OpenShift Container Platform 4.9

분산 추적

분산 추적 설치, 사용법, 릴리스 정보

Last Updated: 2023-05-05

OpenShift Container Platform 4.9 분산 추적

분산 추적 설치, 사용법, 릴리스 정보

법적 공지

초록

이 문서에서는 OpenShift Container Platform에서 분산 추적을 사용하는 방법에 대해 설명합니다.

차례

1장. 분산 추적 릴리스 노트

1.1. 분산 추적 개요

서비스 소유자로 분산 추적을 사용하여 서비스 아키텍처에 대한 정보를 수집할 수 있습니다. 분산 추적을 사용하여 최신 클라우드 네이티브, 마이크로서비스 기반 애플리케이션의 구성 요소 간 상호 작용을 모니터링, 네트워크 프로파일링 및 문제 해결할 수 있습니다.

분산 추적을 사용하면 다음 기능을 수행할 수 있습니다.

- 분산 트랜잭션 모니터링
- 성능 및 대기 시간 최적화
- 근본 원인 분석 수행

Red Hat OpenShift distributed tracing은 다음 두 가지 주요 구성 요소로 구성됩니다.

- **Red Hat OpenShift distributed tracing platform** - 이 구성 요소는 오픈 소스 Jaeger 프로젝트를 기반으로 합니다.
- **Red Hat OpenShift 분산 추적 데이터 수집** - 이 구성 요소는 오픈 소스 OpenTelemetry 프로젝트를 기반으로 합니다.

이러한 두 구성 요소는 모두 벤더 중립 OpenTracing API 및 계측을 기반으로 합니다.

1.2. 보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 Red Hat CTO Chris Wright의 메시지 에서 참조하십시오.

1.3. 지원 요청

이 문서에 설명된 절차 또는 일반적으로 OpenShift Container Platform에 문제가 발생하는 경우 Red Hat 고객 포털에 액세스하십시오. 고객 포털에서 다음을 수행할 수 있습니다.

- Red Hat 제품과 관련된 기사 및 솔루션에 대한 Red Hat 지식베이스를 검색하거나 살펴볼 수 있습니다.
- Red Hat 지원에 대한 지원 케이스 제출할 수 있습니다.
- 다른 제품 설명서에 액세스 가능합니다.

클러스터의 문제를 식별하려면 OpenShift Cluster Manager 에서 Insights를 사용할 수 있습니다. Insights는 문제에 대한 세부 정보 및 문제 해결 방법에 대한 정보를 제공합니다.

이 문서를 개선하기 위한 제안이 있거나 오류를 발견한 경우 가장 관련 있는 문서 구성 요소에 대한 Jira 문제를 제출하십시오. 섹션 이름 및 OpenShift Container Platform 버전과 같은 구체적인 정보를 제공합니다.

1.4. 새로운 기능 및 개선 사항

이 릴리스에는 다음 구성 요소 및 개념과 관련된 개선 사항이 추가되었습니다.

1.4.1. 새로운 기능 및 향상된 Red Hat OpenShift distributed tracing 2.7

이번 Red Hat OpenShift distributed tracing 릴리스는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정을 처리합니다.

1.4.1.1. Red Hat OpenShift distributed tracing 버전 2.7에서 지원되는 구성 요소 버전

Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.39
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.63.1

1.4.2. 새로운 기능 및 향상된 Red Hat OpenShift distributed tracing 2.6

이번 Red Hat OpenShift distributed tracing 릴리스는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정을 처리합니다.

1.4.2.1. Red Hat OpenShift distributed tracing 버전 2.6에서 지원되는 구성 요소 버전

Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.38
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.60

1.4.3. Red Hat OpenShift distributed tracing 2.5의 새로운 기능 및 개선 사항

이번 Red Hat OpenShift distributed tracing 릴리스는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정을 처리합니다.

이번 릴리스에서는 Red Hat OpenShift distributed tracing platform Operator에 대해 OTelemetry 프로토콜(OTLP)을 수집할 수 있는 지원이 추가되었습니다. 이제 Operator에서 OTLP 포트를 자동으로 활성화합니다.

- 포트 4317은 OTLP gRPC 프로토콜에 사용됩니다.
- 포트 4318은 OTLP HTTP 프로토콜에 사용됩니다.

이 릴리스에서는 Red Hat OpenShift distributed tracing data collection Operator에 Kubernetes 리소스 특성을 수집하는 기능도 추가되었습니다.

1.4.3.1. Red Hat OpenShift distributed tracing 버전 2.5에서 지원되는 구성 요소 버전

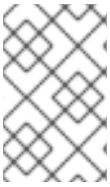
Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.36
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.56

1.4.4. 새로운 기능 및 향상된 Red Hat OpenShift distributed tracing 2.4

이번 Red Hat OpenShift distributed tracing 릴리스는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정을 처리합니다.

이 릴리스에서는 Red Hat Elasticsearch Operator를 사용하여 자동 프로비저닝 인증서도 지원합니다.

- 셀프 프로비저닝: Red Hat OpenShift distributed tracing platform Operator를 사용하여 설치 중에 Red Hat Elasticsearch Operator를 호출합니다. 이 릴리스에서는 자체 프로비저닝이 완전히 지원됩니다.
- 먼저 Elasticsearch 인스턴스 및 인증서를 생성한 다음 인증서를 사용하도록 분산 추적 플랫폼을 구성하는 것은 이번 릴리스의 기술 프리뷰입니다.



참고

Red Hat OpenShift distributed tracing 2.4로 업그레이드할 때 Operator는 Elasticsearch 인스턴스를 다시 생성하므로 5-10분 정도 걸릴 수 있습니다. 분산 추적이 중단되어 해당 기간 동안 사용할 수 없습니다.

1.4.4.1. Red Hat OpenShift distributed tracing 버전 2.4에서 지원되는 구성 요소 버전

Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.34.1
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.49

1.4.5. Red Hat OpenShift distributed tracing 2.3.1의 새로운 기능 및 개선 사항

이번 Red Hat OpenShift distributed tracing 릴리스는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정을 처리합니다.

1.4.5.1. Red Hat OpenShift distributed tracing 버전 2.3.1에서 지원되는 구성 요소 버전

Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.30.2
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.44.1-1

1.4.6. Red Hat OpenShift distributed tracing 2.3.0의 새로운 기능 및 개선 사항

이번 Red Hat OpenShift distributed tracing 릴리스는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정을 처리합니다.

이번 릴리스에서는 Red Hat OpenShift distributed tracing platform Operator가 기본적으로 **openshift-distributed-tracing** 네임스페이스에 설치됩니다. 이번 업데이트 이전에는 기본 설치가 **openshift-operators** 네임스페이스에 있었습니다.

1.4.6.1. Red Hat OpenShift distributed tracing version 2.3.0에서 지원되는 구성 요소 버전

Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.30.1
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.44.0

1.4.7. Red Hat OpenShift distributed tracing 2.2.0 새 기능 및 개선 사항

이번 Red Hat OpenShift distributed tracing 릴리스는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정을 처리합니다.

1.4.7.1. Red Hat OpenShift distributed tracing version 2.2.0에서 지원되는 구성 요소 버전

Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.30.0
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.42.0

1.4.8. Red Hat OpenShift distributed tracing 2.1.0 새로운 기능 및 개선 사항

이번 Red Hat OpenShift distributed tracing 릴리스는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정을 처리합니다.

1.4.8.1. Red Hat OpenShift distributed tracing 버전 2.1.0에서 지원되는 구성 요소 버전

Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.29.1
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.41.1

1.4.9. Red Hat OpenShift distributed tracing 2.0.0 새로운 기능 및 개선 사항

이번 릴리스에서는 Red Hat OpenShift Jaeger의 재배포가 Red Hat OpenShift distributed tracing으로 표시됩니다. 이 릴리스는 다음과 같은 변경, 추가 및 개선 사항으로 구성됩니다.

- Red Hat OpenShift distributed tracing은 다음 두 가지 주요 구성 요소로 구성됩니다.
  - **Red Hat OpenShift distributed tracing platform** - 이 구성 요소는 오픈 소스 Jaeger 프로젝트를 기반으로 합니다.
  - **Red Hat OpenShift 분산 추적 데이터 수집** - 이 구성 요소는 오픈 소스 OpenTelemetry 프로젝트를 기반으로 합니다.
- Red Hat OpenShift distributed tracing platform Operator를 Jaeger 1.28로 업데이트합니다. 앞으로 Red Hat OpenShift distributed tracing은 **stable** Operator 채널만 지원합니다. 개별 릴리스에 대한 채널은 더 이상 지원되지 않습니다.
- OpenTelemetry 0.33을 기반으로 하는 새로운 Red Hat OpenShift 분산 추적 데이터 수집 Operator를 소개합니다. 이 Operator는 기술 프리뷰 기능입니다.
- OpenTelemetry 프로토콜(OTLP) 지원을 쿼리 서비스에 추가합니다.
- OpenShift OperatorHub에 표시되는 새로운 분산 추적 아이콘을 도입합니다.
- 이름 변경 및 새 기능을 지원하는 문서의 롤링 업데이트가 포함되어 있습니다.

이 릴리스에서는 CVE(Common Vulnerabilities and Exposures) 및 버그 수정도 다룹니다.

1.4.9.1. Red Hat OpenShift distributed tracing 버전 2.0.0에서 지원되는 구성 요소 버전

Operator	구성 요소	버전
Red Hat OpenShift distributed tracing platform	Jaeger	1.28.0
Red Hat OpenShift distributed tracing 데이터 수집	OpenTelemetry	0.33.0

1.5. RED HAT OPENSHIFT DISTRIBUTED TRACING TECHNOLOGY PREVIEW

중요

Technology Preview 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

1.5.1. Red Hat OpenShift distributed tracing 2.4.0 Technology Preview

이 릴리스에서는 Red Hat Elasticsearch Operator를 사용하여 자동 프로비저닝 인증서도 지원합니다.

- 셀프 프로비저닝: Red Hat OpenShift distributed tracing platform Operator를 사용하여 설치 중에 Red Hat Elasticsearch Operator를 호출합니다. 이 릴리스에서는 자체 프로비저닝이 완전히 지원됩니다.
- 먼저 Elasticsearch 인스턴스 및 인증서를 생성한 다음 인증서를 사용하도록 분산 추적 플랫폼을 구성하는 것은 이번 릴리스의 기술 프리뷰입니다.

1.5.2. Red Hat OpenShift distributed tracing 2.2.0 Technology Preview

2.1 릴리스에 포함된 지원되지 않는 OpenTelemetry 수집기 구성 요소가 제거되었습니다.

1.5.3. Red Hat OpenShift distributed tracing 2.1.0 기술 프리뷰

이 릴리스에서는 OpenTelemetry 사용자 정의 리소스 파일에서 인증서를 구성하는 방법에 대한 주요 변경 사항이 추가되었습니다. 새 버전의 **ca_file** 은 다음 예와 같이 사용자 정의 리소스의 **tls** 에서 이동합니다.

OpenTelemetry 버전 0.33에 대한 CA 파일 구성

```
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 버전 0.41.1에 대한 CA 파일 구성

```
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 기술 프리뷰

이번 릴리스에는 Red Hat OpenShift distributed tracing 데이터 수집 Operator를 사용하여 설치하는 Red Hat OpenShift distributed tracing 데이터 수집 데이터 수집이 추가되었습니다. Red Hat OpenShift 분산 추적 데이터 수집은 OpenTelemetry API 및 계측을 기반으로 합니다.

Red Hat OpenShift distributed tracing 데이터 컬렉션에는 OpenTelemetry Operator 및 수집기가 포함됩니다. 수집기는 OpenTelemetry 또는 Jaeger 프로토콜의 추적을 수신하고 Red Hat OpenShift 분산 추적에 추적 데이터를 보내는 데 사용할 수 있습니다. 현재 수집기의 다른 기능은 지원되지 않습니다.

OpenTelemetry Collector는 개발자가 벤더와 무관한 API로 코드를 계측하고 벤더 종속을 피하고 관찰 기능 툴링의 증가된 에코시스템을 가능하게 합니다.

1.6. RED HAT OPENSHIFT DISTRIBUTED TRACING 알려진 문제

Red Hat OpenShift distributed tracing에는 다음과 같은 제한 사항이 있습니다.

- Apache Spark가 지원되지 않습니다.
- AMQ/Kafka를 통한 스트리밍 배포는 IBM Z 및 IBM Power Systems에서 지원되지 않습니다.

다음은 Red Hat OpenShift distributed tracing의 알려진 문제입니다.

- OECDHEA-220 분산 추적 데이터 컬렉션을 사용하여 이미지를 가져오려고 하는 경우 이미지 가져오기가 실패하고 **이미지 오류 메시지를 가져오지 못했습니다**. 이 문제에 대한 해결방법이 없습니다.
- TRACING-2057 Kafka API가 Strimzi Kafka Operator 0.23.0을 지원하도록 **v1beta2**로 업데이트되었습니다. 그러나 이 API 버전은 AMQ Streams 1.6.3에서 지원되지 않습니다. 다음 환경의 경우 Jaeger 서비스가 업그레이드되지 않으며 새 Jaeger 서비스를 생성하거나 기존 Jaeger 서비스를 수정할 수 없습니다.
  - Jaeger Operator 채널: **1.17.x stable** 또는 **1.20.x stable**
  - AMQ Streams Operator 채널: **amq-streams-1.6.x**
    이 문제를 해결하려면 AMQ Streams Operator의 서브스크립션 채널을 **amq-streams-1.7.x** 또는 **stable**로 전환합니다.

