OpenShift Container Platform 4.7

Jaeger

Jaeger 的安装、使用和发行注记信息

Last Updated: 2021-05-06

OpenShift Container Platform 4.7 Jaeger

Jaeger 的安装、使用和发行注记信息

法律通告

摘要

本文档提供了有关如何在 OpenShift Container Platform 中使用 Jaeger 的信息。

目录

第 1 章 JAEGER 发行注记

1.1. JAEGER 概述

作为服务所有者，您可以使用 Jaeger 来检测您的服务，以收集与服务架构相关的信息。Jaeger 是一个开源的分布式追踪平台，可用来对现代的、云原生的基于微服务的应用程序中组件间的交互进行监控、创建网络配置集并进行故障排除。

使用 Jaeger 可让您执行以下功能：

- 监控分布式事务
- 优化性能和延迟时间
- 执行根原因分析

Jaeger 基于厂商中立的 OpenTracing API 和工具。

1.2. 使开源包含更多

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

1.3. 获取支持

如果您在执行本文档所述的某个流程或 OpenShift Container Platform 时遇到问题，请访问 红帽客户门户网站。通过红帽客户门户网站：

- 搜索或者浏览红帽知识库，了解与红帽产品相关的文章和解决方案。
- 提交问题单给红帽支持。
- 访问其他产品文档。

为了识别集群中的问题，您可以在 Red Hat OpenShift Cluster Manager 中使用 Insights。Insights 提供了问题的详细信息，并在有可用的情况下，提供了如何解决问题的信息。

如果您对本文档有任何改进建议，或发现了任何错误，请访问 Bugzilla，针对 **OpenShift Container Platform** 产品的 **Documentation**组件提交 Bugzilla 报告。请提供具体详情，如章节名称和 OpenShift Container Platform 版本。

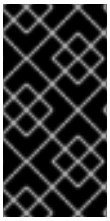
1.3.1. OpenShift Jaeger 1.20.0 的新功能

- 在这个 OpenShift Jaeger 版本中增加了对使用一个"外部" Elasticsearch 集群存储追踪数据的支持。外部是指不是由 OpenShift Elasticsearch Operator 安装和创建的 Elasticsearch 实例。
- 此发行版本添加了对 Jaeger Collector 和 Ingester 的自动扩展支持。

1.3.2. OpenShift Jaeger 1.17.7 的新功能

此 OpenShift Jaeger 发行版本解决了 CVE 报告的安全漏洞问题以及程序错误。

1.4. 技术预览



重要

技术预览功能不被红帽产品服务等级协议 (SLA) 支持，且可能在功能方面有缺陷。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的详情，请参阅 https://access.redhat.com/support/offerings/techpreview/。

1.4.1. OpenShift Jaeger 2.0.0 技术预览 1

安装 Jaeger Operator 时，您可以选择 Jaeger 的技术预览版本。这可让您访问客户端和基础架构，以便根据 OpenTelemetry 框架将追踪数据导出到 Jaeger。请注意：生产环境不支持这个版本。

OpenTelemetry 收集器允许开发人员使用不是针对固定厂商的 API，以避免锁定于特定厂商的情况，而是使用一个不断增大的可观察工具生态系统。

1.5. JAEGER 已知问题

Jaeger 中存在的限制：

- 不支持 Apache spark。
- IBM Z 和 IBM Power Systems 上不支持通过 AMQ/Kafka 进行 Jaeger 流。

Jaeger 中已知的问题：

- BZ-1918920 在更新后，Elasticsearch Pod 不会自动重启。作为临时解决方案，请手动重启 pod。
- TRACING-809 Jaeger Ingester 与 Kafka 2.3 不兼容。当存在两个或多个 Jaeger Ingester 实例时，它会不断在日志中生成重新平衡信息。这是由于在 Kafka 2.3 里存在一个程序错误，它已在 Kafka 2.3.1 中修复。如需更多信息，请参阅 Jaegertracing-1819。

1.6. JAEGER 修复的问题

- TRACING-1725 转入到 TRACING-1631。额外的程序漏洞修复，可确保当存在多个生产环境的 Jaeger 实例，它们使用相同的名称但在不同的命名空间中时，Elasticsearch 证书可以被正确协调。另请参阅 BZ-1918920。
- TRACING-1631 多 Jaeger 生产环境实例使用相同的名称但在不同命名空间中，因此会导致 Elasticsearch 证书问题。安装多个服务网格时，所有 Jaeger Elasticsearch 实例都有相同的 Elasticsearch secret 而不是单独的 secret，这导致 OpenShift Elasticsearch Operator 无法与所有 Elasticsearch 集群通信。
- 在使用 Istio sidecar 时，在 Agent 和 Collector 间的连接会出现 TRACING-1300 失败。对 Jaeger Operator 的更新默认启用了 Jaeger sidecar 代理和 Jaeger Collector 之间的 TLS 通信。
- TRACING-1208 访问 Jaeger UI 时的身份验证 "500 Internal Error" 错误。当尝试使用 OAuth 验证 UI 时，会得到 500 错误，因为 oauth-proxy sidecar 不信任安装时使用 **additionalTrustBundle** 定义的自定义 CA 捆绑包。
- TRACING-1166 目前无法在断开网络连接的环境中使用 Jaeger 流策略。当一个 Kafka 集群被置备时，它会产生一个错误： **Failed to pull image registry.redhat.io/amq7/amq-streams-kafka-24-**

rhel7@sha256:f9ceca004f1b7DCCB3b82d9a8027961f9fe4104e0ed69752c0bdd8078b4a1076。

第 2 章 JAEGER 架构

2.1. JAEGER 架构

每次用户在某个应用程序中执行一项操作时，一个请求都会在所在的系统上执行，而这个系统可能需要几十个不同服务的共同参与才可以做出相应的响应。Jaeger 提供了分布式追踪功能，可以在组成一个应用程序的多个微服务间记录请求的路径。

*分布式追踪*是用来将不同工作单元的信息关联起来的技术，通常是在不同进程或主机中执行的，以便理解分布式事务中的整个事件链。开发人员可以视觉化在大型微服务架构中调用的流程。它对理解序列化、并行性和延迟来源会很有价值。

Jaeger 在微服务的整个堆栈中记录了独立请求的执行过程，并将其显示为 trace。*trace*是系统的数据/执行路径。一个端到端的 trace 由一个或者多个 span 组成。

span 代表 Jaeger 中的逻辑工作单元，它包含操作名称、操作的开始时间和持续时间，以及可能的标签（tag）和日志信息。span 可能会被嵌套并排序以模拟因果关系。

2.1.1. Jaeger 概述

作为服务所有者，您可以使用 Jaeger 来检测您的服务，以收集与服务架构相关的信息。Jaeger 是一个开源的分布式追踪平台，可用来对现代的、云原生的基于微服务的应用程序中组件间的交互进行监控、创建网络配置集并进行故障排除。

使用 Jaeger 可让您执行以下功能：

- 监控分布式事务
- 优化性能和延迟时间
- 执行根原因分析

Jaeger 基于厂商中立的 OpenTracing API 和工具。

2.1.2. Jaeger 特性

Jaeger 追踪提供以下功能：

- 与 Kiali 集成 – 当正确配置时，您可以从 Kiali 控制台查看 Jaeger 数据。
- 高可伸缩性 – Jaeger 后端被设计为没有单一故障点，并可根据需要进行缩放。
- 分布式上下文发布 – 允许您通过不同的组件连接数据以创建完整的端到端的 trace。
- 与 Zipkin 的后向兼容性 – Jaeger 带有 API，让它可以作为 Zipkin 的直接替代项，但红帽在此发行版本中不支持 Zipkin 的兼容性。

2.1.3. Jaeger 架构

Jaeger 由几个组件组成，它们一起收集、存储和显示追踪数据。

- **Jaeger Client** （Tracer、Reporter、instrumented application, client libraries）- Jaeger client 是 OpenTracing API 的具体语言实现。它们可以用来为各种现有开源框架（如 Camel (Fuse)、Spring Boot (RHOAR)、MicroProfile (RHOAR/Thorntail)、Wilfly (EAP) 等提供分布式追踪工具。

