OpenShift Container Platform 4.10

分散トレース

分散トレースインストール、使用法、およびリリースノート

Last Updated: 2023-10-17

OpenShift Container Platform 4.10 分散トレース

分散トレースインストール、使用法、およびリリースノート

法律上の通知

概要

このドキュメントは、OpenShift Container Platform で分散トレースを使用する方法に関する情報を提供します。

目次

第1章 分散トレースに関するリリースノート

1.1. 分散トレースの概要

サービスの所有者は、分散トレースを使用してサービスをインストルメント化し、サービスアーキテクチャーに関する洞察を得ることができます。分散トレースを使用して、現代的なクラウドネイティブのマイクロサービスベースのアプリケーションにおける、コンポーネント間の対話の監視、ネットワークプロファイリング、およびトラブルシューティングを行うことができます。

分散トレースを使用すると、以下の機能を実行できます。

- 分散トランザクションの監視
- パフォーマンスとレイテンシーの最適化
- 根本原因分析の実行

Red Hat OpenShift の分散トレースは、2 つの主要コンポーネントで設定されています。

- **Red Hat OpenShift 分散トレースプラットフォーム**: このコンポーネントは、オープンソースの Jaeger プロジェクト に基づいています。
- **Red Hat OpenShift 分散トレースデータ収集**: このコンポーネントは、オープンソースの OpenTelemetry プロジェクト に基づいています。

重要

Jaeger は、FIPS 検証済みの暗号化モジュールを使用しません。

1.2. 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

1.3. サポート

本書で説明されている手順、または OpenShift Container Platform で問題が発生した場合は、Red Hat カスタマーポータル にアクセスしてください。カスタマーポータルでは、次のことができます。

- Red Hat 製品に関するアーティクルおよびソリューションを対象とした Red Hat ナレッジベースの検索またはブラウズ。
- Red Hat サポートに対するサポートケースの送信。
- その他の製品ドキュメントへのアクセス。

クラスターの問題を特定するには、OpenShift Cluster Manager で Insights を使用できます。Insights により、問題の詳細と、利用可能な場合は問題の解決方法に関する情報が提供されます。

本書の改善への提案がある場合、またはエラーを見つけた場合は、最も関連性の高いドキュメントコンポーネントの Jira Issue を送信してください。セクション名や OpenShift Container Platform バージョンなどの具体的な情報を提供してください。

1.4. 新機能および機能拡張

今回のリリースでは、以下のコンポーネントおよび概念に関連する拡張機能が追加されました。

1.4.1. Red Hat OpenShift 分散トレース 2.8 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

1.4.1.1. Red Hat OpenShift 分散トレースバージョン 2.8 のサポート対象コンポーネントバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.42
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.74.0
Tempo Operator	Tempo	0.1.0

1.4.2. Red Hat OpenShift 分散トレース 2.7 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

1.4.2.1. Red Hat OpenShift 分散トレースバージョン 2.7 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.39
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.63.1

1.4.3. Red Hat OpenShift 分散トレース 2.6 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

1.4.3.1. Red Hat OpenShift 分散トレースバージョン 2.6 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.38
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.60

1.4.4. Red Hat OpenShift 分散トレース 2.5 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

このリリースでは、OpenTelemetry プロトコル (OTLP) を Red Hat OpenShift 分散トレーシングプラットフォーム Operator に取り込むためのサポートが導入されています。Operator は OTLP ポートを自動的に有効にするようになりました。

- ポート 4317 は OTLP gRPC プロトコルに使用されます。
- ポート 4318 は OTLP HTTP プロトコルに使用されます。

このリリースでは、Kubernetes リソース属性を収集するためのサポートも、Red Hat OpenShift 分散トレースデータ収集 Operator に追加されています。

1.4.4.1. Red Hat OpenShift 分散トレースバージョン 2.5 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.36
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.56

1.4.5. Red Hat OpenShift 分散トレース 2.4 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

このリリースでは、Red Hat Elasticsearch Operator を使用した証明書の自動プロビジョニングのサポートも追加されています。

- セルフプロビジョニング。これは、Red Hat OpenShift 分散トレーシングプラットフォーム Operator を使用して、インストール中に Red Hat Elasticsearch Operator を呼び出すことを意味します。このリリースでは、セルフプロビジョニングが完全にサポートされています。
- 最初に Elasticsearch インスタンスと証明書を作成してから、証明書を使用するように分散トレースプラットフォームを設定することは、このリリースのテクノロジープレビューです。

注記

Red Hat OpenShift 分散トレーシング 2.4 にアップグレードする場合、Operator は Elasticsearch インスタンスを再作成します。これには 5 ~ 10 分かかる場合があります。その期間、分散トレースは停止し、使用できなくなります。

1.4.5.1. Red Hat OpenShift 分散トレースバージョン 2.4 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.34.1
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.49

1.4.6. Red Hat OpenShift 分散トレース 2.3.1 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

1.4.6.1. Red Hat OpenShift 分散トレースバージョン 2.3.1 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.30.2
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.44.1-1

1.4.7. Red Hat OpenShift 分散トレース 2.3.0 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

このリリースでは、Red Hat 分散トレースプラットフォーム Operator がデフォルトで **openshift-distributed-tracing** namespace にインストールされるようになりました。この更新の前は、デフォルトのインストールは **openshift-operators** namespace にありました。

1.4.7.1. Red Hat OpenShift 分散トレースバージョン 2.3.0 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.30.1

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.44.0

1.4.8. Red Hat OpenShift 分散トレース 2.2.0 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

1.4.8.1. Red Hat OpenShift 分散トレースバージョン 2.2.0 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.30.0
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.42.0

1.4.9. Red Hat OpenShift 分散トレース 2.1.0 の新機能および機能拡張

Red Hat OpenShift 分散トレースの本リリースでは、CVE (Common Vulnerabilities and Exposures) およびバグ修正に対応しています。

1.4.9.1. Red Hat OpenShift 分散トレースバージョン 2.1.0 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.29.1
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.41.1

1.4.10. Red Hat OpenShift 分散トレース 2.0.0 の新機能および機能拡張

本リリースでは、Red Hat OpenShift Jaeger を Red Hat OpenShift 分散トレースとして再ブランディングしています。このリリースは、以下の変更、追加、および改善点で設定されています。

- Red Hat OpenShift の分散トレースは、以下の 2 つの主要コンポーネントで設定されるようになりました。
  - **Red Hat OpenShift 分散トレースプラットフォーム**: このコンポーネントは、オープンソースの Jaeger プロジェクト に基づいています。

    - **Red Hat OpenShift 分散トレースデータ収集**: このコンポーネントは、オープンソースの OpenTelemetry プロジェクト に基づいています。

- Red Hat OpenShift 分散トレースプラットフォーム Operator を Jaeger 1.28 に更新。今後、Red Hat OpenShift 分散トレースは **stable** Operator チャネルのみをサポートします。個別リリースのチャネルはサポートされなくなりました。
- OpenTelemetry 0.33 をベースとした、新しい Red Hat OpenShift 分散トレースデータ収集 Operator が導入されました。この Operator はテクノロジープレビュー機能である点に注意してください。
- OpenTelemetry プロトコル (OTLP) のサポートを Query サービスに追加されました。
- OpenShift OperatorHub に表示される新しい分散トレースアイコンが導入されました。
- 名前の変更および新機能に対応するためのドキュメントへのローリング更新が含まれます。

本リリースでは、、CVE (Common Vulnerabilities and Exposures) およびバグ修正にも対応しています。

1.4.10.1. Red Hat OpenShift 分散トレースバージョン 2.0.0 でサポートされるコンポーネントのバージョン

Operator	コンポーネント	バージョン
Red Hat OpenShift 分散トレースプラットフォーム	Jaeger	1.28.0
Red Hat OpenShift 分散トレーシングデータ収集	OpenTelemetry	0.33.0

1.5. RED HAT OPENSHIFT 分散トレースのテクノロジープレビュー機能

重要

テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

1.5.1. Red Hat Open Shift 分散トレース 2.8.0 のテクノロジープレビュー機能

本リリースでは、Red Hat OpenShift 分散トレースのテクノロジープレビュー機能として Tempo Operator がサポート対象になりました。この機能は、Tempo Operator のバージョン 0.1.0 と、アップストリームの Tempo コンポーネントのバージョン 2.0.1 を使用します。

Tempo Operator を使用して Jaeger を置き換え、ElasticSearch の代わりに S3 互換ストレージを使用できます。Tempo は Jaeger と同じ取り込みおよびクエリープロトコルをサポートし、同じユーザーインターフェイスを使用するため、Jaeger の代わりに Tempo Operator を使用するほとんどのユーザー

は機能の違いに気付きません。

このテクノロジープレビュー機能を有効にする場合は、現在の実装における以下の制限に注意してください。

- Tempo Operator は現在、非接続インストールをサポートしていません。(TRACING-3145)
- Tempo Operator で Jaeger ユーザーインターフェイス (UI) を使用すると、Jaeger UI は過去 15 分以内にトレースを送信したサービスのみを一覧表示します。過去 15 分以内にトレースを送信していないサービスの場合、トレースは Jaeger UI に表示されませんが、保存はされます。(TRACING-3139)

Red Hat OpenShift 分散トレースの今後のリリースでは、Tempo Operator のサポートを拡張することが予定されています。追加される可能性のある機能には、TLS 認証、マルチテナンシー、複数クラスターのサポートが含まれます。Tempo Operator の詳細は、Community Tempo Operator のドキュメントを参照してください。

1.5.2. Red Hat Open Shift 分散トレース 2.4.0 のテクノロジープレビュー機能

このリリースでは、Red Hat Elasticsearch Operator を使用した証明書の自動プロビジョニングのサポートも追加されています。

- セルフプロビジョニング。これは、Red Hat OpenShift 分散トレーシングプラットフォーム Operator を使用して、インストール中に Red Hat Elasticsearch Operator を呼び出すことを意味します。このリリースでは、セルフプロビジョニングが完全にサポートされています。
- 最初に Elasticsearch インスタンスと証明書を作成してから、証明書を使用するように分散トレースプラットフォームを設定することは、このリリースのテクノロジープレビューです。

1.5.3. Red Hat Open Shift 分散トレース 2.2.0 のテクノロジープレビュー機能

2.1 リリースに含まれるサポートされない OpenTelemetry Collector コンポーネントが削除されました。

1.5.4. Red Hat Open Shift 分散トレース 2.1.0 のテクノロジープレビュー機能

本リリースでは、OpenTelemetry カスタムリソースファイルで証明書を設定する方法に重大な変更が加えられました。以下の例のように、新しいバージョンでは、**ca_file** はカスタムリソースの **tls** の下に移動します。

OpenTelemetry バージョン 0.33 の CA ファイル設定

```
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 バージョン 0.41.1 の CA ファイル設定

