Plate-forme de conteneurs OpenShift 4.12

Traçage distribué

Notes sur l'installation, l'utilisation et la mise à jour du traçage distribué

Last Updated: 2023-04-26

Plate-forme de conteneurs OpenShift 4.12 Traçage distribué

Notes sur l'installation, l'utilisation et la mise à jour du traçage distribué

Notice légale

Résumé

Ce document fournit des informations sur l'utilisation du traçage distribué dans OpenShift Container Platform.

Table des matières

CHAPITRE 1. NOTES DE MISE À JOUR SUR LE TRAÇAGE DISTRIBUÉ

1.1. APERÇU DU TRAÇAGE DISTRIBUÉ

En tant que propriétaire de service, vous pouvez utiliser le traçage distribué pour instrumenter vos services afin de recueillir des informations sur votre architecture de service. Vous pouvez utiliser le traçage distribué pour la surveillance, le profilage du réseau et le dépannage de l'interaction entre les composants dans les applications modernes, cloud-natives et basées sur les microservices.

Le traçage distribué permet d'exécuter les fonctions suivantes :

- Contrôler les transactions distribuées
- Optimiser les performances et la latence
- Effectuer une analyse des causes profondes

Le traçage distribué de Red Hat OpenShift se compose de deux éléments principaux :

- **Red Hat OpenShift distributed tracing platform**- Ce composant est basé sur le projet open source Jaeger.
- **Red Hat OpenShift distributed tracing data collection**- Ce composant est basé sur le projet open source OpenTelemetry.

Ces deux composants sont basés sur les API et l'instrumentation OpenTracing, neutres vis-à-vis des fournisseurs.

1.2. RENDRE L'OPEN SOURCE PLUS INCLUSIF

Red Hat s'engage à remplacer les termes problématiques dans son code, sa documentation et ses propriétés Web. Nous commençons par ces quatre termes : master, slave, blacklist et whitelist. En raison de l'ampleur de cette entreprise, ces changements seront mis en œuvre progressivement au cours de plusieurs versions à venir. Pour plus de détails, voir le message de notre directeur technique Chris Wright.

1.3. OBTENIR DE L'AIDE

Si vous rencontrez des difficultés avec une procédure décrite dans cette documentation, ou avec OpenShift Container Platform en général, visitez le portail client de Red Hat . À partir du portail client, vous pouvez :

- Recherchez ou parcourez la base de connaissances de Red Hat qui contient des articles et des solutions relatifs aux produits Red Hat.
- Soumettre un cas d'assistance à Red Hat Support.
- Accéder à d'autres documents sur les produits.

Pour identifier les problèmes liés à votre cluster, vous pouvez utiliser Insights dans OpenShift Cluster Manager Hybrid Cloud Console. Insights fournit des détails sur les problèmes et, le cas échéant, des informations sur la manière de les résoudre.

Si vous avez une suggestion pour améliorer cette documentation ou si vous avez trouvé une erreur, soumettez un problème Jira pour le composant de documentation le plus pertinent. Veuillez fournir des détails spécifiques, tels que le nom de la section et la version d'OpenShift Container Platform.

1.4. NOUVELLES FONCTIONNALITÉS ET AMÉLIORATIONS

Cette version apporte des améliorations concernant les composants et concepts suivants.

1.4.1. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.7

Cette version de Red Hat OpenShift distributed tracing traite les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

1.4.1.1. Versions de composants prises en charge dans Red Hat OpenShift distributed tracing version 2.7

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.39
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.63.1

1.4.2. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.6

Cette version de Red Hat OpenShift distributed tracing traite les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

1.4.2.1. Versions de composants prises en charge dans Red Hat OpenShift distributed tracing version 2.6

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.38
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.60

1.4.3. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.5

Cette version de Red Hat OpenShift distributed tracing traite les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

Cette version introduit la prise en charge de l'ingestion du protocole OpenTelemetry (OTLP) dans la plateforme de traçage distribuée Operator de Red Hat OpenShift. L'opérateur active désormais automatiquement les ports OTLP :

- Le port 4317 est utilisé pour le protocole OTLP gRPC.
- Le port 4318 est utilisé pour le protocole HTTP OTLP.

Cette version ajoute également la prise en charge de la collecte des attributs de ressources Kubernetes à l'opérateur de collecte de données de traçage distribué Red Hat OpenShift.

1.4.3.1. Versions de composants prises en charge dans Red Hat OpenShift distributed tracing version 2.5

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.36
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.56

1.4.4. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.4

Cette version de Red Hat OpenShift distributed tracing traite les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

Cette version ajoute également la prise en charge du provisionnement automatique des certificats à l'aide de Red Hat Elasticsearch Operator.

- Auto-approvisionnement, ce qui signifie l'utilisation de l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift pour appeler l'opérateur Red Hat Elasticsearch pendant l'installation. L'auto-provisionnement est entièrement pris en charge dans cette version.
- La création de l'instance Elasticsearch et des certificats en premier lieu, puis la configuration de la plateforme de traçage distribuée pour utiliser le certificat est un aperçu technologique pour cette version.

NOTE

Lors de la mise à niveau vers Red Hat OpenShift distributed tracing 2.4, l'opérateur recrée l'instance Elasticsearch, ce qui peut prendre cinq à dix minutes. Le traçage distribué sera en panne et indisponible pendant cette période.

1.4.4.1. Versions de composants prises en charge dans Red Hat OpenShift distributed tracing version 2.4

Opérateur	Composant	Version

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.34.1
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.49

1.4.5. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.3.1

Cette version de Red Hat OpenShift distributed tracing traite les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

1.4.5.1. Versions de composants prises en charge dans la version 2.3.1 de Red Hat OpenShift distributed tracing

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.30.2
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.44.1-1

1.4.6. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.3.0

Cette version de Red Hat OpenShift distributed tracing traite les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

Avec cette version, la plateforme de traçage distribuée Red Hat OpenShift Operator est désormais installée par défaut dans l'espace de noms **openshift-distributed-tracing**. Avant cette mise à jour, l'installation par défaut était dans l'espace de noms **openshift-operators**.

1.4.6.1. Versions de composants prises en charge dans la version 2.3.0 de Red Hat OpenShift distributed tracing

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.30.1
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.44.0

1.4.7. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.2.0

Cette version de Red Hat OpenShift distributed tracing traite les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

1.4.7.1. Versions de composants prises en charge dans Red Hat OpenShift distributed tracing version 2.2.0

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.30.0
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.42.0

1.4.8. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.1.0

Cette version de Red Hat OpenShift distributed tracing traite les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

1.4.8.1. Versions de composants prises en charge dans Red Hat OpenShift distributed tracing version 2.1.0

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.29.1
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.41.1

1.4.9. Nouvelles fonctionnalités et améliorations Red Hat OpenShift distributed tracing 2.0.0

Cette version marque le changement de marque de Red Hat OpenShift Jaeger en Red Hat OpenShift distributed tracing. Cette version comprend les changements, ajouts et améliorations suivants :

- Le traçage distribué de Red Hat OpenShift est désormais constitué des deux composants principaux suivants :
  - **Red Hat OpenShift distributed tracing platform** - Ce composant est basé sur le projet open source Jaeger.
  - **Red Hat OpenShift distributed tracing data collection** - Ce composant est basé sur le projet open source OpenTelemetry.

- Mise à jour de la plateforme de traçage distribuée Red Hat OpenShift Operator vers Jaeger 1.28. À l'avenir, Red Hat OpenShift distributed tracing ne prendra en charge que le canal **stable** Operator. Les canaux pour les versions individuelles ne sont plus pris en charge.
- Présente un nouvel opérateur de collecte de données de traçage distribué Red Hat OpenShift basé sur OpenTelemetry 0.33. Notez que cet opérateur est une fonctionnalité de l'aperçu technologique.
- Ajout de la prise en charge du protocole OpenTelemetry (OTLP) au service Query.
- Introduit une nouvelle icône de traçage distribué qui apparaît dans l'OperatorHub d'OpenShift.
- Comprend des mises à jour régulières de la documentation pour prendre en charge le changement de nom et les nouvelles fonctionnalités.

Cette version aborde également les vulnérabilités et expositions communes (CVE) et les corrections de bogues.

1.4.9.1. Versions de composants prises en charge dans Red Hat OpenShift distributed tracing version 2.0.0

Opérateur	Composant	Version
Plateforme de traçage distribuée Red Hat OpenShift	Jaeger	1.28.0
Collecte de données de traçage distribué Red Hat OpenShift	OpenTelemetry	0.33.0

1.5. APERÇU TECHNOLOGIQUE DE RED HAT OPENSHIFT SUR LE TRAÇAGE DISTRIBUÉ

IMPORTANT

Les fonctionnalités de l'aperçu technologique ne sont pas prises en charge par les accords de niveau de service (SLA) de production de Red Hat et peuvent ne pas être complètes sur le plan fonctionnel. Red Hat ne recommande pas de les utiliser en production. Ces fonctionnalités offrent un accès anticipé aux fonctionnalités des produits à venir, ce qui permet aux clients de tester les fonctionnalités et de fournir un retour d'information pendant le processus de développement.

Pour plus d'informations sur la portée de l'assistance des fonctionnalités de l'aperçu technologique de Red Hat, voir Portée de l'assistance des fonctionnalités de l'aperçu technologique.

1.5.1. Red Hat OpenShift distributed tracing 2.4.0 Technology Preview

Cette version ajoute également la prise en charge du provisionnement automatique des certificats à l'aide de Red Hat Elasticsearch Operator.

- Auto-approvisionnement, ce qui signifie l'utilisation de l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift pour appeler l'opérateur Red Hat Elasticsearch pendant l'installation. L'auto-provisionnement est entièrement pris en charge dans cette version.

- La création de l'instance Elasticsearch et des certificats en premier lieu, puis la configuration de la plateforme de traçage distribuée pour utiliser le certificat est un aperçu technologique pour cette version.

1.5.2. Red Hat OpenShift distributed tracing 2.2.0 Technology Preview

Les composants non pris en charge d'OpenTelemetry Collector inclus dans la version 2.1 ont été supprimés.

1.5.3. Red Hat OpenShift distributed tracing 2.1.0 Technology Preview

Cette version introduit un changement radical dans la manière de configurer les certificats dans le fichier de ressources personnalisées d'OpenTelemetry. Dans la nouvelle version, **ca_file** est déplacé sous **tls** dans la ressource personnalisée, comme le montrent les exemples suivants.

Configuration du fichier CA pour OpenTelemetry version 0.33

```
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"
```

Configuration du fichier CA pour OpenTelemetry version 0.41.1

```
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 Technology Preview

Cette version inclut l'ajout de la collecte de données de traçage distribué Red Hat OpenShift, que vous installez à l'aide de l'opérateur de collecte de données de traçage distribué Red Hat OpenShift. La collecte de données de traçage distribué Red Hat OpenShift est basée sur les API et l'instrumentation OpenTelemetry.

La collecte de données de traçage distribué de Red Hat OpenShift comprend l'opérateur et le collecteur OpenTelemetry. Le Collecteur peut être utilisé pour recevoir des traces dans le protocole OpenTelemetry ou Jaeger et envoyer les données de trace au traçage distribué de Red Hat OpenShift. Les autres capacités du collecteur ne sont pas prises en charge à l'heure actuelle.

L'OpenTelemetry Collector permet aux développeurs d'instrumenter leur code à l'aide d'API indépendantes du fournisseur, évitant ainsi le verrouillage du fournisseur et permettant un écosystème croissant d'outils d'observabilité.

1.6. RED HAT OPENSHIFT DISTRIBUTED TRACING KNOWN ISSUES (PROBLÈMES CONNUS)

Ces limitations existent dans le traçage distribué de Red Hat OpenShift :

- Apache Spark n'est pas pris en charge.
- Le déploiement en continu via AMQ/Kafka n'est pas pris en charge sur les systèmes IBM Z et IBM Power.

Ce sont les problèmes connus pour Red Hat OpenShift distributed tracing :

- OBSDA-220 Dans certains cas, si vous essayez d'extraire une image à l'aide de la collecte de données de traçage distribuées, l'extraction de l'image échoue et un message d'erreur **Failed to pull image** s'affiche. Il n'y a pas de solution à ce problème.
- TRACING-2057 L'API Kafka a été mise à jour sur **v1beta2** pour prendre en charge le Strimzi Kafka Operator 0.23.0. Cependant, cette version de l'API n'est pas prise en charge par AMQ Streams 1.6.3. Si vous avez l'environnement suivant, vos services Jaeger ne seront pas mis à niveau et vous ne pourrez pas créer de nouveaux services Jaeger ou modifier des services Jaeger existants :
  - Canal de l'opérateur Jaeger : **1.17.x stable** ou **1.20.x stable**
  - Canal de l'opérateur AMQ Streams : **amq-streams-1.6.x**
    Pour résoudre ce problème, changez le canal d'abonnement de votre opérateur AMQ Streams sur **amq-streams-1.7.x** ou **stable**.

1.7. RED HAT OPENSHIFT A CORRIGÉ DES PROBLÈMES DE TRAÇAGE DISTRIBUÉ

- OSSM-1910 En raison d'un problème introduit dans la version 2.6, les connexions TLS ne pouvaient pas être établies avec OpenShift Container Platform Service Mesh. Cette mise à jour résout le problème en changeant les noms des ports de service pour qu'ils correspondent aux conventions utilisées par OpenShift Container Platform Service Mesh et Istio.
- OBSDA-208 Avant cette mise à jour, les limites de ressources par défaut de 200 m de CPU et 256 m de mémoire pouvaient entraîner un redémarrage continu de la collecte des données de traçage distribué sur les grands clusters. Cette mise à jour résout le problème en supprimant ces limites de ressources.
- OBSDA-222 Avant cette mise à jour, des spans pouvaient être abandonnés dans la plateforme de traçage distribuée OpenShift Container Platform. Pour aider à prévenir ce problème, cette version met à jour les dépendances de version.
- TRACING-2337 Jaeger enregistre un message d'avertissement répétitif dans les journaux de Jaeger, similaire à ce qui suit :

```
{"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}

Ce problème a été résolu en exposant uniquement le port HTTP(S) du service de requête, et non le port gRPC.

