVE-2020-24736
- CVE-2022-48281
- CVE-2023-1667
- CVE-2023-2283
- CVE-2023-24329
- CVE-2023-26115
- CVE-2023-26136
- CVE-2023-26604
- CVE-2023-28466

1.3.6. Logging 5.7.2

此发行版本包括 OpenShift Logging 程序错误修复 5.7.2。

1.3.6.1. 程序错误修复

- 在此次更新之前，因为存在待处理的终结器，无法直接删除 **openshift-logging** 命名空间。在这个版本中，不再使用终结器 (finalizer)，启用直接删除命名空间。(LOG-3316)
- 在此次更新之前，如果根据 OpenShift Container Platform 文档更改了 **run.sh** 脚本，则 run.sh 脚本会显示一个不正确的 **chunk_limit_size** 值。但是，当通过环境变量 **$BUFFER_SIZE_LIMIT** 设置 **chunk_limit_size** 时，该脚本会显示正确的值。在这个版本中，**run.sh** 脚本会在这两种场景中都一致地显示正确的 **chunk_limit_size** 值。(LOG-3330)
- 在此次更新之前，OpenShift Container Platform Web 控制台的日志记录视图插件不允许自定义节点放置或容限。在这个版本中，增加了为日志记录视图插件定义节点放置和容限的功能。(LOG-3749)
- 在此次更新之前，当尝试通过 Fluentd HTTP 插件将日志发送到 DataDog 时，Cluster Logging Operator 遇到 Unsupported Media Type 异常。在这个版本中，用户可以通过配置 HTTP 标头 Content-Type 为日志转发无缝分配内容类型。提供的值会自动分配给插件中的 **content_type** 参数，确保成功传输日志。(LOG-3784)
- 在此次更新之前，当 **ClusterLogForwarder** 自定义资源 (CR) 中的 **detectMultilineErrors** 字段设置为 **true** 时，PHP 多行错误被记录为单独的日志条目，从而导致堆栈跟踪在多个消息间分割。在这个版本中，启用了 PHP 的多行错误检测，确保整个堆栈追踪包含在单一日志消息中。(LOG-3878)
- 在此次更新之前，**ClusterLogForwarder** 管道的名称中包含空格会导致 Vector 收集器 pod 持续崩溃。在这个版本中，管道名称中的所有空格、短划线 (-) 和点 (.) 都被替换为下划线 (_)。(LOG-3945)
- 在此次更新之前，**log_forwarder_output** 指标不包括 **http** 参数。在这个版本中，在指标中添加缺少的参数。(LOG-3997)
- 在此次更新之前，Fluentd 在以冒号结尾时无法识别一些多行 JavaScript 客户端异常。在这个版本中，Fluentd 缓冲名称的前缀为下划线，从而解决了这个问题。(LOG-4019)
- 在此次更新之前，当将日志转发配置为写入与有效负载中键匹配的 Kafka 输出主题时，日志会因为错误而丢弃。在这个版本中，Fluentd 的缓冲区名称前缀为下划线，从而解决了这个问题。(LOG-4027)
- 在此次更新之前，LokiStack 网关返回命名空间的标签值，而无需应用用户的访问权限。在这个版本中，Loki 网关应用标签值请求的权限，从而解决了这个问题。(LOG-4049)
- 在此次更新之前，当 **tls.insecureSkipVerify** 选项被设置为 **true** 时，Cluster Logging Operator API 需要一个由 secret 提供的证书。在这个版本中，Cluster Logging Operator API 不再需要在这样的情形中由 secret 提供证书。以下配置已添加到 Operator 的 CR 中：

  ```
  tls.verify_certificate = false
  tls.verify_hostname = false
  ```

  (LOG-3445)
- 在此次更新之前，LokiStack 路由配置会导致查询运行时间超过 30 秒。在这个版本中，Loki global 和 per-tenant **queryTimeout** 设置会影响路由超时设置，从而解决了这个问题。(LOG-4052)

- 在此次更新之前，删除 **collection.type** 的默认修复会导致 Operator 不再遵循资源、节点选择和容限已弃用的 spec。在这个版本中，Operator 的行为总是首选 **collection.logs** spec 而不是这些**集合**。这与之前允许使用首选字段和已弃用字段的行为不同，但在填充 **collection.type** 时会忽略已弃用的字段。(LOG-4185)

- 在此次更新之前，如果输出中没有指定代理 URL，Vector 日志收集器不会生成 TLS 配置，用于将日志转发到多个 Kafka 代理。在这个版本中，为多个代理生成 TLS 配置。(LOG-4163)

- 在此次更新之前，为登录到 Kafka 的日志转发启用密码短语的选项不可用。这个限制会导致安全风险，因为它可能会公开敏感信息。在这个版本中，用户有一个无缝选项来为登录到 Kafka 的日志转发启用密码短语。(LOG-3314)

- 在此次更新之前，Vector 日志收集器不会遵循传出 TLS 连接的 **tlsSecurityProfile** 设置。在这个版本中，Vector 可以正确地处理 TLS 连接设置。(LOG-4011)

- 在此次更新之前，在 log_forwarder_output_info 指标中，并非所有可用的输出类型都包括在 **log_forwarder_output_info** 指标中。在这个版本中，指标包含之前缺少的 Splunk 和 Google Cloud Logging 数据。(LOG-4098)

- 在此次更新之前，当将 **follow_inodes** 设置为 **true** 时，Fluentd 收集器可能会在文件轮转时崩溃。在这个版本中，**follow_inodes** 设置不会使收集器崩溃。(LOG-4151)

- 在此次更新之前，Fluentd 收集器可能会因为如何跟踪这些文件而错误地关闭应该监视的文件。在这个版本中，跟踪参数已被修正。(LOG-4149)

- 在此次更新之前，使用 Vector 收集器转发日志，并在 **ClusterLogForwarder** 实例 **audit** 中命名管道，**application** 或 **infrastructure** 会导致收集器 pod 处于 **CrashLoopBackOff** 状态，并在收集器日志中出现以下错误：

  > ERROR vector::cli: Configuration error. error=redefinition of table transforms.audit for key transforms.audit

  在这个版本中，管道名称不再与保留输入名称冲突，管道可以被命名为 **audit**,**application** 或 **infrastructure**。(LOG-4218)

- 在此次更新之前，当将日志转发到带有 Vector 收集器的 syslog 目的地，并将 **addLogSource** 标志设置为 **true** 时，将以下额外空字段添加到转发消息：**namespace_name=**、**container_name=** 和 **pod_name=**。在这个版本中，这些字段不再添加到日志日志中。(LOG-4219)

- 在此次更新之前，当未找到 **structuredTypeKey** 且没有指定 **structuredTypeName** 时，日志消息仍然被解析为结构化对象。在这个版本中，日志的解析如预期。(LOG-4220)

1.3.6.2. CVE

- CVE-2021-26341
- CVE-2021-33655
- CVE-2021-33656
- CVE-2022-1462
- CVE-2022-1679
- CVE-2022-1789

- CVE-2022-2196
- CVE-2022-2663
- CVE-2022-3028
- CVE-2022-3239
- CVE-2022-3522
- CVE-2022-3524
- CVE-2022-3564
- CVE-2022-3566
- CVE-2022-3567
- CVE-2022-3619
- CVE-2022-3623
- CVE-2022-3625
- CVE-2022-3627
- CVE-2022-3628
- CVE-2022-3707
- CVE-2022-3970
- CVE-2022-4129
- CVE-2022-20141
- CVE-2022-25147
- CVE-2022-25265
- CVE-2022-30594
- CVE-2022-36227
- CVE-2022-39188
- CVE-2022-39189
- CVE-2022-41218
- CVE-2022-41674
- CVE-2022-42703
- CVE-2022-42720
- CVE-2022-42721

- CVE-2022-42722
- CVE-2022-43750
- CVE-2022-47929
- CVE-2023-0394
- CVE-2023-0461
- CVE-2023-1195
- CVE-2023-1582
- CVE-2023-2491
- CVE-2023-22490
- CVE-2023-23454
- CVE-2023-23946
- CVE-2023-25652
- CVE-2023-25815
- CVE-2023-27535
- CVE-2023-29007

1.3.7. Logging 5.7.1

此发行版本包括：OpenShift Logging 程序错误修复 5.7.1。

1.3.7.1. 程序错误修复

- 在此次更新之前，Cluster Logging Operator pod 日志中存在大量信息会导致日志的可读性降低，并增加识别重要系统事件的难度。在这个版本中，这个问题可以通过显著降低 Cluster Logging Operator pod 日志中的信息来解决。(LOG-3482)
- 在此次更新之前，API 服务器会将 **CollectorSpec.Type** 字段的值重置为 **vector**，即使自定义资源使用了不同的值。在这个版本中，删除了 **CollectorSpec.Type** 字段的默认设置来恢复之前的行为。(LOG-4086)
- 在此次更新之前，无法通过点击并在日志直方图上拖动，在 OpenShift Container Platform Web 控制台中选择时间范围。在这个版本中，可以使用单击和拖动来成功选择时间范围。(LOG-4501)
- 在此次更新之前，点 OpenShift Container Platform Web 控制台中的 **Show Resources** 链接不会产生任何影响。在这个版本中，通过修复 "Show Resources" 链接的功能来解决这个问题，以切换每个日志条目的资源显示。(LOG-3218)

1.3.7.2. CVE

- CVE-2023-21930
- CVE-2023-21937

- CVE-2023-21938
- CVE-2023-21939
- CVE-2023-21954
- CVE-2023-21967
- CVE-2023-21968
- CVE-2023-28617

1.3.8. Logging 5.7.0

此发行版本包括 OpenShift Logging 程序错误修复 5.7.0。

1.3.8.1. 功能增强

在这个版本中，您可以启用日志记录来检测多行异常，并将其重新编译到一条日志条目中。

要启用日志记录来检测多行异常，并将其重新编译到一个日志条目中，请确保 **ClusterLogForwarder** 自定义资源 (CR) 包含 **detectMultilineErrors** 字段，值为 **true**。

1.3.8.2. 已知问题

无。

1.3.8.3. 程序错误修复

- 在此次更新之前，LokiHost 的 Gateway 组件的 **nodeSelector** 属性不会影响节点调度。在这个版本中，**nodeSelector** 属性可以正常工作。(LOG-3713)

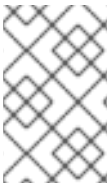
1.3.8.4. CVE

- CVE-2023-1999
- CVE-2023-28617

第 2 章 支持

logging 只支持本文档中介绍的配置选项。

不要使用任何其他配置选项，因为它们不被支持。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。如果您使用本文档中描述的配置以外的配置，您的更改会被覆盖，因为 Operator 旨在协调差异。



注意

如果必须执行 OpenShift Container Platform 文档中未描述的配置，您需要将 Red Hat OpenShift Logging Operator 设置为 **Unmanaged**。不支持非受管日志记录实例，且不会接收更新，直到您将其状态返回为 **Managed** 为止。

注意

日志记录作为一个可安装的组件提供，它有一个不同于 OpenShift Container Platform 的发布周期。Red Hat OpenShift Container Platform 生命周期政策 概述了发行版本兼容性。

Red Hat OpenShift 的 logging 是一个建议的收集器，以及应用程序、基础架构和审计日志的规范化程序。它旨在将日志转发到各种支持的系统。

Logging 不是：

- 一个大规模日志收集系统
- 兼容安全信息和事件监控 (SIEM)
- 历史或长日志的保留或存储
- 保证的日志接收器
- 安全存储 - 默认不存储审计日志

2.1. 支持的 API 自定义资源定义

LokiStack 开发正在进行。目前还不支持所有 API。

表 2.1. Loki API 支持状态

CustomResourceDefinition (CRD)	ApiVersion	支持状态
LokiStack	lokistack.loki.grafana.com/v1	在 5.5 中支持
RulerConfig	rulerconfig.loki.grafana/v1	在 5.7 中支持
AlertingRule	alertingrule.loki.grafana/v1	在 5.7 中支持
RecordingRule	recordingrule.loki.grafana/v1	在 5.7 中支持

2.2. 不支持的配置

您必须将 Red Hat OpenShift Logging Operator 设置为 **Unmanaged** 状态才能修改以下组件：

- **Elasticsearch** 自定义资源(CR)
- Kibana 部署
- **fluent.conf** 文件
- Fluentd 守护进程集

您必须将 OpenShift Elasticsearch Operator 设置为 **Unmanaged** 状态，才能修改 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。

2.3. 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。

2.4. 为红帽支持收集日志记录数据

在提交问题单时，向红帽支持提供有关集群的调试信息会很有帮助。

您可以使用 **must-gather** 工具来收集有关 项目级别资源、集群级资源和每个日志记录组件的诊断信息。

为了获得快速支持，请提供 OpenShift Container Platform 和日志记录的诊断信息。

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。

2.4.1. 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。

对于日志记录，**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。

2.4.2. 收集日志记录数据

您可以使用 **oc adm must-gather** CLI 命令来收集有关日志记录的信息。

流程

使用 **must-gather** 来收集日志信息：

1. 进入要存储 **must-gather** 信息的目录。

2. 针对日志记录镜像运行 **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。

第 3 章 日志故障排除

3.1. 查看日志记录状态

您可以查看 Red Hat OpenShift Logging Operator 的状态和其他日志记录组件。

3.1.1. 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

- 安装了 Red Hat OpenShift Logging Operator 和 OpenShift Elasticsearch Operator。

流程

1. 运行以下命令，切换到 **openshift-logging** 项目：

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

2. 运行以下命令来获取 **ClusterLogging** 实例状态：

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

输出示例

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

3.1.1.1. 情况消息示例

以下是来自 **ClusterLogging** 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:
```

3.1.2. 查看日志记录组件的状态

您可以查看多个日志记录组件的状态。

先决条件

- 安装了 Red Hat OpenShift Logging Operator 和 OpenShift Elasticsearch Operator。

流程

1. 进入 **openshift-logging** 项目。

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

2. 查看日志记录环境的状态：

```
$ 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. 查看日志记录副本集的状态：

   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----
```

3.2. 日志转发故障排除

3.2.1. 重新部署 Fluentd pod

当您创建 **ClusterLogForwarder** 自定义资源 (CR) 时，如果 Red Hat OpenShift Logging Operator 没有自动重新部署 Fluentd Pod，您可以删除 Fluentd Pod 来强制重新部署它们。

先决条件

- 您已创建了 **ClusterLogForwarder** 自定义资源 (CR) 对象。

流程

- 运行以下命令，删除 Fluentd pod 以强制重新部署：

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

3.2.2. Loki 速率限制错误故障排除

如果 Log Forwarder API 将超过速率限制的大量信息转发到 Loki，Loki 会生成速率限制(**429**)错误。

这些错误可能会在正常操作过程中发生。例如，当将 logging 添加到已具有某些日志的集群中时，logging 会尝试充分利用现有日志条目时可能会出现速率限制错误。在这种情况下，如果添加新日志的速度小于总速率限值，历史数据最终会被处理，并且不要求用户干预即可解决速率限制错误。

如果速率限制错误持续发生，您可以通过修改 **LokiStack** 自定义资源(CR)来解决此问题。

重要

LokiStack CR 在 Grafana 托管的 Loki 上不可用。本主题不适用于 Grafana 托管的 Loki 服务器。

Conditions

- Log Forwarder API 配置为将日志转发到 Loki。

- 您的系统向 Loki 发送大于 2 MB 的消息块。例如：

```
"values":[["1630410392689800468","{\"kind\":\"Event\",\"apiVersion\":\
.......
......
......
......
\"received_at\":\"2021-08-31T11:46:32.800278+00:00\",\"version\":\"1.7.4
1.6.0\"}},\"@timestamp\":\"2021-08-
31T11:46:32.799692+00:00\",\"viaq_index_name\":\"audit-
write\",\"viaq_msg_id\":\"MzFjYjJkZjItNjY0MC00YWU4LWIwMTEtNGNmM2E5ZmViMGU4\",\"lo
g_type\":\"audit\"}"]]}]}
```

- 输入 **oc logs -n openshift-logging -l component=collector** 后，集群中的收集器日志会显示包含以下错误消息之一的行：

```
429 Too Many Requests Ingestion rate limit exceeded
```

Vector 错误消息示例

```
2023-08-25T16:08:49.301780Z  WARN sink{component_kind="sink"
component_id=default_loki_infra component_type=loki component_name=default_loki_infra}:
vector::sinks::util::retries: Retrying after error. error=Server responded with an error: 429 Too
Many Requests internal_log_rate_limit=true
```

Fluentd 错误消息示例

```
2023-08-30 14:52:15 +0000 [warn]: [default_loki_infra] failed to flush the buffer. retry_times=2
next_retry_time=2023-08-30 14:52:19 +0000
chunk="604251225bf5378ed1567231a1c03b8b"
error_class=Fluent::Plugin::LokiOutput::LogPostError error="429 Too Many Requests
Ingestion rate limit exceeded for user infrastructure (limit: 4194304 bytes/sec) while
attempting to ingest '4082' lines totaling '7820025' bytes, reduce log volume or contact your
Loki administrator to see if the limit can be increased\n"
```

在接收结束时也会看到这个错误。例如，在 LokiStack ingester pod 中：

Loki ingester 错误消息示例

```
level=warn ts=2023-08-30T14:57:34.155592243Z caller=grpc_logging.go:43
duration=1.434942ms method=/logproto.Pusher/Push err="rpc error: code = Code(429) desc
= entry with timestamp 2023-08-30 14:57:32.012778399 +0000 UTC ignored, reason: 'Per
stream rate limit exceeded (limit: 3MB/sec) while attempting to ingest for stream
```

流程

- 更新 **LokiStack** CR 中的 **ingestionBurstSize** 和 **ingestionRate** 字段：

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
```

```
spec:
  limits:
    global:
      ingestion:
        ingestionBurstSize: 16 (1)
        ingestionRate: 8 (2)
# ...
```

(1) **ingestionBurstSize** 字段定义每个经销商副本的最大本地速率限制示例大小（以 MB 为单位）。这个值是一个硬限制。将此值设置为至少在单个推送请求中预期的最大日志大小。不允许大于 **ingestionBurstSize** 值的单个请求。

(2) **ingestionRate** 字段是每秒最大最大样本量的软限制（以 MB 为单位）。如果日志速率超过限制，则会出现速率限制错误，但收集器会重试发送日志。只要总平均值低于限制，系统就会在没有用户干预的情况下解决错误。

3.3. 日志记录警报故障排除

您可以使用以下步骤排除集群中的日志记录警报。

3.3.1. Elasticsearch 集群健康状态为红色

至少一个主分片及其副本没有分配给节点。使用以下步骤对此警报进行故障排除。

提示

本文档中的一些命令会使用 **$ES_POD_NAME** shell 变量来引用 Elasticsearch pod。如果要直接从本文档中复制并粘贴命令，您必须将此变量设置为对 Elasticsearch 集群有效的值。

您可以运行以下命令来列出可用的 Elasticsearch pod：

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

运行以下命令，选择列出的 pod 并设置 **$ES_POD_NAME** 变量：

```
$ export ES_POD_NAME=<elasticsearch_pod_name>
```

现在，您可以在命令中使用 **$ES_POD_NAME** 变量。

流程

1. 运行以下命令，检查 Elasticsearch 集群健康状况并验证集群**状态**是否为红色：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME -- health
```

2. 运行以下命令，列出已加入集群的节点：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=_cat/nodes?v
```

3. 运行以下命令，列出 Elasticsearch Pod，并将它们与上一步中的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. 运行以下命令并查看输出，确认 Elasticsearch 已选定 master 节点：

   ```
   $ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
     -- es_util --query=_cat/master?v
   ```

   b. 运行以下命令，并查看所选 master 节点的 pod 日志问题：

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

   c. 运行以下命令并查看没有加入集群的节点日志：

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

5. 如果所有节点都已加入集群，请运行以下命令检查集群是否处于恢复过程中，并观察输出：

   ```
   $ oc exec -n openshift-logging -c elasticsearch $ES_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. 运行以下命令并查看输出，检查是否有待处理的任务：

   ```
   $ oc exec -n openshift-logging -c elasticsearch $ES_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，请运行以下命令检查 **cluster.routing.allocation.enable** 值设置为 **none**，然后观察输出：

   ```
   $ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
     -- es_util --query=_cluster/settings?pretty
   ```

9. 如果 **cluster.routing.allocation.enable** 被设为 **none**，请运行以下命令将其设置为 **all** ：

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

10. 运行以下命令并查看输出，检查任何索引仍然是红色的：

    ```
    $ oc exec -n openshift-logging -c elasticsearch $ES_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. 运行以下命令来清除缓存：

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

b. 运行以下命令来增加最大分配重试次数：

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

c. 运行以下命令来删除所有滚动项：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=_search/scroll/_all -X DELETE
```

d. 运行以下命令来增加超时：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_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. 运行以下命令来识别红色索引名称：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=_cat/indices?v
```

b. 运行以下命令来删除红色索引：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_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. 运行以下命令，检查 Elasticsearch JVM 堆使用率是否高：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_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。如需有关 CPU 利用率的更多信息，请参阅 OpenShift Container Platform "查看监控仪表板"文档。

其他资源

- 查看监控仪表板
- 修复红色或黄色集群状态

3.3.2. Elasticsearch 集群健康状态黄色

至少一个主分片的副本分片没有分配给节点。通过调整 **ClusterLogging** 自定义资源 (CR) 中的 **nodeCount** 值来增加节点数。

其他资源

- 修复红色或黄色集群状态

3.3.3. 已达到 Elasticsearch 节点磁盘低水位线

Elasticsearch 不会将分片分配给达到低水位线的节点。

提示

本文档中的一些命令会使用 **$ES_POD_NAME** shell 变量来引用 Elasticsearch pod。如果要直接从本文档中复制并粘贴命令，您必须将此变量设置为对 Elasticsearch 集群有效的值。

您可以运行以下命令来列出可用的 Elasticsearch pod：

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

运行以下命令，选择列出的 pod 并设置 **$ES_POD_NAME** 变量：

```
$ export ES_POD_NAME=<elasticsearch_pod_name>
```

现在，您可以在命令中使用 **$ES_POD_NAME** 变量。

流程

1. 运行以下命令，识别在其上部署 Elasticsearch 的节点：

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

2. 运行以下命令，检查是否有未分配的分片：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=_cluster/health?pretty | grep unassigned_shards
```

3. 如果存在未分配的分片，请运行以下命令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. 在命令输出中，检查 **Use** 列以确定该节点上使用的磁盘百分比。

输出示例

```
elasticsearch-cdm-kcrsda6l-1-586cc95d4f-h8zq8
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme1n1     19G  522M   19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-2-5b548fc7b-cwwk7
Filesystem      Size  Used Avail Use% Mounted on
```

```
/dev/nvme2n1     19G  522M   19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-3-5dfc884d99-59tjw
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme3n1     19G  528M   19G   3% /elasticsearch/persistent
```

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. 要检查当前的 **redundancyPolicy**，请运行以下命令：

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

如果在集群中使用 **ClusterLogging** 资源，请运行以下命令：

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

如果集群 **redundancyPolicy** 值高于 **SingleRedundancy** 值，将其设置为 **SingleRedundancy** 值并保存这个更改。

6. 如果前面的步骤没有解决这个问题，请删除旧的索引。

   a. 运行以下命令，检查 Elasticsearch 上所有索引的状态：

   ```
   $ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME -- indices
   ```

   b. 确定可以删除的旧索引。

   c. 运行以下命令来删除索引：

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

3.3.4. 已达到 Elasticsearch 节点磁盘高水位线

Elasticsearch 会尝试将分片从达到高水位线的节点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。

要将分片分配给特定节点，您必须释放该节点上的一些空间。如果无法增加磁盘空间，请尝试向集群添加新数据节点，或者减少集群冗余策略总数。

提示

本文档中的一些命令会使用 **$ES_POD_NAME** shell 变量来引用 Elasticsearch pod。如果要直接从本文档中复制并粘贴命令，您必须将此变量设置为对 Elasticsearch 集群有效的值。

您可以运行以下命令来列出可用的 Elasticsearch pod：

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

运行以下命令，选择列出的 pod 并设置 **$ES_POD_NAME** 变量：

```
$ export ES_POD_NAME=<elasticsearch_pod_name>
```

现在，您可以在命令中使用 **$ES_POD_NAME** 变量。

流程

1. 运行以下命令，识别在其上部署 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 $ES_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。高水位线的默认值为 90%。

4. 增加所有节点上的磁盘空间。如果无法增加磁盘空间，请尝试向集群添加新数据节点，或者减少集群冗余策略总数。

5. 要检查当前的 **redundancyPolicy**，请运行以下命令：

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

   如果在集群中使用 **ClusterLogging** 资源，请运行以下命令：

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

   如果集群 **redundancyPolicy** 值高于 **SingleRedundancy** 值，将其设置为 **SingleRedundancy** 值并保存这个更改。

6. 如果前面的步骤没有解决这个问题，请删除旧的索引。

   a. 运行以下命令，检查 Elasticsearch 上所有索引的状态：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME -- indices
```

b. 确定可以删除的旧索引。

c. 运行以下命令来删除索引：

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

3.3.5. 已达到 Elasticsearch 节点磁盘水位线

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。

使用以下步骤对此警报进行故障排除。

提示

本文档中的一些命令会使用 **$ES_POD_NAME** shell 变量来引用 Elasticsearch pod。如果要直接从本文档中复制并粘贴命令，您必须将此变量设置为对 Elasticsearch 集群有效的值。

您可以运行以下命令来列出可用的 Elasticsearch pod：

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

运行以下命令，选择列出的 pod 并设置 **$ES_POD_NAME** 变量：

```
$ export ES_POD_NAME=<elasticsearch_pod_name>
```

现在，您可以在命令中使用 **$ES_POD_NAME** 变量。

流程

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 **Avail** 列以确定该节点上的可用磁盘空间。

输出示例

```
elasticsearch-cdm-kcrsda6l-1-586cc95d4f-h8zq8
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme1n1     19G  522M   19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-2-5b548fc7b-cwwk7
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme2n1     19G  522M   19G   3% /elasticsearch/persistent
```

```
elasticsearch-cdm-kcrsda6l-3-5dfc884d99-59tjw
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme3n1     19G  528M   19G   3% /elasticsearch/persistent
```

3. 增加所有节点上的磁盘空间。如果无法增加磁盘空间，请尝试向集群添加新数据节点，或者减少集群冗余策略总数。

4. 要检查当前的 **redundancyPolicy**，请运行以下命令：

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

如果在集群中使用 **ClusterLogging** 资源，请运行以下命令：

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

如果集群 **redundancyPolicy** 值高于 **SingleRedundancy** 值，将其设置为 **SingleRedundancy** 值并保存这个更改。

5. 如果前面的步骤没有解决这个问题，请删除旧的索引。

   a. 运行以下命令，检查 Elasticsearch 上所有索引的状态：

   ```
   $ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME -- indices
   ```

   b. 确定可以删除的旧索引。

   c. 运行以下命令来删除索引：

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

6. 继续释放和监控磁盘空间。在使用的磁盘空间低于 90% 后，运行以下命令来取消阻塞写入此节点：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME \
  -- es_util --query=_all/_settings?pretty \
  -X PUT -d '{"index.blocks.read_only_allow_delete": null}'
```

3.3.6. Elasticsearch JVM 堆使用率很高

使用的 Elasticsearch 节点 Java 虚拟机(JVM)堆内存超过 75%。考虑增大堆大小。

3.3.7. 聚合日志记录系统 CPU 为高

节点上的系统 CPU 使用率高。检查集群节点的 CPU。考虑向节点分配更多 CPU 资源。

3.3.8. Elasticsearch 进程 CPU 为高

节点上的 Elasticsearch 进程 CPU 使用率很高。检查集群节点的 CPU。考虑向节点分配更多 CPU 资源。

3.3.9. Elasticsearch 磁盘空间运行较低

根据当前的磁盘用量，Elasticsearch 被预测在下一个 6 小时内耗尽磁盘空间。使用以下步骤对此警报进行故障排除。

流程

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 **Avail** 列以确定该节点上的可用磁盘空间。

   输出示例

```
elasticsearch-cdm-kcrsda6l-1-586cc95d4f-h8zq8
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme1n1     19G  522M   19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-2-5b548fc7b-cwwk7
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme2n1     19G  522M   19G   3% /elasticsearch/persistent
elasticsearch-cdm-kcrsda6l-3-5dfc884d99-59tjw
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme3n1     19G  528M   19G   3% /elasticsearch/persistent
```

3. 增加所有节点上的磁盘空间。如果无法增加磁盘空间，请尝试向集群添加新数据节点，或者减少集群冗余策略总数。

4. 要检查当前的 **redundancyPolicy**，请运行以下命令：

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

   如果在集群中使用 **ClusterLogging** 资源，请运行以下命令：

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

   如果集群 **redundancyPolicy** 值高于 **SingleRedundancy** 值，将其设置为 **SingleRedundancy** 值并保存这个更改。

5. 如果前面的步骤没有解决这个问题，请删除旧的索引。

   a. 运行以下命令，检查 Elasticsearch 上所有索引的状态：

```
$ oc exec -n openshift-logging -c elasticsearch $ES_POD_NAME -- indices
```

   b. 确定可以删除的旧索引。

   c. 运行以下命令来删除索引：

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

其他资源

- 修复红色或黄色集群状态

3.3.10. Elasticsearch FileDescriptor 使用是高

根据当前的使用趋势，预计节点上的文件描述符数量不足。检查每个节点的 **max_file_descriptors** 值，如 Elasticsearch File Descriptors 文档中所述。

3.4. 查看 ELASTICSEARCH 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。

3.4.1. 查看 Elasticsearch 日志存储的状态

您可以查看 Elasticsearch 日志存储的状态。

先决条件

- 安装了 Red Hat OpenShift Logging Operator 和 OpenShift Elasticsearch Operator。

流程

1. 运行以下命令，切换到 **openshift-logging** 项目：

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

2. 查看状态：

   a. 运行以下命令，获取 Elasticsearch 日志存储实例的名称：

```
$ oc get Elasticsearch
```

   输出示例

```
NAME            AGE
elasticsearch   5h9m
```

   b. 运行以下命令来获取 Elasticsearch 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. Elasticsearch 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。
   - Elasticsearch 日志存储数据节点的数量。
   - Elasticsearch 日志存储节点的总数。

- 待处理的任务数量。
- Elasticsearch 日志存储状态： **绿色**、**红色**、**黄色**。
- 未分配分片的数量。

3 任何状态条件（若存在）。Elasticsearch 日志存储状态指示在无法放置 pod 时来自于调度程序的原因。显示与以下情况有关的所有事件：

- 容器正在等待 Elasticsearch 日志存储和代理容器。
- Elasticsearch 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 Elasticsearch 日志存储节点，带有 **upgradeStatus**。

5 Elasticsearch 日志存储在集群中的客户端、数据和 master pod，列在 **failed**、**notReady** 或 **ready** 状态下。

3.4.1.1. 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: {}
```

以下状态消息表示自定义资源(CR)中的 Elasticsearch 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 Elasticsearch 日志存储 CR 使用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 Elasticsearch 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：

```
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: StorageStructureChangeIgnored
    status: 'True'
    type: StorageStructureChangeIgnored
```

reason 和 **type** 类型字段指定不受支持的更改类型：

StorageClassNameChangeIgnored

不支持更改存储类名称。

StorageSizeChangeIgnored

不支持更改存储大小。

StorageStructureChangeIgnored

不支持在临时存储结构和持久性存储结构间更改。

重要

如果您尝试将 **ClusterLogging** 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 **StorageStructureChangeIgnored** 状态，您必须恢复对 **ClusterLogging** CR 的更改并删除 PVC。

3.4.2. 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                                    iDjscH7aSUGhIdq0LheLBQ   1
1          5          0          0          0
green  open   .kibana_-377444158_kubeadmin
yBywZ9GfSrKebz5gWBZbjw   3   1          1          0          0          0
green  open   infra-000001                                  z6Dpe__ORgiopEpW6Yl44A
3   1     871000          0       874          436
green  open   app-000001                                    hIrazQCeSISewG3c2VIvsQ
3   1       2453          0         3          1
green  open   .kibana_1                                    JCitcBMSQxKOvIq6iQW6wg
1   1          0          0          0          0
green  open   .kibana_-1595131456_user1                          gIYFIEGRRe-
ka0W3okS-mQ   3   1          1          0          0          0
```

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>
```

3.4.3. Elasticsearch 集群状态

OpenShift Container Platform Web 控制台的 **Observe** 部分中的仪表板显示 Elasticsearch 集群的状态。

要获取 OpenShift Elasticsearch 集群的状态，请访问位于 **<cluster_url>/monitoring/dashboards/grafana-dashboard-cluster-logging** 的 OpenShift Container Platform Web 控制台的 **Observe** 部分中的 Grafana 仪表板。

Elasticsearch 状态字段

eo_elasticsearch_cr_cluster_management_state

显示 Elasticsearch 集群是否处于受管状态或非受管状态。例如：

```
eo_elasticsearch_cr_cluster_management_state{state="managed"} 1
eo_elasticsearch_cr_cluster_management_state{state="unmanaged"} 0
```

eo_elasticsearch_cr_restart_total

显示 Elasticsearch 节点重启证书、滚动重启或调度重启的次数。例如：

```
eo_elasticsearch_cr_restart_total{reason="cert_restart"} 1
eo_elasticsearch_cr_restart_total{reason="rolling_restart"} 1
eo_elasticsearch_cr_restart_total{reason="scheduled_restart"} 3
```

es_index_namespaces_total

显示 Elasticsearch 索引命名空间的总数。例如：

```
Total number of Namespaces.
es_index_namespaces_total 5
```

es_index_document_count

显示每个命名空间的记录数。例如：

```
es_index_document_count{namespace="namespace_1"} 25
es_index_document_count{namespace="namespace_2"} 10
es_index_document_count{namespace="namespace_3"} 5
```

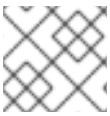
"Secret Elasticsearch fields are either missing or empty" 信息

如果 Elasticsearch 缺少 **admin-cert**、**admin-key**、**logging-es.crt** 或 **logging-es.key** 文件，仪表板会显示类似以下示例的状态消息：

```
message": "Secret \"elasticsearch\" fields are either missing or empty: [admin-cert, admin-key,
logging-es.crt, logging-es.key]",
"reason": "Missing Required Secrets",
```

第 4 章 关于日志记录

作为集群管理员，您可以在 OpenShift Container Platform 集群上部署 logging，并使用它来收集和聚合节点系统日志、应用程序容器日志和基础架构日志。您可以将日志转发到所选的日志输出，包括在线集群、红帽管理的日志存储。您还可以视觉化 OpenShift Container Platform Web 控制台或 Kibana Web 控制台中的日志数据，具体取决于您部署的日志存储解决方案。



注意

Kibana Web 控制台现已弃用，计划在以后的日志记录发行版本中删除。

OpenShift Container Platform 集群管理员可以使用 Operator 部署日志记录。如需更多信息，请参阅安装日志记录。

Operator 负责部署、升级和维护日志记录。安装 Operator 后，您可以创建一个 **ClusterLogging** 自定义资源(CR)来调度日志记录 pod 和支持日志记录所需的其他资源。您还可以创建一个 **ClusterLogForwarder** CR 来指定收集哪些日志、如何转换日志以及它们被转发到的位置。

注意

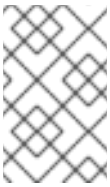
由于内部 OpenShift Container Platform Elasticsearch 日志存储无法为审计日志提供安全存储，所以审计日志默认不会存储在内部 Elasticsearch 实例中。如果要将审计日志发送到默认的内部 Elasticsearch 日志存储，例如要在 Kibana 中查看审计日志，则必须使用 Log Forwarding API，如 将审计日志转发到日志存储 中所述。

4.1. 日志记录架构

日志记录的主要组件有：

Collector

收集器是一个 daemonset，它将 pod 部署到每个 OpenShift Container Platform 节点。它从每个节点收集日志数据，转换数据并将其转发到配置的输出。您可以使用 Vector 收集器或旧的 Fluentd 收集器。

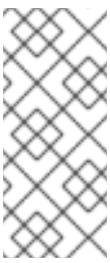


注意

Fluentd 已被弃用，计划在以后的发行版本中删除。红帽将在当前发行生命周期中将提供对这个功能的 bug 修复和支持，但此功能将不再获得改进。作为 Fluentd 的替代选择，您可以使用 Vector。

日志存储

日志存储存储用于分析的日志数据，是日志转发器的默认输出。您可以使用默认的 LokiStack 日志存储、传统的 Elasticsearch 日志存储，或将日志转发到额外的外部日志存储。

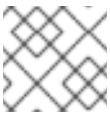


注意

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。如果您目前使用随 Logging 5.8 发布的 OpenShift Elasticsearch Operator，它将继续使用 Logging，直到 Logging 5.8 的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

视觉化

您可以使用 UI 组件查看日志数据的可视化表示。UI 提供了一个图形界面，用于搜索、查询和查看存储的日志。OpenShift Container Platform Web 控制台 UI 通过启用 OpenShift Container Platform 控制台插件来提供。



注意

Kibana Web 控制台现已弃用，计划在以后的日志记录发行版本中删除。

日志记录收集容器日志和节点日志。它们被归类为：

应用程序日志

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）。

基础架构日志

由基础架构命名空间生成的容器日志： **openshift***、**kube*** 或 **default**，以及来自节点的 journald 信息。

审计日志

由 auditd 生成的日志，节点审计系统存储在 **/var/log/audit/audit.log** 文件中，以及 **auditd**、**kube-apiserver**、**openshift-apiserver** 服务以及 **ovn** 项目（如果启用）中的日志。

其他资源

- 使用 Web 控制台进行日志视觉化

4.2. 关于部署日志记录

管理员可以使用 OpenShift Container Platform Web 控制台或 OpenShift CLI (**oc**) 来安装 logging Operator。Operator 负责部署、升级和维护日志记录。