- **Jaeger Agent** （Server Queue, Processor Workers）- Jaeger 代理是一个网络守护进程，它会监听通过 User Datagram Protocol (UDP) 发送的 span，并发送到收集程序。这个代理应被放置在要管理的应用程序的同一主机上。这通常是通过如 Kubernetes 等容器环境中的 sidecar 来实现的。

- **Jaeger Collector** （Que,Weue）- 与代理类似，该收集器可以接收 span，并将其放入内部队列以便进行处理。这允许收集器立即返回到客户端/代理，而不需要等待 span 进入存储。

- **Storage** (Data Store) - 收集器需要一个持久的存储后端。Jaeger 带有一个可插入的机制用于 span 存储。请注意：在这个发行本中，唯一支持的存储是 Elasticsearch。

- **Query** (Query Service) - Query 是一个从存储中检索 trace 的服务。

- **Ingester** (Ingester Service) - Jaeger 可以使用 Apache Kafka 作为收集器和实际后备存储 (Elasticsearch) 之间的缓冲。Ingester 是一个从 Kafka 读取数据并写入另一个存储后端 (Elasticsearch) 的服务。

- **Jaeger Console** – Jaeger 提供了一个用户界面，可让您可视觉地查看所分发的追踪数据。在搜索页面中，您可以查找 trace，并查看组成一个独立 trace 的 span 详情。

第 3 章 安装 JAEGER

3.1. 安装 JAEGER

您可以通过以下两种方式之一在 OpenShift Container Platform 上安装 Jaeger:

- 作为 Red Hat OpenShift Service Mesh 的一部分安装 Jaeger。Service Mesh 安装默认包含了 Jaeger。要将 Jaeger 作为 service mesh 的一部分安装，请按照 Red Hat Service Mesh 安装 中的说明进行。Jaeger 必须与服务网格安装在同一命名空间中，即 **ServiceMeshControlPlane** 和 Jaeger 资源必须位于同一个命名空间中。
- 如果您不想安装服务网格，可以使用 Jaeger Operator 单独安装 OpenShift Jaeger。要在没有 service mesh 的情况下安装 Jaeger，请按照以下说明操作。

3.1.1. 先决条件

在安装 OpenShift Jaeger 前，请查看安装所需的操作，确保满足以下条件：

- 您的红帽帐户中有活跃的 OpenShift Container Platform 订阅。如果您没有相关订阅，请联络您的销售代表以获得更多信息。
- 查看 OpenShift Container Platform 4.7 概述。
- 安装 OpenShift Container Platform 4.7。
  - 在 AWS 上安装 OpenShift Container Platform 4.7
  - 在用户置备的 AWS 上安装 OpenShift Container Platform 4.7
  - 在裸机上安装 OpenShift Container Platform 4.7
  - 在 vSphere 上安装 OpenShift Container Platform 4.7
- 安装与 OpenShift Container Platform 版本匹配的 OpenShift Container Platform 命令行工具（**oc** 客户端工具），并将其添加到执行路径中。
- 具有 **cluster-admin** 角色的帐户。

3.1.2. Jaeger 安装概述

安装 OpenShift Jaeger 的步骤如下：

- 查看文档并确定您的部署策略。
- 如果您的部署策略需要持久性存储，请通过 OperatorHub 安装 OpenShift Elasticsearch Operator。
- 通过 OperatorHub 安装 Jaeger Operator。
- 修改 Jaeger YAML 文件，以支持您的部署策略。
- 将一个或多个 Jaeger 实例部署到 OpenShift Container Platform 环境。

3.1.3. 安装 OpenShift Elasticsearch Operator

默认 Jaeger 部署使用内存存储，这可以使那些评估 Jaeger 、演示或者在测试环境中使用 Jaeger 的用户快速地进行安装。如果要在生产环境中使用 Jaeger，则必须安装并配置持久性存储选项，即 Elasticsearch。

先决条件

- 访问 OpenShift Container Platform Web 控制台。
- 具有 **cluster-admin** 角色的帐户。

警告

不要安装 Operators 的 Community 版本。不支持社区 Operator。

注意

如果您已经安装了 OpenShift Elasticsearch Operator 作为 OpenShift Logging 的一部分，则不需要再次安装 OpenShift Elasticsearch Operator。Jaeger Operator 将使用已安装的 OpenShift Elasticsearch Operator 创建 Elasticsearch 实例。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。
2. 导航至 **Operators → OperatorHub**。
3. 在过滤器框中键入 **Elasticsearch** 以找到 OpenShift Elasticsearch Operator。
4. 点由红帽提供的 **OpenShift Elasticsearch Operator** 来显示有关 Operator 的信息。
5. 点击 **Install**。
6. 在 **Install Operator** 页面中，在 **Installation Mode** 下选择 **All namespaces on the cluster(default)**。这使 Operator 可供集群中的所有项目使用。
7. 在 **Installed Namespaces** 下，从菜单中选择 **openshift-operators-redhat**。

注意

Elasticsearch 安装需要 OpenShift Elasticsearch Operator 的 **openshift-operators-redhat** 命名空间。其他 OpenShift Jaeger operator 安装在 **openshift-operators** 命名空间中。

8. 选择与 OpenShift Container Platform 安装匹配的**更新频道**。例如，如果您要在 OpenShift Container Platform 版本 4.6 上安装，请选择 4.6 更新频道。

注意

对于 OpenShift Container Platform 版本 4.7，选择 4.6 更新频道。

9. 选择 **Automatic** 批准策略。

注意

手动批准策略需要拥有适当凭证的用户批准 Operator 的安装和订阅过程。

10. 点击 **Install**。

11. 在 **Installed Operators** 页面中，选择 **openshift-operators-redhat** 项目。等待 OpenShift Elasticsearch Operator 的状态显示为 "InstallSucceeded" 后再继续进行操作。

3.1.4. 安装 Jaeger Operator

要安装 Jaeger，您需要使用 OperatorHub 来安装 Jaeger Operator。

默认情况下，Operator 安装在 **openshift-operators** 项目中。

先决条件

- 访问 OpenShift Container Platform Web 控制台。
- 具有 **cluster-admin** 角色的帐户。
- 如果需要持久性存储，则必须在安装 Jaeger Operator 前安装 OpenShift Elasticsearch Operator。

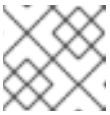
警告

不要安装 Operators 的 Community 版本。不支持社区 Operator。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。
2. 进入 **Operators → OperatorHub**。
3. 在过滤器框中键入 **Jaeger** 来找到 Jaeger Operator。
4. 点由红帽提供的 **Jaeger Operator** 来显示有关 Operator 的信息。
5. 点击 **Install**。
6. 在 **Install Operator** 页中，选择 **stable** Update Channel。这可在发布新版本时自动更新 Jaeger。如果您选择维护频道，例如 **1.17-stable**，则会在支持周期内接收程序错误修复和安全补丁。
7. 选择 **All namespaces on the cluster（默认）**。这会在默认的 **openshift-operators** 项目中安装 Operator，并使其可以被集群中的所有项目使用。
    - 选择一个批准策略您可以选择 **Automatic** 或 **Manual** 更新。如果选择自动更新某个已安装的 Operator，则当相应 Operator 有可用的新版本时，Operator Lifecycle Manager（OLM）将

自动升级该 Operator 的运行实例，而无需人为干预。如果选择手动更新，则当有新版 Operator 可用时，OLM 会创建更新请求。作为集群管理员，您必须手动批准该更新请求，才可将 Operator 更新至新版本。



注意

手动批准策略需要拥有适当凭证的用户批准 Operator 的安装和订阅过程。

8. 点击 **Install**。

9. 在 **Subscription Overview** 页面中，选择 **openshift-operators** 项目。等待 Jaeger Operator 的状态显示为 "InstallSucceeded" 后再继续进行操作。。

3.2. 配置和部署 JAEGER

Jaeger Operator 使用自定义资源定义（CRD）文件定义创建和部署 Jaeger 资源时要使用的架构和配置设置。您可以安装默认配置或修改该文件以更好地满足您的业务要求。