- TRACING-2009 L'opérateur Jaeger a été mis à jour pour inclure le support de l'opérateur Strimzi Kafka 0.23.0.

- TRACING-1907 L'injection du sidecar de l'agent Jaeger échouait en raison de l'absence de cartes de configuration dans l'espace de noms de l'application. Les cartes de configuration étaient automatiquement supprimées en raison d'un paramètre de champ **OwnerReference** incorrect et, par conséquent, les pods d'application ne dépassaient pas l'étape "ContainerCreating". Les paramètres incorrects ont été supprimés.

- TRACING-1725 Suivi de TRACING-1631. Correction supplémentaire pour s'assurer que les certificats Elasticsearch sont correctement réconciliés lorsqu'il y a plusieurs instances de production Jaeger, utilisant le même nom mais dans des espaces de noms différents. Voir aussi BZ-1918920.

- TRACING-1631 Plusieurs instances de production Jaeger, utilisant le même nom mais dans des espaces de noms différents, causant un problème de certificat Elasticsearch. Lorsque plusieurs maillages de services étaient installés, toutes les instances Elasticsearch de Jaeger avaient le même secret Elasticsearch au lieu de secrets individuels, ce qui empêchait l'opérateur Elasticsearch d'OpenShift de communiquer avec tous les clusters Elasticsearch.

- TRACING-1300 Échec de la connexion entre l'agent et le collecteur lors de l'utilisation d'Istio sidecar. Une mise à jour de l'opérateur Jaeger a activé la communication TLS par défaut entre un agent Jaeger sidecar et le collecteur Jaeger.

- TRACING-1208 Authentification "500 Internal Error" lors de l'accès à l'interface utilisateur de Jaeger. Lorsque j'essaie de m'authentifier à l'interface utilisateur en utilisant OAuth, j'obtiens une erreur 500 parce que le sidecar oauth-proxy ne fait pas confiance à l'ensemble d'autorités de certification personnalisées défini lors de l'installation avec l'adresse **additionalTrustBundle**.

- TRACING-1166 Il n'est actuellement pas possible d'utiliser la stratégie de streaming Jaeger dans un environnement déconnecté. Lorsqu'un cluster Kafka est en cours de provisionnement, il en résulte une erreur : **Failed to pull image registry.redhat.io/amq7/amq-streams-kafka-24-rhel7@sha256:f9ceca004f1b7dccb3b82d9a8027961f9fe4104e0ed69752c0bdd8078b4a1076**.

- TRACING-809 Jaeger Ingester est incompatible avec Kafka 2.3. Lorsqu'il y a deux ou plusieurs instances de Jaeger Ingester et un trafic suffisant, il génère continuellement des messages de rééquilibrage dans les journaux. Ceci est dû à une régression dans Kafka 2.3 qui a été corrigée dans Kafka 2.3.1. Pour plus d'informations, voir Jaegertracing-1819.

- BZ-1918920/LOG-1619 Les pods Elasticsearch ne sont pas redémarrés automatiquement après une mise à jour.
  Solution : Redémarrer les pods manuellement.

CHAPITRE 2. ARCHITECTURE DE TRAÇAGE DISTRIBUÉE

2.1. ARCHITECTURE DE TRAÇAGE DISTRIBUÉE

Chaque fois qu'un utilisateur effectue une action dans une application, une requête est exécutée par l'architecture qui peut nécessiter la participation de dizaines de services différents pour produire une réponse. Red Hat OpenShift distributed tracing vous permet d'effectuer un traçage distribué, qui enregistre le chemin d'une requête à travers les différents microservices qui composent une application.

Distributed tracing est une technique utilisée pour relier les informations relatives à différentes unités de travail - généralement exécutées dans différents processus ou hôtes - afin de comprendre toute une chaîne d'événements dans une transaction distribuée. Les développeurs peuvent visualiser les flux d'appels dans les grandes architectures de microservices grâce au traçage distribué. Il est précieux pour comprendre la sérialisation, le parallélisme et les sources de latence.

Le traçage distribué de Red Hat OpenShift enregistre l'exécution de requêtes individuelles à travers l'ensemble de la pile de microservices et les présente sous forme de traces. Un *trace* est un chemin de données/d'exécution à travers le système. Une trace de bout en bout est composée d'une ou plusieurs travées.

Un *span* représente une unité logique de travail dans le traçage distribué de Red Hat OpenShift qui possède un nom d'opération, l'heure de début de l'opération et la durée, ainsi que potentiellement des balises et des journaux. Les portées peuvent être imbriquées et ordonnées pour modéliser les relations de cause à effet.

2.1.1. Aperçu du traçage distribué

En tant que propriétaire de service, vous pouvez utiliser le traçage distribué pour instrumenter vos services afin de recueillir des informations sur votre architecture de service. Vous pouvez utiliser le traçage distribué pour la surveillance, le profilage du réseau et le dépannage de l'interaction entre les composants dans les applications modernes, cloud-natives et basées sur les microservices.

Le traçage distribué permet d'exécuter les fonctions suivantes :

- Contrôler les transactions distribuées
- Optimiser les performances et la latence
- Effectuer une analyse des causes profondes

Le traçage distribué de Red Hat OpenShift se compose de deux éléments principaux :

- **Red Hat OpenShift distributed tracing platform** - Ce composant est basé sur le projet open source Jaeger.
- **Red Hat OpenShift distributed tracing data collection** - Ce composant est basé sur le projet open source OpenTelemetry.

Ces deux composants sont basés sur les API et l'instrumentation OpenTracing, neutres vis-à-vis des fournisseurs.

2.1.2. Fonctionnalités de traçage distribué de Red Hat OpenShift

Le traçage distribué de Red Hat OpenShift offre les capacités suivantes :

- Intégration avec Kiali - Lorsque la configuration est correcte, vous pouvez visualiser les données de traçage distribuées à partir de la console Kiali.
- Grande évolutivité - Le back-end de traçage distribué est conçu pour ne pas avoir de point de défaillance unique et pour s'adapter aux besoins de l'entreprise.
- Propagation du contexte distribué - Permet de relier des données provenant de différents composants afin de créer une trace complète de bout en bout.
- Compatibilité ascendante avec Zipkin - Le traçage distribué Red Hat OpenShift possède des API qui lui permettent d'être utilisé comme un remplacement direct de Zipkin, mais Red Hat ne prend pas en charge la compatibilité avec Zipkin dans cette version.

2.1.3. Architecture de traçage distribuée Red Hat OpenShift

Le traçage distribué de Red Hat OpenShift est constitué de plusieurs composants qui fonctionnent ensemble pour collecter, stocker et afficher les données de traçage.

- **Red Hat OpenShift distributed tracing platform** - Ce composant est basé sur le projet open source Jaeger.
  - **Client** (Client Jaeger, Tracer, Reporter, application instrumentée, bibliothèques clientes) - Les clients de la plateforme de traçage distribué sont des implémentations de l'API OpenTracing spécifiques à chaque langue. Ils peuvent être utilisés pour instrumenter des applications pour le traçage distribué, soit manuellement, soit avec une variété de frameworks open source existants, tels que Camel (Fuse), Spring Boot (RHOAR), MicroProfile (RHOAR/Thorntail), Wildfly (EAP), et bien d'autres, qui sont déjà intégrés à OpenTracing.
  - **Agent** (Agent Jaeger, file d'attente du serveur, travailleurs du processeur) - L'agent de la plate-forme de traçage distribuée est un démon de réseau qui écoute les données envoyées via le protocole UDP (User Datagram Protocol), qu'il met en lots et envoie au collecteur. L'agent est censé être placé sur le même hôte que l'application instrumentée. Cela se fait généralement en ayant un sidecar dans les environnements de conteneurs tels que Kubernetes.
  - **Jaeger Collector** (Collecteur, file d'attente, travailleurs) - Comme l'agent Jaeger, le collecteur Jaeger reçoit les travées et les place dans une file d'attente interne en vue de leur traitement. Cela permet au collecteur Jaeger de retourner immédiatement au client/à l'agent au lieu d'attendre que la travée se rende au stockage.
  - **Storage** (Magasin de données) - Les collecteurs ont besoin d'un backend de stockage persistant. La plateforme de traçage distribuée Red Hat OpenShift dispose d'un mécanisme enfichable pour le stockage de la portée. Notez que pour cette version, le seul stockage pris en charge est Elasticsearch.
  - **Query** (Service d'interrogation) - L'interrogation est un service qui permet d'extraire des traces du stockage.
  - **Ingester** (Ingester Service) - Le traçage distribué Red Hat OpenShift peut utiliser Apache Kafka comme tampon entre le collecteur et le stockage Elasticsearch. Ingester est un service qui lit les données de Kafka et les écrit dans le backend de stockage Elasticsearch.
  - **Jaeger Console** - L'interface utilisateur de la plateforme de traçage distribué Red Hat OpenShift vous permet de visualiser vos données de traçage distribué. Sur la page Recherche, vous pouvez trouver des traces et explorer les détails des portées qui composent une trace individuelle.

- **Red Hat OpenShift distributed tracing data collection**- Ce composant est basé sur le projet open source OpenTelemetry.
    - **OpenTelemetry Collector** - Le collecteur OpenTelemetry permet de recevoir, de traiter et d'exporter des données de télémétrie sans tenir compte des fournisseurs. Le collecteur OpenTelemetry prend en charge les formats de données d'observabilité open-source, par exemple Jaeger et Prometheus, en les envoyant à un ou plusieurs back-ends open-source ou commerciaux. Le collecteur est l'emplacement par défaut où les bibliothèques d'instrumentation exportent leurs données de télémétrie.

CHAPITRE 3. INSTALLATION DE TRAÇAGE DISTRIBUÉ

3.1. INSTALLATION DU TRAÇAGE DISTRIBUÉ

Vous pouvez installer le traçage distribué de Red Hat OpenShift sur OpenShift Container Platform de deux manières :

- Vous pouvez installer Red Hat OpenShift distributed tracing dans le cadre de Red Hat OpenShift Service Mesh. Le traçage distribué est inclus par défaut dans l'installation de Service Mesh. Pour installer le traçage distribué Red Hat OpenShift dans le cadre d'un Service Mesh, suivez les instructions d'installation de Red Hat Service Mesh . Vous devez installer Red Hat OpenShift distributed tracing dans le même espace de noms que votre service mesh, c'est-à-dire que les ressources **ServiceMeshControlPlane** et Red Hat OpenShift distributed tracing doivent être dans le même espace de noms.

- Si vous ne souhaitez pas installer un maillage de services, vous pouvez utiliser les opérateurs de traçage distribué Red Hat OpenShift pour installer le traçage distribué de manière autonome. Pour installer Red Hat OpenShift distributed tracing sans service mesh, utilisez les instructions suivantes.

3.1.1. Conditions préalables

Avant d'installer Red Hat OpenShift distributed tracing, passez en revue les activités d'installation et assurez-vous que vous remplissez les conditions préalables :

- Posséder un abonnement OpenShift Container Platform actif sur votre compte Red Hat. Si vous n'avez pas d'abonnement, contactez votre représentant commercial pour plus d'informations.

- Consultez la présentation de OpenShift Container Platform 4.12 .

- Installez OpenShift Container Platform 4.12.
  - Installer OpenShift Container Platform 4.12 sur AWS
  - Installer OpenShift Container Platform 4.12 sur AWS fourni par l'utilisateur
  - Installer OpenShift Container Platform 4.12 sur du métal nu
  - Installer OpenShift Container Platform 4.12 sur vSphere

- Installez la version de l'OpenShift CLI (**oc**) qui correspond à la version de votre OpenShift Container Platform et ajoutez-la à votre chemin d'accès.

- Un compte avec le rôle **cluster-admin**.

3.1.2. Aperçu de l'installation du traçage distribué de Red Hat OpenShift

Les étapes d'installation du traçage distribué de Red Hat OpenShift sont les suivantes :

- Examinez la documentation et déterminez votre stratégie de déploiement.

- Si votre stratégie de déploiement nécessite un stockage persistant, installez l'OpenShift Elasticsearch Operator via l'OperatorHub.