1.7. RED HAT OPENSHIFT DISTRIBUTED TRACING 수정된 문제

- OSSM-1910 버전 2.6에서 발생한 문제로 인해 OpenShift Container Platform Service Mesh를 사용하여 TLS 연결을 설정할 수 없었습니다. 이번 업데이트에서는 OpenShift Container Platform Service Mesh 및 Istio에서 사용하는 규칙과 일치하도록 서비스 포트 이름을 변경하여 문제를 해결합니다.
- 이 업데이트 이전에는 기본 200m CPU 및 256Mi 메모리 리소스 제한이 분산 추적 데이터 수집이 대규모 클러스터에서 지속적으로 다시 시작될 수 있습니다. 이번 업데이트에서는 이러한 리소스 제한을 제거하여 문제를 해결합니다.
- OECDHEA-222 이번 업데이트 이전에는 OpenShift Container Platform 분산 추적 플랫폼에서 기간을 삭제할 수 있었습니다. 이 문제가 발생하지 않도록하기 위해 이 릴리스에서는 버전 종속성을 업데이트합니다.
- TRACING-2337 Jaeger는 다음과 유사한 Jaeger 로그에 반복적인 경고 메시지를 로깅하고 있습니다.

  ```
  {"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}

이 문제는 gRPC 포트가 아닌 쿼리 서비스의 HTTP(S) 포트만 노출하여 해결되었습니다.

- TRACING-2009 Jaeger Operator가 Strimzi Kafka Operator 0.23.0에 대한 지원을 포함하도록 업데이트되었습니다.
- TRACING-1907 애플리케이션 네임스페이스에서 구성 맵이 누락되어 Jaeger 에이전트 사이드카 삽입이 실패했습니다. 잘못된 **OwnerReference** 필드 설정으로 인해 구성 맵이 자동으로 삭제되었으며 결과적으로 애플리케이션 Pod가 "ContainerCreating" 단계를 통과하지 않았습니다. 잘못된 설정이 제거되었습니다.
- TRACING-1725 TRACING-1631에 대한 후속 조치입니다. 동일한 이름을 사용하지만 다른 네임스페이스 내에 Jaeger 프로덕션 인스턴스가 여러 개인 경우 Elasticsearch 인증서가 올바르게 조정되는지 확인하기 위한 추가 수정 사항입니다. BZ-1918920도 참조하십시오.
- TRACING-1631 동일한 이름을 사용하지만 다른 네임스페이스 내의 여러 Jaeger 프로덕션 인스턴스로, Elasticsearch 인증서 문제를 발생시킵니다. 여러 서비스 메시가 설치되면 모든 Jaeger Elasticsearch 인스턴스에 개별 시크릿 대신 동일한 Elasticsearch 시크릿이 있어 OpenShift Elasticsearch Operator가 모든 Elasticsearch 클러스터와 통신할 수 없습니다.
- TRACING-1300 Istio 사이드카를 사용할 때 에이전트와 수집기 간의 연결에 실패했습니다. Jaeger Operator 업데이트는 Jaeger 사이드카 에이전트와 Jaeger 수집기 간의 TLS 통신을 기본적으로 활성화했습니다.
- TRACING-1208 Jaeger UI에 액세스할 때 인증 "500 Internal Error"입니다. OAuth를 사용하여 UI를 인증할 때 oauth-proxy 사이드카가 **additionalTrustBundle**로 설치할 때 정의된 사용자 정의 CA 번들을 신뢰하지 않기 때문에 500 오류가 발생합니다.
- TRACING-1166 현재 연결이 끊긴 환경에서 Jaeger 스트리밍 전략을 사용할 수 없습니다. Kafka 클러스터가 프로비저닝되면 다음과 같은 오류가 발생합니다. **이미지 registry.redhat.io/amq7/amq-streams-kafka-24-rhel7@sha256:f9ceca004f1b7dccb3b82d9a8027961f9fe4104e0ed69752c0bdd8078b4a1076을 가져오지 못했습니다**.
- TRACING-809 Jaeger Ingester는 Kafka 2.3과 호환되지 않습니다. Jaeger Ingester의 두 개 이상의 인스턴스와 트래픽이 충분한 경우 로그에 지속적으로 리밸런싱 메시지를 생성합니다. 이는 Kafka 2.3.1에서 수정된 Kafka 2.3의 문제의 재발로 인해 발생합니다. 자세한 내용은 Jaegertracing-1819를 참조하십시오.
- BZ-1918920/LOG-1619 Elasticsearch Pod가 업데이트 후 자동으로 다시 시작되지 않습니다. 해결방법: Pod를 수동으로 다시 시작합니다.

2장. 분산 추적 아키텍처

2.1. 분산 추적 아키텍처

사용자가 애플리케이션에서 작업을 수행할 때마다 응답을 생성하기 위해 참여하도록 다양한 서비스를 필요로 할 수 있는 아키텍처에 의해 요청이 실행됩니다. Red Hat OpenShift distributed tracing을 사용하면 애플리케이션을 구성하는 다양한 마이크로 서비스를 통해 요청 경로를 기록하는 분산 추적을 수행할 수 있습니다.

*분산 추적*은 분산 트랜잭션에 있는 전체 이벤트 체인을 이해하기 위해 일반적으로 다양한 프로세스 또는 호스트에서 실행되는 다양한 작업 단위에 대한 정보를 결합하는 데 사용되는 기술입니다. 개발자는 분산 추적을 사용하여 대규모 마이크로 서비스 아키텍처에서 호출 흐름을 시각화할 수 있습니다. 직렬화, 병렬 처리 및 대기 시간 소스를 이해하는 데 유용합니다.

Red Hat OpenShift distributed tracing은 마이크로 서비스의 전체 스택에서 개별 요청 실행을 기록하고 이를 추적으로 제공합니다. *추적*은 시스템을 통한 데이터/실행 경로입니다. 엔드 투 엔드 추적은 하나 이상의 기간으로 구성됩니다.

*기간*은 Red Hat OpenShift distributed tracing에서 작업 이름, 작업 시작 시간, 기간, 잠재적으로 태그 및 로그가 있는 논리적 작업 단위를 나타냅니다. 기간은 중첩되어 인과 관계를 모델링하도록 주문될 수 있습니다.

2.1.1. 분산 추적 개요

서비스 소유자로 분산 추적을 사용하여 서비스 아키텍처에 대한 정보를 수집할 수 있습니다. 분산 추적을 사용하여 최신 클라우드 네이티브, 마이크로서비스 기반 애플리케이션의 구성 요소 간 상호 작용을 모니터링, 네트워크 프로파일링 및 문제 해결할 수 있습니다.

분산 추적을 사용하면 다음 기능을 수행할 수 있습니다.

- 분산 트랜잭션 모니터링
- 성능 및 대기 시간 최적화
- 근본 원인 분석 수행

Red Hat OpenShift distributed tracing은 다음 두 가지 주요 구성 요소로 구성됩니다.

- **Red Hat OpenShift distributed tracing platform** - 이 구성 요소는 오픈 소스 Jaeger 프로젝트를 기반으로 합니다.
- **Red Hat OpenShift 분산 추적 데이터 수집** - 이 구성 요소는 오픈 소스 OpenTelemetry 프로젝트를 기반으로 합니다.

이러한 두 구성 요소는 모두 벤더 중립 OpenTracing API 및 계측을 기반으로 합니다.

2.1.2. Red Hat OpenShift distributed tracing 기능

Red Hat OpenShift distributed tracing은 다음 기능을 제공합니다.

- Kiali와의 통합 - 올바르게 구성된 경우 Kiali 콘솔에서 분산 추적 데이터를 볼 수 있습니다.
- 높은 확장성 - 분산 추적 백엔드는 단일 장애 지점이 없고 비즈니스 요구에 따라 확장할 수 있도록 설계되었습니다.

- 분산 컨텍스트 전파 - 서로 다른 구성 요소의 데이터를 함께 연결하여 완전한 엔드 투 엔드 추적을 만들 수 있습니다.

- Zipkin과의 역호환성 - Red Hat OpenShift distributed tracing에는 Zipkin의 드롭인 교체로 사용할 수 있는 API가 있지만 Red Hat은 이 릴리스에서 Zipkin 호환성을 지원하지 않습니다.

2.1.3. Red Hat OpenShift distributed tracing 아키텍처

Red Hat OpenShift distributed tracing은 추적 데이터를 수집, 저장 및 표시하기 위해 함께 작동하는 여러 구성 요소로 구성됩니다.

- **Red Hat OpenShift distributed tracing platform** - 이 구성 요소는 오픈 소스 Jaeger 프로젝트를 기반으로 합니다.

  - **클라이언트** (Jaeger 클라이언트, Tracer, Reporter, 조정된 애플리케이션, 클라이언트 라이브러리)- 분산 추적 플랫폼 클라이언트는 OpenTracing API의 언어별 구현입니다. 수동으로 또는 이미 OpenTracing과 통합된 Camel(Fuse), Spring Boot(RHOAR), MicroProfile(RHOAR/T©tail), Wildfly(EAP) 등의 다양한 기존 오픈 소스 프레임워크를 사용하여 분산 추적에 대해 애플리케이션을 조정하는 데 사용할 수 있습니다.

  - **에이전트** (Jaeger 에이전트, Server Queue, Processor Workers) - 분산 추적 플랫폼 에이전트는 UDP(User Datagram Protocol)를 통해 전송되는 기간을 수신 대기하는 네트워크 데몬이며 수집기에 배치 및 전송합니다. 에이전트는 조정된 애플리케이션과 동일한 호스트에 배치되어야 합니다. 일반적으로 Kubernetes와 같은 컨테이너 환경에서 사이드카를 보유하여 수행됩니다.

  - **Jaeger Collector** (databindor, Queue, Workers) - Jaeger 수집기는 기간을 수신하여 처리를 위해 내부 대기열에 배치합니다. 이를 통해 Jaeger 수집기는 기간이 스토리지로 이동할 때까지 대기하지 않고 클라이언트/에이전트로 즉시 돌아갈 수 있습니다.

  - **스토리지**(데이터 저장소) - 수집기에는 영구 스토리지 백엔드가 필요합니다. Red Hat OpenShift 분산 추적 플랫폼에는 범위 스토리지를 위한 플러그형 메커니즘이 있습니다. 이 릴리스에서 지원되는 유일한 스토리지는 Elasticsearch입니다.

  - **쿼리**(쿼리 서비스) - 쿼리는 스토리지에서 추적을 검색하는 서비스입니다.

  - **Ingester** (Ingester Service) - Red Hat OpenShift distributed tracing은 수집기와 실제 Elasticsearch 백업 스토리지 간의 버퍼로 Apache Kafka를 사용할 수 있습니다. Ingester는 Kafka에서 데이터를 읽고 Elasticsearch 스토리지 백엔드에 쓰는 서비스입니다.

  - **Jaeger 콘솔** - Red Hat OpenShift 분산 추적 플랫폼 사용자 인터페이스를 통해 분산 추적 데이터를 시각화할 수 있습니다. 검색 페이지에서 추적을 찾고 개별 추적을 구성하는 기간의 세부 사항을 확인할 수 있습니다.

- **Red Hat OpenShift 분산 추적 데이터 수집** - 이 구성 요소는 오픈 소스 OpenTelemetry 프로젝트를 기반으로 합니다.

  - **OpenTelemetry Collector** - OpenTelemetry 수집기는 원격 분석 데이터를 수신, 처리 및 내보낼 수 있는 공급업체와 무관한 방법입니다. OpenTelemetry 수집기는 하나 이상의 오픈 소스 또는 상용 백엔드와 같은 Jaeger 및 Prometheus와 같은 오픈 소스 관찰 기능 데이터 형식을 지원합니다. 수집기는 기본 위치 계측 라이브러리로 Telemetry 데이터를 내보냅니다.

3장. 분산 추적 설치

3.1. 분산 추적 설치

다음 두 가지 방법 중 하나로 OpenShift Container Platform에 Red Hat OpenShift distributed tracing을 설치할 수 있습니다.

- Red Hat OpenShift distributed tracing은 Red Hat OpenShift Service Mesh의 일부로 설치할 수 있습니다. 분산 추적은 기본적으로 서비스 메시 설치에 포함됩니다. Red Hat OpenShift distributed tracing을 서비스 메시의 일부로 설치하려면 Red Hat Service Mesh 설치 지침을 따르십시오. 서비스 메시와 동일한 네임스페이스에 Red Hat OpenShift distributed tracing을 설치해야 합니다. 즉 **ServiceMeshControlPlane** 및 Red Hat OpenShift distributed tracing 리소스는 동일한 네임스페이스에 있어야 합니다.

- 서비스 메시를 설치하지 않으려면 Red Hat OpenShift distributed tracing Operator를 사용하여 자체적으로 분산 추적을 설치할 수 있습니다. 서비스 메시 없이 Red Hat OpenShift 분산 추적을 설치하려면 다음 지침을 사용하십시오.

3.1.1. 전제 조건

Red Hat OpenShift distributed tracing을 설치하려면 설치 활동을 검토하고 사전 요구 사항을 충족해야 합니다.

- Red Hat 계정에 유효한 OpenShift Container Platform 서브스크립션이 있어야 합니다. 서브스크립션이 없는 경우 영업 담당자에게 자세한 내용을 문의하십시오.
- OpenShift Container Platform 4.9 개요 를 검토합니다.
- OpenShift Container Platform 4.9를 설치합니다.
  - AWS에 OpenShift Container Platform 4.9 설치
  - 사용자 프로비저닝 AWS에 OpenShift Container Platform 4.9 설치
  - 베어 메탈에 OpenShift Container Platform 4.9 설치
  - vSphere에 OpenShift Container Platform 4.9 설치
- OpenShift Container Platform 버전과 일치하는**oc**(OpenShift CLI) 버전을 설치하고 경로에 추가합니다.
- **cluster-admin** 역할이 있는 계정.

3.1.2. Red Hat OpenShift distributed tracing 설치 개요

Red Hat OpenShift distributed tracing 설치 단계는 다음과 같습니다.

- 문서를 검토하고 배포 전략을 확인합니다.
- 배포 전략에 영구 스토리지가 필요한 경우 OperatorHub를 통해 OpenShift Elasticsearch Operator를 설치합니다.
- OperatorHub를 통해 Red Hat OpenShift distributed tracing platform Operator를 설치합니다.
- 사용자 정의 리소스 YAML 파일을 수정하여 배포 전략을 지원합니다.

- Red Hat OpenShift distributed tracing Platform 인스턴스를 OpenShift Container Platform 환경에 하나 이상 배포합니다.

3.1.3. OpenShift Elasticsearch Operator 설치

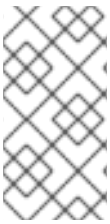
기본 Red Hat OpenShift distributed tracing 플랫폼 배포는 Red Hat OpenShift distributed tracing을 평가하거나 데모를 제공하거나 테스트 환경에서 Red Hat OpenShift distributed tracing 플랫폼을 사용하기 위해 빠르게 설치되도록 설계되었으므로 메모리 내 스토리지를 사용합니다. 프로덕션 환경에서 Red Hat OpenShift distributed tracing 플랫폼을 사용하려는 경우 영구 스토리지 옵션(이 경우 Elasticsearch)을 설치하고 구성해야 합니다.

전제 조건

- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.

주의

Operator의 커뮤니티 버전은 설치하지 마십시오. 커뮤니티 Operator는 지원되지 않습니다.

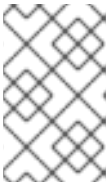


참고

이미 OpenShift Elasticsearch Operator를 OpenShift 로깅의 일부로 설치한 경우 OpenShift Elasticsearch Operator를 다시 설치할 필요가 없습니다. Red Hat OpenShift distributed tracing platform Operator는 설치된 OpenShift Elasticsearch Operator를 사용하여 Elasticsearch 인스턴스를 생성합니다.

프로세스

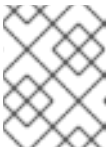
1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform 웹 콘솔에 로그인합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.
2. **Operators → OperatorHub**로 이동합니다.
3. **Elasticsearch**를 필터 상자에 입력하여 OpenShift Elasticsearch Operator를 찾습니다.
4. Red Hat에서 제공하는 **OpenShift Elasticsearch Operator**를 클릭하여 Operator에 대한 정보를 표시합니다.
5. **설치**를 클릭합니다.
6. **Operator 설치** 페이지에서 **stable** 업데이트 채널을 선택합니다. 이렇게 하면 새 버전이 릴리스되면 Operator가 자동으로 업데이트됩니다.
7. **클러스터의 기본 모든 네임스페이스(기본값)를** 수락합니다. 이렇게 하면 기본 **openshift-operators-redhat** 프로젝트에 Operator가 설치되고 클러스터의 모든 프로젝트에서 Operator를 사용할 수 있습니다.



참고

Elasticsearch 설치에는 OpenShift Elasticsearch Operator의 **openshift-operators-redhat** 네임스페이스가 필요합니다. 다른 Red Hat OpenShift distributed tracing Operator는 **openshift-operators** 네임스페이스에 설치됩니다.

- 기본 **자동** 승인 전략을 수락합니다. 기본적으로 이 Operator의 새 버전이 사용 가능하면 OLM(Operator Lifecycle Manager)은 개입 없이 Operator의 실행 중인 인스턴스를 자동으로 업그레이드합니다. **수동** 업데이트를 선택하면 최신 버전의 Operator가 사용 가능할 때 OLM이 업데이트 요청을 생성합니다. 클러스터 관리자는 Operator를 새 버전으로 업데이트하려면 OLM 업데이트 요청을 수동으로 승인해야 합니다.



참고

수동 승인 전략을 사용하려면 적절한 인증 정보를 가진 사용자가 Operator 설치 및 서브스크립션 프로세스를 승인해야 합니다.

8. **설치**를 클릭합니다.
9. **설치된 Operator** 페이지에서 **openshift-operators-redhat** 프로젝트를 선택합니다. 계속하기 전에 OpenShift Elasticsearch Operator에 "InstallSucceeded" 상태가 표시될 때까지 기다립니다.

3.1.4. Red Hat OpenShift distributed tracing Platform Operator 설치

Red Hat OpenShift distributed tracing Platform을 설치하려면 OperatorHub 를 사용하여 Red Hat OpenShift distributed tracing Platform Operator를 설치합니다.

기본적으로 Operator는 **openshift-operators** 프로젝트에 설치됩니다.

전제 조건

- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.
- 영구 스토리지가 필요한 경우 Red Hat OpenShift distributed tracing Platform Operator를 설치하기 전에 OpenShift Elasticsearch Operator도 설치해야 합니다.

주의

Operator의 커뮤니티 버전은 설치하지 마십시오. 커뮤니티 Operator는 지원되지 않습니다.

프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform 웹 콘솔에 로그인합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.
2. **Operators → OperatorHub**로 이동합니다.

3. **distributed tracing platform** 을 필터에 입력하여 Red Hat OpenShift distributed tracing platform Operator를 찾습니다.

4. Red Hat에서 제공하는 **Red Hat OpenShift distributed tracing platform Operator**를 클릭하여 Operator에 대한 정보를 표시합니다.

5. **설치**를 클릭합니다.

6. **Operator 설치** 페이지에서 **stable** 업데이트 채널을 선택합니다. 이렇게 하면 새 버전이 릴리스되면 Operator가 자동으로 업데이트됩니다.

7. **클러스터의 기본 모든 네임스페이스(기본값)**를 수락합니다. 이렇게 하면 기본 **openshift-operators** 프로젝트에서 Operator가 설치되고 클러스터의 모든 프로젝트에서 Operator를 사용할 수 있습니다.

    - 기본 **자동** 승인 전략을 수락합니다. 기본적으로 이 Operator의 새 버전이 사용 가능하면 OLM(Operator Lifecycle Manager)은 개입 없이 Operator의 실행 중인 인스턴스를 자동으로 업그레이드합니다. **수동** 업데이트를 선택하면 최신 버전의 Operator가 사용 가능할 때 OLM이 업데이트 요청을 생성합니다. 클러스터 관리자는 Operator를 새 버전으로 업데이트하려면 OLM 업데이트 요청을 수동으로 승인해야 합니다.

참고

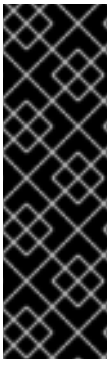
수동 승인 전략을 사용하려면 적절한 인증 정보를 가진 사용자가 Operator 설치 및 서브스크립션 프로세스를 승인해야 합니다.

8. **설치**를 클릭합니다.

9. **Operators → 설치된 Operator**로 이동합니다.

10. **설치된 Operator** 페이지에서 **openshift-operators** 프로젝트를 선택합니다. 계속하기 전에 Red Hat OpenShift distributed tracing Platform Operator에 "Succeeded" 상태가 표시될 때까지 기다립니다.

3.1.5. Red Hat OpenShift distributed tracing data collection Operator 설치



중요

Red Hat OpenShift distributed tracing data collection Operator는 기술 프리뷰 기능 전용입니다. Technology Preview 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

Red Hat OpenShift distributed tracing 데이터 수집을 설치하려면 OperatorHub 를 사용하여 Red Hat OpenShift distributed tracing 데이터 수집 Operator를 설치합니다.

기본적으로 Operator는 **openshift-operators** 프로젝트에 설치됩니다.

전제 조건

- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.

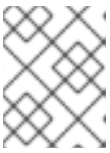
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.

주의

Operator의 커뮤니티 버전은 설치하지 마십시오. 커뮤니티 Operator는 지원되지 않습니다.

프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform 웹 콘솔에 로그인합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.
2. **Operators → OperatorHub**로 이동합니다.
3. **distributed tracing data collection**을 필터에 입력하여 Red Hat OpenShift distributed tracing data collection Operator를 찾습니다.
4. **Red Hat에서 제공하는 Red Hat OpenShift 분산 추적 데이터 수집 Operator**를 클릭하여 Operator에 대한 정보를 표시합니다.
5. **설치**를 클릭합니다.
6. **Operator 설치** 페이지에서 기본 **stable** Update 채널을 수락합니다. 이렇게 하면 새 버전이 릴리스되면 Operator가 자동으로 업데이트됩니다.
7. **클러스터의 기본 모든 네임스페이스(기본값)를** 수락합니다. 이렇게 하면 기본 **openshift-operators** 프로젝트에서 Operator가 설치되고 클러스터의 모든 프로젝트에서 Operator를 사용할 수 있습니다.
8. 기본 **자동** 승인 전략을 수락합니다. 기본적으로 이 Operator의 새 버전이 사용 가능하면 OLM(Operator Lifecycle Manager)은 개입 없이 Operator의 실행 중인 인스턴스를 자동으로 업그레이드합니다. **수동** 업데이트를 선택하면 최신 버전의 Operator가 사용 가능할 때 OLM이 업데이트 요청을 생성합니다. 클러스터 관리자는 Operator를 새 버전으로 업데이트하려면 OLM 업데이트 요청을 수동으로 승인해야 합니다.



참고

수동 승인 전략을 사용하려면 적절한 인증 정보를 가진 사용자가 Operator 설치 및 서브스크립션 프로세스를 승인해야 합니다.

9. **설치**를 클릭합니다.
10. **Operators → 설치된 Operator**로 이동합니다.
11. **설치된 Operator** 페이지에서 **openshift-operators** 프로젝트를 선택합니다. 계속하기 전에 Red Hat OpenShift distributed tracing data collection Operator에 "Succeeded" 상태가 표시될 때까지 기다립니다.

3.2. 분산 추적 구성 및 배포

Red Hat OpenShift distributed tracing platform Operator는 분산 추적 플랫폼 리소스를 생성하고 배포할 때 사용할 아키텍처 및 구성 설정을 정의하는 CRD(사용자 정의 리소스 정의) 파일을 사용합니다. 기본 구성을 설치하거나 비즈니스 요구 사항에 맞게 파일을 수정할 수 있습니다.

Red Hat OpenShift distributed tracing 플랫폼에는 사전 정의된 배포 전략이 있습니다. 사용자 정의 리소스 파일에 배포 전략을 지정합니다. 분산 추적 플랫폼 인스턴스를 생성할 때 Operator는 이 구성 파일을 사용하여 배포에 필요한 오브젝트를 생성합니다.

배포 전략을 표시하는 Jaeger 사용자 정의 리소스 파일

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: MyConfigFile
spec:
  strategy: production 1
```

