OpenShift Container Platform 4.6

分布式追踪

分布式追踪安装、使用与发行注记

Last Updated: 2023-02-27

OpenShift Container Platform 4.6 分布式追踪

分布式追踪安装、使用与发行注记

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本文档提供了有关如何在 OpenShift Container Platform 中使用分布式追踪的信息。

目录

第 1 章 分布式追踪发行注记 **4**
1.1. 分布式追踪概述 4
1.2. 让开源更具包容性 4
1.3. 获取支持 4
1.4. 新功能及功能增强 4
1.4.1. Red Hat OpenShift distributed tracing 2.5 的新功能和增强 4
1.4.1.1. Red Hat OpenShift distributed tracing 2.5 支持的组件版本。 5
1.4.2. Red Hat OpenShift distributed tracing 2.4 的新功能和功能增强 5
1.4.2.1. Red Hat OpenShift distributed tracing 版本 2.4 支持的组件版本 5
1.4.3. Red Hat OpenShift distributed tracing 2.3.1 的新功能和功能增强 5
1.4.3.1. Red Hat OpenShift distributed tracing 版本 2.3.1 支持的组件版本 6
1.4.4. Red Hat OpenShift distributed tracing 2.3.0 的新功能和功能增强 6
1.4.4.1. Red Hat OpenShift distributed tracing 版本 2.3.0 支持的组件版本 6
1.4.5. Red Hat OpenShift distributed tracing 2.2.0 的新功能及改进 6
1.4.5.1. Red Hat OpenShift distributed tracing 版本 2.2.0 支持的组件版本 6
1.4.6. Red Hat OpenShift distributed tracing 2.1.0 的新功能和功能增强 6
1.4.6.1. Red Hat OpenShift distributed tracing 版本 2.1.0 支持的组件版本 6
1.4.7. Red Hat OpenShift distributed tracing 2.0.0 的新功能及改进 7
1.4.7.1. Red Hat OpenShift distributed tracing 版本 2.0.0 支持的组件版本 7
1.5. RED HAT OPENSHIFT DISTRIBUTED TRACING 技术预览 7
1.5.1. Red Hat OpenShift distributed tracing 2.4.0 技术预览 8
1.5.2. Red Hat OpenShift distributed tracing 2.2.0 技术预览 8
1.5.3. Red Hat OpenShift distributed tracing 2.1.0 技术预览 8
1.5.4. Red Hat OpenShift distributed tracing 2.0.0 技术预览 8
1.6. RED HAT OPENSHIFT 分布式追踪已知问题 9
1.7. RED HAT OPENSHIFT 分布式追踪问题 9

第 2 章 分布式追踪架构 **11**
2.1. 分布式追踪架构 11
2.1.1. 分布式追踪概述 11
2.1.2. Red Hat OpenShift distributed tracing 功能 11
2.1.3. Red Hat OpenShift distributed tracing 架构 11

第 3 章 分布式追踪安装 **13**
3.1. 安装分布式追踪 13
3.1.1. 先决条件 13
3.1.2. Red Hat OpenShift distributed tracing 安装概述 13
3.1.3. 安装 OpenShift Elasticsearch Operator 14
3.1.4. 安装 Red Hat OpenShift distributed tracing Platform Operator 15
3.1.5. 安装 Red Hat OpenShift distributed tracing 数据收集 Operator 16
3.2. 配置和部署分布式追踪 17
3.2.1. 从 Web 控制台部署分布式追踪默认策略 18
3.2.1.1. 通过 CLI 部署分布式追踪默认策略 19
3.2.2. 从 Web 控制台部署分布式追踪生产环境策略 20
3.2.2.1. 通过 CLI 部署分布式追踪产品策略 22
3.2.3. 从 Web 控制台部署分布式追踪流策略 22
3.2.3.1. 通过 CLI 部署分布式追踪流策略 24
3.2.4. 验证部署 25
3.2.4.1. 访问 Jaeger 控制台 25
3.2.5. 自定义部署 26
3.2.5.1. 部署最佳实践 26

3.2.5.2. 分布式追踪默认配置选项 26
3.2.5.3. Jaeger Collector 配置选项 28
3.2.5.4. 分布式追踪抽样配置选项 29
3.2.5.5. 分布式追踪存储配置选项 31
3.2.5.5.1. 自动置备 Elasticsearch 实例 32
3.2.5.5.2. 连接到现有 Elasticsearch 实例 35
3.2.5.6. 使用 Elasticsearch 管理证书 43
3.2.5.7. 查询配置选项 45
3.2.5.8. Ingester 配置选项 46
3.2.6. 注入 sidecar 47
3.2.6.1. 自动注入 sidecar 47
3.2.6.2. 手动注入 sidecar 48
3.3. 配置和部署分布式追踪数据收集 49
3.3.1. OpenTelemetry Collector 配置选项 49
3.3.2. 验证部署 52
3.3.3. 访问 Jaeger 控制台 52
3.4. 升级分布式追踪 52
3.4.1. 更改 2.0 的 Operator 频道 53
3.5. 删除分布式追踪 53
3.5.1. 使用 Web 控制台删除 Red Hat OpenShift distributed tracing Platform 实例 53
3.5.2. 通过 CLI 删除 Red Hat OpenShift distributed tracing 平台实例 54
3.5.3. 删除 Red Hat OpenShift distributed tracing Operator 55

第 1 章 分布式追踪发行注记

1.1. 分布式追踪概述

作为服务所有者，您可以使用分布式追踪来检测您的服务，以收集与服务架构相关的信息。您可以使用分布式追踪来监控、网络性能分析，并对现代、云原生的基于微服务的应用中组件之间的交互进行故障排除。

通过分布式追踪，您可以执行以下功能：

- 监控分布式事务
- 优化性能和延迟时间
- 执行根原因分析

Red Hat OpenShift distributed tracing 包括两个主要组件：

- **Red Hat OpenShift distributed tracing Platform** - 此组件基于开源 Jaeger 项目。
- **Red Hat OpenShift distributed tracing 数据收集** - 此组件基于开源 OpenTelemetry 项目。

这两个组件都基于厂商中立的 OpenTracing API 和工具。

1.2. 让开源更具包容性

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。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。

要识别集群中的问题，您可以在 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，请为相关文档组件提交 JIRA 问题。请提供具体详情，如章节名称和 OpenShift Container Platform 版本。

1.4. 新功能及功能增强

此版本对以下方面进行了改进

1.4.1. Red Hat OpenShift distributed tracing 2.5 的新功能和增强

此 Red Hat OpenShift 分布式追踪发行版本解决了 CVE 报告的安全漏洞问题以及程序错误。

此发行版本为 Red Hat OpenShift distributed tracing platform Operator 引进了对 OpenTelemetry 协议 (OTLP)的支持。Operator 现在自动启用 OTLP 端口：

- 端口 4317 用于 OTLP gRPC 协议。
- 端口 4318 用于 OTLP HTTP 协议。

此发行版本还添加了对为 Red Hat OpenShift distributed tracing 数据收集 Operator 收集 Kubernetes 资源属性的支持。

1.4.1.1. Red Hat OpenShift distributed tracing 2.5 支持的组件版本。

Operator	组件	版本
Red Hat OpenShift distributed tracing Platform	Jaeger	1.36
Red Hat OpenShift distributed tracing 数据收集	OpenTelemetry	0.56

1.4.2. Red Hat OpenShift distributed tracing 2.4 的新功能和功能增强

此 Red Hat OpenShift 分布式追踪发行版本解决了 CVE 报告的安全漏洞问题以及程序错误。

此发行版本还添加了对使用 Red Hat Elasticsearch Operator 自动置备证书的支持。

- 自助置备，这意味着在安装过程中使用 Red Hat OpenShift distributed tracing platform Operator 调用 Red Hat Elasticsearch Operator。此发行版本完全支持自助置备。
- 首先创建 Elasticsearch 实例和证书，然后将分布式追踪平台配置为使用此证书是这个发行版本的技术预览。