```
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"
```

1.5.5. Red Hat Open Shift 分散トレース 2.0.0 のテクノロジープレビュー機能

本リリースには、Red Hat OpenShift 分散トレースデータ収集 Operator を使用してインストールする Red Hat OpenShift 分散トレースデータ収集が追加されています。Red Hat OpenShift 分散トレースデータ収集は、OpenTelemetry API およびインストルメンテーションに基づいています。

Red Hat OpenShift 分散トレースデータ収集には、OpenTelemetry Operator および Collector が含まれます。Collector を使用して、OpenTelemetry または Jaeger プロトコルのいずれかでトレースを受信し、トレースデータを Red Hat OpenShift 分散トレースに送信できます。現時点では、Collector のその他の機能はサポートされていません。

OpenTelemetry Collector を使用すると、開発者はベンダーに依存しない API でコードをインストルメント化し、ベンダーのロックインを回避して、可観測性ツールの拡大したエコシステムを有効にできます。

1.6. RED HAT OPENSHIFT 分散トレースの既知の問題

Red Hat OpenShift 分散トレースには、以下の制限があります。

- Apache Spark はサポートされていません。
- AMQ/Kafka 経由のストリーミングデプロイメントは、IBM Z および IBM Power Systems ではサポートされません。

これらは Red Hat OpenShift 分散トレースの既知の問題です。

- OBSDA-220 分散トレースデータコレクションを使用してイメージをプルしようとすると、イメージのプルが失敗し、**Failed to pull image** エラーメッセージが表示される場合があります。この問題に対する回避策はありません。
- TRACING-2057 Kafka API が **v1beta2** に更新され、Strimzi Kafka Operator 0.23.0 がサポートされるようになりました。ただし、この API バージョンは AMQ Streams 1.6.3 ではサポートされません。以下の環境がある場合、Jaeger サービスはアップグレードされず、新規の Jaeger サービスを作成したり、既存の Jaeger サービスを変更したりすることはできません。
  - Jaeger Operator チャネル: **1.17.x stable** または **1.20.x stable**
  - AMQ Streams Operator チャネル: **amq-streams-1.6.x**
    この問題を解決するには、AMQ Streams Operator のサブスクリプションチャネルを **amq-streams-1.7.x** または **stable** のいずれかに切り替えます。

1.7. RED HAT OPENSHIFT 分散トレースの修正された問題

- OSSM-1910 バージョン 2.6 で導入された問題により、OpenShift Container Platform Service Mesh で TLS 接続を確立できませんでした。今回の更新では、OpenShift Container Platform Service Mesh および Istio で使用される規則に一致するようにサービスポート名を変更することで、問題を解決します。
- OBSDA-208 この更新の前は、デフォルトの 200m CPU および 256Mi メモリーリソース制限により、大規模なクラスターで分散トレーシングデータ収集が継続的に再開される可能性があ

りました。この更新プログラムは、これらのリソース制限を削除することで問題を解決します。

- OBSDA-222 この更新の前は、OpenShift Container Platform 分散トレースプラットフォームでスパンがドロップされる可能性がありました。この問題の発生を防ぐために、このリリースではバージョンの依存関係が更新されています。

- TRACING-2337 Jaeger が、Jaeger ログに以下のような警告メッセージを繰り返しロギングします。