1 Red Hat OpenShift distributed tracing platform Operator는 현재 다음과 같은 배포 전략을 지원합니다.

- **allInOne**(기본값) - 이 전략은 개발, 테스트 및 데모 목적으로 설계되었습니다. 이는 프로덕션 사용을 목적으로 하지 않습니다. 기본 백엔드 구성 요소, 에이전트, 수집기 및 쿼리 서비스는 모두 기본적으로 메모리 내 스토리지를 사용하도록 구성된 단일 실행 파일로 패키지됩니다.

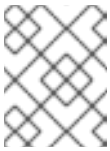
  

  참고

  메모리 내 스토리지는 영구적이지 않습니다. 즉, 분산 추적 플랫폼 인스턴스가 종료, 재시작 또는 교체되면 추적 데이터가 손실됩니다. 각 Pod에 자체 메모리가 있으므로 메모리 내 스토리지를 확장할 수 없습니다. 영구 스토리지의 경우 Elasticsearch를 기본 스토리지로 사용하는 **production** 또는 **streaming** 전략을 사용해야 합니다.

- **프로덕션** - 프로덕션 전략은 장기적인 추적 데이터 저장과 더 확장 가능하고 가용성이 높은 아키텍처가 필요한 프로덕션 환경을 위한 것입니다. 따라서 각 백엔드 구성 요소는 별도로 배포됩니다. 에이전트는 조정된 애플리케이션에서 사이드카로 삽입될 수 있습니다. 쿼리 및 수집기 서비스는 지원되는 스토리지 유형(현재 Elasticsearch)으로 구성됩니다. 이러한 각 구성 요소의 여러 인스턴스는 성능 및 복원에 필요한 대로 프로비저닝할 수 있습니다.