注意

当升级到 Red Hat OpenShift distributed tracing 2.4 时，Operator 会重新创建 Elasticsearch 实例，它可能需要 5 到 10 分钟。在此期间内，分布式追踪将停机且不可用。

1.4.2.1. Red Hat OpenShift distributed tracing 版本 2.4 支持的组件版本

Operator	组件	版本
Red Hat OpenShift distributed tracing Platform	Jaeger	1.34.1
Red Hat OpenShift distributed tracing 数据收集	OpenTelemetry	0.49

1.4.3. Red Hat OpenShift distributed tracing 2.3.1 的新功能和功能增强

此 Red Hat OpenShift 分布式追踪发行版本解决了 CVE 报告的安全漏洞问题以及程序错误。

1.4.3.1. Red Hat OpenShift distributed tracing 版本 2.3.1 支持的组件版本

Operator	组件	版本
Red Hat OpenShift distributed tracing Platform	Jaeger	1.30.2
Red Hat OpenShift distributed tracing 数据收集	OpenTelemetry	0.44.1-1

1.4.4. Red Hat OpenShift distributed tracing 2.3.0 的新功能和功能增强

此 Red Hat OpenShift 分布式追踪发行版本解决了 CVE 报告的安全漏洞问题以及程序错误。

在这个版本中，Red Hat OpenShift distributed tracing 平台 Operator 被默认安装到 **openshift-distributed-tracing** 命名空间。在以前的版本中，默认安装位于 **openshift-operators** 命名空间中。

1.4.4.1. Red Hat OpenShift distributed tracing 版本 2.3.0 支持的组件版本

Operator	组件	版本
Red Hat OpenShift distributed tracing Platform	Jaeger	1.30.1
Red Hat OpenShift distributed tracing 数据收集	OpenTelemetry	0.44.0

1.4.5. Red Hat OpenShift distributed tracing 2.2.0 的新功能及改进

此 Red Hat OpenShift 分布式追踪发行版本解决了 CVE 报告的安全漏洞问题以及程序错误。

1.4.5.1. Red Hat OpenShift distributed tracing 版本 2.2.0 支持的组件版本

Operator	组件	版本
Red Hat OpenShift distributed tracing Platform	Jaeger	1.30.0
Red Hat OpenShift distributed tracing 数据收集	OpenTelemetry	0.42.0

1.4.6. Red Hat OpenShift distributed tracing 2.1.0 的新功能和功能增强

此 Red Hat OpenShift 分布式追踪发行版本解决了 CVE 报告的安全漏洞问题以及程序错误。

1.4.6.1. Red Hat OpenShift distributed tracing 版本 2.1.0 支持的组件版本

Operator	组件	版本
Red Hat OpenShift distributed tracing Platform	Jaeger	1.29.1
Red Hat OpenShift distributed tracing 数据收集	OpenTelemetry	0.41.1

1.4.7. Red Hat OpenShift distributed tracing 2.0.0 的新功能及改进

此发行版本将 Red Hat OpenShift Jaeger 重新构建到 Red Hat OpenShift distributed tracing。此发行版本包括以下变化、增加和增强：

- Red Hat OpenShift distributed tracing 现在由以下两个主要组件组成：
  - **Red Hat OpenShift distributed tracing Platform** - 此组件基于开源 Jaeger 项目。
  - **Red Hat OpenShift distributed tracing 数据收集** - 此组件基于开源 OpenTelemetry 项目。
- 将 Red Hat OpenShift distributed tracing Platform Operator 更新至 Jaeger 1.28。在未来，Red Hat OpenShift distributed tracing 只支持 **stable** Operator 频道。独立发行版本的频道不再被支持。
- 引进了基于 OpenTelemetry 0.33 的新 Red Hat OpenShift distributed tracing 数据收集 Operator。请注意，这个 Operator 是一个技术预览功能。
- 为 Query 服务添加了对 OpenTelemetry 协议(OTLP)的支持。
- 引入了 OpenShift OperatorHub 中出现的新分布式追踪图标。
- 包含对文档的滚动更新，以支持名称更改和新功能。

此发行版本还解决了 CVE 报告的安全漏洞问题以及程序错误。

1.4.7.1. Red Hat OpenShift distributed tracing 版本 2.0.0 支持的组件版本

Operator	组件	版本
Red Hat OpenShift distributed tracing Platform	Jaeger	1.28.0
Red Hat OpenShift distributed tracing 数据收集	OpenTelemetry	0.33.0

1.5. RED HAT OPENSHIFT DISTRIBUTED TRACING 技术预览

重要

技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。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，请参考 https://access.redhat.com/support/offerings/techpreview/。

1.5.1. Red Hat OpenShift distributed tracing 2.4.0 技术预览

此发行版本还添加了对使用 Red Hat Elasticsearch Operator 自动置备证书的支持。

- 自助置备，这意味着在安装过程中使用 Red Hat OpenShift distributed tracing platform Operator 调用 Red Hat Elasticsearch Operator。此发行版本完全支持自助置备。
- 首先创建 Elasticsearch 实例和证书，然后将分布式追踪平台配置为使用此证书是这个发行版本的技术预览。

1.5.2. Red Hat OpenShift distributed tracing 2.2.0 技术预览

2.1 发行版本中包含的 OpenTelemetry Collector 组件已被删除。

1.5.3. Red Hat OpenShift distributed tracing 2.1.0 技术预览

此发行版本引入了一个具有破坏性的更改，这个变化与如何在 OpenTelemetry 自定义资源文件中配置证书相关。在新版本中，**ca_file** 在自定义资源中的 **tls** 下移动，如下例所示。

OpenTelemetry 版本 0.33 的 CA 文件配置

```
spec:
  mode: deployment
  config: |
    exporters:
      jaeger:
        endpoint: jaeger-production-collector-headless.tracing-system.svc:14250
        ca_file: "/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt"
```

OpenTelemetry 版本 0.41.1 的 CA 文件配置

```
spec:
  mode: deployment
  config: |
    exporters:
      jaeger:
        endpoint: jaeger-production-collector-headless.tracing-system.svc:14250
        tls:
          ca_file: "/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt"
```

1.5.4. Red Hat OpenShift distributed tracing 2.0.0 技术预览

此发行版本添加了 Red Hat OpenShift distributed tracing 数据收集，您可以使用 Red Hat OpenShift distributed tracing 数据收集安装。Red Hat OpenShift distributed tracing 数据收集基于 OpenTelemetry API 和工具。

Red Hat OpenShift distributed tracing 数据收集包括 OpenTelemetry Operator 和 Collector。Collector 可用于在 OpenTelemetry 或 Jaeger 协议中接收 trace，并将 trace 数据发送到 Red Hat OpenShift distributed tracing。目前还不支持 Collector 的其他功能。

OpenTelemetry 收集器允许开发人员使用与供应商无关的 API 检测其代码，避免了供应商锁定并启用不断增长的可观察性工具生态系统。

1.6. RED HAT OPENSHIFT 分布式追踪已知问题

Red Hat OpenShift 分布式追踪中存在这些限制：

- 不支持 Apache spark。
- IBM Z 和 IBM Power Systems 上不支持通过 AMQ/Kafka 进行流部署。

Red Hat OpenShift 分布式追踪有以下已知的问题：

- TRACING-2057 Kafka API 已更新至 **v1beta2**，以支持 Strimzi Kafka Operator 0.23.0。但是，AMQ Streams 1.6.3 不支持这个 API 版本。如果您有以下环境，将不会升级 Jaeger 服务，您无法创建新的 Jaeger 服务或修改现有的 Jaeger 服务：
  - Jaeger Operator 频道：**1.17.x stable** 或 **1.20.x stable**
  - AMQ Streams Operator 频道： **amq-streams-1.6.x**
    要解决这个问题，将 AMQ Streams Operator 的订阅频道切换到 **amq-streams-1.7.x** 或 **stable**。

