Red Hat Integration 2020.Q1

Installing Debezium on OpenShift

For use with Debezium 1.0 on OpenShift Container Platform

Last Updated: 2021-02-19

Red Hat Integration 2020.Q1 Installing Debezium on OpenShift

For use with Debezium 1.0 on OpenShift Container Platform

Legal Notice

Abstract

This guide describes how to install Red Hat Debezium on OpenShift Container Platform with AMQ Streams.

Table of Contents

CHAPTER 1. DEBEZIUM OVERVIEW

Red Hat Debezium is a distributed platform that monitors databases and creates change event streams. Red Hat Debezium is built on Apache Kafka and is deployed and integrated with AMQ Streams.

Debezium captures row-level changes to a database table and passes corresponding change events to AMQ Streams. Applications can read these *change event streams* and access the change events in the order in which they occurred.

Debezium has multiple uses, including:

- Data replication
- Updating caches and search indexes
- Simplifying monolithic applications
- Data integration
- Enabling streaming queries

Debezium provides connectors (based on Kafka Connect) for the following common databases:

- MySQL
- PostgreSQL
- SQL Server
- MongoDB

NOTE

This guide refers to Debezium documentation. Debezium is the upstream community project for Debezium.

1.1. DOCUMENT CONVENTIONS

Replaceables

In this document, replaceable text is styled in monospace and italics.

For example, in the following code, you will want to replace ***my-namespace*** with the name of your namespace:

```
sed -i 's/namespace: .*/namespace: my-namespace/' install/cluster-operator/*RoleBinding*.yaml
```

CHAPTER 2. INSTALLING DEBEZIUM CONNECTORS

Install Debezium connectors through AMQ Streams by extending Kafka Connect with connector plugins. Following a deployment of AMQ Streams, you can deploy Debezium as a connector configuration through Kafka Connect.

2.1. PREREQUISITES

A Debezium installation requires the following:

- An OpenShift cluster
- A deployment of AMQ Streams with Kafka Connect S2I
- A user on the OpenShift cluster with **cluster-admin** permissions to set up the required cluster roles and API services

NOTE

Java 8 or later is required to run the Debezium connectors.

To install Debezium, the OpenShift Container Platform command-line interface (CLI) is required.

- For more information about how to install the CLI for OpenShift 3.11, see the OpenShift Container Platform 3.11 documentation.
- For more information about how to install the CLI for OpenShift 4.2, see the OpenShift Container Platform 4.2 documentation.

Additional resources

- For more information about how to install AMQ Streams, see Using AMQ Streams on OpenShift.
- AMQ Streams includes a *Cluster Operator* to deploy and manage Kafka components. For more information about how to install Kafka components using the AMQ Streams Cluster Operator, see Deploying Kafka Connect to your cluster .

2.2. KAFKA TOPIC CREATION RECOMMENDATIONS

Debezium uses multiple Kafka topics for storing data. The topics have to be either created by an administrator, or by Kafka itself by enabling auto-creation for topics using the **auto.create.topics.enable** broker configuration.

The following list describes limitations and recommendations to consider when creating topics:

Database history topics (for MySQL and SQL Server connectors)

- Infinite (or very long retention).
- Replication factor of at least 3 in production.
- Single partition.

Other topics

- Optionally, log compaction enabled (if you wish to only keep the *last* change event for a given record).
  In this case, the **min.compaction.lag.ms** and **delete.retention.ms** topic-level settings in Apache Kafka should be configured so that consumers have enough time to receive all events and delete markers. Specifically, these values should be larger than the maximum downtime you anticipate for the sink connectors (for example, when you update them).

- Replicated in production.

- Single partition.
  You can relax the single partition rule, but your application must handle out-of-order events for different rows in the database (events for a single row are still totally ordered). If multiple partitions are used, Kafka will determine the partition by hashing the key by default. Other partition strategies require using Simple Message Transforms (SMTs) to set the partition number for each record.

2.3. DEPLOYING DEBEZIUM WITH AMQ STREAMS

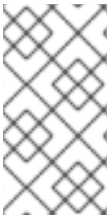
This procedure describes how to set up connectors for Debezium on Red Hat OpenShift Container Platform.

Before you begin

For setting up Apache Kafka and Kafka Connect on OpenShift, Red Hat AMQ Streams is used. AMQ Streams offers operators and images that bring Kafka to OpenShift.

Here we deploy and use Kafka Connect S2I (Source to Image). S2I is a framework to build images that take application source code as an input and produce a new image that runs the assembled application as output.