Jaeger 具有预定义的部署策略。您可以在自定义资源文件中指定一个部署策略。创建 Jaeger 实例时，Operator 会使用此配置文件创建部署所需的对象。

Jaeger 自定义资源文件显示部署策略

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: simple-prod
spec:
  strategy: production (1)
```

(1) Jaeger Operator 目前支持以下部署策略：

- **allInOne** （默认）- 这个策略主要用于开发、测试和演示目的，它不用于生产环境。主要的后端组件 Agent、Collector 和 Query 服务都打包成单一可执行文件，（默认）配置为使用内存存储。

注意

内存存储不是持久性的，这意味着如果 Jaeger 实例关闭、重启或被替换，您的 trace 数据将会丢失。此外，内存存储无法扩展，因为每个 Pod 都有自己的内存。对于持久性存储，您必须使用 **production** 或 **streaming** 策略，这些策略使用 Elasticsearch 作为默认存储。

- **production** - production 策略主要用于生产环境，在生产环境中，对 trace 数据进行长期存储非常重要，同时需要更容易扩展和高度可用的构架。因此，每个后端组件都将单独部署。Agent 可以作为检测应用程序上的 sidecar 注入。Query 和 Collector 服务被配置为使用一个受支持的存储类型 - 当前为 Elasticsearch。可以根据性能和恢复能力的需要提供每个组件的多个实例。

- **streaming** - streaming 策略旨在提供在 Collector 和后端存储 (Elasticsearch) 之间有效发挥作用的流传输功能，以此增强 production 策略。这样做的好处是在高负载情况下降低后端存储压力，并允许其他 trace 后处理功能直接从流传输平台 (AMQ Streams/ Kafka) 中利用实时 span 数据。

注意

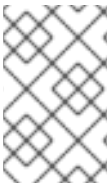
streaming 策略需要额外的 AMQ Streams 订阅。

注意

有两种方法可用来安装和使用 Jaeger，作为服务网格的一部分或作为独立组件。如果您已将 Jaeger 作为 Red Hat OpenShift Service Mesh 的一部分安装，您可以将 Jaeger 配置为 ServiceMeshControlPlane 的一部分，或者在 SMCP 中引用 Jaeger 配置。

3.2.1. 从 Web 控制台部署默认 Jaeger 策略

自定义资源定义 (CRD) 定义部署 Jaeger 实例时使用的配置。Jaeger 的默认 CR 名为 **jaeger-all-in-one-inmemory**，它配置为使用最少资源，以确保您可以在默认的 OpenShift Container Platform 安装中成功安装它。您可以使用此默认配置创建使用 **AllInOne** 部署策略的 Jaeger 实例，或者您可以定义自己的自定义资源文件。

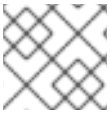


注意

内存存储不是持久性的，这意味着如果 Jaeger Pod 关闭、重启或被替换，您的 trace 数据将会丢失。对于持久性存储，您必须使用 **production** 或 **streaming** 策略，这些策略使用 Elasticsearch 作为默认存储。

先决条件

- 必须安装 Jaeger Operator。
- 查看有关如何自定义 Jaeger 安装的说明。
- 具有 **cluster-admin** 角色的帐户。



注意

目前 IBM Z 上不支持 Jaeger 流。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。
2. 创建一个新项目，如 **jaeger-system**。
   a. 浏览至 **Home → Project**。
   b. 点击 **Create Project**。
   c. 在 **Name** 字段中输入 **jaeger-system**。
   d. 点击 **Create**。
3. 导航到 **Operators → Installed Operators**。
4. 如果需要，请在 Project 菜单中选择 **jaeger-system**。您可能需要等待一些时间，让 Operator 复制到新项目中。

5. 点击 OpenShift Jaeger Operator。在 **Overview** 选项卡上的 **Provided APIs** 下，Operator 提供了单个链接。
6. 在 **Jaeger** 下点击 **Create Instance**。
7. 在 **Create Jaeger** 页面上，要使用默认值进行安装，请点击 **Create** 来创建 Jaeger 实例。
8. 在 **Jaegers** 页面上，点击 Jaeger 实例的名称，如 **jaeger-all-in-one-inmemory**。
9. 在 **Jaeger Details** 页面上，点击 **Resources** 选项卡。等到 Pod 的状态变为"Running"再继续操作。

3.2.1.1. 通过 CLI 部署默认 Jaeger

按照以下步骤，通过命令行创建 Jaeger 实例。

先决条件

- 已安装并验证 OpenShift Jaeger Operator。
- 访问与 OpenShift Container Platform 版本匹配的 OpenShift CLI（**oc**）。
- 具有 **cluster-admin** 角色的帐户。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform CLI。

```
$ oc login https://{HOSTNAME}:8443
```

2. 创建一个名为 **jaeger-system** 的新项目。

```
$ oc new-project jaeger-system
```

3. 创建一个名为 **jaeger.yaml** 的自定义资源文件，其中包含以下文本：

示例 Jaeger-all-in-one.yaml

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: jaeger-all-in-one-inmemory
```

4. 运行以下命令来部署 Jaeger：

```
$ oc create -n jaeger-system -f jaeger.yaml
```

5. 在安装过程中运行以下命令来监控 pod 的进度：

```
$ oc get pods -n jaeger-system -w
```

安装过程完成后，您应该看到类似如下的输出：

```
NAME                                  READY   STATUS    RESTARTS   AGE
jaeger-all-in-one-inmemory-cdff7897b-qhfdx   2/2     Running   0          24s
```

3.2.2. 从 Web 控制台部署 Jaeger production 策略

production 部署策略主要用于生产环境，在生产环境中，对 trace 数据进行长期存储非常重要，同时需要更容易扩展和高度可用的构架。

先决条件

- 必须安装 OpenShift Elasticsearch Operator。
- 必须安装 Jaeger Operator。
- 查看有关如何自定义 Jaeger 安装的说明。
- 具有 **cluster-admin** 角色的帐户。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。
2. 创建一个新项目，如 **jaeger-system**。
   a. 浏览至 **Home → Project**。
   b. 点击 **Create Project**。
   c. 在 **Name** 字段中输入 **jaeger-system**。
   d. 点击 **Create**。
3. 导航到 **Operators → Installed Operators**。
4. 如果需要，请在 Project 菜单中选择 **jaeger-system**。您可能需要等待一些时间，让 Operator 复制到新项目中。
5. 点 Jaeger Operator在 **Overview** 选项卡上的 **Provided APIs** 下，Operator 提供了单个链接。
6. 在 **Jaeger** 下点击 **Create Instance**。
7. 在 **Create Jaeger** 页面上，将默认的 **all-in-one** yaml 文本替换为您的生产环境的 YAML 配置，例如：

使用 Elasticsearch 的示例 jaeger-production.yaml 文件

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: jaeger-production
  namespace:
spec:
  strategy: production
  ingress:
    security: oauth-proxy
```

```
storage:
  type: elasticsearch
  elasticsearch:
    nodeCount: 3
    redundancyPolicy: SingleRedundancy
  esIndexCleaner:
    enabled: true
    numberOfDays: 7
    schedule: 55 23 * * *
  esRollover:
    schedule: '*/30 * * * *'
```

8. 点击 **Create** 创建 Jaeger 实例。
9. 在 **Jaegers** 页面上，点击 Jaeger 实例的名称，如 **jaeger-prod-elasticsearch**。
10. 在 **Jaeger Details** 页面上，点击 **Resources** 选项卡。等到所有 Pod 的状态变为"Running"再继续操作。

3.2.2.1. 通过 CLI 部署 Jaeger 生产环境

按照以下步骤，通过命令行创建 Jaeger 实例。

先决条件

- 已安装并验证 OpenShift Jaeger Operator。
- 访问 OpenShift CLI（**oc**）。
- 具有 **cluster-admin** 角色的帐户。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform CLI。

```
$ oc login https://{HOSTNAME}:8443
```

2. 创建一个名为 **jaeger-system** 的新项目。

```
$ oc new-project jaeger-system
```

3. 创建一个名为 **jaeger-production.yaml** 的自定义资源文件，其中包含上一步中的示例文件文本。
4. 运行以下命令来部署 Jaeger：

```
$ oc create -n jaeger-system -f jaeger-production.yaml
```

5. 在安装过程中运行以下命令来监控 pod 的进度：