1.7. RED HAT OPENSHIFT 分布式追踪问题

- TRACING-2337 Jaeger 在 Jaeger 日志中记录一个重复的警告信息，如下所示：

  ```
  {"level":"warn","ts":1642438880.918793,"caller":"channelz/logging.go:62","msg":"[core]grpc: Server.Serve failed to create ServerTransport: connection error: desc = \"transport: http2Server.HandleStreams received bogus greeting from client: \\\"\\\\x16\\\\x03\\\\x01\\\\x02\\\\x00\\\\x01\\\\x00\\\\x01\\\\xfc\\\\x03\\\\x03vw\\\\x1a\\\\xc9T\\\\xe7\\\\xdaCj\\\\xb7\\\\x8dK\\\\xa6\\\"\"","system":"grpc","grpc_log":true}
  ```

  这个问题已通过只公开查询服务的 HTTP(S)端口而不是 gRPC 端口来解决。

- TRACING-2009 已更新 Jaeger Operator，使其包含对 Strimzi Kafka Operator 0.23.0 的支持。
- TRACING-1907 Jaeger 代理 sidecar 注入失败，因为应用程序命名空间中缺少配置映射。因为 **OwnerReference** 字段设置不正确，配置映射会被自动删除，因此应用程序 pod 不会超过 "ContainerCreating" 阶段。已删除不正确的设置。
- 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。当置备 Kafka 集群时，它会出错：**Failed to pull image registry.redhat.io/amq7/amq-streams-kafka-24-rhel7@sha256:f9ceca004f1b7dccb3b82d9a8027961f9fe4104e0ed69752c0bdd8078b4a1076**.

- 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。

- BZ-1918920/LOG-1619 / LOG-1619，Elasticsearch Pod 在更新后不会自动重启。
临时解决方案：手动重启 pod。

第 2 章 分布式追踪架构

2.1. 分布式追踪架构

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。Red Hat OpenShift distributed tracing 可让您执行分布式追踪，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、并行性和延迟来源有价值。

Red Hat OpenShift distributed tracing 记录了在微服务的整个堆栈间执行单个请求，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 Red Hat OpenShift distributed tracing 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，以及可能的标签和日志。span 可能会被嵌套并排序以模拟因果关系。

2.1.1. 分布式追踪概述

作为服务所有者，您可以使用分布式追踪来检测您的服务，以收集与服务架构相关的信息。您可以使用分布式追踪来监控、网络性能分析，并对现代、云原生的基于微服务的应用中组件之间的交互进行故障排除。

通过分布式追踪，您可以执行以下功能：

- 监控分布式事务
- 优化性能和延迟时间
- 执行根原因分析

Red Hat OpenShift distributed tracing 包括两个主要组件：

- **Red Hat OpenShift distributed tracing Platform** - 此组件基于开源 Jaeger 项目。
- **Red Hat OpenShift distributed tracing 数据收集** - 此组件基于开源 OpenTelemetry 项目。

这两个组件都基于厂商中立的 OpenTracing API 和工具。

2.1.2. Red Hat OpenShift distributed tracing 功能

Red Hat OpenShift distributed tracing 提供了以下功能：

- 与 Kiali 集成 - 当正确配置时，您可以从 Kiali 控制台查看分布式追踪数据。
- 高可伸缩性 - 分布式追踪后端设计具有单一故障点，而且能够按照业务需求进行扩展。
- 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 - Red Hat OpenShift distributed tracing 有 API，它能将其用作 Zipkin 的简易替代品，但红帽在此发行版本中不支持 Zipkin 的兼容性。

2.1.3. Red Hat OpenShift distributed tracing 架构

Red Hat OpenShift distributed tracing 由几个组件组成，它们一起收集、存储和显示追踪数据。

- **Red Hat OpenShift distributed tracing Platform** - 此组件基于开源 Jaeger 项目。
  - **客户端** （Jaeger 客户端、跟踪器、报告程序、客户端库）- 分布式追踪平台客户端是 OpenTracing API 的特定语言实施。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。
  - **代理** （Jaeger 代理，Server Queue, Processor Workers）- 分布式追踪平台代理是一个网络守护进程，侦听通过用户数据报协议(UDP)发送并发送到 Collector。这个代理应被放置在要管理的应用程序的同一主机上。这通常是通过容器环境（如 Kubernetes）中的 sidecar 来实现。
  - **Jaeger Collector** (Collector, Queue, Workers)- 与 Jaeger 代理类似，Jaeger Collector 接收 span，并将它们放置在内部队列中进行处理。这允许 Jaeger Collector 立即返回到客户端/代理，而不是等待 span 变为存储。
  - **Storage** (Data Store) - 收集器需要一个持久的存储后端。Red Hat OpenShift distributed tracing Platform 提供了用于 span 存储的可插拔机制。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)- Red Hat OpenShift distributed tracing 可以使用 Apache Kafka 作为 Collector 和实际的 Elasticsearch 后端存储之间的缓冲。Ingester 是一个从 Kafka 读取数据并写入 Elasticsearch 存储后端的服务。
  - **Jaeger 控制台** - 使用 Red Hat OpenShift distributed tracing 平台用户界面，您可以视觉化您的分布式追踪数据。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。
- **Red Hat OpenShift distributed tracing 数据收集** - 此组件基于开源 OpenTelemetry 项目。
  - **OpenTelemetry Collector** - OpenTelemetry Collector 是一个与厂商无关的方式来接收、处理和导出遥测数据。OpenTelemetry Collector 支持开源可观察数据格式，如 Jaeger 和 Prometheus，发送到一个或多个开源或商业后端。Collector 是默认位置检测库来导出其遥测数据。

第 3 章 分布式追踪安装

3.1. 安装分布式追踪

您可以通过以下两种方式之一在 OpenShift Container Platform 上安装 Red Hat OpenShift distributed tracing：

- 作为 Red Hat OpenShift Service Mesh 的一部分安装 Red Hat OpenShift distributed tracing。Service Mesh 安装中默认包含了分布式追踪。要将 Red Hat OpenShift distributed tracing 作为服务网格的一部分安装，请按照 Red Hat Service Mesh 安装说明进行。您必须在与服务网格相同的命名空间中安装 Red Hat OpenShift distributed tracing，即 **ServiceMeshControlPlane** 和 Red Hat OpenShift distributed tracing 资源必须位于同一命名空间中。

- 如果您不想安装服务网格，您可以使用 Red Hat OpenShift distributed tracing Operator 来自行安装分布式追踪。要在没有服务网格的情况下安装 Red Hat OpenShift distributed tracing，请使用以下说明。

3.1.1. 先决条件

在安装 Red Hat OpenShift distributed tracing 前，请查看安装活动，并确保满足先决条件：

- 您的红帽帐户中拥有活跃的 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 4.6 概述。

- 安装 OpenShift Container Platform 4.6。

  - 在 AWS 上安装 OpenShift Container Platform 4.6

  - 在用户置备的 AWS 上安装 OpenShift Container Platform 4.6

  - 在裸机上安装 OpenShift Container Platform 4.6

  - 在 vSphere 上安装 OpenShift Container Platform 4.6

- 安装与 OpenShift Container Platform 版本匹配的 OpenShift CLI (**oc**) 版本，并将其添加到您的路径中。

- 具有 **cluster-admin** 角色的帐户。

3.1.2. Red Hat OpenShift distributed tracing 安装概述

安装 Red Hat OpenShift distributed tracing 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 Red Hat OpenShift distributed tracing Platform Operator。

- 修改自定义资源 YAML 文件，以支持您的部署策略。

- 在 OpenShift Container Platform 环境中部署一个或多个 Red Hat OpenShift distributed tracing 平台的实例。

3.1.3. 安装 OpenShift Elasticsearch Operator

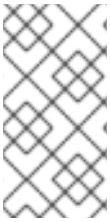
默认 Red Hat OpenShift distributed tracing 平台部署使用内存存储，因为它旨在快速安装用于评估 Red Hat OpenShift distributed tracing、提供演示或在测试环境中使用 Red Hat OpenShift distributed tracing 平台的用户。如果您计划在生产环境中使用 Red Hat OpenShift distributed tracing 平台，则必须安装并配置持久性存储选项，即 Elasticsearch。