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。

4.2.1. 日志记录自定义资源

您可以使用每个 Operator 实施的自定义资源 (CR) YAML 文件配置日志记录部署。

Red Hat OpenShift Logging Operator ：

- **ClusterLogging** (CL) - 安装 Operator 后，您可以创建一个 **ClusterLogging** 自定义资源 (CR) 来调度 logging pod 和支持 logging 所需的其他资源。**ClusterLogging** CR 部署收集器和转发器，当前都由每个节点上运行的 daemonset 实施。Red Hat OpenShift Logging Operator 会监视 **ClusterLogging** CR，并相应地调整日志记录部署。
- **ClusterLogForwarder** (CLF)- 生成收集器配置，以为每个用户配置转发日志。

Loki Operator ：

- **LokiStack** - 将 Loki 集群控制为日志存储，以及带有 OpenShift Container Platform 身份验证集成的 Web 代理，以强制实施多租户。

OpenShift Elasticsearch Operator ：

注意

这些 CR 由 OpenShift Elasticsearch Operator 生成和管理。在 Operator 被覆盖的情况下，无法进行手动更改。

- **Elasticsearch** - 配置和部署 Elasticsearch 实例作为默认日志存储。
- **Kibana** - 配置和部署 Kibana 实例以搜索、查询和查看日志。

4.2.2. 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

4.2.3. 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。

4.2.4. 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

4.2.5. 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。

4.2.6. 关于事件路由

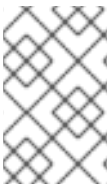
Event Router 是一个 pod，它监视 OpenShift Container Platform 事件，以便通过日志记录来收集这些事件。Event Router 从所有项目收集事件，并将其写入 **STDOUT**。Fluentd 收集这些事件并将其转发到 OpenShift Container Platform Elasticsearch 实例。Elasticsearch 将事件索引到 **infra** 索引。

您必须手动部署 Event Router。

如需更多信息，请参阅收集并存储 Kubernetes 事件。

第 5 章 安装日志记录

您可以通过安装 Red Hat OpenShift Logging Operator 来部署日志记录。Red Hat OpenShift Logging Operator 会创建和管理日志记录堆栈的组件。



注意

日志记录作为一个可安装的组件提供，它有一个不同于 OpenShift Container Platform 的发布周期。Red Hat OpenShift Container Platform 生命周期政策 概述了发行版本兼容性。

重要

对于新安装，使用 Vector 和 LokiStack。Elasticsearch 和 Fluentd 已被弃用，计划在以后的发行版本中删除。

5.1. 使用 WEB 控制台安装 RED HAT OPENSHIFT LOGGING OPERATOR

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

先决条件

- 有管理员权限。
- 访问 OpenShift Container Platform web 控制台。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 在 **Filter by keyword** 框中键入 **OpenShift Logging**。
3. 从可用的 Operator 列表中选择 **Red Hat OpenShift Logging**，然后点 **Install**。
4. 选择 **stable-5.y** 作为 **更新频道**。

注意

stable 频道只为日志记录的最新版本提供更新。要继续获得之前版本的更新，您必须将订阅频道改为 **stable-x.y**，其中 **x.y** 代表您安装的日志记录的主版本和次版本。例如，**stable-5.7**。

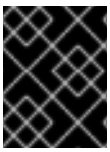
5. 选择 **版本**。
6. 确定在 **Installation mode** 下选择了 **A specific namespace on the cluster**。
7. 确定在 **Installed Namespace** 下的 **Operator recommended namespace** 是 **openshift-logging**。
8. 选择一个 **Update approval**。
   - **Automatic** 策略允许 Operator Lifecycle Manager（OLM）在有新版本可用时自动更新 Operator。

- **Manual** 策略需要拥有适当凭证的用户批准 Operator 更新。

9. 为 **Console 插件**选择 **Enable** 或 **Disable**。
10. 点 **Install**。

验证

1. 通过切换到 **Operators → Installed Operators** 页来验证 **Red Hat OpenShift Logging Operator** 是否已安装。
2. 在 **Status** 列中，验证您看到了绿色的对勾标记，并为 **InstallSucceeded**，文本 **Up to date**。



重要

Operator 可能会在安装完成前显示 **Failed** 状态。如果 Operator 安装完成并显示 **InstallSucceeded** 信息，请刷新页面。

如果 Operator 没有显示已安装状态，请选择以下故障排除选项之一：

- 进入 **Operators → Installed Operators** 页面，检查 **Status** 列中是否有任何错误或故障。
- 进入 **Workloads → Pods** 页面，检查 **openshift-logging** 项目中报告问题的 pod 的日志。

5.2. 使用 WEB 控制台创建 CLUSTERLOGGING 对象

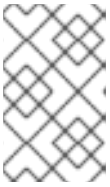
安装日志记录 Operator 后，您必须创建一个 **ClusterLogging** 自定义资源来为集群配置日志存储、视觉化和日志收集器。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 您可以访问 OpenShift Container Platform Web 控制台 **Administrator** 视角。

流程

1. 进入 **Custom Resource Definitions** 页面。
2. 在 **Custom Resource Definitions** 页面上，点 **ClusterLogging**。
3. 在 **Custom Resource Definition details** 页中，从 **Actions** 菜单中选择 **View Instances**。
4. 在 **ClusterLoggings** 页中，点 **Create ClusterLogging**。
5. 在 **collection** 部分中，选择一个 **Collector Implementation**。



注意

Fluentd 已被弃用，计划在以后的发行版本中删除。红帽将在当前发行生命周期中将提供对这个功能的 bug 修复和支持，但此功能将不再获得改进。作为 Fluentd 的替代选择，您可以使用 Vector。

6. 在 **logStore** 部分中，选择一个类型。

注意

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。如果您目前使用随 Logging 5.8 发布的 OpenShift Elasticsearch Operator，它将继续使用 Logging，直到 Logging 5.8 的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

7. 点 **Create**。

5.3. 使用 CLI 安装 RED HAT OPENSHIFT LOGGING OPERATOR

您可以使用 OpenShift CLI (**oc**) 安装 Red Hat OpenShift Logging Operator。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 创建一个 **Namespace** 对象作为一个 YAML 文件：

   Namespace 对象示例

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

   1 您必须将 **openshift-logging** 指定为日志记录版本 5.7 及更早的版本的命名空间名称。对于日志记录 5.8 及更新的版本，您可以使用任何名称。

2. 运行以下命令来应用 **Namespace** 对象：

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

3. 以 YAML 文件形式创建 **OperatorGroup** 对象：

   OperatorGroup 对象示例

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

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

(1) (2) 您必须为日志记录版本 5.7 及更早的版本指定 **openshift-logging** 命名空间。对于日志记录 5.8 及更新的版本，您可以使用任何命名空间。

4. 运行以下命令来应用 **OperatorGroup** 对象：

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

5. 创建一个 **Subscription** 对象来订阅 Red Hat OpenShift Logging Operator 的命名空间：

Subscription 对象示例

```
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) 您必须为日志记录版本 5.7 及更早的版本指定 **openshift-logging** 命名空间。对于日志记录 5.8 及更新的版本，您可以使用任何命名空间。

(2) 指定 **stable** 或 **stable-x.y** 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。

6. 运行以下命令来应用订阅：

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

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

验证

1. 运行以下命令:

```
$ oc get csv -n <namespace>
```

2. 观察输出，并确认命名空间中存在 Red Hat OpenShift Logging Operator：

输出示例

```
NAMESPACE                                       NAME                                         DISPLAY
VERSION             REPLACES   PHASE
...
openshift-logging                               clusterlogging.5.8.0-202007012112.p0
OpenShift Logging         5.8.0-202007012112.p0             Succeeded
...
```

5.4. 使用 CLI 创建 CLUSTERLOGGING 对象

此默认日志记录配置支持广泛的环境。参阅有关调优和配置组件的主题，以了解有关您可以进行的修改的信息。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 您已为日志存储安装了 OpenShift Elasticsearch Operator。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 将 **ClusterLogging** 对象创建为 YAML 文件：

   ClusterLogging 对象示例

```
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:
    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）。其他 Elasticsearch 节点使用客户端和数据角色作为仅数据节点创建。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

```
NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
cluster-logging-operator        1/1     1            1           18h
elasticsearch-cd-x6kdekli-1     1/1     1            1           6m54s
elasticsearch-cdm-x6kdekli-1    1/1     1            1           18h
elasticsearch-cdm-x6kdekli-2    1/1     1            1           6m49s
elasticsearch-cdm-x6kdekli-3    1/1     1            1           6m44s
```

索引模板的主分片数量等于 Elasticsearch 数据节点的数目。

验证

您可以通过列出 **openshift-logging** 项目中的 pod 来验证安装。

- 运行以下命令列出 pod：

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

  观察日志组件的 pod，类似于以下列表：

  输出示例

  ```
  NAME                                        READY   STATUS    RESTARTS   AGE
  cluster-logging-operator-66f77ffccb-ppzbg   1/1     Running   0          7m
  elasticsearch-cdm-ftuhduuw-1-ffc4b9566-q6bhp   2/2     Running   0          2m40s
  elasticsearch-cdm-ftuhduuw-2-7b4994dbfc-rd2gc   2/2     Running   0          2m36s
  elasticsearch-cdm-ftuhduuw-3-84b5ff7ff8-gqnm2   2/2     Running   0          2m4s
  collector-587vb                             1/1     Running   0          2m26s
  collector-7mpb9                             1/1     Running   0          2m30s
  collector-flm6j                             1/1     Running   0          2m33s
  collector-gn4rn                             1/1     Running   0          2m26s
  collector-nlgb6                             1/1     Running   0          2m30s
  collector-snpkt                             1/1     Running   0          2m28s
  kibana-d6d5668c5-rppqm                      2/2     Running   0          2m39s
  ```

5.5. 安装后的任务

安装 Red Hat OpenShift Logging Operator 后，您可以通过创建和修改 **ClusterLogging** 自定义资源(CR)来配置部署。

提示

如果没有使用 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。请参阅如果没有使用 Elasticsearch 日志存储，删除未使用的组件。

5.5.1. 关于 ClusterLogging 自定义资源

要更改日志记录环境，请创建并修改 **ClusterLogging** 自定义资源（CR）。

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
# ...
```

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。当状态设置为 **unmanaged** 时，Operator 处于不支持的状态，且不会接收更新。

5.5.2. 配置日志存储

您可以通过修改 **ClusterLogging** 自定义资源(CR)来配置日志使用的日志存储类型。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 Red Hat OpenShift Logging Operator 和一个内部日志存储，它是 LokiStack 或 Elasticsearch。
- 您已创建了 **ClusterLogging** CR。

注意

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。如果您目前使用随 Logging 5.8 发布的 OpenShift Elasticsearch Operator，它将继续使用 Logging，直到 Logging 5.8 的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

流程

1. 修改 **ClusterLogging** CR **logStore** 规格：

ClusterLogging CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  logStore:
    type: <log_store_type> 1
    elasticsearch: 2
      nodeCount: <integer>
      resources: {}
      storage: {}
      redundancyPolicy: <redundancy_type> 3
    lokistack: 4
      name: {}
# ...
```

1. 指定日志存储类型。这可以是 **lokistack** 或 **elasticsearch**。
2. Elasticsearch 日志存储的可选配置选项。
3. 指定冗余类型。这个值可以是 **ZeroRedundancy**、**SingleRedundancy**、**MultipleRedundancy** 或 **FullRedundancy**。
4. LokiStack 的可选配置选项。

将 LokiStack 指定为日志存储的 ClusterLogging CR 示例

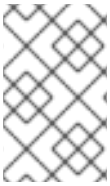
```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
    lokistack:
      name: logging-loki
# ...
```

2. 运行以下命令来应用 **ClusterLogging** CR：

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

5.5.3. 配置日志收集器

您可以通过修改 **ClusterLogging** 自定义资源(CR)来配置日志使用哪个日志收集器类型。



注意

Fluentd 已被弃用，计划在以后的发行版本中删除。红帽将在当前发行生命周期中将提供对这个功能的 bug 修复和支持，但此功能将不再获得改进。作为 Fluentd 的替代选择，您可以使用 Vector。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 Red Hat OpenShift Logging Operator。
- 您已创建了 **ClusterLogging** CR。

流程

1. 修改 **ClusterLogging** CR **collection** 规格：

   ClusterLogging CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  collection:
    type: <log_collector_type> 1
    resources: {}
    tolerations: {}
# ...
```

   1 要用于日志记录的日志收集器类型。这可以是 **vector** 或 **fluentd**。

2. 运行以下命令来应用 **ClusterLogging** CR：

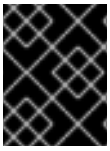
```
$ oc apply -f <filename>.yaml
```

5.5.4. 配置日志可视化工具

您可以通过修改 **ClusterLogging** 自定义资源(CR)来配置日志可视化工具类型。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 Red Hat OpenShift Logging Operator。
- 您已创建了 **ClusterLogging** CR。



重要

如果要使用 OpenShift Container Platform Web 控制台进行视觉化，您必须启用日志记录控制台插件。请参阅有关 "Log visualization with the web console" 的文档。

流程

1. 修改 **ClusterLogging** CR **visualization** 规格：

ClusterLogging CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  visualization:
    type: <visualizer_type> 1
    kibana: 2
      resources: {}
      nodeSelector: {}
      proxy: {}
      replicas: {}
      tolerations: {}
    ocpConsole: 3
      logsLimit: {}
      timeout: {}
# ...
```

1. 要用于日志记录的可视化工具类型。这可以是 **kibana** 或 **ocp-console**。Kibana 控制台仅与使用 Elasticsearch 日志存储的部署兼容，而 OpenShift Container Platform 控制台只与 LokiStack 部署兼容。
2. Kibana 控制台的可选配置。
3. OpenShift Container Platform Web 控制台的可选配置。

2. 运行以下命令来应用 **ClusterLogging** CR：

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

5.5.5. 启用网络隔离时允许项目间的流量

集群网络插件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。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 为网络插件(OpenShift SDN 和 OVN-Kubernetes)提供了两个支持的选择。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，它允许从 **openshift-operators-redhat**、**openshift-monitoring** 和 **openshift-ingress** 项目的入站流量到 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
- 关于 OVN-Kubernetes 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 6 章 更新日志记录

有两种日志记录更新： 次版本更新(5.y.z)和主版本更新(5.y)。

6.1. 次发行版本更新

如果您使用 **Automatic** 更新批准选项安装日志记录 Operator，您的 Operator 会自动接收次版本更新。您不需要完成任何手动更新步骤。

如果使用 **Manual** 更新批准选项安装日志记录 Operator，您必须手动批准次版本更新。如需更多信息，请参阅 手动批准待处理的 Operator 更新。

6.2. 主发行版本更新

对于主版本更新，您必须完成一些手动步骤。

有关主版本的兼容性和支持信息，请参阅 OpenShift Operator 生命周期。

6.3. 升级 RED HAT OPENSHIFT LOGGING OPERATOR 以监视所有命名空间

在日志记录 5.7 和旧版本中，Red Hat OpenShift Logging Operator 只监视 **openshift-logging** 命名空间。如果您希望 Red Hat OpenShift Logging Operator 监视集群中的所有命名空间，您必须重新部署 Operator。您可以完成以下步骤在不删除日志记录组件的情况下重新部署 Operator。

先决条件

- 已安装 OpenShift CLI（**oc**）。
- 有管理员权限。

流程

1. 运行以下命令来删除订阅：

```
$ oc -n openshift-logging delete subscription <subscription>
```

2. 运行以下命令来删除 Operator 组：

```
$ oc -n openshift-logging delete operatorgroup <operator_group_name>
```

3. 运行以下命令来删除集群服务版本 (CSV)：

```
$ oc delete clusterserviceversion cluster-logging.<version>
```

4. 按照"安装日志记录"文档重新部署 Red Hat OpenShift Logging Operator。

验证

- 检查 **OperatorGroup** 资源中的 **targetNamespaces** 字段是否不存在或设置为空字符串。要做到这一点，请运行以下命令并检查输出：

```
$ oc get operatorgroup <operator_group_name> -o yaml
```

输出示例

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-logging-f52cn
  namespace: openshift-logging
spec:
  upgradeStrategy: Default
status:
  namespaces:
  - ""
# ...
```

6.4. 更新 RED HAT OPENSHIFT LOGGING OPERATOR

要将 Red Hat OpenShift Logging Operator 更新至新的主版本，您必须修改 Operator 订阅的更新频道。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 有管理员权限。
- 您可以访问 OpenShift Container Platform Web 控制台，并查看 **Administrator** 视角。

流程

1. 导航到 **Operators → Installed Operators**。
2. 选择 **openshift-logging** 项目。
3. 点 **Red Hat OpenShift Logging** Operator。
4. 点 **Subscription**。在 **Subscription details** 部分，点 **Update channel** 链接。根据您的当前更新频道，这个链接文本可能是 **stable** 或 **stable-5.y**。
5. 在 **Change Subscription Update Channel** 窗口中，选择最新的主版本更新频道 **stable-5.y**，然后点 **Save**。请注意 **cluster-logging.v5.y.z** 版本。

验证

1. 等待几秒钟，然后点 **Operators → Installed Operators**。验证 Red Hat OpenShift Logging Operator 版本是否与最新的 **cluster-logging.v5.y.z** 版本匹配。
2. 在 **Operators → Installed Operators** 页面中，等待 **Status** 字段报告 **Succeeded**。

6.5. 更新 LOKI OPERATOR

要将 Loki Operator 更新至一个新的主版本，您必须修改 Operator 订阅的更新频道。

先决条件

- 已安装 Loki Operator。
- 有管理员权限。
- 您可以访问 OpenShift Container Platform Web 控制台，并查看 **Administrator** 视角。

流程

1. 导航到 **Operators → Installed Operators**。
2. 选择 **openshift-operators-redhat** 项目。
3. 点 **Loki Operator**。
4. 点 **Subscription**。在 **Subscription details** 部分，点 **Update channel** 链接。根据您的当前更新频道，这个链接文本可能是 **stable** 或 **stable-5.y**。
5. 在 **Change Subscription Update Channel** 窗口中，选择最新的主版本更新频道 **stable-5.y**，然后点 **Save**。请注意 **loki-operator.v5.y.z** 版本。

验证

1. 等待几秒钟，然后点 **Operators → Installed Operators**。验证 Loki Operator 版本是否与最新的 **loki-operator.v5.y.z** 版本匹配。
2. 在 **Operators → Installed Operators** 页面中，等待 **Status** 字段报告 **Succeeded**。

6.6. 更新 OPENSHIFT ELASTICSEARCH OPERATOR

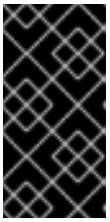
要将 OpenShift Elasticsearch Operator 更新至当前版本，您必须修改订阅。

注意

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。如果您目前使用随 Logging 5.8 发布的 OpenShift Elasticsearch Operator，它将继续使用 Logging，直到 Logging 5.8 的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

先决条件

- 如果您使用 Elasticsearch 作为默认日志存储，且 Kibana 作为 UI，请在更新 Red Hat OpenShift Logging Operator 前更新 OpenShift Elasticsearch Operator。



重要

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)。要解决这个问题，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。

- Logging 处于健康状态：

- 所有 pod 都处于 **ready** 状态。
- Elasticsearch 集群处于健康状态。

- 您的 Elasticsearch 和 Kibana 数据已被备份。
- 有管理员权限。
- 您已安装了 OpenShift CLI (**oc**) 进行验证步骤。

流程

1. 在 OpenShift Container Platform web 控制台中，点击 **Operators → Installed Operators**。
2. 选择 **openshift-operators-redhat** 项目。
3. 点 **OpenShift Elasticsearch Operator**。
4. 点 **Subscription → Channel**。
5. 在 **Change Subscription Update Channel**窗口中，选择 **stable-5.y** 并点 **Save**。注意 **elasticsearch-operator.v5.y.z** 版本。
6. 等待几秒钟，然后点 **Operators → Installed Operators**。验证 OpenShift Elasticsearch Operator 版本是否与最新的 **elasticsearch-operator.v5.y.z** 版本匹配。
7. 在 **Operators → Installed Operators** 页面中，等待 **Status** 字段报告 **Succeeded**。

验证

1. 输入以下命令并查看输出，验证所有 Elasticsearch pod 的状态是否为 **Ready** ：

```
$ 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
```

2. 输入以下命令并查看输出来验证 Elasticsearch 集群状态是否为**绿色** ：

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

输出示例

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

3. 输入以下命令并查看输出来验证 Elasticsearch cron 作业是否已创建：

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

```
$ oc get cronjob
```

输出示例

```
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
```

4. 输入以下命令并验证日志存储是否已更新至正确的版本，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 6.1. 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                                                1SJdCqlZTPWlIAaOUd78yg
1 1          4            0         0            0
green  open   infra-000010
iXwL3bnqTuGEABbUDa6OVw   3 1      248368            0       317          158
green  open   infra-000009
YN9EsULWSNaxWeeNvOs0RA   3 1      258799            0       337          168
green  open   infra-000014
YP0U6R7FQ_GVQVQZ6Yh9Ig   3 1      223788            0       292          146
green  open   infra-000015
JRBbAbEmSMqK5X40df9HbQ   3 1      224371            0       291          145
green  open   .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA   3 1          9            0         0            0
green  open   infra-000007
llkkAVSzSOmosWTSAJM_hg   3 1      228584            0       296          148
green  open   infra-000005
d9BoGQdiQASsS3BBFm2iRA   3 1      227987            0       297          148
green  open   infra-000003                                              1-
goREK1QUKlQPAIVkWVaQ   3 1      226719            0       295          147
green  open   .security                                                 zeT65uOuRTKZMjg_bbUc1g
1 1          5            0         0            0
green  open   .kibana-377444158_kubeadmin                                wvMhDwJkR-
mRZQO84K0gUQ   3 1          1            0         0            0
green  open   infra-000006                                               5H-
KBSXGQKiO7hdapDE23g   3 1      226676            0       295          147
green  open   infra-000001                                             eH53BQ-
bSxSWR5xYZB6lVg   3 1      341800            0       443          220
```

```
green  open   .kibana-6
RVp7TemSSemGJcsSUmuf3A   1 1          4          0        0            0
green  open   infra-000011
J7XWBauWSTe0jnzX02fU6A   3 1     226100          0      293          146
green  open   app-000001
axSAFfONQDmKwatkjPXdtw   3 1     103186          0      126           57
green  open   infra-000016
m9c1iRLtStWSF1GopaRyCg   3 1      13685          0       19            9
green  open   infra-000002                                    Hz6WvINtTvKcQzw-
ewmbYg   3 1     228994          0      296          148
green  open   infra-000013                                    KR9mMFUpQl-
jraYtanyIGw   3 1     228166          0      298          148
green  open   audit-000001
eERqLdLmQOiQDFES1LBATQ   3 1          0          0        0            0
```

5. 输入以下命令并查看输出，验证日志可视化工具是否已更新至正确的版本：

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

验证输出是否包含具有 **ready** 状态的 Kibana Pod：

例 6.2. 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
}
]
```

-

第 7 章 可视化日志

7.1. 关于日志视觉化

您可以根据部署的日志存储解决方案，在 OpenShift Container Platform Web 控制台中视觉化您的日志数据，或 Kibana Web 控制台。Kibana 控制台可用于 ElasticSearch 日志存储，OpenShift Container Platform Web 控制台可用于 ElasticSearch 日志存储或 LokiStack。

注意

Kibana Web 控制台现已弃用，计划在以后的日志记录发行版本中删除。

7.1.1. 配置日志可视化工具

您可以通过修改 **ClusterLogging** 自定义资源(CR)来配置日志可视化工具类型。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 Red Hat OpenShift Logging Operator。
- 您已创建了 **ClusterLogging** CR。

重要

如果要使用 OpenShift Container Platform Web 控制台进行视觉化，您必须启用日志记录控制台插件。请参阅有关 "Log visualization with the web console" 的文档。

流程

1. 修改 **ClusterLogging** CR **visualization** 规格：

   ClusterLogging CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  visualization:
    type: <visualizer_type> 1
    kibana: 2
      resources: {}
      nodeSelector: {}
      proxy: {}
      replicas: {}
      tolerations: {}
    ocpConsole: 3
```

```
      logsLimit: {}
      timeout: {}
# ...
```

1. 要用于日志记录的可视化工具类型。这可以是 **kibana** 或 **ocp-console**。Kibana 控制台仅与使用 Elasticsearch 日志存储的部署兼容，而 OpenShift Container Platform 控制台只与 LokiStack 部署兼容。
2. Kibana 控制台的可选配置。
3. OpenShift Container Platform Web 控制台的可选配置。

2. 运行以下命令来应用 **ClusterLogging** CR：

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

7.1.2. 查看资源的日志

资源日志是一个默认功能，可提供有限的日志查看功能。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。

提示

为增强日志检索和查看体验，请安装 logging。日志记录将 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 控制台或 OpenShift Container Platform Web 控制台查询、发现和视觉化您的日志数据。资源日志无法访问日志记录日志存储。

7.1.2.1. 查看资源日志

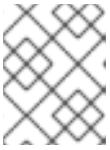
您可以在 OpenShift CLI (**oc**) 和 Web 控制台中查看各种资源的日志。日志从日志的尾部或末尾读取。

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 **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。

7.2. 使用 WEB 控制台进行日志视觉化

您可以通过配置 logging 控制台插件，使用 OpenShift Container Platform Web 控制台来视觉化日志数据。

有关在日志记录安装过程中配置插件的详情，请参考使用 Web 控制台安装 logging。

如果您已经安装了日志记录并希望配置插件，请使用以下步骤之一。

7.2.1. 安装 Red Hat OpenShift Logging Operator 后启用 logging 控制台插件

作为 Red Hat OpenShift Logging Operator 安装的一部分，您可以启用 logging 控制台插件，但如果您已禁用了插件，也可以启用插件。

先决条件

- 有管理员权限。
- 已安装 Red Hat OpenShift Logging Operator，并为 **Console 插件**选择了 **Disabled**。
- 访问 OpenShift Container Platform web 控制台。

流程

1. 在 OpenShift Container Platform web 控制台的 **Administrator** 视角中，进入 **Operators → Installed Operators**。
2. 点 **Red Hat OpenShift Logging**。这会进入 Operator **Details** 页面。
3. 在 **Details** 页面中，为 **控制台插件**选项点 **Disabled**。
4. 在**控制台插件启用**对话框中，选择 **Enable**。
5. 点击 **Save**。
6. 验证 **控制台插件**选项现在显示 **Enabled**。
7. 应用更改后，web 控制台会显示一个弹出窗口。窗口提示您重新加载 Web 控制台。当您看到弹出窗口以应用更改时，刷新浏览器。

7.2.2. 安装 Elasticsearch 日志存储和 LokiStack 时配置日志记录控制台插件

在 logging 版本 5.8 及更高版本中，如果 Elasticsearch 日志存储是默认的日志存储，您也可以按照以下流程启用 logging 控制台插件。

先决条件

- 有管理员权限。
- 已安装 Red Hat OpenShift Logging Operator、OpenShift Elasticsearch Operator 和 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。
- 您已创建了 **ClusterLogging** 自定义资源 (CR)。

流程

1. 运行以下命令，确保启用了日志记录控制台插件：

```
$ oc get consoles.operator.openshift.io cluster -o yaml |grep logging-view-plugin  \
|| oc patch consoles.operator.openshift.io cluster  --type=merge \
--patch '{ "spec": { "plugins": ["logging-view-plugin"]}}'
```

2. 运行以下命令，将 **.metadata.annotations.logging.openshift.io/ocp-console-migration-target: lokistack-dev** 注解添加到 **ClusterLogging** CR：

```
$ oc patch clusterlogging instance --type=merge --patch \
'{ "metadata": { "annotations": { "logging.openshift.io/ocp-console-migration-target":
"lokistack-dev" }}}' \
-n openshift-logging
```

输出示例

```
clusterlogging.logging.openshift.io/instance patched
```

验证

- 运行以下命令并查看输出，验证注解是否已成功添加：

```
$ oc get clusterlogging instance \
-o=jsonpath='{.metadata.annotations.logging\.openshift\.io/ocp-console-migration-target}' \
-n openshift-logging
```

输出示例

```
"lokistack-dev"
```

现在，日志记录控制台插件 pod 已被部署。您可以通过进入到 OpenShift Container Platform Web 控制台并查看 **Observe → Logs** 页面来查看日志数据。

7.3. 查看集群仪表板

OpenShift Container Platform web 控制台中的 **Logging/Elasticsearch Nodes** 和 **Openshift Logging** 仪表板包括有关 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。

7.3.1. 访问 Elasticsearch 和 OpenShift 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 **Observe → 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。

有关仪表板图表的信息，请参阅 关于 OpenShift Logging 仪表板 和 关于 Logging/Elastisearch Nodes 仪表板。

7.3.2. 关于 OpenShift 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 7.1. OpenShift Logging chart

指标	描述
Elastic 集群状态	当前的 Elasticsearch 状态： • ONLINE - 表示 Elasticsearch 实例在线。 • OFFLINE - 表示 Elasticsearch 实例离线。
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 待处理的任务	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	• Query latency:Elasticsearch 搜索查询执行的平均时间。 • 获取延迟：每个 Elasticsearch 搜索查询的平均时间获取数据。 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。
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。

指标	描述
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。
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 缓冲使用	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。

7.3.3. 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 7.2. Elasticsearch 状态字段

指标	描述
集群状态	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： • 0 - 表示 Elasticsearch 实例处于绿色状态，这意味着分配了所有分片。 • 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。 • 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。
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。

指标	描述
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。索引分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 7.3. 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 7.4. Elasticsearch 节点指标图表

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 7.5. Elasticsearch 节点字段数据图表

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 7.6. 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 7.7. 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 7.8. 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。

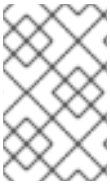
7.4. 使用 KIBANA 进行日志视觉化

如果使用 ElasticSearch 日志存储，您可以使用 Kibana 控制台来视觉化收集的日志数据。

使用 Kibana，您可以使用您的数据进行以下操作：

- 使用 **Discover** 标签页搜索并浏览数据。
- 使用 **Visualize** 选项卡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。有关使用接口的更多信息，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。

7.4.1. 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 中探索和视觉化数据，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 --subresource 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视觉化：

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。

7.4.2. 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 Red Hat OpenShift Logging 和 Elasticsearch Operator。
- 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 --subresource 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 7.1. Kibana 中的基础架构日志条目示例


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

```
        "version": "1.7.4 1.6.0"
      }
    },
    "@timestamp": "2020-09-23T20:47:03.422465+00:00",
    "viaq_msg_id": "YmJmYTBlNDktMDMGQtMjE3NmFiOGUyOWM3",
    "openshift": {
      "labels": {
        "logging": "infra"
      }
    }
  },
  "fields": {
    "@timestamp": [
      "2020-09-23T20:47:03.422Z"
    ],
    "pipeline_metadata.collector.received_at": [
      "2020-09-23T20:47:15.007Z"
    ]
  },
  "sort": [
    1600894023422
  ]
}
```


7.4.3. 配置 Kibana

您可以通过修改 **ClusterLogging** 自定义资源(CR) 来使用 Kibana 控制台配置。

7.4.3.1. 配置 CPU 和内存限值

日志记录组件允许对 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 和内存限值和请求。

4 根据需要指定日志收集器的 CPU 和内存限值及请求。

7.4.3.2. 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。

第 8 章 配置日志部署

8.1. 为日志记录组件配置 CPU 和内存限值

您可以根据需要配置每个日志记录组件的 CPU 和内存限值。

8.1.1. 配置 CPU 和内存限值

日志记录组件允许对 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 和内存限值和请求。

4 根据需要指定日志收集器的 CPU 和内存限值及请求。

8.2. 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。

8.2.1. 为 OpenShift Logging 配置 systemd-journald

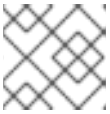
随着项目的扩展，默认的日志记录环境可能需要进行一些调整。

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. 创建一个 Butane 配置文件 **40-worker-custom-journald.bu**，其中包含带有所需设置的 **/etc/systemd/journald.conf** 文件。



注意

有关 Butane 的信息，请参阅"使用 Butane 创建机器配置"。

```
variant: openshift
version: 4.16.0
metadata:
  name: 40-worker-custom-journald
  labels:
```

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

(1) 设置 **journald.conf** 文件的权限。建议把选项设置为 **0644**。

(2) 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**。

(3) 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。
- **ForwardToKMsg** 将日志转发到内核日志缓冲区。
- **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。

(4) 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**。

(5) 配置速率限制。如果在 **RateLimitIntervalSec** 定义的时间间隔内收到 **RateLimitBurst** 中指定的日志数，则该时间段内的所有进一步信息都会被丢弃，直到间隔结束。建议您设置 **RateLimitIntervalSec=30s** 和 **RateLimitBurst=10000**，它们是默认值。

(6) 指定日志的存储方式。默认为 **persistent**：

- **volatile**，将日志存储在 **/run/log/journal/** 中的内存中。这些日志在重启后会丢失。
- **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** 将日志存储在 **/var/log/journal/** 中（如果存在这个目录）。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。

(7) 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**。

8 指定日志可以使用的最大值。默认值为 **8G**。

9 指定 systemd 必须保留多少磁盘空间。默认值为 **20%**。

10 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. 使用 Butane 生成 **MachineConfig** 对象文件 **40-worker-custom-journald.yaml**，它包含要提供给节点的配置：

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

3. 应用机器配置。例如：

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

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

4. 监控新配置在每个节点中的应用状态：

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

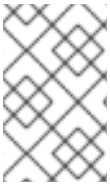
输出示例

```
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
```

第 9 章 日志收集和转发

9.1. 关于日志收集和转发

Red Hat OpenShift Logging Operator 根据 **ClusterLogForwarder** 资源规格部署一个收集器。此 Operator 支持两个收集器选项：旧的 Fluentd 收集器和 Vector 收集器。



注意

Fluentd 已被弃用，计划在以后的发行版本中删除。红帽将在当前发行生命周期中将提供对这个功能的 bug 修复和支持，但此功能将不再获得改进。作为 Fluentd 的替代选择，您可以使用 Vector。

9.1.1. 日志集合

日志收集器是一个守护进程集，它将 Pod 部署到每个 OpenShift Container Platform 节点，以收集容器和节点日志。

默认情况下，日志收集器使用以下源：

- 由来自操作系统、容器运行时和 OpenShift Container Platform 的 journald 日志消息生成的系统和基础架构日志。
- **/var/log/containers/*.log** 用于所有容器日志