```
$ oc get pods -n jaeger-system -w
```

安装过程完成后，您应该看到类似如下的输出：

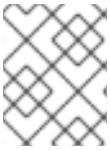
```
NAME                                  READY   STATUS    RESTARTS   AGE
```

```
elasticsearch-cdm-jaegersystemjaegerproduction-1-6676cf568gwhlw   2/2     Running   0
10m
elasticsearch-cdm-jaegersystemjaegerproduction-2-bcd4c8bf5l6g6w   2/2     Running   0
10m
elasticsearch-cdm-jaegersystemjaegerproduction-3-844d6d9694hhst   2/2     Running   0
10m
jaeger-production-collector-94cd847d-jwjlj                        1/1     Running   3          8m32s
jaeger-production-query-5cbfbd499d-tv8zf                          3/3     Running   3          8m32s
```

3.2.3. 从 Web 控制台部署 Jaeger streaming 策略

streaming 部署策略主要用于生产环境，在生产环境中，对 trace 数据进行长期存储非常重要，同时需要更容易扩展和高度可用的构架。

streaming 策略提供了在 Collector 和存储 (Elasticsearch) 之间发挥作用的流传输功能。这在高负载情况下降低了存储压力，并允许其他 trace 后处理功能直接从流传输平台 (Kafka) 中利用实时 span 数据。



注意

streaming 策略需要额外的 AMQ Streams 订阅。如果您没有 AMQ Streams 订阅，请联络您的销售代表以了解更多信息。

先决条件

- 必须安装 AMQ Streams Operator。如果使用 1.4.0 或更高版本，您可以使用自助置备。否则，您需要创建 Kafka 实例。
- 必须安装 Jaeger Operator。
- 查看有关如何自定义 Jaeger 安装的说明。
- 具有 **cluster-admin** 角色的帐户。

注意

目前 IBM Z 上不支持 Jaeger 流。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。
2. 创建一个新项目，如 **jaeger-system**。
   a. 浏览至 **Home → Project**。
   b. 点击 **Create Project**。
   c. 在 **Name** 字段中输入 **jaeger-system**。
   d. 点击 **Create**。
3. 导航到 **Operators → Installed Operators**。
4. 如果需要，请在 Project 菜单中选择 **jaeger-system**。您可能需要等待一些时间，让 Operator 复制到新项目中。

5. 点 Jaeger Operator在 **Overview** 选项卡上的 **Provided APIs** 下，Operator 提供了单个链接。
6. 在 **Jaeger** 下点击 **Create Instance**。
7. 在 **Create Jaeger** 页面上，将默认的 **all-in-one** yaml 文本替换为您的流传输 YAML 配置，例如：

示例 Jaeger-streaming.yaml 文件

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: jaeger-streaming
spec:
  strategy: streaming
  collector:
    options:
      kafka:
        producer:
          topic: jaeger-spans
          #Note: If brokers are not defined,AMQStreams 1.4.0+ will self-provision Kafka.
          brokers: my-cluster-kafka-brokers.kafka:9092
  storage:
    type: elasticsearch
  ingester:
    options:
      kafka:
        consumer:
          topic: jaeger-spans
          brokers: my-cluster-kafka-brokers.kafka:9092
```

1. 点击 **Create** 创建 Jaeger 实例。
2. 在 **Jaegers** 页面上，点击 Jaeger 实例的名称，如 **jaeger-streaming**。
3. 在 **Jaeger Details** 页面上，点击 **Resources** 选项卡。等到所有 Pod 的状态变为“Running”再继续操作。

3.2.3.1. 通过 CLI 部署 Jaeger 流

按照以下步骤，通过命令行创建 Jaeger 实例。

先决条件

- 已安装并验证 OpenShift Jaeger Operator。
- 访问 OpenShift CLI（**oc**）。
- 具有 **cluster-admin** 角色的帐户。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform CLI。

```
$ oc login https://{HOSTNAME}:8443
```

2. 创建一个名为 **jaeger-system** 的新项目。

```
$ oc new-project jaeger-system
```

3. 创建一个名为 **jaeger-streaming.yaml** 的自定义资源文件，其中包含上一步中的示例文件文本。

4. 运行以下命令来部署 Jaeger：

```
$ oc create -n jaeger-system -f jaeger-streaming.yaml
```

5. 在安装过程中运行以下命令来监控 pod 的进度：

```
$ oc get pods -n jaeger-system -w
```

安装过程完成后，您应该看到类似如下的输出：

```
NAME                                                  READY   STATUS    RESTARTS   AGE
elasticsearch-cdm-jaegersystemjaegerstreaming-1-697b66d6fcztcnn   2/2     Running   0          5m40s
elasticsearch-cdm-jaegersystemjaegerstreaming-2-5f4b95c78b9gckz   2/2     Running   0          5m37s
elasticsearch-cdm-jaegersystemjaegerstreaming-3-7b6d964576nnz97   2/2     Running   0          5m5s
jaeger-streaming-collector-6f6db7f99f-rtcfm                       1/1     Running   0          80s
jaeger-streaming-entity-operator-6b6d67cc99-4lm9q                 3/3     Running   2          2m18s
jaeger-streaming-ingester-7d479847f8-5h8kc                        1/1     Running   0          80s
jaeger-streaming-kafka-0                                          2/2     Running   0          3m1s
jaeger-streaming-query-65bf5bb854-ncnc7                           3/3     Running   0          80s
jaeger-streaming-zookeeper-0                                      2/2     Running   0          3m39s
```

3.2.4. 自定义 Jaeger 部署

3.2.4.1. 部署最佳实践

- Jaeger 实例名称必须是唯一的。如果您要有多个 Jaeger 实例，并且正在使用 sidecar 注入的 Jaeger 代理，则 Jaeger 实例应具有唯一的名称，注入注解应明确指定应报告追踪数据的 Jaeger 实例名称。
- 如果您有多租户实现，且租户由命名空间分开，将 Jaeger 实例部署到每个租户命名空间中。
- 如果您要作为 Red Hat OpenShift Service Mesh 的一部分安装 Jaeger，则必须在与 **ServiceMeshControlPlane** 资源相同的命名空间中安装 Jaeger 资源。

3.2.4.2. Jaeger 默认配置选项

Jaeger 自定义资源 (CR) 定义创建 Jaeger 资源时要使用的架构和设置。您可以根据您的业务需求修改这些参数以自定义 Jaeger 实现。

Jaeger 通用 YAML 示例

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
```

```
metadata:
  name: name
spec:
  strategy: <deployment_strategy>
  allInOne:
    options: {}
    resources: {}
  agent:
    options: {}
    resources: {}
  collector:
    options: {}
    resources: {}
  sampling:
    options: {}
  storage:
    type:
    options: {}
  query:
    options: {}
    resources: {}
  ingester:
    options: {}
    resources: {}
  options: {}
```

表 3.1. Jaeger 参数

参数	描述	值	默认值
apiVersion：	创建对象时使用的应用程序接口版本。	**jaegertracing.io/v1**	**jaegertracing.io/v1**
kind：	定义要创建的 Kubernetes 对象的种类。	**jaeger**	
metadata：	有助于唯一标识对象的数据，包括 **name** 字符串、**UID** 和可选 **namespace**。		OpenShift 会自动生成 **UID** 并使用创建对象的项目名称完成 **namespace**。
name：	对象的名称。	Jaeger 实例的名称。	**jaeger-all-in-one-inmemory**

参数	描述	值	默认值
spec：	要创建的对象的规格。	包含 Jaeger 实例的所有配置参数。当需要一个通用定义（用于所有 Jaeger 组件）时，会在 spec 节点下定义它。当该定义与单个组件相关时，会将它放置在 spec/<component> 节点下。	N/A
strategy：	Jaeger 部署策略	**allInOne**、**production** 或 **streaming**	**allInOne**
allInOne：	由于 allInOne 镜像在单个 Pod 中部署了 agent、collector、query、ingester 和 Jaeger UI，因此该部署的配置应该在 allInOne 参数下嵌套组件配置。		
agent：	定义 Jaeger 代理的配置选项。		
collector：	定义 Jaeger Collector 的配置选项。		
sampling：	定义用于追踪的抽样策略的配置选项。		
storage：	定义存储的配置选项。所有与存储相关的选项都应放在 **storage** 下，而不是放在 **allInOne** 或者其他组件选项下。		
query：	定义 Query 服务的配置选项。		
ingester：	定义 Ingester 服务的配置选项。		

以下示例 YAML 是使用默认设置创建 Jaeger 实例的最低要求。

jaeger-all-in-one.yaml 最低要求示例

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
```

```
name: jaeger-all-in-one-inmemory
```