先决条件

- 访问 OpenShift Container Platform web 控制台。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-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。Red Hat OpenShift distributed tracing Platform Operator 使用已安装的 OpenShift Elasticsearch Operator 创建 Elasticsearch 实例。

步骤

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。
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 **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您的 Operator。
7. 接受默认的 **All namespaces on the cluster (default)**。这会在默认的 **openshift-operators-redhat** 项目中安装 Operator，并使 Operator 可供集群中的所有项目使用。

注意

Elasticsearch 安装需要 OpenShift Elasticsearch Operator 的 **openshift-operators-redhat** 命名空间。其他 Red Hat OpenShift distributed tracing Operator 安装在 **openshift-operators** 命名空间中。

- 接受默认的 **Automatic** 批准策略。默认情况下，当这个 Operator 的新版本可用时，Operator Lifecycle Manager(OLM)将自动升级 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

Manual 批准策略需要具有适当凭证的用户批准 Operator 的安装和订阅过程。

8. 点 **Install**。

9. 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 Red Hat OpenShift distributed tracing Platform Operator

要安装 Red Hat OpenShift distributed tracing 平台，请使用 OperatorHub 安装 Red Hat OpenShift distributed tracing Platform 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。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。
- 如果需要持久性存储，则必须在安装 Red Hat OpenShift distributed tracing Platform 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。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。
2. 导航至 **Operators → OperatorHub**。
3. 在过滤器中输入 **distributing tracing platform** 找到 Red Hat OpenShift distributed tracing platform Operator。
4. 点由红帽提供的 **Red Hat OpenShift distributed tracing platform Operator** 来现实与 Operator 相关的信息。
5. 点 **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您的 Operator。

7. 接受默认的 **All namespaces on the cluster (default)**。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。

   - 接受默认的 **Automatic** 批准策略。默认情况下，当这个 Operator 的新版本可用时，Operator Lifecycle Manager(OLM)将自动升级 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

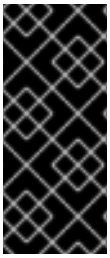
Manual 批准策略需要具有适当凭证的用户批准 Operator 的安装和订阅过程。

8. 点 **Install**。

9. 导航到 **Operators → Installed Operators**。

10. 在 **Installed Operators** 页面中，选择 **openshift-operators** 项目。等待 Red Hat OpenShift distributed tracing Platform Operator 的状态显示为 "Succeed" 状态，然后再继续。

3.1.5. 安装 Red Hat OpenShift distributed tracing 数据收集 Operator



重要

Red Hat OpenShift distributed tracing 数据收集 Operator 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。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，请参考 https://access.redhat.com/support/offerings/techpreview/。

要安装 Red Hat OpenShift distributed tracing 数据收集，您可以使用 OperatorHub 安装 Red Hat OpenShift distributed tracing 数据收集 Operator。

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

先决条件

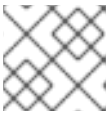
- 访问 OpenShift Container Platform web 控制台。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。

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。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。
2. 导航至 **Operators → OperatorHub**。
3. 在过滤器中输入 **distributing tracing data collection** 找到 Red Hat OpenShift distributed tracing data collection Operator。
4. 点红帽提供的 **Red Hat OpenShift distributed tracing 数据收集 Operator** 来显示有关 Operator 的信息。
5. 点 **Install**。
6. 在 **Install Operator** 页面中，接受默认的 **stable** 更新频道。这可在发布新版本时自动更新您的 Operator。
7. 接受默认的 **All namespaces on the cluster (default)**。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。
8. 接受默认的 **Automatic** 批准策略。默认情况下，当这个 Operator 的新版本可用时，Operator Lifecycle Manager(OLM)将自动升级 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

Manual 批准策略需要具有适当凭证的用户批准 Operator 的安装和订阅过程。

9. 点 **Install**。
10. 导航到 **Operators → Installed Operators**。
11. 在 **Installed Operators** 页面中，选择 **openshift-operators** 项目。等待 Red Hat OpenShift distributed tracing 数据收集 Operator 的状态显示为 "Succeeded"。

3.2. 配置和部署分布式追踪

Red Hat OpenShift distributed tracing Platform Operator 使用一个自定义资源定义(CRD)文件来定义创建和部署分布式追踪平台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。

Red Hat OpenShift distributed tracing Platform 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。当您创建分布式追踪平台实例时，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: MyConfigFile
spec:
  strategy: production (1)
```

(1) Red Hat OpenShift distributed tracing Platform 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

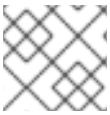
内存存储不是持久性的，这意味着如果分布式追踪平台实例关闭、重启或被替换，您的 trace 数据将会丢失。此外，内存存储无法扩展，因为每个 Pod 都有自己的内存。对于持久性存储，您必须使用 **production** 或 **streaming** 策略，这些策略使用 Elasticsearch 作为默认存储。

- **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 Elasticsearch 后端存储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。

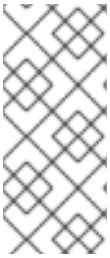
注意

streaming 策略需要额外的 AMQ Streams 订阅。



注意

目前 IBM Z 不支持 streaming 部署策略。

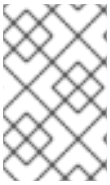


注意

有两种方法可用来安装和使用 Red Hat OpenShift distributed tracing，作为服务网格的一部分或作为独立组件。如果您已将分布式追踪作为 Red Hat OpenShift Service Mesh 的一部分安装，您可以作为 ServiceMeshControlPlane 的一部分执行基本配置，但为了完全控制，应配置一个 Jaeger CR，然后在 ServiceMeshControlPlane 中引用您的分布式追踪配置文件。

3.2.1. 从 Web 控制台部署分布式追踪默认策略

自定义资源定义(CRD)定义部署 Red Hat OpenShift distributed tracing 实例时使用的配置。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 Red Hat OpenShift distributed tracing 平台实例，或者您可以定义自己的自定义资源文件。



注意

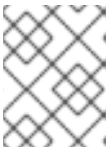
内存存储不是持久性的。如果 Jaeger pod 关闭、重启或被替换，您的 trace 数据将会丢失。对于持久性存储，您必须使用 **production** 或 **streaming** 策略，这些策略使用 Elasticsearch 作为默认存储。

先决条件

- 已安装 Red Hat OpenShift distributed tracing Platform Operator。
- 您已查看了如何自定义部署的说明。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

步骤

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 **tracing-system**。



注意

如果作为 Service Mesh 的一部分安装，则需要在与 **ServiceMeshControlPlane** 资源相同的命名空间中安装分布式追踪资源，如 **istio-system**。

   a. 浏览至 **Home** → **Project**。
   b. 点击 **Create Project**。
   c. 在 **Name** 字段中输入 **tracing-system**。
   d. 点 **Create**。

3. 导航到 **Operators** → **Installed Operators**。
4. 如有必要，从 **Project** 菜单中选择 **tracing-system**。您可能需要等待一些时间，让 Operator 复制到新项目中。
5. 点 Red Hat OpenShift distributed tracing Platform Operator。在 **Details** 标签页中的 **Provided APIs** 下，Operator 提供了一个单个链接。
6. 在 **Jaeger** 下，点 **Create Instance**。
7. 在 **Create Jaeger** 页面上，要使用默认值进行安装，请点击 **Create** 来创建分布式追踪平台实例。
8. 在 **Jaegers** 页面上，点击分布式追踪平台实例的名称，如 **jaeger-all-in-one-inmemory**。
9. 在 **Jaeger Details** 页面上，点击 **Resources** 选项卡。等待 pod 的状态变为"Running"再继续操作。

3.2.1.1. 通过 CLI 部署分布式追踪默认策略

按照以下步骤从命令行创建分布式追踪平台实例。

先决条件

- 已安装并验证 Red Hat OpenShift distributed tracing 平台 Operator。
- 您已查看了如何自定义部署的说明。
- 您可以访问与 OpenShift Container Platform 版本匹配的 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

步骤

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