  ```
  {"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}
  ```

  この問題は、gRPC ポートではなく、クエリーサービスの HTTP(S) ポートのみを公開することで解決されました。

- TRACING-2009 Jaeger Operator が Strimzi Kafka Operator 0.23.0 のサポートを追加するように更新されました。

- TRACING-1907: アプリケーション namespace に Config Map がないため、Jaeger エージェントサイドカーの挿入が失敗していました。Config Map は間違った **OwnerReference** フィールドの設定により自動的に削除され、そのため、アプリケーション Pod は ContainerCreating ステージから移動しませんでした。誤った設定が削除されました。

- TRACING-1725 は TRACING-1631 に対応しています。これはもう 1 つの修正であり、同じ名前だが異なる namespace にある複数の Jaeger 実稼働インスタンスがある場合に Elasticsearch 証明書を適切に調整できるようになりました。BZ-1918920 も参照してください。

- TRACING-1631 同じ名前を使用するが、異なる namespace 内にある複数の Jaeger 実稼働インスタンスを使用すると、Elasticsearch 証明書に問題が発生します。複数のサービスメッシュがインストールされている場合は、すべての Jaeger Elasticsearch インスタンスが個別のシークレットではなく同じ Elasticsearch シークレットを持っていたため、OpenShift Elasticsearch Operator がすべての Elasticsearch クラスターと通信できなくなりました。

- TRACING-1300 Istio サイドカーの使用時に、Agent と Collector 間の接続に失敗しました。Jaeger Operator で有効になっている TLS 通信の更新は、Jaeger サイドカーエージェントと Jaeger Collector 間でデフォルトで提供されます。

- TRACING-1208 Jaeger UI にアクセスする際に、認証の 500 Internal Error が出力されます。OAuth を使用して UI に対する認証を試行すると、oauth-proxy サイドカーが **additionalTrustBundle** でインストール時に定義されたカスタム CA バンドルを信頼しないため、500 エラーが出力されます。

- TRACING-1166 現時点で、Jaeger ストリーミングストラテジーを非接続環境で使用することはできません。Kafka クラスターがプロビジョニングされる際に、**Failed to pull image registry.redhat.io/amq7/amq-streams-kafka-24-rhel7@sha256:f9ceca004f1b7dccb3b82d9a8027961f9fe4104e0ed69752c0bdd8078b4a1076** のエラーが出力されます。

- Trace-809 Jaeger Ingester には Kafka 2.3 との互換性がありません。Jaeger Ingester のインスタンスが複数あり、十分なトラフィックがある場合は、リバランスメッセージがログに継続的に生成されます。これは、Kafka 2.3.1 で修正された Kafka 2.3 のリグレッションによって生じます。詳細は、Jaegertracing-1819 を参照してください。

- BZ-1918920/LOG-1619 Elasticsearch Pod は更新後に自動的に再起動しません。
  回避策: Pod を手動で再起動します。

第2章 分散トレースのアーキテクチャー

2.1. 分散トレースのアーキテクチャー

ユーザーがアプリケーションでアクションを実行するたびに、応答を生成するために多数の異なるサービスに参加を要求する可能性のあるアーキテクチャーによって要求が実行されます。Red Hat OpenShift 分散トレースを使用すると、分散トレースを実行できます。これは、アプリケーションを設定するさまざまなマイクロサービスによる要求のパスを記録します。

分散トレースは、さまざまな作業ユニットの情報を連携させるために使用される技術です。これは、分散トランザクションでのイベントのチェーン全体を理解するために、通常さまざまなプロセスまたはホストで実行されます。分散トレースを使用すると、開発者は大規模なマイクロサービスアーキテクチャーで呼び出しフローを可視化できます。これは、シリアル化、並行処理、およびレイテンシーのソースについての理解にも役立ちます。

Red Hat OpenShift 分散トレースは、マイクロサービスのスタック全体での個々の要求の実行を記録し、トレースとして表示します。**トレース** とは、システムにおけるデータ/実行パスです。エンドツーエンドトレースは、1 つ以上のスパンで設定されます。

スパンは、オペレーション名、オペレーションの開始時間および期間を持ち、タグやログを持つ可能性もある Red Hat OpenShift 分散トレースの作業の論理単位を表しています。スパンは因果関係をモデル化するためにネスト化され、順序付けられます。

2.1.1. 分散トレースの概要

サービスの所有者は、分散トレースを使用してサービスをインストルメント化し、サービスアーキテクチャーに関する洞察を得ることができます。分散トレースを使用して、現代的なクラウドネイティブのマイクロサービスベースのアプリケーションにおける、コンポーネント間の対話の監視、ネットワークプロファイリング、およびトラブルシューティングを行うことができます。

分散トレースを使用すると、以下の機能を実行できます。

- 分散トランザクションの監視
- パフォーマンスとレイテンシーの最適化
- 根本原因分析の実行

Red Hat OpenShift の分散トレースは、2 つの主要コンポーネントで設定されています。

- **Red Hat OpenShift 分散トレースプラットフォーム**: このコンポーネントは、オープンソースの Jaeger プロジェクト に基づいています。
- **Red Hat OpenShift 分散トレースデータ収集**: このコンポーネントは、オープンソースの OpenTelemetry プロジェクト に基づいています。

重要

Jaeger は、FIPS 検証済みの暗号化モジュールを使用しません。

2.1.2. Red Hat OpenShift 分散トレースの機能

Red Hat OpenShift 分散トレースには、以下の機能が含まれます。

- Kiali との統合: 適切に設定されている場合は、Kiali コンソールから分散トレースデータを表示できます。
- 高いスケーラビリティー: 分散トレースバックエンドは、単一障害点がなく、ビジネスニーズに合わせてスケーリングできるように設計されています。
- 分散コンテキストの伝播: さまざまなコンポーネントからのデータをつなぎ、完全なエンドツーエンドトレースを作成できます。
- Zipkin との後方互換性: Red Hat OpenShift 分散トレースには、Zipkin のドロップイン置き換えで使用できるようにする API がありますが、本リリースでは、Red Hat は Zipkin の互換性をサポートしていません。

2.1.3. Red Hat OpenShift 分散トレースのアーキテクチャー

Red Hat OpenShift 分散トレースは、複数のコンポーネントで設定されており、トレースデータを収集し、保存し、表示するためにそれらが連携します。

- **Red Hat OpenShift 分散トレースプラットフォーム**: このコンポーネントは、オープンソースの Jaeger プロジェクト に基づいています。
  - **クライアント** (Jaeger クライアント、Tracer、Reporter、インストルメント化されたアプリケーション、クライアントライブラリー): 分散トレースプラットフォームクライアントは、OpenTracing API の言語固有の実装です。それらは、手動または (Camel (Fuse)、Spring Boot (RHOAR)、MicroProfile (RHOAR/Thorntail)、Wildfly (EAP)、その他 OpenTracing にすでに統合されているものを含む) 各種の既存オープンソースフレームワークを使用して、分散トレース用にアプリケーションをインストルメント化するために使用できます。
  - **エージェント** (Jaeger エージェント、Server Queue、Processor Worker): 分散トレースプラットフォームエージェントは、User Datagram Protocol (UDP) で送信されるスパンをリッスンするネットワークデーモンで、 Collector にバッチ処理や送信を実行します。このエージェントは、インストルメント化されたアプリケーションと同じホストに配置されることが意図されています。これは通常、Kubernetes などのコンテナー環境にサイドカーコンテナーを配置することによって実行されます。
  - **Jaeger Collector** (Collector、Queue、Worker): Jaeger エージェントと同様に、Jaeger Collector はスパンを受信し、これらを処理するために内部キューに配置します。これにより、Jaeger Collector はスパンがストレージに移動するまで待機せずに、クライアント/エージェントにすぐに戻ることができます。
  - **Storage** (Data Store): コレクターには永続ストレージのバックエンドが必要です。Red Hat OpenShift 分散トレースプラットフォームには、スパンストレージ用のプラグ可能なメカニズムがあります。このリリースでは、サポートされているストレージは Elasticsearch のみであることに注意してください。
  - **Query** (Query Service): Query は、ストレージからトレースを取得するサービスです。
  - **Ingester** (Ingester Service): Red Hat OpenShift 分散トレースは Apache Kafka を Collector と実際の Elasticsearch バッキングストレージ間のバッファーとして使用できます。Ingester は、Kafka からデータを読み取り、Elasticsearch ストレージバックエンドに書き込むサービスです。
  - **Jaeger Console**: Red Hat OpenShift 分散トレースプラットフォームユーザーインターフェイスを使用すると、分散トレースデータを可視化できます。検索ページで、トレースを検索し、個別のトレースを設定するスパンの詳細を確認できます。

- **Red Hat OpenShift 分散トレースデータ収集**: このコンポーネントは、オープンソースの OpenTelemetry プロジェクト に基づいています。
    - **OpenTelemetry Collector**: OpenTelemetry Collector は、テレメトリーデータを受信、処理、エクスポートするためのベンダーに依存しない方法です。OpenTelemetry Collector は、Jaeger や Prometheus などのオープンソースの可観測性データ形式をサポートし、1 つ以上のオープンソースまたは商用バックエンドに送信します。Collector は、インストルメンテーションライブラリーがテレメトリーデータをエクスポートするデフォルトの場所です。

第3章 分散トレースのインストール

3.1. 分散トレースのインストール

Red Hat OpenShift 分散トレースを OpenShift Container Platform にインストールするには、以下のいずれかの方法を使用できます。

- Red Hat OpenShift 分散トレースは、Red Hat OpenShift Service Mesh の一部としてインストールできます。分散トレースは、デフォルトでサービスメッシュインストールに含まれています。サービスメッシュの一部として Red Hat OpenShift 分散トレースをインストールするには、Red Hat Service Mesh Installation の手順に従います。Red Hat OpenShift 分散トレースはサービスメッシュと同じ namespace にインストールする必要があります。つまり、**ServiceMeshControlPlane** と Red Hat OpenShift 分散トレースリソースが同じ namespace にある必要があります。

- サービスメッシュをインストールする必要がない場合は、Red Hat OpenShift 分散トレース Operator を使用して分散トレース自体をインストールできます。サービスメッシュなしで Red Hat OpenShift 分散トレースをインストールするには、以下の手順を実行します。

3.1.1. 前提条件

Red Hat OpenShift 分散トレースをインストールするには、インストールアクティビティーを確認し、前提条件を満たしていることを確認してください。

- お使いの Red Hat アカウントに有効な OpenShift Container Platform サブスクリプションを用意します。サブスクリプションをお持ちでない場合は、営業担当者にお問い合わせください。

- OpenShift Container Platform 4.10 の概要 を確認します。

- OpenShift Container Platform 4.10 をインストールします。

  - AWS への OpenShift Container Platform 4.10 のインストール

  - ユーザーによってプロビジョニングされた AWS への OpenShift Container Platform 4.10 のインストール

  - ベアメタルへの OpenShift Container Platform 4.10 のインストール

  - vSphere への OpenShift Container Platform 4.10 のインストール

- OpenShift Container Platform バージョンに一致する OpenShift CLI (**oc**) のバージョンをインストールし、これをパスに追加します。

- **cluster-admin** ロールを持つアカウントがある。

3.1.2. Red Hat OpenShift 分散トレースインストールの概要

Red Hat OpenShift 分散トレースのインストール手順は以下のとおりです。

- 本書を確認し、デプロイメントストラテジーを確認します。

- デプロイメントストラテジーに永続ストレージが必要な場合は、OperatorHub を使用して OpenShift Elasticsearch Operator をインストールします。

- OperatorHub を使用して Red Hat OpenShift 分散トレースプラットフォーム Operator をインストールします。
- デプロイメントストラテジーをサポートするよう、カスタムリソース YAML ファイルを変更します。
- Red Hat OpenShift 分散トレースプラットフォームの 1 つ以上のインスタンスを OpenShift Container Platform 環境にデプロイします。

3.1.3. OpenShift Elasticsearch Operator のインストール

デフォルトの Red Hat OpenShift 分散トレースプラットフォームのデプロイメントでは、インメモリーのストレージが使用されます。これは、Red Hat OpenShift 分散トレースの評価、デモの提供、またはテスト環境での Red Hat OpenShift 分散トレースプラットフォームの使用を希望するユーザー用に、迅速にインストール行うために設計されているためです。実稼働環境で Red Hat OpenShift 分散トレースプラットフォームを使用する予定がある場合、永続ストレージのオプション (この場合は Elasticsearch) をインストールし、設定する必要があります。

前提条件

- OpenShift Container Platform Web コンソールにアクセスできる。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。

警告

Operator のコミュニティーバージョンはインストールしないでください。コミュニティー Operator はサポートされていません。

注記

OpenShift Logging の一部として OpenShift Elasticsearch Operator がすでにインストールされている場合は、OpenShift Elasticsearch Operator を再びインストールする必要はありません。Red Hat OpenShift 分散トレースプラットフォーム Operator はインストールされた OpenShift Elasticsearch Operator を使用して Elasticsearch インスタンスを作成します。

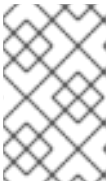
手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。
2. **Operators → OperatorHub** に移動します。
3. **Elasticsearch** とフィルターボックスに入力して、OpenShift Elasticsearch Operator を検索します。
4. Red Hat が提供する **OpenShift Elasticsearch Operator** をクリックし、Operator に関する情報を表示します。

5. **Install** をクリックします。

6. **Install Operator** ページで、**stable** 更新チャネルを選択します。これにより、新しいバージョンがリリースされると Operator が自動的に更新されます。

7. デフォルトの **All namespaces on the cluster (default)**を受け入れます。これにより、Operator がデフォルトの **openshift-operators-redhat** プロジェクトにインストールされ、Operator はクラスター内のすべてのプロジェクトで利用可能になります。



注記

Elasticsearch インストールでは、 Elasticsearch Operator に **openshift-operators-redhat** namespace が必要です。他の Red Hat OpenShift 分散トレース Operator は、**openshift-operators** namespace にインストールされます。

- デフォルトの **Automatic** 承認ストラテジーを受け入れます。デフォルトを受け入れることで、Operator の新規バージョンが利用可能になると、Operator Lifecycle Manager (OLM) は人の介入なしに、Operator の実行中のインスタンスを自動的にアップグレードします。**手動** 更新を選択する場合は、Operator の新規バージョンが利用可能になると、OLM は更新要求を作成します。クラスター管理者は、Operator が新規バージョンに更新されるように更新要求を手動で承認する必要があります。

注記

手動 の承認ストラテジーには、Operator のインストールおよびサブスクリプションプロセスを承認するための適切な認証情報を持つユーザーが必要です。

8. **Install** をクリックします。

9. **Installed Operators** ページで、**openshift-operators-redhat** プロジェクトを選択します。OpenShift Elasticsearch Operator が InstallSucceeded のステータスを表示するまで待機してから続行します。

3.1.4. Red Hat OpenShift 分散トレースプラットフォーム Operator のインストール

Red Hat OpenShift 分散トレースプラットフォームをインストールするには、OperatorHub を使用して Red Hat OpenShift 分散トレースプラットフォーム Operator をインストールします。

デフォルトでは、Operator は **openshift-operators** プロジェクトにインストールされます。

前提条件

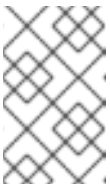
- OpenShift Container Platform Web コンソールにアクセスできる。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。
- 永続ストレージが必要な場合は、Red Hat OpenShift 分散トレースプラットフォーム Operator をインストールする前に OpenShift Elasticsearch Operator もインストールする必要があります。

警告

Operator のコミュニティーバージョンはインストールしないでください。コミュニティー Operator はサポートされていません。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。
2. **Operators → OperatorHub** に移動します。
3. フィルターに **distributed tracing platform** と入力して、Red Hat OpenShift 分散トレースプラットフォーム Operator を探します。
4. Red Hat が提供する **Red Hat OpenShift distributed tracing platform Operator** をクリックし、Operator に関する情報を表示します。
5. **Install** をクリックします。
6. **Install Operator** ページで、**stable** 更新チャネルを選択します。これにより、新しいバージョンがリリースされると Operator が自動的に更新されます。
7. デフォルトの **All namespaces on the cluster (default)** を受け入れます。これにより、Operator がデフォルトの **openshift-operators** プロジェクトにインストールされ、Operator はクラスター内のすべてのプロジェクトで利用可能になります。
   - デフォルトの **Automatic** 承認ストラテジーを受け入れます。デフォルトを受け入れることで、Operator の新規バージョンが利用可能になると、Operator Lifecycle Manager (OLM) は人の介入なしに、Operator の実行中のインスタンスを自動的にアップグレードします。**手動** 更新を選択する場合は、Operator の新規バージョンが利用可能になると、OLM は更新要求を作成します。クラスター管理者は、Operator が新規バージョンに更新されるように更新要求を手動で承認する必要があります。



注記

手動 の承認ストラテジーには、Operator のインストールおよびサブスクリプションプロセスを承認するための適切な認証情報を持つユーザーが必要です。

8. **Install** をクリックします。
9. **Operators → Installed Operators** に移動します。
10. **Installed Operators** ページで、**openshift-operators** プロジェクトを選択します。Red Hat OpenShift 分散トレースプラットフォーム Operator が Succeeded のステータスを表示するまで待機してから続行します。

3.1.5. Red Hat OpenShift 分散トレースデータ収集 Operator のインストール

重要

Red Hat OpenShift 分散トレースデータ収集 Operator は、テクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

Red Hat OpenShift 分散トレースデータ収集をインストールするには、OperatorHub を使用して Red Hat OpenShift 分散トレースデータ収集 Operator をインストールします。

デフォルトでは、Operator は **openshift-operators** プロジェクトにインストールされます。

前提条件

- OpenShift Container Platform Web コンソールにアクセスできる。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。

警告

Operator のコミュニティーバージョンはインストールしないでください。コミュニティー Operator はサポートされていません。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。
2. **Operators → OperatorHub** に移動します。
3. フィルターに **distributing tracing data collection** と入力して、Red Hat OpenShift 分散トレースデータ収集 Operator を探します。
4. Red Hat が提供する **Red Hat OpenShift distributed tracing data collection Operator**をクリックし、Operator についての情報を表示します。
5. **Install** をクリックします。
6. **Install Operator** ページで、デフォルトの **stable** Update チャネルを受け入れます。これにより、新しいバージョンがリリースされると Operator が自動的に更新されます。
7. デフォルトの **All namespaces on the cluster (default)**を受け入れます。これにより、Operator がデフォルトの **openshift-operators** プロジェクトにインストールされ、Operator はクラスター内のすべてのプロジェクトで利用可能になります。

8. デフォルトの **Automatic** 承認ストラテジーを受け入れます。デフォルトを受け入れることで、Operator の新規バージョンが利用可能になると、Operator Lifecycle Manager (OLM) は人の介入なしに、Operator の実行中のインスタンスを自動的にアップグレードします。**手動** 更新を選択する場合は、Operator の新規バージョンが利用可能になると、OLM は更新要求を作成します。クラスター管理者は、Operator が新規バージョンに更新されるように更新要求を手動で承認する必要があります。

注記

手動 の承認ストラテジーには、Operator のインストールおよびサブスクリプションプロセスを承認するための適切な認証情報を持つユーザーが必要です。

9. **Install** をクリックします。

10. **Operators → Installed Operators** に移動します。

11. **Installed Operators** ページで、**openshift-operators** プロジェクトを選択します。Red Hat OpenShift 分散トレースデータ収集 Operator が Succeeded のステータスを表示するまで待機してから続行します。

3.2. 分散トレースの設定およびデプロイ

Red Hat OpenShift 分散トレースプラットフォーム Operator は、分散トレースプラットフォームリソースの作成およびデプロイ時に使用されるアーキテクチャーおよび設定を定義するカスタムリソース定義 (CRD) ファイルを使用します。デフォルト設定をインストールするか、ビジネス要件に合わせてファイルを変更することができます。

Red Hat OpenShift 分散トレースプラットフォームには、事前に定義されたデプロイメントストラテジーがあります。カスタムリソースファイルでデプロイメントストラテジーを指定します。分散トレースプラットフォームインスタンスの作成時に、Operator はこの設定ファイルを使用してデプロイメントに必要なオブジェクトを作成します。

デプロイメントストラテジーを表示する Jaeger カスタムリソースファイル

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

(1) Red Hat OpenShift 分散トレースプラットフォーム Operator は現時点で以下のデプロイメントストラテジーをサポートします。

- **allInOne** (デフォルト): このストラテジーは、開発、テストおよびデモの目的で使用されることが意図されています。主なバックエンドコンポーネントである Agent、Collector、および Query サービスはすべて、デフォルトでインメモリーストレージを使用するように設定された単一の実行可能ファイルにパッケージ化されます。

注記

インメモリーストレージには永続性がありません。つまり、分散トレースプラットフォームインスタンスがシャットダウンするか、再起動するか、置き換えられると、トレースデータが失われます。各 Pod には独自のメモリーがあるため、インメモリーストレージはスケーリングできません。永続ストレージの場合、デフォルトのストレージとして Elasticsearch を使用する **production** または **streaming** ストラテジーを使用する必要があります。

- **production**: production ストラテジーは、実稼働環境向けのストラテジーであり、トレースデータの長期の保存が重要となり、より拡張性および高可用性のあるアーキテクチャーも必要になります。そのため、バックエンドコンポーネントはそれぞれ別々にデプロイされます。エージェントは、インストルメント化されたアプリケーションのサイドカーとして挿入できます。Query および Collector サービスは、サポートされているストレージタイプ (現時点では Elasticsearch) で設定されます。これらの各コンポーネントの複数のインスタンスは、パフォーマンスと回復性を確保するために、必要に応じてプロビジョニングできます。

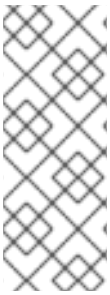
- **streaming**: streaming ストラテジーは、Collector と Elasticsearch バックエンドストレージ間に効果的に配置されるストリーミング機能を提供することで、production ストラテジーを増強する目的で設計されています。これにより、負荷の高い状況でバックエンドストレージに加わる圧力を軽減し、他のトレース処理後の機能がストリーミングプラットフォーム (AMQ Streams/ Kafka) から直接リアルタイムのスパンデータを利用できるようにします。

注記

streaming ストラテジーには、AMQ Streams 用の追加の Red Hat サブスクリプションが必要です。

注記

IBM Z では、現在ストリーミングデプロイメントストラテジーはサポートされていません。

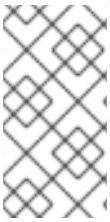


注記

サービスメッシュの一部としてか、スタンドアロンのコンポーネントとして Red Hat OpenShift 分散トレースをインストールし、使用するには 2 つの方法があります。Red Hat OpenShift Service Mesh の一部として分散トレースをインストールしている場合、ServiceMeshControlPlane の一部として基本的な設定を実行できますが、完全な制御を行うためには、Jaeger CR を設定してから、ServiceMeshControlPlane の分散トレース設定ファイルを参照する 必要があります。

3.2.1. Web コンソールからの分散トレースのデフォルトストラテジーのデプロイ

カスタムリソース定義 (CRD) は、Red Hat OpenShift 分散トレースのインスタンスをデプロイする際に使用される設定を定義します。デフォルト CR は **jaeger-all-in-one-inmemory** という名前で、デフォルトの OpenShift Container Platform インストールに正常にインストールできるように最小リソースで設定されます。このデフォルト設定を使用して、**AllInOne** デプロイメントストラテジーを使用する Red Hat OpenShift 分散トレースプラットフォームのインスタンスを作成するか、独自のカスタムリソースファイルを定義できます。



注記

インメモリーストレージには永続性がありません。Jaeger Pod がシャットダウンするか、再起動するか、置き換えられると、トレースデータが失われます。永続ストレージの場合、デフォルトのストレージとして Elasticsearch を使用する **production** または **streaming** ストラテジーを使用する必要があります。

前提条件

- Red Hat OpenShift 分散トレースプラットフォーム Operator がインストールされている。
- デプロイメントのカスタマイズ手順を確認している。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。
2. 新規プロジェクト (例: **tracing-system**) を作成します。

注記

サービスメッシュの一部としてインストールする場合、分散トレースリソースは、**istio-system** など、**ServiceMeshControlPlane** リソースと同じ namespace にインストールする必要があります。

   a. **Home → Projects** に移動します。
   b. **Create Project** をクリックします。
   c. **Name** フィールドに **tracing-system** を入力します。
   d. **Create** をクリックします。
3. **Operators → Installed Operators** に移動します。
4. 必要な場合は、**Project** メニューから **tracing-system** を選択します。Operator が新規プロジェクトにコピーされるまでに数分待機する必要がある場合があります。
5. Red Hat OpenShift distributed tracing platform Operator をクリックします。**Details** タブの **Provided APIs** で、Operator は単一リンクを提供します。
6. **Jaeger** で、**Create Instance** をクリックします。
7. **Create Jaeger** ページで、デフォルトを使用してインストールするには、 **Create** をクリックして分散トレースプラットフォームのインスタンスを作成します。
8. **Jaegers** ページで、分散トレースプラットフォームインスタンスの名前 (例: **jaeger-all-in-one-inmemory**) をクリックします。
9. **Jaeger Details** ページで、**Resources** タブをクリックします。Pod のステータスが Running になるまで待機してから続行します。

3.2.1.1. CLI からの分散トレースのデフォルトストラテジーのデプロイ

以下の手順に従って、コマンドラインから分散トレースプラットフォームのインスタンスを作成します。

前提条件

- Red Hat OpenShift 分散トレースプラットフォーム Operator がインストールされ検証されている。
- デプロイメントのカスタマイズ手順を確認している。
- OpenShift Container Platform バージョンに一致する OpenShift CLI (**oc**) にアクセスできる。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。

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

2. **tracing-system** という名前の新規プロジェクトを作成します。

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

3. 以下のテキストが含まれる **jaeger.yaml** という名前のカスタムリソースファイルを作成します。

例: jaeger-all-in-one.yaml

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

4. 以下のコマンドを実行して、分散トレースプラットフォームをデプロイします。

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

5. 以下のコマンドを実行して、インストールプロセス時の Pod の進捗を確認します。

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

インストールプロセスが完了すると、以下の例ような出力が表示されるはずです。

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

3.2.2. Web コンソールからの分散トレースの production ストラテジーのデプロイ

production デプロイメントストラテジーは、よりスケーラブルで高可用性のあるアーキテクチャーを必要とし、トレースデータの長期保存が重要となる実稼働環境向けのものです。

前提条件

- OpenShift Elasticsearch Operator がインストールされている。
- Red Hat OpenShift 分散トレースプラットフォーム Operator がインストールされている。
- デプロイメントのカスタマイズ手順を確認している。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。
2. 新規プロジェクト (例: **tracing-system**) を作成します。

注記

サービスメッシュの一部としてインストールする場合、分散トレースリソースは、**istio-system** など、**ServiceMeshControlPlane** リソースと同じ namespace にインストールする必要があります。

   a. **Home → Projects** に移動します。
   b. **Create Project** をクリックします。
   c. **Name** フィールドに **tracing-system** を入力します。
   d. **Create** をクリックします。
3. **Operators → Installed Operators** に移動します。
4. 必要な場合は、**Project** メニューから **tracing-system** を選択します。Operator が新規プロジェクトにコピーされるまでに数分待機する必要がある場合があります。
5. Red Hat OpenShift distributed tracing platform Operator をクリックします。**Overview** タブの **Provided APIs** で、Operator は単一リンクを提供します。
6. **Jaeger** で、**Create Instance** をクリックします。
7. **Create Jaeger** ページで、デフォルトの **all-in-one** YAML テキストを実稼働用の YAML 設定に置き換えます。以下は例になります。

Elasticsearch を含む jaeger-production.yaml ファイルの例