3.2.4.3. Jaeger Collector 配置选项

Jaeger Collector 组件负责接收 tracer 捕获的 span，在使用 **production** 策略时将其写入持久性存储 (Elasticsearch)，在使用 **streaming** 策略时将其写入 AMQ Streams。

收集器是无状态的，因此许多 Jaeger Collector 实例可以并行运行。除了 Elasticsearch 集群的位置，收集器几乎不需要任何配置。

表 3.2. Operator 用来定义 Jaeger Collector 的参数

参数	描述	值
`collector:` `  replicas:`	指定要创建的 Collector 副本数。	整数，如 **5**。

表 3.3. 传递给 Collector 的 Jaeger 参数

参数	描述	值

`spec:`
`  collector:`
`  options: {}`
	定义 Jaeger Collector 的配置选项。	

`options:`
`  collector:`
`  num-workers:`
	从队列中拉取的 worker 数量。	整数，如 **50**。

`options:`
`  collector:`
`  queue-size:`
	Collector 队列的大小。	整数，如 **2000**。

`options:`
`  kafka:`
`  producer:`
`  topic: jaeger-spans`
	topic 参数标识收集器用来生成消息的 Kafka 配置以及要使用消息的 ingester。	producer 的标签

`kafka:`
`  producer:`
`  brokers: my-cluster-kafka-brokers.kafka:9092`
	标识 Collector 用来生成消息的 Kafka 配置。如果没有指定代理，并且安装了 AMQ Streams 1.4.0+，Jaeger 将自助置备 Kafka。	

参数	描述	值
log-level:	收集器的日志记录级别。	**trace**、**debug**、**info**、**warning**、**error**、**fatal**、**panic**。
maxReplicas:	指定在自动扩展 Collector 时创建的最大副本数。	整数，如 **100**。
num-workers:	从队列中拉取的 worker 数量。	整数，如 **50**。
queue-size:	Collector 队列的大小。	整数，如 **2000**。
replicas:	指定要创建的 Collector 副本数。	整数，如 **5**。

3.2.4.3.1. 配置 Collector 进行自动扩展

注意

自动扩展只支持 Jaeger 1.20 或更高版本。

您可以将 Collector 配置为自动扩展，Collector 将根据 CPU 和/或内存的使用情况进行扩展或缩减。将 Collector 配置为自动扩展可帮助您确保在负载增加时扩展 Jaeger 环境，并在需要较少资源时缩减资源以节约成本。您可以通过将 **autoscale** 参数设置为 **true** 来配置自动扩展，并为您希望 Collector 的 pod 使用的资源指定一个 **.spec.collector.maxReplicas** 的值。如果没有为 **.spec.collector.maxReplicas** 设置值，Operator 将把它设置为 **100**。

默认情况下，当没有为 **.spec.collector.replicas** 提供值时，Jaeger Operator 会为 Collector 创建 Horizontal Pod Autoscaler（HPA）配置。如需有关 HPA 的更多信息，请参阅 Kubernetes 文档。

以下是一个自动扩展配置示例，设置 Collector 的限制以及最大副本数：

Collector 自动扩展示例

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: simple-prod
spec:
  strategy: production
  collector:
    maxReplicas: 5
    resources:
      limits:
        cpu: 100m
        memory: 128Mi
```

3.2.4.4. Jaeger 抽样配置选项

Operator 可用于定义抽样策略，以提供给已经被配置为使用远程 sampler 的 tracer。

虽然生成了所有 trace，但只有几个会被抽样。对某个 trace 进行抽样会标记该 trace 用于进一步处理和存储。

注意

如果某个 trace 是由 Istio 代理启动的，则不相关，因为抽样决定是在那里做出的。只有在应用程序使用 Jaeger tracer 启动 trace 时，Jaeger 抽样决定才相关。

当服务收到不包含 trace 上下文的请求时，Jaeger tracer 会启动一个新的 trace，为它分配一个随机的 trace ID,并根据当前安装的抽样策略做出抽样决定。抽样决定被传播到 trace 中的所有后续请求，这样其他服务便不会再做出抽样决定。

Jaeger 库支持以下 sampler：

- **Probabilistic（概率）** - sampler 做出一个随机抽样决定，其抽样的概率等于 **sampling.param** 属性的值。例如：如果 sample.param=0.1，则会对大约十分之一的 trace 进行抽样。
- **Rate Limiting（速率限制）** - sampler 使用泄漏存储桶速率限制器来确保 trace 使用某种恒定速率进行抽样。例如，当 sample.param=2.0 时，它将对请求进行抽样，速率是每秒 2 个 trace。

表 3.4. Jaeger 抽样选项

参数	描述	值	默认值

`spec:`
`sampling:`
`options: {}`
`default_strategy:`
 	
`service_strategy:`
	定义用于追踪的抽样策略的配置选项。		如果没有提供配置，则收集器会返回默认的概率抽样策略，所有服务都可能为 0.001（0.1%）。

`default_strategy:`
`type:`
`service_strategy:`
`type:`
	要使用的抽样策略。（请参阅上述描述。）	有效值是 **probabilistic** 和 **ratelimiting**。	**probabilistic**

`default_strategy:`
`param:`
`service_strategy:`
`param:`
	所选抽样策略的参数。	小数值和整数值 (0, .1, 1, 10)	1

这个示例定义了一种概率性的默认抽样策略，trace 实例被抽样的几率为 50%。

概率抽样示例

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: with-sampling
spec:
  sampling:
    options:
      default_strategy:
        type: probabilistic
        param: 0.5
      service_strategies:
        - service: alpha
          type: probabilistic
          param: 0.8
          operation_strategies:
            - operation: op1
              type: probabilistic
              param: 0.2
            - operation: op2
              type: probabilistic
              param: 0.4
        - service: beta
          type: ratelimiting
          param: 5
```

如果没有用户提供的配置，Jaeger 将使用以下设置。

默认抽样

```
spec:
  sampling:
    options:
      default_strategy:
        type: probabilistic
        param: 1
```

3.2.4.5. Jaeger 存储配置选项

您可在 **spec:storage** 下为 Collector、Ingester 和 Query 服务配置存储。可以根据性能和恢复能力的需要提供每个组件的多个实例。

表 3.5. Operator 用来定义 Jaeger 存储的一般存储参数

参数	描述	值	默认值

`spec:`
`  storage:`
`  type:`
	要在部署中使用的存储类型。	**memory** 或 **elasticsearch**。内存存储仅适用于开发、测试、演示和验证概念环境，因在关闭 pod 时，数据不会保留。对于生产环境，Jaeger 支持 Elasticsearch 的持久性存储。	**memory**

参数	描述	值	默认值
storage: secretname:	secret 的名称，如 **jaeger-secret**。		N/A
storage: options: {}	定义存储的配置选项。		

表 3.6. Elasticsearch 索引清理参数

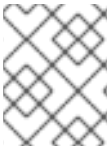
参数	描述	值	默认值
storage: esIndexCleaner: enabled:	当使用 Elasticsearch 存储时，默认会创建一个任务来清理索引中的旧 trace。这个参数用于启用或禁用索引清理任务。	**true**/ **false**	**true**
storage: esIndexCleaner: numberOfDays:	删除索引前等待的天数。	整数值	**7**
storage: esIndexCleaner: schedule:	为 Elasticsearch 索引的清理频率定义调度。	Cron 表达式	"55 23 * * *"

3.2.4.5.1. 自动置备 Elasticsearch 实例

当 **storage:type** 设为 **elasticsearch** 但没有为 **spec:storage:options:es:server-urls** 设置值时，Jaeger Operator 会使用 OpenShift Elasticsearch Operator 根据自定义资源文件的 **storage** 部分中提供的配置创建一个 Elasticsearch 集群。