- Installez la plateforme de traçage distribuée Red Hat OpenShift Operator via OperatorHub.
- Modifiez le fichier YAML des ressources personnalisées pour soutenir votre stratégie de déploiement.
- Déployez une ou plusieurs instances de la plateforme de traçage distribuée Red Hat OpenShift dans votre environnement OpenShift Container Platform.

3.1.3. Installation de l'opérateur OpenShift Elasticsearch

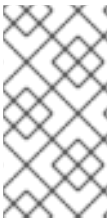
Le déploiement par défaut de la plateforme de traçage distribué Red Hat OpenShift utilise le stockage en mémoire car il est conçu pour être installé rapidement pour ceux qui évaluent le traçage distribué Red Hat OpenShift, font des démonstrations ou utilisent la plateforme de traçage distribué Red Hat OpenShift dans un environnement de test. Si vous prévoyez d'utiliser la plateforme de traçage distribué Red Hat OpenShift en production, vous devez installer et configurer une option de stockage persistant, dans ce cas, Elasticsearch.

Conditions préalables

- Vous avez accès à la console web de OpenShift Container Platform.
- Vous avez accès au cluster en tant qu'utilisateur avec le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.

AVERTISSEMENT

N'installez pas les versions communautaires des opérateurs. Les opérateurs communautaires ne sont pas pris en charge.



NOTE

Si vous avez déjà installé l'OpenShift Elasticsearch Operator dans le cadre d'OpenShift Logging, vous n'avez pas besoin d'installer à nouveau l'OpenShift Elasticsearch Operator. L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift crée l'instance Elasticsearch à l'aide de l'opérateur OpenShift Elasticsearch installé.

Procédure

1. Connectez-vous à la console web de OpenShift Container Platform en tant qu'utilisateur avec le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.
2. Naviguez jusqu'à **Operators → OperatorHub**.
3. Tapez **Elasticsearch** dans la boîte de filtre pour trouver l'OpenShift Elasticsearch Operator.
4. Cliquez sur le site **OpenShift Elasticsearch Operator** fourni par Red Hat pour afficher des informations sur l'opérateur.
5. Cliquez sur **Install**.

6. Sur la page **Install Operator**, sélectionnez le canal de mise à jour **stable**. Ce canal met automatiquement à jour votre opérateur au fur et à mesure que de nouvelles versions sont publiées.

7. Accepter le projet par défaut **All namespaces on the cluster (default)** L'opérateur est ainsi installé dans le projet par défaut **openshift-operators-redhat** et mis à la disposition de tous les projets du cluster.

NOTE

L'installation d'Elasticsearch nécessite l'espace de noms **openshift-operators-redhat** pour l'opérateur OpenShift Elasticsearch. Les autres opérateurs de traçage distribués de Red Hat OpenShift sont installés dans l'espace de noms **openshift-operators**.

- Accepter la stratégie d'approbation par défaut **Automatic**. En acceptant la stratégie par défaut, lorsqu'une nouvelle version de cet opérateur est disponible, Operator Lifecycle Manager (OLM) met automatiquement à niveau l'instance en cours d'exécution de votre opérateur sans intervention humaine. Si vous sélectionnez **Manual** updates, lorsqu'une nouvelle version d'un opérateur est disponible, OLM crée une demande de mise à jour. En tant qu'administrateur de cluster, vous devez alors approuver manuellement cette demande de mise à jour pour que l'opérateur soit mis à jour avec la nouvelle version.

NOTE

La stratégie d'approbation **Manual** exige qu'un utilisateur disposant des informations d'identification appropriées approuve le processus d'installation et d'abonnement de l'opérateur.

8. Cliquez sur **Install**.

9. Sur la page **Installed Operators**, sélectionnez le projet **openshift-operators-redhat**. Attendez de voir que l'OpenShift Elasticsearch Operator affiche le statut "InstallSucceeded" avant de continuer.

3.1.4. Installation de la plateforme de traçage distribuée Red Hat OpenShift Opérateur

Pour installer Red Hat OpenShift distributed tracing platform, vous utilisez OperatorHub pour installer l'opérateur Red Hat OpenShift distributed tracing platform.

Par défaut, l'opérateur est installé dans le projet **openshift-operators**.

Conditions préalables

- Vous avez accès à la console web de OpenShift Container Platform.
- Vous avez accès au cluster en tant qu'utilisateur avec le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.
- Si vous avez besoin d'un stockage persistant, vous devez également installer l'Opérateur OpenShift Elasticsearch avant d'installer l'Opérateur de la plateforme de traçage distribuée Red Hat OpenShift.

AVERTISSEMENT

N'installez pas les versions communautaires des opérateurs. Les opérateurs communautaires ne sont pas pris en charge.

Procédure

1. Connectez-vous à la console web de OpenShift Container Platform en tant qu'utilisateur avec le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.
2. Naviguez jusqu'à **Operators → OperatorHub**.
3. Tapez **distributed tracing platform** dans le filtre pour localiser l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift.
4. Cliquez sur le site **Red Hat OpenShift distributed tracing platform Operator** fourni par Red Hat pour afficher des informations sur l'opérateur.
5. Cliquez sur **Install**.
6. Sur la page **Install Operator**, sélectionnez le canal de mise à jour **stable**. Ce canal met automatiquement à jour votre opérateur au fur et à mesure que de nouvelles versions sont publiées.
7. Accepter le projet par défaut **All namespaces on the cluster (default)** L'opérateur est ainsi installé dans le projet par défaut **openshift-operators** et mis à la disposition de tous les projets du cluster.
   - Accepter la stratégie d'approbation par défaut **Automatic**. En acceptant la stratégie par défaut, lorsqu'une nouvelle version de cet opérateur est disponible, Operator Lifecycle Manager (OLM) met automatiquement à niveau l'instance en cours d'exécution de votre opérateur sans intervention humaine. Si vous sélectionnez **Manual** updates, lorsqu'une nouvelle version d'un opérateur est disponible, OLM crée une demande de mise à jour. En tant qu'administrateur de cluster, vous devez alors approuver manuellement cette demande de mise à jour pour que l'opérateur soit mis à jour avec la nouvelle version.

NOTE

La stratégie d'approbation **Manual** exige qu'un utilisateur disposant des informations d'identification appropriées approuve le processus d'installation et d'abonnement de l'opérateur.

8. Cliquez sur **Install**.
9. Naviguez jusqu'à **Operators → Installed Operators**.
10. Sur la page **Installed Operators**, sélectionnez le projet **openshift-operators**. Attendez de voir que l'opérateur de la plate-forme de traçage distribuée Red Hat OpenShift affiche un état de "Réussi" avant de continuer.

3.1.5. Installation de l'opérateur de collecte de données de traçage distribué Red Hat OpenShift

IMPORTANT

L'opérateur de collecte de données de traçage distribué de Red Hat OpenShift est une fonctionnalité d'aperçu technologique uniquement. Les fonctionnalités de l'aperçu technologique ne sont pas prises en charge par les accords de niveau de service (SLA) de production de Red Hat et peuvent ne pas être complètes sur le plan fonctionnel. Red Hat ne recommande pas de les utiliser en production. Ces fonctionnalités offrent un accès anticipé aux fonctionnalités des produits à venir, ce qui permet aux clients de tester les fonctionnalités et de fournir un retour d'information pendant le processus de développement.

Pour plus d'informations sur la portée de l'assistance des fonctionnalités de l'aperçu technologique de Red Hat, voir Portée de l'assistance des fonctionnalités de l'aperçu technologique.

Pour installer Red Hat OpenShift distributed tracing data collection, vous utilisez OperatorHub pour installer l'opérateur Red Hat OpenShift distributed tracing data collection.

Par défaut, l'opérateur est installé dans le projet **openshift-operators**.

Conditions préalables

- Vous avez accès à la console web de OpenShift Container Platform.
- Vous avez accès au cluster en tant qu'utilisateur avec le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.

AVERTISSEMENT

N'installez pas les versions communautaires des opérateurs. Les opérateurs communautaires ne sont pas pris en charge.

Procédure

1. Connectez-vous à la console web de OpenShift Container Platform en tant qu'utilisateur avec le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.
2. Naviguez jusqu'à **Operators → OperatorHub**.
3. Tapez **distributed tracing data collection** dans le filtre pour localiser l'opérateur de collecte de données de traçage distribué Red Hat OpenShift.
4. Cliquez sur le site **Red Hat OpenShift distributed tracing data collection Operator** fourni par Red Hat pour afficher des informations sur l'opérateur.
5. Cliquez sur **Install**.

6. Sur la page **Install Operator**, acceptez le canal de mise à jour par défaut **stable**. Ce canal met automatiquement à jour votre opérateur au fur et à mesure que de nouvelles versions sont publiées.

7. Accepter le projet par défaut **All namespaces on the cluster (default)** L'opérateur est ainsi installé dans le projet par défaut **openshift-operators** et mis à la disposition de tous les projets du cluster.

8. Accepter la stratégie d'approbation par défaut **Automatic**. En acceptant la stratégie par défaut, lorsqu'une nouvelle version de cet opérateur est disponible, Operator Lifecycle Manager (OLM) met automatiquement à niveau l'instance en cours d'exécution de votre opérateur sans intervention humaine. Si vous sélectionnez **Manual** updates, lorsqu'une nouvelle version d'un opérateur est disponible, OLM crée une demande de mise à jour. En tant qu'administrateur de cluster, vous devez alors approuver manuellement cette demande de mise à jour pour que l'opérateur soit mis à jour avec la nouvelle version.

NOTE

La stratégie d'approbation **Manual** exige qu'un utilisateur disposant des informations d'identification appropriées approuve le processus d'installation et d'abonnement de l'opérateur.

9. Cliquez sur **Install**.

10. Naviguez jusqu'à **Operators → Installed Operators**.

11. Sur la page **Installed Operators**, sélectionnez le projet **openshift-operators**. Attendez de voir que l'opérateur de collecte de données de traçage distribué de Red Hat OpenShift affiche un état de "Réussi" avant de continuer.

3.2. CONFIGURER ET DÉPLOYER LE TRAÇAGE DISTRIBUÉ

L'opérateur de plateforme de traçage distribuée Red Hat OpenShift utilise un fichier de définition de ressource personnalisée (CRD) qui définit l'architecture et les paramètres de configuration à utiliser lors de la création et du déploiement des ressources de la plateforme de traçage distribuée. Vous pouvez soit installer la configuration par défaut, soit modifier le fichier pour mieux répondre aux exigences de votre entreprise.

La plateforme de traçage distribuée Red Hat OpenShift dispose de stratégies de déploiement prédéfinies. Vous spécifiez une stratégie de déploiement dans le fichier de ressources personnalisées. Lorsque vous créez une instance de plateforme de traçage distribuée, l'opérateur utilise ce fichier de configuration pour créer les objets nécessaires au déploiement.

Fichier de ressources personnalisées Jaeger indiquant la stratégie de déploiement

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

(1) L'opérateur de plateforme de traçage distribuée Red Hat OpenShift prend actuellement en charge les stratégies de déploiement suivantes :

- **allInOne** (Par défaut) - Cette stratégie est destinée au développement, aux tests et aux démonstrations ; elle n'est pas destinée à une utilisation en production. Les principaux composants du backend, l'agent, le collecteur et le service de requête, sont tous regroupés dans un seul exécutable qui est configuré, par défaut, pour utiliser le stockage en mémoire.

NOTE

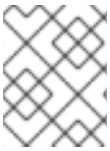
Le stockage en mémoire n'est pas persistant, ce qui signifie que si l'instance de la plateforme de traçage distribuée s'arrête, redémarre ou est remplacée, vos données de traçage seront perdues. De plus, le stockage en mémoire ne peut pas être mis à l'échelle, puisque chaque module possède sa propre mémoire. Pour le stockage persistant, vous devez utiliser les stratégies **production** ou **streaming**, qui utilisent Elasticsearch comme stockage par défaut.

- **production** - La stratégie de production est destinée aux environnements de production, où le stockage à long terme des données de traçage est important, et où une architecture plus évolutive et hautement disponible est nécessaire. Chacun des composants du backend est donc déployé séparément. L'agent peut être injecté en tant que sidecar dans l'application instrumentée. Les services Query et Collector sont configurés avec un type de stockage pris en charge - actuellement Elasticsearch. Plusieurs instances de chacun de ces composants peuvent être provisionnées si nécessaire à des fins de performance et de résilience.

- **streaming** - La stratégie de diffusion en continu est conçue pour compléter la stratégie de production en fournissant une capacité de diffusion en continu qui se situe effectivement entre le collecteur et le stockage dorsal Elasticsearch. Cela permet de réduire la pression sur le stockage dorsal dans les situations de forte charge et permet à d'autres capacités de post-traitement des traces d'exploiter les données en temps réel directement à partir de la plateforme de diffusion en continu(AMQ Streams/ Kafka).

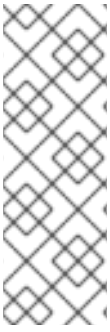
NOTE

La stratégie de streaming nécessite un abonnement Red Hat supplémentaire pour AMQ Streams.