```
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. **Create** をクリックして分散トレースプラットフォームのインスタンスを作成します。
9. **Jaegers** ページで、分散トレースプラットフォームインスタンスの名前 (例: **jaeger-prod-elasticsearch**) をクリックします。
10. **Jaeger Details** ページで、**Resources** タブをクリックします。すべての Pod のステータスが Running になるまで待機してから続行します。

3.2.2.1. CLI からの分散トレースの production ストラテジーのデプロイ

以下の手順に従って、コマンドラインから分散トレースプラットフォームのインスタンスを作成します。

前提条件

- OpenShift Elasticsearch Operator がインストールされている。
- Red Hat OpenShift 分散トレースプラットフォーム Operator がインストールされている。
- デプロイメントのカスタマイズ手順を確認している。
- OpenShift Container Platform バージョンに一致する OpenShift CLI (**oc**) にアクセスできる。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。

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

2. **tracing-system** という名前の新規プロジェクトを作成します。

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

3. 直前の手順のサンプルファイルのテキストが含まれる **jaeger-production.yaml** という名前のカスタムリソースファイルを作成します。
4. 以下のコマンドを実行して、分散トレースプラットフォームをデプロイします。

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

5. 以下のコマンドを実行して、インストールプロセス時の Pod の進捗を確認します。

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

インストールプロセスが完了すると、以下の例ような出力が表示されるはずです。

```
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
```

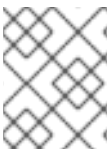
3.2.3. Web コンソールからの分散トレースの streaming ストラテジーのデプロイ

streaming デプロイメントストラテジーは、よりスケーラブルで高可用性のあるアーキテクチャーを必要とし、トレースデータの長期保存が重要となる実稼働環境向けのものです。

streaming ストラテジーは、Collector と Elasticsearch ストレージ間に配置されるストリーミング機能を提供します。これにより、負荷の高い状況でストレージに加わる圧力を軽減し、他のトレースの後処理機能が Kafka ストリーミングプラットフォームから直接リアルタイムのスパンデータを利用できるようにします。

注記

streaming ストラテジーには、AMQ Streams 用の追加の Red Hat サブスクリプションが必要です。AMQ Streams サブスクリプションをお持ちでない場合は、営業担当者にお問い合わせください。



注記

IBM Z では、現在ストリーミングデプロイメントストラテジーはサポートされていません。

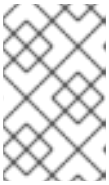
前提条件

- AMQ Streams Operator がインストールされている。バージョン 1.4.0 以降を使用している場合は、セルフプロビジョニングを使用できます。それ以外の場合は、Kafka インスタンスを作成する必要があります。
- Red Hat OpenShift 分散トレースプラットフォーム Operator がインストールされている。
- デプロイメントのカスタマイズ手順を確認している。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。

2. 新規プロジェクト (例: **tracing-system**) を作成します。



注記

サービスメッシュの一部としてインストールする場合、分散トレースリソースは、**istio-system** など、**ServiceMeshControlPlane** リソースと同じ namespace にインストールする必要があります。

   a. **Home → Projects** に移動します。

   b. **Create Project** をクリックします。

   c. **Name** フィールドに **tracing-system** を入力します。

   d. **Create** をクリックします。

3. **Operators → Installed Operators** に移動します。

4. 必要な場合は、**Project** メニューから **tracing-system** を選択します。Operator が新規プロジェクトにコピーされるまでに数分待機する必要がある場合があります。

5. Red Hat OpenShift distributed tracing platform Operator をクリックします。**Overview** タブの **Provided APIs** で、Operator は単一リンクを提供します。

6. **Jaeger** で、**Create Instance** をクリックします。

7. **Create Jaeger** ページで、デフォルトの **all-in-one** YAML テキストをストリーミング用の YAML 設定に置き換えます。以下は例になります。

例: jaeger-streaming.yaml ファイル

```
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. **Create** をクリックして分散トレースプラットフォームのインスタンスを作成します。
2. **Jaegers** ページで、分散トレースプラットフォームインスタンスの名前 (例: **jaeger-streaming**) をクリックします。
3. **Jaeger Details** ページで、**Resources** タブをクリックします。すべての Pod のステータスが Running になるまで待機してから続行します。

3.2.3.1. CLI からの分散トレースの streaming ストラテジーのデプロイ

以下の手順に従って、コマンドラインから分散トレースプラットフォームのインスタンスを作成します。

前提条件

- AMQ Streams Operator がインストールされている。バージョン 1.4.0 以降を使用している場合は、セルフプロビジョニングを使用できます。それ以外の場合は、Kafka インスタンスを作成する必要があります。
- Red Hat OpenShift 分散トレースプラットフォーム Operator がインストールされている。
- デプロイメントのカスタマイズ手順を確認している。
- OpenShift Container Platform バージョンに一致する OpenShift CLI (**oc**) にアクセスできる。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。

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

2. **tracing-system** という名前の新規プロジェクトを作成します。

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

3. 直前の手順のサンプルファイルのテキストが含まれる **jaeger-streaming.yaml** という名前のカスタムリソースファイルを作成します。
4. 以下のコマンドを実行して Jaeger をデプロイします。

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

5. 以下のコマンドを実行して、インストールプロセス時の Pod の進捗を確認します。

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

インストールプロセスが完了すると、以下の例ような出力が表示されるはずです。