```
$ oc login --username=<NAMEOFUSER> https://<HOSTNAME>:8443
```

2. 创建一个名为 **tracing-system** 的新项目。

```
$ oc new-project tracing-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分布式追踪平台：

```
$ oc create -n tracing-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 tracing-system -w
```

安装过程完成后，您应该看到类似以下示例的输出：

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

3.2.2. 从 Web 控制台部署分布式追踪生产环境策略

production 部署策略主要用于生产环境，它需要更具扩展性和高度可用的架构，在此情况下，对 trace 数据进行长期存储非常重要。

先决条件

- 已安装 OpenShift Elasticsearch Operator。
- 已安装 Red Hat OpenShift distributed tracing Platform Operator。
- 您已查看了如何自定义部署的说明。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

步骤

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 **tracing-system**。

注意

如果作为 Service Mesh 的一部分安装，则需要在与 **ServiceMeshControlPlane** 资源相同的命名空间中安装分布式追踪资源，如 **istio-system**。

a. 浏览至 **Home → Project**。

b. 点击 **Create Project**。

c. 在 **Name** 字段中输入 **tracing-system**。

d. 点 **Create**。

3. 导航到 **Operators → Installed Operators**。

4. 如有必要，从 **Project** 菜单中选择 **tracing-system**。您可能需要等待一些时间，让 Operator 复制到新项目中。

5. 点 Red Hat OpenShift distributed tracing Platform Operator。在 **Overview** 选项卡上的 **Provided APIs** 下，Operator 提供了单个链接。

6. 在 **Jaeger** 下，点 **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. 单击 **Create** 以创建分布式追踪平台实例。

9. 在 **Jaegers** 页面上，点击分布式追踪平台实例的名称，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 部署分布式追踪产品策略

按照以下步骤从命令行创建分布式追踪平台实例。

先决条件

- 已安装 OpenShift Elasticsearch Operator。
- 已安装 Red Hat OpenShift distributed tracing Platform Operator。
- 您已查看了如何自定义部署的说明。
- 您可以访问与 OpenShift Container Platform 版本匹配的 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

步骤

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

```
$ oc login --username=<NAMEOFUSER> https://<HOSTNAME>:8443
```

2. 创建一个名为 **tracing-system** 的新项目。

```
$ oc new-project tracing-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分布式追踪平台：

```
$ oc create -n tracing-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 tracing-system -w
```

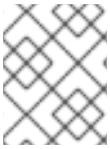
安装过程完成后，您应该看到类似以下示例的输出：

```
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 控制台部署分布式追踪流策略

streaming 部署策略主要用于生产环境，在生产环境中需要更具扩展性和高度可用的架构，其中，对 trace 数据进行长期存储非常重要。

streaming 策略提供了位于 Collector 和 Elasticsearch 存储之间的流功能。这可在高负载情况下减少存储压力，并允许其他 trace 后处理功能直接从 Kafka 流平台利用实时 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。

注意

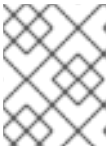
目前 IBM Z 不支持 streaming 部署策略。

先决条件

- 已安装 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，您必须创建 Kafka 实例。
- 已安装 Red Hat OpenShift distributed tracing Platform Operator。
- 您已查看了如何自定义部署的说明。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

步骤

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 **tracing-system**。



注意

如果作为 Service Mesh 的一部分安装，则需要在与 **ServiceMeshControlPlane** 资源相同的命名空间中安装分布式追踪资源，如 **istio-system**。

   a. 浏览至 **Home → Project**。
   b. 点击 **Create Project**。
   c. 在 **Name** 字段中输入 **tracing-system**。
   d. 点 **Create**。
3. 导航到 **Operators → Installed Operators**。
4. 如有必要，从 **Project** 菜单中选择 **tracing-system**。您可能需要等待一些时间，让 Operator 复制到新项目中。
5. 点 Red Hat OpenShift distributed tracing Platform Operator。在 **Overview** 选项卡上的 **Provided APIs** 下，Operator 提供了单个链接。
6. 在 **Jaeger** 下，点 **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. 单击 **Create** 以创建分布式追踪平台实例。
2. 在 **Jaegers** 页面上，点击分布式追踪平台实例的名称，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 部署分布式追踪流策略

按照以下步骤从命令行创建分布式追踪平台实例。

先决条件

- 已安装 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，您必须创建 Kafka 实例。
- 已安装 Red Hat OpenShift distributed tracing Platform Operator。
- 您已查看了如何自定义部署的说明。
- 您可以访问与 OpenShift Container Platform 版本匹配的 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

步骤

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

```
$ oc login --username=<NAMEOFUSER> https://<HOSTNAME>:8443
```

2. 创建一个名为 **tracing-system** 的新项目。

```
$ oc new-project tracing-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 tracing-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 tracing-system -w
```

安装过程完成后，您应该看到类似以下示例的输出：

```
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. 验证部署

3.2.4.1. 访问 Jaeger 控制台

要访问 Jaeger 控制台，您必须安装 Red Hat OpenShift Service Mesh 或 Red Hat OpenShift distributed tracing，并且安装、配置和部署了 Red Hat OpenShift distributed tracing 平台。

安装过程会创建路由来访问 Jaeger 控制台。

如果您知道 Jaeger 控制台的 URL，您可以直接访问它。如果您不知道 URL，请使用以下指示：

从 OpenShift 控制台的步骤

1. 以具有 cluster-admin 权限的用户身份登录到 OpenShift Container Platform web 控制台。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。

2. 进入 **Networking → Routes**。

3. 在 **Routes** 页面中，从 **Namespace** 菜单中选择 control plane 项目，如 **tracing-system**。**Location** 列显示每个路由的链接地址。

4. 如有必要，使用过滤器来查找 **jaeger** 路由。单击路由 **位置** 以启动控制台。

5. 单击 **Log In With OpenShift**。

通过 CLI 操作的步骤

1. 以具有 **cluster-admin** 角色的用户身份登录 OpenShift Container Platform CLI。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。

```
$ oc login --username=<NAMEOFUSER> https://<HOSTNAME>:6443
```

2. 要使用命令行查询路由详情，请输入以下命令。在本例中，**trace-system** 是 control plane 命名空间。