A Kafka Connect builder image with S2I support is provided on the Red Hat Container Catalog as part of the **registry.redhat.io/amq7/amq-streams-kafka-24:1.4.0** image. The S2I process takes your binaries (with plugins and connectors) and stores them in the **/tmp/kafka-plugins/s2i** directory. It creates a new Kafka Connect image from this directory, which can then be used with the Kafka Connect deployment. When started using the enhanced image, Kafka Connect loads any third-party plug-ins from the **/tmp/kafka-plugins/s2i** directory.



NOTE

Instead of deploying and using the Kafka Connect S2I, you can create a new *Dockerfile* based on an AMQ Streams Kafka image to include the connectors.

See Chapter 3, *Creating a container image from the Kafka Connect base image* .

In this procedure, we:

- Deploy a Kafka cluster to OpenShift
- Download and configure the Debezium connectors
- Deploy Kafka Connect with the connectors

If you have a Kafka cluster deployed already, you can skip the first step.

NOTE

The pod names must correspond with your AMQ Streams deployment.

Procedure

1. Deploy your Kafka cluster.
   a. Install the AMQ Streams operator by following the steps in Installing AMQ Streams and deploying components.
   b. Select the desired configuration and deploy your Kafka Cluster .
   c. Deploy Kafka Connect s2i.

   We now have a working Kafka cluster running in OpenShift with Kafka Connect S2I.

2. Check that your pods are running:

```
$ oc get pods

NAME                                          READY STATUS
<cluster-name>-entity-operator-7b6b9d4c5f-k7b92    3/3   Running
<cluster-name>-kafka-0                             2/2   Running
<cluster-name>-zookeeper-0                         2/2   Running
<cluster-name>-operator-97cd5cf7b-l58bq            1/1   Running
```

   In addition to running pods you should have a **DeploymentConfig** associated with your Connect **S2I**.

3. Select release 1.0, and download the Debezium connector archive for your database from the Red Hat Integration download site .

4. Extract the archive to create a directory structure for the connector plugins.

```
$ tree ./my-plugin/
./my-plugin/
├── debezium-connector-mongodb
|   ├── ...
├── debezium-connector-mysql
|   ├── ...
├── debezium-connector-postgres
|   ├── ...
└── debezium-connector-sqlserver
    ├── ...
```

   Now we trigger the Kafka Connect S2I build.

5. Check the name of the build config.

```
$ oc get buildconfigs

NAME                          TYPE     FROM     LATEST
<cluster-name>-cluster-connect   Source   Binary   2
```

6. Use the **oc start-build** command to start a new build of the Kafka Connect image using the Debezium directory:

```
oc start-build <cluster-name>-cluster-connect --from-dir ./my-plugin/
```

NOTE

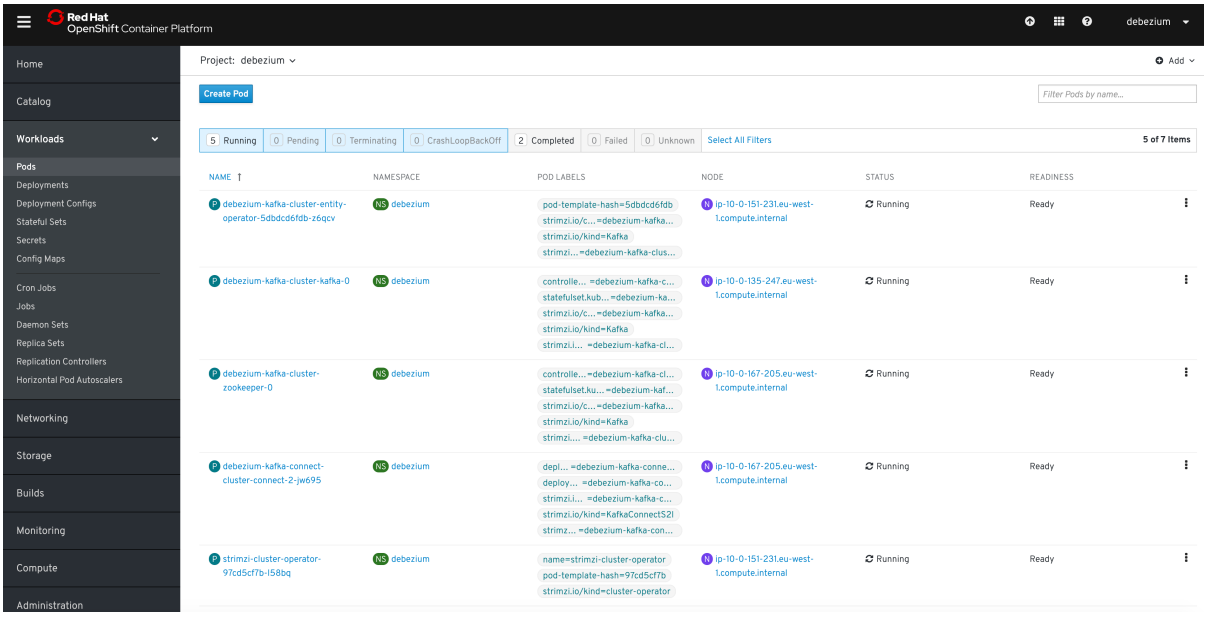
The name of the build is the same as the name of the deployed Kafka Connect cluster.