```
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. デプロイメントの検証

3.2.4.1. Jaeger コンソールへのアクセス

Jaeger コンソールにアクセスするには、Red Hat OpenShift Service Mesh または Red Hat OpenShift 分散トレースがインストールされ、Red Hat OpenShift 分散トレースプラットフォームがインストール、設定、およびデプロイされている必要があります。

インストールプロセスにより、Jaeger コンソールにアクセスするためのルートが作成されます。

Jaeger コンソールの URL が分かっている場合は、これに直接アクセスできます。URL が分からない場合は、以下の指示を使用します。

OpenShift コンソールからの手順

1. cluster-admin 権限を持つユーザーとして OpenShift Container Platform Web コンソールにログインします。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。

2. **Networking** → **Routes** に移動します。

3. **Routes** ページで、**Namespace** メニューからコントロールプレーンプロジェクトを選択します (例:**tracing-system**)。
   Location 列には、各ルートのリンク先アドレスが表示されます。

4. 必要な場合は、フィルターを使用して **jaeger** ルートを検索します。ルートの **Location** をクリックしてコンソールを起動します。

5. **Log In With OpenShift** をクリックします。

CLI からの手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。

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

2. コマンドラインを使用してルートの詳細をクエリーするには、以下のコマンドを入力します。この例では、**tracing-system** がコントロールプレーン namespace です。

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

3. ブラウザーを起動し、**https://<JAEGER_URL>** に移動します。ここで、**<JAEGER_URL>** は直前の手順で検出されたルートです。

4. OpenShift Container Platform コンソールへアクセスするときに使用するものと同じユーザー名とパスワードを使用してログインします。

5. サービスメッシュにサービスを追加し、トレースを生成している場合は、フィルターと **Find Traces** ボタンを使用してトレースデータを検索します。
コンソールインストールを検証すると、表示するトレースデータはありません。

3.2.5. デプロイメントのカスタマイズ

3.2.5.1. デプロイメントのベストプラクティス

- Red Hat OpenShift 分散トレースインスタンスの名前は一意でなければなりません。複数の Red Hat OpenShift 分散トレースプラットフォームインスタンスがあり、サイドカーが挿入されたエージェントを使用している場合、Red Hat OpenShift 分散トレースプラットフォームインスタンスには一意の名前が必要となり、挿入 (injection) のアノテーションはトレースデータを報告する必要のある Red Hat OpenShift 分散トレースプラットフォームインスタンスの名前を明示的に指定する必要があります。

- マルチテナントの実装があり、テナントが namespace で分離されている場合は、Red Hat OpenShift 分散トレースプラットフォームインスタンスを各テナント namespace にデプロイします。

  - デーモンセットとしてのエージェントは、マルチテナントインストールまたは Red Hat OpenShift Dedicated ではサポートされません。サイドカーとしてのエージェントは、これらのユースケースでサポートされる唯一の設定です。

- Red Hat OpenShift Service Mesh の一部として分散トレースをインストールする場合、分散トレースリソースは、**ServiceMeshControlPlane** リソースと同じ namespace にインストールする必要があります。

永続ストレージの設定は、永続ストレージについて と、選択したストレージオプションに適した設定トピックを参照してください。

3.2.5.2. 分散トレースのデフォルト設定オプション

Jaeger カスタムリソース (CR) は、分散トレースプラットフォームリソースの作成時に使用されるアーキテクチャーおよび設定を定義します。これらのパラメーターを変更して、分散トレースプラットフォームの実装をビジネスニーズに合わせてカスタマイズできます。

Jaeger 汎用 YAML の例

```
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 パラメーター

パラメーター	説明	値	デフォルト値
apiVersion:		オブジェクトの作成時に使用する API バージョン。	**jaegertracing.io/v1**
jaegertracing.io/v1	**kind:**	作成する Kubernetes オブジェクトの種類を定義します。	**jaeger**
	metadata:	**name** 文字列、**UID**、およびオプションの **namespace** などのオブジェクトを一意に特定するのに役立つデータ。	
OpenShift Container Platform は **UID** を自動的に生成し、オブジェクトが作成されるプロジェクトの名前で **namespace** を完了します。	**name:**	オブジェクトの名前。	分散トレースプラットフォームインスタンスの名前。

パラメーター	説明	値	デフォルト値
jaeger-all-in-one-inmemory	**spec:**	作成するオブジェクトの仕様。	分散トレースプラットフォームインスタンスのすべての設定パラメーターが含まれます。すべての Jaeger コンポーネントの共通定義が必要な場合、これは **spec** ノードで定義されます。定義が個々のコンポーネントに関連する場合は、**spec/<component>** ノードに置かれます。
該当なし	**strategy:**	Jaeger デプロイメントストラテジー	**allInOne**、**production**、または **streaming**
allInOne	**allInOne:**	**allInOne** イメージは Agent、Collector、Query、Ingester、および Jaeger UI を単一 Pod にデプロイするため、このデプロイメントの設定は、コンポーネント設定を **allInOne** パラメーターの下でネストする必要があります。	
	agent:	Agent を定義する設定オプション。	
	collector:	Jaeger Collector を定義する設定オプション。	
	sampling:	トレース用のサンプリングストラテジーを定義する設定オプション。	
	storage:	ストレージを定義する設定オプション。すべてのストレージ関連のオプションは、**allInOne** または他のコンポーネントオプションではなく、**storage** に配置される必要があります。	

パラメーター	説明	値	デフォルト値
	query:	Query サービスを定義する設定オプション。	
	ingester:	Ingester サービスを定義する設定オプション。	

以下の YAML の例は、デフォルト設定を使用して Red Hat OpenShift 分散トレースプラットフォームのデプロイメントを作成するために最低限必要なものです。

最小限必要な dist-tracing-all-in-one.yaml の例

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

3.2.5.3. Jaeger Collector 設定オプション

Jaeger Collector は、トレーサーによってキャプチャーされたスパンを受信し、**production** ストラテジーを使用する場合はそれらを永続 Elasticsearch ストレージに書き込み、**streaming** ストラテジーを使用する場合は AMQ Streams に書き込むコンポーネントです。

Collector はステートレスであるため、Jaeger Collector のインスタンスの多くは並行して実行できます。Elasticsearch クラスターの場所を除き、Collector では設定がほとんど必要ありません。

表3.2 Operator によって使用される Jaeger Collector パラメーターを定義するためのパラメーター

パラメーター	説明	値
`collector:` `  replicas:`	作成する Collector レプリカの数を指定します。	整数 (例: **5**)。

表3.3 Collector に渡される設定パラメーター

パラメーター	説明	値

`spec:`
` collector:`
`  options: {}`
	Jaeger Collector を定義する設定オプション。	

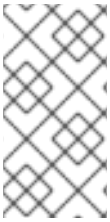
`options:`
`  collector:`
`  num-workers:`
	キューからプルするワーカーの数。	整数 (例: **50**)。

パラメーター	説明	値
options: collector: queue-size:	Collector キューのサイズ。	整数 (例: **2000**)。
options: kafka: producer: topic: jaeger-spans	**topic** パラメーターは、Collector によってメッセージを生成するために使用され、Ingester によってメッセージを消費するために使用される Kafka 設定を特定します。	プロデューサーのラベル。
options: kafka: producer: brokers: my-cluster-kafka-brokers.kafka:9092	メッセージを生成するために Collector によって使用される Kafka 設定を特定します。ブローカーが指定されていない場合で、AMQ Streams 1.4.0+ がインストールされている場合、Red Hat OpenShift 分散トレースプラットフォーム Operator は Kafka をセルフプロビジョニングします。	
options: log-level:	Collector のロギングレベル。	使用できる値は、**debug**、**info**、**warn**、**error**、**fatal**、**panic** です。

3.2.5.4. 分散トレースのサンプリング設定オプション

この Red Hat OpenShift 分散トレースプラットフォーム Operator は、リモートサンプラーを使用するように設定されているトレーサーに提供されるサンプリングストラテジーを定義するために使用できます。

すべてのトレースが生成される間に、それらの一部のみがサンプリングされます。トレースをサンプリングすると、追加の処理や保存のためにトレースにマークが付けられます。



注記

これは、トレースがサンプリングの意思決定が行われる際に Envoy プロキシーによって開始されている場合に関連がありません。Jaeger サンプリングの意思決定は、トレースがクライアントを使用してアプリケーションによって開始される場合にのみ関連します。

サービスがトレースコンテキストが含まれていない要求を受信すると、クライアントは新しいトレースを開始し、これにランダムなトレース ID を割り当て、現在インストールされているサンプリングストラテジーに基づいてサンプリングの意思決定を行います。サンプリングの意思決定はトレース内の後続のすべての要求に伝播され、他のサービスが再度サンプリングの意思決定を行わないようにします。

分散トレースプラットフォームライブラリーは、以下のサンプラーをサポートします。

- **Probabilistic**: サンプラーは、**sampling.param** プロパティーの値と等しいサンプリングの確率で、ランダムなサンプリングの意思決定を行います。たとえば、**sampling.param=0.1** を使用した場合は、約 10 のうち 1 トレースがサンプリングされます。

- **Rate Limiting**: サンプラーは、リーキーバケット (leaky bucket) レートリミッターを使用して、トレースが一定のレートでサンプリングされるようにします。たとえば、**sampling.param=2.0** を使用した場合は、1 秒あたり 2 トレースの割合で要求がサンプリングされます。

表3.4 Jaeger サンプリングのオプション

パラメーター	説明	値	デフォルト値
spec: sampling: options: {} default_strategy: service_strategy:	トレース用のサンプリングストラテジーを定義する設定オプション。		設定を指定しない場合、Collector はすべてのサービスの確率 0.001 (0.1%) のデフォルトの確率的なサンプリングポリシーを返します。
default_strategy: type: service_strategy: type:	使用するサンプリングストラテジー。上記の説明を参照してください。	有効な値は **probabilistic**、および **ratelimiting** です。	**probabilistic**
default_strategy: param: service_strategy: param:	選択したサンプリングストラテジーのパラメーター	10 進値および整数値 (0、.1、1、10)	1

この例では、トレースインスタンスをサンプリングする確率が 50% の確率的なデフォルトサンプリングストラテジーを定義します。

確率的なサンプリングの例