- **스트리밍** - 스트리밍 전략은 수집기와 Elasticsearch 백엔드 스토리지 간에 효과적으로 적용되는 스트리밍 기능을 제공하여 프로덕션 전략을 보강하도록 설계되었습니다. 이를 통해 높은 로드 상황에서 백엔드 스토리지의 부담을 줄이고 다른 추적 후 처리 기능을 통해 스트리밍 플랫폼(AMQ Streams/ Kafka)에서 직접 실시간 데이터를 가져올 수 있습니다.

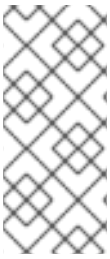
  

  참고

  스트리밍 전략에는 AMQ Streams에 대한 추가 Red Hat 서브스크립션이 필요합니다.

참고

streaming 배포 전략은 현재 IBM Z에서 지원되지 않습니다.



참고

Red Hat OpenShift distributed tracing을 서비스 메시의 일부로 설치 및 사용하거나 독립형 구성 요소로 사용하는 방법은 다음 두 가지가 있습니다. Red Hat OpenShift Service Mesh의 일부로 분산 추적을 설치한 경우 ServiceMeshControlPlane 의 일부로 기본 구성을 수행할 수 있지만 완전히 제어하려면 Jaeger CR을 구성하고 ServiceMeshControlPlane 에서 분산 추적 구성 파일을 참조해야 합니다.

3.2.1. 웹 콘솔에서 분산 추적 기본 전략 배포

CRD(사용자 정의 리소스 정의)는 Red Hat OpenShift 분산 추적 인스턴스를 배포할 때 사용되는 구성을 정의합니다. 기본 CR의 이름은 **jaeger-all-in-one-inmemory** 로 지정되며 기본 OpenShift Container Platform 설치에 성공적으로 설치할 수 있도록 최소한의 리소스로 구성됩니다. 이 기본 구성을 사용하여 **AllInOne** 배포 전략을 사용하는 Red Hat OpenShift 분산 추적 플랫폼 인스턴스를 생성하거나 사용자 정의 리소스 파일을 정의할 수 있습니다.

참고

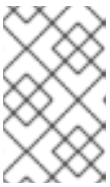
메모리 내 스토리지는 영구적이지 않습니다. Jaeger Pod가 종료, 재시작 또는 교체되면 추적 데이터가 손실됩니다. 영구 스토리지의 경우 Elasticsearch를 기본 스토리지로 사용하는 **production** 또는 **streaming** 전략을 사용해야 합니다.

전제 조건

- Red Hat OpenShift distributed tracing platform Operator가 설치되어 있습니다.
- 배포를 사용자 지정하는 방법에 대한 지침을 검토했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform 웹 콘솔에 로그인합니다.
2. 새 프로젝트(예: **tracing-system**)를 생성합니다.

   참고

   Service Mesh의 일부로 설치하는 경우 분산 추적 리소스를 **ServiceMeshControlPlane** 리소스와 동일한 네임스페이스에 설치해야 합니다 (예: **istio-system**).

   a. **홈 → 프로젝트**로 이동합니다.

   b. **Create Project**를 클릭합니다.

   c. **Name** 필드에 **tracing-system** 을 입력합니다.

   d. **Create**를 클릭합니다.
3. **Operators → 설치된 Operator**로 이동합니다.
4. 필요한 경우 **프로젝트** 메뉴에서 **추적 시스템을** 선택합니다. Operator가 새 프로젝트에 복사될 때까지 몇 분 정도 기다려야 할 수 있습니다.

5. Red Hat OpenShift distributed tracing platform Operator를 클릭합니다. **세부 정보** 탭의 **제공된 API** 에서 Operator는 단일 링크를 제공합니다.

6. **Jaeger** 에서 **인스턴스 생성을** 클릭합니다.

7. **Jaeger 생성** 페이지에서 기본값을 사용하여 설치하려면 **생성을 클릭하여 분산 추적 플랫폼 인스턴스를 생성합니다**.

8. **Jaegers** 페이지에서 분산 추적 플랫폼 인스턴스의 이름을 클릭합니다(예: **jaeger-all-one-inmemory**).

9. **Jaeger 세부 정보** 페이지에서 **리소스** 탭을 클릭합니다. 계속하기 전에 Pod 상태가 "실행 중"이 될 때까지 기다립니다.

3.2.1.1. CLI에서 분산 추적 기본 전략 배포

명령줄에서 분산 추적 플랫폼의 인스턴스를 생성하려면 다음 절차를 따르십시오.

전제 조건

- Red Hat OpenShift distributed tracing platform Operator가 설치 및 검증되었습니다.
- 배포를 사용자 지정하는 방법에 대한 지침을 검토했습니다.
- OpenShift Container Platform 버전과 일치하는 OpenShift CLI(**oc**)에 액세스할 수 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.

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

2. **tracing-system** 이라는 새 프로젝트를 생성합니다.

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

3. 다음 텍스트가 포함된 **jaeger.yaml**이라는 사용자 정의 리소스 파일을 생성합니다.

예: jaeger-all-one.yaml

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

4. 다음 명령을 실행하여 분산 추적 플랫폼을 배포합니다.

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

5. 다음 명령을 실행하여 설치 프로세스 중에 Pod의 진행 상황을 확인합니다.

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

설치 프로세스가 완료되면 다음 예와 유사한 출력이 표시됩니다.

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

3.2.2. 웹 콘솔에서 분산 추적 프로덕션 전략 배포

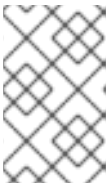
프로덕션 배포 전략은 보다 확장 가능하고 가용성이 높은 아키텍처가 필요한 프로덕션 환경을 위한 것이며 추적 데이터의 장기 스토리지가 중요합니다.

전제 조건

- OpenShift Elasticsearch Operator가 설치되었습니다.
- Red Hat OpenShift distributed tracing platform Operator가 설치되어 있습니다.
- 배포를 사용자 지정하는 방법에 대한 지침을 검토했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform 웹 콘솔에 로그인합니다.
2. 새 프로젝트(예: **tracing-system**)를 생성합니다.



참고

Service Mesh의 일부로 설치하는 경우 분산 추적 리소스를 **ServiceMeshControlPlane** 리소스와 동일한 네임스페이스에 설치해야 합니다 (예: **istio-system**).

   a. **홈 → 프로젝트**로 이동합니다.
   b. **Create Project**를 클릭합니다.
   c. **Name** 필드에 **tracing-system** 을 입력합니다.
   d. **Create**를 클릭합니다.

3. **Operators → 설치된 Operator**로 이동합니다.
4. 필요한 경우 **프로젝트** 메뉴에서 **추적 시스템을** 선택합니다. Operator가 새 프로젝트에 복사될 때까지 몇 분 정도 기다려야 할 수 있습니다.
5. Red Hat OpenShift distributed tracing platform Operator를 클릭합니다. **개요** 탭의 **제공된 API**에서 Operator는 단일 링크를 제공합니다.
6. **Jaeger** 에서 **인스턴스 생성을** 클릭합니다.
7. **Jaeger 생성** 페이지에서 기본 **all-in-one** YAML 텍스트를 프로덕션 YAML 구성으로 교체합니다. 예를 들면 다음과 같습니다.

예 jaeger-production.yaml 파일을 Elasticsearch로

■

```
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. **만들기를** 클릭하여 분산 추적 플랫폼 인스턴스를 만듭니다.
9. **Jaegers** 페이지에서 분산 추적 플랫폼 인스턴스의 이름을 클릭합니다(예: **jaeger-prod-elasticsearch**).
10. **Jaeger 세부 정보** 페이지에서 **리소스** 탭을 클릭합니다. 계속하기 전에 모든 Pod 상태가 "실행 중"이 될 때까지 기다립니다.

3.2.2.1. CLI에서 분산 추적 프로덕션 전략 배포

명령줄에서 분산 추적 플랫폼의 인스턴스를 생성하려면 다음 절차를 따르십시오.

전제 조건

- OpenShift Elasticsearch Operator가 설치되었습니다.
- Red Hat OpenShift distributed tracing platform Operator가 설치되어 있습니다.
- 배포를 사용자 지정하는 방법에 대한 지침을 검토했습니다.
- OpenShift Container Platform 버전과 일치하는 OpenShift CLI(**oc**)에 액세스할 수 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.

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

2. **tracing-system** 이라는 새 프로젝트를 생성합니다.

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

3. 이전 프로세스의 예제 텍스트가 포함된 **jaeger-production.yaml**이라는 사용자 정의 리소스 파일을 생성합니다.

4. 다음 명령을 실행하여 분산 추적 플랫폼을 배포합니다.

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

5. 다음 명령을 실행하여 설치 프로세스 중에 Pod의 진행 상황을 확인합니다.

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

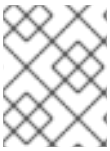
설치 프로세스가 완료되면 다음 예와 유사한 출력이 표시됩니다.

```
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. 웹 콘솔에서 분산 추적 스트리밍 전략 배포

streaming 배포 전략은 보다 확장 가능하고 가용성이 높은 아키텍처가 필요한 프로덕션 환경을 위해 고안되었으며 추적 데이터의 장기 스토리지가 중요합니다.

streaming 전략은 수집기와 Elasticsearch 스토리지 사이에 있는 스트리밍 기능을 제공합니다. 이렇게 하면 높은 로드 상황에서 스토리지의 부담을 줄이고 다른 추적 후 처리 기능을 통해 Kafka 스트리밍 플랫폼에서 직접 실시간 데이터를 가져올 수 있습니다.



참고

스트리밍 전략에는 AMQ Streams에 대한 추가 Red Hat 서브스크립션이 필요합니다. AMQ Streams 서브스크립션이 없는 경우 영업 담당자에게 자세한 내용을 문의하십시오.

참고

streaming 배포 전략은 현재 IBM Z에서 지원되지 않습니다.

전제 조건

- AMQ Streams Operator가 설치되었습니다. 버전 1.4.0 이상을 사용하는 경우 자체 프로비저닝을 사용할 수 있습니다. 그러지 않으면 Kafka 인스턴스를 생성해야 합니다.
- Red Hat OpenShift distributed tracing platform Operator가 설치되어 있습니다.
- 배포를 사용자 지정하는 방법에 대한 지침을 검토했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform 웹 콘솔에 로그인합니다.
2. 새 프로젝트(예: **tracing-system**)를 생성합니다.

참고

Service Mesh의 일부로 설치하는 경우 분산 추적 리소스를 **ServiceMeshControlPlane** 리소스와 동일한 네임스페이스에 설치해야 합니다 (예: **istio-system**).

   a. **홈 → 프로젝트**로 이동합니다.
   b. **Create Project**를 클릭합니다.
   c. **Name** 필드에 **tracing-system** 을 입력합니다.
   d. **Create**를 클릭합니다.

3. **Operators → 설치된 Operator**로 이동합니다.
4. 필요한 경우 **프로젝트** 메뉴에서 **추적 시스템을** 선택합니다. Operator가 새 프로젝트에 복사될 때까지 몇 분 정도 기다려야 할 수 있습니다.
5. Red Hat OpenShift distributed tracing platform Operator를 클릭합니다. **개요** 탭의 **제공된 API**에서 Operator는 단일 링크를 제공합니다.
6. **Jaeger** 에서 **인스턴스 생성을** 클릭합니다.
7. **Jaeger 생성** 페이지에서 기본 **all-in-one** YAML 텍스트를 스트리밍 YAML 구성으로 교체합니다. 예를 들면 다음과 같습니다.

jaeger-streaming.yaml 파일 예

```
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. **만들기를** 클릭하여 분산 추적 플랫폼 인스턴스를 만듭니다.
2. **Jaegers** 페이지에서 분산 추적 플랫폼 인스턴스의 이름을 클릭합니다(예: **jaeger-streaming**).
3. **Jaeger 세부 정보** 페이지에서 **리소스** 탭을 클릭합니다. 계속하기 전에 모든 Pod 상태가 "실행 중"이 될 때까지 기다립니다.

3.2.3.1. CLI에서 분산 추적 스트리밍 전략 배포

명령줄에서 분산 추적 플랫폼의 인스턴스를 생성하려면 다음 절차를 따르십시오.

전제 조건

- AMQ Streams Operator가 설치되었습니다. 버전 1.4.0 이상을 사용하는 경우 자체 프로비저닝을 사용할 수 있습니다. 그러지 않으면 Kafka 인스턴스를 생성해야 합니다.
- Red Hat OpenShift distributed tracing platform Operator가 설치되어 있습니다.
- 배포를 사용자 지정하는 방법에 대한 지침을 검토했습니다.
- OpenShift Container Platform 버전과 일치하는 OpenShift CLI(**oc**)에 액세스할 수 있습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.

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

2. **tracing-system** 이라는 새 프로젝트를 생성합니다.

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

3. 이전 프로세스의 예제 텍스트가 포함된 **jaeger-streaming.yaml**이라는 사용자 정의 리소스 파일을 생성합니다.
4. 다음 명령을 실행하여 Jaeger를 배포합니다.

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

5. 다음 명령을 실행하여 설치 프로세스 중에 Pod의 진행 상황을 확인합니다.

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

설치 프로세스가 완료되면 다음 예와 유사한 출력이 표시됩니다.

```
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. 배포 검증

3.2.4.1. Jaeger 콘솔에 액세스

Jaeger 콘솔에 액세스하려면 Red Hat OpenShift Service Mesh 또는 Red Hat OpenShift distributed tracing이 설치되어 있고 Red Hat OpenShift distributed tracing Platform이 설치되어 있어야 합니다.