NOTE

La stratégie de déploiement en continu n'est actuellement pas prise en charge sur les systèmes IBM z.



NOTE

Il y a deux façons d'installer et d'utiliser le traçage distribué de Red Hat OpenShift, en tant que partie d'un maillage de services ou en tant que composant autonome. Si vous avez installé le traçage distribué dans le cadre de Red Hat OpenShift Service Mesh, vous pouvez effectuer une configuration de base dans le cadre du ServiceMeshControlPlane, mais pour un contrôle complet, vous devez configurer un Jaeger CR et ensuite référencer votre fichier de configuration de traçage distribué dans le ServiceMeshControlPlane.

3.2.1. Déployer la stratégie par défaut de traçage distribué à partir de la console web

La définition de ressource personnalisée (CRD) définit la configuration utilisée lorsque vous déployez une instance de Red Hat OpenShift distributed tracing. La CRD par défaut est nommée **jaeger-all-in-one-inmemory** et elle est configurée avec des ressources minimales pour s'assurer que vous pouvez l'installer avec succès sur une installation OpenShift Container Platform par défaut. Vous pouvez utiliser cette configuration par défaut pour créer une instance de plateforme de traçage distribué Red Hat OpenShift qui utilise la stratégie de déploiement **AllInOne**, ou vous pouvez définir votre propre fichier de ressources personnalisé.

NOTE

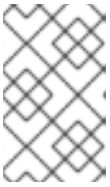
Le stockage en mémoire n'est pas persistant. Si le pod Jaeger s'arrête, redémarre ou est remplacé, vos données de traçage seront perdues. Pour un stockage persistant, vous devez utiliser les stratégies **production** ou **streaming**, qui utilisent Elasticsearch comme stockage par défaut.

Conditions préalables

- La plateforme de traçage distribuée Red Hat OpenShift Operator a été installée.
- Vous avez passé en revue les instructions relatives à la personnalisation du déploiement.
- Vous avez accès au cluster en tant qu'utilisateur ayant le rôle **cluster-admin**.

Procédure

1. Connectez-vous à la console web de OpenShift Container Platform en tant qu'utilisateur ayant le rôle **cluster-admin**.
2. Créez un nouveau projet, par exemple **tracing-system**.



NOTE

Si vous effectuez l'installation dans le cadre du Service Mesh, les ressources de traçage distribuées doivent être installées dans le même espace de noms que la ressource **ServiceMeshControlPlane**, par exemple **istio-system**.

   a. Naviguez jusqu'à **Home → Projects**.
   b. Cliquez sur **Create Project**.
   c. Saisissez **tracing-system** dans le champ **Name**.
   d. Cliquez sur **Create**.

3. Naviguez jusqu'à **Operators → Installed Operators**.
4. Si nécessaire, sélectionnez **tracing-system** dans le menu **Project**. Il se peut que vous deviez attendre quelques instants pour que les opérateurs soient copiés dans le nouveau projet.
5. Cliquez sur l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift. Dans l'onglet **Details**, sous **Provided APIs**, l'opérateur fournit un lien unique.
6. Sous **Jaeger**, cliquez sur **Create Instance**.

7. Sur la page **Create Jaeger**, pour installer en utilisant les valeurs par défaut, cliquez sur **Create** pour créer l'instance de la plate-forme de traçage distribuée.

8. Sur la page **Jaegers**, cliquez sur le nom de l'instance de la plate-forme de traçage distribuée, par exemple **jaeger-all-in-one-inmemory**.

9. Sur la page **Jaeger Details**, cliquez sur l'onglet **Resources**. Attendez que le pod ait un statut de "Running" avant de continuer.

3.2.1.1. Déployer la stratégie par défaut de traçage distribué à partir de l'interface CLI

Suivez cette procédure pour créer une instance de plate-forme de traçage distribuée à partir de la ligne de commande.

Conditions préalables

- L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift a été installé et vérifié.
- Vous avez passé en revue les instructions relatives à la personnalisation du déploiement.
- Vous avez accès à l'OpenShift CLI (**oc**) qui correspond à votre version d'OpenShift Container Platform.
- Vous avez accès au cluster en tant qu'utilisateur ayant le rôle **cluster-admin**.

Procédure

1. Connectez-vous au CLI de OpenShift Container Platform en tant qu'utilisateur ayant le rôle **cluster-admin**.

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

2. Créez un nouveau projet nommé **tracing-system**.

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

3. Créez un fichier de ressources personnalisé nommé **jaeger.yaml** qui contient le texte suivant :

Exemple jaeger-all-in-one.yaml

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

4. Exécutez la commande suivante pour déployer la plateforme de traçage distribuée :

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

5. Exécutez la commande suivante pour observer la progression des modules pendant le processus d'installation :

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

Une fois le processus d'installation terminé, vous devriez obtenir un résultat similaire à l'exemple suivant :

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

3.2.2. Déployer la stratégie de production de traçage distribué à partir de la console web

La stratégie de déploiement **production** est destinée aux environnements de production qui nécessitent une architecture plus évolutive et hautement disponible, et où le stockage à long terme des données de traçage est important.

Conditions préalables

- L'opérateur OpenShift Elasticsearch a été installé.
- La plateforme de traçage distribuée Red Hat OpenShift Operator a été installée.
- Vous avez passé en revue les instructions relatives à la personnalisation du déploiement.
- Vous avez accès au cluster en tant qu'utilisateur ayant le rôle **cluster-admin**.

Procédure

1. Connectez-vous à la console web de OpenShift Container Platform en tant qu'utilisateur ayant le rôle **cluster-admin**.
2. Créez un nouveau projet, par exemple **tracing-system**.

NOTE

Si vous effectuez l'installation dans le cadre du Service Mesh, les ressources de traçage distribuées doivent être installées dans le même espace de noms que la ressource **ServiceMeshControlPlane**, par exemple **istio-system**.

   a. Naviguez jusqu'à **Home → Projects**.
   b. Cliquez sur **Create Project**.
   c. Saisissez **tracing-system** dans le champ **Name**.
   d. Cliquez sur **Create**.

3. Naviguez jusqu'à **Operators → Installed Operators**.
4. Si nécessaire, sélectionnez **tracing-system** dans le menu **Project**. Il se peut que vous deviez attendre quelques instants pour que les opérateurs soient copiés dans le nouveau projet.
5. Cliquez sur l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift. Dans l'onglet **Overview**, sous **Provided APIs**, l'opérateur fournit un lien unique.
6. Sous **Jaeger**, cliquez sur **Create Instance**.

7. Sur la page **Create Jaeger**, remplacez le texte YAML par défaut de **all-in-one** par votre configuration YAML de production, par exemple :

   Exemple de fichier jaeger-production.yaml avec Elasticsearch

```
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. Cliquez sur **Create** pour créer l'instance de la plate-forme de traçage distribuée.

9. Sur la page **Jaegers**, cliquez sur le nom de l'instance de la plate-forme de traçage distribuée, par exemple **jaeger-prod-elasticsearch**.

10. Sur la page **Jaeger Details**, cliquez sur l'onglet **Resources**. Attendez que tous les pods aient un statut de "Running" avant de continuer.

3.2.2.1. Déployer la stratégie de production de traçage distribué à partir de l'interface de ligne de commande

Suivez cette procédure pour créer une instance de plate-forme de traçage distribuée à partir de la ligne de commande.

Conditions préalables

- L'opérateur OpenShift Elasticsearch a été installé.
- La plateforme de traçage distribuée Red Hat OpenShift Operator a été installée.
- Vous avez passé en revue les instructions relatives à la personnalisation du déploiement.
- Vous avez accès à l'OpenShift CLI (**oc**) qui correspond à votre version d'OpenShift Container Platform.
- Vous avez accès au cluster en tant qu'utilisateur ayant le rôle **cluster-admin**.

Procédure

1. Connectez-vous au CLI de OpenShift Container Platform en tant qu'utilisateur ayant le rôle **cluster-admin**.

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

2. Créez un nouveau projet nommé **tracing-system**.

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

3. Créez un fichier de ressources personnalisé nommé **jaeger-production.yaml** qui contient le texte du fichier d'exemple de la procédure précédente.

4. Exécutez la commande suivante pour déployer la plateforme de traçage distribuée :

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

5. Exécutez la commande suivante pour observer la progression des modules pendant le processus d'installation :

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

Une fois le processus d'installation terminé, vous devriez obtenir un résultat similaire à l'exemple suivant :

```
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. Déploiement de la stratégie de traçage distribué en continu à partir de la console web

La stratégie de déploiement **streaming** est destinée aux environnements de production qui nécessitent une architecture plus évolutive et hautement disponible, et où le stockage à long terme des données de traçage est important.

La stratégie **streaming** offre une capacité de diffusion en continu entre le collecteur et le stockage Elasticsearch. Cela réduit la pression sur le stockage dans les situations de charge élevée et permet à d'autres capacités de post-traitement des traces d'exploiter les données en temps réel directement à partir de la plateforme de streaming Kafka.

NOTE

La stratégie de streaming nécessite un abonnement Red Hat supplémentaire pour AMQ Streams. Si vous n'avez pas d'abonnement AMQ Streams, contactez votre représentant commercial pour plus d'informations.

NOTE

La stratégie de déploiement en continu n'est actuellement pas prise en charge sur les systèmes IBM z.

Conditions préalables

- L'opérateur AMQ Streams a été installé. Si vous utilisez la version 1.4.0 ou une version supérieure, vous pouvez utiliser l'auto-provisionnement. Sinon, vous devez créer l'instance Kafka.
- La plateforme de traçage distribuée Red Hat OpenShift Operator a été installée.
- Vous avez passé en revue les instructions relatives à la personnalisation du déploiement.
- Vous avez accès au cluster en tant qu'utilisateur ayant le rôle **cluster-admin**.

Procédure

1. Connectez-vous à la console web de OpenShift Container Platform en tant qu'utilisateur ayant le rôle **cluster-admin**.
2. Créez un nouveau projet, par exemple **tracing-system**.

NOTE

Si vous effectuez l'installation dans le cadre du Service Mesh, les ressources de traçage distribuées doivent être installées dans le même espace de noms que la ressource **ServiceMeshControlPlane**, par exemple **istio-system**.

   a. Naviguez jusqu'à **Home → Projects**.
   b. Cliquez sur **Create Project**.
   c. Saisissez **tracing-system** dans le champ **Name**.
   d. Cliquez sur **Create**.

3. Naviguez jusqu'à **Operators → Installed Operators**.
4. Si nécessaire, sélectionnez **tracing-system** dans le menu **Project**. Il se peut que vous deviez attendre quelques instants pour que les opérateurs soient copiés dans le nouveau projet.
5. Cliquez sur l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift. Dans l'onglet **Overview**, sous **Provided APIs**, l'opérateur fournit un lien unique.
6. Sous **Jaeger**, cliquez sur **Create Instance**.
7. Sur la page **Create Jaeger**, remplacez le texte YAML par défaut de **all-in-one** par votre configuration YAML de streaming, par exemple :

Exemple de fichier jaeger-streaming.yaml

```
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. Cliquez sur **Create** pour créer l'instance de la plate-forme de traçage distribuée.
2. Sur la page **Jaegers**, cliquez sur le nom de l'instance de la plate-forme de traçage distribuée, par exemple **jaeger-streaming**.
3. Sur la page **Jaeger Details**, cliquez sur l'onglet **Resources**. Attendez que tous les pods aient un statut de "Running" avant de continuer.

3.2.3.1. Déployer la stratégie de streaming de traçage distribué à partir de l'interface de ligne de commande

Suivez cette procédure pour créer une instance de plate-forme de traçage distribuée à partir de la ligne de commande.

Conditions préalables

- L'opérateur AMQ Streams a été installé. Si vous utilisez la version 1.4.0 ou une version supérieure, vous pouvez utiliser l'auto-provisionnement. Sinon, vous devez créer l'instance Kafka.
- La plateforme de traçage distribuée Red Hat OpenShift Operator a été installée.
- Vous avez passé en revue les instructions relatives à la personnalisation du déploiement.
- Vous avez accès à l'OpenShift CLI (**oc**) qui correspond à votre version d'OpenShift Container Platform.
- Vous avez accès au cluster en tant qu'utilisateur ayant le rôle **cluster-admin**.

Procédure

1. Connectez-vous au CLI de OpenShift Container Platform en tant qu'utilisateur ayant le rôle **cluster-admin**.

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

2. Créez un nouveau projet nommé **tracing-system**.

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

3. Créez un fichier de ressources personnalisé nommé **jaeger-streaming.yaml** qui contient le texte du fichier d'exemple de la procédure précédente.

4. Exécutez la commande suivante pour déployer Jaeger :

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

5. Exécutez la commande suivante pour observer la progression des modules pendant le processus d'installation :

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

Une fois le processus d'installation terminé, vous devriez obtenir un résultat similaire à l'exemple suivant :

```
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. Valider votre déploiement

3.2.4.1. Accéder à la console Jaeger

Pour accéder à la console Jaeger, vous devez avoir soit Red Hat OpenShift Service Mesh, soit Red Hat OpenShift distributed tracing installé, et Red Hat OpenShift distributed tracing platform installé, configuré et déployé.