```
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
```

ユーザーによって指定される設定がない場合、分散トレースプラットフォームは以下の設定を使用します。

デフォルトのサンプリング

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

3.2.5.5. 分散トレースのストレージ設定オプション

spec.storage の下で Collector、Ingester、および Query サービスのストレージを設定します。これらの各コンポーネントの複数のインスタンスは、パフォーマンスと回復性を確保するために、必要に応じてプロビジョニングできます。

表3.5 分散トレースストレージを定義するために Red Hat OpenShift 分散トレースプラットフォーム Operator によって使用される一般的なストレージパラメーター

パラメーター	説明	値	デフォルト値

`spec:`
`  storage:`
`  type:`
	デプロイメントに使用するストレージのタイプ。	**memory** または **elasticsearch**メモリーストレージは、Pod がシャットダウンした場合にデータが永続化されないため、開発、テスト、デモ、および概念検証用の環境にのみに適しています。実稼働環境では、分散トレースプラットフォームは永続ストレージの Elasticsearch をサポートします。	**memory**

`storage:` `  secretname:`	シークレットの名前 (例:**tracing-secret**)。		該当なし

パラメーター	説明	値	デフォルト値
storage: options: {}	ストレージを定義する設定オプション。		

表3.6 Elasticsearch インデックスクリーナーのパラメーター

パラメーター	説明	値	デフォルト値
storage: esIndexCleaner: enabled:	Elasticsearch ストレージを使用する場合は、デフォルトでジョブが作成され、古いトレースをインデックスからクリーンアップします。このパラメーターは、インデックスクリーナージョブを有効または無効にします。	**true**/ **false**	**true**
storage: esIndexCleaner: numberOfDays:	インデックスの削除を待機する日数。	整数値	**7**
storage: esIndexCleaner: schedule:	Elasticsearch インデックスを消去する頻度に関するスケジュールを定義します。	cron 式	"55 23 * * *"

3.2.5.5.1. Elasticsearch インスタンスの自動プロビジョニング

Jaeger カスタムリソースをデプロイする場合に、Red Hat OpenShift 分散トレースプラットフォーム Operator は、OpenShift Elasticsearch Operator を使用して、カスタムリソースファイルの **ストレージ** セクションで提供される設定に基づいて Elasticsearch クラスターを作成します。以下の設定が設定されている場合は、Red Hat 分散トレースプラットフォーム Operator は Elasticsearch をプロビジョニングします。

- **spec.storage:type** は **elasticsearch** に設定されている
- **spec.storage.elasticsearch.doNotProvision** は **false** に設定されている
- **spec.storage.options.es.server-urls** が定義されていない。つまり、Red Hat Elasticsearch Operator によってプロビジョニングされていない Elasticsearch インスタンスへの接続がない。

Elasticsearch をプロビジョニングする場には、Red Hat OpenShift 分散トレースプラットフォーム Operator は、Elasticsearch カスタムリソース **名** を Jaeger カスタムリソースの **spec.storage.elasticsearch.name** の値に設定します。**spec.storage.elasticsearch.name** に値を指定しない場合、Operator は **elasticsearch** を使用します。

制約

- namespace ごとにセルフプロビジョニングされた Elasticsearch インスタンスがある分散トレースプラットフォーム 1 つだけを使用できます。Elasticsearch クラスターは単一の 分散トレースプラットフォームインスタンスの専用のクラスターになります。
- namespace ごとに 1 つの Elasticsearch のみを使用できます。

注記

Elasticsearch を OpenShift ロギングの一部としてインストールしている場合、Red Hat OpenShift 分散トレースプラットフォーム Operator はインストールされた OpenShift Elasticsearch Operator を使用してストレージをプロビジョニングできます。

以下の設定パラメーターは、**セルフプロビジョニングされた** Elasticsearch インスタンスに対するものです。これは、OpenShift Elasticsearch Operator を使用して Red Hat OpenShift 分散トレースプラットフォーム Operator によって作成されるインスタンスです。セルフプロビジョニングされた Elasticsearch の設定オプションは、設定ファイルの **spec:storage:elasticsearch** の下で指定します。

表3.7 Elasticsearch リソース設定パラメーター

パラメーター	説明	値	デフォルト値
elasticsearch: properties: doNotProvision:	Elasticsearch インスタンスを Red Hat 分散トレースプラットフォーム Operator がプロビジョニングする必要があるかどうかを指定するために使用します。	**true**/**false**	**true**
elasticsearch: properties: name:	Elasticsearch インスタンスの名前。Red Hat OpenShift 分散トレースプラットフォーム Operator は、このパラメーターで指定された Elasticsearch インスタンスを使用して Elasticsearch に接続します。	string	**elasticsearch**
elasticsearch: nodeCount:	Elasticsearch ノードの数。高可用性を確保するには、少なくとも 3 つのノードを使用します。スプリットブレインの問題が生じる可能性があるため、2 つのノードを使用しないでください。	整数値。例: 概念実証用 = 1、最小デプロイメント = 3	3

パラメーター	説明	値	デフォルト値
elasticsearch: resources: requests: cpu:	ご使用の環境設定に基づく、要求に対する中央処理単位の数。	コアまたはミリコアで指定されます (例: 200m、0.5、1)。例: 概念実証用 = 500m、最小デプロイメント = 1	1
elasticsearch: resources: requests: memory:	ご使用の環境設定に基づく、要求に使用できるメモリー。	バイト単位で指定します (例:200Ki、50Mi、5Gi)。例: 概念実証用 = 1Gi、最小デプロイメント = 16Gi*	16Gi
elasticsearch: resources: limits: cpu:	ご使用の環境設定に基づく、中央処理単位数の制限。	コアまたはミリコアで指定されます (例: 200m、0.5、1)。例: 概念実証用 = 500m、最小デプロイメント = 1	
elasticsearch: resources: limits: memory:	ご使用の環境設定に基づく、利用可能なメモリー制限。	バイト単位で指定します (例:200Ki、50Mi、5Gi)。例: 概念実証用 = 1Gi、最小デプロイメント = 16Gi*	
elasticsearch: redundancyPolicy:	データレプリケーションポリシーは、Elasticsearch シャードをクラスター内のデータノードにレプリケートする方法を定義します。指定されていない場合、Red Hat OpenShift 分散トレースプラットフォーム Operator はノード数に基づいて最も適切なレプリケーションを自動的に判別します。	**ZeroRedundancy**(レプリカシャードなし)、**SingleRedundancy**(レプリカシャード 1 つ)、**MultipleRedundancy**(各インデックスはデータノードの半分に分散される)、**FullRedundancy**(各インデックスはクラスター内のすべてのデータノードに完全にレプリケートされます)	

パラメーター	説明	値	デフォルト値
elasticsearch: useCertManagement:	分散トレースプラットフォームが Red Hat の証明書管理機能を使用するかどうかを指定するために使用します。この機能は、OpenShift Container Platform 4.7 の Red Hat OpenShift 5.2 向けのロギングサブシステムに追加されており、新しい Jaeger デプロイメントに推奨の設定です。	**true**/**false**	**true**
	各 Elasticsearch ノードはこれより低い値のメモリー設定でも動作しますが、これは実稼働環境でのデプロイメントには推奨されません。実稼働環境で使用する場合は、デフォルトで各 Pod に割り当てる設定を 16Gi 未満にすることはできず、Pod ごとに最大 64Gi を割り当てることを推奨します。		

実稼働ストレージの例

```
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
```

永続ストレージを含むストレージの例:

```
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 永続ストレージの設定。この場合、AWS **gp2** のサイズは **5Gi** です。値の指定がない場合、分散トレースプラットフォームは **emptyDir** を使用します。OpenShift Elasticsearch Operator は、分散トレースプラットフォームインスタンスで削除されない**PersistentVolumeClaim** および **PersistentVolume** をプロビジョニングします。同じ名前および namespace で分散トレースプラットフォームインスタンスを作成する場合は、同じボリュームをマウントできます。

3.2.5.5.2. 既存の Elasticsearch インスタンスへの接続

分散トレースを使用したストレージには、既存の Elasticsearch クラスターを使用できます。**外部** Elasticsearch インスタンスとも呼ばれる既存の Elasticsearch クラスターは、Red Hat 分散トレースプラットフォーム Operator または Red Hat Operator によってインストールされなかったインスタンスです。

以下の設定が指定されている場合に、Jaeger カスタムリソースをデプロイすると、Red Hat 分散トレースプラットフォーム Operator は Elasticsearch をプロビジョニングしません。

- **spec.storage.elasticsearch.doNotProvision** が **true** に設定されている
- **spec.storage.options.es.server-urls** に値がある
- **spec.storage.elasticsearch.name** に値がある場合、または Elasticsearch インスタンス名が **elasticsearch** の場合。

Red Hat OpenShift 分散トレースプラットフォーム Operator は、**spec.storage.elasticsearch.name** で指定された Elasticsearch インスタンスを使用して Elasticsearch に接続します。

制約

- 分散トレースプラットフォームで OpenShift Container Platform ロギング Elasticsearch インスタンスを共有したり、再利用したりすることはできません。Elasticsearch クラスターは単一の分散トレースプラットフォームインスタンスの専用のクラスターになります。

注記

Red Hat は、外部 Elasticsearch インスタンスのサポートを提供しません。カスタマーポータル でテスト済み統合マトリックスを確認できます。

以下の設定パラメーターは、**外部** Elasticsearch インスタンスとして知られる、既存の Elasticsearch インスタンス向けです。この場合は、カスタムリソースファイルの **spec:storage:options:es** で、Elasticsearch の設定オプションを指定します。

表3.8 汎用 ES 設定パラメーター

パラメーター	説明	値	デフォルト値
es: server-urls:	Elasticsearch インスタンスの URL。	Elasticsearch サーバーの完全修飾ドメイン名。	**http://elasticsearch.<namespace>.svc:9200**
es: max-doc-count:	Elasticsearch クエリーから返す最大ドキュメント数。これは集約にも適用されます。**es.max-doc-count** と **es.max-num-spans** の両方を設定する場合は、Elasticsearch は 2 つの内の小さい方の値を使用します。		10000
es: max-num-spans:	[**非推奨**: 今後のリリースで削除されます。代わりに **es.max-doc-count** を使用してください。] Elasticsearch のクエリーごとに、1 度にフェッチするスパンの最大数。**es.max-num-spans** と **es.max-doc-count** の両方を設定する場合、Elasticsearch は 2 つの内の小さい方の値を使用します。		10000
es: max-span-age:	Elasticsearch のスパンの最大ルックバック。		72h0m0s
es: sniffer:	Elasticsearch のスニファー設定。クライアントはスニッフィングプロセスを使用してすべてのノードを自動的に検索します。デフォルトでは無効になっています。	**true**/ **false**	**false**

パラメーター	説明	値	デフォルト値
es: sniffer-tls-enabled:	Elasticsearch クラスターに対してスニッフィングする際に TLS を有効にするためのオプション。クライアントはスニッフィングプロセスを使用してすべてのノードを自動的に検索します。デフォルトでは無効になっています。	**true/ false**	**false**
es: timeout:	クエリーに使用されるタイムアウト。ゼロに設定するとタイムアウトはありません。		0s
es: username:	Elasticsearch で必要なユーザー名。Basic 認証は、指定されている場合に CA も読み込みます。**es.password** も参照してください。		
es: password:	Elasticsearch で必要なパスワード。**es.username** も参照してください。		
es: version:	主要な Elasticsearch バージョン。指定されていない場合、値は Elasticsearch から自動検出されます。		0

表3.9 ES データレプリケーションパラメーター

パラメーター	説明	値	デフォルト値
es: num-replicas:	Elasticsearch のインデックスごとのレプリカ数。		1
es: num-shards:	Elasticsearch のインデックスごとのシャード数。		5

表3.10 ES インデックス設定パラメーター

パラメーター	説明	値	デフォルト値
es: create-index-templates:	**true** に設定されている場合は、アプリケーションの起動時にインデックステンプレートを自動的に作成します。テンプレートが手動でインストールされる場合は、**false** に設定されます。	**true**/ **false**	**true**
es: index-prefix:	分散トレースプラットフォームインデックスのオプション接頭辞。たとえば、これを production に設定すると、production-tracing-*という名前のインデックスが作成されます。		