如果您将日志收集器配置为收集审计日志，它会从 **/var/log/audit/audit.log** 收集它们。

日志收集器从这些源收集日志，并根据日志记录配置在内部或外部转发它们。

9.1.1.1. 日志收集器类型

Vector 是一个日志收集器，作为日志记录的 Fluentd 的一个替代方案。

您可以通过修改 **ClusterLogging** 自定义资源(CR) **collection** 规格来配置集群使用的日志记录收集器类型：

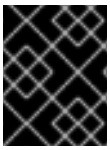
将 Vector 配置为收集器的 ClusterLogging CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  collection:
    logs:
      type: vector
      vector: {}
# ...
```

9.1.1.2. 日志收集限制

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。这种限制意味着日志收集和规范化被视为 *最佳工作*。



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。

9.1.1.3. 按类型划分的日志收集器功能

表 9.1. 日志源

功能	Fluentd	Vector
应用程序容器日志	✓	✓
特定于应用程序的路由	✓	✓
命名空间划分应用程序特定路由	✓	✓
Infra 容器日志	✓	✓
Infra 日志	✓	✓
kube API 审计日志	✓	✓
OpenShift API 审计日志	✓	✓
打开虚拟网络 (OVN) 审计日志	✓	✓

表 9.2. 授权和身份验证

功能	Fluentd	Vector
Elasticsearch 证书	✓	✓
Elasticsearch 用户名/密码	✓	✓
Amazon Cloudwatch 密钥	✓	✓
Amazon Cloudwatch STS	✓	✓
Kafka 证书	✓	✓
Kafka 用户名/密码	✓	✓
Kafka SASL	✓	✓

功能	Fluentd	Vector
Loki bearer 令牌	✓	✓

表 9.3. 规范化和转换

功能	Fluentd	Vector
ViaQ 数据模型 - 应用程序	✓	✓
ViaQ 数据模型 - infra	✓	✓
ViaQ 数据模型 - infra(journal)	✓	✓
ViaQ 数据模型 - Linux 审计	✓	✓
ViaQ 数据模型 - kube-apiserver 审计	✓	✓
ViaQ 数据模型 - OpenShift API 审计	✓	✓
ViaQ 数据模型 - OVN	✓	✓
loglevel Normalization	✓	✓
JSON 解析	✓	✓
结构化索引	✓	✓
多行错误检测	✓	✓
multicontainer/ split 索引	✓	✓
Flatten 标签	✓	✓
CLF 静态标签	✓	✓

表 9.4. Tuning

功能	Fluentd	Vector
Fluentd readlinelimit	✓	
Fluentd 缓冲	✓	

功能	Fluentd	Vector
- chunklimitsize	✓	
- totallimitsize	✓	
- overflowaction	✓	
- flushthreadcount	✓	
- flushmode	✓	
- flushinterval	✓	
- retrywait	✓	
- retrytype	✓	
- retrymaxinterval	✓	
- retrytimeout	✓	

表 9.5. 可见性

功能	Fluentd	Vector
指标	✓	✓
Dashboard	✓	✓
警报	✓	✓

表 9.6. 其它

功能	Fluentd	Vector
全局代理支持	✓	✓
x86 支持	✓	✓
ARM 支持	✓	✓
IBM Power® 支持	✓	✓
IBM Z® 支持	✓	✓

功能	Fluentd	Vector
IPv6 支持	✓	✓
日志事件缓冲	✓	
断开连接的集群	✓	✓

9.1.1.4. 收集器输出

支持以下收集器输出：

表 9.7. 支持的输出

功能	Fluentd	Vector
Elasticsearch v6-v8	✓	✓
Fluent 转发	✓	
Syslog RFC3164	✓	✓ (Logging 5.7+)
Syslog RFC5424	✓	✓ (Logging 5.7+)
Kafka	✓	✓
Amazon Cloudwatch	✓	✓
Amazon Cloudwatch STS	✓	✓
Loki	✓	✓
HTTP	✓	✓ (Logging 5.7+)
Google Cloud Logging	✓	✓
Splunk		✓ (Logging 5.6+)

9.1.2. 日志转发

管理员可以创建 **ClusterLogForwarder** 资源，以指定要收集哪些日志、它们的转换方式以及它们被转发到的位置。

ClusterLogForwarder 资源可用于将容器、基础架构和审计日志转发到集群内部或外部的特定端点。支持传输层安全性(TLS)，以便可以配置日志转发来安全地发送日志。

管理员也可以授权 RBAC 权限来定义哪些服务帐户和用户可以访问和转发哪些日志类型。

9.1.2.1. 日志转发实现

可用的日志转发实现有两个：旧的实现和多日志转发器功能。

重要

仅支持 Vector 收集器与多日志转发器功能一起使用。Fluentd 收集器只能用于旧的实现。

9.1.2.1.1. 旧实施

在旧的实现中，集群中只能使用一个日志转发器。此模式的 **ClusterLogForwarder** 资源必须命名为 **instance**，且必须在 **openshift-logging** 命名空间中创建。**ClusterLogForwarder** 资源还需要 **openshift-logging** 命名空间中名为 **instance** 的对应 **ClusterLogging** 资源。

9.1.2.1.2. 多日志转发器功能

日志记录 5.8 及更高版本中提供了多日志转发器功能，并提供以下功能：

- 管理员可以控制哪些用户被允许定义日志收集以及允许收集哪些日志。
- 具有所需权限的用户可以指定额外的日志收集配置。
- 从已弃用的 Fluentd 收集器迁移到 Vector 收集器的管理员可以独立于现有部署部署新的日志转发程序。在迁移工作负载时，现有和新的日志转发程序可以同时运行。

在多日志转发器实现中，您不需要为 **ClusterLogForwarder** 资源创建对应的 **ClusterLogging** 资源。您可以使用任何命名空间中的任何名称创建多个 **ClusterLogForwarder** 资源，但以下例外：

- 您无法在 **openshift-logging** 命名空间中创建一个名为 **instance** 的 **ClusterLogForwarder** 资源，因为它为支持使用 Fluentd 收集器的传统工作流的日志转发器保留。
- 您无法在 **openshift-logging** 命名空间中创建一个名为 **collector** 的 **ClusterLogForwarder** 资源，因为这为收集器保留。

9.1.2.2. 为集群启用多日志转发器功能

要使用多日志转发器功能，您必须为该服务帐户创建服务帐户和集群角色绑定。然后，您可以在 **ClusterLogForwarder** 资源中引用服务帐户来控制访问权限。

重要

要在 **openshift-logging** 命名空间以外的额外命名空间中支持多日志转发功能，您必须更新 Red Hat OpenShift Logging Operator 以监视所有命名空间。在新的 Red Hat OpenShift Logging Operator 版本 5.8 版本中默认支持此功能。

9.1.2.2.1. 授权日志收集 RBAC 权限

在日志记录 5.8 及更高版本中，Red Hat OpenShift Logging Operator 提供了 **collect-audit-logs**、**collect-application-logs** 和 **collect-infrastructure-logs** 集群角色，该角色可让收集器分别收集审计日志、应用程序日志和基础架构日志。

您可以通过将所需的集群角色绑定到服务帐户来授权日志收集的 RBAC 权限。

先决条件

- Red Hat OpenShift Logging Operator 安装在 **openshift-logging** 命名空间中。
- 有管理员权限。

流程

1. 为收集器创建服务帐户。如果要将日志写入需要令牌进行身份验证的存储，则必须在服务帐户中包含令牌。
2. 将适当的集群角色绑定到服务帐户：

绑定命令示例

```
$ oc adm policy add-cluster-role-to-user <cluster_role_name> system:serviceaccount:
<namespace_name>:<service_account_name>
```

其他资源

- 使用 RBAC 定义和应用权限
- 在应用程序中使用服务帐户
- 使用 RBAC 授权 Kubernetes 文档

9.2. 日志输出类型

输出 (output) 定义了日志转发器将日志发送到的目的地。您可以在 **ClusterLogForwarder** 自定义资源 (CR) 中配置多种输出类型，将日志发送到支持不同协议的服务器。

9.2.1. 支持的日志转发输出

输出可以是以下任意类型：

表 9.8. 支持的日志输出类型

输出类型	协议	测试使用	日志记录版本	支持的收集器类型
Elasticsearch v6	HTTP 1.1	6.8.1, 6.8.23	5.6+	Fluentd, Vector
Elasticsearch v7	HTTP 1.1	7.12.2, 7.17.7, 7.10.1	5.6+	Fluentd, Vector
Elasticsearch v8	HTTP 1.1	8.4.3, 8.6.1	5.6+	Fluentd [1], Vector
Fluent Forward	Fluentd forward v1	Fluentd 1.14.6, Logstash 7.10.1, Fluentd 1.14.5	5.4+	Fluentd
Google Cloud Logging	通过 HTTPS 的 REST	Latest	5.7+	Vector

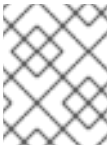
输出类型	协议	测试使用	日志记录版本	支持的收集器类型
HTTP	HTTP 1.1	Fluentd 1.14.6, Vector 0.21	5.7+	Fluentd, Vector
Kafka	Kafka 0.11	Kafka 2.4.1, 2.7.0, 3.3.1	5.4+	Fluentd, Vector
Loki	使用 HTTP 和 HTTPS 的 REST	2.3.0, 2.5.0, 2.7, 2.2.1	5.4+	Fluentd, Vector
Splunk	HEC	8.2.9, 9.0.0	5.7+	Vector
Syslog	RFC3164, RFC5424	rsyslog 8.37.0-9.el7, rsyslog-8.39.0	5.4+	Fluentd, Vector [2]
Amazon CloudWatch	通过 HTTPS 的 REST	Latest	5.4+	Fluentd, Vector

1. Fluentd 不支持日志记录版本 5.6.2 中的 Elasticsearch 8。
2. Vector 支持日志记录版本 5.7 及更高版本中的 Syslog。

9.2.2. 输出类型描述

default

On-cluster、Red Hat 管理的日志存储。您不需要配置默认输出。



注意

如果您配置了**默认**输出，您会收到错误消息，因为保留了 **default** 输出名称以引用 on-cluster，Red Hat managed log store。

loki

Loki，一个可横向扩展的、高可用性、多租户日志聚合系统。

kafka

Kafka 代理。**kafka** 输出可以使用 TCP 或 TLS 连接。

elasticsearch

一个外部 Elasticsearch 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。**fluentForward** 输出可以使用 TCP 或 TLS 连接，并通过在 secret 中提供 **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。

重要

只有在使用 Fluentd 收集器时，才会支持 **fluentdForward** 输出。如果您使用 Vector 收集器，则不支持它。如果使用 Vector 收集器，您可以使用 **http** 输出将日志转发到 Fluentd。

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。

cloudwatch

Amazon CloudWatch，一种由 Amazon Web Services (AWS) 托管的监控和日志存储服务。

cloudlogging

Google Cloud Logging，由 Google Cloud Platform (GCP) 托管的监控和日志存储服务。

9.3. 启用 JSON 日志转发

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

9.3.1. 解析 JSON 日志

您可以使用 **ClusterLogForwarder** 对象将 JSON 日志解析到结构化对象，并将它们转发到受支持的输出。

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"}
```

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。

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。

9.3.2. 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。

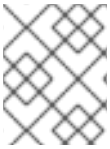
重要

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：

- **structuredTypeKey** 是 message 字段的名称。该字段的值用于构造索引名称。
  - **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。
- **structuredTypeName**: 如果没有设置 **structuredTypeKey** 字段，或者其键不存在，则 **structuredTypeName** 值将用作结构化类型。当您将 **structuredTypeKey** 字段和 **structuredTypeName** 字段一起使用时，如果 JSON 日志数据中缺少 **structuredTypeKey** 字段中的密钥，则 **structuredTypeName** 值将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。

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
# ...
  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", ...}}
```

```
}
```

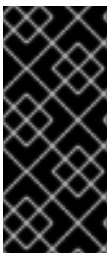
其他注意事项

- 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**。

9.3.3. 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。

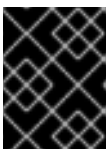
为避免与索引数量过多相关的性能问题，请考虑通过标准化到常见模式来保持可能的模式数量较低。

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. 使用 **structuredTypeKey** 字段指定其中一个日志记录字段。
3. 使用 **structuredTypeName** 字段指定名称。



重要

要解析 JSON 日志，您必须同时设置 **structuredTypeKey** 和 **structuredTypeName** 字段。

4. 对于 **inputRefs**，指定要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 <filename>.yaml
```

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

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

9.3.4. 将同一 pod 中的容器的 JSON 日志转发到单独的索引

您可以将来自同一 pod 的不同容器的结构化日志转发到不同的索引。要使用此功能，您必须使用多容器支持配置管道并注解 pod。日志被写入带有 **app-** 前缀的索引。建议将 Elasticsearch 配置为使用别名来容纳此目的。

重要

日志的 JSON 格式化因应用程序而异。因为创建太多索引会影响性能，所以请限制使用此功能，仅对与 JSON 格式不兼容的日志创建索引。使用查询将日志与不同命名空间分离，或使用兼容 JSON 格式的应用程序进行隔离。

先决条件

- Red Hat OpenShift 的日志记录： 5.5

流程

1. 创建或编辑定义 **ClusterLogForwarder** CR 对象的 YAML 文件：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputDefaults:
    elasticsearch:
      structuredTypeKey: kubernetes.labels.logFormat 1
      structuredTypeName: nologformat
      enableStructuredContainerLogs: true 2
  pipelines:
  - inputRefs:
    - application
    name: application-logs
    outputRefs:
    - default
    parse: json
```

1. 使用 Kubernetes **logFormat** 标签形成的键值对值。
2. 启用多容器输出。

2. 创建或编辑定义 **Pod** CR 对象的 YAML 文件：

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    containerType.logging.openshift.io/heavy: heavy (1)
    containerType.logging.openshift.io/low: low
spec:
  containers:
  - name: heavy (2)
    image: heavyimage
  - name: low
    image: lowimage
```

1. 格式：**containerType.logging.openshift.io/<container-name>: <index>**
2. 注解名称必须与容器名称匹配

警告

此配置可能会显著增加集群中的分片数量。

其它资源

- Kubernetes 注解

其他资源

- 关于日志转发

9.4. 配置日志转发

在日志记录部署中，容器和基础架构日志默认转发到 **ClusterLogging** 自定义资源(CR)中定义的内部日志存储。

默认情况下，审计日志不会转发到内部日志存储，因为这不提供安全存储。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。

如果此默认配置满足您的需要，则不需要配置一个 **ClusterLogForwarder** CR。如果存在 **ClusterLogForwarder** CR，日志不会转发到内部日志存储，除非定义了包含 **default** 输出的管道。

9.4.1. 关于将日志转发到第三方系统

要将日志发送到 OpenShift Container Platform 集群内部和外部的特定端点，您可以在 **ClusterLogForwarder** 自定义资源(CR)中指定 *输出*和 *管道*的组合。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。身份验证由 Kubernetes *Secret* 对象提供。

pipeline

定义从一个日志类型到一个或多个输出的简单路由，或定义您要发送的日志。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**.由节点审计系统、**auditd**、Kubernetes API 服务器、OpenShift API 服务器和 OVN 网络生成的审计日志。

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。

输入

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。

Secret

包含机密数据的 **key:value 映射**，如用户凭据。

注意以下几点：

- 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。

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: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
   - name: elasticsearch-secure 4
     type: "elasticsearch"
     url: https://elasticsearch.secure.com:9200
     secret:
       name: elasticsearch
   - name: elasticsearch-insecure 5
     type: "elasticsearch"
     url: http://elasticsearch.insecure.com:9200
   - name: kafka-app 6
     type: "kafka"
     url: tls://kafka.secure.com:9093/app-topic
```

```
inputs: 7
 - name: my-app-logs
   application:
     namespaces:
     - my-project
pipelines:
 - name: audit-logs 8
   inputRefs:
    - audit
   outputRefs:
    - elasticsearch-secure
    - default
   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 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。

2 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。

3 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。

4 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。

5 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。

6 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。

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

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

- 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。

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。

- 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。

支持的授权密钥
这里提供了常见的密钥类型。某些输出类型支持额外的专用密钥，记录在特定于输出的配置字段中。所有 secret 密钥都是可选的。通过设置相关密钥来启用您想要的安全功能。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。Open Shift Logging 不会尝试验证授权组合间的不匹配。

传输层安全性(TLS)

使用没有 secret 的 TLS URL (**http://...** 或 **ssl://...**) 启用基本的 TLS 服务器端身份验证。可通过包含 Secret 并设置以下可选字段来启用额外的 TLS 功能：

- **密码短语** ：（字符串）对编码的 TLS 私钥进行解码。需要 **tls.key**。
- **ca-bundle.crt**: （字符串）用于服务器身份验证的客户 CA 的文件名。

用户名和密码

- **username** ：（字符串）身份验证用户名。需要 **password**。
- **password** ：（字符串）身份验证密码。需要 **username**。

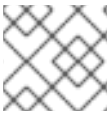
简单身份验证安全层(SASL)

- **sasl.enable** （布尔值）明确指定启用或禁用 SASL。如果缺失，则设置了任何其他 **sasl.** 密钥时自动启用 SASL。
- **sasl.mechanisms** ：(array)允许的 SASL 机制名称列表。如果缺少或为空，则使用系统默认值。
- **sasl.allow-insecure** ：（布尔值）允许发送明文密码的机制。默认为false。

9.4.1.1. 创建 Secret

您可以使用以下命令在包含您的证书和密钥文件的目录中创建 secret：

```
$ oc create secret generic -n <namespace> <secret_name> \
  --from-file=ca-bundle.crt=<your_bundle_file> \
  --from-literal=username=<your_username> \
  --from-literal=password=<your_password>
```



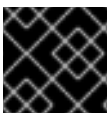
注意

建议使用通用或不透明 secret 来获得最佳结果。

9.4.2. 创建日志转发器

要创建日志转发器，您必须创建一个 **ClusterLogForwarder** CR，以指定服务帐户可以收集的日志输入类型。您还可以指定日志可以转发到的输出。如果使用多日志转发器功能，还必须在 **ClusterLogForwarder** CR 中引用服务帐户。

如果您在集群中使用多日志转发器功能，您可以使用任何名称在任意命名空间中创建 **ClusterLogForwarder** 自定义资源 (CR)。如果使用旧的实现， **ClusterLogForwarder** CR 必须命名为 **instance**，且必须在 **openshift-logging** 命名空间中创建。



重要

创建 **ClusterLogForwarder** CR 的命名空间需要管理员权限。

ClusterLogForwarder 资源示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  pipelines:
   - inputRefs:
     - <log_type> 4
     outputRefs:
     - <output_name> 5
  outputs:
  - name: <output_name> 6
    type: <output_type> 7
    url: <log_output_url> 8
# ...
```

1. 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。
2. 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。
3. 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。
4. 收集的日志类型。此字段的值可以是 **audit**(用于审计日志）、**application**（用于应用程序日志）、**infrastructure**（用于基础架构日志），或输入为您的应用程序定义的名称。
5. 7 要将日志转发到的输出类型。此字段的值可以是 **default**,**loki**,**kafka**,**elasticsearch**,**fluentdForward**,**syslog**, 或 **cloudwatch**。

注意

多日志转发器实现不支持 **default** 输出类型。

6. 要将日志转发到的输出的名称。
8. 要将日志转发到的输出的 URL。

9.4.3. 调整日志有效负载和交付

在日志记录 5.9 及更新版本中，**ClusterLogForwarder** 自定义资源(CR)中的 **tuning** spec 提供了配置部署以优先选择日志吞吐量或持久性的方法。

例如，如果您需要减少收集器重启时日志丢失的可能性，或者您需要在收集器重启后收集日志消息来支持规范，您可以调整部署以优先选择日志持久性。如果您使用对可以接收的批处理大小有硬限制的输出，您可能需要调整部署以优先处理日志吞吐量。

重要

要使用这个功能，您的日志记录部署必须配置为使用 Vector 收集器。使用 Fluentd 收集器时，不支持 **ClusterLogForwarder** CR 中的 **tuning** spec。

以下示例显示了您可以修改的 **ClusterLogForwarder** CR 选项来调整日志转发器输出：

ClusterLogForwarder CR 调整选项示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  tuning:
    delivery: AtLeastOnce 1
    compression: none 2
    maxWrite: <integer> 3
    minRetryDuration: 1s 4
    maxRetryDuration: 1s 5
# ...
```

1. 指定日志转发的交付模式。
   - **AtLeastOnce** 交付表示如果日志转发器崩溃或重启，则任何在崩溃前读取的日志都会重新发送到其目的地。有些日志可能会在崩溃后重复。
   - **AtMostOnce** 交付意味着日志转发器不会努力恢复崩溃期间丢失的日志。这个模式可以提供更好的吞吐量，但可能会导致日志丢失。
2. 指定 **compression** 配置会导致在通过网络发送数据前压缩数据。请注意，并非所有输出类型都支持压缩，如果输出不支持指定的压缩类型，这会导致错误。此配置的可能值为 **none**（不压缩）、**gzip**、**snappy**、**zlib** 或 **zstd**。如果您使用 Kafka 输出，也可以使用 **lz4** 压缩。如需更多信息，请参阅表"支持压缩类型用于调优输出"。
3. 为向输出发送操作的最大有效负载指定限制。
4. 指定在失败后重试发送前在尝试之间等待的时间。这个值是一个字符串，可指定为毫秒(**ms**)、秒(**s**)或分钟(**m**)。
5. 指定在失败后重试发送前在尝试之间等待的最长时间。这个值是一个字符串，可指定为毫秒(**ms**)、秒(**s**)或分钟(**m**)。

表 9.9. 支持的调整输出的压缩类型

压缩算法	Splunk	Amazon Cloudwatch	Elasticsearch 8	LokiStack	Apache Kafka	HTTP	Syslog	Google Cloud	Microsoft Azure Monitoring
gzip	X	X	X	X		X			

压缩算法	Splunk	Amazon Cloudwatch	Elasticsearch 8	LokiStack	Apache Kafka	HTTP	Syslog	Google Cloud	Microsoft Azure Monitoring
snappy		X		X	X	X			
zlib		X	X			X			
zstd		X			X	X			
lz4					X				

9.4.4. 启用多行异常检测

启用容器日志的多行错误检测。

警告

启用此功能可能会对性能有影响，可能需要额外的计算资源或备用日志记录解决方案。

日志解析器通常会错误地将同一个例外中的不同的行识别为不同的例外。这会导致额外的日志条目，以及要跟踪的信息的不完整或不正确。

java 异常示例

```
java.lang.NullPointerException: Cannot invoke "String.toString()" because "<param1>" is null
    at testjava.Main.handle(Main.java:47)
    at testjava.Main.printMe(Main.java:19)
    at testjava.Main.main(Main.java:10)
```

- 要启用日志记录来检测多行异常，并将其重新编译到一个日志条目中，请确保 **ClusterLogForwarder** 自定义资源 (CR) 包含 **detectMultilineErrors** 字段，值为 **true**。

ClusterLogForwarder CR 示例

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

```
- name: my-app-logs
  inputRefs:
    - application
  outputRefs:
    - default
  detectMultilineErrors: true
```

9.4.4.1. 详情

当日志消息作为一系列针对一个例外的信息出现时，会将它们合并到一个统一的日志记录中。第一个日志消息的内容被替换为序列中所有消息字段的连接内容。

表 9.10. 每个收集器支持的语言：

语言	Fluentd	Vector
Java	✓	✓
JS	✓	✓
Ruby	✓	✓
Python	✓	✓
Golang	✓	✓
PHP	✓	✓
Dart	✓	✓

9.4.4.2. 故障排除

启用后，收集器配置将包括一个新的部分，类型是：**detect_exceptions**

vector 配置部分的示例

```
[transforms.detect_exceptions_app-logs]
 type = "detect_exceptions"
 inputs = ["application"]
 languages = ["All"]
 group_by = ["kubernetes.namespace_name","kubernetes.pod_name","kubernetes.container_name"]
 expire_after_ms = 2000
 multiline_flush_interval_ms = 1000
```

fluentd config 部分的示例

```
<label @MULTILINE_APP_LOGS>
  <match kubernetes.**>
    @type detect_exceptions
    remove_tag_prefix 'kubernetes'
    message message
```

```
    force_line_breaks true
    multiline_flush_interval .2
  </match>
</label>
```

9.4.5. 将日志转发到 Google Cloud Platform (GCP)

除了默认的 OpenShift Container Platform 日志存储外，您还可以将日志转发到 Google Cloud Logging。

注意

不支持在 Fluentd 中使用此功能。

先决条件

- Red Hat OpenShift Logging Operator 5.5.1 及更新的版本

流程

1. 使用 Google 服务帐户密钥创建 secret。

```
$ oc -n openshift-logging create secret generic gcp-secret --from-file google-application-credentials.json=<your_service_account_key_file.json>
```

2. 使用以下模板创建 **ClusterLogForwarder** 自定义资源 YAML：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
    - name: gcp-1
      type: googleCloudLogging
      secret:
        name: gcp-secret
      googleCloudLogging:
        projectId : "openshift-gce-devel" 4
        logId : "app-gcp" 5
  pipelines:
    - name: test-app
      inputRefs: 6
        - application
      outputRefs:
        - gcp-1
```

1. 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。

2. 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。

3. 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。

4. 根据您要将日志存储在 GCP 资源层次结构中的位置，设置 **projectId**, **folderId**, **organizationId**, 或 **billingAccountId** 的项及其相应的值。

5. 将值设为添加到 Log Entry 的 **logName** 字段的值。

6. 使用管道指定要转发的日志类型：**application**, **infrastructure**, 或 **audit**。

其他资源

- Google Cloud Billing 文档
- Google Cloud Logging Query Language 文档

9.4.6. 将日志转发到 Splunk

除了内部的默认 OpenShift Container Platform 日志存储外，您还可以将日志转发到 Splunk HTTP 事件收集器 (HEC)。

注意

不支持在 Fluentd 中使用此功能。

先决条件

- Red Hat OpenShift Logging Operator 5.6 或更高版本
- 带有指定了 **vector** 的 **ClusterLogging** 实例作为收集器
- Base64 编码的 Splunk HEC 令牌

流程

1. 使用您的 Base64 编码的 Splunk HEC 令牌创建 secret。

```
$ oc -n openshift-logging create secret generic vector-splunk-secret --from-literal hecToken=
<HEC_Token>
```

2. 使用以下模板创建或编辑 **ClusterLogForwarder** 自定义资源 (CR)：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
    - name: splunk-receiver 4
      secret:
```

```
      name: vector-splunk-secret 5
    type: splunk 6
    url: <http://your.splunk.hec.url:8088> 7
  pipelines: 8
   - inputRefs:
      - application
      - infrastructure
    name: 9
    outputRefs:
      - splunk-receiver 10
```

1. 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。
2. 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。
3. 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。
4. 指定输出的名称。
5. 指定包含 HEC 令牌的 secret 名称。
6. 将输出类型指定为 **mvapich**。
7. 指定 Splunk HEC 的 URL （包括端口）。
8. 使用管道指定要转发的日志类型：**application**, **infrastructure**, 或 **audit**。
9. 可选：指定管道的名称。
10. 指定使用此管道转发日志时使用的输出名称。

9.4.7. 通过 HTTP 转发日志

Fluentd 和 Vector 日志收集器都支持通过 HTTP 转发日志。要启用，在 **ClusterLogForwarder** 自定义资源 (CR) 中指定 **http** 作为输出类型。

流程

- 使用以下模板创建或编辑 **ClusterLogForwarder** CR：

  ClusterLogForwarder CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
    - name: httpout-app
```

```
    type: http
    url: 4
    http:
      headers: 5
        h1: v1
        h2: v2
      method: POST
    secret:
      name: 6
    tls:
      insecureSkipVerify: 7
pipelines:
  - name:
    inputRefs:
      - application
    outputRefs:
      - 8
```

1. 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。
2. 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。
3. 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。
4. 日志的目标地址。
5. 使用日志记录发送的其他标头。
6. 目标凭证的 secret 名称。
7. 值可以是 **true** 或 **false**。
8. 这个值应当与输出名称相同。

9.4.8. 转发到 Azure Monitor 日志

使用日志记录 5.9 及之后的版本时，除了默认的日志存储外，您还可以将日志转发到 Azure Monitor 日志。这个功能由 Vector Azure Monitor Logs sink 提供。

先决条件

- 熟悉如何管理和创建 **ClusterLogging** 自定义资源 (CR) 实例。
- 熟悉如何管理和创建 **ClusterLogForwarder** CR 实例。
- 您了解 **ClusterLogForwarder** CR 规格。
- 您对 Azure 服务有一定的了解。
- 您已为 Azure Portal 或 Azure CLI 访问配置了 Azure 帐户。
- 您已获取了 Azure Monitor Logs 主或从安全密钥。

- 您已确定要转发的日志类型。

通过 HTTP Data Collector API 启用日志转发到 Azure Monitor 日志：

使用您的共享密钥创建 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
  namespace: openshift-logging
type: Opaque
data:
  shared_key: <your_shared_key> 1
```

1 必须包含生成请求的 Log Analytics 工作区 的主或从密钥。

要获取共享密钥，您可以在 Azure CLI 中使用这个命令：

```
Get-AzOperationalInsightsWorkspaceSharedKey -ResourceGroupName "<resource_name>" -Name
"<workspace_name>”
```

使用与日志选择匹配的模板创建或编辑 **ClusterLogForwarder** CR。

转发所有日志

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: azure-monitor
    type: azureMonitor
    azureMonitor:
      customerId: my-customer-id 1
      logType: my_log_type 2
    secret:
       name: my-secret
  pipelines:
  - name: app-pipeline
    inputRefs:
    - application
    outputRefs:
    - azure-monitor
```

1 Log Analytics 工作区的唯一标识符。必填字段。

2 正在提交的数据的 Azure 记录类型。只能包含字母、数字和下划线 (_)，且不得超过 100 个字符。

转发应用程序和基础架构日志

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: azure-monitor-app
    type: azureMonitor
    azureMonitor:
      customerId: my-customer-id
      logType: application_log 1
    secret:
      name: my-secret
  - name: azure-monitor-infra
    type: azureMonitor
    azureMonitor:
      customerId: my-customer-id
      logType: infra_log #
    secret:
      name: my-secret
  pipelines:
    - name: app-pipeline
      inputRefs:
      - application
      outputRefs:
      - azure-monitor-app
    - name: infra-pipeline
      inputRefs:
      - infrastructure
      outputRefs:
      - azure-monitor-infra
```

1 正在提交的数据的 Azure 记录类型。只能包含字母、数字和下划线 (_)，且不得超过 100 个字符。

高级配置选项

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: azure-monitor
    type: azureMonitor
    azureMonitor:
      customerId: my-customer-id
      logType: my_log_type
      azureResourceId: "/subscriptions/111111111" 1
      host: "ods.opinsights.azure.com" 2
    secret:
       name: my-secret
  pipelines:
```

```
- name: app-pipeline
 inputRefs:
 - application
 outputRefs:
 - azure-monitor
```

1. 数据应与之关联的 Azure 资源的资源 ID。可选字段。
2. 专用 Azure 区域的替代主机。可选字段。默认值为 **ods.opinsights.azure.com**。Azure Government 的默认值为 **ods.opinsights.azure.us**。

9.4.9. 从特定项目转发应用程序日志

除了内部日志存储外，您还可以将特定项目的应用程序日志副本转发到外部日志聚合器。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. 创建或编辑定义 **ClusterLogForwarder** CR 的 YAML 文件：

   ClusterLogForwarder CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
   - name: fluentd-server-secure 3
     type: fluentdForward 4
     url: 'tls://fluentdserver.security.example.com:24224' 5
     secret: 6
        name: fluentd-secret
   - name: fluentd-server-insecure
     type: fluentdForward
     url: 'tcp://fluentdserver.home.example.com:24224'
  inputs: 7
   - name: my-app-logs
     application:
        namespaces:
        - my-project 8
  pipelines:
   - name: forward-to-fluentd-insecure 9
     inputRefs: 10
     - my-app-logs
```

```
  outputRefs: 11
  - fluentd-server-insecure
  labels:
    project: "my-project" 12
- name: forward-to-fluentd-secure 13
  inputRefs:
  - application 14
  - audit
  - infrastructure
  outputRefs:
  - fluentd-server-secure
  - default
  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. 输出的名称。
4. 输出类型：**elasticsearch**, **fluentdForward**, **syslog**, 或 **kafka**。
5. 外部日志聚合器的 URL 和端口作为有效的绝对 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。
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，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. 如果没有指定命名空间，则会从所有命名空间收集日志。
9. 管道配置将来自一个命名输入的日志定向到一个命名的输出。在本例中，名为 **forward-to-fluentd-insecure** 的管道将日志从一个名为 **my-app-logs** 的输入转发到名为 **fluentd-server-insecure** 的输出。
10. 输入列表。
11. 要使用的输出名称。
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。
    - 使用管道指定要转发的日志类型：**application**、**infrastructure** 或 **audit**。
    - 指定使用此管道转发日志时使用的输出名称。
    - 可选：指定将日志转发到默认日志存储的**默认**输出。
    - 可选：字符串。要添加到日志中的一个或多个标签。

14 请注意，使用此配置时，会从所有命名空间收集应用程序日志。

2. 运行以下命令来应用 **ClusterLogForwarder** CR：

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

9.4.10. 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. 创建或编辑定义 **ClusterLogForwarder** CR 对象的 YAML 文件。在文件中，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: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  pipelines:
    - inputRefs: [ myAppLogData ] 3
      outputRefs: [ default ] 4
  inputs: 5
    - name: myAppLogData
      application:
        selector:
          matchLabels: 6
            environment: production
            app: nginx
        namespaces: 7
        - app1
        - app2
  outputs: 8
    - <output_name>
    ...
```

1 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。

2 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。

3. 指定来自 **inputs[].name** 的一个或多个以逗号分隔的值。
4. 指定来自 **outputs[]** 的一个或多个以逗号分隔的值。
5. 为具有一组唯一 pod 标签的每个应用程序定义唯一的 **inputs[].name**。
6. 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。
7. 可选：指定一个或多个命名空间。
8. 指定要将日志数据转发到的一个或多个输出。

2. 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。
3. 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 ]
```

4. 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。

9.4.11. API 审计过滤器概述

OpenShift API 服务器为每个 API 调用生成审计事件，详细说明请求者的请求、响应和请求者的身份，从而导致大量数据。API 审计过滤器使用规则启用非必要事件和事件大小减少，从而提高更易于管理的审计跟踪。按顺序检查规则，检查会在第一个匹配项时停止。事件中包含的数据量由 **level** 字段的值决定：

- **None**: 事件被丢弃。
- **Metadata**：只包含审计元数据，请求和响应正文会被删除。
- **Request**：包含审计元数据和请求正文，响应正文会被删除。
- **RequestResponse**：包含所有数据：元数据、请求正文和响应正文。响应正文可能非常大。例如，**oc get pods -A** 生成包含集群中每个 pod 的 YAML 描述的响应正文。

在日志记录 5.8 及更高版本中，**ClusterLogForwarder** 自定义资源 (CR) 使用与标准 Kubernetes Audit 策略相同的格式，同时提供以下附加功能：

通配符

用户、组、命名空间和资源的名称可以在前导或尾部带有 * 星号字符。例如，命名空间 **openshift-*** 与 **openshift-apiserver** 或 **openshift-authentication** 匹配。资源 ***/status** 匹配 **Pod/status** 或 **Deployment/status**。

默认规则

与策略中任何规则不匹配的事件将被过滤，如下所示：

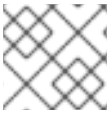
- 只读系统事件（如 **get**、**list**、**watch**）将被丢弃。
- 服务帐户写入发生在与服务帐户相同的命名空间中的事件将被丢弃。
- 所有其他事件都会被转发，受任何配置的速率限制。

要禁用这些默认值，请使用只有一个 **level** 字段的规则结束您的规则列表，或者添加一条空规则。

省略响应代码

要省略的整数状态代码列表。您可以使用 **OmitResponseCodes** 字段（没有创建事件）的 HTTP 状态代码列表根据响应中的 HTTP 状态代码丢弃事件。默认值为 **[404, 409, 422, 429]**。如果该值为空列表 **[]**，则不会省略状态代码。

ClusterLogForwarder CR Audit 策作为 OpenShift Container Platform 审计策略外的补充起作用。**ClusterLogForwarder** CR 审计过滤器更改日志收集器转发的内容，并提供按操作动词、用户、组、命名空间或资源过滤的功能。您可以创建多个过滤器，将同一审计流的不同摘要发送到不同的位置。例如，您可以将详细的流发送到本地集群日志存储，并将不太详细的流发送到远程站点。



注意

提供的示例旨在说明审计策略中可能的规则范围，不是推荐的配置。

Audit 策略示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  pipelines:
    - name: my-pipeline
      inputRefs: audit (1)
      filterRefs: my-policy (2)
      outputRefs: default
  filters:
    - name: my-policy
      type: kubeAPIAudit
      kubeAPIAudit:
        # Don't generate audit events for all requests in RequestReceived stage.
        omitStages:
          - "RequestReceived"

        rules:
          # Log pod changes at RequestResponse level
          - level: RequestResponse
            resources:
```

```
  - group: ""
    resources: ["pods"]

# Log "pods/log", "pods/status" at Metadata level
- level: Metadata
  resources:
  - group: ""
    resources: ["pods/log", "pods/status"]

# Don't log requests to a configmap called "controller-leader"
- level: None
  resources:
  - group: ""
    resources: ["configmaps"]
    resourceNames: ["controller-leader"]

# Don't log watch requests by the "system:kube-proxy" on endpoints or services
- level: None
  users: ["system:kube-proxy"]
  verbs: ["watch"]
  resources:
  - group: "" # core API group
    resources: ["endpoints", "services"]

# Don't log authenticated requests to certain non-resource URL paths.
- level: None
  userGroups: ["system:authenticated"]
  nonResourceURLs:
  - "/api*" # Wildcard matching.
  - "/version"

# Log the request body of configmap changes in kube-system.
- level: Request
  resources:
  - group: "" # core API group
    resources: ["configmaps"]
  # This rule only applies to resources in the "kube-system" namespace.
  # The empty string "" can be used to select non-namespaced resources.
  namespaces: ["kube-system"]

# Log configmap and secret changes in all other namespaces at the Metadata level.
- level: Metadata
  resources:
  - group: "" # core API group
    resources: ["secrets", "configmaps"]

# Log all other resources in core and extensions at the Request level.
- level: Request
  resources:
  - group: "" # core API group
  - group: "extensions" # Version of group should NOT be included.

# A catch-all rule to log all other requests at the Metadata level.
- level: Metadata
```

1. 收集的日志类型。此字段的值可以是 **audit**(用于审计日志）、**application**（用于应用程序日志）、**infrastructure**（用于基础架构日志），或输入为您的应用程序定义的名称。
2. 审计策略的名称。

其他资源

- 记录网络策略事件[Logging 用于出口防火墙和网络策略规则]

9.4.12. 将日志转发到外部 Loki 日志记录系统

除了默认的日志存储外，您还可以将日志转发到外部 Loki 日志记录系统。

要配置日志转发到 Loki，您必须创建一个 **ClusterLogForwarder** 自定义资源 (CR)，并创建一个输出到 Loki 的 ClusterLogForwarder 自定义资源 (CR)，以及使用输出的管道。到 Loki 的输出可以使用 HTTP（不安全）或 HTTPS（安全 HTTP）连接。

先决条件

- 您必须有一个 Loki 日志记录系统在您通过 CR 中的 **url** 字段指定的 URL 中运行。

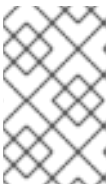
流程

1. 创建或编辑定义 **ClusterLogForwarder** CR 对象的 YAML 文件：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
  - name: loki-insecure 4
    type: "loki" 5
    url: http://loki.insecure.com:3100 6
    loki:
      tenantKey: kubernetes.namespace_name
      labelKeys:
      - kubernetes.labels.foo
  - name: loki-secure 7
    type: "loki"
    url: https://loki.secure.com:3100
    secret:
      name: loki-secret 8
    loki:
      tenantKey: kubernetes.namespace_name 9
      labelKeys:
      - kubernetes.labels.foo 10
  pipelines:
  - name: application-logs 11
    inputRefs: 12
    - application
```