Le processus d'installation crée une route pour accéder à la console Jaeger.

Si vous connaissez l'URL de la console Jaeger, vous pouvez y accéder directement. Si vous ne connaissez pas l'URL, suivez les instructions suivantes.

Procédure à partir de la console OpenShift

1. Connectez-vous à la console web de OpenShift Container Platform en tant qu'utilisateur disposant des droits cluster-admin. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.

2. Naviguez jusqu'à **Networking → Routes**.

3. Sur la page **Routes**, sélectionnez le projet de plan de contrôle, par exemple **tracing-system**, dans le menu **Namespace**.
   La colonne **Location** affiche l'adresse liée à chaque itinéraire.

4. Si nécessaire, utilisez le filtre pour trouver la route **jaeger**. Cliquez sur la route **Location** pour lancer la console.

5. Cliquez sur **Log In With OpenShift**

Procédure à partir du CLI

1. Connectez-vous au CLI de OpenShift Container Platform en tant qu'utilisateur ayant le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.

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

2. Pour demander des détails sur l'itinéraire à l'aide de la ligne de commande, entrez la commande suivante. Dans cet exemple, **tracing-system** est l'espace de noms du plan de contrôle.

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

3. Lancez un navigateur et accédez à **https://<JAEGER_URL>**, où **<JAEGER_URL>** est l'itinéraire que vous avez découvert à l'étape précédente.

4. Connectez-vous en utilisant le même nom d'utilisateur et le même mot de passe que ceux utilisés pour accéder à la console OpenShift Container Platform.

5. Si vous avez ajouté des services au maillage de services et généré des traces, vous pouvez utiliser les filtres et le bouton **Find Traces** pour rechercher vos données de traces.
   Si vous validez l'installation de la console, il n'y a pas de données de trace à afficher.

3.2.5. Personnaliser votre déploiement

3.2.5.1. Meilleures pratiques de déploiement

- Les noms des instances de traçage distribuées Red Hat OpenShift doivent être uniques. Si vous souhaitez avoir plusieurs instances de plateforme de traçage distribuée Red Hat OpenShift et que vous utilisez des agents injectés sidecar, les instances de plateforme de traçage distribuée Red Hat OpenShift doivent avoir des noms uniques, et l'annotation d'injection doit explicitement spécifier le nom de l'instance de plateforme de traçage distribuée Red Hat OpenShift vers laquelle les données de traçage doivent être rapportées.

- Si vous avez une implémentation multi-locataires et que les locataires sont séparés par des espaces de noms, déployez une instance de plateforme de traçage distribuée Red Hat OpenShift dans l'espace de noms de chaque locataire.

  - L'agent en tant que daemonset n'est pas pris en charge pour les installations multitenant ou Red Hat OpenShift Dedicated. L'agent en tant que sidecar est la seule configuration supportée pour ces cas d'utilisation.

- Si vous installez le traçage distribué dans le cadre de Red Hat OpenShift Service Mesh, les ressources de traçage distribuées doivent être installées dans le même espace de noms que la ressource **ServiceMeshControlPlane**.

Pour plus d'informations sur la configuration du stockage persistant, voir Comprendre le stockage persistant et la rubrique de configuration appropriée pour l'option de stockage choisie.

3.2.5.2. Options de configuration par défaut du traçage distribué

La ressource personnalisée Jaeger (CR) définit l'architecture et les paramètres à utiliser lors de la création des ressources de la plate-forme de traçage distribuée. Vous pouvez modifier ces paramètres pour adapter la mise en œuvre de votre plate-forme de traçage distribuée aux besoins de votre entreprise.

Exemple de YAML générique Jaeger

```
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: {}
```

Tableau 3.1. Paramètres Jaeger

Paramètres	Description	Valeurs	Valeur par défaut
apiVersion:		Version de l'API à utiliser lors de la création de l'objet.	**jaegertracing.io/v1**
jaegertracing.io/v1	**kind:**	Définit le type d'objet Kubernetes à créer.	**jaeger**

Paramètres	Description	Valeurs	Valeur par défaut
	metadata:	Données permettant d'identifier l'objet de manière unique, y compris une chaîne de caractères **name**, **UID** et, en option, **namespace**.	
OpenShift Container Platform génère automatiquement le **UID** et complète le **namespace** avec le nom du projet où l'objet est créé.	**name:**	Nom de l'objet.	Le nom de l'instance de la plate-forme de traçage distribuée.
jaeger-all-in-one-inmemory	**spec:**	Spécification de l'objet à créer.	Contient tous les paramètres de configuration de votre instance de plateforme de traçage distribuée. Lorsqu'une définition commune à tous les composants Jaeger est requise, elle est définie sous le nœud **spec**. Lorsque la définition concerne un composant individuel, elle est placée sous le nœud **spec/<component>**.
N/A	**strategy:**	Stratégie de déploiement de Jaeger	**allInOne**, **production**, ou **streaming**
allInOne	**allInOne:**	Comme l'image **allInOne** déploie l'agent, le collecteur, la requête, l'ingestion et l'interface utilisateur Jaeger dans un seul pod, la configuration de ce déploiement doit imbriquer la configuration des composants sous le paramètre **allInOne**.	
	agent:	Options de configuration qui définissent l'agent.	

Paramètres	Description	Valeurs	Valeur par défaut
	collector:	Options de configuration qui définissent le collecteur Jaeger.	
	sampling:	Options de configuration qui définissent les stratégies d'échantillonnage pour le traçage.	
	storage:	Options de configuration qui définissent le stockage. Toutes les options liées au stockage doivent être placées sous **storage**, plutôt que sous **allInOne** ou sous d'autres options de composants.	
	query:	Options de configuration qui définissent le service Query.	
	ingester:	Options de configuration qui définissent le service Ingester.	

L'exemple YAML suivant est le minimum requis pour créer un déploiement de plateforme de traçage distribuée Red Hat OpenShift en utilisant les paramètres par défaut.

Exemple minimum requis dist-tracing-all-in-one.yaml

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

3.2.5.3. Options de configuration du collecteur Jaeger

Le collecteur Jaeger est le composant responsable de la réception des intervalles capturés par le traceur et de leur écriture dans un stockage Elasticsearch persistant lors de l'utilisation de la stratégie **production**, ou dans des flux AMQ lors de l'utilisation de la stratégie **streaming**.

Les collecteurs sont sans état et de nombreuses instances de Jaeger Collector peuvent donc être exécutées en parallèle. Les collecteurs ne nécessitent pratiquement aucune configuration, à l'exception de l'emplacement du cluster Elasticsearch.

Tableau 3.2. Paramètres utilisés par l'opérateur pour définir le collecteur Jaeger

Paramètres	Description	Valeurs
`collector:` `  replicas:`	Spécifie le nombre de répliques du collecteur à créer.	Entier, par exemple, **5**

Tableau 3.3. Paramètres de configuration transmis au collecteur

Paramètres	Description	Valeurs

`spec:`
`  collector:`
`  options: {}`
	Options de configuration qui définissent le collecteur Jaeger.	

`options:`
`  collector:`
`  num-workers:`
	Le nombre de travailleurs tirés de la file d'attente.	Entier, par exemple, **50**

`options:`
`  collector:`
`  queue-size:`
	Taille de la file d'attente du collecteur.	Entier, par exemple, **2000**

`options:`
`  kafka:`
`  producer:`
`  topic: jaeger-spans`
	Le paramètre **topic** identifie la configuration Kafka utilisée par le collecteur pour produire les messages et par l'ingérateur pour consommer les messages.	Label pour le producteur.

`options:`
`  kafka:`
`  producer:`
`  brokers: my-cluster-kafka-brokers.kafka:9092`
	Identifie la configuration Kafka utilisée par le collecteur pour produire les messages. Si les courtiers ne sont pas spécifiés, et que vous avez installé AMQ Streams 1.4.0, l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift fournira lui-même Kafka.	

Paramètres	Description	Valeurs
options: log-level:	Niveau de journalisation pour le collecteur.	Valeurs possibles : **debug**, **info**, **warn**, **error**, **fatal**, **panic**.

3.2.5.4. Options de configuration de l'échantillonnage de traçage distribué

La plateforme de traçage distribuée Red Hat OpenShift Operator peut être utilisée pour définir les stratégies d'échantillonnage qui seront fournies aux traceurs qui ont été configurés pour utiliser un échantillonneur distant.

Toutes les traces sont générées, mais seules quelques-unes sont échantillonnées. L'échantillonnage d'une trace la marque pour un traitement et un stockage ultérieurs.

NOTE

Ceci n'est pas pertinent si une trace a été lancée par le proxy Envoy, car la décision d'échantillonnage est prise à ce niveau. La décision d'échantillonnage de Jaeger n'est pertinente que lorsque la trace est lancée par une application utilisant le client.

Lorsqu'un service reçoit une requête qui ne contient pas de contexte de trace, le client lance une nouvelle trace, lui attribue un identifiant aléatoire et prend une décision d'échantillonnage basée sur la stratégie d'échantillonnage actuellement installée. La décision d'échantillonnage se propage à toutes les demandes ultérieures de la trace, de sorte que les autres services ne prennent pas à nouveau la décision d'échantillonnage.

les bibliothèques de la plate-forme de traçage distribué prennent en charge les échantillonneurs suivants :

- **Probabilistic** - L'échantillonneur prend une décision d'échantillonnage aléatoire avec une probabilité d'échantillonnage égale à la valeur de la propriété **sampling.param**. Par exemple, l'utilisation de **sampling.param=0.1** permet d'échantillonner environ 1 trace sur 10.
- **Rate Limiting** - L'échantillonneur utilise un limiteur de taux de leaky bucket pour s'assurer que les traces sont échantillonnées à un certain taux constant. Par exemple, l'utilisation de **sampling.param=2.0** permet d'échantillonner les demandes au rythme de 2 traces par seconde.

Tableau 3.4. Options d'échantillonnage Jaeger

Paramètres	Description	Valeurs	Valeur par défaut
spec: sampling: options: {} default_strategy: service_strategy:	Options de configuration qui définissent les stratégies d'échantillonnage pour le traçage.		Si vous ne fournissez pas de configuration, les collecteurs renverront la politique d'échantillonnage probabiliste par défaut avec une probabilité de 0,001 (0,1 %) pour tous les services.

Paramètres	Description	Valeurs	Valeur par défaut
`default_strategy: type: service_strategy: type:`	Stratégie d'échantillonnage à utiliser. Voir les descriptions ci-dessus.	Les valeurs valables sont **probabilistic**, et **ratelimiting**.	**probabilistic**
`default_strategy: param: service_strategy: param:`	Paramètres de la stratégie d'échantillonnage sélectionnée.	Valeurs décimales et entières (0, .1, 1, 10)	1

Cet exemple définit une stratégie d'échantillonnage par défaut qui est probabiliste, avec une probabilité de 50 % que les instances de la trace soient échantillonnées.

Exemple d'échantillonnage probabiliste

```
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
```

Si aucune configuration n'est fournie par l'utilisateur, la plate-forme de traçage distribuée utilise les paramètres suivants :

Échantillonnage par défaut

```
spec:
  sampling:
    options:
```

```
    default_strategy:
      type: probabilistic
      param: 1
```

3.2.5.5. Options de configuration de la mémoire de traçage distribuée

Vous configurez le stockage pour les services Collector, Ingester et Query à l'adresse **spec.storage**. Plusieurs instances de chacun de ces composants peuvent être provisionnées si nécessaire à des fins de performance et de résilience.

Tableau 3.5. Paramètres généraux de stockage utilisés par l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift pour définir le stockage du traçage distribué

Paramètres	Description	Valeurs	Valeur par défaut

`spec:`
`  storage:`
`  type:`
	Type de stockage à utiliser pour le déploiement.	**memory** **elasticsearch** le stockage en mémoire n'est approprié que pour les environnements de développement, de test, de démonstration et de validation de principe, car les données ne persistent pas si le module est arrêté. Pour les environnements de production, la plateforme de traçage distribuée prend en charge Elasticsearch pour le stockage persistant.	**memory**

`storage:` `  secretname:`	Nom du secret, par exemple **tracing-secret**.		N/A
`storage:` `  options: {}`	Options de configuration qui définissent le stockage.		

Tableau 3.6. Paramètres du nettoyeur d'index Elasticsearch

Paramètres	Description	Valeurs	Valeur par défaut

Paramètres	Description	Valeurs	Valeur par défaut
storage: esIndexCleaner: enabled:	Lors de l'utilisation du stockage Elasticsearch, une tâche est créée par défaut pour nettoyer les anciennes traces de l'index. Ce paramètre permet d'activer ou de désactiver la tâche de nettoyage de l'index.	**true**/ **false**	**true**
storage: esIndexCleaner: numberOfDays:	Nombre de jours à attendre avant de supprimer un index.	Valeur entière	**7**
storage: esIndexCleaner: schedule:	Définit la fréquence de nettoyage de l'index Elasticsearch.	Cron expression	"55 23 * * *"

3.2.5.5.1. Auto-provisionnement d'une instance Elasticsearch

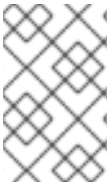
Lorsque vous déployez une ressource personnalisée Jaeger, l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift utilise l'opérateur OpenShift Elasticsearch pour créer un cluster Elasticsearch basé sur la configuration fournie dans la section **storage** du fichier de la ressource personnalisée. L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift provisionnera Elasticsearch si les configurations suivantes sont définies :

- **spec.storage:type** est fixé à **elasticsearch**
- **spec.storage.elasticsearch.doNotProvision** fixé à **false**
- **spec.storage.options.es.server-urls** n'est pas définie, c'est-à-dire qu'il n'y a pas de connexion à une instance d'Elasticsearch qui n'a pas été provisionnée par l'Opérateur Red Hat Elasticsearch.