表3.11 ES バルクプロセッサー設定パラメーター

パラメーター	説明	値	デフォルト値
es: bulk: actions:	バルクプロセッサーがディスクへの更新のコミットを決定する前にキューに追加できる要求の数。		1000
es: bulk: flush-interval:	**time.Duration**: この後に、他のしきい値に関係なく一括要求がコミットされます。バルクプロセッサーのフラッシュ間隔を無効にするには、これをゼロに設定します。		200ms
es: bulk: size:	バルクプロセッサーがディスクへの更新をコミットするまでに一括要求が発生する可能性のあるバイト数。		5000000
es: bulk: workers:	一括要求を受信し、Elasticsearch にコミットできるワーカーの数。		1

表3.12 ES TLS 設定パラメーター

パラメーター	説明	値	デフォルト値

es:
  tls:
  ca:
	リモートサーバーの検証に使用される TLS 認証局 (CA) ファイルへのパス。		デフォルトではシステムトラストストアを使用します。

es:
  tls:
  cert:
	リモートサーバーに対するこのプロセスの特定に使用される TLS 証明書ファイルへのパス。		

es:
  tls:
  enabled:
	リモートサーバーと通信する際に、トランスポート層セキュリティー (TLS) を有効にします。デフォルトでは無効になっています。	**true**/ **false**	**false**

es:
  tls:
  key:
	リモートサーバーに対するこのプロセスの特定に使用される TLS 秘密鍵ファイルへのパス。		

es:
  tls:
  server-name:
	リモートサーバーの証明書の予想される TLS サーバー名を上書きします。		

es: token-file:	ベアラートークンが含まれるファイルへのパス。このフラグは、指定されている場合は認証局 (CA) ファイルも読み込みます。		

表3.13 ES アーカイブ設定パラメーター

パラメーター	説明	値	デフォルト値

es-archive:
  bulk:
  actions:
	バルクプロセッサーがディスクへの更新のコミットを決定する前にキューに追加できる要求の数。		0

パラメーター	説明	値	デフォルト値
es-archive: bulk: flush-interval:	**time.Duration**: この後に、他のしきい値に関係なく一括要求がコミットされます。バルクプロセッサーのフラッシュ間隔を無効にするには、これをゼロに設定します。		0s
es-archive: bulk: size:	バルクプロセッサーがディスクへの更新をコミットするまでに一括要求が発生する可能性のあるバイト数。		0
es-archive: bulk: workers:	一括要求を受信し、Elasticsearch にコミットできるワーカーの数。		0
es-archive: create-index-templates:	**true** に設定されている場合は、アプリケーションの起動時にインデックステンプレートを自動的に作成します。テンプレートが手動でインストールされる場合は、**false** に設定されます。	**true**/ **false**	**false**
es-archive: enabled:	追加ストレージを有効にします。	**true**/ **false**	**false**
es-archive: index-prefix:	分散トレースプラットフォームインデックスのオプション接頭辞。たとえば、これを production に設定すると、production-tracing-*という名前のインデックスが作成されます。		
es-archive: max-doc-count:	Elasticsearch クエリーから返す最大ドキュメント数。これは集約にも適用されます。		0

パラメーター	説明	値	デフォルト値
es-archive: max-num-spans:	[**非推奨**: 今後のリリースで削除されます。代わりに **es-archive.max-doc-count** を使用してください。] Elasticsearch のクエリーごとに、1度にフェッチするスパンの最大数。		0
es-archive: max-span-age:	Elasticsearch のスパンの最大ルックバック。		0s
es-archive: num-replicas:	Elasticsearch のインデックスごとのレプリカ数。		0
es-archive: num-shards:	Elasticsearch のインデックスごとのシャード数。		0
es-archive: password:	Elasticsearch で必要なパスワード。**es.username** も参照してください。		
es-archive: server-urls:	Elasticsearch サーバーのコンマ区切りの一覧。完全修飾 URL(例: **http://localhost:9200**) として指定される必要があります。		
es-archive: sniffer:	Elasticsearch のスニファー設定。クライアントはスニッフィングプロセスを使用してすべてのノードを自動的に検索します。デフォルトでは無効になっています。	**true**/ **false**	**false**

パラメーター	説明	値	デフォルト値
es-archive: sniffer-tls- enabled:	Elasticsearch クラスターに対してスニッフィングする際に TLS を有効にするためのオプション。クライアントはスニッフィングプロセスを使用してすべてのノードを自動的に検索します。デフォルトでは無効になっています。	**true**/ **false**	**false**
es-archive: timeout:	クエリーに使用されるタイムアウト。ゼロに設定するとタイムアウトはありません。		0s
es-archive: tls: ca:	リモートサーバーの検証に使用される TLS 認証局 (CA) ファイルへのパス。		デフォルトではシステムトラストストアを使用します。
es-archive: tls: cert:	リモートサーバーに対するこのプロセスの特定に使用される TLS 証明書ファイルへのパス。		
es-archive: tls: enabled:	リモートサーバーと通信する際に、トランスポート層セキュリティー (TLS) を有効にします。デフォルトでは無効になっています。	**true**/ **false**	**false**
es-archive: tls: key:	リモートサーバーに対するこのプロセスの特定に使用される TLS 秘密鍵ファイルへのパス。		
es-archive: tls: server-name:	リモートサーバーの証明書の予想される TLS サーバー名を上書きします。		

パラメーター	説明	値	デフォルト値
es-archive: token-file:	ベアラートークンが含まれるファイルへのパス。このフラグは、指定されている場合は認証局 (CA) ファイルも読み込みます。		
es-archive: username:	Elasticsearch で必要なユーザー名。Basic 認証は、指定されている場合に CA も読み込みます。**es-archive.password** も参照してください。		
es-archive: version:	主要な Elasticsearch バージョン。指定されていない場合、値は Elasticsearch から自動検出されます。		0

ボリュームマウントを含むストレージの例

```
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
```

以下の例は、ボリュームからマウントされる TLS CA 証明書およびシークレットに保存されるユーザー/パスワードを使用して外部 Elasticsearch クラスターを使用する Jaeger CR を示しています。

外部 Elasticsearch の例:

```
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) デフォルト namespace で実行されている Elasticsearch サービスへの URL。

(2) TLS 設定。この場合は、CA 証明書のみを使用できますが、相互 TLS を使用する場合に es.tls.key および es.tls.cert を含めることもできます。

(3) 環境変数 ES_PASSWORD および ES_USERNAME を定義するシークレット。kubectl create secret generic tracing-secret --from-literal=ES_PASSWORD=changeme --from-literal=ES_USERNAME=elastic により作成されます

(4) すべてのストレージコンポーネントにマウントされるボリュームのマウントとボリューム。

3.2.5.6. Elasticsearch を使用した証明書の管理

Red Hat Elasticsearch Operator を使用して、証明書を作成および管理できます。Red Hat Elasticsearch Operator を使用して証明書を管理すると、複数の Jaeger Collector で単一の Elasticsearch クラスターを使用することもできます。

重要

Elasticsearch を使用した証明書の管理は、テクノロジープレビュー機能のみです。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

バージョン 2.4 以降、Red Hat OpenShift 分散トレースプラットフォーム Operator は、Elasticsearch カスタムリソースで次のアノテーションを使用して、証明書の作成を Red Hat Elasticsearch Operator に委譲します。

- **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"**

ここで、**<shared-es-node-name>** は Elasticsearch ノードの名前です。たとえば、 **custom-es** という名前の Elasticsearch ノードを作成する場合に、カスタムリソースは次の例のようになります。

アノテーションを表示する Elasticsearch CR の例

```
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
```

前提条件

- OpenShift Container Platform 4.7
- Red Hat OpenShift のロギングサブシステム: 5.2
- Elasticsearch ノードと Jaeger インスタンスは同じ namespace にデプロイする必要があります。(例: **traceing-system**)。

Jaeger カスタムリソースで **spec.storage.elasticsearch.useCertManagement** を **true** に設定して、証明書管理を有効にします。

useCertManagement を示す例

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

Red Hat OpenShift 分散トレースプラットフォーム Operator は、Elasticsearch カスタムリソース **名** を Jaeger カスタムリソースの **spec.storage.elasticsearch.name** の値に設定します。

証明書は Red Hat Operator によってプロビジョニングされ、Red Hat 分散トレースプラットフォーム Operator が証明書を挿入します。

3.2.5.7. クエリー設定オプション

Query とは、ストレージからトレースを取得し、ユーザーインターフェイスをホストしてそれらを表示するサービスです。

表3.14 Query を定義するために Red Hat OpenShift 分散トレースプラットフォーム Operator によって使用されるパラメーター

パラメーター	説明	値	デフォルト値

`spec:`
`  query:`
`  replicas:`
	作成する Query レプリカの数を指定します。	整数 (例: **2**)	

表3.15 Query に渡される設定パラメーター

パラメーター	説明	値	デフォルト値

`spec:`
`  query:`
`  options: {}`
	Query サービスを定義する設定オプション。		

`options:` `  log-level:`	Query のロギングレベル。	使用できる値は、**debug**、**info**、**warn**、**error**、**fatal**、**panic** です。	

パラメーター	説明	値	デフォルト値

`options:`
`  query:`
`  base-path:`
	すべての jaeger-query HTTP ルートのベースパスは、root 以外の値に設定できます。たとえば、**/jaeger** ではすべての UI URL が **/jaeger** で開始するようになります。これは、リバースプロキシーの背後で jaeger-query を実行する場合に役立ちます。	/<path>	

Query 設定の例

```
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 設定オプション

Ingester は、Kafka トピックから読み取り、Elasticsearch ストレージバックエンドに書き込むサービスです。**allInOne** または **production** デプロイメントストラテジーを使用している場合は、Ingester サービスを設定する必要はありません。

表3.16 Ingester に渡される Jaeger パラメーター

パラメーター	説明	値

`spec:`
`  ingester:`
`  options: {}`
	Ingester サービスを定義する設定オプション。	

パラメーター	説明	値
options: deadlockInterval:	Ingester が終了するまでメッセージを待機する間隔 (秒単位または分単位) を指定します。システムの初期化中にメッセージが到達されない場合に Ingester が終了しないように、デッドロックの間隔はデフォルトで無効に (**0**に設定) されます。	分と秒 (例: **1m0s**)デフォルト値は **0** です。
options: kafka: consumer: topic:	**topic** パラメーターは、コレクターによってメッセージを生成するために使用され、Ingester によってメッセージを消費するために使用される Kafka 設定を特定します。	コンシューマーのラベル例: **jaeger-spans**
options: kafka: consumer: brokers:	メッセージを消費するために Ingester によって使用される Kafka 設定を特定します。	ブローカーのラベル (例: **my-cluster-kafka-brokers.kafka:9092**)
options: log-level:	Ingester のロギングレベル。	使用できる値は、**debug**、**info**、**warn**、**error**、**fatal**、**dpanic**、**panic** です。

ストリーミング Collector および Ingester の例

```
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. サイドカーコンテナーの挿入

Red Hat OpenShift 分散トレースプラットフォームは、アプリケーションの Pod 内のプロキシーサイドカーコンテナーを使用してエージェントを提供します。Red Hat OpenShift 分散トレースプラットフォーム Operator は、Agent サイドカーを Deployment ワークロードに挿入できます。自動のサイドカーコンテナー挿入を有効にしたり、手動で管理したりできます。

3.2.6.1. サイドカーコンテナーの自動挿入

Red Hat OpenShift 分散トレースプラットフォーム Operator は、Jaeger Agent サイドカーを Deployment ワークロードに挿入できます。サイドカーの自動挿入を有効にするには、**sidecar.jaegertracing.io/inject** アノテーションセットを文字列 **true** または **$ oc get jaegers** を実行して返される分散トレースプラットフォームインスタンス名に追加します。**true** を指定する場合、デプロイメントとして単一の分散トレースプラットフォームインスタンスのみが同じ namespace に存在する必要があります。そうでない場合に、Operator は使用する分散トレースプラットフォームインスタンスを判別することができません。デプロイメントの特定の分散トレースプラットフォームインスタンス名は、その namespace に適用される **true** よりも優先されます。

以下のスニペットは、サイドカーコンテナーを挿入する単純なアプリケーションを示しています。エージェントは、同じ namespace で利用可能な単一の分散トレースプラットフォームインスタンスを参照します。

サイドカーの自動挿入の例