限制

- 每个命名空间只能有一个 Elasticsearch。
- 您无法将 OpenShift Jaeger 日志记录 Elasticsearch 实例和 Jaeger 共享，或重复用于 Jaeger。Elasticsearch 集群意在专用于单个 Jaeger 实例。



注意

如果您已经安装了 Elasticsearch 作为 OpenShift 日志记录的一部分，Jaeger Operator 可以使用已安装的 OpenShift Elasticsearch Operator 来置备存储。

以下配置参数用于一个*自置备的* Elasticsearch 实例，这是由 Jaeger Operator 使用 OpenShift Elasticsearch Operator 创建的实例。在配置文件中，您可以在 **spec:storage:elasticsearch** 下为自助置备 Elasticsearch 指定配置选项。

表 3.7. Elasticsearch 资源配置参数

参数	描述	值	默认值
elasticsearch: nodeCount:	Elasticsearch 节点数量。对于高可用性，需要至少 3 个节点。不要只使用 2 个节点，因为可能会出现“脑裂”问题。	整数值。例如，概念验证 = 1，最小部署 = 3	3
elasticsearch: resources: requests: cpu:	根据您的环境配置，请求的 CPU 数量。	以 core 或者 millicores 指定（例如: 200m, 0.5, 1）。例如，概念证明 = 500m，最小部署 =1	1
elasticsearch: resources: requests: memory:	根据您的环境配置，可用于请求的内存。	以字节为单位指定（例如: 200Ki, 50Mi, 5Gi）。例如，概念证明 = 1Gi，最小部署 = 16Gi*	16Gi
elasticsearch: resources: limits: cpu:	根据您的环境配置，CPU 数量的限值。	以 core 或者 millicores 指定（例如: 200m, 0.5, 1）。例如，概念证明 = 500m，最小部署 =1	
elasticsearch: resources: limits: memory:	根据您的环境配置，可用的内存限值。	以字节为单位指定（例如: 200Ki, 50Mi, 5Gi）。例如，概念证明 = 1Gi，最小部署 = 16Gi*	

参数	描述	值	默认值
elasticsearch: redundancyPolicy:	数据复制策略定义如何在集群中的数据节点之间复制 Elasticsearch 分片：如果没有指定，Jaeger Operator 会自动根据节点数量决定最合适的复制。	**ZeroRedundancy**（无副本分片）、**SingleRedundancy**（一个副本分片）、**MultipleRedundancy**（每个索引分散于一半的 Data 节点）、**FullRedundancy**（每个索引在集群中的每个 Data 节点上完全复制）。	
	*通过这个设置可以使每个 Elasticsearch 节点使用较低内存进行操作，但对于生产环境部署，不建议这样做。对于生产环境，您应该默认为每个 Pod 分配不少于 16Gi 内存，但最好为每个 Pod 最多分配 64Gi 内存。		

生产环境存储示例

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: simple-prod
spec:
  strategy: production
  storage:
    type: elasticsearch
    elasticsearch:
      nodeCount: 3
      resources:
        requests:
          cpu: 1
          memory: 16Gi
        limits:
          memory: 16Gi
```

具有持久性存储的存储示例：

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: simple-prod
spec:
  strategy: production
  storage:
    type: elasticsearch
    elasticsearch:
      nodeCount: 1
      storage: (1)
        storageClassName: gp2
        size: 5Gi
      resources:
```

```
      requests:
        cpu: 200m
        memory: 4Gi
      limits:
        memory: 4Gi
    redundancyPolicy: ZeroRedundancy
```

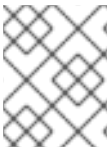
1 持久性存储配置。在本例中，AWS **gp2** 的大小为 **5Gi**。如果没有指定值，Jaeger 将使用 **emptyDir**。OpenShift Elasticsearch Operator 置备 **PersistentVolumeClaim** 和 **PersistentVolume**，它们不会在 Jaeger 实例中删除。如果创建具有相同名称和命名空间的 Jaeger 实例，则可以挂载同一卷。

3.2.4.5.2. 连接到现有 Elasticsearch 实例

Jaeger 还允许您使用现有（自置备的）Elasticsearch 集群进行存储。您可以在配置中将 **spec:storage:options:es:server-urls** 设置为已存在集群的 URL。

限制

- 每个命名空间只能有一个具有自助置备 Elasticsearch 实例的 Jaeger。



注意

红帽不为外部 Elasticsearch 实例提供支持。您可以在客户门户网站中查看经过测试的集成列表。

以下配置参数适用于一个 *外部* Elasticsearch 实例，即不是由 OpenShift Elasticsearch Operator 创建的实例。在配置文件中，您可以在 **spec:storage:options:es** 下为外部 Elasticsearch 指定配置选项。

表 3.8. 常规 ES 配置参数

参数	描述	值	默认值
`es:` `  server-urls:`	Elasticsearch 实例的 URL。	Elasticsearch 服务器的完全限定域名。	**http://elasticsearch.<namespace>.svc:9200**
`es:` `  max-doc-count:`	从 Elasticsearch 查询返回的最大文档数量。这也适用于聚合。如果同时设置了 **es.max-doc-count** 和 **es.max-num-spans**，Elasticsearch 将使用两者中的较小的值。		10000

参数	描述	值	默认值
es: max-num-spans:	[**已弃用** - 将在以后的版本中删除，使用 **es.max-doc-count** 代替。] 在 Elasticsearch 中每个查询每次抓取的最大 span 数量。如果同时设置了 **es.max-num-spans** 和 **es.max-doc-count**，Elasticsearch 将使用两者中的较小的值。		10000
es: max-span-age:	Elasticsearch 中 span 的最大查询。		72h0m0s
es: sniffer:	Elasticsearch 的侦察器配置。客户端使用侦察过程自动查找所有节点。默认禁用此选项。	**true**/ **false**	**false**
es: sniffer-tls-enabled:	在对 Elasticsearch 集群进行 sniffing 时启用 TLS 的选项，客户端使用 sniffing 过程自动查找所有节点。默认禁用	**true**/ **false**	**false**
es: timeout:	用于查询的超时。当设为零时，则没有超时。		0s
es: username:	Elasticsearch 所需的用户名。如果指定，基本身份验证也会加载 CA。另请参阅 **es.password**。		
es: password:	Elasticsearch 所需的密码。另请参阅 **es.username**。		
es: version:	主要的 Elasticsearch 版本。如果没有指定，则该值将从 Elasticsearch 中自动探测到。		0

表 3.9. ES 数据复制参数

参数	描述	值	默认值
es: num-replicas:	Elasticsearch 中每个索引的副本数。		1
es: num-shards:	Elasticsearch 中每个索引的分片数量。		5

表 3.10. ES 索引配置参数

参数	描述	值	默认值
es: create-index-templates:	设置为 **true** 时，应用程序启动时自动创建索引模板。手动安装模板时，设置为 **false**。	**true**/ **false**	**true**
es: index-prefix:	Jaeger 索引的可选前缀。例如，将它设置为 "production" 会创建名为 "production-jaeger-*" 的索引。		

表 3.11. ES 批量处理器配置参数

参数	描述	值	默认值
es: bulk: actions:	在批量处理器决定向磁盘提交更新前可添加到队列的请求数。		1000
es: bulk: flush-interval:	**提交批量请求的时间**. 要禁用批量处理器清除间隔，请将其设置为零。		200ms
es: bulk: size:	在批量处理器决定提交更新之前，批量请求可以处理的字节数。		5000000

参数	描述	值	默认值
es: bulk: workers:	可以接收并将批量请求提交 Elasticsearch 的 worker 数量。		1

表 3.12. ES TLS 配置参数

参数	描述	值	默认值
es: tls: ca:	用于验证远程服务器的 TLS 认证机构（CA）文件的路径。		默认将使用系统信任存储。
es: tls: cert:	TLS 证书文件的路径，用来识别此进程到远程服务器。		
es: tls: enabled:	与远程服务器对话时启用传输层安全（TLS）。默认禁用此选项。	**true/ false**	**false**
es: tls: key:	TLS 私钥文件的路径，用于识别此进程到远程服务器。		
es: tls: server-name:	覆盖远程服务器证书中的预期 TLS 服务器名称。		
es: token-file:	包含 bearer 令牌的文件的路径。如果指定该标志，该标志也会载入认证机构（CA）文件。		