```
$ export JAEGER_URL=$(oc get route -n tracing-system jaeger -o jsonpath='{.spec.host}')
```

3. 启动浏览器并进入 **https://<JAEGER_URL>**，其中 **<JAEGER_URL>** 是您在上一步中发现的路由。

4. 使用您用于访问 OpenShift Container Platform 控制台的相同用户名和密码登录。

5. 如果您已将服务添加到服务网格中并生成了 trace，您可以使用过滤器和 **Find Traces** 按钮搜索 trace 数据。
   如果您要验证控制台安装，则不会显示 trace 数据。

3.2.5. 自定义部署

3.2.5.1. 部署最佳实践

- Red Hat OpenShift distributed tracing 实例名称必须是唯一的。如果您有多个 Red Hat OpenShift distributed tracing 平台实例并使用 sidecar 注入的代理，则 Red Hat OpenShift distributed tracing 平台实例应具有唯一的名称，注入注解应该明确指定追踪数据的名称。

- 如果您有多租户实现，且租户由命名空间分开，请将 Red Hat OpenShift distributed tracing 平台实例部署到每个租户命名空间中。

  - 多租户安装或 Red Hat OpenShift Dedicated 不支持作为 daemonset 的代理。代理作为 sidecar 是这些用例唯一支持的配置。

- 如果您要作为 Red Hat OpenShift Service Mesh 的一部分安装分布式追踪，则分布追踪资源必须与 **ServiceMeshControlPlane** 资源在同一个命名空间中。

有关配置持久性存储的详情，请参考 了解持久性存储 以及您选择的存储选项的适当配置主题。

3.2.5.2. 分布式追踪默认配置选项

Jaeger 自定义资源(CR)定义创建分布式追踪平台资源时要使用的架构和设置。您可以修改这些参数以根据您的业务需求自定义分布式追踪平台实施。

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：		创建对象时要使用的 API 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 Container Platform 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。	分布式追踪平台实例的名称。
jaeger-all-in-one-inmemory	**spec：**	要创建的对象的规格。	包含您分布式追踪平台实例的所有配置参数。当需要所有 Jaeger 组件的通用定义时，会在 **spec** 节点下定义它。当该定义与单个组件相关时，它将放置在 **spec/<component>** 节点下。

参数	描述	值	默认值
N/A	**strategy：**	Jaeger 部署策略	**allInOne**、**production** 或 **streaming**
allInOne	**allInOne：**	因为 **allInOne** 镜像在单个 pod 中部署了 Agent、Collector、Query、Ingester 和 Jaeger UI，所以此部署的配置必须在 **allInOne** 参数下嵌套组件配置。	
	agent：	定义代理的配置选项。	
	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必须放在 **storage** 下，而不是放在 **allInOne** 或其他组件选项下。	
	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 Red Hat OpenShift distributed tracing 平台部署的最低要求。

最低要求示例 dist-tracing-all-in-one.yaml

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

3.2.5.3. 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 Elasticsearch 存储，或使用 **streaming** 策略时将其写入 AMQ Streams。

Collector 是无状态的，因此很多 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 的配置参数

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。
options: 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+，Red Hat OpenShift distributed tracing Platform Operator 将自助置备 Kafka。	
options: log-level:	Collector 的日志记录级别。	可能的值有：**debug**、**info**、**warn**、**error**、**fatal**、**panic**。

3.2.5.4. 分布式追踪抽样配置选项

Red Hat OpenShift distributed tracing Platform 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

如果一个 trace 由 Envoy 代理启动，则不会相关，因为抽样决定是在那里做出的。只有在应用程序使用客户端启动 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，客户端会启动新的 trace，为它分配一个随机 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，以便其他服务不会再次做出抽样决定。

分布式追踪平台库支持以下抽样：

- **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，使用 **sampling.param=0.1** 代表大约为 10 个 trace 抽样 1 次。
- **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，使用 **sampling.param=2.0** 抽样请求，速率为每秒 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，Collector 会返回默认的概率抽样策略，所有服务都为 0.001(0.1%)概率。
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，分布式追踪平台将使用以下设置：

默认抽样

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

3.2.5.5. 分布式追踪存储配置选项

您可以在 **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. Red Hat OpenShift distributed tracing Platform Operator 用来定义分布式追踪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，分布式追踪平台支持 Elasticsearch 进行持久性存储。	**memory**

参数	描述	值	默认值
storage: secretname:	secret 名称，例如 **tracing-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 * * *"

3.2.5.5.1. 自动置备 Elasticsearch 实例

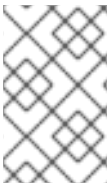
部署 Jaeger 自定义资源时，Red Hat OpenShift distributed tracing platform Operator 会使用 OpenShift Elasticsearch Operator 根据自定义资源文件的 **storage** 部分中提供的配置创建 Elasticsearch 集群。如果设置了以下配置，Red Hat OpenShift distributed tracing Platform Operator 将置备 Elasticsearch：

- **spec.storage:type** 设置为 **elasticsearch**
- **spec.storage.elasticsearch.doNotProvision** 设置为 **false**
- 未定义 **spec.storage.options.es.server-urls**，因此没有连接到 Red Hat Elasticsearch Operator 未置备的 Elasticsearch 实例。

在置备 Elasticsearch 时，Red Hat OpenShift distributed tracing platform Operator 会将 Elasticsearch 自定义资源名称设置为 Jaeger 自定义资源的 **spec.storage.elasticsearch. name** 的值。如果没有为 **spec.storage.elasticsearch.name** 指定一个值，Operator 会使用 **elasticsearch**。

限制

- 每个命名空间只能有一个具有自助置备 Elasticsearch 实例的分布式追踪平台。Elasticsearch 集群旨在专用于单个分布式追踪平台实例。
- 每个命名空间只能有一个 Elasticsearch。



注意

如果您已经安装了 Elasticsearch 作为 OpenShift Logging 的一部分，Red Hat OpenShift distributed tracing Platform Operator 可使用已安装的 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 Red Hat OpenShift distributed tracing Platform 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: properties: doNotProvision:	使用 指定 Red Hat OpenShift distributed tracing platform Operator 是否应该置备 Elasticsearch 实例。	**true**/**false**	**true**
elasticsearch: properties: name:	Elasticsearch 实例的名称。Red Hat OpenShift distributed tracing platform Operator 使用此参数中指定的 Elasticsearch 实例连接到 Elasticsearch。	字符串	**elasticsearch**
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。	以内核数或 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

参数	描述	值	默认值
elasticsearch: resources: limits: cpu:	根据您的环境配置，CPU 数量的限值。	以内核数或 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*	
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，Red Hat OpenShift distributed tracing Platform 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）。	
elasticsearch: useCertManagement:	使用 指定分布式追踪平台是否应使用 Red Hat Elasticsearch Operator 的证书管理功能。此功能被添加到 OpenShift Container Platform 4.7 中的 Red Hat OpenShift 5.2 的日志记录子系统中，是新 Jaeger 部署的首选设置。	**true**/**false**	**true**
	*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，则分布式追踪平台将使用 **emptyDir**。OpenShift Elasticsearch Operator 置备 **PersistentVolumeClaim** 和 **PersistentVolume**，它们不会在分布式追踪平台实例中删除。如果您创建具有相同名称和命名空间的分布式追踪平台实例，则可以挂载同一卷。

3.2.5.5.2. 连接到现有 Elasticsearch 实例

您可以使用现有 Elasticsearch 集群进行分布式追踪存储。现有的 Elasticsearch 集群（也称为 *外部* Elasticsearch 实例）是由 Red Hat OpenShift distributed tracing platform Operator 或 Red Hat Elasticsearch Operator 安装的实例。

部署 Jaeger 自定义资源时，如果设置了以下配置，Red Hat OpenShift distributed tracing Platform Operator 不会置备 Elasticsearch：

- **spec.storage.elasticsearch.doNotProvision** 设置为 **true**
- **spec.storage.options.es.server-urls** 有一个值
- **spec.storage.elasticsearch.name** 具有一个值，或者 Elasticsearch 实例名称是 **elasticsearch**。

Red Hat OpenShift distributed tracing platform Operator 使用 **spec.storage.elasticsearch.name** 中指定的 Elasticsearch 实例连接到 Elasticsearch。

限制

- 您无法将 OpenShift Container Platform 日志记录 Elasticsearch 实例与分布式追踪平台共享或重复使用。Elasticsearch 集群旨在专用于单个分布式追踪平台实例。

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已经存在的 Elasticsearch 实例，也称为 *外部* Elasticsearch 实例。在本例中，您可以在自定义资源文件中的 **spec:storage:options:es** 下为 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
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

参数	描述	值	默认值
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:	在监控 Elasticsearch 集群时启用 TLS 的选项。客户端使用侦察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:	分布式追踪平台索引的可选前缀。例如，将其设置为 "production" 会创建名为 "production-tracing-*" 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
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 证书颁发机构(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:
  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，用来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预期的 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

参数	描述	值	默认值
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:	分布式追踪平台索引的可选前缀。例如，将其设置为 "production" 会创建名为 "production-tracing-*" 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
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

参数	描述	值	默认值
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:	在监控 Elasticsearch 集群时启用 TLS 的选项。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: 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 证书颁发机构(CA)文件的路径。		默认将使用系统信任存储。
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，用来识别此进程到远程服务器。		

参数	描述	值	默认值
es-archive: tls: server-name:	覆盖远程服务器证书中预期的 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: tracing-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: tracing-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。

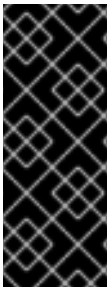
(2) TLS 配置。在这种情况下，只有 CA 证书，但在使用 mutual TLS 时，它也可以包含 es.tls.key 和 es.tls.cert。

(3) 定义环境变量 ES_PASSWORD 和 ES_USERNAME 的 Secret。由 kubectl create secret generic tracing-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。

3.2.5.6. 使用 Elasticsearch 管理证书

您可以使用 Red Hat Elasticsearch Operator 创建和管理证书。使用 Red Hat Elasticsearch Operator 管理证书还可让您使用带有多个 Jaeger Collector 的单个 Elasticsearch 集群。



重要

使用 Elasticsearch 管理证书只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。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，请参考 https://access.redhat.com/support/offerings/techpreview/。

从版本 2.4 开始，Red Hat OpenShift distributed tracing 平台 Operator 使用 Elasticsearch 自定义资源中的以下注解将证书创建委派给 Red Hat Elasticsearch Operator：

- **logging.openshift.io/elasticsearch-cert-management: "true"**
- **logging.openshift.io/elasticsearch-cert.jaeger-<shared-es-node-name>: "user.jaeger"**
- **logging.openshift.io/elasticsearch-cert.curator-<shared-es-node-name>: "system.logging.curator"**

其中 &**lt;shared-es-node-name** > 是 Elasticsearch 节点的名称。例如，如果您创建一个名为 **custom-es** 的 Elasticsearch 节点，您的自定义资源可能类似以下示例。

显示注解的 Elasticsearch CR 示例