```
    - audit
    outputRefs: 13
    - loki-secure
```

1. 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。
2. 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。
3. 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。
4. 指定输出的名称。
5. 将类型指定为 **"loki"**。
6. 将 Loki 系统的 URL 和端口指定为有效的绝对 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。Loki 用于 HTTP(S) 通讯的默认端口为 3100。
7. 对于安全连接，您可以通过指定 **secret** 来指定您进行身份验证的 **https** 或 **http** URL。
8. 对于 **https** 前缀，请指定 TLS 通信端点所需的 secret 名称。secret 必须包含一个 **ca-bundle.crt** 键，它指向它所代表的证书。否则，对于 **http** 和 **https** 前缀，您可以指定一个包含用户名和密码的 secret。在旧的实现中，secret 必须存在于 **openshift-logging** 项目中。如需更多信息，请参阅以下"示例：设置包含用户名和密码的 secret"。
9. 可选：指定一个 metadata key 字段，为 Loki 中的 **TenantID** 字段生成值。例如，设置 **tenantKey: kubernetes.namespace_name** 使用 Kubernetes 命名空间的名称作为 Loki 中的租户 ID 的值。要查看您可以指定的其他日志记录字段，请查看以下"Additional resources"部分中的"Log Record Fields"链接。
10. 可选：指定一个 metadata 字段键列表来替换默认的 Loki 标签。Loki 标签名称必须与正则表达式 **[a-zA-Z_:][a-zA-Z0-9_:]*** 匹配。无数据键中的非法字符被替换为 _，以组成标签名称。例如，**kubernetes.labels.foo** 元数据键变成 Loki 标签 **kubernetes_labels_foo**。如果没有设置 **labelKeys**，则默认值为：**[log_type, kubernetes.namespace_name, kubernetes.pod_name, kubernetes_host]**。尽量保持标签数量少，因为 Loki 会限制允许标签的大小和数量。请参阅配置 Loki、limit_config。您仍然可以使用查询过滤器基于任何日志记录字段进行查询。
11. 可选：指定管道的名称。
12. 使用管道指定要转发的日志类型：**application**、**infrastructure** 或 **audit**。
13. 指定使用此管道转发日志时使用的输出名称。



注意

由于 Loki 要求按时间戳正确排序日志流，**labelKeys** 始终包含 **kubernetes_host** 标签，即使您没有指定它。此包含确保每个流源自单一主机，这样可防止因为不同主机上的时钟差异而导致时间戳出现问题。

2. 运行以下命令来应用 **ClusterLogForwarder** CR 对象：

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

其他资源

- 配置 Loki 服务器

9.4.13. 将日志转发到外部 Elasticsearch 实例

除了内部日志存储外，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。

注意

如果您只想将日志转发到内部 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. 创建或编辑定义 **ClusterLogForwarder** CR 的 YAML 文件：

   ClusterLogForwarder CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
   - name: elasticsearch-example 4
     type: elasticsearch 5
     elasticsearch:
       version: 8 6
     url: http://elasticsearch.example.com:9200 7
     secret:
       name: es-secret 8
  pipelines:
   - name: application-logs 9
     inputRefs: 10
     - application
```

```
    - audit
    outputRefs:
    - elasticsearch-example 11
    - default 12
    labels:
      myLabel: "myValue" 13
# ...
```

1. 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。
2. 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。
3. 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。
4. 指定输出的名称。
5. 指定 **elasticsearch** 类型。
6. 指定 Elasticsearch 版本。这可以是 **6**、**7** 或 **8**。
7. 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。
8. 对于 **https** 前缀，请指定 TLS 通信端点所需的 secret 名称。secret 必须包含一个 **ca-bundle.crt** 键，它指向它所代表的证书。否则，对于 **http** 和 **https** 前缀，您可以指定一个包含用户名和密码的 secret。在旧的实现中，secret 必须存在于 **openshift-logging** 项目中。如需更多信息，请参阅以下"示例：设置包含用户名和密码的 secret"。
9. 可选：指定管道的名称。
10. 使用管道指定要转发的日志类型：**application**、**infrastructure** 或 **audit**。
11. 指定使用此管道转发日志时使用的输出名称。
12. 可选：指定将日志发送到内部 Elasticsearch 实例的 **default** 输出。
13. 可选：字符串。要添加到日志中的一个或多个标签。

2. 应用 **ClusterLogForwarder** CR：

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

示例：设置包含用户名和密码的 secret

您可以使用包含用户名和密码的 secret 来验证与外部 Elasticsearch 实例的安全连接。

例如，如果无法使用 mutual TLS (mTLS) 密钥，因为第三方运行 Elasticsearch 实例，您可以使用 HTTP 或 HTTPS 并设置包含用户名和密码的 secret。

1. 创建类似于以下示例的 **Secret** YAML 文件。将 base64 编码的值用于 **username** 和 **password** 字段。secret 类型默认为 opaque。

```
apiVersion: v1
kind: Secret
metadata:
  name: openshift-test-secret
data:
  username: <username>
  password: <password>
# ...
```

2. 创建 secret：

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

3. 在 **ClusterLogForwarder** CR 中指定 secret 的名称：

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

注意

在 **url** 字段中，前缀可以是 **http** 或 **https**。

4. 应用 CR 对象：

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

9.4.14. 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。

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. 创建或编辑定义 **ClusterLogForwarder** CR 对象的 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
     labels:
       clusterId: "C1234" 11
   - name: forward-to-fluentd-insecure 12
     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，且必须包含指向它所代表证书的 **ca-bundle.crt** 键。
7. 可选：指定管道的名称。
8. 使用管道指定要转发的日志类型：**application**、**infrastructure** 或 **audit**。
9. 指定使用此管道转发日志时使用的输出名称。

10 可选：指定将日志转发到内部 Elasticsearch 实例的 **default** 输出。

11 可选：字符串。要添加到日志中的一个或多个标签。

12 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：

- 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
```

9.4.14.1. 为 Logstash 启用 nanosecond 精度来从 fluentd 摄取数据

对于 Logstash 从 fluentd 摄取数据，您必须在 Logstash 配置文件中启用 nanosecond 精度。

流程

- 在 Logstash 配置文件中，将 **nanosecond_precision** 设置为 **true**。

Logstash 配置文件示例

```
input { tcp { codec => fluent { nanosecond_precision => true } port => 24114 } }
filter { }
output { stdout { codec => rubydebug } }
```

9.4.15. 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。

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. 创建或编辑定义 **ClusterLogForwarder** CR 对象的 YAML 文件：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
```

```
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
   - name: rsyslog-east 4
     type: syslog 5
     syslog: 6
       facility: local0
       rfc: RFC3164
       payloadKey: message
       severity: informational
     url: 'tls://rsyslogserver.east.example.com:514' 7
     secret: 8
        name: syslog-secret
   - name: rsyslog-west
     type: syslog
     syslog:
      appName: myapp
      facility: user
      msgID: mymsg
      procID: myproc
      rfc: RFC5424
      severity: debug
     url: 'tcp://rsyslogserver.west.example.com:514'
  pipelines:
   - name: syslog-east 9
     inputRefs: 10
     - audit
     - application
     outputRefs: 11
     - rsyslog-east
     - default 12
     labels:
       secure: "true" 13
       syslog: "east"
   - name: syslog-west 14
     inputRefs:
     - infrastructure
     outputRefs:
     - rsyslog-west
     - default
     labels:
       syslog: "west"
```

1. 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。
2. 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。
3. 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。
4. 指定输出的名称。

5. 指定 **syslog** 类型。
6. 可选：指定 syslog 参数，如下所列。
7. 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。
8. 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 必须包含一个 **ca-bundle.crt** 键，它指向它所代表的证书。在旧的实现中，secret 必须存在于 **openshift-logging** 项目中。
9. 可选：指定管道的名称。
10. 使用管道指定要转发的日志类型：**application**、**infrastructure** 或 **audit**。
11. 指定使用此管道转发日志时使用的输出名称。
12. 可选：指定将日志转发到内部 Elasticsearch 实例的 **default** 输出。
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：
    - 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：

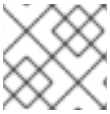
```
$ oc create -f <filename>.yaml
```

9.4.15.1. 在消息输出中添加日志消息

您可以通过将 **AddLogSource** 字段添加到 **ClusterLogForwarder** 自定义资源(CR)将 **namespace_name**、**pod_name** 和 **container_name** 元素添加到记录的 **message** 字段中。

```
spec:
  outputs:
  - name: syslogout
    syslog:
      addLogSource: true
      facility: user
      payloadKey: message
      rfc: RFC3164
      severity: debug
      tag: mytag
    type: syslog
    url: tls://syslog-receiver.openshift-logging.svc:24224
  pipelines:
```

```
- inputRefs:
  - application
  name: test-app
  outputRefs:
  - syslogout
```



注意

这个配置与 RFC3164 和 RFC5424 兼容。

没有 AddLogSource 的 syslog 消息输出示例

```
<15>1 2020-11-15T17:06:14+00:00 fluentd-9hkb4 mytag - - - {"msgcontent"=>"Message Contents",
"timestamp"=>"2020-11-15 17:06:09", "tag_key"=>"rec_tag", "index"=>56}
```

带有 AddLogSource 的 syslog 消息输出示例

```
<15>1 2020-11-16T10:49:37+00:00 crc-j55b9-master-0 mytag - - -  namespace_name=clo-test-
6327,pod_name=log-generator-ff9746c49-qxm7l,container_name=log-generator,message=
{"msgcontent":"My life is my message", "timestamp":"2020-11-16 10:49:36", "tag_key":"rec_tag",
"index":76}
```

9.4.15.2. 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。

9.4.15.3. 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。

9.4.16. 将日志转发到 Kafka 代理

除了默认的日志存储外，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. 创建或编辑定义 **ClusterLogForwarder** CR 对象的 YAML 文件：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
   - name: app-logs 4
     type: kafka 5
     url: tls://kafka.example.devlab.com:9093/app-topic 6
     secret:
       name: kafka-secret 7
   - name: infra-logs
     type: kafka
     url: tcp://kafka.devlab2.example.com:9093/infra-topic 8
   - name: audit-logs
     type: kafka
     url: tls://kafka.qelab.example.com:9093/audit-topic
     secret:
        name: kafka-secret-qe
  pipelines:
   - name: app-topic 9
     inputRefs: 10
     - application
     outputRefs: 11
     - app-logs
     labels:
       logType: "application" 12
   - name: infra-topic 13
     inputRefs:
     - infrastructure
     outputRefs:
     - infra-logs
     labels:
       logType: "infra"
   - name: audit-topic
     inputRefs:
     - audit
     outputRefs:
     - audit-logs
     labels:
       logType: "audit"
```

1 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。

2 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。

3 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。

4 指定输出的名称。

5 指定 **kafka** 类型。

6 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。

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 必须包含一个 **ca-bundle.crt** 键，它指向它所代表的证书。在旧的实现中，secret 必须存在于 **openshift-logging** 项目中。

8 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**。

9 可选：指定管道的名称。

10 使用管道指定要转发的日志类型：**application**、**infrastructure** 或 **audit**。

11 指定使用此管道转发日志时使用的输出名称。

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：

- 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. 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. 运行以下命令来应用 **ClusterLogForwarder** CR：

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

9.4.17. 将日志转发到 Amazon CloudWatch

您可以将日志转发到 Amazon CloudWatch，这是由 Amazon Web Services (AWS) 托管的监控和日志存储服务。除了默认的日志存储外，您还可以将日志转发到 CloudWatch。

要配置日志转发到 CloudWatch，您必须创建一个 **ClusterLogForwarder** 自定义资源 (CR)，其中包含 CloudWatch 的输出，以及使用输出的管道。

流程

1. 创建一个 **Secret** YAML 文件，它使用 **aws_access_key_id** 和 **aws_secret_access_key** 字段来指定您的 base64 编码的 AWS 凭证。例如：

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

2. 创建 secret.例如：

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

3. 创建或编辑定义 **ClusterLogForwarder** CR 对象的 YAML 文件。在文件中，指定 secret 的名称。例如：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> 1
  namespace: <log_forwarder_namespace> 2
spec:
  serviceAccountName: <service_account_name> 3
  outputs:
   - name: cw 4
     type: cloudwatch 5
     cloudwatch:
       groupBy: logType 6
```

```
      groupPrefix: <group prefix> 7
      region: us-east-2 8
    secret:
      name: cw-secret 9
  pipelines:
   - name: infra-logs 10
     inputRefs: 11
       - infrastructure
       - audit
       - application
     outputRefs:
       - cw 12
```

1. 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。
2. 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。
3. 服务帐户的名称。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。
4. 指定输出的名称。
5. 指定 **cloudwatch** 类型。
6. 可选：指定如何对日志进行分组：
   - **logType** 为每个日志类型创建日志组。
   - **namespaceName** 为每个应用程序命名空间创建一个日志组。它还会为基础架构和审计日志创建单独的日志组。
   - **namespaceUUID** 为每个应用命名空间 UUID 创建一个新的日志组。它还会为基础架构和审计日志创建单独的日志组。
7. 可选：指定一个字符串来替换日志组名称中的默认 **infrastructureName** 前缀。
8. 指定 AWS 区域。
9. 指定包含 AWS 凭证的 secret 名称。
10. 可选：指定管道的名称。
11. 使用管道指定要转发的日志类型：**application**、**infrastructure** 或 **audit**。
12. 指定使用此管道转发日志时使用的输出名称。

4. 创建 CR 对象。

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

示例：在 Amazon CloudWatch 中使用 ClusterLogForwarder

在这里，您会看到 **ClusterLogForwarder** 自定义资源 (CR) 示例及其输出到 Amazon CloudWatch 的日志数据。

假设您正在运行名为 **mycluster** 的 OpenShift Container Platform 集群。以下命令返回集群的 **infrastructureName**，稍后您将用它来编写 **aws** 命令：

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

要为本例生成日志数据，您可以在名为 **app** 的命名空间中运行 **busybox** pod。**busybox** pod 每隔三秒钟将消息写入 stdout：

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

您可以查找 **busybox** pod 运行的 **app** 命名空间的 UUID：

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

在 **ClusterLogForwarder** 自定义资源 (CR) 中，您可以将 **infrastructure**、**audit** 和 **application** 日志类型配置为 **all-logs** 管道的输入。您还可以将此管道连接到 **cw** 输出，输出将日志转发到 **us-east-2** 区域的 CloudWatch 实例：

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

CloudWatch 中的每个地区都包含三个级别的对象：

- 日志组

- 日志流
  - 日志事件

使用 **ClusterLogForwarding** CR 中的 **groupBy: logType**，**inputRefs** 中的三种日志类型会在 Amazon Cloudwatch 中生成三个日志组：

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

每个日志组都包含日志流：

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

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

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

每个日志流都包含日志事件。要查看 **busybox** Pod 的日志事件，您可以从 **application** 日志组中指定其日志流：

```
$ aws logs get-log-events --log-group-name mycluster-7977k.application --log-stream-name
kubernetes.var.log.containers.busybox_app_busybox-
da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76.log
{
    "events": [
        {
            "timestamp": 1629422704178,
            "message": "{\"docker\":
{\"container_id\":\"da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76\"},\"kub
ernetes\":
{\"container_name\":\"busybox\",\"namespace_name\":\"app\",\"pod_name\":\"busybox\",\"container_ima
ge\":\"docker.io/library/busybox:latest\",\"container_image_id\":\"docker.io/library/busybox@sha256:0f35
```

```
4ec1728d9ff32edcd7d1b8bbdfc798277ad36120dc3dc683be44524c8b60\",\"pod_id\":\"870be234-
90a3-4258-b73f-4f4d6e2777c7\",\"host\":\"ip-10-0-216-3.us-east-2.compute.internal\",\"labels\":
{\"run\":\"busybox\"},\"master_url\":\"https://kubernetes.default.svc\",\"namespace_id\":\"794e1e1a-
b9f5-4958-a190-e76a9b53d7bf\",\"namespace_labels\":
{\"kubernetes_io/metadata_name\":\"app\"}},\"message\":\"My life is my
message\",\"level\":\"unknown\",\"hostname\":\"ip-10-0-216-3.us-east-
2.compute.internal\",\"pipeline_metadata\":{\"collector\":
{\"ipaddr4\":\"10.0.216.3\",\"inputname\":\"fluent-plugin-
systemd\",\"name\":\"fluentd\",\"received_at\":\"2021-08-
20T01:25:08.085760+00:00\",\"version\":\"1.7.4 1.6.0\"}},\"@timestamp\":\"2021-08-
20T01:25:04.178986+00:00\",\"viaq_index_name\":\"app-
write\",\"viaq_msg_id\":\"NWRjZmUyMWQtZjgzNC00MjI4LTk3MjMtNTk3NmY3ZjU4NDk1\",\"log_type\":
\"application\",\"time\":\"2021-08-20T01:25:04+00:00\"}",
            "ingestionTime": 1629422744016
        },
...
```

示例：在日志组群名称中自定义前缀

在日志组名称中，您可以将默认的 **infrastructureName** 前缀 **mycluster-7977k** 替换为一个任意字符串，如 **demo-group-prefix**。要进行此更改，您需要更新 **ClusterLogForwarding** CR 中的 **groupPrefix** 字段：

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

groupPrefix 的值替换默认的 **infrastructureName** 前缀：

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

示例：应用程序命名空间名称后命名日志组

对于集群中的每个应用程序命名空间，您可以在 CloudWatch 中创建日志组，其名称基于应用程序命名空间的名称。

如果您删除应用程序命名空间对象并创建名称相同的新对象，CloudWatch 会继续使用与以前相同的日志组。

如果您认为名称相同的连续应用程序命名空间对象相互等效，请使用本例中描述的方法。否则，如果您需要将生成的日志组相互区分，请参阅以下"为应用命名空间 UUID 注入日志组"部分。

要创建名称基于应用程序命名空间名称的应用程序日志组，您可以在 **ClusterLogForwarder** CR 中将 **groupBy** 字段的值设置为 **namespaceName** ：

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

将 **groupBy** 设置为 **namespaceName** 只会影响应用程序日志组。它不会影响 **audit** 和 **infrastructure** 日志组。

在 Amazon Cloudwatch 中，命名空间名称显示在每个日志组名称的末尾。因为只有一个应用程序命名空间 "app"，以下输出显示一个新的 **mycluster-7977k.app** 日志组，而不是 **mycluster-7977k.application**：

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

如果本例中的集群包含多个应用命名空间，则输出中会显示多个日志组，每个命名空间对应一个日志组。

groupBy 字段仅影响应用日志组。它不会影响 **audit** 和 **infrastructure** 日志组。

示例：应用程序命名空间 UUID 后命名日志组

对于集群中的每个应用程序命名空间，您可以在 CloudWatch 中创建日志组，其名称是基于应用程序命名空间的 UUID。

如果您删除应用程序命名空间对象并创建新对象，CloudWatch 会创建一个新的日志组。

如果您考虑使用名称相同的连续应用程序命名空间对象，请使用本例中描述的方法。否则，请参阅前面的 "Example: Naming log groups for application namespace name" 部分。

要在应用程序命名空间 UUID 后命名日志组，您可以在 **ClusterLogForwarder** CR 中将 **groupBy** 字段的值设置为 **namespaceUUID**：

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

在 Amazon Cloudwatch 中，命名空间 UUID 出现在每个日志组名称的末尾。因为有一个应用程序命名空间 "app"，以下输出显示一个新的 **mycluster-7977k.794e1e1a-b9f5-4958-a190-e76a9b53d7bf** 日志组，而不是 **mycluster-7977k.application**：

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

groupBy 字段仅影响应用日志组。它不会影响 **audit** 和 **infrastructure** 日志组。

9.4.18. 使用现有 AWS 角色为 AWS CloudWatch 创建 secret

如果您有一个 AWS 的现有角色，您可以使用 **oc create secret --from-literal** 命令为 AWS 使用 STS 创建 secret。

流程

- 在 CLI 中，输入以下内容来为 AWS 生成 secret：

  ```
  $ oc create secret generic cw-sts-secret -n openshift-logging --from-
  literal=role_arn=arn:aws:iam::123456789012:role/my-role_with-permissions
  ```

  Secret 示例

  -

```
apiVersion: v1
kind: Secret
metadata:
  namespace: openshift-logging
  name: my-secret-name
stringData:
  role_arn: arn:aws:iam::123456789012:role/my-role_with-permissions
```

9.4.19. 从启用了 STS 的集群将日志转发到 Amazon CloudWatch

对于启用了 AWS Security Token Service (STS)的集群，您可以手动创建 AWS 服务帐户，或使用 Cloud Credential Operator (CCO) 实用程序 **ccoctl** 创建凭证请求。

先决条件

- Logging for Red Hat OpenShift: 5.5 及更新的版本

流程

1. 使用以下模板创建 **CredentialsRequest** 自定义资源 YAML：

   Cloudwatch 凭证请求模板

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <your_role_name>-credrequest
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
      - action:
          - logs:PutLogEvents
          - logs:CreateLogGroup
          - logs:PutRetentionPolicy
          - logs:CreateLogStream
          - logs:DescribeLogGroups
          - logs:DescribeLogStreams
        effect: Allow
        resource: arn:aws:logs:*:*:*
  secretRef:
    name: <your_role_name>
    namespace: openshift-logging
  serviceAccountNames:
    - logcollector
```

2. 使用 **ccoctl** 命令，使用 **CredentialsRequest** CR 为 AWS 创建角色。使用 **CredentialsRequest** 对象时，这个 **ccoctl** 命令会创建一个带有信任策略的 IAM 角色，该角色与指定的 OIDC 身份提供程序相关联，以及一个授予对 CloudWatch 资源执行操作的权限策略。此命令在 **/<path_to_ccoctl_output_dir>/manifests/openshift-logging-<your_role_name>-credentials.yaml** 中创建了一个 YAML 配置文件。此 secret 文件包含与 AWS IAM 身份提供程序进行身份验证时使用的 **role_arn** 键/值。

```
$ ccoctl aws create-iam-roles \
--name=<name> \
--region=<aws_region> \
--credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests \
--identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com (1)
```

(1) <name> 是用于标记云资源的名称，应该与 STS 集群安装过程中使用的名称匹配

3. 应用创建的 secret：

```
$ oc apply -f output/manifests/openshift-logging-<your_role_name>-credentials.yaml
```

4. 创建或编辑 **ClusterLogForwarder** 自定义资源：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: <log_forwarder_name> (1)
  namespace: <log_forwarder_namespace> (2)
spec:
  serviceAccountName: clf-collector (3)
  outputs:
   - name: cw (4)
     type: cloudwatch (5)
     cloudwatch:
       groupBy: logType (6)
       groupPrefix: <group prefix> (7)
       region: us-east-2 (8)
     secret:
        name: <your_secret_name> (9)
  pipelines:
    - name: to-cloudwatch (10)
      inputRefs: (11)
        - infrastructure
        - audit
        - application
      outputRefs:
        - cw (12)
```

(1) 在传统的实现中，CR 名称必须是 **instance**。在多日志转发器实现中，您可以使用任何名称。

(2) 在旧的实现中，CR 命名空间必须是 **openshift-logging**。在多日志转发器实现中，您可以使用任何命名空间。

(3) 指定 **clf-collector** 服务帐户。如果没有在 **openshift-logging** 命名空间中部署日志转发器，则只有多日志转发器实现中才需要服务帐户。

(4) 指定输出的名称。

5. 指定 **cloudwatch** 类型。
6. 可选：指定如何对日志进行分组：
   - **logType** 为每个日志类型创建日志组。
   - **namespaceName** 为每个应用程序命名空间创建一个日志组。基础架构和审计日志不受影响，剩余的日志按照 **logType** 分组。
   - **namespaceUUID** 为每个应用命名空间 UUID 创建一个新的日志组。它还会为基础架构和审计日志创建单独的日志组。
7. 可选：指定一个字符串来替换日志组名称中的默认 **infrastructureName** 前缀。
8. 指定 AWS 区域。
9. 指定包含 AWS 凭证的 secret 名称。
10. 可选：指定管道的名称。
11. 使用管道指定要转发的日志类型：**application**、**infrastructure** 或 **audit**。
12. 指定使用此管道转发日志时使用的输出名称。

其他资源

- AWS STS API 参考
- Cloud Credential Operator (CCO)

9.5. 配置日志记录收集器

Red Hat OpenShift 的 logging 从集群中收集操作和应用程序日志，并使用 Kubernetes pod 和项目元数据丰富数据。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** 小节来执行。

9.5.1. 配置日志收集器

您可以通过修改 **ClusterLogging** 自定义资源(CR)来配置日志使用哪个日志收集器类型。

注意

Fluentd 已被弃用，计划在以后的发行版本中删除。红帽将在当前发行生命周期中将提供对这个功能的 bug 修复和支持，但此功能将不再获得改进。作为 Fluentd 的替代选择，您可以使用 Vector。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 Red Hat OpenShift Logging Operator。

- 您已创建了 **ClusterLogging** CR。

流程

1. 修改 **ClusterLogging** CR **collection** 规格：

   ClusterLogging CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  collection:
    type: <log_collector_type> 1
    resources: {}
    tolerations: {}
# ...
```

   1 要用于日志记录的日志收集器类型。这可以是 **vector** 或 **fluentd**。

2. 运行以下命令来应用 **ClusterLogging** CR：

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

9.5.2. 创建 LogFileMetricExporter 资源

在日志记录版本 5.8 及更新版本中，LogFileMetricExporter 不再默认使用收集器部署。您需要手动创建一个 **LogFileMetricExporter** 自定义资源 (CR)，从运行容器生成的日志中生成指标。

如果没有创建 **LogFileMetricExporter** CR，您可能会在 OpenShift Container Platform Web 控制台仪表板中看到 **Produced Logs** 的 **No datapoints found** 信息。

先决条件

- 有管理员权限。
- 已安装 Red Hat OpenShift Logging Operator。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 创建一个 **LogFileMetricExporter** CR 作为 YAML 文件：

   Example LogFileMetricExporter CR

```
apiVersion: logging.openshift.io/v1alpha1
kind: LogFileMetricExporter
metadata:
  name: instance
  namespace: openshift-logging
```

```
spec:
  nodeSelector: {} (1)
  resources: (2)
    limits:
      cpu: 500m
      memory: 256Mi
    requests:
      cpu: 200m
      memory: 128Mi
  tolerations: [] (3)
# ...
```

1. 可选： **nodeSelector** 小节定义 pod 调度到哪些节点上。
2. **resources** 小节定义了 **LogFileMetricExporter** CR 的资源要求。
3. 可选：**tolerations** 小节定义 pod 接受的容限。

2. 运行以下命令来应用 **LogFileMetricExporter** CR：

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

验证

logfilesmetricexporter pod 在每个节点上同时运行 **collector** pod。

- 运行以下命令，验证 **logfilesmetricexporter** pod 是否在您创建 **LogFileMetricExporter** CR 的命名空间中运行：

```
$ oc get pods -l app.kubernetes.io/component=logfilesmetricexporter -n openshift-logging
```

输出示例

```
NAME                          READY   STATUS    RESTARTS   AGE
logfilesmetricexporter-9qbjj   1/1     Running   0          2m46s
logfilesmetricexporter-cbc4v   1/1     Running   0          2m46s
```

9.5.3. 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

- 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:
    type: 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。

9.5.4. 配置输入接收器

Red Hat OpenShift Logging Operator 为每个配置的输入接收器部署服务，以便客户端可以写入收集器。此服务公开为输入接收器指定的端口。服务名称基于以下内容生成：

- 对于多日志转发器 **ClusterLogForwarder** CR 部署，服务名称格式为 **<ClusterLogForwarder_CR_name>-<input_name>**。例如，**example-http-receiver**。
- 对于旧的 **ClusterLogForwarder** CR 部署，这意味着名为 **instance** 且位于 **openshift-logging** 命名空间中，服务名称采用 **collector-<input_name>** 格式。例如，**collector-http-receiver**。

9.5.4.1. 配置收集器以接收审计日志作为 HTTP 服务器

您可以通过在 **ClusterLogForwarder** 自定义资源(CR)中将 **http** 指定为接收器输入，将日志收集器配置为侦听 HTTP 连接，并将审计日志作为 HTTP 服务器接收。这可让您对从 OpenShift Container Platform 集群内部和外部收集的审计日志使用通用日志存储。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 Red Hat OpenShift Logging Operator。
- 您已创建了 **ClusterLogForwarder** CR。

流程

1. 修改 **ClusterLogForwarder** CR，以添加 **http** 接收器输入的配置：

   使用多日志转发器部署的 ClusterLogForwarder CR 示例

```
apiVersion: logging.openshift.io/v1beta1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  serviceAccountName: <service_account_name>
  inputs:
    - name: http-receiver 1
```

```
    receiver:
      type: http 2
      http:
        format: kubeAPIAudit 3
        port: 8443 4
  pipelines: 5
    - name: http-pipeline
      inputRefs:
        - http-receiver
# ...
```

1 为您的输入接收器指定一个名称。

2 将输入接收器类型指定为 **http**。

3 目前，**http** 输入接收器只支持 **kube-apiserver** Webhook 格式。

4 可选：指定输入接收器侦听的端口。这必须是 **1024** 到 **65535** 之间的值。如果没有指定，则默认值为 **8443**。

5 为您的输入接收器配置管道。

使用旧部署的 ClusterLogForwarder CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  inputs:
    - name: http-receiver 1
      receiver:
        type: http 2
        http:
          format: kubeAPIAudit 3
          port: 8443 4
  pipelines: 5
  - inputRefs:
    - http-receiver
    name: http-pipeline
# ...
```

1 为您的输入接收器指定一个名称。

2 将输入接收器类型指定为 **http**。

3 目前，**http** 输入接收器只支持 **kube-apiserver** Webhook 格式。

4 可选：指定输入接收器侦听的端口。这必须是 **1024** 到 **65535** 之间的值。如果没有指定，则默认值为 **8443**。

5 为您的输入接收器配置管道。

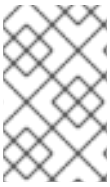
2. 运行以下命令，将更改应用到 **ClusterLogForwarder** CR：

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

其他资源

- API 审计过滤器概述

9.5.5. Fluentd 日志转发器的高级配置



注意

Fluentd 已被弃用，计划在以后的发行版本中删除。红帽将在当前发行生命周期中将提供对这个功能的 bug 修复和支持，但此功能将不再获得改进。作为 Fluentd 的替代选择，您可以使用 Vector。

日志记录包括多个 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：

- 块和块缓冲大小
- 块清除行为
- 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。

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 9.11. 高级 Fluentd 配置参数

参数	描述	默认
chunkLimitSize	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。	**8m**
totalLimitSize	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。	大约 15% 的节点磁盘分布在所有输出中。
flushInterval	块清除之间的间隔。您可以使用 **s**（秒）、**m**（分钟）、**h**（小时）或 **d** （天）。	**1s**
flushMode	执行清除的方法： • **lazy**:基于 **timekey** 参数对块进行清理。您无法修改 **timekey** 参数。 • **interval**：基于 **flushInterval** 参数清理块。 • **Immediate**: 在将数据添加到一个块后马上清理块。	**interval**
flushThreadCount	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。	**2**
overflowAction	当队列满时块的行为： • **throw_exception**：发出一个异常并在日志中显示。 • **block**：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。 • **drop_oldest_chunk**：删除旧的块以接受新传入的块。旧块的价值比新块要小。	**block**
retryMaxInterval	**exponential_backoff** 重试方法的最大时间（以秒为单位）。	**300s**

参数	描述	默认
retryType	flushing 失败时重试的方法： • **exponential_backoff** ：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 **retry_max_interval** 参数指定的值。 • **periodic**：基于 **retryWait** 参数，定期重试清理操作。	**exponential_backoff**
retryTimeOut	在放弃记录前尝试重试的最长时间。	**60m**
retryWait	下一次块清除前的时间（以秒为单位）。	**1s**

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:
  collection:
    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 -l component=collector -n openshift-logging
```

4. 检查 **fluentd** 配置映射中的新值：

```
$ oc extract configmap/collector-config --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 "#{ENV['TOTAL_LIMIT_SIZE_PER_BUFFER'] || '8589934592'}"
  chunk_limit_size 8m
  overflow_action throw_exception
  disable_chunk_backup true
</buffer>
```

9.6. 收集并存储 KUBERNETES 事件

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

Event Router 从所有项目收集事件，并将其写入 **STDOUT**。然后，收集器将这些事件转发到 **ClusterLogForwarder** 自定义资源(CR)中定义的存储。

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。

9.6.1. 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。

注意

Event Router 镜像不是 Red Hat OpenShift Logging Operator 的一部分，必须单独下载。

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编辑模板来更改部署对象 CPU 和内存请求。

先决条件

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

流程

1. 为事件路由器创建模板：

```
apiVersion: template.openshift.io/v1
kind: Template
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
            securityContext:
              allowPrivilegeEscalation: false
              capabilities:
                drop: ["ALL"]
        securityContext:
```

```
          runAsNonRoot: true
          seccompProfile:
            type: RuntimeDefault
        volumes:
        - name: config-volume
          configMap:
            name: eventrouter
parameters:
  - name: IMAGE 6
    displayName: Image
    value: "registry.redhat.io/openshift-logging/eventrouter-rhel9:v0.4"
  - 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 **v0.4**。
7. 指定分配给事件路由器 pod 的最小 CPU 量。默认值为**100m**。
8. 指定分配给事件路由器 pod 的最小内存量。默认值为**128Mi**。
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.rbac.authorization.k8s.io/event-reader created
clusterrolebinding.rbac.authorization.k8s.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-
   manager-remover.1632d931e88fcd8f","namespace":"openshift-service-catalog-
   removed","selfLink":"/api/v1/namespaces/openshift-service-catalog-
   removed/events/openshift-service-catalog-controller-manager-
   remover.1632d931e88fcd8f","uid":"787d7b26-3d2f-4017-b0b0-
   420db4ae62c0","resourceVersion":"21399","creationTimestamp":"2020-09-
   08T15:40:26Z"},"involvedObject":{"kind":"Job","namespace":"openshift-service-catalog-
   removed","name":"openshift-service-catalog-controller-manager-
   remover","uid":"fac9f479-4ad5-4a57-8adc-
   cb25d3d9cf8f","apiVersion":"batch/v1","resourceVersion":"21280"},"reason":"Completed","
   message":"Job completed","source":{"component":"job-
   controller"},"firstTimestamp":"2020-09-08T15:40:26Z","lastTimestamp":"2020-09-
   08T15:40:26Z","count":1,"type":"Normal"}}
   ```

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

第 10 章 日志存储

10.1. 关于日志存储

您可以使用集群中的内部 Loki 或 Elasticsearch 日志存储来存储日志，也可以使用 **ClusterLogForwarder** 自定义资源(CR) 将日志转发到外部存储。

10.1.1. 日志存储类型

Loki 是一个可横向扩展的、高度可用且多租户的日志聚合系统，目前作为日志记录的日志存储提供。

Elasticsearch 在 ingestion 过程中完全索引传入的日志记录。Loki 仅在 ingestion 过程中索引几个固定标签，并延迟更复杂的解析，直到存储日志为止。这意味着 Loki 可以更快地收集日志。

10.1.1.1. 关于 Elasticsearch 日志存储

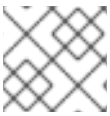
日志记录 Elasticsearch 实例经过优化并测试，用于大约 7 天的简短存储。如果要更长时间保留日志，建议您将数据移至第三方存储系统。

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。

10.1.2. 查询日志存储

您可以使用 LogQL 日志查询语言查询 Loki。

10.1.3. 其他资源

- Loki 组件文档
- Loki Object Storage 文档

10.2. 安装日志存储

您可以使用 OpenShift CLI (**oc**)或 OpenShift Container Platform Web 控制台在 OpenShift Container Platform 集群上部署日志存储。

注意

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。如果您目前使用随 Logging 5.8 发布的 OpenShift Elasticsearch Operator，它将继续使用 Logging，直到 Logging 5.8 的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

10.2.1. 部署 Loki 日志存储

您可以使用 Loki Operator 在 OpenShift Container Platform 集群中部署内部 Loki 日志存储。安装 Loki Operator 后，您必须通过创建一个 secret 来配置 Loki 对象存储，并创建一个 **LokiStack** 自定义资源 (CR)。

10.2.1.1. Loki 部署大小

Loki 的大小使用 **1x.<size>** 格式，其中值 **1x** 是实例数量，**<size>** 指定性能功能。

重要

对于部署大小，无法更改 **1x** 值。

表 10.1. Loki 大小

	1x.demo	1x.extra-small	1x.small	1x.medium
数据传输	仅用于演示	100GB/day	500GB/day	2TB/day
每秒查询数 (QPS)	仅用于演示	1-25 QPS at 200ms	25-50 QPS at 200ms	25-75 QPS at 200ms
复制因子	None	2	2	2
总 CPU 请求	None	14 个 vCPU	34 个 vCPU	54 个 vCPU
使用标尺的 CPU 请求总数	None	16 个 vCPU	42 个 vCPU	70 个 vCPU
内存请求总数	None	31Gi	67Gi	139Gi
使用规则器的内存请求总数	None	35Gi	83Gi	171Gi
磁盘请求总数	40Gi	430Gi	430Gi	590Gi

	1x.demo	1x.extra-small	1x.small	1x.medium
使用标尺的磁盘请求总数	80Gi	750Gi	750Gi	910Gi

10.2.1.2. 使用 OpenShift Container Platform Web 控制台安装 Loki Operator

要在 OpenShift Container Platform 集群中安装和配置日志记录，必须安装额外的 Operator。这可以通过 web 控制台中的 Operator Hub 完成。

OpenShift Container Platform Operator 使用自定义资源(CR)来管理应用程序及其组件。高级配置和设置由 CR 中的用户提供。Operator 根据 Operator 逻辑中嵌入的最佳实践，将高级别指令转换为低级操作。自定义资源定义(CRD)定义了一个 CR，并列出 Operator 用户可用的所有配置。安装 Operator 会创建 CRD，然后用于生成 CR。

先决条件

- 您可以访问受支持的对象存储 (AWS S3、Google Cloud Storage、Azure、Swift、Minio、OpenShift Data Foundation)。
- 有管理员权限。
- 访问 OpenShift Container Platform web 控制台。

流程

1. 在 OpenShift Container Platform Web 控制台 **Administrator** 视角中，进入 **Operators → OperatorHub**。
2. 在 **Filter by keyword** 字段中输入 Loki Operator。点可用 Operator 列表中的 **Loki Operator**，然后点 **Install**。

重要

红帽不支持 Community Loki Operator。

3. 选择 **stable** 或 **stable-x.y** 作为 **更新频道**。

注意

stable 频道只为日志记录的最新版本提供更新。要继续获得之前版本的更新，您必须将订阅频道改为 **stable-x.y**，其中 **x.y** 代表您安装的日志记录的主版本和次版本。例如，**stable-5.7**。

Loki Operator 必须部署到全局 operator 组命名空间 **openshift-operators-redhat**，因此已选择了 **Installation mode** 和 **Installed Namespace**。如果此命名空间不存在，则会为您创建它。

4. 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"** 标签。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。
5. 对于 **Update approval**，请选择 **Automatic**，然后点 **Install**。

如果订阅中的批准策略被设置为 **Automatic**，则更新过程会在所选频道中提供新的 Operator 版本时立即启动。如果批准策略设为 **Manual**，则必须手动批准待处理的更新。

验证

1. 进入 **Operators → Installed Operators**。
2. 确保已选中 **openshift-logging** 项目。
3. 在 **Status** 列中，验证您看到了绿色的对勾标记，并为 **InstallSucceeded**，文本 **Up to date**。

注意

Operator 可能会在安装完成前显示 **Failed** 状态。如果 Operator 安装完成并显示 **InstallSucceeded** 信息，请刷新页面。

10.2.1.3. 使用 Web 控制台为 Loki 对象存储创建 secret

要配置 Loki 对象存储，您必须创建一个 secret。您可以使用 OpenShift Container Platform Web 控制台创建 secret。

先决条件

- 有管理员权限。
- 访问 OpenShift Container Platform web 控制台。
- 已安装 Loki Operator。

流程

1. 在 OpenShift Container Platform Web 控制台的 **Administrator** 视角中进入 **Workloads → Secrets**。
2. 从 **Create** 下拉列表中选择 **From YAML**。
3. 创建一个 secret，它使用 **access_key_id** 和 **access_key_secret** 字段指定您的凭证和 **bucketnames**、**endpoint** 和 **region** 字段来定义对象存储位置。AWS 在以下示例中使用：

Secret 对象示例

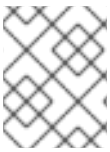
```
apiVersion: v1
kind: Secret
metadata:
  name: logging-loki-s3
  namespace: openshift-logging
stringData:
  access_key_id: AKIAIOSFODNN7EXAMPLE
  access_key_secret: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
  bucketnames: s3-bucket-name
  endpoint: https://s3.eu-central-1.amazonaws.com
  region: eu-central-1
```

其他资源

- Loki 对象存储

10.2.2. 在使用短期凭证的集群中部署 Loki 日志存储

对于某些存储供应商，您可以在安装过程中使用 CCO 实用程序(**ccoctl**) 来实现短期凭证。这些凭证在 OpenShift Container Platform 集群外创建和管理。组件带有简短凭证的手动模式。



注意

在使用此凭证策略的集群上，必须在 Loki Operator 的新安装过程中配置短期凭证身份验证。您无法重新配置使用不同凭证策略的现有集群，以使用此功能。

10.2.2.1. 工作负载身份联邦

工作负载身份联邦允许使用简短的令牌对基于云的日志存储进行身份验证。

先决条件

- OpenShift Container Platform 4.14 及更新的版本
- 日志记录 5.9 及更新的版本

流程

- 如果使用 OpenShift Container Platform Web 控制台安装 Loki Operator，则会自动检测到使用简短令牌的集群。系统将提示您创建角色，并提供 Loki Operator 所需的数据，以创建 **CredentialsRequest** 对象，该对象填充 secret。
- 如果使用 OpenShift CLI (**oc**) 安装 Loki Operator，则必须使用存储供应商的适当模板手动创建订阅对象，如下例所示。此身份验证策略只支持所示的存储供应商。

Azure 订阅示例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat
spec:
  channel: "stable-5.9"
  installPlanApproval: Manual
  name: loki-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  config:
    env:
      - name: CLIENTID
        value: <your_client_id>
      - name: TENANTID
        value: <your_tenant_id>
      - name: SUBSCRIPTIONID
        value: <your_subscription_id>
      - name: REGION
        value: <your_region>
```

