Red Hat Integration 2021.Q3

Installing Debezium on OpenShift

For use with Debezium 1.5 on OpenShift Container Platform

Last Updated: 2021-08-19

Red Hat Integration 2021.Q3 Installing Debezium on OpenShift

For use with Debezium 1.5 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

PREFACE

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

CHAPTER 1. DEBEZIUM OVERVIEW

Red Hat Debezium is a distributed platform that captures database operations, creates data change event records for row-level operations, and streams change event records to Kafka topics. 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:

- Db2
- MySQL
- MongoDB
- Oracle (Developer Preview)
- PostgreSQL
- SQL Server

Debezium is the upstream community project for Red Hat Debezium.

CHAPTER 2. INSTALLING DEBEZIUM CONNECTORS

Install Debezium connectors through AMQ Streams by extending Kafka Connect with connector plug-ins. 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
- 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 information about how to install the CLI for OpenShift 4.7, see the OpenShift Container Platform 4.7 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 stores data in multiple Kafka topics. The topics must either be created in advance by an administrator, or you can configure Kafka Connect to configure topics automatically.

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

Database history topics for MySQL, SQL Server, Db2, and Oracle connectors

- Infinite or very long retention.
- Replication factor of at least three in production environments.
- Single partition.

Other topics

- When you enable Kafka log compaction so that only the *last* change event for a given record is saved, set the following topic properties in Apache Kafka:
  - **min.compaction.lag.ms**

- **delete.retention.ms**
  To ensure that consumers have enough time to receive all events and delete markers, specify values for the preceding properties that are larger than the maximum downtime that you expect for your sink connectors. For example, consider the downtime that might occur when you apply updates to sink connectors.

- 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 you use multiple partitions, the default behavior is that Kafka determines the partition by hashing the key. Other partition strategies require the use of single message transformations (SMTs) to set the partition number for each record.

2.3. DEPLOYING DEBEZIUM WITH AMQ STREAMS

To set up connectors for Debezium on Red Hat OpenShift Container Platform, deploy a Kafka cluster to OpenShift, download and configure Debezium connectors, and deploy Kafka Connect with the connectors.

Prerequisites

- You used Red Hat AMQ Streams to set up Apache Kafka and Kafka Connect on OpenShift. AMQ Streams offers operators and images that bring Kafka to OpenShift.

- Podman or Docker is installed.

Procedure

1. Deploy your Kafka cluster. If you already have a Kafka cluster deployed, skip the following three sub-steps.

   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.

   You now have a working Kafka cluster that is running in OpenShift with Kafka Connect.

2. Check that your pods are running. The pod names correspond with your AMQ Streams deployment.

```
$ 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 Kafka Connect.

3. Go to the Red Hat Integration download site .
4. Download the Debezium connector archive(s) for your database(s).
5. Extract the archive(s) to create a directory structure for the connector plug-in(s). If you downloaded and extracted multiple archives, the structure looks like this:

```
$ tree ./my-plugins/
./my-plugins/
├── debezium-connector-db2
│   ├── ...
├── debezium-connector-mongodb
│   ├── ...
├── debezium-connector-mysql
│   ├── ...
├── debezium-connector-postgres
│   ├── ...
└── debezium-connector-sqlserver
    ├── ...
```

6. Create a new **Dockerfile** by using **registry.redhat.io/amq7/amq-streams-kafka-28-rhel8:1.8.0** as the base image:

```
FROM registry.redhat.io/amq7/amq-streams-kafka-28-rhel8:1.8.0
USER root:root
COPY ./my-plugins/ /opt/kafka/plugins/
USER 1001
```

7. Build the container image. If the **Dockerfile** you created in the previous step is in the current directory, enter one of the following commands:

```
podman build -t my-new-container-image:latest .
```

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

8. Push your custom image to your container registry. Enter one of the following commands:

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

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

9. Point to the new container image. Complete one of the following tasks to specify the name of the image that you created to run your Debezium connector:

   - Edit the **spec.image** field of the **KafkaConnect** custom resource.
     If you set this property, the value overrides the **STRIMZI_DEFAULT_KAFKA_CONNECT_IMAGE** variable in the cluster Operator. For example:

     ```
     apiVersion: kafka.strimzi.io/v1beta2
     kind: KafkaConnect
     ```

```
metadata:
  name: my-connect-cluster
  annotations: strimzi.io/use-connector-resources: "true"
spec:
  #...
  image: my-new-container-image
```

- 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.

The Kafka Connect deployment starts to use the new image.

Next steps

- For each Debezium connector that you want to deploy, create and apply a **KafkaConnect** custom resource that configures a connector instance. This starts running the connector against the configured database. When the connector starts, it connects to the configured database and generates change event records for each inserted, updated, and deleted row or document. Details for deploying a connector are in the following sections:
  - Deploying the MySQL connector
  - Deploying the MongoDB connector
  - Deploying the PostgreSQL connector
  - Deploying the SQL Server connector
  - Deploying the Db2 connector
    To use the Db2 connector, you must have a license for the IBM InfoSphere Data Replication (IIDR) product. However, IIDR does not need to be installed.
- 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

Integration 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 2021-08-19 14:30:44 UTC