```
apiVersion: logging.openshift.io/v1
kind: Elasticsearch
metadata:
  annotations:
    logging.openshift.io/elasticsearch-cert-management: "true"
    logging.openshift.io/elasticsearch-cert.jaeger-custom-es: "user.jaeger"
    logging.openshift.io/elasticsearch-cert.curator-custom-es: "system.logging.curator"
  name: custom-es
spec:
  managementState: Managed
  nodeSpec:
    resources:
      limits:
        memory: 16Gi
      requests:
        cpu: 1
        memory: 16Gi
  nodes:
    - nodeCount: 3
      proxyResources: {}
      resources: {}
      roles:
        - master
        - client
        - data
      storage: {}
  redundancyPolicy: ZeroRedundancy
```

先决条件

- OpenShift Container Platform 4.7
- logging subsystem for Red Hat OpenShift 5.2
- Elasticsearch 节点和 Jaeger 实例必须部署到同一命名空间中。例如，**追踪系统**。

您可以通过在 Jaeger 自定义资源中将 **spec.storage.elasticsearch.useCertManagement** 设置为 **true** 来启用证书管理。

显示 使用CertManagement示例

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

```
metadata:
  name: jaeger-prod
spec:
  strategy: production
  storage:
    type: elasticsearch
    elasticsearch:
      name: custom-es
      doNotProvision: true
      useCertManagement: true
```

在置备 Elasticsearch 时，Red Hat OpenShift distributed tracing platform Operator 将 Elasticsearch 自定义资源名称设置为 Jaeger 自定义资源的 **spec.storage.elasticsearch. name** 的值。

证书由 Red Hat Elasticsearch Operator 和 Red Hat OpenShift distributed tracing 平台 Operator 注入证书。

3.2.5.7. 查询配置选项

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. Red Hat OpenShift distributed tracing Platform Operator 用来定义 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 的配置参数

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：**debug**、**info**、**warn**、**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
```

3.2.5.8. Ingester 配置选项

Ingester 是一个从 Kafka 主题读取并写入 Elasticsearch 存储后端的服务。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 在终止前必须等待消息的时间间隔（以秒为单位）。默认情况下，死锁时间间隔被禁用（设置为 **0**），以避免在系统初始化过程中没有信息到达 Ingester。	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，并使用 Ingester。	consumer 的标签例如，**jaeger-spans**.

`options:`
`  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**。

`options:` `  log-level:`	Ingester 的日志记录级别。	可能的值有：**debug**,**info**,**warn**,**error**,**fatal**,**dpanic**,**panic**.

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
```

3.2.6. 注入 sidecar

Red Hat OpenShift distributed tracing 平台依赖于应用程序 pod 中的代理 sidecar 来提供代理。Red Hat OpenShift distributed tracing Platform Operator 可以将 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。

3.2.6.1. 自动注入 sidecar

Red Hat OpenShift distributed tracing platform Operator 可将 Jaeger Agent sidecar 注入 Deployment 工作负载。要启用 sidecar 的自动注入，请将 **sidecar.jaegertracing.io/inject** 注解设置为字符串 **true** 或运行 **$ oc get jaegers** 返回的分布式追踪平台实例名称。当您指定 **true** 时，与部署相同的命名空间只能有一个分布式追踪平台实例。否则，Operator 无法决定要使用的分布式追踪平台实例。部署中的特定分布式追踪平台实例名称的优先级高于其命名空间中应用的 **true**。

以下片段演示了一个简单的应用程序，它将注入一个 sidecar，代理指向同一命名空间中可用的单个分布式追踪平台实例：

自动 sidecar 注入示例

```
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
```

1 设置为 **true** 字符串或 Jaeger 实例名称。

当 sidecar 被注入后，可以在 **localhost** 上的默认位置访问代理。

3.2.6.2. 手动注入 sidecar

Red Hat OpenShift distributed tracing platform Operator 只能自动将 Jaeger Agent sidecar 注入 Deployment 工作负载。对于 **Deployment 以外的控制器类型，如 StatefulSets 和 'DaemonSets**，您可以在规格中手动定义 Jaeger 代理 sidecar。

以下片段显示了您可以在 Jaeger 代理 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，代理可以在 localhost 上的默认位置访问。

3.3. 配置和部署分布式追踪数据收集

Red Hat OpenShift distributed tracing 数据收集 Operator 使用一个自定义资源定义(CRD)文件来定义创建和部署 Red Hat OpenShift distributed tracing 数据收集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。

3.3.1. OpenTelemetry Collector 配置选项

重要

Red Hat OpenShift distributed tracing 数据收集 Operator 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。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，请参考 https://access.redhat.com/support/offerings/techpreview/。

OpenTelemetry Collector 由访问遥测数据的三个组件组成：

- **Receivers** - 用于控制数据如何进入收集器的接收器，它基于推送或拉取。通常，接收器接受指定格式的数据，将其转换为内部格式，并将其传递给适用管道中定义的处理器和导出器。默认情况下，不会配置接收器。必须配置一个或多个接收器。接收器可以支持一个或多个数据源。
- **Processors** -（可选）处理器在收到和导出的数据中运行。默认情况下，不启用处理器。每个数据源都必须启用处理器。不是所有处理器都支持所有数据源。根据数据源，建议启用多个处理器。此外，请务必注意处理器的顺序，这很重要。
- **Exporters** - 用于控制数据如何发送到一个或多个后端/目的地的导出器，它基于推送或拉取。默认情况下，不会配置导出器。必须配置一个或多个导出器。导出器可能支持一个或多个数据源。导出器可能会附带默认设置，但一般还需要指定目标和安全设置。

您可以在自定义资源 YAML 文件中定义多个组件实例。配置后，必须通过 YAML 文件的 **spec.config.service** 部分中定义的管道启用这些组件。作为最佳实践，您应该只启用您需要的组件。

OpenTelemetry 收集器自定义资源文件示例

```
apiVersion: opentelemetry.io/v1alpha1
kind: OpenTelemetryCollector
metadata:
  name: cluster-collector
  namespace: tracing-system
spec:
```