AWS 订阅示例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat
spec:
  channel: "stable-5.9"
  installPlanApproval: Manual
  name: loki-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  config:
    env:
    - name: ROLEARN
      value: <role_ARN>
```

10.2.2.2. 使用 Web 控制台创建 LokiStack 自定义资源

您可以使用 OpenShift Container Platform Web 控制台创建 **LokiStack** 自定义资源(CR)。

先决条件

- 有管理员权限。
- 访问 OpenShift Container Platform web 控制台。
- 已安装 Loki Operator。

流程

1. 进入 **Operators → Installed Operators** 页面。点 **All instances** 选项卡。
2. 在 **Create new** 下拉列表中，选择 **LokiStack**。
3. 选择 **YAML 视图**，然后使用以下模板来创建 **LokiStack** CR：

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki (1)
  namespace: openshift-logging
spec:
  size: 1x.small (2)
  storage:
    schemas:
      - effectiveDate: '2023-10-15'
        version: v13
    secret:
      name: logging-loki-s3 (3)
      type: s3 (4)
      credentialMode: (5)
```

```
    storageClassName: <storage_class_name> 6
  tenants:
    mode: openshift-logging
```

1. 使用名称 **logging-loki**。
2. 指定部署大小。在日志记录 5.8 及更新的版本中，Loki 实例支持的大小选项为 **1x.extra-small**、**1x.small** 或 **1x.medium**。
3. 指定用于日志存储的 secret。
4. 指定对应的存储类型。
5. 可选字段，日志记录 5.9 及更新的版本。支持的用户配置值如下：**static** 是所有受支持的对象存储类型的默认身份验证模式，使用存储在 Secret 中的凭证。**token** 是从凭证源检索的短期令牌。在这个模式中，静态配置不包含对象存储所需的凭证。相反，它们会使用服务在运行时生成，允许提供较短的凭证，以及更精细的控制。并不是所有对象存储类型都支持这个身份验证模式。当 Loki 在受管 STS 模式下运行并使用 CCO on STS/WIF 集群时，**token-cco** 是默认值。
6. 为临时存储输入存储类的名称。为获得最佳性能，请指定分配块存储的存储类。可以使用 **oc get storageclasses** 命令列出集群的可用存储类。

10.2.2.3. 使用 CLI 安装 Loki Operator

要在 OpenShift Container Platform 集群中安装和配置日志记录，必须安装额外的 Operator。这可以通过 OpenShift Container Platform CLI 完成。

OpenShift Container Platform Operator 使用自定义资源(CR)来管理应用程序及其组件。高级配置和设置由 CR 中的用户提供。Operator 根据 Operator 逻辑中嵌入的最佳实践，将高级别指令转换为低级操作。自定义资源定义(CRD)定义了一个 CR，并列出 Operator 用户可用的所有配置。安装 Operator 会创建 CRD，然后用于生成 CR。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 您可以访问受支持的对象存储。例如：AWS S3、Google Cloud Storage、Azure、Swift、Minio 或 OpenShift Data Foundation。

流程

1. 创建 **Subscription** 对象：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: loki-operator
  namespace: openshift-operators-redhat 1
spec:
  channel: stable 2
```

```
name: loki-operator
source: redhat-operators 3
sourceNamespace: openshift-marketplace
```

1. 您必须指定 **openshift-operators-redhat** 命名空间。
2. 指定 **stable**, 或 **stable-5.<y>** 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。

2. 应用 **Subscription** 对象：

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

10.2.2.4. 使用 CLI 为 Loki 对象存储创建 secret

要配置 Loki 对象存储，您必须创建一个 secret。您可以使用 OpenShift CLI (**oc**)完成此操作。

先决条件

- 有管理员权限。
- 已安装 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。

流程

- 运行以下命令，在包含您的证书和密钥文件的目录中创建 secret：

```
$ oc create secret generic -n openshift-logging <your_secret_name> \
 --from-file=tls.key=<your_key_file>
 --from-file=tls.crt=<your_crt_file>
 --from-file=ca-bundle.crt=<your_bundle_file>
 --from-literal=username=<your_username>
 --from-literal=password=<your_password>
```

注意

使用通用或 opaque secret 以获得最佳结果。

验证

- 运行以下命令验证 secret 是否已创建：

```
$ oc get secrets
```

其他资源

- Loki 对象存储

10.2.2.5. 使用 CLI 创建 LokiStack 自定义资源

您可以使用 OpenShift CLI (**oc**)创建 **LokiStack** 自定义资源(CR)。

先决条件

- 有管理员权限。
- 已安装 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 创建 **LokiStack** CR：

LokiStack CR 示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki (1)
  namespace: openshift-logging
spec:
  size: 1x.small (2)
  storage:
    schemas:
      - effectiveDate: '2023-10-15'
        version: v13
    secret:
      name: logging-loki-s3 (3)
      type: s3 (4)
      credentialMode: (5)
  storageClassName: <storage_class_name> (6)
  tenants:
    mode: openshift-logging
```

(1) 使用名称 **logging-loki**。

(2) 指定部署大小。在日志记录 5.8 及更新的版本中，Loki 实例支持的大小选项为 **1x.extra-small**、**1x.small** 或 **1x.medium**。

(3) 指定用于日志存储的 secret。

(4) 指定对应的存储类型。

(5) 可选字段，日志记录 5.9 及更新的版本。支持的用户配置值如下：**static** 是所有受支持的对象存储类型的默认身份验证模式，使用存储在 Secret 中的凭证。**token** 是从凭证源检索的短期令牌。在这个模式中，静态配置不包含对象存储所需的凭证。相反，它们会使用服务在运行时生成，允许提供较短的凭证，以及更精细的控制。并不是所有对象存储类型都支持这个身份验证模式。当 Loki 在受管 STS 模式下运行并使用 CCO on STS/WIF 集群时，**token-cco** 是默认值。

(6) 为临时存储输入存储类的名称。为获得最佳性能，请指定分配块存储的存储类。可以使用 **oc get storageclasses** 命令列出集群的可用存储类。

1. 运行以下命令来应用 **LokiStack** CR：

验证

- 运行以下命令并查看输出，在 **openshift-logging** 项目中列出 pod 来验证安装：

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

确认您看到了多个用于日志记录组件的 pod，类似于以下列表：

输出示例

```
NAME                                           READY   STATUS    RESTARTS   AGE
cluster-logging-operator-78fddc697-mnl82       1/1     Running   0          14m
collector-6cglq                                2/2     Running   0          45s
collector-8r664                                2/2     Running   0          45s
collector-8z7px                                2/2     Running   0          45s
collector-pdxl9                                2/2     Running   0          45s
collector-tc9dx                                2/2     Running   0          45s
collector-xkd76                                2/2     Running   0          45s
logging-loki-compactor-0                       1/1     Running   0          8m2s
logging-loki-distributor-b85b7d9fd-25j9g       1/1     Running   0          8m2s
logging-loki-distributor-b85b7d9fd-xwjs6       1/1     Running   0          8m2s
logging-loki-gateway-7bb86fd855-hjhl4          2/2     Running   0          8m2s
logging-loki-gateway-7bb86fd855-qjtlb          2/2     Running   0          8m2s
logging-loki-index-gateway-0                   1/1     Running   0          8m2s
logging-loki-index-gateway-1                   1/1     Running   0          7m29s
logging-loki-ingester-0                        1/1     Running   0          8m2s
logging-loki-ingester-1                        1/1     Running   0          6m46s
logging-loki-querier-f5cf9cb87-9fdjd           1/1     Running   0          8m2s
logging-loki-querier-f5cf9cb87-fp9v5           1/1     Running   0          8m2s
logging-loki-query-frontend-58c579fcb7-lfvbc   1/1     Running   0          8m2s
logging-loki-query-frontend-58c579fcb7-tjf9k   1/1     Running   0          8m2s
logging-view-plugin-79448d8df6-ckgmx           1/1     Running   0          46s
```

10.2.3. Loki 对象存储

Loki Operator 支持 AWS S3，以及 Minio 和 OpenShift Data Foundation 等其他 S3 兼容对象存储。Azure、GCS 和 Swift 也支持。

对于 Loki 存储，推荐的 nomenclature 是 **logging-loki-*<your_storage_provider>***。

下表显示了每个存储供应商 **LokiStack** 自定义资源(CR) 中的 **type** 值。如需更多信息，请参阅存储供应商部分。

表 10.2. Secret 类型快速参考

存储供应商	Secret **type** 值
AWS	s3
Azure	azure

存储供应商	Secret type 值
Google Cloud	gcs
Minio	s3
OpenShift Data Foundation	s3
Swift	swift

10.2.3.1. AWS 存储

先决条件

- 已安装 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。
- 您在 AWS 上创建了存储桶。
- 您创建了 AWS IAM 策略和 IAM 用户。

流程

- 运行以下命令，创建一个名为 **logging-loki-aws** 的对象存储 secret：

```
$ oc create secret generic logging-loki-aws \
  --from-literal=bucketnames="<bucket_name>" \
  --from-literal=endpoint="<aws_bucket_endpoint>" \
  --from-literal=access_key_id="<aws_access_key_id>" \
  --from-literal=access_key_secret="<aws_access_key_secret>" \
  --from-literal=region="<aws_region_of_your_bucket>"
```

10.2.3.1.1. 启用 STS 的集群的 AWS 存储

如果您的集群启用了 STS，Cloud Credential Operator (CCO) 支持使用 AWS 令牌进行短期身份验证。

您可以运行以下命令来手动创建 Loki 对象存储 secret：

```
$ oc -n openshift-logging create secret generic "logging-loki-aws" \
--from-literal=bucketnames="<s3_bucket_name>" \
--from-literal=region="<bucket_region>" \
--from-literal=audience="<oidc_audience>" 1
```

1 可选注解，默认值为 **openshift**。

10.2.3.2. Azure 存储

先决条件

- 已安装 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。
- 您在 Azure 上创建了存储桶。

流程

- 运行以下命令，使用名称 **logging-loki-azure** 创建对象存储 secret：

```
$ oc create secret generic logging-loki-azure \
  --from-literal=container="<azure_container_name>" \
  --from-literal=environment="<azure_environment>" \ 1
  --from-literal=account_name="<azure_account_name>" \
  --from-literal=account_key="<azure_account_key>"
```

1 支持的环境值包括 **AzureGlobal**、**AzureChinaCloud**、**AzureGermanCloud** 或 **AzureUSGovernment**。

10.2.3.2.1. 为 Microsoft Entra Workload ID 启用集群的 Azure 存储

如果您的集群启用了 Microsoft Entra Workload ID，Cloud Credential Operator (CCO) 支持使用 Workload ID 进行短期身份验证。

您可以运行以下命令来手动创建 Loki 对象存储 secret：

```
$ oc -n openshift-logging create secret generic logging-loki-azure \
--from-literal=environment="<azure_environment>" \
--from-literal=account_name="<storage_account_name>" \
--from-literal=container="<container_name>"
```

10.2.3.3. Google Cloud Platform 存储

先决条件

- 已安装 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。
- 您在 Google Cloud Platform (GCP)上创建了一个 项目。
- 您在同一项目中创建了存储桶。
- 您在用于 GCP 身份验证的同一项目中创建了服务帐户。

流程

1. 将从 GCP 接收的服务帐户凭证复制到名为 **key.json** 的文件中。
2. 运行以下命令，使用名称 **logging-loki-gcs** 创建对象存储 secret：

```
$ oc create secret generic logging-loki-gcs \
  --from-literal=bucketname="<bucket_name>" \
  --from-file=key.json="<path/to/key.json>"
```

10.2.3.4. Minio 存储

先决条件

- 已安装 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。
- 在集群中部署了 Minio。
- 您在 Minio 上创建了存储桶。

流程

- 运行以下命令，创建一个名为 **logging-loki-minio** 的对象存储 secret：

```
$ oc create secret generic logging-loki-minio \
  --from-literal=bucketnames="<bucket_name>" \
  --from-literal=endpoint="<minio_bucket_endpoint>" \
  --from-literal=access_key_id="<minio_access_key_id>" \
  --from-literal=access_key_secret="<minio_access_key_secret>"
```

10.2.3.5. OpenShift Data Foundation 存储

先决条件

- 已安装 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。
- 您已部署了 OpenShift Data Foundation。
- 为对象存储配置了 OpenShift Data Foundation 集群。

流程

1. 在 **openshift-logging** 命名空间中创建 **ObjectBucketClaim** 自定义资源：

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: loki-bucket-odf
  namespace: openshift-logging
spec:
  generateBucketName: loki-bucket-odf
  storageClassName: openshift-storage.noobaa.io
```

2. 运行以下命令，从关联的 **ConfigMap** 对象获取存储桶属性：

```
BUCKET_HOST=$(oc get -n openshift-logging configmap loki-bucket-odf -o
jsonpath='{.data.BUCKET_HOST}')
BUCKET_NAME=$(oc get -n openshift-logging configmap loki-bucket-odf -o
jsonpath='{.data.BUCKET_NAME}')
BUCKET_PORT=$(oc get -n openshift-logging configmap loki-bucket-odf -o
jsonpath='{.data.BUCKET_PORT}')
```

3. 运行以下命令，从关联的 secret 获取存储桶访问密钥：

```
ACCESS_KEY_ID=$(oc get -n openshift-logging secret loki-bucket-odf -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 -d)
SECRET_ACCESS_KEY=$(oc get -n openshift-logging secret loki-bucket-odf -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 -d)
```

4. 运行以下命令，创建一个名为 **logging-loki-odf** 的对象存储 secret：

```
$ oc create -n openshift-logging secret generic logging-loki-odf \
--from-literal=access_key_id="<access_key_id>" \
--from-literal=access_key_secret="<secret_access_key>" \
--from-literal=bucketnames="<bucket_name>" \
--from-literal=endpoint="https://<bucket_host>:<bucket_port>"
```

10.2.3.6. Swift 存储

先决条件

- 已安装 Loki Operator。
- 已安装 OpenShift CLI（**oc**）。
- 您在 Swift 上创建了一个存储桶。

流程

- 运行以下命令，创建一个名为 **logging-loki-swift** 的对象存储 secret：

```
$ oc create secret generic logging-loki-swift \
  --from-literal=auth_url="<swift_auth_url>" \
  --from-literal=username="<swift_usernameclaim>" \
  --from-literal=user_domain_name="<swift_user_domain_name>" \
  --from-literal=user_domain_id="<swift_user_domain_id>" \
  --from-literal=user_id="<swift_user_id>" \
  --from-literal=password="<swift_password>" \
  --from-literal=domain_id="<swift_domain_id>" \
  --from-literal=domain_name="<swift_domain_name>" \
  --from-literal=container_name="<swift_container_name>"
```

- 您可以通过运行以下命令来提供项目特定数据、区域或两者：

```
$ oc create secret generic logging-loki-swift \
  --from-literal=auth_url="<swift_auth_url>" \
  --from-literal=username="<swift_usernameclaim>" \
  --from-literal=user_domain_name="<swift_user_domain_name>" \
```

```
--from-literal=user_domain_id="<swift_user_domain_id>" \
--from-literal=user_id="<swift_user_id>" \
--from-literal=password="<swift_password>" \
--from-literal=domain_id="<swift_domain_id>" \
--from-literal=domain_name="<swift_domain_name>" \
--from-literal=container_name="<swift_container_name>" \
--from-literal=project_id="<swift_project_id>" \
--from-literal=project_name="<swift_project_name>" \
--from-literal=project_domain_id="<swift_project_domain_id>" \
--from-literal=project_domain_name="<swift_project_domain_name>" \
--from-literal=region="<swift_region>"
```

10.2.4. 部署 Elasticsearch 日志存储

您可以使用 OpenShift Elasticsearch Operator 在 OpenShift Container Platform 集群上部署内部 Elasticsearch 日志存储。

注意

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。如果您目前使用随 Logging 5.8 发布的 OpenShift Elasticsearch Operator，它将继续使用 Logging，直到 Logging 5.8 的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

10.2.4.1. Elasticsearch 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，这使用持久性卷声明(PVC)来实现。

注意

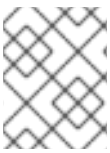
如果将本地卷用于持久性存储，请不要使用原始块卷，这在 **LocalVolume** 对象中的 **volumeMode: block** 描述。Elasticsearch 无法使用原始块卷。

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

Fluentd 将 **systemd journal** 和 **/var/log/containers/*.log** 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。

10.2.4.2. 使用 Web 控制台安装 OpenShift Elasticsearch Operator

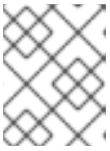
OpenShift Elasticsearch Operator 会创建和管理 OpenShift Logging 使用的 Elasticsearch 集群。

先决条件

- Elasticsearch 是内存密集型应用程序。每个 Elasticsearch 节点都需要至少 16GB 内存来满足内存请求和限值的需要，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 64GB 个内存。

  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。

- 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。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 从可用的 Operator 列表中选择 **OpenShift Elasticsearch Operator**，然后点 **Install**。
3. 确保在 **Installation mode** 下选择了 **All namespaces on the cluster**。
4. 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。
5. 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"** 标签。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。
6. 选择 **stable-5.x** 作为 **更新频道**。
7. 选择一个 **更新批准**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。
8. 点 **Install**。

验证

1. 通过切换到 **Operators → Installed Operators** 页来验证 OpenShift Elasticsearch Operator 已被安装。
2. 确定 **OpenShift Elasticsearch Operator** 在所有项目中被列出，请 **Status** 为 **Succeeded**。

10.2.4.3. 使用 CLI 安装 OpenShift Elasticsearch Operator

您可以使用 OpenShift CLI (**oc**)安装 OpenShift Elasticsearch Operator。

先决条件

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

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。

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 创建一个 **Namespace** 对象作为一个 YAML 文件：

```
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。要防止可能与指标冲突，请将 Prometheus Cluster Monitoring 堆栈配置为从 **openshift-operators-redhat** 命名空间中提取指标，而不是从 **openshift-operators** 命名空间中提取。**openshift-operators** 命名空间可能会包含社区 Operator，这些 Operator 不被信任，并可能会发布与 metric 的名称相同的指标，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。

2. 运行以下命令来应用 **Namespace** 对象：

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

3. 以 YAML 文件形式创建 **OperatorGroup** 对象：

```
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。

4. 运行以下命令来应用 **OperatorGroup** 对象：

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

5. 创建一个 **Subscription** 对象来订阅 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 1
spec:
  channel: stable-x.y 2
  installPlanApproval: Automatic 3
  source: redhat-operators 4
  sourceNamespace: openshift-marketplace
  name: elasticsearch-operator
```

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

2 指定 **stable**, 或 **stable-x.y** 作为频道。请参见以下注释。

3 **Automatic** 允许 Operator Lifecycle Manager (OLM) 在有新版本可用时自动更新 Operator。**Manual** 要求具有适当凭证的用户批准 Operator 更新。

4 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版本。

指定 **stable-x.y** 会安装特定主版本的当前次版本。使用带有 **installPlanApproval: "Automatic"** 的 **stable-x.y** 会自动将 Operator 升级到主发行版本中的最新稳定次版本。

6. 运行以下命令来应用订阅：

```
$ oc apply -f <filename>.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。

验证

1. 运行以下命令:

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

2. 观察输出，并确认每个命名空间中存在 OpenShift Elasticsearch Operator 的 Pod

输出示例

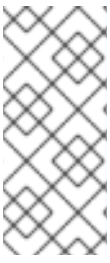
```
NAMESPACE                                NAME                  DISPLAY
VERSION        REPLACES                 PHASE
default                                  elasticsearch-operator.v5.8.1   OpenShift Elasticsearch
Operator   5.8.1          elasticsearch-operator.v5.8.0   Succeeded
kube-node-lease                          elasticsearch-operator.v5.8.1   OpenShift
Elasticsearch Operator   5.8.1          elasticsearch-operator.v5.8.0   Succeeded
kube-public                              elasticsearch-operator.v5.8.1   OpenShift Elasticsearch
Operator   5.8.1          elasticsearch-operator.v5.8.0   Succeeded
kube-system                              elasticsearch-operator.v5.8.1   OpenShift Elasticsearch
Operator   5.8.1          elasticsearch-operator.v5.8.0   Succeeded
non-destructive-test                     elasticsearch-operator.v5.8.1   OpenShift
Elasticsearch Operator   5.8.1          elasticsearch-operator.v5.8.0   Succeeded
openshift-apiserver-operator             elasticsearch-operator.v5.8.1   OpenShift
Elasticsearch Operator   5.8.1          elasticsearch-operator.v5.8.0   Succeeded
openshift-apiserver                      elasticsearch-operator.v5.8.1   OpenShift
Elasticsearch Operator   5.8.1          elasticsearch-operator.v5.8.0   Succeeded
...
```

10.2.5. 配置日志存储

您可以通过修改 **ClusterLogging** 自定义资源(CR)来配置日志使用的日志存储类型。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 Red Hat OpenShift Logging Operator 和一个内部日志存储，它是 LokiStack 或 Elasticsearch。
- 您已创建了 **ClusterLogging** CR。



注意

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。如果您目前使用随 Logging 5.8 发布的 OpenShift Elasticsearch Operator，它将继续使用 Logging，直到 Logging 5.8 的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

流程

1. 修改 **ClusterLogging** CR **logStore** 规格：

ClusterLogging CR 示例

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

```
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  logStore:
    type: <log_store_type> 1
    elasticsearch: 2
      nodeCount: <integer>
      resources: {}
      storage: {}
      redundancyPolicy: <redundancy_type> 3
    lokistack: 4
      name: {}
# ...
```

1. 指定日志存储类型。这可以是 **lokistack** 或 **elasticsearch**。
2. Elasticsearch 日志存储的可选配置选项。
3. 指定冗余类型。这个值可以是 **ZeroRedundancy**、**SingleRedundancy**、**MultipleRedundancy** 或 **FullRedundancy**。
4. LokiStack 的可选配置选项。

将 LokiStack 指定为日志存储的 ClusterLogging CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
    lokistack:
      name: logging-loki
# ...
```

2. 运行以下命令来应用 **ClusterLogging** CR：

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

10.3. 配置 LOKISTACK 日志存储

在日志记录文档中，*LokiStack* 指的是 Loki 和 Web 代理与 OpenShift Container Platform 身份验证集成的日志记录组合。LokiStack 的代理使用 OpenShift Container Platform 身份验证来强制实施多租户。*Loki* 将日志存储指代为单个组件或外部存储。

10.3.1. 为 cluster-admin 用户角色创建新组

重要

以 **cluster-admin** 用户身份查询多个命名空间的应用程序日志，其中集群中所有命名空间的字符总和大于 5120，会导致错误 **Parse error: input size too long (XXXX > 5120)**。为了更好地控制 LokiStack 中日志的访问，请使 **cluster-admin** 用户成为 **cluster-admin** 组的成员。如果 **cluster-admin** 组不存在，请创建它并将所需的用户添加到其中。

使用以下步骤为具有 **cluster-admin** 权限的用户创建新组。

流程

1. 输入以下命令创建新组：

```
$ oc adm groups new cluster-admin
```

2. 输入以下命令将所需的用户添加到 **cluster-admin** 组中：

```
$ oc adm groups add-users cluster-admin <username>
```

3. 输入以下命令在组中添加 **cluster-admin** 用户角色：

```
$ oc adm policy add-cluster-role-to-group cluster-admin cluster-admin
```

10.3.2. 集群重启过程中的 LokiStack 行为

在日志记录版本 5.8 及更新版本中，当 OpenShift Container Platform 集群重启时，Loki ingestion 和查询路径将继续在节点的可用 CPU 和内存资源中运行。这意味着 OpenShift Container Platform 集群更新过程中，LokiStack 没有停机。此行为通过使用 **PodDisruptionBudget** 资源来实现。Loki Operator 为 Loki 置备 **PodDisruptionBudget** 资源，它决定了每个组件必须可用的最少 pod 数量，以确保特定条件下正常操作。

其他资源

- Pod 中断预算 Kubernetes 文档

10.3.3. 配置 Loki 以容忍节点故障

在日志记录 5.8 及更新的版本中，Loki Operator 支持设置 pod 反关联性规则，以请求同一组件的 pod 调度到集群中的不同可用节点上。

关联性是 pod 的一个属性，用于控制它们希望调度到的节点。反关联性是 pod 的一个属性，用于阻止 pod 调度到某个节点上。

在 OpenShift Container Platform 中，可以借助 *pod 关联性*和 *pod 反关联性*来根据其他 pod 上的键/值标签限制 pod 有资格调度到哪些节点。

Operator 会为所有 Loki 组件设置默认、首选的 **podAntiAffinity** 规则，其中包括 **compactor**, **distributor**, **gateway**, **indexGateway**, **ingester**, **querier**, **queryFrontend**, 和 **ruler** 组件。

您可以通过在 **requiredDuringSchedulingIgnoredDuringExecution** 字段中配置所需的设置来覆盖 Loki 组件的首选 **podAntiAffinity** 设置：

ingester 组件的用户设置示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  template:
    ingester:
      podAntiAffinity:
      # ...
        requiredDuringSchedulingIgnoredDuringExecution: 1
        - labelSelector:
            matchLabels: 2
              app.kubernetes.io/component: ingester
          topologyKey: kubernetes.io/hostname
# ...
```

1 定义必要规则的小节。

2 必须匹配键-值对（标签）才能应用该规则。

其他资源

- **PodAntiAffinity** v1 core Kubernetes 文档
- 将 Pod 分配给节点 Kubernetes 文档
- 使用关联性和反关联性规则相对于其他 pod 放置 pod

10.3.4. 支持区域的数据复制

在日志记录 5.8 及更新的版本中，Loki Operator 通过 pod 拓扑分布限制支持区感知数据复制。启用这个功能可提高可靠性，并防止出现单一区域故障的日志丢失。在将部署大小配置为 **1x.extra.small**、**1x.small** 或 **1x.medium** 时，**replication.factor** 字段会自动设置为 2。

为确保正确复制，您需要至少具有与复制因子指定的可用区数量。虽然可用区可能会比复制因素更多，但区域数量较少可能会导致写入失败。每个区域应托管相等的实例数量，以实现最佳操作。

启用区复制的 LokiStack CR 示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
 name: logging-loki
 namespace: openshift-logging
spec:
 replicationFactor: 2 1
 replication:
  factor: 2 2
  zones:
  - maxSkew: 1 3
    topologyKey: topology.kubernetes.io/zone 4
```

1. 弃用的字段，输入的值会被 **replication.factor** 覆盖。
2. 当在设置时选择部署大小时，会自动设置这个值。
3. 两个拓扑域间的 pod 数量的最大差别。默认值为 1，您无法指定 0。
4. 以与节点标签对应的拓扑键的形式定义区域。

10.3.4.1. 从失败的区恢复 Loki pod

在 OpenShift Container Platform 中，当特定可用区资源无法访问时，会发生区失败。可用性区域是云提供商数据中心内的隔离区域，旨在增强冗余和容错能力。如果您的 OpenShift Container Platform 集群没有配置为处理此操作，则区失败可能会导致服务或数据丢失。

Loki pod 是 StatefulSet 的一部分，它们附带 **StorageClass** 对象置备的 PVC。每个 Loki pod 及其 PVC 驻留在同一区域中。当在集群中发生区故障时，StatefulSet 控制器会自动尝试恢复失败的区中受影响的 pod。

警告

以下流程将删除失败的区中的 PVC，以及其中包含的所有数据。为了避免完成数据丢失的 **LokiStack** CR 的 replication factor 字段，应该始终设置为大于 1 的值，以确保 Loki 复制。

先决条件

- 日志记录版本 5.8 或更高版本。
- 验证 **LokiStack** CR 是否具有大于 1 的复制因素。
- control plane 检测到区失败，故障区中的节点由云供应商集成标记。

StatefulSet 控制器会自动尝试重新调度失败的区中的 pod。因为关联的 PVC 也位于失败的区中，所以自动重新调度到不同的区无法正常工作。您必须手动删除失败的区中 PVC，以便在新区中成功重新创建有状态 Loki Pod 及其置备的 PVC。

流程

1. 运行以下命令，列出处于 **Pending** 状态的 pod：

```
oc get pods --field-selector status.phase==Pending -n openshift-logging
```

oc get pods 输出示例

```
NAME                           READY   STATUS    RESTARTS   AGE (1)
logging-loki-index-gateway-1   0/1     Pending   0          17m
logging-loki-ingester-1        0/1     Pending   0          16m
logging-loki-ruler-1           0/1     Pending   0          16m
```

1 这些 pod 处于 **Pending** 状态，因为它们对应的 PVC 位于失败的区中。

2. 运行以下命令，列出处于 **Pending** 状态的 PVC：

```
oc get pvc -o=json -n openshift-logging | jq '.items[] | select(.status.phase == "Pending") | .metadata.name' -r
```

oc get pvc 输出示例

```
storage-logging-loki-index-gateway-1
storage-logging-loki-ingester-1
wal-logging-loki-ingester-1
storage-logging-loki-ruler-1
wal-logging-loki-ruler-1
```

3. 运行以下命令，删除 pod 的 PVC：

```
oc delete pvc __<pvc_name>__  -n openshift-logging
```

4. 然后，运行以下命令来删除 pod：

```
oc delete pod __<pod_name>__  -n openshift-logging
```

成功删除这些对象后，应在可用区域中自动重新调度它们。

10.3.4.1.1. 对处于终止状态的 PVC 进行故障排除

如果 PVC 元数据终结器被设置为 **kubernetes.io/pv-protection**，PVC 可能会处于 terminating 状态。删除终结器应该允许 PVC 成功删除。

1. 运行以下命令删除每个 PVC 的终结器，然后重试删除。

```
oc patch pvc __<pvc_name>__ -p '{"metadata":{"finalizers":null}}' -n openshift-logging
```

其他资源

- 拓扑分布限制 Kubernetes 文档
- Kubernetes 存储文档.
- 使用 pod 拓扑分布限制控制 pod 放置

10.3.5. 对 Loki 日志的精细访问

在日志记录 5.8 及更高版本中，Red Hat OpenShift Logging Operator 默认不授予所有用户对日志的访问权限。作为管理员，您需要配置用户访问权限，除非 Operator 已升级并且以前的配置已就位。根据您的配置和需要，您可以使用以下内容配置对日志的精细访问：

- 集群范围内的策略
- 命名空间范围策略

- 创建自定义 admin 组

作为管理员，您需要创建适合部署的角色绑定和集群角色绑定。Red Hat OpenShift Logging Operator 提供以下集群角色：

- **cluster-logging-application-view** 授予读取应用程序日志的权限。
- **cluster-logging-infrastructure-view** 授予读取基础架构日志的权限。
- **cluster-logging-audit-view** 授予读取审计日志的权限。

如果您从以前的版本升级，则额外的集群角色 **logging-application-logs-reader** 和关联的集群角色绑定 **logging-all-authenticated-application-logs-reader** 提供向后兼容性，允许任何经过身份验证的用户在命名空间中读取访问权限。

注意

在查询应用程序日志时，具有命名空间权限的用户必须提供命名空间。

10.3.5.1. 集群范围内的访问

集群角色绑定资源引用集群角色，以及设置集群范围的权限。

ClusterRoleBinding 示例

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: logging-all-application-logs-reader
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-logging-application-view 1
subjects: 2
- kind: Group
  name: system:authenticated
  apiGroup: rbac.authorization.k8s.io
```

1. 额外的 **ClusterRole** 是 **cluster-logging-infrastructure-view** 和 **cluster-logging-audit-view**。
2. 指定此对象应用到的用户或组。

10.3.5.2. 命名空间访问

RoleBinding 资源可用于 **ClusterRole** 对象来定义用户或组可以访问日志的命名空间。

RoleBinding 示例

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: allow-read-logs
  namespace: log-test-0 1
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-logging-application-view
subjects:
- kind: User
  apiGroup: rbac.authorization.k8s.io
  name: testuser-0
```

1 指定此 **RoleBinding** 应用到的命名空间。

10.3.5.3. 自定义 admin 组访问

如果您的部署有很多需要更广泛的用户，您可以使用 **adminGroup** 字段创建自定义组。属于 **LokiStack** CR 的 **adminGroups** 字段中指定的任何组的成员的用户被视为 admin。如果还被分配了 **cluster-logging-application-view** 角色，则管理员用户有权访问所有命名空间中的所有应用程序日志。

LokiStack CR 示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  tenants:
    mode: openshift-logging 1
    openshift:
      adminGroups: 2
      - cluster-admin
      - custom-admin-group 3
```

1 自定义管理组仅在此模式中可用。

2 为此字段输入空 list **[]** 值会禁用 admin 组。

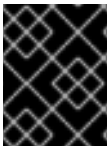
3 覆盖默认组(**system:cluster-admins**,**cluster-admin**,**dedicated-admin**)

其他资源

- 使用 RBAC 定义和应用权限

10.3.6. 使用 Loki 启用基于流的保留

使用日志记录版本 5.6 及更高版本，您可以根据日志流配置保留策略。这些规则可全局设置，每个租户或两个都设置。如果同时配置这两个，则租户规则会在全局规则之前应用。



重要

如果没有在 s3 存储桶或 LokiStack 自定义资源 (CR) 中定义保留周期，则不会修剪日志，它们会永久保留在 s3 存储桶中，这可能会填满 s3 存储。

注意

虽然日志记录版本 5.9 和更高版本支持模式 v12，但建议使用 v13。

1. 要启用基于流的保留，请创建一个 **LokiStack** CR：

AWS 的基于流的全局保留示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
   global: 1
     retention: 2
       days: 20
       streams:
       - days: 4
         priority: 1
         selector: '{kubernetes_namespace_name=~"test.+"}' 3
       - days: 1
         priority: 1
         selector: '{log_type="infrastructure"}'
  managementState: Managed
  replicationFactor: 1
  size: 1x.small
  storage:
    schemas:
    - effectiveDate: "2020-10-11"
      version: v11
    secret:
      name: logging-loki-s3
      type: aws
  storageClassName: gp3-csi
  tenants:
    mode: openshift-logging
```

1. 为所有日志流设置保留策略。**注：此字段不会影响存储在对象存储中的保留周期。**
2. 当此块被添加到 CR 时，集群中会启用保留。
3. 包含用于定义日志 stream.spec: limits 的 LogQL 查询 ：