Lors du provisionnement d'Elasticsearch, l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift définit la ressource personnalisée Elasticsearch **name** à la valeur de **spec.storage.elasticsearch.name** de la ressource personnalisée Jaeger. Si vous ne spécifiez pas de valeur pour **spec.storage.elasticsearch.name**, l'opérateur utilise **elasticsearch**.

Restrictions

- Vous ne pouvez avoir qu'une seule plateforme de traçage distribuée avec instance Elasticsearch auto-provisionnée par espace de noms. Le cluster Elasticsearch doit être dédié à une seule instance de plateforme de traçage distribuée.
- Il ne peut y avoir qu'un seul Elasticsearch par espace de noms.



NOTE

Si vous avez déjà installé Elasticsearch dans le cadre d'OpenShift Logging, l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift peut utiliser l'opérateur OpenShift Elasticsearch installé pour provisionner le stockage.

Les paramètres de configuration suivants concernent une instance Elasticsearch *self-provisioned*, c'est-à-dire une instance créée par l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift à l'aide de l'opérateur OpenShift Elasticsearch. Vous spécifiez les options de configuration pour Elasticsearch auto-provisionné sous **spec:storage:elasticsearch** dans votre fichier de configuration.

Tableau 3.7. Paramètres de configuration des ressources Elasticsearch

Paramètres	Description	Valeurs	Valeur par défaut
elasticsearch: properties: doNotProvision:	À utiliser pour spécifier si une instance Elasticsearch doit être provisionnée par l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift.	**true**/**false**	**true**
elasticsearch: properties: name:	Nom de l'instance Elasticsearch. L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift utilise l'instance Elasticsearch spécifiée dans ce paramètre pour se connecter à Elasticsearch.	chaîne de caractères	**elasticsearch**
elasticsearch: nodeCount:	Nombre de nœuds Elasticsearch. Pour la haute disponibilité, utilisez au moins 3 nœuds. Ne pas utiliser 2 nœuds car un problème de "split brain" peut se produire.	Valeur entière. Par exemple, Preuve de concept = 1, Déploiement minimum =3	3
elasticsearch: resources: requests: cpu:	Nombre d'unités centrales de traitement pour les requêtes, en fonction de la configuration de votre environnement.	Spécifié en cœurs ou en millicores, par exemple 200m, 0,5, 1. Par exemple, preuve de concept = 500m, déploiement minimum =1	1

Paramètres	Description	Valeurs	Valeur par défaut
elasticsearch: resources: requests: memory:	Mémoire disponible pour les requêtes, en fonction de la configuration de votre environnement.	Spécifié en octets, par exemple 200Ki, 50Mi, 5Gi. Par exemple, preuve de concept = 1Gi, déploiement minimal = 16Gi*	16Gi
elasticsearch: resources: limits: cpu:	Limitation du nombre d'unités centrales de traitement, en fonction de la configuration de votre environnement.	Spécifié en cœurs ou en millicores, par exemple 200m, 0,5, 1. Par exemple, preuve de concept = 500m, déploiement minimum =1	
elasticsearch: resources: limits: memory:	Limite de mémoire disponible en fonction de la configuration de votre environnement.	Spécifié en octets, par exemple 200Ki, 50Mi, 5Gi. Par exemple, preuve de concept = 1Gi, déploiement minimal = 16Gi*	
elasticsearch: redundancyPolicy:	La politique de réplication des données définit comment les shards Elasticsearch sont répliqués à travers les nœuds de données dans le cluster. S'il n'est pas spécifié, l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift détermine automatiquement la réplication la plus appropriée en fonction du nombre de nœuds.	**ZeroRedundancy**(pas de répliques), **SingleRedundancy**(une réplique), **MultipleRedundancy** (chaque index est réparti sur la moitié des nœuds de données), **FullRedundancy** (chaque index est entièrement répliqué sur chaque nœud de données de la grappe).	

Paramètres	Description	Valeurs	Valeur par défaut
elasticsearch: useCertManagement:	À utiliser pour spécifier si la plateforme de traçage distribuée doit ou non utiliser la fonctionnalité de gestion des certificats de Red Hat Elasticsearch Operator. Cette fonctionnalité a été ajoutée au sous-système de journalisation pour Red Hat OpenShift 5.2 dans OpenShift Container Platform 4.7 et est le paramètre préféré pour les nouveaux déploiements de Jaeger.	**true**/**false**	**true**
	*Chaque nœud Elasticsearch peut fonctionner avec un paramètre de mémoire inférieur, mais cela n'est PAS recommandé pour les déploiements en production. Pour une utilisation en production, vous ne devriez pas avoir moins de 16Gi alloués à chaque pod par défaut, mais de préférence allouez autant que possible, jusqu'à 64Gi par pod.		

Exemple de stockage de production

```
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
```

Exemple de stockage avec stockage persistant :

```
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 Configuration du stockage persistant. Dans ce cas, AWS **gp2** avec une taille de **5Gi**. Si aucune valeur n'est spécifiée, la plateforme de traçage distribuée utilise **emptyDir**. L'OpenShift Elasticsearch Operator fournit **PersistentVolumeClaim** et **PersistentVolume** qui ne sont pas supprimés avec l'instance de plateforme de traçage distribuée. Vous pouvez monter les mêmes volumes si vous créez une instance de plateforme de traçage distribuée avec le même nom et le même espace de noms.

3.2.5.5.2. Connexion à une instance Elasticsearch existante

Vous pouvez utiliser un cluster Elasticsearch existant pour le stockage avec le traçage distribué. Un cluster Elasticsearch existant, également connu sous le nom d'instance Elasticsearch *external*, est une instance qui n'a pas été installée par l'opérateur de plateforme de traçage distribué Red Hat OpenShift ou par l'opérateur Red Hat Elasticsearch.

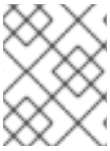
Lorsque vous déployez une ressource personnalisée Jaeger, l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift ne provisionnera pas Elasticsearch si les configurations suivantes sont définies :

- **spec.storage.elasticsearch.doNotProvision** fixé à **true**
- **spec.storage.options.es.server-urls** a une valeur
- **spec.storage.elasticsearch.name** a une valeur, ou si le nom de l'instance Elasticsearch est **elasticsearch**.

L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift utilise l'instance Elasticsearch spécifiée dans **spec.storage.elasticsearch.name** pour se connecter à Elasticsearch.

Restrictions

- Vous ne pouvez pas partager ou réutiliser une instance Elasticsearch de la plateforme OpenShift Container avec une plateforme de traçage distribuée. Le cluster Elasticsearch doit être dédié à une seule instance de plateforme de traçage distribuée.



NOTE

Red Hat ne fournit pas de support pour votre instance Elasticsearch externe. Vous pouvez consulter la matrice des intégrations testées sur le portail client.

Les paramètres de configuration suivants s'appliquent à une instance Elasticsearch déjà existante, également appelée instance Elasticsearch *external*. Dans ce cas, vous spécifiez les options de configuration pour Elasticsearch sous **spec:storage:options:es** dans votre fichier de ressources personnalisé.

Tableau 3.8. Paramètres généraux de configuration de l'ES

Paramètres	Description	Valeurs	Valeur par défaut
`es: server-urls:`	URL de l'instance Elasticsearch.	Le nom de domaine complet du serveur Elasticsearch.	**http://elasticsearch.<namespace>.svc:9200**
`es: max-doc-count:`	Nombre maximal de documents à renvoyer à partir d'une requête Elasticsearch. Cette valeur s'applique également aux agrégations. Si vous définissez à la fois **es.max-doc-count** et **es.max-num-spans**, Elasticsearch utilisera la plus petite valeur des deux.		10000
`es: max-num-spans:`	[**Deprecated** - Sera supprimé dans une prochaine version, utilisez **es.max-doc-count** à la place] Le nombre maximum de portées à récupérer à la fois, par requête, dans Elasticsearch. Si vous définissez à la fois **es.max-num-spans** et **es.max-doc-count**, Elasticsearch utilisera la plus petite valeur des deux.		10000
`es: max-span-age:`	Le délai maximal de consultation pour les périodes dans Elasticsearch.		72h0m0s

Paramètres	Description	Valeurs	Valeur par défaut
es: sniffer:	La configuration du renifleur pour Elasticsearch. Le client utilise le processus de reniflage pour trouver automatiquement tous les nœuds. Désactivé par défaut.	**true**/ **false**	**false**
es: sniffer-tls-enabled:	Option permettant d'activer TLS lors du sniffing d'un cluster Elasticsearch. Le client utilise le processus de reniflage pour trouver automatiquement tous les nœuds. Désactivé par défaut	**true**/ **false**	**false**
es: timeout:	Délai d'attente utilisé pour les requêtes. S'il est fixé à zéro, il n'y a pas de délai d'attente.		0s
es: username:	Le nom d'utilisateur requis par Elasticsearch. L'authentification de base charge également l'AC si elle est spécifiée. Voir aussi **es.password**.		
es: password:	Le mot de passe requis par Elasticsearch. Voir aussi **es.username**.		
es: version:	La version majeure d'Elasticsearch. Si elle n'est pas spécifiée, la valeur sera auto-détectée à partir d'Elasticsearch.		0

Tableau 3.9. Paramètres de réplication des données ES

Paramètres	Description	Valeurs	Valeur par défaut
es: num-replicas:	Le nombre de répliques par index dans Elasticsearch.		1
es: num-shards:	Le nombre d'unités par index dans Elasticsearch.		5

Tableau 3.10. Paramètres de configuration de l'index ES

Paramètres	Description	Valeurs	Valeur par défaut
es: create-index-templates:	Créer automatiquement des modèles d'index au démarrage de l'application lorsque le paramètre est fixé à **true**. Lorsque les modèles sont installés manuellement, le paramètre est fixé à **false**.	**true**/ **false**	**true**
es: index-prefix:	Préfixe facultatif pour les index de la plate-forme de traçage distribuée. Par exemple, la valeur "production" crée des index nommés "production-tracing-*".		

Tableau 3.11. Paramètres de configuration du processeur ES bulk

Paramètres	Description	Valeurs	Valeur par défaut
es: bulk: actions:	Nombre de demandes pouvant être ajoutées à la file d'attente avant que le processeur de masse ne décide d'enregistrer les mises à jour sur le disque.		1000

Paramètres	Description	Valeurs	Valeur par défaut
es: bulk: flush-interval:	**time.Duration** - Intervalle après lequel les demandes groupées sont validées, quels que soient les autres seuils. Pour désactiver l'intervalle de vidange du processeur de traitement en masse, fixer cette valeur à zéro.		200ms
es: bulk: size:	Nombre d'octets que les demandes groupées peuvent occuper avant que le processeur de traitement groupé ne décide d'enregistrer les mises à jour sur le disque.		5000000
es: bulk: workers:	Le nombre de travailleurs capables de recevoir et d'envoyer des requêtes en masse à Elasticsearch.		1

Tableau 3.12. Paramètres de configuration ES TLS

Paramètres	Description	Valeurs	Valeur par défaut
es: tls: ca:	Chemin d'accès à un fichier d'autorité de certification (CA) TLS utilisé pour vérifier les serveurs distants.		Utilise par défaut le truststore du système.
es: tls: cert:	Chemin d'accès à un fichier de certificat TLS, utilisé pour identifier ce processus auprès des serveurs distants.		
es: tls: enabled:	Activer la sécurité de la couche transport (TLS) lors de la communication avec les serveurs distants. Cette option est désactivée par défaut.	**true**/ **false**	**false**

Paramètres	Description	Valeurs	Valeur par défaut
es: tls: key:	Chemin d'accès à un fichier de clé privée TLS, utilisé pour identifier ce processus auprès des serveurs distants.		
es: tls: server-name:	Remplacer le nom du serveur TLS prévu dans le certificat des serveurs distants.		
es: token-file:	Chemin d'accès au fichier contenant le jeton du porteur. Cet indicateur charge également le fichier de l'autorité de certification (CA) s'il est spécifié.		

Tableau 3.13. Paramètres de configuration de l'archive ES

Paramètres	Description	Valeurs	Valeur par défaut
es-archive: bulk: actions:	Nombre de demandes pouvant être ajoutées à la file d'attente avant que le processeur de masse ne décide d'enregistrer les mises à jour sur le disque.		0
es-archive: bulk: flush-interval:	**time.Duration** - Intervalle après lequel les demandes groupées sont validées, quels que soient les autres seuils. Pour désactiver l'intervalle de vidange du processeur de traitement en masse, fixer cette valeur à zéro.		0s

Paramètres	Description	Valeurs	Valeur par défaut
es-archive: bulk: size:	Nombre d'octets que les demandes groupées peuvent occuper avant que le processeur de traitement groupé ne décide d'enregistrer les mises à jour sur le disque.		0
es-archive: bulk: workers:	Le nombre de travailleurs capables de recevoir et d'envoyer des requêtes en masse à Elasticsearch.		0
es-archive: create-index-templates:	Créer automatiquement des modèles d'index au démarrage de l'application lorsque le paramètre est fixé à **true**. Lorsque les modèles sont installés manuellement, le paramètre est fixé à **false**.	**true**/ **false**	**false**
es-archive: enabled:	Permettre un stockage supplémentaire.	**true**/ **false**	**false**
es-archive: index-prefix:	Préfixe facultatif pour les index de la plate-forme de traçage distribuée. Par exemple, la valeur "production" crée des index nommés "production-tracing-*".		
es-archive: max-doc-count:	Nombre maximal de documents à renvoyer à partir d'une requête Elasticsearch. Cette valeur s'applique également aux agrégations.		0

Paramètres	Description	Valeurs	Valeur par défaut
es-archive: max-num-spans:	[**Deprecated** - Sera supprimé dans une prochaine version, utilisez **es-archive.max-doc-count** à la place] Le nombre maximum d'étendues à récupérer à la fois, par requête, dans Elasticsearch.		0
es-archive: max-span-age:	Le délai maximal de consultation pour les périodes dans Elasticsearch.		0s
es-archive: num-replicas:	Le nombre de répliques par index dans Elasticsearch.		0
es-archive: num-shards:	Le nombre d'unités par index dans Elasticsearch.		0
es-archive: password:	Le mot de passe requis par Elasticsearch. Voir aussi **es.username**.		
es-archive: server-urls:	La liste des serveurs Elasticsearch, séparée par des virgules. Les serveurs doivent être spécifiés sous forme d'URL complètes, par exemple, **http://localhost:9200**.		
es-archive: sniffer:	La configuration du renifleur pour Elasticsearch. Le client utilise le processus de reniflage pour trouver automatiquement tous les nœuds. Désactivé par défaut.	**true**/ **false**	**false**

Paramètres	Description	Valeurs	Valeur par défaut
es-archive: sniffer-tls-enabled:	Option permettant d'activer TLS lors du sniffing d'un cluster Elasticsearch. Le client utilise le processus de reniflage pour trouver automatiquement tous les nœuds. Cette option est désactivée par défaut.	**true**/ **false**	**false**
es-archive: timeout:	Délai d'attente utilisé pour les requêtes. S'il est fixé à zéro, il n'y a pas de délai d'attente.		0s
es-archive: tls: ca:	Chemin d'accès à un fichier d'autorité de certification (CA) TLS utilisé pour vérifier les serveurs distants.		Utilise par défaut le truststore du système.
es-archive: tls: cert:	Chemin d'accès à un fichier de certificat TLS, utilisé pour identifier ce processus auprès des serveurs distants.		
es-archive: tls: enabled:	Activer la sécurité de la couche transport (TLS) lors de la communication avec les serveurs distants. Cette option est désactivée par défaut.	**true**/ **false**	**false**
es-archive: tls: key:	Chemin d'accès à un fichier de clé privée TLS, utilisé pour identifier ce processus auprès des serveurs distants.		
es-archive: tls: server-name:	Remplacer le nom du serveur TLS prévu dans le certificat des serveurs distants.		

Paramètres	Description	Valeurs	Valeur par défaut
es-archive: token-file:	Chemin d'accès au fichier contenant le jeton du porteur. Cet indicateur charge également le fichier de l'autorité de certification (CA) s'il est spécifié.		
es-archive: username:	Le nom d'utilisateur requis par Elasticsearch. L'authentification de base charge également l'AC si elle est spécifiée. Voir aussi **es-archive.password**.		
es-archive: version:	La version majeure d'Elasticsearch. Si elle n'est pas spécifiée, la valeur sera auto-détectée à partir d'Elasticsearch.		0

Exemple de stockage avec des montages de volumes