```
mode: deployment
config: |
  receivers:
    otlp:
      protocols:
        grpc:
        http:
  processors:
  exporters:
    jaeger:
      endpoint: jaeger-production-collector-headless.tracing-system.svc:14250
      tls:
        ca_file: "/var/run/secrets/kubernetes.io/serviceaccount/service-ca.crt"
  service:
    pipelines:
      traces:
        receivers: [otlp]
        processors: []
        exporters: [jaeger]
```

注意

如果一个组件已配置，但没有在 **service** 部分中定义，则不会启用它。

表 3.17. Operator 用来定义 OpenTelemetry Collector 的参数

参数	描述	值	default
receivers:	接收器用于控制数据如何进入 Collector。默认情况下，不会配置接收器。必须至少有一个启用的接收器才能使配置被视为有效。接收器通过添加到管道中来启用。	**otlp**, **jaeger**	无
receivers: otlp:	**oltp** 和 **jaeger** 接收器附带默认设置，一般只需要再指定接收方的名称。		
processors:	处理器在被接收和导出的数据上运行。默认情况下，不启用处理器。		无

参数	描述	值	default
exporters:	导出器将数据发送到一个或多个后端/目标。默认情况下，不会配置导出器。必须至少启用了一个 exporter 时，配置才被视为有效。将导出器添加到管道中即可启用。导出器可能会附带默认设置，但一般还需要指定目标和安全设置。	**logging**, **jaeger**	无
exporters: jaeger: endpoint:	**jaeger** exporter 的端点必须是 **<name>-collector-headless.<namespace>.svc** 的形式，其中包含 Jaeger 部署的名称和命名空间，以便建立安全连接。		
exporters: jaeger: tls: ca_file:	CA 证书的路径。对于客户端，这将验证服务器证书。对于服务器，这会验证客户端证书。如果为空使用系统 root CA。		
service: pipelines:	组件通过将组件添加到 **services.pipeline** 下的管道中来启用。		
service: pipelines: traces: receivers:	您可以通过在 **service.pipelines.traces** 下添加用于追踪的接收器。		无
service: pipelines: traces: processors:	您可以通过在 **service.pipelines.traces** 下添加处理器来启用追踪的处理器。		无
service: pipelines: traces: exporters:	您可以通过在 **service.pipelines.traces** 下添加用于追踪的导出器。		无

3.3.2. 验证部署

3.3.3. 访问 Jaeger 控制台

要访问 Jaeger 控制台，您必须安装 Red Hat OpenShift Service Mesh 或 Red Hat OpenShift distributed tracing，并且安装、配置和部署了 Red Hat OpenShift distributed tracing 平台。

安装过程会创建路由来访问 Jaeger 控制台。

如果您知道 Jaeger 控制台的 URL，您可以直接访问它。如果您不知道 URL，请使用以下指示：

从 OpenShift 控制台的步骤

1. 以具有 cluster-admin 权限的用户身份登录到 OpenShift Container Platform web 控制台。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。
2. 进入 **Networking → Routes**。
3. 在 **Routes** 页面中，从 **Namespace** 菜单中选择 control plane 项目，如 **tracing-system**。**Location** 列显示每个路由的链接地址。
4. 如有必要，使用过滤器来查找 **jaeger** 路由。单击路由 **位置** 以启动控制台。
5. 单击 **Log In With OpenShift**。

通过 CLI 操作的步骤

1. 以具有 **cluster-admin** 角色的用户身份登录 OpenShift Container Platform CLI。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。

```
$ oc login --username=<NAMEOFUSER> https://<HOSTNAME>:6443
```

2. 要使用命令行查询路由详情，请输入以下命令。在本例中，**trace-system** 是 control plane 命名空间。

```
$ export JAEGER_URL=$(oc get route -n tracing-system jaeger -o jsonpath='{.spec.host}')
```

3. 启动浏览器并进入 **https://<JAEGER_URL>**，其中 **<JAEGER_URL>** 是您在上一步中发现的路由。
4. 使用您用于访问 OpenShift Container Platform 控制台的相同用户名和密码登录。
5. 如果您已将服务添加到服务网格中并生成了 trace，您可以使用过滤器和 **Find Traces** 按钮搜索 trace 数据。
   如果您要验证控制台安装，则不会显示 trace 数据。

3.4. 升级分布式追踪

Operator Lifecycle Manager (OLM) 能够控制集群中 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 会查询可用的 Operator 以及已安装的 Operator 的升级。如需有关 OpenShift Container Platform 如何处理升级的更多信息，请参阅 Operator Lifecycle Manager 文档。

在更新过程中，Red Hat OpenShift distributed tracing Operator 会将受管的分布式追踪实例升级到与

Operator 相关的版本。安装新版本的 Red Hat OpenShift distributed tracing Platform Operator 时，由 Operator 管理的所有分布式追踪平台应用程序实例都会升级到 Operator 的版本。例如，在将 Operator 从 1.10 升级到 1.11 后，Operator 扫描运行分布式追踪平台实例并将其升级到 1.11。

有关如何更新 OpenShift Elasticsearch Operator 的具体说明，请参阅更新 OpenShift Logging。

3.4.1. 更改 2.0 的 Operator 频道

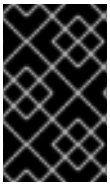
Red Hat OpenShift distributed tracing 2.0.0 进行了以下更改：

- 将 Red Hat OpenShift Jaeger Operator 重命名为 Red Hat OpenShift distributed tracing Platform Operator。
- 停止对单个发行频道的支持。在未来，Red Hat OpenShift distributed tracing Platform Operator 只支持 **stable** Operator 频道。以后的 Operator 不再支持 Maintenance 频道，如 **1.24-stable**。

作为升级到版本 2.0 的一部分，您必须更新 OpenShift Elasticsearch 和 Red Hat OpenShift distributed tracing Platform Operator 订阅。

先决条件

- OpenShift Container Platform 版本为 4.6 或更高版本。
- 您已更新了 OpenShift Elasticsearch Operator。
- 已备份 Jaeger 自定义资源文件。
- 具有 **cluster-admin** 角色的帐户。如果使用 Red Hat OpenShift Dedicated，则必须有一个具有 **dedicated-admin** 角色的帐户。



重要

如果您还没有更新 OpenShift Elasticsearch Operator（如更新 OpenShift Logging 所述），在更新 Red Hat OpenShift distributed tracing Platform Operator 前对它进行更新。

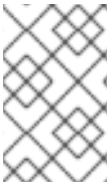
有关如何更新 Operator 频道的步骤，请参阅升级已安装的 Operator。

3.5. 删除分布式追踪

从 OpenShift Container Platform 集群中删除 Red Hat OpenShift distributed tracing 的步骤如下：

1. 关闭任何 Red Hat OpenShift distributed tracing pod。
2. 删除任何 Red Hat OpenShift distributed tracing 实例。
3. 删除 Red Hat OpenShift distributed tracing Platform Operator。
4. 删除 Red Hat OpenShift distributed tracing 数据收集 Operator。

3.5.1. 使用 Web 控制台删除 Red Hat OpenShift distributed tracing Platform 实例



注意

当删除使用内存存储的实例时，所有数据都会永久丢失。当一个 Red Hat OpenShift distributed tracing Platform 实例被删除时，不会删除存储在持久性存储中的数据，如 Elasticsearch。

步骤

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 **openshift-operators**。
4. 点 Red Hat OpenShift distributed tracing Platform 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**。

3.5.2. 通过 CLI 删除 Red Hat OpenShift distributed tracing 平台实例

1. 登录 OpenShift Container Platform CLI。

```
$ oc login --username=<NAMEOFUSER>
```

2. 运行以下命令显示分布式追踪平台实例：

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

例如，

```
$ oc get deployments -n openshift-operators
```

Operator 的名称具有后缀 **-operator**。以下示例显示了两个 Red Hat OpenShift distributed tracing Platform Operator 和四个分布式追踪平台实例：

```
$ oc get deployments -n openshift-operators
```

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. 要删除分布式追踪平台的实例，请运行以下命令：

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

例如：

```
$ oc delete jaeger tracing2 -n openshift-operators
```

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

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

例如：

```
$ oc get deployments -n openshift-operators
```

您应该看到与以下示例类似的输出：

```
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
```

3.5.3. 删除 Red Hat OpenShift distributed tracing Operator

步骤

1. 按照从集群中删除 Operator 的说明进行操作。
   - 删除 Red Hat OpenShift distributed tracing Platform Operator。

- 删除 Red Hat OpenShift distributed tracing Platform Operator 后，请删除 OpenShift Elasticsearch Operator。