AWS 的基于流的基于流的保留示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
```

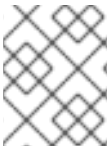
```
  global:
    retention:
      days: 20
  tenants: 1
    application:
      retention:
        days: 1
        streams:
          - days: 4
            selector: '{kubernetes_namespace_name=~"test.+"}' 2
    infrastructure:
      retention:
        days: 5
        streams:
          - days: 1
            selector: '{kubernetes_namespace_name=~"openshift-cluster.+"}'
managementState: Managed
replicationFactor: 1
size: 1x.small
storage:
  schemas:
  - effectiveDate: "2020-10-11"
    version: v11
  secret:
    name: logging-loki-s3
    type: aws
storageClassName: gp3-csi
tenants:
  mode: openshift-logging
```

1 根据租户设置保留策略。有效的租户类型是 **application**、**audit** 和 **infrastructure**。

2 包含用于定义日志流的 LogQL 查询。

2 应用 **LokiStack** CR：

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



注意

这不适用于为存储的日志管理保留。使用对象存储配置存储在支持的最大 30 天的全局保留周期。

10.3.7. Loki 速率限制错误故障排除

如果 Log Forwarder API 将超过速率限制的大量信息转发到 Loki，Loki 会生成速率限制(**429**)错误。

这些错误可能会在正常操作过程中发生。例如，当将 logging 添加到已具有某些日志的集群中时，logging 会尝试充分利用现有日志条目时可能会出现速率限制错误。在这种情况下，如果添加新日志的速度小于总速率限值，历史数据最终会被处理，并且不要求用户干预即可解决速率限制错误。

如果速率限制错误持续发生，您可以通过修改 **LokiStack** 自定义资源(CR)来解决此问题。

重要

LokiStack CR 在 Grafana 托管的 Loki 上不可用。本主题不适用于 Grafana 托管的 Loki 服务器。

Conditions

- Log Forwarder API 配置为将日志转发到 Loki。
- 您的系统向 Loki 发送大于 2 MB 的消息块。例如：

```
"values":[["1630410392689800468","{\"kind\":\"Event\",\"apiVersion\":\
.......
......
......
......
\"received_at\":\"2021-08-31T11:46:32.800278+00:00\",\"version\":\"1.7.4
1.6.0\"}},\"@timestamp\":\"2021-08-
31T11:46:32.799692+00:00\",\"viaq_index_name\":\"audit-
write\",\"viaq_msg_id\":\"MzFjYjJkZjItNjY0MC00YWU4LWIwMTEtNGNmM2E5ZmViMGU4\",\"log_type\":\"audit\"}"]]}]}
```

- 输入 **oc logs -n openshift-logging -l component=collector** 后，集群中的收集器日志会显示包含以下错误消息之一的行：

```
429 Too Many Requests Ingestion rate limit exceeded
```

Vector 错误消息示例

```
2023-08-25T16:08:49.301780Z  WARN sink{component_kind="sink"
component_id=default_loki_infra component_type=loki component_name=default_loki_infra}:
vector::sinks::util::retries: Retrying after error. error=Server responded with an error: 429 Too
Many Requests internal_log_rate_limit=true
```

Fluentd 错误消息示例

```
2023-08-30 14:52:15 +0000 [warn]: [default_loki_infra] failed to flush the buffer. retry_times=2
next_retry_time=2023-08-30 14:52:19 +0000
chunk="604251225bf5378ed1567231a1c03b8b"
error_class=Fluent::Plugin::LokiOutput::LogPostError error="429 Too Many Requests
Ingestion rate limit exceeded for user infrastructure (limit: 4194304 bytes/sec) while
attempting to ingest '4082' lines totaling '7820025' bytes, reduce log volume or contact your
Loki administrator to see if the limit can be increased\n"
```

在接收结束时也会看到这个错误。例如，在 LokiStack ingester pod 中：

Loki ingester 错误消息示例

```
level=warn ts=2023-08-30T14:57:34.155592243Z caller=grpc_logging.go:43
duration=1.434942ms method=/logproto.Pusher/Push err="rpc error: code = Code(429) desc
= entry with timestamp 2023-08-30 14:57:32.012778399 +0000 UTC ignored, reason: 'Per
stream rate limit exceeded (limit: 3MB/sec) while attempting to ingest for stream
```

流程

- 更新 **LokiStack** CR 中的 **ingestionBurstSize** 和 **ingestionRate** 字段：

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      ingestion:
        ingestionBurstSize: 16 (1)
        ingestionRate: 8 (2)
# ...
```

(1) **ingestionBurstSize** 字段定义每个经销商副本的最大本地速率限制示例大小（以 MB 为单位）。这个值是一个硬限制。将此值设置为至少在单个推送请求中预期的最大日志大小。不允许大于 **ingestionBurstSize** 值的单个请求。

(2) **ingestionRate** 字段是每秒最大最大样本量的软限制（以 MB 为单位）。如果日志速率超过限制，则会出现速率限制错误，但收集器会重试发送日志。只要总平均值低于限制，系统就会在没有用户干预的情况下解决错误。

10.3.8. 配置 Loki 以容许 memberlist 创建失败

在 OpenShift 集群中，管理员通常使用非专用 IP 网络范围。因此，Loki memberlist 配置会失败，因为默认情况下，它只使用私有 IP 网络。

作为管理员，您可以为 memberlist 配置选择 pod 网络。您可以修改 LokiStack CR，以使用 **hashRing** spec 中的 **podIP**。要配置 LokiStack CR，请使用以下命令：

```
$ oc patch LokiStack logging-loki -n openshift-logging  --type=merge -p '{"spec": {"hashRing":
{"memberlist":{"instanceAddrType":"podIP","type": "memberlist"}}}}'
```

LokiStack 示例，使其包含 podIP

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  hashRing:
    type: memberlist
    memberlist:
      instanceAddrType: podIP
# ...
```

10.3.9. 其它资源

- Loki 组件文档
- Loki Query Language (LogQL)文档
- Grafana 仪表板文档
- Loki Object Storage 文档
- Loki Operator **IngestionLimitSpec** 文档
- Loki Storage Schema 文档

10.4. 配置 ELASTICSEARCH 日志存储

您可以使用 Elasticsearch 6 来存储和组织日志数据。

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之间复制分片，从完整复制到不复制
- 外部访问 Elasticsearch 数据

10.4.1. 配置日志存储

您可以通过修改 **ClusterLogging** 自定义资源(CR)来配置日志使用的日志存储类型。

先决条件

- 有管理员权限。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 Red Hat OpenShift Logging Operator 和一个内部日志存储，它是 LokiStack 或 Elasticsearch。
- 您已创建了 **ClusterLogging** CR。

注意

Logging 5.9 发行版本不包含 OpenShift Elasticsearch Operator 的更新版本。如果您目前使用随 Logging 5.8 发布的 OpenShift Elasticsearch Operator，它将继续使用 Logging，直到 Logging 5.8 的 EOL 为止。您可以使用 Loki Operator 作为 OpenShift Elasticsearch Operator 的替代方案来管理默认日志存储。如需有关日志记录生命周期日期的更多信息，请参阅平台 Agnostic Operator。

流程

1. 修改 **ClusterLogging** CR **logStore** 规格：

   ClusterLogging CR 示例

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

```
metadata:
# ...
spec:
# ...
  logStore:
    type: <log_store_type> 1
    elasticsearch: 2
      nodeCount: <integer>
      resources: {}
      storage: {}
      redundancyPolicy: <redundancy_type> 3
    lokistack: 4
      name: {}
# ...
```

1. 指定日志存储类型。这可以是 **lokistack** 或 **elasticsearch**。
2. Elasticsearch 日志存储的可选配置选项。
3. 指定冗余类型。这个值可以是 **ZeroRedundancy**、**SingleRedundancy**、**MultipleRedundancy** 或 **FullRedundancy**。
4. LokiStack 的可选配置选项。

将 LokiStack 指定为日志存储的 ClusterLogging CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
    lokistack:
      name: logging-loki
# ...
```

2. 运行以下命令来应用 **ClusterLogging** CR：

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

10.4.2. 将审计日志转发到日志存储

在日志记录部署中，容器和基础架构日志默认转发到 **ClusterLogging** 自定义资源(CR)中定义的内部日志存储。

默认情况下，审计日志不会转发到内部日志存储，因为这不提供安全存储。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。

如果此默认配置满足您的需要，则不需要配置一个 **ClusterLogForwarder** CR。如果存在 **ClusterLogForwarder** CR，日志不会转发到内部日志存储，除非定义了包含 **default** 输出的管道。

流程

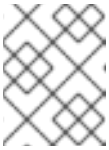
使用 Log Forward API 将审计日志转发到内部 Elasticsearch 实例：

1. 创建或编辑定义 **ClusterLogForwarder** CR 对象的 YAML 文件：

   - 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

- 关于日志收集和转发

10.4.3. 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 Red Hat OpenShift Logging Operator 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: (1)
      - name: infra-policy
        phases:
          delete:
            minAge: 7d (2)
          hot:
            actions:
              rollover:
                maxAge: 8h (3)
        pollInterval: 15m (4)
...
```

(1) 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。

(2) 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**。

(3) 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。

4 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
```

10.4.4. 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 Red Hat OpenShift Logging 和 Elasticsearch Operator。

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 所需的最小资源。

(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**。

在调整 Elasticsearch 内存量时，相同的值应该用于**请求**和**限值**。

例如：

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

(1) 资源的最大数量。

(2) 最低要求。

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。

10.4.5. 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 Red Hat OpenShift Logging 和 Elasticsearch Operator。

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。

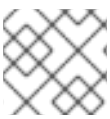
注意

索引模板的主分片数量等于 Elasticsearch 数据节点的数目。

10.4.6. 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。

10.4.7. 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

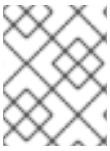
- 必须安装 Red Hat OpenShift Logging 和 Elasticsearch Operator。

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。

10.4.8. 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 Red Hat OpenShift Logging 和 Elasticsearch Operator。

流程

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

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

10.4.9. 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 Red Hat OpenShift Logging 和 Elasticsearch Operator。

流程

执行集群滚动重启：

1. 进入 **openshift-logging** 项目：

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

2. 获取 Elasticsearch Pod 的名称：

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

3. 缩减收集器 Pod，以便它们停止向 Elasticsearch 发送新日志：

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

4. 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}}
```

5. 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":
```

6. 完成后，会在每个部署中都有一个 ES 集群：

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

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

   例如：

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

   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 -l component=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. 扩展收集器 Pod，以便它们会将新日志发送到 Elasticsearch。

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

10.4.10. 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日志记录的日志存储。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 Red Hat OpenShift Logging 和 Elasticsearch Operator。
- 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-tJzcR3KOdpgeMJo-MQ",
  "version" : {
  "number" : "6.8.1",
  "build_flavor" : "oss",
  "build_type" : "zip",
  "build_hash" : "Unknown",
  "build_date" : "Unknown",
  "build_snapshot" : true,
  "lucene_version" : "7.7.0",
  "minimum_wire_compatibility_version" : "5.6.0",
  "minimum_index_compatibility_version" : "5.0.0"
},
  "<tagline>" : "<for search>"
}
```

10.4.11. 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

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:
    type: "fluentd"
    fluentd: {}
```

4. 验证收集器 Pod 是否已重新部署：

```
$ oc get pods -l component=collector -n openshift-logging
```

第 11 章 日志记录警报

11.1. 默认日志记录警报

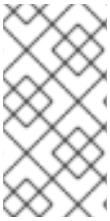
日志记录警报作为 Red Hat OpenShift Logging Operator 安装的一部分安装。警报取决于日志收集和日志存储后端导出的指标。如果在安装 Red Hat OpenShift Logging Operator 时选择 **Enable operator recommended cluster monitoring** 选项来启用这些指标。有关安装日志记录 Operator 的更多信息，请参阅使用 Web 控制台安装日志记录。

默认日志记录警报发送到 **openshift-monitoring** 命名空间中的 OpenShift Container Platform 监控堆栈 Alertmanager，除非您禁用了本地 Alertmanager 实例。

11.1.1. 在 Administrator 和 Developer 视角中访问 Alerting UI

Alerting UI 可通过 OpenShift Container Platform Web 控制台的 **Administrator** 视角和 **Developer** 视角访问。

- 在 **Administrator** 视角中，进入 **Observe → Alerting**。此视角中的 Alerting UI 中的三个主要页面是 **Alerts**、**Silences** 和 **Alerting 规则** 页面。
- 在 **Developer** 视角中，进入 **Observe → <project_name> → Alerts**。在这个视角中，警报、静默和警报规则都通过 **Alerts** 页面管理。**Alerts** 页面中显示的结果特定于所选项目。



注意

在 **Developer** 视角中，您可以从可在 **Project: <project_name>** 列表中访问的核心 OpenShift Container Platform 和用户定义的项目中选择。但是，如果您没有以集群管理员身份登录，则不会显示与 OpenShift Container Platform 核心项目相关的警报、静默和警报规则。

11.1.2. 日志记录收集器警报

在日志记录 5.8 及更新的版本中，Red Hat OpenShift Logging Operator 生成以下警报。您可以在 OpenShift Container Platform Web 控制台中查看这些警报。

警报名称	消息	描述	重要性
CollectorNodeDown	Prometheus 无法刮除（scrape）**namespace/pod** 收集器组件超过 10m。	无法刮除收集器。	Critical
CollectorHighErrorRate	**value**% 的记录会导致 **namespace/pod** 错误。	**namespace/pod** 收集器组件错误高。	Critical
CollectorVeryHighErrorRate	**value**% 的记录会导致 **namespace/pod** 错误。	**namespace/pod** 收集器错误非常高。	Critical

11.1.3. Vector 收集器警报

在日志记录 5.7 及更新的版本中，向量收集器生成以下警报。您可以在 OpenShift Container Platform Web 控制台中查看这些警报。

表 11.1. Vector 收集器警报

警报	消息	描述	重要性
CollectorHighErrorRate	**<value> of records have resulted in an error by vector <instance>.**	在前 15 分钟内，向量输出错误的数量很高，默认为 10。	Warning
CollectorNodeDown	**Prometheus could not scrape vector <instance> for more than 10m.**	向量报告 Prometheus 无法提取特定的 Vector 实例。	Critical
CollectorVeryHighErrorRate	**<value> of records have resulted in an error by vector <instance>.**	向量组件错误的数量很高，默认为在前 15 分钟内有 25 个。	Critical
FluentdQueueLengthIncreasing	**In the last 1h, fluentd <instance> buffer queue length constantly increased more than 1.Current value is <value>.**	Fluentd 报告队列大小正在增加。	Warning

11.1.4. Fluentd 收集器警报

以下警报由旧的 Fluentd 日志收集器生成。您可以在 OpenShift Container Platform Web 控制台中查看这些警报。

表 11.2. Fluentd 收集器警报

警报	消息	描述	重要性
FluentDHighErrorRate	**<value> of records have resulted in an error by fluentd <instance>.**	FluentD 输出错误数量很高，在前 15 分钟中默认超过 10。	Warning
FluentdNodeDown	**Prometheus could not scrape fluentd <instance> for more than 10m.**	Fluentd 报告 Prometheus 可能无法抓取特定的 Fluentd 实例。	Critical
FluentdQueueLengthIncreasing	**In the last 1h, fluentd <instance> buffer queue length constantly increased more than 1.Current value is <value>.**	Fluentd 报告队列大小正在增加。	Warning

警报	消息	描述	重要性
FluentDVeryHighErrorRate	**<value> of records have resulted in an error by fluentd <instance>.**	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

11.1.5. Elasticsearch 警报规则

您可以在 OpenShift Container Platform Web 控制台中查看这些警报规则。

表 11.3. 警报规则

警报	描述	重要性
ElasticsearchClusterNotHealthy	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
ElasticsearchClusterNotHealthy	集群健康状态为 YELLOW 至少 20 分钟。某些分片副本尚未分配。	Warning
ElasticsearchDiskSpaceRunningLow	集群预期在以后的 6 小时内处于磁盘空间之外。	Critical
ElasticsearchHighFileDescriptorUsage	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。	Warning
ElasticsearchJVMHeapUseHigh	指定节点上的 JVM 堆使用率很高。	警报
ElasticsearchNodeDiskWatermarkReached	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
ElasticsearchNodeDiskWatermarkReached	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。	Warning
ElasticsearchNodeDiskWatermarkReached	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
ElasticsearchJVMHeapUseHigh	指定节点上的 JVM 堆使用率太高。	警报
ElasticsearchWriteRequestsRejectionJumps	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。	Warning
AggregatedLoggingSystemCPUHigh	该系统在指定节点上使用的 CPU 太高。	警报

警报	描述	重要性
ElasticsearchProcessCPUHigh	Elasticsearch 在指定节点上使用的 CPU 太高。	警报

11.1.6. 其他资源

- 修改核心平台警报规则

11.2. 自定义日志记录警报

在日志记录 5.7 及更新的版本中，用户可以配置 LokiStack 部署来生成自定义警报和记录的指标。如果要使用自定义 警报和记录规则，您必须启用 LokiStack 规则器组件。

LokiStack 基于日志的警报和记录的指标通过将 LogQL 表达式提供给 ruler 组件来触发。Loki Operator 管理了一个针对所选 LokiStack 大小优化的标尺，可以是 **1x.extra-small**、**1x.small** 或 **1x.medium**。

要提供这些表达式，您必须创建一个 **AlertingRule** 自定义资源 (CR)，其中包含与 Prometheus 兼容的 警报规则，或包含 Prometheus 兼容的 记录规则 的 **RecordingRule** CR。

管理员可以为 **application**, **audit**, 或 **infrastructure** 租户配置基于日志的警报或记录指标数据。没有管理员权限的用户可为他们有权访问的**应用程序**租户配置基于日志的警报或记录指标。

应用程序、审计和基础架构警报默认发送到 **openshift-monitoring** 命名空间中的 OpenShift Container Platform 监控堆栈 Alertmanager，除非您禁用了本地 Alertmanager 实例。如果启用了用于监控 **openshift-user-workload-monitoring** 命名空间中的用户定义的项目的 Alertmanager，应用程序警报默认发送到此命名空间中的 Alertmanager。

11.2.1. 配置规则器

启用 LokiStack 规则器组件后，用户可以定义一组 LogQL 表达式，用于触发日志记录警报或记录指标。

管理员可以通过修改 **LokiStack** 自定义资源(CR) 来启用规则器。

先决条件

- 已安装 Red Hat OpenShift Logging Operator 和 Loki Operator。
- 您已创建了 **LokiStack** CR。
- 有管理员权限。

流程

- 通过确保 **LokiStack** CR 包含以下 spec 配置来启用规则器：

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: <name>
  namespace: <namespace>
spec:
```

```
# ...
  rules:
    enabled: true 1
    selector:
      matchLabels:
        openshift.io/<label_name>: "true" 2
    namespaceSelector:
      matchLabels:
        openshift.io/<label_name>: "true" 3
```

1. 在集群中启用 Loki 警报和记录规则。
2. 添加可添加到要启用日志记录警报和指标的命名空间的自定义标签。
3. 添加可添加到要启用日志记录警报和指标的命名空间的自定义标签。

11.2.2. 授权 LokiStack 规则 RBAC 权限

管理员可以允许用户通过将集群角色绑定到 username 来创建和管理自己的警报和记录规则。集群角色定义为 **ClusterRole** 对象，其中包含用户所需的基于角色的访问控制(RBAC)权限。

在日志记录 5.8 及更高版本中，为 LokiStack 提供了以下用于警报和记录规则的集群角色：

运行名称	描述
alertingrules.loki.grafana.com-v1-admin	具有此角色的用户具有管理级别访问权限来管理警报规则。此集群角色授予在 **loki.grafana.com/v1** API 组中创建、读取、更新、删除、列出和监视 **AlertingRule** 资源的权限。
alertingrules.loki.grafana.com-v1-crdview	具有此角色的用户可以查看与 **loki.grafana.com/v1** API 组中的 **AlertingRule** 资源相关的自定义资源定义(CRD)的定义，但没有修改或管理这些资源的权限。
alertingrules.loki.grafana.com-v1-edit	具有此角色的用户有权创建、更新和删除 **AlertingRule** 资源。
alertingrules.loki.grafana.com-v1-view	具有此角色的用户可以读取 **loki.grafana.com/v1** API 组中的 **AlertingRule** 资源。它们可以检查现有警报规则的配置、标签和注解，但不能对它们进行任何修改。
recordingrules.loki.grafana.com-v1-admin	具有此角色的用户具有管理记录规则的管理级别访问权限。此集群角色授予在 **loki.grafana.com/v1** API 组中创建、读取、更新、删除、列出和监视 **RecordingRule** 资源的权限。
recordingrules.loki.grafana.com-v1-crdview	具有此角色的用户可以查看与 **loki.grafana.com/v1** API 组中的 **RecordingRule** 资源相关的自定义资源定义(CRD)的定义，但没有修改或管理这些资源的权限。

运行名称	描述
recordingrules.loki.grafana.com-v1-edit	具有此角色的用户有权创建、更新和删除 **RecordingRule** 资源。
recordingrules.loki.grafana.com-v1-view	具有此角色的用户可以读取 **loki.grafana.com/v1** API 组中的 **RecordingRule** 资源。它们可以检查现有警报规则的配置、标签和注解，但不能对它们进行任何修改。

11.2.2.1. 例子

要为用户应用集群角色，您必须将现有集群角色绑定到特定用户名。

集群角色可以是集群或命名空间范围，具体取决于您使用的角色绑定。使用 **RoleBinding** 对象时，如使用 **oc adm policy add-role-to-user** 命令时，集群角色仅适用于指定的命名空间。当使用 **ClusterRoleBinding** 对象时，如使用 **oc adm policy add-cluster-role-to-user** 命令时，集群角色会应用到集群中的所有命名空间。

以下示例命令为指定用户在集群中的特定命名空间中创建、读取、更新和删除(CRUD)权限：

特定命名空间中警报规则 CRUD 权限的集群角色绑定命令示例

```
$ oc adm policy add-role-to-user alertingrules.loki.grafana.com-v1-admin -n <namespace>
<username>
```

以下命令为所有命名空间中的警报规则授予指定用户管理员权限：

管理员权限的集群角色绑定命令示例

```
$ oc adm policy add-cluster-role-to-user alertingrules.loki.grafana.com-v1-admin <username>
```

其他资源

- 使用 RBAC 定义和应用权限

11.2.3. 使用 Loki 创建基于日志的警报规则

AlertingRule CR 包含一组规格和 webhook 验证定义，用于声明单个 **LokiStack** 实例的警报规则组。另外，webhook 验证定义支持规则验证条件：

- 如果 **AlertingRule** CR 包含无效的 **interval** 周期，则它是一个无效的警报规则
- 如果 **AlertingRule** CR 包含无效的 **for** 周期，则它是一个无效的警报规则
- 如果 **AlertingRule** CR 包含无效的 LogQL **expr**，则它是一个无效的警报规则。
- 如果 **AlertingRule** CR 包含两个同名的组，则它是一个无效的警报规则。
- 如果以上都不适用，则警报规则被视为有效。

租户类型	AlertingRule CR 的有效命名空间
application	
audit	**openshift-logging**
infrastructure	**openshift-/***, **kube-/***, **default**

先决条件

- Red Hat OpenShift Logging Operator 5.7 及更新的版本
- OpenShift Container Platform 4.13 及更新的版本

流程

1. 创建 **AlertingRule** 自定义资源 (CR)：

   基础架构 AlertingRule CR 示例

```
apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
  name: loki-operator-alerts
  namespace: openshift-operators-redhat (1)
  labels: (2)
    openshift.io/<label_name>: "true"
spec:
  tenantID: "infrastructure" (3)
  groups:
    - name: LokiOperatorHighReconciliationError
      rules:
        - alert: HighPercentageError
          expr: | (4)
            sum(rate({kubernetes_namespace_name="openshift-operators-redhat",
kubernetes_pod_name=~"loki-operator-controller-manager.*"} |= "error" [1m])) by (job)
              /
            sum(rate({kubernetes_namespace_name="openshift-operators-redhat",
kubernetes_pod_name=~"loki-operator-controller-manager.*"}[1m])) by (job)
              > 0.01
          for: 10s
          labels:
            severity: critical (5)
          annotations:
            summary: High Loki Operator Reconciliation Errors (6)
            description: High Loki Operator Reconciliation Errors (7)
```

(1) 创建此 **AlertingRule** CR 的命名空间必须具有与 LokiStack **spec.rules.namespaceSelector** 定义匹配的标签。

(2) **labels** 块必须与 LokiStack **spec.rules.selector** 定义匹配。

3. **infrastructure** 租户的 **AlertingRule** CR 只在 **openshift-***, **kube-***, 或 **default** 命名空间中被支持。

4. **kubernetes_namespace_name:** 的值必须与 **metadata.namespace** 的值匹配。

5. 此必需字段的值必须是 **critical**、**warning** 或 **info**。

6. 这个字段是必须的。

7. 这个字段是必须的。

应用程序 AlertingRule CR 示例

```
apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
  name: app-user-workload
  namespace: app-ns 1
  labels: 2
    openshift.io/<label_name>: "true"
spec:
  tenantID: "application"
  groups:
    - name: AppUserWorkloadHighError
      rules:
        - alert:
          expr: | 3
          sum(rate({kubernetes_namespace_name="app-ns",
kubernetes_pod_name=~"podName.*"} |= "error" [1m])) by (job)
          for: 10s
          labels:
            severity: critical 4
          annotations:
            summary: 5
            description: 6
```

1. 创建此 **AlertingRule** CR 的命名空间必须具有与 LokiStack **spec.rules.namespaceSelector** 定义匹配的标签。

2. **labels** 块必须与 LokiStack **spec.rules.selector** 定义匹配。

3. **kubernetes_namespace_name:** 的值必须与 **metadata.namespace** 的值匹配。

4. 此必需字段的值必须是 **critical**、**warning** 或 **info**。

5. 此必需字段的值是规则的摘要。

6. 此必填字段的值是规则的详细描述。

2. 应用 **AlertingRule** CR：

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

11.2.4. 其他资源

- 关于 OpenShift Container Platform 监控
- 配置警报通知

第 12 章 性能和可靠性调整

12.1. 流控制机制

如果生成日志比可以收集的速度快，很难预测或控制发送到输出的日志卷。无法预测或控制发送到输出的日志卷可能会导致日志丢失。如果有系统中断，且日志缓冲区在没有用户控制的情况下被累计，则在连接被恢复时可能会造成长时间恢复时间和高延迟。

作为管理员，您可以通过为日志记录配置流控制机制来限制日志记录率。

12.1.1. 流控制机制的好处

- 提前可以更准确地预测日志的成本和卷。
- 无容器无法生成分离其他容器的未绑定日志流量。
- 忽略低值日志可减少日志记录基础架构的负载。
- 通过分配更高的速率限制，可以首选使用高值日志。

12.1.2. 配置速率限制

每个收集器配置速率限制，这意味着日志收集器的最大速率是收集器实例的数量乘以速率限制。

因为从每个节点的文件系统收集日志，所以会在每个集群节点上部署收集器。例如，在 3 节点集群中，每个收集器的最大速率限制为每秒 10 个记录，日志集合的最大速率为每秒 30 个记录。

因为写入输出的记录的确切字节大小可能会因转换、不同的编码或其他因素而不同，因此速率限制以记录数而不是字节设置。

您可以通过两种方式在 **ClusterLogForwarder** 自定义资源 (CR) 中配置速率限制：

输出速率限制

将出站日志的速度限制为所选输出，例如，与输出的网络或存储容量匹配。输出速率限制控制每个输出率的聚合。

输入速率限制

限制所选容器的日志收集的每容器速率。

12.1.3. 配置日志转发器输出速率限制

您可以通过配置 **ClusterLogForwarder** 自定义资源 (CR) 来限制出站日志到指定输出的速度。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 有管理员权限。

流程

1. 在指定输出的 **ClusterLogForwarder** CR 中添加 **maxRecordsPerSecond** limit 值。
   以下示例演示了如何为名为 **kafka-example** 的 Kafka 代理输出配置每个收集器输出速率限制：

ClusterLogForwarder CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
# ...
  outputs:
    - name: kafka-example 1
      type: kafka 2
      limit:
        maxRecordsPerSecond: 1000000 3
# ...
```

1. 输出名称。
2. 输出的类型。
3. 日志输出速率限制。这个值设定每秒发送到 Kafka 代理的日志的最大数量。默认情况下不设置这个值。默认行为是最佳工作，如果日志转发器无法保持启动，则会丢弃记录。如果这个值为 **0**，则不会转发任何日志。

2. 应用 **ClusterLogForwarder** CR：

示例命令

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

其他资源

- 日志输出类型

12.1.4. 配置日志转发器输入速率限制

您可以通过配置 **ClusterLogForwarder** 自定义资源(CR)来限制收集的传入日志率。您可以根据每个容器或每个命名空间设置输入限制。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 有管理员权限。

流程

1. 在一个指定输出的 **ClusterLogForwarder** CR 中添加 **maxRecordsPerSecond** limit 值。以下示例演示了如何为不同的场景配置输入速率限制：

ClusterLogForwarder CR 示例，为具有特定标签的容器设置每个容器限制

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

```
kind: ClusterLogForwarder
metadata:
# ...
spec:
# ...
  inputs:
    - name: <input_name> 1
      application:
        selector:
          matchLabels: { example: label } 2
        containerLimit:
          maxRecordsPerSecond: 0 3
# ...
```

1 输入名称。

2 标签列表。如果这些标签与应用到 pod 的标签匹配，则 **maxRecordsPerSecond** 字段中指定的每个容器限制将应用到这些容器。

3 配置速率限制。将 **maxRecordsPerSecond** 字段设置为 **0** 表示没有为容器收集任何日志。将 **maxRecordsPerSecond** 字段设置为其他值意味着为容器收集每秒记录数上限。

ClusterLogForwarder CR 示例，为所选命名空间中的容器设置每个容器限制

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
# ...
  inputs:
    - name: <input_name> 1
      application:
        namespaces: [ example-ns-1, example-ns-2 ] 2
        containerLimit:
          maxRecordsPerSecond: 10 3
    - name: <input_name>
      application:
        namespaces: [ test ]
        containerLimit:
          maxRecordsPerSecond: 1000
# ...
```

1 输入名称。

2 命名空间列表。**maxRecordsPerSecond** 字段中指定的每个容器限制应用到列出的命名空间中的所有容器。

3 配置速率限制。将 **maxRecordsPerSecond** 字段设置为 **10** 表示为列出的命名空间中的每个容器收集最多 10 个记录。

2. 应用 **ClusterLogForwarder** CR：

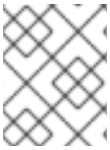
示例命令

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

12.2. 按内容过滤日志

从集群中收集所有日志可能会产生大量数据，传输和存储这些数据可能比较昂贵。

您可以通过过滤不需要存储的低优先级数据来减少日志数据的卷。日志记录提供内容过滤器，可用于减少日志数据的卷。



注意

内容过滤器与 **input** 选择器不同。**input** 选择器选择或忽略基于源元数据的整个日志流。内容过滤器编辑日志流，以根据记录内容删除和修改记录。

您可以使用以下方法之一减少日志数据卷：

- 配置内容过滤器以丢弃不需要的日志记录
- 配置内容过滤器以修剪日志记录

12.2.1. 配置内容过滤器以丢弃不需要的日志记录

配置 **drop** 过滤器后，日志收集器会根据过滤器在转发前评估日志流。收集器丢弃与指定配置匹配的不需要的日志记录。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 有管理员权限。
- 您已创建了 **ClusterLogForwarder** 自定义资源 (CR)。

流程

1. 将过滤器的配置添加到 **ClusterLogForwarder** CR 中的 **filters** spec 中。
   以下示例演示了如何配置 **ClusterLogForwarder** CR，以根据正则表达式丢弃日志记录：

   ClusterLogForwarder CR 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  filters:
  - name: <filter_name>
    type: drop 1
    drop: 2
      test: 3
      - field: .kubernetes.labels."foo-bar/baz" 4
```

```
      matches: .+ 5
    - field: .kubernetes.pod_name
      notMatches: "my-pod" 6
  pipelines:
  - name: <pipeline_name> 7
    filterRefs: ["<filter_name>"]
# ...
```

1 指定过滤器的类型。**drop** 过滤器丢弃与过滤器配置匹配的日志记录。

2 指定应用 **drop** 过滤器的配置选项。

3 指定用于评估是否丢弃日志记录的测试配置。

- 如果为测试指定的所有条件都为 true，则测试会通过，记录将被丢弃。
- 当为 **drop** 过滤器配置指定多个测试时，如果有任何测试通过，则会丢弃记录。
- 如果评估条件时出错，例如，被评估的日志记录中缺少该字段，则条件评估为 false。

4 指定点分隔的字段路径，它是日志记录中字段的路径。该路径可以包含字母数字字符和下划线 (**a-zA-Z0-9_**)，例如 **.kubernetes.namespace_name**。如果网段包含此范围之外的字符，段必须放在引号内，例如，**.kubernetes.labels."foo.bar-bar/baz"**。您可以在单个 **test** 配置中包含多个字段路径，但它们都必须评估为 true 才能通过测试以及要应用的 **drop** 过滤器。

5 指定正则表达式。如果日志记录与此正则表达式匹配，它们将被丢弃。您可以为单个 **field** 路径设置 **matches** 或 **notMatches** 条件，但不能同时设置这两个条件。

6 指定正则表达式。如果日志记录与此正则表达式不匹配，它们将被丢弃。您可以为单个 **field** 路径设置 **matches** 或 **notMatches** 条件，但不能同时设置这两个条件。

7 指定 **drop** 过滤器应用到的管道。

2. 运行以下命令来应用 **ClusterLogForwarder** CR：

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

其他示例

下面的额外示例演示了如何将 **drop** 过滤器配置为仅保留更高优先级的日志记录：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  filters:
  - name: important
    type: drop
    drop:
      test:
      - field: .message
        notMatches: "(?i)critical|error"
```

```
      - field: .level
        matches: "info|warning"
# ...
```

除了在单一 **test** 配置中包含多个字段路径外，您还可以包含被视为 *OR* 检查的额外测试。在以下示例中，如果 **test** 配置评估为 true，则记录将被丢弃。但是，对于第二个 **test** 配置，两个字段 specs 都必须是 true，才能将其评估为 true ：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
# ...
spec:
  filters:
  - name: important
    type: drop
    drop:
      test:
      - field: .kubernetes.namespace_name
        matches: "^open"
      test:
      - field: .log_type
        matches: "application"
      - field: .kubernetes.pod_name
        notMatches: "my-pod"
# ...
```

12.2.2. 配置内容过滤器以修剪日志记录

配置 **prune** 过滤器时，日志收集器会根据过滤器在转发前评估日志流。收集器通过删除 pod 注解等低值字段来修剪日志记录。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 有管理员权限。
- 您已创建了 **ClusterLogForwarder** 自定义资源 (CR)。

流程

1. 将过滤器的配置添加到 **ClusterLogForwarder** CR 中的 **prune** spec 中。
   以下示例演示了如何配置 **ClusterLogForwarder** CR，以根据字段路径修剪日志记录：

重要

如果指定了这两个信息，则首先根据 **notIn** 数组修剪记录，这优先于 **in** 数组。在使用 **notIn** 数组修剪记录后，会使用 **in** 数组来修剪这些记录。

ClusterLogForwarder CR 示例

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

```
kind: ClusterLogForwarder
metadata:
# ...
spec:
  filters:
  - name: <filter_name>
    type: prune (1)
    prune: (2)
      in: [.kubernetes.annotations, .kubernetes.namespace_id] (3)
      notIn: [.kubernetes,.log_type,.message,."@timestamp"] (4)
  pipelines:
  - name: <pipeline_name> (5)
    filterRefs: ["<filter_name>"]
# ...
```

1. 指定过滤器的类型。**prune** 过滤器根据配置的字段修剪日志记录。
2. 指定应用 **prune** 过滤器的配置选项。**in** 和 **notIn** 字段被指定为点分隔字段路径的数组，它们是日志记录中字段的路径。这些路径可以包含字母数字字符和下划线 (**a-zA-Z0-9_**)，例如 **.kubernetes.namespace_name**。如果网段包含此范围之外的字符，段必须放在引号内，例如，**.kubernetes.labels."foo.bar-bar/baz"**。
3. 可选：此阵列中指定的任何字段都会从日志记录中删除。
4. 可选：没有在此阵列中指定的任何字段都会从日志记录中删除。
5. 指定 **prune** 过滤器应用到的管道。

2. 运行以下命令来应用 **ClusterLogForwarder** CR：

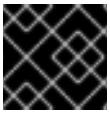
```
$ oc apply -f <filename>.yaml
```

12.2.3. 其他资源

- 关于将日志转发到第三方系统

12.3. 按元数据过滤日志

您可以使用 **input** 选择器过滤 **ClusterLogForwarder** CR 中的日志，以根据元数据选择或忽略整个日志流。作为管理员或开发人员，您可以包含或排除日志记录，以减少收集器上的内存和 CPU 负载。



重要

只有在日志记录部署中设置了 Vector 收集器时，才能使用此功能。

注意

input spec 过滤与内容过滤不同。**input** 选择器选择或忽略基于源元数据的整个日志流。内容过滤器编辑日志流，以根据记录内容删除和修改记录。

12.3.1. 通过包含或排除命名空间或容器名称，在输入中过滤应用程序日志

您可以使用 **input** 选择器根据命名空间和容器名称包含或排除应用程序日志。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 有管理员权限。
- 您已创建了 **ClusterLogForwarder** 自定义资源 (CR)。

流程

1. 添加配置，以在 **ClusterLogForwarder** CR 中包含或排除命名空间和容器名称。
   以下示例演示了如何配置 **ClusterLogForwarder** CR 以包含或排除命名空间和容器名称：