```
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
```

L'exemple suivant montre un Jaeger CR utilisant un cluster Elasticsearch externe avec un certificat TLS CA monté à partir d'un volume et un utilisateur/mot de passe stocké dans un secret.

Exemple d'Elasticsearch externe :

```
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. URL du service Elasticsearch fonctionnant dans l'espace de noms par défaut.
2. Configuration TLS. Dans ce cas, il s'agit uniquement du certificat de l'autorité de certification, mais il peut également contenir es.tls.key et es.tls.cert lors de l'utilisation de TLS mutuel.
3. Secret qui définit les variables d'environnement ES_PASSWORD et ES_USERNAME. Créé par kubectl create secret generic tracing-secret --from-literal=ES_PASSWORD=changeme --from-literal=ES_USERNAME=elastic
4. Les montages de volumes et les volumes qui sont montés dans tous les composants de stockage.

3.2.5.6. Gestion des certificats avec Elasticsearch

Vous pouvez créer et gérer des certificats à l'aide de l'Opérateur Red Hat Elasticsearch. La gestion des certificats à l'aide de Red Hat Elasticsearch Operator vous permet également d'utiliser un seul cluster Elasticsearch avec plusieurs collecteurs Jaeger.

IMPORTANT

La gestion des certificats avec Elasticsearch est une fonctionnalité d'aperçu technologique uniquement. Les fonctionnalités de l'aperçu technologique ne sont pas prises en charge par les accords de niveau de service (SLA) de production de Red Hat et peuvent ne pas être complètes sur le plan fonctionnel. Red Hat ne recommande pas de les utiliser en production. Ces fonctionnalités offrent un accès anticipé aux fonctionnalités des produits à venir, ce qui permet aux clients de tester les fonctionnalités et de fournir un retour d'information pendant le processus de développement.

Pour plus d'informations sur la portée de l'assistance des fonctionnalités de l'aperçu technologique de Red Hat, voir Portée de l'assistance des fonctionnalités de l'aperçu technologique.

À partir de la version 2.4, l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift délègue la création de certificats à l'opérateur Red Hat Elasticsearch en utilisant les annotations suivantes dans la ressource personnalisée Elasticsearch :

- **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"**

Où **<shared-es-node-name>** est le nom du nœud Elasticsearch. Par exemple, si vous créez un nœud Elasticsearch nommé **custom-es**, votre ressource personnalisée pourrait ressembler à l'exemple suivant.

Exemple de CR Elasticsearch montrant les annotations

```
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
```

Conditions préalables

- OpenShift Container Platform 4.7
- sous-système de journalisation pour Red Hat OpenShift 5.2
- Le nœud Elasticsearch et les instances Jaeger doivent être déployés dans le même espace de noms. Par exemple, **tracing-system**.

Vous activez la gestion des certificats en définissant **spec.storage.elasticsearch.useCertManagement** sur **true** dans la ressource personnalisée Jaeger.

Exemple montrant useCertManagement

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

L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift définit la ressource personnalisée Elasticsearch **name** à la valeur de **spec.storage.elasticsearch.name** de la ressource personnalisée Jaeger lors du provisionnement d'Elasticsearch.

Les certificats sont fournis par l'opérateur Red Hat Elasticsearch et l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift injecte les certificats.

3.2.5.7. Options de configuration des requêtes

Query est un service qui récupère les traces du stockage et héberge l'interface utilisateur pour les afficher.

Tableau 3.14. Paramètres utilisés par l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift pour définir la requête

Paramètres	Description	Valeurs	Valeur par défaut

`spec:`
`  query:`
`  replicas:`
	Spécifie le nombre de répliques de requêtes à créer.	Entier, par exemple, **2**	

Tableau 3.15. Paramètres de configuration transmis à la requête

Paramètres	Description	Valeurs	Valeur par défaut

`spec:`
`  query:`
`  options: {}`
	Options de configuration qui définissent le service Query.		

`options:` `  log-level:`	Niveau de journalisation pour Query.	Valeurs possibles : **debug**, **info**, **warn**, **error**, **fatal**, **panic**.	

`options:`
`  query:`
`  base-path:`
	Le chemin de base de toutes les routes HTTP de jaeger-query peut être défini à une valeur non racine, par exemple, **/jaeger** fera commencer toutes les URL de l'interface utilisateur par **/jaeger**. Cela peut s'avérer utile lorsque jaeger-query est exécuté derrière un proxy inverse.	/<path>	

Exemple de configuration d'une requête

```
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. Options de configuration de l'ingestionur

Ingester est un service qui lit à partir d'un sujet Kafka et écrit dans le backend de stockage Elasticsearch. Si vous utilisez les stratégies de déploiement **allInOne** ou **production**, vous n'avez pas besoin de configurer le service Ingester.

Tableau 3.16. Paramètres du Jaeger transmis à l'injecteur

Paramètres	Description	Valeurs

`spec:`
`  ingester:`
`  options: {}`
	Options de configuration qui définissent le service Ingester.	

`options:` `  deadlockInterval:`	Spécifie l'intervalle, en secondes ou en minutes, pendant lequel l'intégrateur doit attendre un message avant de s'arrêter. L'intervalle d'impasse est désactivé par défaut (défini sur **0**), afin d'éviter que l'ingester ne se termine lorsqu'aucun message n'arrive lors de l'initialisation du système.	Minutes et secondes, par exemple **1m0s**. La valeur par défaut est**0**.

`options:`
`  kafka:`
`  consumer:`
`  topic:`
	Le paramètre **topic** identifie la configuration Kafka utilisée par le collecteur pour produire les messages et par l'ingérant pour consommer les messages.	Étiquette destinée au consommateur. Par exemple, **jaeger-spans**.

`options:`
`  kafka:`
`  consumer:`
`  brokers:`
	Identifie la configuration Kafka utilisée par l'ingester pour consommer les messages.	Label pour le courtier, par exemple, **my-cluster-kafka-brokers.kafka:9092**.

`options:` `  log-level:`	Niveau de journalisation pour l'ingérant.	Valeurs possibles : **debug**, **info**, **warn**, **error**, **fatal**, **dpanic**, **panic**.

Exemple de collecteur et d'ingérateur de flux

```
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. Injection des side-cars

La plateforme de traçage distribuée Red Hat OpenShift s'appuie sur un sidecar de proxy dans le pod de l'application pour fournir l'agent. L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift peut injecter des sidecars d'agent dans les charges de travail de déploiement. Vous pouvez activer l'injection automatique de sidecars ou la gérer manuellement.

3.2.6.1. Injection automatique de sidecars

L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift peut injecter des sidecars de l'agent Jaeger dans les charges de travail de déploiement. Pour activer l'injection automatique de sidecars, ajoutez l'ensemble d'annotations **sidecar.jaegertracing.io/inject** à la chaîne **true** ou au nom de l'instance de la plate-forme de traçage distribuée qui est renvoyé par l'exécution de **$ oc get jaegers**. Lorsque vous spécifiez **true**, il ne doit y avoir qu'une seule instance de plate-forme de traçage distribuée pour le même espace de noms que le déploiement, sinon l'opérateur ne peut pas déterminer l'instance de plate-forme de traçage distribuée à utiliser. Un nom d'instance de plate-forme de traçage distribuée spécifique sur un déploiement a une priorité plus élevée que **true** appliqué à son espace de noms.