表 3.13. ES 归档配置参数

参数	描述	值	默认值
es-archive: bulk: actions:	在批量处理器决定向磁盘提交更新前可添加到队列的请求数。		0
es-archive: bulk: flush-interval:	**提交批量请求的时间**. 要禁用批量处理器清除间隔，请将其设置为零。		0s
es-archive: bulk: size:	在批量处理器决定提交更新之前，批量请求可以处理的字节数。		0
es-archive: bulk: workers:	可以接收并将批量请求提交 Elasticsearch 的 worker 数量。		0
es-archive: create-index-templates:	设置为 **true** 时，应用程序启动时自动创建索引模板。手动安装模板时，设置为 **false**。	**true**/ **false**	**false**
es-archive: enabled:	启用额外的存储。	**true**/ **false**	**false**
es-archive: index-prefix:	Jaeger 索引的可选前缀。例如，将它设置为 "production" 会创建名为 "production-jaeger-*" 的索引。		
es-archive: max-doc-count:	从 Elasticsearch 查询返回的最大文档数量。这也适用于聚合。		0
es-archive: max-num-spans:	[**已弃用** - 将在以后的版本中删除，使用 **es-archive.max-doc-count** 替代。] Elasticsearch 中的每个查询一次获取的最大 span 数量。		0

参数	描述	值	默认值
es-archive: max-span-age:	Elasticsearch 中 span 的最大查询。		0s
es-archive: num-replicas:	Elasticsearch 中每个索引的副本数。		0
es-archive: num-shards:	Elasticsearch 中每个索引的分片数量。		0
es-archive: password:	Elasticsearch 所需的密码。另请参阅 **es.username**。		
es-archive: server-urls:	以逗号分隔的 Elasticsearch 服务器列表。必须指定为完全限定的 URL，例如 **http://localhost:9200**。		
es-archive: sniffer:	Elasticsearch 的侦察器配置。客户端使用侦察过程自动查找所有节点。默认禁用此选项。	**true**/ **false**	**false**
es-archive: sniffer-tls-enabled:	在对 Elasticsearch 集群进行 sniffing 时启用 TLS 的选项，客户端使用 sniffing 过程自动查找所有节点。默认禁用此选项。	**true**/ **false**	**false**
es-archive: timeout:	用于查询的超时。当设为零时，则没有超时。		0s
es-archive: tls: ca:	用于验证远程服务器的 TLS 认证机构（CA）文件的路径。		默认将使用系统信任存储。

参数	描述	值	默认值
es-archive: tls: cert:	TLS 证书文件的路径，用来识别此进程到远程服务器。		
es-archive: tls: enabled:	与远程服务器对话时启用传输层安全（TLS）。默认禁用此选项。	**true**/ **false**	**false**
es-archive: tls: key:	TLS 私钥文件的路径，用于识别此进程到远程服务器。		
es-archive: tls: server-name:	覆盖远程服务器证书中的预期 TLS 服务器名称。		
es-archive: token-file:	包含 bearer 令牌的文件的路径。如果指定该标志，该标志也会载入认证机构（CA）文件。		
es-archive: username:	Elasticsearch 所需的用户名。如果指定，基本身份验证也会加载 CA。请参阅 **es-archive.password**。		
es-archive: version:	主要的 Elasticsearch 版本。如果没有指定，则该值将从 Elasticsearch 中自动探测到。		0

使用卷挂载的存储示例

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: simple-prod
spec:
  strategy: production
  storage:
    type: elasticsearch
    options:
      es:
```

```
      server-urls: https://quickstart-es-http.default.svc:9200
      index-prefix: my-prefix
      tls:
        ca: /es/certificates/ca.crt
  secretName: jaeger-secret
 volumeMounts:
  - name: certificates
    mountPath: /es/certificates/
    readOnly: true
 volumes:
  - name: certificates
    secret:
      secretName: quickstart-es-http-certs-public
```

以下示例显示了使用从存储在 secret 中的卷和用户/密码挂载了 TLS CA 证书的外部 Elasticsearch 集群的 Jaeger CR。

外部 Elasticsearch 示例：

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: simple-prod
spec:
  strategy: production
  storage:
    type: elasticsearch
    options:
      es:
        server-urls: https://quickstart-es-http.default.svc:9200 (1)
        index-prefix: my-prefix
        tls: (2)
          ca: /es/certificates/ca.crt
    secretName: jaeger-secret (3)
  volumeMounts: (4)
    - name: certificates
      mountPath: /es/certificates/
      readOnly: true
  volumes:
    - name: certificates
      secret:
        secretName: quickstart-es-http-certs-public
```

(1) 在默认命名空间中运行的 Elasticsearch 服务 URL。

(2) TLS 配置。在这种情况下，只有 CA 证书，但在使用 mutual TLS 时，它也可以包含 es.tls.key 和 es.tls.cert。

(3) 定义环境变量 ES_PASSWORD 和 ES_USERNAME 的 Secret。由 kubectl create secret generic jaeger-secret --from-literal=ES_PASSWORD=changeme --from-literal=ES_USERNAME=elastic 创建

(4) 被挂载到所有存储组件的卷挂载和卷。

3.2.4.6. Jaeger Query 配置选项

Query 是一个从存储中检索 trace 并托管用户界面来显示它们的服务。

表 3.14. Operator 用来定义 Jaeger Query 的参数

参数	描述	值	默认值

`spec:`
`  query:`
`  replicas:`
	指定要创建的 Query 副本数。	整数，如 **2**。	

表 3.15. 传递给 Query 的 Jaeger 参数

参数	描述	值	默认值

`spec:`
`  query:`
`  options: {}`
	定义 Query 服务的配置选项。		

`options:` `  log-level:`	Query 的日志记录级别。	可能的值有：**trace**、**debug**、**info**、**warning**、**error**、**fatal**、**panic**。	

`options:`
`  query:`
`  base-path:`
	所有 jaeger-query HTTP 路由的基本路径都可设置为非 root 值，例如，**/jaeger** 将导致所有 UI URL 以 **/jaeger** 开头。当在反向代理后面运行 Jaeger-query 时，这很有用。	/{path}	

示例 Query 配置

```
apiVersion: jaegertracing.io/v1
kind: "Jaeger"
metadata:
  name: "my-jaeger"
spec:
  strategy: allInOne
  allInOne:
    options:
      log-level: debug
      query:
        base-path: /jaeger
```

3.2.4.7. Jaeger Ingester 配置选项

Ingester 是一个从 Kafka 主题读取并写入另一个存储后端 (Elasticsearch) 的服务。如果您使用 **allInOne** 或 **production** 部署策略，则不需要配置 Ingester 服务。

表 3.16. 传递给 Ingester 的 Jaeger 参数

参数	描述	值

`spec:`
`  ingester:`
`  options: {}`
	定义 Ingester 服务的配置选项。	

`options:` `  deadlockInterval:`	指定 Ingester 在终止前应该等待消息的时间间隔（以秒或分钟为单位）。默认情况下，死锁时间间隔是禁用的（设为 0），以免在系统初始化时没有消息到达，导致 Ingester 被终止。	分钟和秒，例如 **1m0s**。默认值为 **0**。

`options:`
`  kafka:`
`  consumer:`
`  topic:`
	topic 参数标识收集器用来生成消息的 Kafka 配置以及要使用消息的 ingester。	consumer 的标签例如，**jaeger-spans**.