   ClusterLogForwarder CR 示例

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
# ...
spec:
  inputs:
    - name: mylogs
      application:
        includes:
          - namespace: "my-project" 1
            container: "my-container" 2
        excludes:
          - container: "other-container*" 3
            namespace: "other-namespace" 4
# ...
```

   1 指定日志只从这些命名空间中收集数据。

   2 指定日志只从这些容器中收集数据。

   3 指定收集日志时要忽略的命名空间模式。

   4 指定收集日志时要忽略的容器集合。

2. 运行以下命令来应用 **ClusterLogForwarder** CR：

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

excludes 选项优先于 **includes**。

12.3.2. 在输入 ny 过滤应用程序日志，包括标签表达式或匹配的标签键和值

您可以使用 **input** 选择器，根据标签表达式或匹配的标签键及其值包含应用程序日志。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。

- 有管理员权限。
- 您已创建了 **ClusterLogForwarder** 自定义资源 (CR)。

流程

1. 将过滤器的配置添加到 **ClusterLogForwarder** CR 中的 **input** spec 中。
以下示例演示了如何配置 **ClusterLogForwarder** CR，使其包含基于标签表达式或匹配的标签键/值的日志：

   ClusterLogForwarder CR 示例

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
# ...
spec:
  inputs:
    - name: mylogs
      application:
        selector:
          matchExpressions:
          - key: env 1
            operator: In 2
            values: [“prod”, “qa”] 3
          - key: zone
            operator: NotIn
            values: [“east”, “west”]
          matchLabels: 4
            app: one
            name: app1
# ...
```

1. 指定要匹配的标签键。
2. 指定 operator。有效值包括： **In**,**NotIn**,**Exists**, 和 **DoesNotExist**。
3. 指定字符串值的数组。如果 **operator** 值为 **Exists** 或 **DoesNotExist**，则值数组必须为空。
4. 指定准确的键或值映射。

2. 运行以下命令来应用 **ClusterLogForwarder** CR：

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

12.3.3. 根据源过滤审计和基础架构日志输入

您可以使用 **input** 选择器定义 **audit** 和 **infrastructure** 源列表，以收集日志。

先决条件

- 已安装 Red Hat OpenShift Logging Operator。
- 有管理员权限。

- 您已创建了 **ClusterLogForwarder** 自定义资源 (CR)。

流程

1. 添加配置，以在 **ClusterLogForwarder** CR 中定义 **audit** 和 **infrastructure** 源。
以下示例演示了如何配置 **ClusterLogForwarder** CR 以定义 **aduit** 和 **infrastructure** 源：

ClusterLogForwarder CR 示例

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
# ...
spec:
  inputs:
    - name: mylogs1
      infrastructure:
        sources: 1
          - node
    - name: mylogs2
      audit:
        sources: 2
          - kubeAPI
          - openshiftAPI
          - ovn
# ...
```

1 指定要收集的基础架构源列表。有效的源包括：

- **node** ：来自节点的日志
- **container** ：命名空间中部署的工作负载的日志

2 指定要收集的审计源列表。有效的源包括：

- **kubeAPI**: 来自 Kubernetes API 服务器的日志
- **openshiftAPI**: 来自 OpenShift API 服务器的日志
- **auditd**: 来自节点 auditd 服务的日志
- **ovn** ：来自开放虚拟网络服务的日志

2. 运行以下命令来应用 **ClusterLogForwarder** CR：

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

第 13 章 调度资源

13.1. 使用节点选择器移动日志记录资源

*节点选择器*指定一个键/值对映射，该映射使用 pod 中指定的自定义标签和选择器定义。

要使 pod 有资格在节点上运行，pod 必须具有与节点上标签相同的键值节点选择器。

13.1.1. 关于节点选择器

您可以使用节点上的 pod 和标签上的节点选择器来控制 pod 的调度位置。使用节点选择器时，OpenShift Container Platform 会将 pod 调度到包含匹配标签的节点。

您可以使用节点选择器将特定的 pod 放置到特定的节点上，集群范围节点选择器将新 pod 放置到集群中的任何特定节点上，以及项目节点选择器，将新 pod 放置到特定的节点上。

例如，作为集群管理员，您可以创建一个基础架构，应用程序开发人员可以通过在创建的每个 pod 中包括节点选择器，将 pod 部署到最接近其地理位置的节点。在本例中，集群由五个数据中心组成，分布在两个区域。在美国，将节点标记为 **us-east**、**us-central** 或 **us-west**。在亚太地区（APAC），将节点标记为 **apac-east** 或 **apac-west**。开发人员可在其创建的 pod 中添加节点选择器，以确保 pod 调度到这些节点上。

如果 **Pod** 对象包含节点选择器，但没有节点具有匹配的标签，则不会调度 pod。

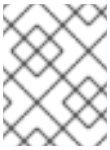
重要

如果您在同一 pod 配置中使用节点选择器和节点关联性，则以下规则控制 pod 放置到节点上：

- 如果同时配置了 **nodeSelector** 和 **nodeAffinity**，则必须满足这两个条件时 pod 才能调度到候选节点。
- 如果您指定了多个与 **nodeAffinity** 类型关联的 **nodeSelectorTerms**，那么其中一个 **nodeSelectorTerms** 满足时 pod 就能调度到节点上。
- 如果您指定了多个与 **nodeSelectorTerms** 关联的 **matchExpressions**，那么只有所有 **matchExpressions** 都满足时 pod 才能调度到节点上。

特定 pod 和节点上的节点选择器

您可以使用节点选择器和标签控制特定 pod 调度到哪些节点上。
要使用节点选择器和标签，首先标记节点以避免 pod 被取消调度，然后将节点选择器添加到 pod。



注意

您不能直接将节点选择器添加到现有调度的 pod 中。您必须标记控制 pod 的对象，如部署配置。

例如，以下 **Node** 对象具有 **region: east** 标签：

带有标识的 Node 对象示例

```
kind: Node
apiVersion: v1
```

```
metadata:
  name: ip-10-0-131-14.ec2.internal
  selfLink: /api/v1/nodes/ip-10-0-131-14.ec2.internal
  uid: 7bc2580a-8b8e-11e9-8e01-021ab4174c74
  resourceVersion: '478704'
  creationTimestamp: '2019-06-10T14:46:08Z'
  labels:
    kubernetes.io/os: linux
    topology.kubernetes.io/zone: us-east-1a
    node.openshift.io/os_version: '4.5'
    node-role.kubernetes.io/worker: ''
    topology.kubernetes.io/region: us-east-1
    node.openshift.io/os_id: rhcos
    node.kubernetes.io/instance-type: m4.large
    kubernetes.io/hostname: ip-10-0-131-14
    kubernetes.io/arch: amd64
    region: east (1)
    type: user-node
#...
```

1 与 pod 节点选择器匹配的标签。

pod 具有 **type: user-node,region: east** 节点选择器：

使用节点选择器的 Pod 对象示例

```
apiVersion: v1
kind: Pod
metadata:
  name: s1
#...
spec:
  nodeSelector: (1)
    region: east
    type: user-node
#...
```

1 与节点标签匹配的节点选择器。节点必须具有每个节点选择器的标签。

使用示例 pod 规格创建 pod 时，它可以调度到示例节点上。

默认集群范围节点选择器

使用默认集群范围节点选择器时，如果您在集群中创建 pod,OpenShift Container Platform 会将默认节点选择器添加到 pod，并将该 pod 调度到具有匹配标签的节点。
例如，以下 **Scheduler** 对象具有默认的集群范围的 **region=east** 和 **type=user-node** 节点选择器：

Scheduler Operator 自定义资源示例

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  name: cluster
#...
```

```
spec:
  defaultNodeSelector: type=user-node,region=east
#...
```

集群中的节点具有 **type=user-node,region=east** 标签：

Node 对象示例

```
apiVersion: v1
kind: Node
metadata:
  name: ci-ln-qg1il3k-f76d1-hlmhl-worker-b-df2s4
#...
  labels:
    region: east
    type: user-node
#...
```

使用节点选择器的 Pod 对象示例

```
apiVersion: v1
kind: Pod
metadata:
  name: s1
#...
spec:
  nodeSelector:
    region: east
#...
```

当您使用示例集群中的 pod spec 创建 pod 时，该 pod 会使用集群范围节点选择器创建，并调度到标记的节点：

在标记的节点上带有 pod 的 pod 列表示例

```
NAME    READY   STATUS    RESTARTS   AGE   IP           NODE
NOMINATED NODE   READINESS GATES
pod-s1   1/1    Running   0          20s   10.131.2.6   ci-ln-qg1il3k-f76d1-hlmhl-worker-b-df2s4
<none>           <none>
```

注意

如果您在其中创建 pod 的项目具有项目节点选择器，则该选择器优先于集群范围节点选择器。如果 pod 没有项目节点选择器，则 pod 不会被创建或调度。

项目节点选择器

使用项目节点选择器时，如果您在此项目中创建 pod,OpenShift Container Platform 会将节点选择器添加到 pod，并将 pod 调度到具有匹配标签的节点。如果存在集群范围默认节点选择器，则以项目节点选择器为准。

例如，以下项目具有 **region=east** 节点选择器：

Namespace 对象示例

```
apiVersion: v1
kind: Namespace
metadata:
  name: east-region
  annotations:
    openshift.io/node-selector: "region=east"
#...
```

以下节点具有 **type=user-node,region=east** 标签：

Node 对象示例

```
apiVersion: v1
kind: Node
metadata:
  name: ci-ln-qg1il3k-f76d1-hlmhl-worker-b-df2s4
#...
  labels:
    region: east
    type: user-node
#...
```

当您使用本例项目中的示例 pod 规格创建 pod 时，pod 会使用项目节点选择器创建，并调度到标记的节点：

Pod 对象示例

```
apiVersion: v1
kind: Pod
metadata:
  namespace: east-region
#...
spec:
  nodeSelector:
    region: east
    type: user-node
#...
```

在标记的节点上带有 pod 的 pod 列表示例

```
NAME    READY   STATUS    RESTARTS   AGE   IP           NODE
NOMINATED NODE   READINESS GATES
pod-s1   1/1     Running   0          20s   10.131.2.6   ci-ln-qg1il3k-f76d1-hlmhl-worker-b-df2s4
<none>           <none>
```

如果 pod 包含不同的节点选择器，则项目中的 pod 不会被创建或调度。例如，如果您将以下 Pod 部署到示例项目中，则不会创建它：

带有无效节点选择器的 Pod 对象示例

```
apiVersion: v1
kind: Pod
metadata:
```

```
    name: west-region
  #...
  spec:
    nodeSelector:
      region: west
  #...
```

13.1.2. Loki pod 放置

您可以通过在 pod 上使用容忍度或节点选择器来控制 Loki pod 在哪些节点上运行，并防止其他工作负载使用这些节点。

您可以使用 LokiStack 自定义资源 (CR) 将容限应用到日志存储 pod，并将污点应用到具有节点规格的节点。节点上的污点是一个 **key:value** 对，它指示节点排斥所有不允许污点的 pod。通过使用不在其他 pod 上的特定 **key:value** 对，可确保只有日志存储 pod 能够在该节点上运行。

带有节点选择器的 LokiStack 示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  template:
    compactor: 1
      nodeSelector:
        node-role.kubernetes.io/infra: "" 2
    distributor:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    gateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    indexGateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    ingester:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    querier:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    queryFrontend:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    ruler:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
# ...
```

1 指定应用到节点选择器的组件 pod 类型。

2 指定移到包含定义标签的节点的 pod。

在上例配置中，所有 Loki pod 都移到包含 **node-role.kubernetes.io/infra: ""** 标签的节点。

带有节点选择器和容限的 LokiStack CR 示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  template:
    compactor:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    distributor:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    indexGateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    ingester:
      nodeSelector:
```

```
      node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved
  querier:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved
  queryFrontend:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved
  ruler:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved
  gateway:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      value: reserved
    - effect: NoExecute
      key: node-role.kubernetes.io/infra
      value: reserved
# ...
```

要配置 LokiStack (CR) 的 **nodeSelector** 和 **tolerations** 字段，您可以使用 **oc explain** 命令查看特定资源的描述和字段：

```
$ oc explain lokistack.spec.template
```

输出示例

```
KIND:     LokiStack
VERSION:  loki.grafana.com/v1

RESOURCE: template <Object>

DESCRIPTION:
     Template defines the resource/limits/tolerations/nodeselectors per
     component

FIELDS:
   compactor <Object>
     Compactor defines the compaction component spec.

   distributor <Object>
     Distributor defines the distributor component spec.
...
```

如需更多信息，您可以添加一个特定字段：

```
$ oc explain lokistack.spec.template.compactor
```

输出示例

```
KIND:     LokiStack
VERSION:  loki.grafana.com/v1

RESOURCE: compactor <Object>

DESCRIPTION:
     Compactor defines the compaction component spec.

FIELDS:
   nodeSelector <map[string]string>
     NodeSelector defines the labels required by a node to schedule the
     component onto it.
...
```

13.1.3. 配置日志记录收集器的资源和调度

管理员可以通过创建位于同一命名空间中的 **ClusterLogging** 自定义资源(CR)来修改收集器的资源或调度，其名称与它支持的 **ClusterLogForwarder** CR 的名称相同。

在部署中使用多个日志转发器时，**ClusterClusterLogging** CR 的适用小节是 **managementState** 和 **collection**。所有其他小节将被忽略。

先决条件

- 有管理员权限。
- 已安装 Red Hat OpenShift Logging Operator 版本 5.8 或更新版本。
- 您已创建了 **ClusterLogForwarder** CR。

流程

1. 创建支持现有 **ClusterLogForwarder** CR 的 **ClusterLogging** CR：

   ClusterLogging CR YAML 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name:  <name> 1
  namespace: <namespace> 2
spec:
  managementState: "Managed"
  collection:
    type: "vector"
    tolerations:
    - key: "logging"
      operator: "Exists"
      effect: "NoExecute"
      tolerationSeconds: 6000
    resources:
      limits:
        memory: 1Gi
      requests:
        cpu: 100m
        memory: 1Gi
    nodeSelector:
      collector: needed
# ...
```

   1 名称必须与 **ClusterLogForwarder** CR 的名称相同。

   2 命名空间必须与 **ClusterLogForwarder** CR 相同。

2. 运行以下命令来应用 **ClusterLogging** CR：

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

13.1.4. 查看日志记录收集器 Pod

您可以查看日志记录收集器 Pod 及其运行的对应节点。

流程

- 在项目中运行以下命令查看日志记录收集器 Pod 及其详情：

```
$ oc get pods --selector component=collector -o wide -n <project_name>
```

   输出示例

```
NAME          READY  STATUS    RESTARTS   AGE     IP           NODE
NOMINATED NODE   READINESS GATES
collector-8d69v  1/1    Running   0          134m    10.130.2.30   master1.example.com
```

```
<none>          <none>
collector-bd225  1/1    Running   0          134m    10.131.1.11   master2.example.com
<none>          <none>
collector-cvrzs  1/1    Running   0          134m    10.130.0.21   master3.example.com   <none>
<none>
collector-gpqg2  1/1    Running   0          134m    10.128.2.27   worker1.example.com
<none>          <none>
collector-l9j7j  1/1    Running   0          134m    10.129.2.31   worker2.example.com   <none>
<none>
```

13.1.5. 其他资源

- 使用节点选择器将 pod 放置到特定节点

13.2. 使用污点和容限来控制日志记录 POD 放置

通过污点和容限，节点可以控制哪些 pod 应该（或不应该）调度到节点上。

13.2.1. 了解污点和容限

通过使用*污点（taint）*，节点可以拒绝调度 pod，除非 pod 具有匹配的*容限（toleration）*。

您可以通过节点规格（**NodeSpec**）将污点应用到节点，并通过 **Pod** 规格（**PodSpec**）将容限应用到 pod。当您应用污点时，调度程序无法将 pod 放置到该节点上，除非 pod 可以容限该污点。

节点规格中的污点示例

```
apiVersion: v1
kind: Node
metadata:
  name: my-node
#...
spec:
  taints:
  - effect: NoExecute
    key: key1
    value: value1
#...
```

Pod 规格中的容限示例

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
#...
spec:
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoExecute"
    tolerationSeconds: 3600
#...
```

污点与容限由 key、value 和 effect 组成。

表 13.1. 污点和容限组件

参数	描述
key	**key** 是任意字符串，最多 253 个字符。key 必须以字母或数字开头，可以包含字母、数字、连字符、句点和下划线。
value	**value** 是任意字符串，最多 63 个字符。value 必须以字母或数字开头，可以包含字母、数字、连字符、句点和下划线。
effect	effect 的值包括： **NoSchedule** [1]： • 与污点不匹配的新 pod 不会调度到该节点上。 • 该节点上现有的 pod 会保留。 **PreferNoSchedule**： • 与污点不匹配的新 pod 可以调度到该节点上，但调度程序会尽量不这样调度。 • 该节点上现有的 pod 会保留。 **NoExecute**： • 与污点不匹配的新 pod 无法调度到该节点上。 • 节点上没有匹配容限的现有 pod 将被移除。
operator	**Equal**：**key**/**value**/**effect** 参数必须匹配。这是默认值。 **Exists**：**key**/**effect** 参数必须匹配。您必须保留一个空的 **value** 参数，这将匹配任何值。

1. 如果向 control plane 节点添加了一个 **NoSchedule** 污点，节点必须具有 **node-role.kubernetes.io/master=:NoSchedule** 污点，这默认会添加。
例如：

```
apiVersion: v1
kind: Node
metadata:
  annotations:
    machine.openshift.io/machine: openshift-machine-api/ci-ln-62s7gtb-f76d1-v8jxv-master-0
```

```
    machineconfiguration.openshift.io/currentConfig: rendered-master-
cdc1ab7da414629332cc4c3926e6e59c
  name: my-node
#...
spec:
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
#...
```

容限与污点匹配：

- 如果 **operator** 参数设为 **Equal**：
  - **key** 参数相同；
  - **value** 参数相同；
  - **effect** 参数相同。
- 如果 **operator** 参数设为 **Exists**：
  - **key** 参数相同；
  - **effect** 参数相同。

OpenShift Container Platform 中内置了以下污点：

- **node.kubernetes.io/not-ready**：节点未就绪。这与节点状况 **Ready=False** 对应。
- **node.kubernetes.io/unreachable**：节点无法从节点控制器访问。这与节点状况 **Ready=Unknown** 对应。
- **node.kubernetes.io/memory-pressure**：节点存在内存压力问题。这与节点状况 **MemoryPressure=True** 对应。
- **node.kubernetes.io/disk-pressure**：节点存在磁盘压力问题。这与节点状况 **DiskPressure=True** 对应。
- **node.kubernetes.io/network-unavailable**：节点网络不可用。
- **node.kubernetes.io/unschedulable**：节点不可调度。
- **node.cloudprovider.kubernetes.io/uninitialized**：当节点控制器通过外部云提供商启动时，在节点上设置这个污点来将其标记为不可用。在云控制器管理器中的某个控制器初始化这个节点后，kubelet 会移除此污点。
- **node.kubernetes.io/pid-pressure** ：节点具有 pid 压力。这与节点状况 **PIDPressure=True** 对应。

重要

OpenShift Container Platform 不设置默认的 pid.available **evictionHard**。

13.2.2. Loki pod 放置

您可以通过在 pod 上使用容忍度或节点选择器来控制 Loki pod 在哪些节点上运行，并防止其他工作负载使用这些节点。

您可以使用 LokiStack 自定义资源 (CR) 将容限应用到日志存储 pod，并将污点应用到具有节点规格的节点。节点上的污点是一个 **key:value** 对，它指示节点排斥所有不允许污点的 pod。通过使用不在其他 pod 上的特定 **key:value** 对，可确保只有日志存储 pod 能够在该节点上运行。

带有节点选择器的 LokiStack 示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
# ...
  template:
    compactor: 1
      nodeSelector:
        node-role.kubernetes.io/infra: "" 2
    distributor:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    gateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    indexGateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    ingester:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    querier:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    queryFrontend:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    ruler:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
# ...
```

1 指定应用到节点选择器的组件 pod 类型。

2 指定移到包含定义标签的节点的 pod。

在上例配置中，所有 Loki pod 都移到包含 **node-role.kubernetes.io/infra: ""** 标签的节点。

带有节点选择器和容限的 LokiStack CR 示例

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
```

```
  namespace: openshift-logging
spec:
# ...
  template:
    compactor:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    distributor:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    indexGateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    ingester:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    querier:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
```

```
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    queryFrontend:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    ruler:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    gateway:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
# ...
```

要配置 LokiStack (CR) 的 **nodeSelector** 和 **tolerations** 字段，您可以使用 **oc explain** 命令查看特定资源的描述和字段：

```
$ oc explain lokistack.spec.template
```

输出示例

```
KIND:     LokiStack
VERSION:  loki.grafana.com/v1

RESOURCE: template <Object>

DESCRIPTION:
     Template defines the resource/limits/tolerations/nodeselectors per
     component
```

```
FIELDS:
  compactor <Object>
    Compactor defines the compaction component spec.

  distributor <Object>
    Distributor defines the distributor component spec.
...
```

如需更多信息，您可以添加一个特定字段：

```
$ oc explain lokistack.spec.template.compactor
```

输出示例

```
KIND:     LokiStack
VERSION:  loki.grafana.com/v1

RESOURCE: compactor <Object>

DESCRIPTION:
     Compactor defines the compaction component spec.

FIELDS:
   nodeSelector <map[string]string>
     NodeSelector defines the labels required by a node to schedule the
     component onto it.
...
```

13.2.3. 使用容忍度来控制日志收集器 pod 放置

默认情况下，日志收集器 pod 具有以下 **tolerations** 配置：

```
apiVersion: v1
kind: Pod
metadata:
  name: collector-example
  namespace: openshift-logging
spec:
# ...
  collection:
    type: vector
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/master
      operator: Exists
    - effect: NoSchedule
      key: node.kubernetes.io/disk-pressure
      operator: Exists
    - effect: NoExecute
      key: node.kubernetes.io/not-ready
      operator: Exists
    - effect: NoExecute
      key: node.kubernetes.io/unreachable
      operator: Exists
```

```
    - effect: NoSchedule
      key: node.kubernetes.io/memory-pressure
      operator: Exists
    - effect: NoSchedule
      key: node.kubernetes.io/pid-pressure
      operator: Exists
    - effect: NoSchedule
      key: node.kubernetes.io/unschedulable
      operator: Exists
# ...
```

先决条件

- 已安装 Red Hat OpenShift Logging Operator 和 OpenShift CLI (**oc**)。

流程

1. 运行以下命令，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>
```

示例命令

```
$ 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：

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  collection:
    type: vector
    tolerations:
    - key: collector (1)
      operator: Exists (2)
      effect: NoExecute (3)
      tolerationSeconds: 6000 (4)
    resources:
      limits:
        memory: 2Gi
      requests:
        cpu: 100m
        memory: 1Gi
# ...
```

(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。

13.2.4. 配置日志记录收集器的资源和调度

管理员可以通过创建位于同一命名空间中的 **ClusterLogging** 自定义资源(CR)来修改收集器的资源或调度，其名称与它支持的 **ClusterLogForwarder** CR 的名称相同。

在部署中使用多个日志转发器时，**ClusterClusterLogging** CR 的适用小节是 **managementState** 和 **collection**。所有其他小节将被忽略。

先决条件

- 有管理员权限。
- 已安装 Red Hat OpenShift Logging Operator 版本 5.8 或更新版本。
- 您已创建了 **ClusterLogForwarder** CR。

流程

1. 创建支持现有 **ClusterLogForwarder** CR 的 **ClusterLogging** CR：

   ClusterLogging CR YAML 示例

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name:  <name> 1
  namespace: <namespace> 2
spec:
  managementState: "Managed"
  collection:
    type: "vector"
    tolerations:
    - key: "logging"
      operator: "Exists"
      effect: "NoExecute"
      tolerationSeconds: 6000
    resources:
      limits:
        memory: 1Gi
      requests:
        cpu: 100m
        memory: 1Gi
    nodeSelector:
      collector: needed
# ...
```

1. 名称必须与 **ClusterLogForwarder** CR 的名称相同。
2. 命名空间必须与 **ClusterLogForwarder** CR 相同。

2. 运行以下命令来应用 **ClusterLogging** CR：

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

13.2.5. 查看日志记录收集器 Pod

您可以查看日志记录收集器 Pod 及其运行的对应节点。

流程

- 在项目中运行以下命令查看日志记录收集器 Pod 及其详情：

```
$ oc get pods --selector component=collector -o wide -n <project_name>
```

输出示例

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

13.2.6. 其他资源

- 使用节点污点控制 pod 放置

第 14 章 卸载日志记录

您可以通过删除安装的 Operator 和相关自定义资源 (CR) 从 OpenShift Container Platform 集群中删除日志记录。

14.1. 卸载日志记录

您可以通过删除 Red Hat OpenShift Logging Operator 和 **ClusterLogging** 自定义资源(CR)来停止聚合日志。

先决条件

- 有管理员权限。
- 您可以访问 OpenShift Container Platform Web 控制台的 **Administrator** 视角。

流程

1. 进入 **Administration** → **Custom Resource Definitions**页面，然后点 **ClusterLogging**。
2. 在 **Custom Resource Definition Details**页面中点 **Instances**。
3. 点实例旁的 Options 菜单 ,然后选择 **Delete ClusterLogging**。
4. 进入到 **Administration** → **Custom Resource Definitions**页面。
5. 点 **ClusterLogging** 旁边的 Options 菜单 ,然后选择 **Delete Custom Resource Definition**。

> **警告**
>
> 删除 **ClusterLogging** CR 不会删除持久性卷声明（PVC）。要删除剩余的 PVC、持久性卷（PV）和相关数据，您必须执行进一步操作。释放或删除 PVC 可能会导致 PV 删除并导致数据丢失。

6. 如果您已创建了 **ClusterLogForwarder** CR，点 **ClusterLogForwarder** 旁边的选项菜单 ，然后点 **Delete Custom Resource Definition**。
7. 进入 **Operators** → **Installed Operators** 页面。
8. 点 Red Hat OpenShift Logging Operator 旁边的选项菜单 ，然后点 **Uninstall Operator**。
9. 可选：删除 **openshift-logging** 项目。

警告

删除 **openshift-logging** 项目会删除该命名空间中的所有内容，包括任何持久性卷声明 (PVC)。如果要保留日志记录数据，请不要删除 **openshift-logging** 项目。

a. 进入 **Home → Projects** 页面。

b. 点 **openshift-logging** 项目旁边的选项菜单  ，然后点 **Delete Project**。

c. 通过在对话框中输入 **openshift-logging** 并点 **Delete** 来确认删除。

14.2. 删除日志记录 PVC

要保留持久性卷声明 (PVC) 以与其他 pod 重复使用，保留标签或 PVC 名称，以回收 PVC。如果您不想保留 PVC，可以删除它们。如果要恢复存储空间，您还可以删除持久性卷 (PV)。

先决条件

- 有管理员权限。
- 您可以访问 OpenShift Container Platform Web 控制台的 **Administrator** 视角。

流程

1. 进入 **Storage → Persistent Volume Claims** 页面。
2. 点每个 PVC 旁边的选项菜单  ，然后选择 **Delete Persistent Volume Claim**。

14.3. 卸载 LOKI

先决条件

- 有管理员权限。
- 您可以访问 OpenShift Container Platform Web 控制台的 **Administrator** 视角。
- 如果您还没有删除 Red Hat OpenShift Logging Operator 和相关资源，已从 **ClusterLogging** 自定义资源中删除了对 LokiStack 的引用。

流程

1. 进入 **Administration → Custom Resource Definitions** 页面，然后点 **LokiStack**。
2. 在 **Custom Resource Definition Details** 页面中点 **Instances**。

3. 点实例旁的选项菜单 ，然后点 **Delete LokiStack**。
4. 进入到 **Administration → Custom Resource Definitions**页面。
5. 点 **LokiStack** 旁边的选项菜单 ，然后选择 **Delete Custom Resource Definition**。
6. 删除对象存储 secret。
7. 进入 **Operators → Installed Operators** 页面。
8. 点 Loki Operator 旁边的选项菜单 ，然后点 **Uninstall Operator**。
9. 可选：删除 **openshift-operators-redhat** 项目。

重要

如果在这个命名空间中安装了其他全局 Operator，请不要删除 **openshift-operators-redhat** 项目。

a. 进入 **Home → Projects** 页面。

b. 点 **openshift-operators-redhat** 项目旁的选项菜单 ，然后点 **Delete Project**。

c. 通过在对话框中输入 **openshift-operators-redhat** 并点 **Delete** 来确认删除。

14.4. 卸载 ELASTICSEARCH

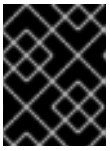
先决条件

- 有管理员权限。
- 您可以访问 OpenShift Container Platform Web 控制台的 **Administrator** 视角。
- 如果您还没有删除 Red Hat OpenShift Logging Operator 和相关资源，则必须从 **ClusterLogging** 自定义资源中删除对 Elasticsearch 的引用。

流程

1. 进入 **Administration → Custom Resource Definitions**页面，然后点 **Elasticsearch**。
2. 在 **Custom Resource Definition Details**页面中点 **Instances**。
3. 点实例旁的选项菜单 ，然后点 **Delete Elasticsearch**。
4. 进入到 **Administration → Custom Resource Definitions**页面。

5. 点 **Elasticsearch** 旁边的 Options 菜单 并选择 **Delete Custom Resource Definition**。
6. 删除对象存储 secret。
7. 进入 **Operators → Installed Operators** 页面。
8. 点 OpenShift Elasticsearch Operator 旁边的选项菜单 ，然后点 **Uninstall Operator**。
9. 可选：删除 **openshift-operators-redhat** 项目。



重要

如果在这个命名空间中安装了其他全局 Operator，请不要删除 **openshift-operators-redhat** 项目。

a. 进入 **Home → Projects** 页面。

b. 点 **openshift-operators-redhat** 项目旁的选项菜单 ，然后点 **Delete Project**。

c. 通过在对话框中输入 **openshift-operators-redhat** 并点 **Delete** 来确认删除。

14.5. 使用 CLI 从集群中删除 OPERATOR

集群管理员可以使用 CLI 从所选命名空间中删除已安装的 Operator。

先决条件

- 可以使用具有 **cluster-admin** 权限的账户访问 OpenShift Container Platform 集群。
- OpenShift CLI (**oc**)安装在您的工作站上。

流程

1. 确保在 **currentCSV** 字段中标识了订阅 Operator 的最新版本（如 **serverless-operator**）。

```
$ oc get subscription.operators.coreos.com serverless-operator -n openshift-serverless -o yaml | grep currentCSV
```

输出示例

```
currentCSV: serverless-operator.v1.28.0
```

2. 删除订阅（如 **serverless-operator**）：

```
$ oc delete subscription.operators.coreos.com serverless-operator -n openshift-serverless
```

输出示例

```
subscription.operators.coreos.com "serverless-operator" deleted
```

3. 使用上一步中的 **currentCSV** 值来删除目标命名空间中相应 Operator 的 CSV：

```
$ oc delete clusterserviceversion serverless-operator.v1.28.0 -n openshift-serverless
```

输出示例

```
clusterserviceversion.operators.coreos.com "serverless-operator.v1.28.0" deleted
```

其他资源

- 手动重新声明持久性卷

第 15 章 日志记录字段

日志导出的日志记录中可以包括以下字段。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。

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	**HAPPY**

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]

@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	**2015-01-24 14:06:05.071000000 Z**

主机名

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

IPADDR4

源服务器的 IPv4 地址。可以是一个数组。

数据类型	ip

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

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**，当日志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	**info**

PID
日志记录实体的进程 ID（若有）。

数据类型	关键字

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 16 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

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

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 17 章 KUBERNETES

特定于 Kubernetes 元数据的命名空间

数据类型	group

17.1. KUBERNETES.POD_NAME

pod 的名称

数据类型	关键字

17.2. KUBERNETES.POD_ID

pod 的 Kubernetes ID

数据类型	关键字

17.3. KUBERNETES.NAMESPACE_NAME

Kubernetes 中命名空间的名称

数据类型	关键字

17.4. KUBERNETES.NAMESPACE_ID

Kubernetes 中命名空间的 ID

数据类型	关键字

17.5. KUBERNETES.HOST

Kubernetes 节点名称

数据类型	关键字

17.6. KUBERNETES.CONTAINER_NAME

Kubernetes 中容器的名称

数据类型	关键字

17.7. KUBERNETES.ANNOTATIONS

与 Kubernetes 对象关联的注解

数据类型	group

17.8. KUBERNETES.LABELS

原始 Kubernetes Pod 上存在的标签

数据类型	group

17.9. 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

17.9.1. kubernetes.event.verb

事件类型，**ADDED**、**MODIFIED** 或 **DELETED**

数据类型	关键字
示例值	**ADDED**

17.9.2. kubernetes.event.metadata

与事件创建位置和时间相关的信息

数据类型	group

17.9.2.1. 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	**java-mainclass-1.14d888a4cfc24890**

17.9.2.2. 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	**default**

17.9.2.3. kubernetes.event.metadata.selfLink

到事件的链接

数据类型	关键字
示例值	**/api/v1/namespaces/javaj/events/java-mainclass-1.14d888a4cfc24890**

17.9.2.4. kubernetes.event.metadata.uid

事件的唯一 ID

数据类型	关键字
示例值	**d828ac69-7b58-11e7-9cf5-5254002f560c**

17.9.2.5. 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	**311987**

17.9.3. kubernetes.event.involvedObject

事件所针对的对象。

数据类型	group

17.9.3.1. kubernetes.event.involvedObject.kind

对象的类型

数据类型	关键字
示例值	**ReplicationController**

17.9.3.2. 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	**default**

17.9.3.3. kubernetes.event.involvedObject.name

触发事件的对象名称

数据类型	关键字
示例值	**java-mainclass-1**

17.9.3.4. kubernetes.event.involvedObject.uid

对象的唯一 ID

数据类型	关键字
示例值	**e6bff941-76a8-11e7-8193-5254002f560c**

17.9.3.5. kubernetes.event.involvedObject.apiVersion

kubernetes master API 的版本

数据类型	关键字
示例值	**v1**

17.9.3.6. 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	**308882**

17.9.4. 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	**SuccessfulCreate**

17.9.5. kubernetes.event.source_component

报告此事件的组件

数据类型	关键字
示例值	**replication-controller**

17.9.6. kubernetes.event.firstTimestamp

事件首次记录的时间

数据类型	date
示例值	**2017-08-07 10:11:57.000000000 Z**

17.9.7. kubernetes.event.count

发生此事件的次数

数据类型	整数
示例值	**1**

17.9.8. 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	**Normal**

第 18 章 OPENSHIFT

openshift-logging 特定元数据的命名空间

数据类型	group

18.1. OPENSHIFT.LABELS

由 Cluster Log Forwarder 配置添加的标签

数据类型	group

第 19 章 API 参考

19.1. 5.6 日志记录 API 参考

19.1.1. Logging 5.6 API 参

19.1.1.1. ClusterLogForwarder

ClusterLogForwarder 是一个 API，用于配置转发日志。

您可以通过指定一个 **pipelines** 列表来配置转发，该列表从一组命名输入转发到一组命名输出。

常用日志类别有内置输入名称，您可以定义自定义输入来执行额外的过滤。

默认 openshift 日志存储有一个内置输出名称，但您可以使用 URL 和其他连接信息定义您自己的输出，将日志转发到集群内部或处理器的其他连接信息。

如需了解更多详细信息，请参阅 API 字段的文档。

属性	类型	描述
spec	对象	ClusterLogForwarder 所需的行为规格
status	对象	ClusterLogForwarder 的状态

19.1.1.1.1. .spec

19.1.1.1.1.1. 描述

ClusterLogForwarderSpec 定义如何将日志转发到远程目标。

19.1.1.1.1.1.1. 类型

- 对象

属性	类型	描述
输入	数组	**（可选）** 输入被命名过滤器，用于转发日志消息。
outputDefaults	对象	**（可选）** DEPRECATED OutputDefaults 为默认存储明确指定 forwarder 配置。
输出	数组	**（可选）** 输出的名称是日志消息的目的地。

属性	类型	描述
pipelines	数组	Pipelines 将一组输入选择的消息转发到一组输出。

19.1.1.1.2. .spec.inputs[]

19.1.1.1.2.1. 描述

InputSpec 定义日志消息的选择器。

19.1.1.1.2.1.1. 类型

- 数组

属性	类型	描述
application	对象	**（可选）** 如果存在，应用程序启用命名的**应用程序**日志集合
name	字符串	用于引用**管道**输入的名称。

19.1.1.1.3. .spec.inputs[].application

19.1.1.1.3.1. 描述

应用程序日志选择器。必须满足选择器中的所有条件（逻辑 AND）才能选择日志。

19.1.1.1.3.1.1. 类型

- 对象

属性	类型	描述
命名空间	数组	**（可选）** 从中收集应用程序日志的命名空间。
selector	对象	**（可选）** 匹配标签的 pod 的日志的 Selector。

19.1.1.1.4. .spec.inputs[].application.namespaces[]

19.1.1.1.4.1. 描述

19.1.1.1.4.1.1. 类型

- 数组

19.1.1.1.5. .spec.inputs[].application.selector

19.1.1.1.5.1. 描述

标签选择器，即一组资源的标签查询。

19.1.1.1.5.1.1. 类型

- 对象

属性	类型	描述
matchLabels	对象	（可选） matchLabels 是 {key,value} 对的映射。matchLabels 中的单个 {key,value}

19.1.1.1.6. .spec.inputs[].application.selector.matchLabels

19.1.1.1.6.1. 描述

19.1.1.1.6.1.1. 类型

- 对象

19.1.1.1.7. .spec.outputDefaults

19.1.1.1.7.1. 描述

19.1.1.1.7.1.1. 类型

- 对象

属性	类型	描述
elasticsearch	对象	（可选） Elasticsearch OutputSpec 默认值

19.1.1.1.8. .spec.outputDefaults.elasticsearch

19.1.1.1.8.1. 描述

ElasticsearchStructuredSpec 与结构化日志更改相关的 spec，以确定 elasticsearch 索引

19.1.1.1.8.1.1. 类型

- 对象

属性	类型	描述
enableStructuredContainerLogs	bool	**（可选）** 启用 StructuredContainerLogs 启用多容器结构化日志来允许
structuredTypeKey	字符串	**（可选）** StructuredTypeKey 指定要用作 elasticsearch 索引名称的元数据键
structuredTypeName	字符串	**（可选）** StructuredTypeName 指定 elasticsearch 模式的名称