L'extrait suivant montre une application simple qui injectera un sidecar, l'agent pointant vers l'instance unique de plate-forme de traçage distribuée disponible dans le même espace de noms :

Exemple d'injection automatique du side-car

```
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 Il s'agit soit de la chaîne **true**, soit du nom de l'instance Jaeger.

Lorsque le sidecar est injecté, l'agent est alors accessible à son emplacement par défaut sur **localhost**.

3.2.6.2. Injection manuelle des side-cars

L'opérateur de la plateforme de traçage distribuée Red Hat OpenShift peut uniquement injecter automatiquement les sidecars de l'agent Jaeger dans les charges de travail de déploiement. Pour les types de contrôleurs autres que **Deployments**, tels que **StatefulSets`and `DaemonSets**, vous pouvez définir manuellement le sidecar de l'agent Jaeger dans votre spécification.

L'extrait suivant montre la définition manuelle que vous pouvez inclure dans la section des conteneurs pour un side-car d'agent Jaeger :

Exemple de définition d'un side-car StatefulSet

```
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
```

L'agent est alors accessible à son emplacement par défaut sur localhost.

3.3. CONFIGURATION ET DÉPLOIEMENT DE LA COLLECTE DE DONNÉES DE TRAÇAGE DISTRIBUÉES

L'opérateur de collecte de données de traçage distribué Red Hat OpenShift utilise un fichier de définition de ressource personnalisée (CRD) qui définit l'architecture et les paramètres de configuration à utiliser lors de la création et du déploiement des ressources de collecte de données de traçage distribué Red Hat OpenShift. Vous pouvez soit installer la configuration par défaut, soit modifier le fichier pour mieux répondre aux exigences de votre entreprise.

3.3.1. Options de configuration du collecteur OpenTelemetry

IMPORTANT

L'opérateur de collecte de données de traçage distribué de Red Hat OpenShift est une fonctionnalité d'aperçu technologique uniquement. Les fonctionnalités de l'aperçu technologique ne sont pas prises en charge par les accords de niveau de service (SLA) de production de Red Hat et peuvent ne pas être complètes sur le plan fonctionnel. Red Hat ne recommande pas de les utiliser en production. Ces fonctionnalités offrent un accès anticipé aux fonctionnalités des produits à venir, ce qui permet aux clients de tester les fonctionnalités et de fournir un retour d'information pendant le processus de développement.

Pour plus d'informations sur la portée de l'assistance des fonctionnalités de l'aperçu technologique de Red Hat, voir Portée de l'assistance des fonctionnalités de l'aperçu technologique.

Le collecteur OpenTelemetry est constitué de trois composants qui accèdent aux données télémétriques :

- **Receivers** - Un récepteur, qui peut être de type "push" ou "pull", est la façon dont les données entrent dans le collecteur. En général, un récepteur accepte des données dans un format spécifique, les traduit dans le format interne et les transmet aux processeurs et aux exportateurs définis dans les pipelines applicables. Par défaut, aucun récepteur n'est configuré. Un ou plusieurs récepteurs doivent être configurés. Les récepteurs peuvent prendre en charge une ou plusieurs sources de données.

- **Processors** - (Facultatif) Les processeurs sont exécutés sur les données entre leur réception et leur exportation. Par défaut, aucun processeur n'est activé. Les processeurs doivent être activés pour chaque source de données. Tous les processeurs ne prennent pas en charge toutes les sources de données. Selon la source de données, il peut être recommandé d'activer plusieurs processeurs. En outre, il est important de noter que l'ordre des processeurs est important.

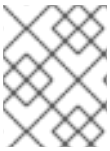
- **Exporters** - Un exportateur, qui peut être de type "push" ou "pull", permet d'envoyer des données à un ou plusieurs backends/destinations. Par défaut, aucun exportateur n'est configuré. Un ou plusieurs exportateurs doivent être configurés. Les exportateurs peuvent prendre en charge une ou plusieurs sources de données. Les exportateurs peuvent être livrés avec des paramètres par défaut, mais beaucoup nécessitent une configuration pour spécifier au moins la destination et les paramètres de sécurité.

Vous pouvez définir plusieurs instances de composants dans un fichier YAML de ressources personnalisées. Une fois configurés, ces composants doivent être activés au moyen de pipelines définis dans la section **spec.config.service** du fichier YAML. La meilleure pratique consiste à n'activer que les

composants dont vous avez besoin.

exemple de fichier de ressources personnalisées du collecteur OpenTelemetry

```
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]
```



NOTE

Si un composant est configuré, mais n'est pas défini dans la section **service**, il n'est pas activé.

Tableau 3.17. Paramètres utilisés par l'opérateur pour définir le collecteur OpenTelemetry

Paramètres	Description	Valeurs	Défaut
récepteurs :	Un récepteur permet aux données d'entrer dans le collecteur. Par défaut, aucun récepteur n'est configuré. Il doit y avoir au moins un récepteur activé pour qu'une configuration soit considérée comme valide. Les récepteurs sont activés lorsqu'ils sont ajoutés à un pipeline.	**otlp**, **jaeger**	Aucun

Paramètres	Description	Valeurs	Défaut
receivers: otlp:	Les récepteurs **oltp** et **jaeger** sont livrés avec des paramètres par défaut, il suffit de spécifier le nom du récepteur pour le configurer.		
des transformateurs :	Les processeurs traitent les données entre leur réception et leur exportation. Par défaut, aucun processeur n'est activé.		Aucun
exportateurs :	Un exportateur envoie des données à un ou plusieurs backends/destinations. Par défaut, aucun exportateur n'est configuré. Il doit y avoir au moins un exportateur activé pour qu'une configuration soit considérée comme valide. Les exportateurs sont activés lorsqu'ils sont ajoutés à un pipeline. Les exportateurs peuvent être livrés avec des paramètres par défaut, mais beaucoup nécessitent une configuration pour spécifier au moins la destination et les paramètres de sécurité.	**logging**, **jaeger**	Aucun

Paramètres	Description	Valeurs	Défaut
exporters: jaeger: endpoint:	Le point de terminaison de l'exportateur **jaeger** doit être de la forme **<name>-collector-headless.<namespace>.svc**, avec le nom et l'espace de noms du déploiement Jaeger, pour qu'une connexion sécurisée puisse être établie.		
exporters: jaeger: tls: ca_file:	Chemin d'accès au certificat de l'autorité de certification. Pour un client, cela permet de vérifier le certificat du serveur. Pour un serveur, cela permet de vérifier les certificats des clients. Si empty utilise l'autorité de certification racine du système.		
service: pipelines:	Les composants sont activés en les ajoutant à un pipeline sous **services.pipeline**.		
service: pipelines: traces: receivers:	Vous activez les récepteurs pour le traçage en les ajoutant sous **service.pipelines.traces**.		Aucun
service: pipelines: traces: processors:	Vous activez les processeurs pour le traçage en les ajoutant sous **service.pipelines.traces**.		Aucun
service: pipelines: traces: exporters:	Vous activez les exportateurs pour le traçage en les ajoutant sous **service.pipelines.traces**.		Aucun

3.3.2. Valider votre déploiement

3.3.3. Accéder à la console Jaeger

Pour accéder à la console Jaeger, vous devez avoir soit Red Hat OpenShift Service Mesh, soit Red Hat OpenShift distributed tracing installé, et Red Hat OpenShift distributed tracing platform installé, configuré et déployé.

Le processus d'installation crée une route pour accéder à la console Jaeger.

Si vous connaissez l'URL de la console Jaeger, vous pouvez y accéder directement. Si vous ne connaissez pas l'URL, suivez les instructions suivantes.

Procédure à partir de la console OpenShift

1. Connectez-vous à la console web de OpenShift Container Platform en tant qu'utilisateur disposant des droits cluster-admin. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.
2. Naviguez jusqu'à **Networking → Routes**.
3. Sur la page **Routes**, sélectionnez le projet de plan de contrôle, par exemple **tracing-system**, dans le menu **Namespace**.
   La colonne **Location** affiche l'adresse liée à chaque itinéraire.
4. Si nécessaire, utilisez le filtre pour trouver la route **jaeger**. Cliquez sur la route **Location** pour lancer la console.
5. Cliquez sur **Log In With OpenShift**

Procédure à partir du CLI

1. Connectez-vous au CLI de OpenShift Container Platform en tant qu'utilisateur ayant le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.

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

2. Pour demander des détails sur l'itinéraire à l'aide de la ligne de commande, entrez la commande suivante. Dans cet exemple, **tracing-system** est l'espace de noms du plan de contrôle.

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

3. Lancez un navigateur et accédez à **https://<JAEGER_URL>**, où **<JAEGER_URL>** est l'itinéraire que vous avez découvert à l'étape précédente.
4. Connectez-vous en utilisant le même nom d'utilisateur et le même mot de passe que ceux utilisés pour accéder à la console OpenShift Container Platform.
5. Si vous avez ajouté des services au maillage de services et généré des traces, vous pouvez utiliser les filtres et le bouton **Find Traces** pour rechercher vos données de traces.
   Si vous validez l'installation de la console, il n'y a pas de données de trace à afficher.

3.4. MISE À JOUR DU TRAÇAGE DISTRIBUÉ

Operator Lifecycle Manager (OLM) contrôle l'installation, la mise à niveau et le contrôle d'accès basé sur les rôles (RBAC) des opérateurs dans un cluster. L'OLM s'exécute par défaut dans OpenShift Container Platform. L'OLM recherche les opérateurs disponibles ainsi que les mises à niveau des opérateurs installés. Pour plus d'informations sur la façon dont OpenShift Container Platform gère les mises à niveau, voir la documentation Operator Lifecycle Manager.

Lors d'une mise à jour, les opérateurs de traçage distribué Red Hat OpenShift mettent à niveau les instances de traçage distribué gérées vers la version associée à l'opérateur. Chaque fois qu'une nouvelle version de l'Opérateur de plateforme de traçage distribuée Red Hat OpenShift est installée, toutes les instances d'application de plateforme de traçage distribuée gérées par l'Opérateur sont mises à niveau vers la version de l'Opérateur. Par exemple, après avoir mis à niveau l'Opérateur de la version 1.10 installée à la version 1.11, l'Opérateur recherche les instances de plateforme de traçage distribuée en cours d'exécution et les met à niveau vers la version 1.11 également.

Pour des instructions spécifiques sur la façon de mettre à jour l'OpenShift Elasticsearch Operator, voir Mise à jour de l'OpenShift Logging .

3.4.1. Changer le canal de l'opérateur pour 2.0

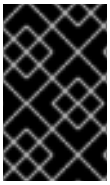
Red Hat OpenShift distributed tracing 2.0.0 a apporté les modifications suivantes :

- Renommé le Red Hat OpenShift Jaeger Operator en Red Hat OpenShift distributed tracing platform Operator.
- Arrêt de la prise en charge des canaux de publication individuels. À l'avenir, la plateforme de traçage distribuée Operator de Red Hat OpenShift ne prendra en charge que le canal Operator **stable**. Les canaux de maintenance, par exemple **1.24-stable**, ne seront plus pris en charge par les futurs opérateurs.

Dans le cadre de la mise à jour vers la version 2.0, vous devez mettre à jour vos abonnements OpenShift Elasticsearch et Red Hat OpenShift distributed tracing platform Operator.

Conditions préalables

- La version d'OpenShift Container Platform est 4.6 ou plus récente.
- Vous avez mis à jour l'OpenShift Elasticsearch Operator.
- Vous avez sauvegardé le fichier de ressources personnalisées de Jaeger.
- Un compte avec le rôle **cluster-admin**. Si vous utilisez Red Hat OpenShift Dedicated, vous devez avoir un compte avec le rôle **dedicated-admin**.



IMPORTANT

Si vous n'avez pas encore mis à jour votre OpenShift Elasticsearch Operator comme décrit dans Mise à jour de OpenShift Logging , complétez cette mise à jour avant de mettre à jour votre Red Hat OpenShift distributed tracing platform Operator.

Pour savoir comment mettre à jour le canal de l'opérateur, voir Mise à jour des opérateurs installés .

3.5. SUPPRESSION DU TRAÇAGE DISTRIBUÉ

Les étapes pour supprimer le traçage distribué de Red Hat OpenShift d'un cluster OpenShift Container Platform sont les suivantes :

1. Arrêtez tous les pods de traçage distribués par Red Hat OpenShift.
2. Supprimez toutes les instances de traçage distribuées de Red Hat OpenShift.
3. Supprimez la plateforme de traçage distribuée Red Hat OpenShift Operator.
4. Supprimez l'opérateur de collecte de données de traçage distribué Red Hat OpenShift.

3.5.1. Suppression d'une instance de plateforme de traçage distribuée Red Hat OpenShift à l'aide de la console Web

NOTE

Lors de la suppression d'une instance qui utilise le stockage en mémoire, toutes les données sont définitivement perdues. Les données stockées dans un stockage persistant tel qu'Elasticsearch ne sont pas supprimées lorsqu'une instance de plateforme de traçage distribuée Red Hat OpenShift est supprimée.

Procédure

1. Connectez-vous à la console web de OpenShift Container Platform.
2. Naviguez jusqu'à **Operators → Installed Operators**.
3. Sélectionnez le nom du projet dans lequel les opérateurs sont installés dans le menu **Project**, par exemple, **openshift-operators**.
4. Cliquez sur l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift.
5. Cliquez sur l'onglet **Jaeger**.
6. Cliquez sur le menu Options  à côté de l'instance que vous souhaitez supprimer et sélectionnez **Delete Jaeger**.
7. Dans le message de confirmation, cliquez sur **Delete**.

3.5.2. Supprimer une instance de plateforme de traçage distribuée Red Hat OpenShift à partir de la CLI

1. Connectez-vous au CLI de OpenShift Container Platform.

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

2. Pour afficher les instances de la plate-forme de traçage distribuée, exécutez la commande suivante :

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

Par exemple,

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

Les noms des opérateurs ont le suffixe **-operator**. L'exemple suivant montre deux opérateurs de plate-forme de traçage distribuée Red Hat OpenShift et quatre instances de plate-forme de traçage distribuée :

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

Vous devriez obtenir un résultat similaire à celui qui suit :

```
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. Pour supprimer une instance de la plate-forme de traçage distribuée, exécutez la commande suivante :

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

Par exemple :

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

4. Pour vérifier la suppression, exécutez à nouveau la commande **oc get deployments**:

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

Par exemple :

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

Vous devriez obtenir un résultat similaire à l'exemple suivant :

```
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. Suppression des opérateurs de traçage distribués de Red Hat OpenShift

Procédure

1. Suivez les instructions relatives à la suppression d'opérateurs d'un cluster.
   - Supprimez la plateforme de traçage distribuée Red Hat OpenShift Operator.

- Une fois que l'opérateur de la plateforme de traçage distribuée Red Hat OpenShift a été supprimé, le cas échéant, supprimez l'opérateur OpenShift Elasticsearch.