설치 프로세스는 Jaeger 콘솔에 액세스하기 위한 경로를 생성합니다.

Jaeger 콘솔의 URL을 알고 있으면 직접 액세스할 수 있습니다. URL을 모르는 경우 다음 지침을 사용하십시오.

OpenShift 콘솔의 프로세스

1. OpenShift Container Platform 웹 콘솔에 cluster-admin 권한이 있는 사용자로 로그인합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.
2. **네트워킹 → 경로로** 이동합니다.
3. **경로** 페이지의 **네임스페이스 메뉴에서 컨트롤 플레인 프로젝트 (예:tracing-system)**를 선택합니다.
   Location 열에는 각 경로에 대한 연결된 주소가 표시됩니다.
4. 필요한 경우 필터를 사용하여 **jaeger** 경로를 찾습니다. 경로 **위치를** 클릭하여 콘솔을 시작합니다.
5. **OpenShift와 함께 로그인을** 클릭합니다.

CLI의 프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.

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

2. 명령줄을 사용하여 경로 세부 정보를 쿼리하려면 다음 명령을 입력합니다. 이 예제에서 **tracing-system** 은 컨트롤 플레인 네임스페이스입니다.

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

3. 브라우저를 시작하고 **https://<JAEGER_URL** > 으로 이동합니다. 여기서 < **JAEGER_URL** >은 이전 단계에서 검색한 경로입니다.
4. OpenShift Container Platform 콘솔에 액세스하는 데 사용하는 것과 동일한 사용자 이름 및 암호를 사용하여 로그인합니다.

5. 서비스 메시에 서비스를 추가하고 추적을 생성한 경우 필터를 사용하여 추적 데이터를 검색할 수 있습니다.
콘솔 설치를 검증하는 경우 표시할 추적 데이터가 없습니다.

3.2.5. 배포 사용자 정의

3.2.5.1. 배포 모범 사례

- Red Hat OpenShift distributed tracing 인스턴스 이름은 고유해야 합니다. Red Hat OpenShift distributed tracing 플랫폼 인스턴스가 여러 개 있고 사이드카 삽입 에이전트를 사용하려면 Red Hat OpenShift distributed tracing Platform 인스턴스에 고유한 이름이 있어야 하며 삽입 주석에서 추적 데이터가 보고되어야 하는 Red Hat OpenShift 분산 추적 플랫폼 인스턴스 이름을 명시적으로 지정해야 합니다.
- 다중 테넌트 구현과 테넌트가 네임스페이스로 구분된 경우 Red Hat OpenShift distributed tracing Platform 인스턴스를 각 테넌트 네임스페이스에 배포합니다.
    - 다중 테넌트 설치 또는 Red Hat OpenShift Dedicated에서는 데몬 세트로 에이전트가 지원되지 않습니다. 사이드카로서의 에이전트는 이러한 사용 사례에 대해 지원되는 유일한 구성입니다.
- Red Hat OpenShift Service Mesh의 일부로 분산 추적을 설치하는 경우 분산 추적 리소스를 **ServiceMeshControlPlane** 리소스와 동일한 네임스페이스에 설치해야 합니다.

영구 스토리지 구성에 대한 자세한 내용은 영구 스토리지 이해 및 선택한 스토리지 옵션에 대한 적절한 구성 항목을 참조하십시오.

3.2.5.2. 분산 추적 기본 구성 옵션

Jaeger CR(사용자 정의 리소스)은 분산 추적 플랫폼 리소스를 생성할 때 사용할 아키텍처 및 설정을 정의합니다. 이러한 매개변수를 수정하여 분산 추적 플랫폼 구현을 비즈니스 요구 사항에 맞게 사용자 지정할 수 있습니다.

Jaeger 일반 YAML 예

```
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: {}
```

표 3.1. Jaeger 매개변수

매개변수	설명	값	기본값
apiVersion:		오브젝트를 생성할 때 사용할 API 버전입니다.	**jaegertracing.io/v1**
jaegertracing.io/v1	**kind:**	생성할 Kubernetes 오브젝트를 정의합니다.	**jaeger**
	metadata:	**name** 문자열, **UID** 및 선택적 **namespace** 를 포함하여 오브젝트를 고유하게 식별할 수 있는 데이터입니다.	
OpenShift Container Platform은 **UID**를 자동으로 생성하고 오브젝트가 생성된 프로젝트의 이름으로 **namespace**를 완료합니다.	**name:**	개체의 이름입니다.	분산 추적 플랫폼 인스턴스의 이름입니다.
jaeger-all-in-one-inmemory	**spec:**	생성할 오브젝트의 사양입니다.	분산 추적 플랫폼 인스턴스에 대한 모든 구성 매개변수를 포함합니다. 모든 Jaeger 구성 요소에 대한 공통 정의가 필요한 경우 **사양** 노드 아래에 정의됩니다. 정의가 개별 구성 요소와 관련된 경우 **spec/**<component> 노드 아래에 배치됩니다.
해당 없음	**strategy:**	Jaeger 배포 전략	**allInOne**, **production** 또는 **streaming**
allInOne	**allInOne:**	**allInOne** 이미지가 에이전트, 수집기, 쿼리, Ingester 및 Jaeger UI를 단일 Pod에 배포하므로 이 배포에 대한 구성은 **allInOne** 매개변수 아래에 구성 요소를 중첩해야 합니다.	

매개변수	설명	값	기본값
	agent:	에이전트를 정의하는 구성 옵션입니다.	
	collector:	Jaeger 수집기를 정의하는 구성 옵션입니다.	
	sampling:	추적을 위한 샘플링 전략을 정의하는 구성 옵션입니다.	
	storage:	스토리지를 정의하는 구성 옵션입니다. 모든 스토리지 관련 옵션은 **allInOne** 또는 기타 구성 요소 옵션에 있지 않고 **스토리지** 아래에 배치되어야 합니다.	
	query:	쿼리 서비스를 정의하는 구성 옵션입니다.	
	ingester:	Ingester 서비스를 정의하는 구성 옵션입니다.	

다음 예제 YAML은 기본 설정을 사용하여 Red Hat OpenShift 분산 추적 플랫폼 배포를 생성하는 데 필요한 최소 크기입니다.

최소 필수 dist-tracing-all-in-one.yaml의 예

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

3.2.5.3. Jaeger 수집기 구성 옵션

Jaeger 수집기는 추적기에서 캡처한 기간을 수신하고 프로덕션 전략을 사용할 때 영구 Elasticsearch 스토리지에 기록하거나 **streaming** 전략을 사용할 때 AMQ Streams에 기록하는 구성 요소입니다.

수집기는 상태 비저장이므로 Jaeger 수집기의 많은 인스턴스가 병렬로 실행될 수 있습니다. 수집기는 Elasticsearch 클러스터의 위치를 제외하고 거의 구성이 필요하지 않습니다.

표 3.2. Operator에서 Jaeger Collector를 정의하는 데 사용하는 매개변수

매개변수	설명	값
collector: replicas:	생성할 수집기 복제본 수를 지정합니다.	정수(예: **5**)

표 3.3. 수집기에 전달된 구성 매개변수

매개변수	설명	값
spec: collector: options: {}	Jaeger 수집기를 정의하는 구성 옵션입니다.	
options: collector: num-workers:	큐에서 가져온 작업자 수입니다.	정수(예: **50**)
options: collector: queue-size:	수집기 큐의 크기입니다.	정수(예: **2000**)
options: kafka: producer: topic: jaeger-spans	**topic** 매개변수는 수집기에서 메시지를 생성하기 위해 사용하는 Kafka 구성을 식별하고 메시지를 사용하는 Ingester를 식별합니다.	생산자의 레이블입니다.
options: kafka: producer: brokers: my-cluster-kafka-brokers.kafka:9092	메시지를 생성하기 위해 수집기에서 사용하는 Kafka 구성을 식별합니다. 브로커를 지정하지 않고 AMQ Streams 1.4.0 이상이 설치되어 있는 경우 Red Hat OpenShift distributed tracing Platform Operator가 Kafka를 자체적으로 프로비저닝합니다.	
options: log-level:	수집기의 로깅 수준입니다.	가능한 값: **debug**,**info**,**warn**,**error**,**fatal**,**panic**.

3.2.5.4. 분산 추적 샘플링 구성 옵션

Red Hat OpenShift distributed tracing platform Operator는 원격 샘플러를 사용하도록 구성된 추적기에 제공될 샘플링 전략을 정의하는 데 사용할 수 있습니다.

모든 추적이 생성되지만 소수만 샘플링됩니다. 추적 샘플링은 추가 처리 및 스토리지의 추적을 나타냅니다.

참고

샘플링 결정이 내려지기 때문에 Envoy 프록시에서 추적을 시작한 경우와 관련이 없습니다. Jaeger 샘플링 결정은 클라이언트를 사용하여 애플리케이션에서 추적을 시작할 때만 관련이 있습니다.

서비스에서 추적 컨텍스트가 없는 요청을 수신하면 클라이언트는 새 추적을 시작하고 임의의 추적 ID를 할당하고 현재 설치된 샘플링 전략에 따라 샘플링 결정을 내립니다. 샘플링 결정은 추적의 모든 후속 요청으로 전파되어 다른 서비스가 샘플링 결정을 다시 생성하지 않도록 합니다.

분산 추적 플랫폼 라이브러리는 다음 샘플을 지원합니다.

- 확률론 - 샘플러는 샘플링(**sampling.param**) 속성의 값과 동일한 샘플링의 확률로 임의 샘플링 결정을 내립니다. 예를 들어 **sampling.param=0.1** 을 사용하면 추적 10개에서 약 1개씩 샘플링됩니다.
- 속도 제한 - 샘플러는 누수된 버킷 속도 제한기를 사용하여 추적을 특정한 일정 속도로 샘플링합니다. 예를 들어, **sampling.param=2.0** 을 사용하면 초당 2개의 추적 속도가 있는 요청을 샘플링합니다.

표 3.4. Jaeger 샘플링 옵션

매개변수	설명	값	기본값
spec: sampling: options: {} default_strategy: service_strategy:	추적을 위한 샘플링 전략을 정의하는 구성 옵션입니다.		구성을 제공하지 않으면 수집기는 모든 서비스에 대해 0.001(0.1%) 확률로 기본 확률 샘플링 정책을 반환합니다.
default_strategy: type: service_strategy: type:	사용할 샘플링 전략입니다. 위의 설명을 참조하십시오.	유효한 값은 **probabilistic** 및 **ratelimiting**입니다.	**probabilistic**
default_strategy: param: service_strategy: param:	선택한 샘플링 전략에 대한 매개변수입니다.	10진수 및 정수 값(0, .1, 1, 10)	1

이 예에서는 추적 인스턴스가 샘플링될 가능성이 50%인 비율로 확률적인 기본 샘플링 전략을 정의합니다.

확률 샘플링 예

```
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
```

사용자 제공 구성이 없는 경우 분산 추적 플랫폼은 다음 설정을 사용합니다.

기본 샘플링

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

3.2.5.5. 분산 추적 스토리지 구성 옵션

spec:storage에서 Collector, Ingester 및 쿼리 서비스에 대한 스토리지를 구성합니다. 이러한 각 구성 요소의 여러 인스턴스는 성능 및 복원에 필요한 대로 프로비저닝할 수 있습니다.

표 3.5. Red Hat OpenShift distributed tracing platform Operator에서 분산 추적 스토리지를 정의하는 데 사용하는 일반 스토리지 매개변수

매개변수	설명	값	기본값

`spec:`
`  storage:`
`  type:`
	배포에 사용할 스토리지 유형입니다.	**memory** 또는 **elasticsearch**. 메모리 스토리지는 Pod가 종료되면 데이터가 유지되지 않으므로 개념 환경의 개발, 테스트, 시연 및 검증에만 적합합니다. 프로덕션 환경의 분산 추적 플랫폼은 영구 스토리지를 위해 Elasticsearch를 지원합니다.	**memory**

매개변수	설명	값	기본값
storage: secretname:	시크릿 이름(예: **tracing-secret**)입니다.		해당 없음
storage: options: {}	스토리지를 정의하는 구성 옵션입니다.		

표 3.6. Elasticsearch 인덱스 정리 매개변수

매개변수	설명	값	기본값
storage: esIndexCleaner: enabled:	Elasticsearch 스토리지를 사용하는 경우 기본적으로 인덱스에서 오래된 추적을 정리하는 작업이 생성됩니다. 이 매개변수는 인덱스 정리 작업을 활성화하거나 비활성화합니다.	**true/ false**	**true**
storage: esIndexCleaner: numberOfDays:	인덱스를 삭제하기 전에 대기하는 날의 수입니다.	정수 값	**7**
storage: esIndexCleaner: schedule:	Elasticsearch 인덱스를 정리하는 빈도에 대한 일정을 정의합니다.	Cron 표현식	"55 23 * * *"

3.2.5.5.1. Elasticsearch 인스턴스 자동 프로비저닝

Jaeger 사용자 정의 리소스를 배포할 때 Red Hat OpenShift distributed tracing platform Operator는 OpenShift Elasticsearch Operator를 사용하여 사용자 정의 리소스 파일의 스토리지 섹션에 제공된 구성을 기반으로 Elasticsearch 클러스터를 생성합니다. Red Hat OpenShift distributed tracing platform Operator는 다음 구성이 설정된 경우 Elasticsearch를 프로비저닝합니다.