19.1.1.1.9. .spec.outputs[]

19.1.1.1.9.1. 描述

输出定义日志消息的目的地。

19.1.1.1.9.1.1. 类型

- 数组

属性	类型	描述
syslog	对象	**（可选）**
fluentdForward	对象	**（可选）**
elasticsearch	对象	**（可选）**
kafka	对象	**（可选）**
cloudwatch	对象	**（可选）**
loki	对象	**（可选）**
googleCloudLogging	对象	**（可选）**
splunk	对象	**（可选）**
name	字符串	用于引用来自**管道**的输出的名称。
secret	对象	**（可选）** 用于身份验证的 Secret。
tls	对象	TLS 包含控制 TLS 客户端连接上的选项的设置。

属性	类型	描述
type	字符串	输出插件的类型。
url	字符串	**（可选）** 将日志记录发送到的 URL。

19.1.1.1.10. .spec.outputs[].secret

19.1.1.1.10.1. 描述

OutputSecretSpec 是仅包含名称的一个 secret 引用，没有命名空间。

19.1.1.1.10.1.1. 类型

- 对象

属性	类型	描述
name	字符串	为日志转发器 secret 配置的命名空间中 secret 的名称。

19.1.1.1.11. .spec.outputs[].tls

19.1.1.1.11.1. 描述

OutputTLSSpec 包含与输出类型无关的 TLS 连接选项。

19.1.1.1.11.1.1. 类型

- 对象

属性	类型	描述
insecureSkipVerify	bool	如果 InsecureSkipVerify 为 true，则将配置 TLS 客户端来忽略证书的错误。

19.1.1.1.12. .spec.pipelines[]

19.1.1.1.12.1. 描述

PipelinesSpec 将一组输入链接到一组输出。

19.1.1.1.12.1.1. 类型

- 数组

属性	类型	描述
detectMultilineErrors	bool	**（可选）** DetectMultilineErrors 启用容器日志的多行错误检测
inputRefs	数组	inputRefs 列出此管道输入的名称（**input.name**）。
labels	对象	**（可选）** 应用于通过此管道传递的记录的标签。
name	字符串	**（可选）** 名称是可选的，但如果提供，则必须在 **pipelines** 列表中唯一。
outputRefs	数组	outputRefs 列出此管道输出的名称（**output.name**）。
parse	字符串	**（可选）** Parse 允许将日志条目解析为结构化日志中

19.1.1.1.13. .spec.pipelines[].inputRefs[]

19.1.1.1.13.1. 描述

19.1.1.1.13.1.1. 类型

- 数组

19.1.1.1.14. .spec.pipelines[].labels

19.1.1.1.14.1. 描述

19.1.1.1.14.1.1. 类型

- 对象

19.1.1.1.15. .spec.pipelines[].outputRefs[]

19.1.1.1.15.1. 描述

19.1.1.1.15.1.1. 类型

- 数组

19.1.1.1.16. .status

19.1.1.1.16.1. 描述

ClusterLogForwarderStatus 定义 ClusterLogForwarder 的观察状态

19.1.1.1.16.1.1. 类型

- 对象

属性	类型	描述
conditions	对象	日志转发器的条件。
输入	Conditions	输入将输入名称映射到输入条件。
输出	Conditions	输出将输出名称映射到输出的条件。
pipelines	Conditions	Pipelines 将管道名称映射到管道的条件。

19.1.1.1.17. .status.conditions

19.1.1.1.17.1. 描述

19.1.1.1.17.1.1. 类型

- 对象

19.1.1.1.18. .status.inputs

19.1.1.1.18.1. 描述

19.1.1.1.18.1.1. 类型

- Conditions

19.1.1.1.19. .status.outputs

19.1.1.1.19.1. 描述

19.1.1.1.19.1.1. 类型

- Conditions

19.1.1.1.20. .status.pipelines

19.1.1.1.20.1. 描述

19.1.1.1.20.1.1. 类型

- conditions== ClusterLogging 一个 Red Hat OpenShift Logging 实例。ClusterLogging 是 clusterloggings API 的 Schema

属性	类型	描述
spec	对象	ClusterLogging 所需的行为规格
status	对象	Status 定义 ClusterLogging 的观察状态

19.1.1.1.21. .spec

19.1.1.1.21.1. 描述

ClusterLoggingSpec 定义 ClusterLogging 的所需状态

19.1.1.1.21.1.1. 类型

- 对象

属性	类型	描述
集合	对象	集群的 Collection 组件的规格
curation	对象	**(已弃用）（可选）** 已弃用。集群的 Curation 组件的规格
forwarder	对象	**(已弃用）（可选）** 已弃用。集群的 Forwarder 组件的规格
logStore	对象	**（可选）** 集群的日志存储组件的规格
managementState	字符串	**（可选）** 如果 Operator 是 'Managed' 或 'Unmanaged'，则查询
visualization	对象	**（可选）** 集群的可视化组件的规格

19.1.1.1.22. .spec.collection

19.1.1.1.22.1. 描述

这是包含日志和事件集合信息的结构

19.1.1.1.22.1.1. 类型

- 对象

属性	类型	描述
资源	对象	**（可选）** 收集器的资源要求
nodeSelector	对象	**（可选）** 定义 Pod 调度到哪些节点上。
容限（tolerations）	数组	**（可选）** 定义 Pod 将接受的容限
fluentd	对象	**（可选）** Fluentd 代表类型为 fluentd 的转发器的配置。
logs	对象	**(已弃用)（可选）** 已弃用。集群的 Log Collection 规格
type	字符串	**（可选）** 要配置的 Log Collection 类型

19.1.1.1.23. .spec.collection.fluentd

19.1.1.1.23.1. 描述

FluentdForwarderSpec 代表类型为 fluentd 的转发器的配置。

19.1.1.1.23.1.1. 类型

- 对象

属性	类型	描述
buffer	对象	
inFile	对象	

19.1.1.1.24. .spec.collection.fluentd.buffer

19.1.1.1.24.1. 描述

FluentdBufferSpec 代表 fluentd 缓冲参数的子集，用于调整所有 fluentd 输出的缓冲配置。它支持参数子集来配置缓冲区和队列大小、清空操作和重试清除。

有关常规参数，请参阅：https://docs.fluentd.org/configuration/buffer-section#buffering-parameters

有关 flush 参数，请参阅：https://docs.fluentd.org/configuration/buffer-section#flushing-parameters

有关重试参数请参考：https://docs.fluentd.org/configuration/buffer-section#retries-parameters

19.1.1.1.24.1.1. 类型

- 对象

属性	类型	描述
chunkLimitSize	字符串	**（可选）** ChunkLimitSize 代表每个块的最大大小。事件将是
flushInterval	字符串	**（可选）** FlushInterval 代表两个连续清除之间等待的时长
flushMode	字符串	**（可选）** FlushMode 代表要写入块的清除线程的模式。模式
flushThreadCount	int	**（可选）** FlushThreadCount reprents 缓冲区使用的线程数量
overflowAction	字符串	**（可选）** OverflowAction 代表 fluentd 缓冲插件的操作
retryMaxInterval	字符串	**（可选）** RetryMaxInterval 代表 exponential backoff 的最大时间间隔
retryTimeout	字符串	**（可选）** RetryTimeout 代表在放弃前尝试重试的最长时间
retryType	字符串	**（可选）** RetryType 代表重试清除操作的类型。flush 操作可以
retryWait	字符串	**（可选）** RetryWait 代表两个连续重试刷新之间的持续时间
totalLimitSize	字符串	**（可选）** TotalLimitSize 代表每个 fluentd 允许的节点空间阈值

19.1.1.1.25. .spec.collection.fluentd.inFile

19.1.1.1.25.1. 描述

FluentdInFileSpec 代表 fluentd in-tail 插件参数的子集，用于调整所有 fluentd in-tail 输入的配置。

有关常规参数，请参阅： https://docs.fluentd.org/input/tail#parameters

19.1.1.1.25.1.1. 类型

- 对象

属性	类型	描述
readLinesLimit	int	**（可选）** ReadlinesLimit 代表要随每个 I/O 操作读取的行数

19.1.1.1.26. .spec.collection.logs

19.1.1.1.26.1. 描述

19.1.1.1.26.1.1. 类型

- 对象

属性	类型	描述
fluentd	对象	Fluentd Log Collection 组件的规格
type	字符串	要配置的日志集合类型

19.1.1.1.27. .spec.collection.logs.fluentd

19.1.1.1.27.1. 描述

CollectorSpec 是 spec，用于定义收集器的调度和资源

19.1.1.1.27.1.1. 类型

- 对象

属性	类型	描述
nodeSelector	对象	**（可选）** 定义 Pod 调度到哪些节点上。
资源	对象	**（可选）** 收集器的资源要求
容限（tolerations）	数组	**（可选）** 定义 Pod 将接受的容限

19.1.1.1.28. .spec.collection.logs.fluentd.nodeSelector

19.1.1.1.28.1. 描述

19.1.1.1.28.1.1. 类型

- 对象

19.1.1.1.29. .spec.collection.logs.fluentd.resources

19.1.1.1.29.1. 描述

19.1.1.1.29.1.1. 类型

- 对象

属性	类型	描述
limits	对象	**（可选）** 限制描述了允许的最大计算资源量。
requests	对象	**（可选）** 请求描述了所需的最少计算资源。

19.1.1.1.30. .spec.collection.logs.fluentd.resources.limits

19.1.1.1.30.1. 描述

19.1.1.1.30.1.1. 类型

- 对象

19.1.1.1.31. .spec.collection.logs.fluentd.resources.requests

19.1.1.1.31.1. 描述

19.1.1.1.31.1.1. 类型

- 对象

19.1.1.1.32. .spec.collection.logs.fluentd.tolerations[]

19.1.1.1.32.1. 描述

19.1.1.1.32.1.1. 类型

- 数组

属性	类型	描述
effect	字符串	**（可选）** 效果表示要匹配的污点效果。空意味着匹配所有污点效果。
key	字符串	**（可选）** key 是容限应用到的污点键。empty 表示与所有污点键匹配。

属性	类型	描述
operator	字符串	**（可选）** Operator 代表键与值的关系。
tolerationSeconds	int	**（可选）** TolerationSeconds 代表容限的期间（必须是
value	字符串	**（可选）** 值是容限匹配的污点值。

19.1.1.1.33. .spec.collection.logs.fluentd.tolerations[].tolerationSeconds

19.1.1.1.33.1. 描述

19.1.1.1.33.1.1. 类型

- int

19.1.1.1.34. .spec.curation

19.1.1.1.34.1. 描述

这是包含日志策展信息的结构 (Curator)

19.1.1.1.34.1.1. 类型

- 对象

属性	类型	描述
curator	对象	要配置的策展规格
type	字符串	要配置的策展类型

19.1.1.1.35. .spec.curation.curator

19.1.1.1.35.1. 描述

19.1.1.1.35.1.1. 类型

- 对象

属性	类型	描述
nodeSelector	对象	定义 Pod 调度到哪些节点上。

属性	类型	描述
资源	对象	**（可选）** Curator 的资源要求
调度	字符串	Curator 作业运行的 cron 调度。默认为 "30 3 * * *"
容限（tolerations）	数组	

19.1.1.1.36. .spec.curation.curator.nodeSelector

19.1.1.1.36.1. 描述

19.1.1.1.36.1.1. 类型

- 对象

19.1.1.1.37. .spec.curation.curator.resources

19.1.1.1.37.1. 描述

19.1.1.1.37.1.1. 类型

- 对象

属性	类型	描述
limits	对象	**（可选）** 限制描述了允许的最大计算资源量。
requests	对象	**（可选）** 请求描述了所需的最少计算资源。

19.1.1.1.38. .spec.curation.curator.resources.limits

19.1.1.1.38.1. 描述

19.1.1.1.38.1.1. 类型

- 对象

19.1.1.1.39. .spec.curation.curator.resources.requests

19.1.1.1.39.1. 描述

19.1.1.1.39.1.1. 类型

- 对象

19.1.1.1.40. .spec.curation.curator.tolerations[]

19.1.1.1.40.1. 描述

19.1.1.1.40.1.1. 类型

- 数组

属性	类型	描述
effect	字符串	**（可选）** 效果表示要匹配的污点效果。空意味着匹配所有污点效果。
key	字符串	**（可选）** key 是容限应用到的污点键。empty 表示与所有污点键匹配。
operator	字符串	**（可选）** Operator 代表键与值的关系。
tolerationSeconds	int	**（可选）** TolerationSeconds 代表容限的期间（必须是
value	字符串	**（可选）** 值是容限匹配的污点值。

19.1.1.1.41. .spec.curation.curator.tolerations[].tolerationSeconds

19.1.1.1.41.1. 描述

19.1.1.1.41.1.1. 类型

- int

19.1.1.1.42. .spec.forwarder

19.1.1.1.42.1. 描述

ForwarderSpec 包含特定转发器实现的全局调优参数。一般用途不需要此字段，用户可以熟悉底层转发器技术的用户进行性能调优。目前支持： **fluentd**。

19.1.1.1.42.1.1. 类型

- 对象

属性	类型	描述
fluentd	对象	

19.1.1.1.43. .spec.forwarder.fluentd

19.1.1.1.43.1. 描述

FluentdForwarderSpec 代表类型为 fluentd 的转发器的配置。

19.1.1.1.43.1.1. 类型

- 对象

属性	类型	描述
buffer	对象	
inFile	对象	

19.1.1.1.44. .spec.forwarder.fluentd.buffer

19.1.1.1.44.1. 描述

FluentdBufferSpec 代表 fluentd 缓冲参数的子集，用于调整所有 fluentd 输出的缓冲配置。它支持参数子集来配置缓冲区和队列大小、清空操作和重试清除。

有关常规参数，请参阅：https://docs.fluentd.org/configuration/buffer-section#buffering-parameters

有关 flush 参数，请参阅：https://docs.fluentd.org/configuration/buffer-section#flushing-parameters

有关重试参数请参考：https://docs.fluentd.org/configuration/buffer-section#retries-parameters

19.1.1.1.44.1.1. 类型

- 对象

属性	类型	描述
chunkLimitSize	字符串	**（可选）** ChunkLimitSize 代表每个块的最大大小。事件将是
flushInterval	字符串	**（可选）** FlushInterval 代表两个连续清除之间等待的时长
flushMode	字符串	**（可选）** FlushMode 代表要写入块的清除线程的模式。模式

属性	类型	描述
flushThreadCount	int	（可选） FlushThreadCount reprents 缓冲区使用的线程数量
overflowAction	字符串	（可选） OverflowAction 代表 fluentd 缓冲插件的操作
retryMaxInterval	字符串	（可选） RetryMaxInterval 代表 exponential backoff 的最大时间间隔
retryTimeout	字符串	（可选） RetryTimeout 代表在放弃前尝试重试的最长时间
retryType	字符串	（可选） RetryType 代表重试清除操作的类型。flush 操作可以
retryWait	字符串	（可选） RetryWait 代表两个连续重试刷新之间的持续时间
totalLimitSize	字符串	（可选） TotalLimitSize 代表每个 fluentd 允许的节点空间阈值

19.1.1.1.45. .spec.forwarder.fluentd.inFile

19.1.1.1.45.1. 描述

FluentdInFileSpec 代表 fluentd in-tail 插件参数的子集，用于调整所有 fluentd in-tail 输入的配置。

有关常规参数，请参阅： https://docs.fluentd.org/input/tail#parameters

19.1.1.1.45.1.1. 类型

- 对象

属性	类型	描述
readLinesLimit	int	（可选） ReadlinesLimit 代表要随每个 I/O 操作读取的行数

19.1.1.1.46. .spec.logStore

19.1.1.1.46.1. 描述

LogStoreSpec 包含有关日志存储方式的信息。

19.1.1.1.46.1.1. 类型

- 对象

属性	类型	描述
elasticsearch	对象	Elasticsearch 日志存储组件的规格
lokistack	对象	LokiStack 包含有关当 Type 设置为 LogStoreTypeLokiStack 时用于日志存储的信息。
retentionPolicy	对象	**（可选）** 保留策略定义了应删除它的索引的最长期限
type	字符串	要配置的日志存储的类型。Operator 目前支持使用 ElasticSearch

19.1.1.1.47. .spec.logStore.elasticsearch

19.1.1.1.47.1. 描述

19.1.1.1.47.1.1. 类型

- 对象

属性	类型	描述
nodeCount	int	为 Elasticsearch 部署的节点数量
nodeSelector	对象	定义 Pod 调度到哪些节点上。
proxy	对象	Elasticsearch Proxy 组件的规格
redundancyPolicy	字符串	**（可选）**
资源	对象	**（可选）** Elasticsearch 的资源要求
storage	对象	**（可选）** Elasticsearch 数据节点的存储规格
容限（tolerations）	数组	

19.1.1.1.48. .spec.logStore.elasticsearch.nodeSelector

19.1.1.1.48.1. 描述

19.1.1.1.48.1.1. 类型

- 对象

19.1.1.1.49. .spec.logStore.elasticsearch.proxy

19.1.1.1.49.1. 描述

19.1.1.1.49.1.1. 类型

- 对象

属性	类型	描述
资源	对象	

19.1.1.1.50. .spec.logStore.elasticsearch.proxy.resources

19.1.1.1.50.1. 描述

19.1.1.1.50.1.1. 类型

- 对象

属性	类型	描述
limits	对象	**（可选）** 限制描述了允许的最大计算资源量。
requests	对象	**（可选）** 请求描述了所需的最少计算资源。

19.1.1.1.51. .spec.logStore.elasticsearch.proxy.resources.limits

19.1.1.1.51.1. 描述

19.1.1.1.51.1.1. 类型

- 对象

19.1.1.1.52. .spec.logStore.elasticsearch.proxy.resources.requests

19.1.1.1.52.1. 描述

19.1.1.1.52.1.1. 类型

- 对象

19.1.1.1.53. .spec.logStore.elasticsearch.resources

19.1.1.1.53.1. 描述

19.1.1.1.53.1.1. 类型

- 对象

属性	类型	描述
limits	对象	**（可选）** 限制描述了允许的最大计算资源量。
requests	对象	**（可选）** 请求描述了所需的最少计算资源。

19.1.1.1.54. .spec.logStore.elasticsearch.resources.limits

19.1.1.1.54.1. 描述

19.1.1.1.54.1.1. 类型

- 对象

19.1.1.1.55. .spec.logStore.elasticsearch.resources.requests

19.1.1.1.55.1. 描述

19.1.1.1.55.1.1. 类型

- 对象

19.1.1.1.56. .spec.logStore.elasticsearch.storage

19.1.1.1.56.1. 描述

19.1.1.1.56.1.1. 类型

- 对象

属性	类型	描述
size	对象	要置备的节点的最大存储容量。
storageClassName	字符串	**（可选）** 用于创建节点的 PVC 的存储类的名称。

19.1.1.1.57. .spec.logStore.elasticsearch.storage.size

19.1.1.1.57.1. 描述

19.1.1.1.57.1.1. 类型

- 对象

属性	类型	描述
æ ¼å¼	字符串	更改格式将：有关 Reonicalize 的评论信息
d	对象	如果 d.Dec != nil, d 是 inf.Dec 表单的数量
i	int	如果 d.Dec == nil，i 是 int64 扩展形式的数量
s	字符串	s 是生成的这个数量的值，以避免重新计算

19.1.1.1.58. .spec.logStore.elasticsearch.storage.size.d

19.1.1.1.58.1. 描述

19.1.1.1.58.1.1. 类型

- 对象

属性	类型	描述
Dec	对象	

19.1.1.1.59. .spec.logStore.elasticsearch.storage.size.d.Dec

19.1.1.1.59.1. 描述

19.1.1.1.59.1.1. 类型

- 对象

属性	类型	描述
scale	int	
unscaled	对象	

19.1.1.1.60. .spec.logStore.elasticsearch.storage.size.d.Dec.unscaled

19.1.1.1.60.1. 描述

19.1.1.1.60.1.1. 类型

- 对象

属性	类型	描述
abs	Word	sign
neg	bool	

19.1.1.1.61. .spec.logStore.elasticsearch.storage.size.d.Dec.unscaled.abs

19.1.1.1.61.1. 描述

19.1.1.1.61.1.1. 类型

- Word

19.1.1.1.62. .spec.logStore.elasticsearch.storage.size.i

19.1.1.1.62.1. 描述

19.1.1.1.62.1.1. 类型

- int

属性	类型	描述
scale	int	
value	int	

19.1.1.1.63. .spec.logStore.elasticsearch.tolerations[]

19.1.1.1.63.1. 描述

19.1.1.1.63.1.1. 类型

- 数组

属性	类型	描述
effect	字符串	**（可选）** 效果表示要匹配的污点效果。空意味着匹配所有污点效果。

属性	类型	描述
key	字符串	**（可选）** key 是容限应用到的污点键。empty 表示与所有污点键匹配。
operator	字符串	**（可选）** Operator 代表键与值的关系。
tolerationSeconds	int	**（可选）** TolerationSeconds 代表容限的期间（必须是
value	字符串	**（可选）** 值是容限匹配的污点值。

19.1.1.1.64. .spec.logStore.elasticsearch.tolerations[].tolerationSeconds

19.1.1.1.64.1. 描述

19.1.1.1.64.1.1. 类型

- int

19.1.1.1.65. .spec.logStore.lokistack

19.1.1.1.65.1. 描述

LokiStackStoreSpec 用来设置 cluster-logging 以使用 LokiStack 作为日志存储。它指向同一命名空间中的现有 LokiStack。

19.1.1.1.65.1.1. 类型

- 对象

属性	类型	描述
name	字符串	LokiStack 资源的名称。

19.1.1.1.66. .spec.logStore.retentionPolicy

19.1.1.1.66.1. 描述

19.1.1.1.66.1.1. 类型

- 对象

属性	类型	描述
application	对象	
audit	对象	
Infra	对象	

19.1.1.1.67. .spec.logStore.retentionPolicy.application

19.1.1.1.67.1. 描述

19.1.1.1.67.1.1. 类型

- 对象

属性	类型	描述
diskThresholdPercent	int	**（可选）** 一个 ES 磁盘用量的阈值，当达到这个阈值时应该删除旧索引（如 75）
maxAge	字符串	**（可选）**
namespaceSpec	数组	**（可选）** 每个命名空间规格，用于删除超过给定最小年龄的文档
pruneNamespacesInterval	字符串	**（可选）** 运行新修剪命名空间作业的频率

19.1.1.1.68. .spec.logStore.retentionPolicy.application.namespaceSpec[]

19.1.1.1.68.1. 描述

19.1.1.1.68.1.1. 类型

- 数组

属性	类型	描述
minAge	字符串	**（可选）** 删除与这个 MinAge 旧的命名空间匹配的记录（例如 1d）
namespace	字符串	目标命名空间删除早于 MinAge 的日志（默认为 7d）

19.1.1.1.69. .spec.logStore.retentionPolicy.audit

19.1.1.1.69.1. 描述

19.1.1.1.69.1.1. 类型

- 对象

属性	类型	描述
diskThresholdPercent	int	**（可选）** 一个 ES 磁盘用量的阈值，当达到这个阈值时应该删除旧索引（如 75）
maxAge	字符串	**（可选）**
namespaceSpec	数组	**（可选）** 每个命名空间规格，用于删除超过给定最小年龄的文档
pruneNamespacesInterval	字符串	**（可选）** 运行新修剪命名空间作业的频率

19.1.1.1.70. .spec.logStore.retentionPolicy.audit.namespaceSpec[]

19.1.1.1.70.1. 描述

19.1.1.1.70.1.1. 类型

- 数组

属性	类型	描述
minAge	字符串	**（可选）** 删除与这个 MinAge 旧的命名空间匹配的记录（例如 1d）
namespace	字符串	目标命名空间删除早于 MinAge 的日志（默认为 7d）

19.1.1.1.71. .spec.logStore.retentionPolicy.infra

19.1.1.1.71.1. 描述

19.1.1.1.71.1.1. 类型

- 对象

属性	类型	描述
diskThresholdPercent	int	**（可选）** 一个 ES 磁盘用量的阈值，当达到这个阈值时应该删除旧索引（如 75）
maxAge	字符串	**（可选）**
namespaceSpec	数组	**（可选）** 每个命名空间规格，用于删除超过给定最小年龄的文档
pruneNamespacesInterval	字符串	**（可选）** 运行新修剪命名空间作业的频率

19.1.1.1.72. .spec.logStore.retentionPolicy.infra.namespaceSpec[]

19.1.1.1.72.1. 描述

19.1.1.1.72.1.1. 类型

- 数组

属性	类型	描述
minAge	字符串	**（可选）** 删除与这个 MinAge 旧的命名空间匹配的记录（例如 1d）
namespace	字符串	目标命名空间删除早于 MinAge 的日志（默认为 7d）

19.1.1.1.73. .spec.visualization

19.1.1.1.73.1. 描述

这是包含日志视觉化信息的结构 (Kibana)

19.1.1.1.73.1.1. 类型

- 对象

属性	类型	描述
kibana	对象	Kibana 视觉化组件的规格
type	字符串	要配置的可视化类型

19.1.1.1.74. .spec.visualization.kibana

19.1.1.1.74.1. 描述

19.1.1.1.74.1.1. 类型

- 对象

属性	类型	描述
nodeSelector	对象	定义 Pod 调度到哪些节点上。
proxy	对象	Kibana Proxy 组件的规格
replicas	int	为 Kibana 部署部署的实例数量
资源	对象	**（可选）** Kibana 的资源要求
容限（tolerations）	数组	

19.1.1.1.75. .spec.visualization.kibana.nodeSelector

19.1.1.1.75.1. 描述

19.1.1.1.75.1.1. 类型

- 对象

19.1.1.1.76. .spec.visualization.kibana.proxy

19.1.1.1.76.1. 描述

19.1.1.1.76.1.1. 类型

- 对象

属性	类型	描述
资源	对象	

19.1.1.1.77. .spec.visualization.kibana.proxy.resources

19.1.1.1.77.1. 描述

19.1.1.1.77.1.1. 类型

- 对象

属性	类型	描述
limits	对象	**（可选）** 限制描述了允许的最大计算资源量。
requests	对象	**（可选）** 请求描述了所需的最少计算资源。

19.1.1.1.78. .spec.visualization.kibana.proxy.resources.limits

19.1.1.1.78.1. 描述

19.1.1.1.78.1.1. 类型

- 对象

19.1.1.1.79. .spec.visualization.kibana.proxy.resources.requests

19.1.1.1.79.1. 描述

19.1.1.1.79.1.1. 类型

- 对象

19.1.1.1.80. .spec.visualization.kibana.replicas

19.1.1.1.80.1. 描述

19.1.1.1.80.1.1. 类型

- int

19.1.1.1.81. .spec.visualization.kibana.resources

19.1.1.1.81.1. 描述

19.1.1.1.81.1.1. 类型

- 对象

属性	类型	描述
limits	对象	**（可选）** 限制描述了允许的最大计算资源量。
requests	对象	**（可选）** 请求描述了所需的最少计算资源。

19.1.1.1.82. .spec.visualization.kibana.resources.limits

19.1.1.1.82.1. 描述

19.1.1.1.82.1.1. 类型

- 对象

19.1.1.1.83. .spec.visualization.kibana.resources.requests

19.1.1.1.83.1. 描述

19.1.1.1.83.1.1. 类型

- 对象

19.1.1.1.84. .spec.visualization.kibana.tolerations[]

19.1.1.1.84.1. 描述

19.1.1.1.84.1.1. 类型

- 数组

属性	类型	描述
effect	字符串	**（可选）** 效果表示要匹配的污点效果。空意味着匹配所有污点效果。
key	字符串	**（可选）** key 是容限应用到的污点键。empty 表示与所有污点键匹配。
operator	字符串	**（可选）** Operator 代表键与值的关系。
tolerationSeconds	int	**（可选）** TolerationSeconds 代表容限的期间（必须是
value	字符串	**（可选）** 值是容限匹配的污点值。

19.1.1.1.85. .spec.visualization.kibana.tolerations[].tolerationSeconds

19.1.1.1.85.1. 描述

19.1.1.1.85.1.1. 类型

- int

19.1.1.1.86. .status

19.1.1.1.86.1. 描述

ClusterLoggingStatus 定义 ClusterLogging 的观察状态

19.1.1.1.86.1.1. 类型

- 对象

属性	类型	描述
集合	对象	**(可选)**
conditions	对象	**(可选)**
curation	对象	**(可选)**
logStore	对象	**(可选)**
visualization	对象	**(可选)**

19.1.1.1.87. .status.collection

19.1.1.1.87.1. 描述

19.1.1.1.87.1.1. 类型

- 对象

属性	类型	描述
logs	对象	**(可选)**

19.1.1.1.88. .status.collection.logs

19.1.1.1.88.1. 描述

19.1.1.1.88.1.1. 类型

- 对象

属性	类型	描述
fluentdStatus	对象	**(可选)**

19.1.1.1.89. .status.collection.logs.fluentdStatus

19.1.1.1.89.1. 描述

19.1.1.1.89.1.1. 类型

- 对象

属性	类型	描述
clusterCondition	对象	**（可选）**
daemonSet	字符串	**（可选）**
节点	对象	**（可选）**
pods	字符串	**（可选）**

19.1.1.1.90. .status.collection.logs.fluentdStatus.clusterCondition

19.1.1.1.90.1. 描述

operator-sdk generate crds 不允许映射内容，必须使用命名类型。

19.1.1.1.90.1.1. 类型

- 对象

19.1.1.1.91. .status.collection.logs.fluentdStatus.nodes

19.1.1.1.91.1. 描述

19.1.1.1.91.1.1. 类型

- 对象

19.1.1.1.92. .status.conditions

19.1.1.1.92.1. 描述

19.1.1.1.92.1.1. 类型

- 对象

19.1.1.1.93. .status.curation

19.1.1.1.93.1. 描述

19.1.1.1.93.1.1. 类型

- 对象

属性	类型	描述
curatorStatus	数组	**(可选)**

19.1.1.1.94. .status.curation.curatorStatus[]

19.1.1.1.94.1. 描述

19.1.1.1.94.1.1. 类型

- 数组

属性	类型	描述
clusterCondition	对象	**(可选)**
cronJobs	字符串	**(可选)**
调度	字符串	**(可选)**
暂停	bool	**(可选)**

19.1.1.1.95. .status.curation.curatorStatus[].clusterCondition

19.1.1.1.95.1. 描述

operator-sdk generate crds 不允许映射内容，必须使用命名类型。

19.1.1.1.95.1.1. 类型

- 对象

19.1.1.1.96. .status.logStore

19.1.1.1.96.1. 描述

19.1.1.1.96.1.1. 类型

- 对象

属性	类型	描述
elasticsearchStatus	数组	**(可选)**

19.1.1.1.97. .status.logStore.elasticsearchStatus[]

19.1.1.1.97.1. 描述

19.1.1.1.97.1.1. 类型

- 数组

属性	类型	描述
cluster	对象	**（可选）**
clusterConditions	对象	**（可选）**
clusterHealth	字符串	**（可选）**
clusterName	字符串	**（可选）**
部署	数组	**（可选）**
nodeConditions	对象	**（可选）**
nodeCount	int	**（可选）**
pods	对象	**（可选）**
replicaSets	数组	**（可选）**
shardAllocationEnabled	字符串	**（可选）**
statefulSets	数组	**（可选）**

19.1.1.1.98. .status.logStore.elasticsearchStatus[].cluster

19.1.1.1.98.1. 描述

19.1.1.1.98.1.1. 类型

- 对象

属性	类型	描述
activePrimaryShards	int	Elasticsearch 集群的活跃主分片数量
activeShards	int	Elasticsearch 集群的活跃分片数量
initializingShards	int	Elasticsearch 集群的 Initializing Shards 数量

属性	类型	描述
numDataNodes	int	Elasticsearch 集群的数据节点数量
numNodes	int	Elasticsearch 集群的节点数量
pendingTasks	int	
relocatingShards	int	Elasticsearch 集群的重定位分片的数量
status	字符串	Elasticsearch 集群的当前状态
unassignedShards	int	Elasticsearch 集群的未分配的分片数量

19.1.1.1.99. .status.logStore.elasticsearchStatus[].clusterConditions

19.1.1.1.99.1. 描述

19.1.1.1.99.1.1. 类型

- 对象

19.1.1.1.100. .status.logStore.elasticsearchStatus[].deployments[]

19.1.1.1.100.1. 描述

19.1.1.1.100.1.1. 类型

- 数组

19.1.1.1.101. .status.logStore.elasticsearchStatus[].nodeConditions

19.1.1.1.101.1. 描述

19.1.1.1.101.1.1. 类型

- 对象

19.1.1.1.102. .status.logStore.elasticsearchStatus[].pods

19.1.1.1.102.1. 描述

19.1.1.1.102.1.1. 类型

- 对象

19.1.1.1.103. .status.logStore.elasticsearchStatus[].replicaSets[]

19.1.1.1.103.1. 描述

19.1.1.1.103.1.1. 类型

- 数组

19.1.1.1.104. .status.logStore.elasticsearchStatus[].statefulSets[]

19.1.1.1.104.1. 描述

19.1.1.1.104.1.1. 类型

- 数组

19.1.1.1.105. .status.visualization

19.1.1.1.105.1. 描述

19.1.1.1.105.1.1. 类型

- 对象

属性	类型	描述
kibanaStatus	数组	**（可选）**

19.1.1.1.106. .status.visualization.kibanaStatus[]

19.1.1.1.106.1. 描述

19.1.1.1.106.1.1. 类型

- 数组

属性	类型	描述
clusterCondition	对象	**（可选）**
部署	字符串	**（可选）**
pods	字符串	**（可选）** Visualization 组件的每个 Kibana Pod 的状态
replicaSets	数组	**（可选）**
replicas	int	**（可选）**

属性	类型	描述

19.1.1.1.107. .status.visualization.kibanaStatus[].clusterCondition

19.1.1.1.107.1. 描述

19.1.1.1.107.1.1. 类型

- 对象

19.1.1.1.108. .status.visualization.kibanaStatus[].replicaSets[]

19.1.1.1.108.1. 描述

19.1.1.1.108.1.1. 类型

- 数组

第 20 章 术语表

该术语表定义了日志记录文档中使用的常用术语。

注解

您可以使用注解将元数据附加到对象。

Red Hat OpenShift Logging Operator

Red Hat OpenShift Logging Operator 提供了一组 API，用于控制应用程序、基础架构和审计日志的集合和转发。

自定义资源 (CR)

CR 是 Kubernetes API 的扩展。要配置日志记录和日志转发，您可以自定义 **ClusterLogging** 和 **ClusterLogForwarder** 自定义资源。

事件路由器

事件路由器是一个 pod，它监视 OpenShift Container Platform 事件。它使用日志记录来收集日志。

Fluentd

Fluentd 是一个日志收集器，它驻留在每个 OpenShift Container Platform 节点上。它收集应用程序、基础架构和审计日志并将其转发到不同的输出。

垃圾回收

垃圾回收是清理集群资源的过程，如终止的容器和没有被任何正在运行的 pod 引用的镜像。

Elasticsearch

Elasticsearch 是一个分布式搜索和分析引擎。OpenShift Container Platform 使用 Elasticsearch 作为日志的默认日志存储。

OpenShift Elasticsearch Operator

OpenShift Elasticsearch Operator 用于在 OpenShift Container Platform 上运行 Elasticsearch 集群。OpenShift Elasticsearch Operator 为 Elasticsearch 集群操作提供自助服务，供日志记录使用。

索引

索引是一种数据结构技术，用于快速查找和访问数据。索引通过最大程度减少处理查询时所需的磁盘访问量来优化性能。

JSON 日志记录

Log Forwarding API 可让您将 JSON 日志解析到结构化对象，并将其转发到受管 Elasticsearch 的 logging 或 Log Forwarding API 支持的任何其他第三方系统。

Kibana

Kibana 是基于浏览器的控制台界面，可通过直方图、行图和 pie chart 查询、发现和视觉化您的 Elasticsearch 数据。

Kubernetes API 服务器

Kubernetes API 服务器验证并配置 API 对象的数据。

标签

标签是可用于组织和选择对象子集（如 pod）的键值对。

日志记录

通过日志记录，您可以聚合整个集群中的应用程序、基础架构和审计日志。您还可以将它们存储在默认日志存储中，将它们转发到第三方系统，并查询和视觉化存储在默认日志存储中的存储日志。

日志记录收集器

日志记录收集器从集群收集日志，对其进行格式化，并将它们转发到日志存储或第三方系统。

日志存储

日志存储用于存储聚合的日志。您可以使用内部日志存储或将日志转发到外部日志存储。

日志可视化工具

日志可视化工具是用户界面 (UI) 组件，可用于查看日志、图形、图表和其他指标等信息。

节点

节点是 OpenShift Container Platform 集群中的 worker 机器。节点是虚拟机 (VM) 或物理计算机。

Operator

Operator 是 OpenShift Container Platform 集群中打包、部署和管理 Kubernetes 应用程序的首选方法。Operator 将人类操作知识编码到一个软件程序中，易于打包并与客户共享。

Pod

pod 是 Kubernetes 中的最小逻辑单元。pod 由一个或多个容器组成，并在 worker 节点上运行。

基于角色的访问控制(RBAC)

RBAC 是一个关键安全控制，可确保集群用户和工作负载只能访问执行其角色所需的资源。

分片

Elasticsearch 将日志数据从 Fluentd 整理到数据存储或索引中，然后将每个索引划分为多个碎片，称为分片。

污点（taint）

污点可确保 pod 调度到适当的节点上。您可以在节点上应用一个或多个污点。

容限（toleration）

您可以将容限应用到 pod。容限 (toleration) 允许调度程序调度具有匹配污点的 pod。

Web 控制台

用于管理 OpenShift Container Platform 的用户界面(UI)。