7. Check the updated deployment is running:

```
oc get pods

NAME                                          READY     STATUS
<cluster-name>-entity-operator-7b6b9d4c5f-k7b92   3/3       Running
<cluster-name>-kafka-0                            2/2       Running
<cluster-name>-zookeeper-0                        2/2       Running
<cluster-name>-cluster-connect-2-jw695            1/1       Running
<cluster-name>-cluster-connect-2-deploy           0/1       Completed
strimzi-cluster-operator-97cd5cf7b-l58bq          1/1       Running
```

Alternatively, you can go to the *Pods* view of your OpenShift Web Console to confirm the pods are running:



Updating Kafka Connect

If you need to update your deployment, amend your JAR files in the Debezium directory and rebuild Kafka Connect.

Verifying the Deployment

Once the build has finished, the new image is used automatically by the Kafka Connect deployment.

When the connector starts, it will connect to the source and produce events for each inserted, updated, and deleted row or document.

Verify that deployment is correct by following the procedures in *Getting Started with Debezium*, Starting the services, which provides instructions for setting up a Kafka cluster and configuring Kafka Connect.

To set up a particular connector, see:

- Deploying the MySQL connector
- Deploying the MongoDB connector
- Deploying the PostgreSQL connector
- Deploying the SQL Server connector

CHAPTER 3. CREATING A CONTAINER IMAGE FROM THE KAFKA CONNECT BASE IMAGE

An alternative to using Kafka Connect S2I is to build your own CDC image using Docker. You can use the Kafka container image on Red Hat Container Catalog as a base image for creating your own custom image with additional connector plugins.

The following procedure explains how to create your custom image and add it to the **/opt/kafka/plugins** directory. At startup, the Debezium version of Kafka Connect loads any third-party connector plug-ins contained in the **/opt/kafka/plugins** directory.

Prerequisites

- AMQ Streams Cluster Operator is deployed

Procedure

1. Create a new **Dockerfile** using **registry.redhat.io/amq7/amq-streams-kafka-24:1.4.0** as the base image:

```
FROM registry.redhat.io/amq7/amq-streams-kafka-24:1.4.0
USER root:root
COPY ./my-plugins/ /opt/kafka/plugins/
USER 1001
```

2. Build the container image.

```
docker build -t my-new-container-image:latest
```

3. Push your custom image to your container registry.

```
docker push my-new-container-image:latest
```

4. Point to the new container image.
   You can either:

   - Edit the **KafkaConnect.spec.image** property of the **KafkaConnect** custom resource. If set, this property overrides the **STRIMZI_DEFAULT_KAFKA_CONNECT_IMAGE** variable in the Cluster Operator.

```
apiVersion: kafka.strimzi.io/v1beta1
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
  #...
  image: my-new-container-image
```

   or

   - In the **install/cluster-operator/050-Deployment-strimzi-cluster-operator.yaml** file, edit the **STRIMZI_DEFAULT_KAFKA_CONNECT_IMAGE** variable to point to the new container image and reinstall the Cluster Operator. If you edit this file you will need to apply

it to your OpenShift cluster.

Additional resources

- For more information on the **KafkaConnect.spec.image property** and **STRIMZI_DEFAULT_KAFKA_CONNECT_IMAGE** variable, see Using AMQ Streams on OpenShift.

APPENDIX A. USING YOUR SUBSCRIPTION

Debezium is provided through a software subscription. To manage your subscriptions, access your account at the Red Hat Customer Portal.

Accessing Your Account

1. Go to access.redhat.com.
2. If you do not already have an account, create one.
3. Log in to your account.

Activating a Subscription

1. Go to access.redhat.com.
2. Navigate to **My Subscriptions**.
3. Navigate to **Activate a subscription** and enter your 16-digit activation number.

Downloading Zip and Tar Files

To access zip or tar files, use the customer portal to find the relevant files for download. If you are using RPM packages, this step is not required.

1. Open a browser and log in to the Red Hat Customer Portal **Product Downloads** page at access.redhat.com/downloads.
2. Scroll down to **INTEGRATION AND AUTOMATION**.
3. Click **Red Hat Integration** to display the Red Hat Integration downloads page.
4. Click the **Download** link for your component.

Revised on 2021-02-19 08:56:23 UTC