- **spec.storage:type** 이 **elasticsearch**로 설정되어 있습니다.
- **spec.storage.elasticsearch.doNotProvision** 를 **false**로 설정
- **spec.storage.options.es.server-urls** 는 정의되지 않습니다. 즉 Red Hat Elasticsearch Operator에서 프로비저닝하지 않은 Elasticsearch 인스턴스에 대한 연결이 없습니다.

Elasticsearch를 프로비저닝할 때 Red Hat OpenShift distributed tracing Platform Operator는 Elasticsearch 사용자 정의 리소스 이름을 Jaeger 사용자 정의 리소스에서 **spec.storage.elasticsearch.name** 값으로 설정합니다. **spec.storage.elasticsearch.name** 에 대한 값을 지정하지 않으면 Operator는 **elasticsearch** 를 사용합니다.

제한 사항

- 네임스페이스당 자체 프로비저닝 Elasticsearch 인스턴스가 있는 하나의 분산 추적 플랫폼만 가질 수 있습니다. Elasticsearch 클러스터는 단일 분산 추적 플랫폼 인스턴스 전용입니다.
- 네임스페이스당 Elasticsearch가 하나만 있을 수 있습니다.

참고

OpenShift Logging의 일부로 Elasticsearch를 이미 설치한 경우 Red Hat OpenShift distributed tracing platform Operator는 설치된 OpenShift Elasticsearch Operator를 사용하여 스토리지를 프로비저닝할 수 있습니다.

다음 구성 매개변수는 OpenShift Elasticsearch Operator를 사용하여 Red Hat OpenShift distributed tracing Platform Operator에서 생성한 인스턴스인 *자체 프로비저닝* Elasticsearch 인스턴스에 대한 것입니다. 구성 파일의 **spec:storage:elasticsearch**에서 자체 프로비저닝 Elasticsearch에 대한 구성 옵션을 지정합니다.

표 3.7. Elasticsearch 리소스 구성 매개변수

매개변수	설명	값	기본값
elasticsearch: properties: doNotProvision:	을 사용하여 Red Hat OpenShift distributed tracing Platform Operator에서 Elasticsearch 인스턴스를 프로비저닝할지 여부를 지정합니다.	**true/false**	**true**
elasticsearch: properties: name:	Elasticsearch 인스턴스의 이름입니다. Red Hat OpenShift distributed tracing platform Operator는 이 매개변수에 지정된 Elasticsearch 인스턴스를 사용하여 Elasticsearch에 연결합니다.	string	**elasticsearch**
elasticsearch: nodeCount:	Elasticsearch 노드 수입니다. 고가용성의 경우 최소 3개의 노드를 사용합니다. "스플릿 브레인" 문제가 발생할 수 있으므로 2개의 노드를 사용하지 마십시오.	정수 값입니다. 예를 들면 개념 증명 = 1, 최소 배포 =3입니다.	3
elasticsearch: resources: requests: cpu:	사용자 환경 구성에 따른 요청에 대한 중앙 처리 단위 수입니다.	코어 또는 밀리코어(예: 200m, 0.5, 1)에 지정됩니다. 예를 들면 개념 증명 = 500m, 최소 배포 =1입니다.	1

매개변수	설명	값	기본값
elasticsearch: resources: requests: memory:	환경 구성에 따른 요청에 사용 가능한 메모리입니다.	200Ki, 50Mi, 5Gi와 같이 바이트 단위로 지정됩니다. 예를 들면 개념 증명 = 1Gi, 최소 배포 = 16Gi*입니다.	16Gi
elasticsearch: resources: limits: cpu:	사용자 환경 구성에 따른 중앙 처리 장치 수에 대한 제한입니다.	코어 또는 밀리코어(예: 200m, 0.5, 1)에 지정됩니다. 예를 들면 개념 증명 = 500m, 최소 배포 =1입니다.	
elasticsearch: resources: limits: memory:	사용자 환경 구성에 따라 사용 가능한 메모리 제한입니다.	200Ki, 50Mi, 5Gi와 같이 바이트 단위로 지정됩니다. 예를 들면 개념 증명 = 1Gi, 최소 배포 = 16Gi*입니다.	
elasticsearch: redundancyPolicy:	데이터 복제 정책은 Elasticsearch shard가 클러스터의 데이터 노드에 복제되는 방법을 정의합니다. 지정하지 않으면 Red Hat OpenShift distributed tracing platform Operator가 노드 수에 따라 가장 적절한 복제를 자동으로 결정합니다.	**ZeroRedundancy**(replica shard 없음), **SingleRedundancy**(하나의 replica shard), **MultipleRedundancy**(각 인덱스가 데이터 노드의 반을 넘어 분산됨), **FullRedundancy**(각 인덱스가 클러스터의 모든 데이터 노드에 전체적으로 복제됨).	
elasticsearch: useCertManagement:	을 사용하여 분산 추적 플랫폼에서 Red Hat Elasticsearch Operator의 인증서 관리 기능을 사용해야 하는지 여부를 지정합니다. 이 기능은 OpenShift Container Platform 4.7에서 Red Hat OpenShift 5.2의 로깅 하위 시스템에 추가되었으며 새 Jaeger 배포에 권장되는 설정입니다.	**true/false**	**true**
	*각 Elasticsearch 노드는 더 낮은 메모리 설정으로 작동할 수 있지만 프로덕션 배포에는 권장되지 않습니다. 프로덕션 용도의 경우 기본적으로 각 Pod에 할당된 16Gi 미만이 있어야 하지만 Pod당 최대 64Gi까지 할당할 수도 있습니다.		

프로덕션 스토리지 예

```
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
```

영구 스토리지가 있는 스토리지 예:

```
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) 영구 스토리지 구성. 이 경우 AWS **gp2**에 **5Gi** 크기가 있습니다. 값을 지정하지 않으면 분산 추적 플랫폼은 **emptyDir** 을 사용합니다. OpenShift Elasticsearch Operator는 분산 추적 플랫폼 인스턴스로 제거되지 않은 **PersistentVolumeClaim** 및 **PersistentVolume** 을 프로비저닝합니다. 동일한 이름과 네임스페이스를 사용하여 분산 추적 플랫폼 인스턴스를 생성하는 경우 동일한 볼륨을 마운트할 수 있습니다.

3.2.5.5.2. 기존 Elasticsearch 인스턴스에 연결

분산 추적이 있는 스토리지에 기존 Elasticsearch 클러스터를 사용할 수 있습니다. *외부* Elasticsearch 인스턴스라고도 하는 기존 Elasticsearch 클러스터는 Red Hat OpenShift distributed tracing Platform Operator 또는 Red Hat Elasticsearch Operator에 의해 설치되지 않은 인스턴스입니다.

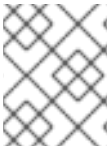
Jaeger 사용자 정의 리소스를 배포할 때 다음 구성이 설정된 경우 Red Hat OpenShift distributed tracing platform Operator가 Elasticsearch를 프로비저닝하지 않습니다.

- **spec.storage.elasticsearch.doNotProvision** 를 **true**로 설정
- **spec.storage.options.es.server-urls** has a value
- **spec.storage.elasticsearch.name** 에는 값이 있거나 Elasticsearch 인스턴스 이름이 **elasticsearch** 인 경우.

Red Hat OpenShift distributed tracing platform Operator는 **spec.storage.elasticsearch.name** 에 지정된 Elasticsearch 인스턴스를 사용하여 Elasticsearch에 연결합니다.

제한 사항

- 분산 추적 플랫폼에서 OpenShift Container Platform 로깅 Elasticsearch 인스턴스를 공유하거나 재사용할 수 없습니다. Elasticsearch 클러스터는 단일 분산 추적 플랫폼 인스턴스 전용입니다.



참고

Red Hat은 외부 Elasticsearch 인스턴스를 지원하지 않습니다. Customer Portal에서 테스트된 통합 매트릭스를 검토할 수 있습니다.

다음 구성 매개변수는 *외부* Elasticsearch 인스턴스 라고도 하는 기존 Elasticsearch 인스턴스에 대한 것입니다. 이 경우 **spec:storage:options:es** 사용자 지정 리소스 파일에서 Elasticsearch에 대한 구성 옵션을 지정합니다.

표 3.8. 일반 ES 구성 매개변수

매개변수	설명	값	기본값
es: server-urls:	Elasticsearch 인스턴스의 URL입니다.	Elasticsearch 서버의 정규화된 도메인 이름입니다.	**http://elasticsearch.<namespace>.svc:9200**
es: max-doc-count:	Elasticsearch 쿼리에서 반환하는 최대 문서 수입니다. 이는 집계에도 적용됩니다. **es.max-doc-count** 및 **es.max-num-spans**를 모두 설정하면 Elasticsearch에서 이 둘 중 작은 값을 사용합니다.		10000

매개변수	설명	값	기본값
es: max-num-spans:	[**더 이상 사용되지 않음** - 향후 릴리스에서 제거되며 대신 **es.max-doc-count**를 사용합니다.] Elasticsearch에서 쿼리 당 한 번에 가져올 최대 기간 수입니다. **es.max-num-spans** 및 **es.max-doc-count**를 모두 설정하면 Elasticsearch는 이 둘 중 작은 값을 사용합니다.		10000
es: max-span-age:	Elasticsearch에서 기간에 대한 최대 조회 수입니다.		72h0m0s
es: sniffer:	Elasticsearch의 스니퍼 구성입니다. 클라이언트는 스니핑 프로세스를 사용하여 모든 노드를 자동으로 찾습니다. 기본적으로 비활성되어 있습니다.	**true**/ **false**	**false**
es: sniffer-tls-enabled:	Elasticsearch 클러스터를 스니핑할 때 TLS를 활성화하는 옵션입니다. 클라이언트는 스니핑 프로세스를 사용하여 모든 노드를 자동으로 찾습니다. 기본적으로 비활성화되어 있습니다.	**true**/ **false**	**false**
es: timeout:	쿼리에 사용되는 시간 제한입니다. 0으로 설정하면 시간 제한이 없습니다.		0s
es: username:	Elasticsearch에 필요한 사용자 이름입니다. 기본 인증은 지정된 경우 CA도 로드합니다. **es.password**도 참조하십시오.		
es: password:	Elasticsearch에 필요한 암호입니다. **es.username**도 참조하십시오.		

매개변수	설명	값	기본값
es: version:	주요 Elasticsearch 버전입니다. 지정하지 않으면 Elasticsearch에서 값을 자동으로 탐지합니다.		0

표 3.9. ES 데이터 복제 매개변수

매개변수	설명	값	기본값
es: num-replicas:	Elasticsearch의 인덱스당 복제본 수입니다.		1
es: num-shards:	Elasticsearch의 인덱스당 shard 수입니다.		5

표 3.10. ES 인덱스 구성 매개변수

매개변수	설명	값	기본값
es: create-index-templates:	**true**로 설정할 때 애플리케이션 시작 시 인덱스 템플릿을 자동으로 생성합니다. 템플릿이 수동으로 설치되면 **false**로 설정합니다.	**true/ false**	**true**
es: index-prefix:	분산 추적 플랫폼 인덱스에 대한 선택적 접두사입니다. 예를 들어 이 값을 "production"으로 설정하면 "production-tracing-*"이라는 인덱스가 생성됩니다.		

표 3.11. ES 일괄 프로세서 구성 매개변수

매개변수	설명	값	기본값
es: bulk: actions:	대규모 프로세서가 디스크에 업데이트를 커밋하기 전에 큐에 추가할 수 있는 요청 수입니다.		1000

매개변수	설명	값	기본값
es: bulk: flush-interval:	다른 임계값에 관계없이 대규모 요청이 커밋된 후 **time.Duration**입니다. 대규모 프로세서 플러시 간격을 비활성화하려면 이를 0으로 설정합니다.		200ms
es: bulk: size:	대규모 프로세서가 디스크에 업데이트를 커밋하기 전에 대규모 요청이 수행할 수 있는 바이트 수입니다.		5000000
es: bulk: workers:	Elasticsearch에 대규모 요청을 수신하고 커밋할 수 있는 작업자 수입니다.		1

표 3.12. ES TLS 구성 매개변수

매개변수	설명	값	기본값
es: tls: ca:	원격 서버를 확인하는 데 사용되는 TLS 인증 기관(CA) 파일의 경로입니다.		기본적으로 시스템 신뢰 저장소를 사용합니다.
es: tls: cert:	이 프로세스를 원격 서버로 식별하는 데 사용되는 TLS 인증서 파일의 경로입니다.		
es: tls: enabled:	원격 서버에 연결할 때 TLS(Transport Layer Security)를 활성화합니다. 기본적으로 비활성되어 있습니다.	**true/ false**	**false**
es: tls: key:	이 프로세스를 원격 서버에 식별하는 데 사용되는 TLS 개인 키 파일의 경로입니다.		

매개변수	설명	값	기본값
es: tls: server-name:	원격 서버의 인증서에서 예상 TLS 서버 이름을 재정의합니다.		
es: token-file:	전달자 토큰이 포함된 파일의 경로입니다. 이 플래그는 지정된 경우 CA(인증 기관) 파일도 로드합니다.		

표 3.13. ES 아카이브 구성 매개변수

매개변수	설명	값	기본값
es-archive: bulk: actions:	대규모 프로세서가 디스크에 업데이트를 커밋하기 전에 큐에 추가할 수 있는 요청 수입니다.		0
es-archive: bulk: flush-interval:	다른 임계값에 관계없이 대규모 요청이 커밋된 후 **time.Duration**입니다. 대규모 프로세서 플러시 간격을 비활성화하려면 이를 0으로 설정합니다.		0s
es-archive: bulk: size:	대규모 프로세서가 디스크에 업데이트를 커밋하기 전에 대규모 요청이 수행할 수 있는 바이트 수입니다.		0
es-archive: bulk: workers:	Elasticsearch에 대규모 요청을 수신하고 커밋할 수 있는 작업자 수입니다.		0
es-archive: create-index-templates:	**true**로 설정할 때 애플리케이션 시작 시 인덱스 템플릿을 자동으로 생성합니다. 템플릿이 수동으로 설치되면 **false**로 설정합니다.	**true**/ **false**	**false**

매개변수	설명	값	기본값
es-archive: enabled:	추가 스토리지를 활성화합니다.	**true/ false**	**false**
es-archive: index-prefix:	분산 추적 플랫폼 인덱스에 대한 선택적 접두사입니다. 예를 들어 이 값을 "production"으로 설정하면 "production-tracing-*"이라는 인덱스가 생성됩니다.		
es-archive: max-doc-count:	Elasticsearch 쿼리에서 반환하는 최대 문서 수입니다. 이는 집계에도 적용됩니다.		0
es-archive: max-num-spans:	[**더 이상 사용되지 않음** - 향후 릴리스에서 제거되며 대신 **es-archive.max-doc-count**를 사용합니다.] Elasticsearch에서 쿼리 당 한 번에 가져올 최대 기간 수입니다.		0
es-archive: max-span-age:	Elasticsearch에서 기간에 대한 최대 조회 수입니다.		0s
es-archive: num-replicas:	Elasticsearch의 인덱스 당 복제본 수입니다.		0
es-archive: num-shards:	Elasticsearch의 인덱스 당 shard 수입니다.		0
es-archive: password:	Elasticsearch에 필요한 암호입니다. **es.username**도 참조하십시오.		
es-archive: server-urls:	Elasticsearch 서버의 쉼표로 구분된 목록입니다. 정규화된 URL로 지정해야 합니다(예: http://localhost:9200).		

매개변수	설명	값	기본값
es-archive: sniffer:	Elasticsearch의 스니퍼 구성입니다. 클라이언트는 스니핑 프로세스를 사용하여 모든 노드를 자동으로 찾습니다. 기본적으로 비활성되어 있습니다.	**true**/ **false**	**false**
es-archive: sniffer-tls-enabled:	Elasticsearch 클러스터를 스니핑할 때 TLS를 활성화하는 옵션입니다. 클라이언트는 스니핑 프로세스를 사용하여 모든 노드를 자동으로 찾습니다. 기본적으로 비활성되어 있습니다.	**true**/ **false**	**false**
es-archive: timeout:	쿼리에 사용되는 시간 제한입니다. 0으로 설정하면 시간 제한이 없습니다.		0s
es-archive: tls: ca:	원격 서버를 확인하는 데 사용되는 TLS 인증 기관(CA) 파일의 경로입니다.		기본적으로 시스템 신뢰 저장소를 사용합니다.
es-archive: tls: cert:	이 프로세스를 원격 서버로 식별하는 데 사용되는 TLS 인증서 파일의 경로입니다.		
es-archive: tls: enabled:	원격 서버에 연결할 때 TLS(Transport Layer Security)를 활성화합니다. 기본적으로 비활성되어 있습니다.	**true**/ **false**	**false**
es-archive: tls: key:	이 프로세스를 원격 서버에 식별하는 데 사용되는 TLS 개인 키 파일의 경로입니다.		
es-archive: tls: server-name:	원격 서버의 인증서에서 예상 TLS 서버 이름을 재정의합니다.		

매개변수	설명	값	기본값
es-archive: token-file:	전달자 토큰이 포함된 파일의 경로입니다. 이 플래그는 지정된 경우 CA(인증 기관) 파일도 로드합니다.		
es-archive: username:	Elasticsearch에 필요한 사용자 이름입니다. 기본 인증은 지정된 경우 CA도 로드합니다. **es-archive.password**도 참조하십시오.		
es-archive: version:	주요 Elasticsearch 버전입니다. 지정하지 않으면 Elasticsearch에서 값을 자동으로 탐지합니다.		0

볼륨 마운트가 있는 스토리지 예