`kafka:`
`  consumer:`
`  brokers:`
	Ingester 用来使用消息的 Kafka 配置的标识。	代理的标签，如 **my-cluster-kafka-brokers.kafka:9092**。

`ingester:` `  deadlockInterval:`	指定 Ingester 在终止前应该等待消息的时间间隔（以秒或分钟为单位）。默认情况下，死锁时间间隔是禁用的（设为 0），以免在系统初始化时没有消息到达，导致 Ingester 被终止。	分钟和秒，例如 **1m0s**。默认值为 **0**。
`log-level:`	Ingester 的日志记录级别。	可能的值有：**trace**、**debug**、**info**、**warning**、**error**、**fatal**、**panic**。
`maxReplicas:`	指定在自动扩展 Ingester 时创建的最大副本数。	整数，如 **100**。

流传输 Collector 和 Ingester 示例

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
```

```
metadata:
  name: simple-streaming
spec:
  strategy: streaming
  collector:
    options:
      kafka:
        producer:
          topic: jaeger-spans
          brokers: my-cluster-kafka-brokers.kafka:9092
  ingester:
    options:
      kafka:
        consumer:
          topic: jaeger-spans
          brokers: my-cluster-kafka-brokers.kafka:9092
      ingester:
        deadlockInterval: 5
  storage:
    type: elasticsearch
    options:
      es:
        server-urls: http://elasticsearch:9200
```

3.2.4.7.1. 配置 Ingester 进行自动扩展

注意

自动扩展只支持 Jaeger 1.20 或更高版本。

您可以将 Ingester 配置为自动扩展，Ingester 将根据 CPU 和/或内存的使用情况进行扩展或缩减。将 Ingester 配置为自动扩展可帮助您确保在负载增加时扩展 Jaeger 环境，并在需要较少资源时缩减资源以节约成本。您可以通过将 **autoscale** 参数设置为 **true** 来配置自动扩展，并为您希望 Ingester 的 pod 使用的资源指定一个 **.spec.ingester.maxReplicas** 的值。如果没有为 **.spec.ingester.maxReplicas** 设置值，Operator 将把它设置为 **100**。

默认情况下，当没有为 **.spec.ingester.replicas** 提供值时，Jaeger Operator 会为 Ingester 创建 Horizontal Pod Autoscaler（HPA）配置。如需有关 HPA 的更多信息，请参阅 Kubernetes 文档。

以下是一个自动扩展配置示例，设置 Ingester 的限制以及最大副本数：

Ingester 自动扩展示例

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: simple-streaming
spec:
  strategy: streaming
  ingester:
    maxReplicas: 8
    resources:
```

```
    limits:
      cpu: 100m
      memory: 128Mi
```

3.2.5. 注入 sidecar

OpenShift Jaeger 依赖于应用程序 Pod 中的代理 sidecar 来提供代理。Jaeger Operator 可以将 Jaeger Agent sidecar 注入 Deployment 工作负载。您可以使用自动 sidecar 注入功能或手动管理它。

3.2.5.1. 自动注入 sidecar

要启用此功能，您需要将注解 **sidecar.jaegertracing.io/inject** 设置为 **true** 或 **oc get jaegers** 返回的 Jaeger 实例名称。当您指定 **true** 时，与部署相同的命名空间应该只有一个 Jaeger 实例。否则，Operator 无法决定使用哪个 Jaeger 实例。部署中的特定 Jaeger 实例名称的优先级高于其命名空间中应用的 **true**。

以下片段显示一个简单的应用程序，它将注入一个 sidecar，其中 Jaeger Agent 指向同一个命名空间中可用的单个 Jaeger 实例：

示例自动 sidecar 注入

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
  annotations:
    "sidecar.jaegertracing.io/inject": "true" 1
spec:
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
        image: acme/myapp:myversion
```

当 sidecar 注入时，Ceger Agent 便可以通过 **localhost** 上的默认位置来访问。

3.2.5.2. 手动注入 sidecar

对于 **Deployment** 以外的控制器类型（如 **StatefulSets**、**DaemonSet** 等），您可以在规格中手动定义 Jaeger Agent sidecar。

以下片段显示了您可以在 Jaeger Agent sidecar 的 containers 部分中包含的手动定义：

StatefulSet 的 sidecar 定义示例

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
```

```
  name: example-statefulset
  namespace: example-ns
  labels:
    app: example-app
spec:

    spec:
      containers:
        - name: example-app
          image: acme/myapp:myversion
          ports:
            - containerPort: 8080
              protocol: TCP
        - name: jaeger-agent
          image: registry.redhat.io/distributed-tracing/jaeger-agent-rhel7:<version>
           # The agent version must match the operator version
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 5775
              name: zk-compact-trft
              protocol: UDP
            - containerPort: 5778
              name: config-rest
              protocol: TCP
            - containerPort: 6831
              name: jg-compact-trft
              protocol: UDP
            - containerPort: 6832
              name: jg-binary-trft
              protocol: UDP
            - containerPort: 14271
              name: admin-http
              protocol: TCP
          args:
            - --reporter.grpc.host-port=dns:///jaeger-collector-headless.example-ns:14250
            - --reporter.type=grpc
```

然后，Jaeger Agent 可以通过 localhost 上的默认位置来访问。

3.3. 升级 JAEGER

Operator Lifecycle Manager (OLM) 控制集群中 Operator 的安装、升级和基于角色的访问控制 (RBAC)。OLM 在 OpenShift Container Platform 中默认运行。OLM 可以查询可用的 Operator 以及已安装的 Operator 的升级。如需了解有关 OpenShift Container Platform 如何处理升级的更多信息，请参阅 Operator Lifecycle Manager 文档。

Jaeger Operator 使用的更新方法是将受管 Jaeger 实例升级到与 Operator 相关的版本。安装新版本的 Jaeger Operator 时，由 Operator 管理的所有 Jaeger 应用程序实例都会升级到 Operator 的版本。例如，如果安装了 1.10 版本（包括 Operator 和后端组件），Operator 升级到了 1.11 版本，那么 Operator 就会在 Operator 升级完成后扫描运行 Jaeger 实例并将其升级到 1.11。

3.4. 删除 JAEGER

从 OpenShift Container Platform 集群中删除 Jaeger 的步骤如下：

1. 关闭任何 Jaeger pod。
2. 删除任何 Jaeger 实例。
3. 删除 Jaeger Operator

3.4.1. 使用 Web 控制台删除 Jaeger 实例

注意

当删除使用内存存储的实例时，其所有数据将永久丢失。当 Jaeger 实例被删除时，存储在持久性存储中的数据（如 Elasticsearch）不会被删除。

流程

1. 登陆到 OpenShift Container Platform Web 控制台。
2. 导航到 **Operators → Installed Operators**。
3. 从 Project 菜单中选择 Operators 安装的项目名称，如 **jaeger-system**。
4. 点 Jaeger Operator
5. 点 **Jaeger** 标签页。
6. 点击您要删除  的实例旁的 Options 菜单，然后选择 **Delete Jaeger**。
7. 在确认信息中，点击 **Delete**。

3.4.2. 通过 CLI 删除 Jaeger 实例

1. 登录 OpenShift Container Platform CLI。

```
$ oc login
```

2. 要显示 Jaeger 实例，请运行以下命令：

```
$ oc get deployments -n <jaeger-project>
```

Operator 的名称具有后缀 **-operator**。以下示例显示了两个 Jaeger Operators 和四个 Jaeger 实例：

```
$ oc get deployments -n jaeger-system
```

您应该看到类似如下的输出：

```
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
elasticsearch-operator   1/1     1            1           93m
jaeger-operator          1/1     1            1           49m
jaeger-test              1/1     1            1           7m23s
```

```
jaeger-test2          1/1    1          1          6m48s
tracing1              1/1    1          1          7m8s
tracing2              1/1    1          1          35m
```

3. 要删除 Jaeger 实例，请运行以下命令：

```
$ oc delete jaeger <deployment-name> -n <jaeger-project>
```

例如，

```
$ oc delete jaeger tracing2 -n jaeger-system
```

4. 要验证删除过程，请再次运行 **oc get deployment**：

```
$ oc get deployments -n <jaeger-project>
```

例如，

```
$ oc get deployments -n jaeger-system
```

您应该看到类似如下的输出：

```
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
elasticsearch-operator   1/1     1          1          94m
jaeger-operator          1/1     1          1          50m
jaeger-test              1/1     1          1          8m14s
jaeger-test2             1/1     1          1          7m39s
tracing1                 1/1     1          1          7m59s
```

3.4.3. 删除 Jaeger Operator

流程

1. 按照从集群中删除 Operator 的说明进行操作。
   - 删除 Jaeger Operator
   - 删除 Jaeger Operator 后（如果适用），删除 OpenShift Elasticsearch Operator。