```
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 文字列 **true** または Jaeger インスタンスの名前のいずれかに設定します。

サイドカーコンテナーが挿入されると、エージェントは **localhost** のデフォルトの場所でアクセスできます。

3.2.6.2. サイドカーコンテナーの手動挿入

Red Hat OpenShift 分散トレースプラットフォーム Operator は、Jaeger Agent サイドカーを Deployment ワークロードに自動的に挿入することしかできません。**Deployments** 以外 (**StatefulSets**、**DaemonSets** など) のコントローラータイプの場合、仕様で Jaeger エージェントサイドカーを手動で定義できます。

以下のスニペットは、Jaeger エージェントサイドカーのコンテナーセクションに追加できる手動の定義を示しています。

StatefulSet のサイドカー定義の例

```
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
```

その後、エージェントは localhost のデフォルトの場所でアクセスできます。

3.3. 分散トレースデータ収集の設定およびデプロイ

Red Hat OpenShift 分散トレースデータ収集 Operator は、Red Hat OpenShift 分散トレースデータ収集リソースの作成およびデプロイ時に使用されるアーキテクチャーおよび設定を定義するカスタムリソース定義 (CRD) ファイルを使用します。デフォルト設定をインストールするか、ビジネス要件に合わせてファイルを変更することができます。

3.3.1. OpenTelemetry Collector 設定オプション

重要

Red Hat OpenShift 分散トレースデータ収集 Operator は、テクノロジープレビュー機能としてのみご利用いただけます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

OpenTelemetry Collector は、テレメトリーデータにアクセスする 3 つのコンポーネントで設定されます。

- **レシーバー**: レシーバー (プッシュまたはプルベース) は、データが Collector に到達する方法です。通常、レシーバーは指定された形式のデータを受け入れ、これを内部形式に変換し、それを適用可能なパイプラインで定義されるプロセッサーおよびエクスポートャーに渡します。デフォルトでは、レシーバーは設定されていません。1 つまたは複数のレシーバーを設定する必要があります。レシーバーは 1 つまたは複数のデータソースをサポートする場合があります。

- **プロセッサー**:(オプション) プロセッサーは、受信され、エクスポートされる間のデータで実行されます。デフォルトでは、プロセッサーは有効になっていません。プロセッサーは、すべてのデータソースに対して有効にする必要があります。すべてのプロセッサーがすべてのデータソースをサポートするわけではありません。データソースによっては、複数のプロセッサーを有効にすることが推奨される場合があります。さらに、プロセッサーの順序が重要である点に気を留めることが大切です。

- **エクスポーター**: エクスポーター (プッシュまたはプルベース) は、データを 1 つまたは複数のバックエンド/宛先に送信する方法です。デフォルトでは、エクスポーターは設定されていません。1 つまたは複数のエクスポーターを設定する必要があります。エクスポーターは 1 つまたは複数のデータソースをサポートする場合があります。レポーターにはデフォルト設定が定義されている場合がありますが、多くの場合、少なくとも宛先およびセキュリティー設定を指定するための設定が必要になります。

カスタムリソース YAML ファイルで、コンポーネントのインスタンスを複数定義できます。これらのコンポーネントを設定したら、YAML ファイルの **spec.config.service** セクションで定義されたパイプラインで有効にする必要があります。ベストプラクティスとしては、必要なコンポーネントのみを有効にする必要があります。

OpenTelemetry コレクターカスタムリソースファイルの例

```
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]
```

注記

コンポーネントが設定されていても、**service** セクション内で定義されていない場合、有効になりません。

表3.17 Operator が OpenTelemetry Collector を定義するために使用するパラメーター

パラメーター	説明	値	デフォルト
`receivers:`	レシーバーは、データが Collector に到達する方法です。デフォルトでは、レシーバーは設定されていません。設定が有効とみなされるためには、少なくとも 1 つの有効なレシーバーが必要です。レシーバーは、パイプラインに追加して有効にされます。	**otlp**、**jaeger**	なし
`receivers:` `  otlp:`	**oltp** および **jaeger** レシーバーにはデフォルト設定が定義されていて、設定するにはレシーバーの名前を指定するだけで十分です。		

パラメーター	説明	値	デフォルト
processors:	プロセッサーは、受信され、エクスポートされる間のデータで実行されます。デフォルトでは、プロセッサーは有効になっていません。		なし
exporters:	エクスポーターは、データを 1 つまたは複数のバックエンド/宛先に送信します。デフォルトでは、エクスポーターは設定されていません。設定が有効とみなされるためには、少なくとも 1 つの有効なエクスポーターが必要です。エクスポーターは、パイプラインに追加して有効にされます。レポーターにはデフォルト設定が定義されている場合がありますが、多くの場合、少なくとも宛先およびセキュリティー設定を指定するための設定が必要になります。	**logging**、**jaeger**	なし
exporters: jaeger: endpoint:	セキュアな接続を確立するには、**jaeger** エクスポーターのエンドポイントは **<name>-collector-headless.<namespace>.svc** の形式にする必要があり、Jaeger デプロイメントの名前および namespace を指定します。		
exporters: jaeger: tls: ca_file:	CA 証明書へのパス。クライアントの場合、サーバー証明書を検証します。サーバーの場合、クライアント証明書を検証します。空の場合、システムのルート CA が使用されます。		

パラメーター	説明	値	デフォルト
service: pipelines:	コンポーネントは、それらを **services.pipeline** セクションのパイプラインに追加して有効にされます。		
service: pipelines: traces: receivers:	レシーバーは、それらを **service.pipelines.traces** セクションに追加してトレース用に有効にします。		なし
service: pipelines: traces: processors:	プロセッサーは、それらを **service.pipelines.traces** セクションに追加してトレース用に有効にします。		なし
service: pipelines: traces: exporters:	エクスポーターは、それらを **service.pipelines.traces** セクションに追加してトレース用に有効にします。		なし

3.4. 分散トレースのアップグレード

Operator Lifecycle Manager (OLM) は、クラスター内の Operator のインストール、アップグレード、ロールベースのアクセス制御 (RBAC) を制御します。OLM はデフォルトで OpenShift Container Platform で実行されます。OLM は利用可能な Operator のクエリーやインストールされた Operator のアップグレードを実行します。OpenShift Container Platform のアップグレードの処理方法についての詳細は、Operator Lifecycle Manager のドキュメントを参照してください。

更新時に、Red Hat OpenShift 分散トレース Operator は、管理対象の分散トレースインスタンスを Operator に関連付けられたバージョンにアップグレードします。Red Hat OpenShift 分散トレースプラットフォーム Operator の新規バージョンがインストールされるたびに、Operator によって管理されるすべての分散トレースプラットフォームアプリケーションインスタンスがその Operator のバージョンにアップグレードされます。たとえば、Operator をインストールされた 1.10 から 1.11 にアップグレードした後、Operator は実行中の分散トレースプラットフォームインスタンスをスキャンし、それらも 1.11 にアップグレードします。

OpenShift Elasticsearch Operator の更新方法に関する具体的な手順は、 Updating OpenShift Loggingを参照してください。

3.4.1. 2.0 の Operator チャネルの変更

Red Hat OpenShift 分散トレース 2.0.0 には、以下の変更が加えられています。

- Red Hat OpenShift Jaeger Operator の名前を Red Hat OpenShift 分散トレースプラットフォーム Operator に変更しました。
- 個別リリースチャンネルのサポートを停止しました。今後、Red Hat OpenShift 分散トレースプラットフォーム Operator は **stable** Operator チャネルのみをサポートします。**1.24-stable** などのメンテナンスチャネルは、今後の Operator ではサポートされなくなります。

バージョン 2.0 への更新の一環として、OpenShift Elasticsearch および Red Hat OpenShift 分散トレースプラットフォーム Operator サブスクリプションを更新する必要があります。

前提条件

- OpenShift Container Platform のバージョンが 4.6 以降である。
- OpenShift Elasticsearch Operator を更新している。
- Jaeger カスタムリソースファイルをバックアップしている。
- **cluster-admin** ロールを持つアカウントがある。(Red Hat OpenShift Dedicated を使用する場合) **dedicated-admin** ロールがあるアカウント。

重要

まだUpdating OpenShift Loggingの手順に従って OpenShift Elasticsearch Operator を更新していない場合は、Red Hat OpenShift 分散トレースプラットフォーム Operator を更新する前に更新を完了してください。

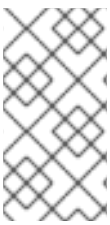
Operator チャネルの更新方法は、インストールした Operators の更新を参照してください。

3.5. 分散トレースの削除

OpenShift Container Platform クラスターから Red Hat OpenShift 分散トレースを削除する手順は、以下のとおりです。

1. Red Hat OpenShift 分散トレース Pod をすべてシャットダウンします。
2. Red Hat OpenShift 分散トレースインスタンスをすべて削除します。
3. Red Hat OpenShift 分散トレースプラットフォーム Operator を削除します。
4. Red Hat OpenShift 分散トレースデータ収集 Operator を削除します。

3.5.1. Web コンソールを使用した Red Hat OpenShift 分散トレースプラットフォームインスタンスの削除



注記

インメモリーストレージを使用するインスタンスを削除すると、すべてのデータが完全に失われます。永続ストレージ (Elasticsearch など) に保存されているデータは、Red Hat OpenShift 分散トレースプラットフォームインスタンスが削除されても削除されません。

手順

1. OpenShift Container Platform Web コンソールにログインします。
2. **Operators → Installed Operators** に移動します。
3. **Project** メニューから Operator がインストールされているプロジェクトの名前 (例: **openshift-operators**) を選択します。
4. Red Hat OpenShift distributed tracing platform Operator をクリックします。
5. **Jaeger** タブをクリックします。
6. 削除するインスタンスの横にある Options メニュー  をクリックし、**Delete Jaeger** を選択します。
7. 確認ウィンドウで **Delete** をクリックします。

3.5.2. CLI からの Red Hat OpenShift 分散トレースプラットフォームインスタンスの削除

1. OpenShift Container Platform CLI にログインします。

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

2. 分散トレースプラットフォームインスタンスを表示するには、以下のコマンドを実行します。

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

以下に例を示します。

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

Operator の名前には、接尾辞の **-operator** が付きます。以下の例は、2 つの Red Hat OpenShift 分散トレースプラットフォーム Operator と 4 つの分散トレースプラットフォームインスタンスを示しています。

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

以下のような出力が表示されるはずです。

```
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. 分散トレースプラットフォームのインスタンスを削除するには、以下のコマンドを実行します。

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

以下に例を示します。

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

4. 削除を確認するには、**oc get deployments** コマンドを再度実行します。

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

以下に例を示します。

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

以下の例のような出力が生成されて表示されるはずです。

```
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 分散トレース Operator の削除

手順

1. クラスターからの Operator の削除 の手順に従います。
   - Red Hat OpenShift 分散トレースプラットフォーム Operator を削除します。

- Red Hat OpenShift 分散トレースプラットフォーム Operator の削除後、(該当する場合は) OpenShift Elasticsearch Operator を削除します。