```
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
```

다음 예는 시크릿에 저장된 볼륨 및 사용자/암호에서 마운트된 TLS CA 인증서가 포함된 외부 Elasticsearch 클러스터를 사용하는 Jaeger CR을 보여줍니다.

외부 Elasticsearch 예:

-

```
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) 기본 네임스페이스에서 실행되는 Elasticsearch 서비스에 대한 URL입니다.

(2) TLS 구성입니다. 이 경우 CA 인증서만 해당하지만 상호 TLS를 사용하는 경우 es.tls.key 및 es.tls.cert를 포함할 수 있습니다.

(3) 환경 변수 ES_PASSWORD 및 ES_USERNAME을 정의하는 시크릿입니다. kubectl create secret generic tracing-secret --from-literal=ES_PASSWORD=changeme --from-literal=ES_USERNAME=elastic에서 생성됩니다.

(4) 모든 스토리지 구성 요소에 마운트되는 볼륨 마운트 및 볼륨입니다.

3.2.5.6. Elasticsearch로 인증서 관리

Red Hat Elasticsearch Operator를 사용하여 인증서를 생성하고 관리할 수 있습니다. Red Hat Elasticsearch Operator를 사용하여 인증서를 관리하면 여러 Jaeger 수집기가 포함된 단일 Elasticsearch 클러스터를 사용할 수도 있습니다.

중요

Elasticsearch를 사용하여 인증서를 관리하는 것은 기술 프리뷰 기능 전용입니다. Technology Preview 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

버전 2.4부터 Red Hat OpenShift distributed tracing platform Operator는 Elasticsearch 사용자 정의 리소스에서 다음 주석을 사용하여 인증서 생성을 Red Hat Elasticsearch Operator에 위임합니다.

- **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"**

여기서 **<shared-es-node-name>**은 Elasticsearch 노드의 이름입니다. 예를 들어**custom-es** 라는 Elasticsearch 노드를 생성하는 경우 사용자 정의 리소스는 다음 예와 유사할 수 있습니다.

주석을 표시하는 Elasticsearch CR의 예

```
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
```

전제 조건

- OpenShift Container Platform 4.7
- logging subsystem for Red Hat OpenShift 5.2
- Elasticsearch 노드와 Jaeger 인스턴스는 동일한 네임스페이스에 배포해야 합니다. 예: **tracing-system**.

Jaeger 사용자 정의 리소스에서 **spec.storage.elasticsearch.useCertManagement** 를 **true** 로 설정하여 인증서 관리를 활성화합니다.

useCertManagement표시 예

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: jaeger-prod
spec:
  strategy: production
  storage:
    type: elasticsearch
    elasticsearch:
      name: custom-es
      doNotProvision: true
      useCertManagement: true
```

Red Hat OpenShift distributed tracing platform Operator는 Elasticsearch 사용자 정의 리소스 이름을 Elasticsearch 사용자 정의 리소스를 프로비저닝할 때 Jaeger 사용자 정의 리소스에서 **spec.storage.elasticsearch.name** 값으로 설정합니다.

인증서는 Red Hat Elasticsearch Operator 및 Red Hat OpenShift distributed tracing Platform Operator에서 인증서를 삽입합니다.

3.2.5.7. 쿼리 구성 옵션

쿼리는 스토리지에서 추적을 검색하고 사용자 인터페이스에서 표시하도록 호스팅하는 서비스입니다.

표 3.14. Red Hat OpenShift distributed tracing platform Operator에서 쿼리를 정의하는 데 사용하는 매개변수

매개변수	설명	값	기본값

`spec:`
`  query:`
`  replicas:`
	생성할 쿼리 복제본 수를 지정합니다.	예: 정수 **2**)	

표 3.15. 쿼리에 전달된 구성 매개변수

매개변수	설명	값	기본값

`spec:`
`  query:`
`  options: {}`
	쿼리 서비스를 정의하는 구성 옵션입니다.		

`options:` `  log-level:`	쿼리의 로깅 수준입니다.	가능한 값: **debug**,**info**,**warn**,**error**,**fatal**,**panic**.	

매개변수	설명	값	기본값
options: query: base-path:	모든 jaeger-query HTTP 경로의 기본 경로는 root 값이 아닌 값으로 설정할 수 있습니다(예: **/jaeger**는 모든 UI URL을 **/jaeger**로 시작합니다). 이는 리버스 프록시 뒤에서 jaeger-query를 실행할 때 유용할 수 있습니다.	/<path>	

샘플 쿼리 구성

```
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 구성 옵션

Ingester는 Kafka 항목에서 읽고 Elasticsearch 스토리지 백엔드에 쓰는 서비스입니다. allInOne 또는 production 배포 전략을 사용하는 경우 Ingester 서비스를 구성할 필요가 없습니다.

표 3.16. Ingester에 전달된 Jaeger 매개변수

매개변수	설명	값
spec: ingester: options: {}	Ingester 서비스를 정의하는 구성 옵션입니다.	
options: deadlockInterval:	Ingester가 종료되기 전에 메시지를 기다려야 하는 간격(초 또는 분)을 지정합니다. 교착 상태 간격은 시스템 초기화 중에 메시지가 도착하지 않을 때 Ingester를 종료하지 않도록 기본적으로 (**0**)로 설정됩니다.	분 및 초(예: **1m0s**)입니다. 기본값은 **0**입니다.

매개변수	설명	값
options: kafka: consumer: topic:	**topic** 매개변수는 수집기에서 메시지를 생성하기 위해 사용하는 Kafka 구성을 식별하고 메시지를 사용하는 Ingester를 식별합니다.	소비자의 레이블입니다. 예를 들면 **jaeger-spans**입니다.
options: kafka: consumer: brokers:	메시지를 사용하려면 Ingester에서 사용하는 Kafka 구성을 식별합니다.	브로커의 레이블은 예를 들면 **my-cluster-kafka-brokers.kafka:9092**입니다.
options: log-level:	Ingester의 로깅 수준입니다.	가능한 값: **debug**,**info**,**warn**,**error**,**fatal**,**dpanic**,**panic**.

스트리밍 수집기 및 Ingester 예

```
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. 사이드카 삽입

Red Hat OpenShift distributed tracing 플랫폼은 애플리케이션 Pod 내의 프록시 사이드카를 사용하여 에이전트를 제공합니다. Red Hat OpenShift distributed tracing platform Operator는 에이전트 사이드카를 배포 워크로드에 삽입할 수 있습니다. 자동 사이드카 삽입을 활성화하거나 수동으로 관리할 수 있습니다.

3.2.6.1. 자동으로 사이드카 삽입

Red Hat OpenShift distributed tracing platform Operator는 Jaeger 에이전트 사이드카를 배포 워크로드에 삽입할 수 있습니다. 사이드카 자동 삽입을 사용하려면 **sidecar.jaegertracing.io/inject** 주석을 문자열 **true** 에 추가하거나**$ oc get jaegers** 를 실행하여 반환된 분산 추적 플랫폼 인스턴스 이름에 추가합니다. **true** 를 지정하면 배포와 동일한 네임스페이스에 대한 단일 분산 추적 플랫폼 인스턴스만 있어야 합니다. 그러지 않으면 Operator에서 사용할 분산 추적 플랫폼 인스턴스를 결정할 수 없습니다. 배포의 특정 분산 추적 플랫폼 인스턴스 이름은 해당 네임스페이스에 적용된 **true** 보다 우선 순위가 높습니다.

다음 스니펫에서는 에이전트가 동일한 네임스페이스에서 사용 가능한 단일 분산 추적 플랫폼 인스턴스를 가리키는 사이드카를 삽입할 간단한 애플리케이션을 보여줍니다.

자동 사이드카 삽입 예

```
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 문자열 **true** 또는 Jaeger 인스턴스 이름으로 설정합니다.

사이드카가 삽입되면 에이전트는 **localhost** 의 기본 위치에서 액세스할 수 있습니다.

3.2.6.2. 수동으로 사이드카 삽입

Red Hat OpenShift distributed tracing platform Operator는 Jaeger 에이전트 사이드카를 배포 워크로드에만 자동으로 삽입할 수 있습니다. **StatefulSets'** 및 **'DaemonSets** 같은 배포 이외의 컨트롤러 유형의 경우 사양에 Jaeger 에이전트 사이드카를 수동으로 정의할 수 있습니다.

다음 스니펫에서는 Jaeger 에이전트 사이드카의 컨테이너 섹션에 포함할 수 있는 수동 정의를 보여줍니다.

StatefulSet의 사이드카 정의 예

```
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
```

그런 다음 에이전트는 localhost의 기본 위치에서 액세스할 수 있습니다.

3.3. 분산 추적 데이터 수집 구성 및 배포

Red Hat OpenShift distributed tracing data collection Operator는 Red Hat OpenShift distributed tracing 데이터 수집 리소스를 생성하고 배포할 때 사용할 아키텍처 및 구성 설정을 정의하는 CRD(사용자 정의 리소스 정의) 파일을 사용합니다. 기본 구성을 설치하거나 비즈니스 요구 사항에 맞게 파일을 수정할 수 있습니다.

3.3.1. OpenTelemetry 수집기 구성 옵션

중요

Red Hat OpenShift distributed tracing data collection Operator는 기술 프리뷰 기능 전용입니다. Technology Preview 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

OpenTelemetry Collector는 Telemetry 데이터에 액세스하는 세 가지 구성 요소로 구성됩니다.

- 수신자 - 기반으로 푸시하거나 가져올 수 있는 수신자는 데이터가 수집기로 들어오는 방법입니다. 일반적으로 수신자는 지정된 형식으로 데이터를 수락하고 내부 형식으로 변환하여 해당 파이프라인에 정의된 프로세서 및 내보내기로 전달합니다. 기본적으로 수신자는 구성되지 않습니다. 하나 이상의 수신자를 구성해야 합니다. 수신자는 하나 이상의 데이터 소스를 지원할 수 있습니다.

- 프로세서 - (선택 사항) 프로세서는 수신 및 내보내기 간의 데이터에서 실행됩니다. 기본적으로 프로세서는 사용할 수 없습니다. 모든 데이터 소스에 대해 프로세서를 활성화해야 합니다. 일부 프로세서는 모든 데이터 소스를 지원하는 것은 아닙니다. 데이터 소스에 따라 여러 프로세서를 사용하도록 설정하는 것이 좋습니다.Depending on the data source, it may be recommended that multiple processors be enabled. 또한 프로세서의 순서가 중요합니다.

- Exporters - 푸시 또는 가져오기를 기반으로 할 수 있는 내보내기는 하나 이상의 백엔드/대상으로 데이터를 전송하는 방법입니다. 기본적으로 내보내기는 구성되지 않습니다. 하나 이상의 내보내기를 구성해야 합니다. 내보내기자는 하나 이상의 데이터 소스를 지원할 수 있습니다. 내보내기에는 기본 설정이 있을 수 있지만 적어도 대상 및 보안 설정을 지정하려면 구성이 필요합니다.

사용자 정의 리소스 YAML 파일에서 구성 요소의 여러 인스턴스를 정의할 수 있습니다. 구성되면 YAML 파일의 **spec.config.service** 섹션에 정의된 파이프라인을 통해 이러한 구성 요소를 활성화해야 합니다. 모범 사례로 필요한 구성 요소만 활성화해야 합니다.

OpenTelemetry 사용자 정의 리소스 파일 샘플

```
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]
```

참고

구성 요소가 구성되어 있지만 **service** 섹션에 정의되지 않은 경우 활성화되지 않습니다.

표 3.17. Operator에서 OpenTelemetry 수집기를 정의하는 데 사용하는 매개변수

매개변수	설명	값	기본값
receivers:	수신자는 데이터가 수집기로 들어오는 방법입니다. 기본적으로 수신자는 구성되지 않습니다. 구성을 유효한 것으로 간주하려면 하나 이상의 활성화된 수신자가 있어야 합니다. 수신자는 파이프라인에 추가하여 활성화됩니다.	**otlp**, **jaeger**	없음
receivers: otlp:	**oltp** 및 **jaeger** 수신자는 기본 설정과 함께 제공되며 수신자 이름을 지정하기에 충분합니다.		
processors:	프로세서는 수신 및 내보내기 간의 데이터에서 실행됩니다. 기본적으로 프로세서는 사용할 수 없습니다.		없음
exporters:	내보내기자는 하나 이상의 백엔드/대상으로 데이터를 보냅니다. 기본적으로 내보내기는 구성되지 않습니다. 구성을 유효한 것으로 간주하려면 하나 이상의 내보내기가 활성화되어 있어야 합니다. 내보내기를 파이프라인에 추가하여 사용할 수 있습니다. 내보내기에는 기본 설정이 있을 수 있지만 적어도 대상 및 보안 설정을 지정하려면 구성이 필요합니다.	로깅,**jaeger**	없음

매개변수	설명	값	기본값
exporters: jaeger: endpoint:	**jaeger** exporter의 끝점은 < **name>-collector-headless. <namespace>.svc** 여야 하며 보안 연결이 설정되려면 Jaeger 배포의 이름과 네임스페이스가 있어야 합니다.		
exporters: jaeger: tls: ca_file:	CA 인증서의 경로입니다. 클라이언트의 경우 서버 인증서를 확인합니다. 서버의 경우 클라이언트 인증서를 확인합니다. 비어 있는 경우 시스템 루트 CA를 사용합니다.		
service: pipelines:	구성 요소는 **services.pipeline** 아래의 파이프라인에 추가하여 활성화됩니다.		
service: pipelines: traces: receivers:	**service.pipelines.traces** 에 추가하여 추적에 대한 수신자를 활성화합니다.		없음
service: pipelines: traces: processors:	**service.pipelines.traces** 에서 추적을 위해 프로세서를 활성화합니다.		없음
service: pipelines: traces: exporters:	**service.pipelines.traces** 에서 추적에 대해 내보내기를 활성화합니다.		없음

3.3.2. 배포 검증

3.3.3. Jaeger 콘솔에 액세스

Jaeger 콘솔에 액세스하려면 Red Hat OpenShift Service Mesh 또는 Red Hat OpenShift distributed tracing이 설치되어 있고 Red Hat OpenShift distributed tracing Platform이 설치되어 있어야 합니다.

설치 프로세스는 Jaeger 콘솔에 액세스하기 위한 경로를 생성합니다.

Jaeger 콘솔의 URL을 알고 있으면 직접 액세스할 수 있습니다. URL을 모르는 경우 다음 지침을 사용하십시오.

OpenShift 콘솔의 프로세스

1. OpenShift Container Platform 웹 콘솔에 cluster-admin 권한이 있는 사용자로 로그인합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.
2. 네트워킹 → 경로로 이동합니다.
3. 경로 페이지의 네임스페이스 메뉴에서 컨트롤 플레인 프로젝트 (예:**tracing-system**)를 선택합니다.
   Location 열에는 각 경로에 대한 연결된 주소가 표시됩니다.
4. 필요한 경우 필터를 사용하여 **jaeger** 경로를 찾습니다. 경로 위치를 클릭하여 콘솔을 시작합니다.
5. OpenShift와 함께 로그인을 클릭합니다.

CLI의 프로세스

1. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다. Red Hat OpenShift Dedicated를 사용하는 경우 **dedicated-admin** 역할의 계정이 있어야 합니다.

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

2. 명령줄을 사용하여 경로 세부 정보를 쿼리하려면 다음 명령을 입력합니다. 이 예제에서 **tracing-system** 은 컨트롤 플레인 네임스페이스입니다.

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

3. 브라우저를 시작하고 **https://<JAEGER_URL >** 으로 이동합니다. 여기서 **<JAEGER_URL >**은 이전 단계에서 검색한 경로입니다.
4. OpenShift Container Platform 콘솔에 액세스하는 데 사용하는 것과 동일한 사용자 이름 및 암호를 사용하여 로그인합니다.
5. 서비스 메시에 서비스를 추가하고 추적을 생성한 경우 필터를 사용하여 추적 데이터를 검색할 수 있습니다.
   콘솔 설치를 검증하는 경우 표시할 추적 데이터가 없습니다.

3.4. 분산 추적 업그레이드

OLM(Operator Lifecycle Manager)은 클러스터에서 Operator의 설치, 업그레이드, RBAC(역할 기반 액세스 제어)를 제어합니다. OLM은 OpenShift Container Platform에서 기본적으로 실행됩니다. 사용 가능한 Operator 및 설치된 Operator의 업그레이드에 대한 OLM 쿼리입니다. OpenShift Container Platform에서 업그레이드를 처리하는 방법에 대한 자세한 내용은 Operator Lifecycle Manager 설명서를 참조하십시오.

업데이트 중에 Red Hat OpenShift distributed tracing Operator는 관리되는 분산 추적 인스턴스를 Operator와 연결된 버전으로 업그레이드합니다. Red Hat OpenShift distributed tracing Platform Operator의 새 버전이 설치될 때마다 Operator에서 관리하는 모든 분산 추적 플랫폼 애플리케이션 인스턴

스가 Operator 버전으로 업그레이드됩니다. 예를 들어 Operator를 1.10에서 1.11로 업그레이드한 후 Operator는 분산 추적 플랫폼 인스턴스를 실행하고 1.11로 업그레이드합니다.

OpenShift Elasticsearch Operator를 업데이트하는 방법에 대한 자세한 내용은 OpenShift Logging 업데이트에서 참조하십시오.

3.4.1. 2.0의 Operator 채널 변경

Red Hat OpenShift distributed tracing 2.0.0에서는 다음과 같은 사항이 변경되었습니다.

- Red Hat OpenShift Jaeger Operator의 이름을 Red Hat OpenShift distributed tracing Platform Operator로 변경합니다.
- 개별 릴리스 채널에 대한 지원이 중지되었습니다. 앞으로 Red Hat OpenShift distributed tracing platform Operator는 stable Operator 채널만 지원합니다. 유지 관리 채널(예:1.24-stable)은 향후 Operator에서 더 이상 지원되지 않습니다.

버전 2.0 업데이트의 일부로 OpenShift Elasticsearch 및 Red Hat OpenShift distributed tracing Platform Operator 서브스크립션을 업데이트해야 합니다.

전제 조건

- OpenShift Container Platform 버전은 4.6 이상입니다.
- OpenShift Elasticsearch Operator를 업데이트했습니다.
- Jaeger 사용자 정의 리소스 파일을 백업했습니다.
- **cluster-admin** 역할이 있는 계정. Red Hat OpenShift Dedicated를 사용하는 경우**dedicated-admin** 역할의 계정이 있어야 합니다.

중요

Update OpenShift Logging에 설명된 대로 OpenShift Elasticsearch Operator를 업데이트하지 않은 경우 Red Hat OpenShift distributed tracing Platform Operator를 업데이트하기 전에 해당 업데이트가 완료됩니다.

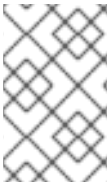
Operator 채널을 업데이트하는 방법에 대한 자세한 내용은 설치된 Operator 업데이트를 참조하십시오.

3.5. 분산 추적 제거

OpenShift Container Platform 클러스터에서 Red Hat OpenShift 분산 추적을 제거하는 단계는 다음과 같습니다.

1. Red Hat OpenShift distributed tracing pod를 종료합니다.
2. Red Hat OpenShift distributed tracing 인스턴스를 제거합니다.
3. Red Hat OpenShift distributed tracing platform Operator를 제거합니다.
4. Red Hat OpenShift distributed tracing 데이터 수집 Operator를 제거합니다.

3.5.1. 웹 콘솔을 사용하여 Red Hat OpenShift distributed tracing 플랫폼 인스턴스 제거



참고

메모리 내 스토리지를 사용하는 인스턴스를 삭제하면 모든 데이터가 영구적으로 손실됩니다. Red Hat OpenShift 분산 추적 플랫폼 인스턴스가 제거되면 Elasticsearch와 같은 영구 스토리지에 저장된 데이터는 삭제되지 않습니다.

프로세스

1. OpenShift Container Platform 웹 콘솔에 로그인합니다.
2. Operators → 설치된 Operator로 이동합니다.
3. 프로젝트 메뉴에서 Operator가 설치된 프로젝트의 이름을 선택합니다(예:**openshift-operators**).
4. Red Hat OpenShift distributed tracing platform Operator를 클릭합니다.
5. Jaeger 탭을 클릭합니다.
6. 인스턴스 옆에 있는 옵션 메뉴  를 클릭하고 Jaeger 삭제를 선택합니다.
7. 확인 메시지에서 삭제를 클릭합니다.

3.5.2. CLI에서 Red Hat OpenShift distributed tracing 플랫폼 인스턴스 제거

1. OpenShift Container Platform CLI에 로그인합니다.

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

2. 분산 추적 플랫폼 인스턴스를 표시하려면 명령을 실행합니다.

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

예를 들면 다음과 같습니다.

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

Operator 이름에는 접미사 **-operator** 가 있습니다. 다음 예제에서는 두 개의 Red Hat OpenShift distributed tracing 플랫폼 Operator와 4개의 분산 추적 플랫폼 인스턴스를 보여줍니다.

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

출력은 다음과 유사합니다.

```
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. 분산 추적 플랫폼의 인스턴스를 제거하려면 다음 명령을 실행합니다.

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

예를 들어 다음과 같습니다.

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

4. 삭제를 확인하려면 **oc get deployments** 명령을 다시 실행합니다.

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

예를 들어 다음과 같습니다.

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

다음 예와 유사한 생성된 출력이 표시되어야 합니다.

```
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. Red Hat OpenShift distributed tracing Operator 제거

프로세스

1. 클러스터에서 Operator 삭제에 대한 지침을 따르십시오.
   - Red Hat OpenShift distributed tracing platform Operator를 제거합니다.
- Red Hat OpenShift distributed tracing platform Operator가 제거된 후 해당하는 경우 OpenShift Elasticsearch Operator를 제거합니다.