



# Red Hat Integration 2020-Q3

## Installing Debezium on OpenShift

For use with Debezium 1.2 on OpenShift Container Platform



# Red Hat Integration 2020-Q3 Installing Debezium on OpenShift

---

For use with Debezium 1.2 on OpenShift Container Platform

## Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

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

---

## Table of Contents

<b>CHAPTER 1. DEBEZIUM OVERVIEW</b> .....	<b>3</b>
<b>CHAPTER 2. INSTALLING DEBEZIUM CONNECTORS</b> .....	<b>4</b>
2.1. PREREQUISITES	4
2.2. KAFKA TOPIC CREATION RECOMMENDATIONS	4
2.3. DEPLOYING DEBEZIUM WITH AMQ STREAMS	5
<b>APPENDIX A. USING YOUR SUBSCRIPTION</b> .....	<b>8</b>
Accessing your account	8
Activating a subscription	8
Downloading zip and tar files	8



# 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.

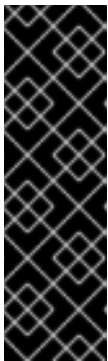
Debezium captures row-level changes to a database table and passes corresponding change events to AMQ Streams. Applications can read these *change event streams* and access the change events in the order in which they occurred.

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:

- MySQL
- PostgreSQL
- SQL Server
- MongoDB
- Db2



## IMPORTANT

The Debezium Db2 connector is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service-level agreements (SLAs) and might not be functionally complete; therefore, Red Hat does not recommend implementing any Technology Preview features in production environments. This Technology Preview feature provides early access to upcoming product innovations, enabling you to test functionality and provide feedback during the development process. For more information about support scope, see [Technology Preview Features Support Scope](#).

[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 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
- 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 4.4, see the [OpenShift Container Platform 4.4 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 must be created by an administrator, or by Kafka itself by [enabling auto-creation for topics using the `auto.create.topics.enable` broker configuration property](#).

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

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

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

#### Other topics

- When [Kafka log compaction](#) is enabled because you want to keep only the *last* change event for a given record, configure the **`min.compaction.lag.ms`** and **`delete.retention.ms`** topic-



level settings in Apache Kafka. You want to ensure that consumers have enough time to receive all events and delete markers. Consequently, set these values to be larger than the maximum downtime you anticipate for the sink connectors. For example, consider the downtime when you update the 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 using simple message transforms (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 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-25-rhel7:1.5.0** as the base image:

```
FROM registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0
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, run the following command:

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

8. Push your custom image to your container registry:

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

9. Point to the new container image. Do one of the following:

- Edit the **spec.image** field of the **KafkaConnector** custom resource. If set, this property overrides the **STRIMZI\_DEFAULT\_KAFKA\_CONNECT\_IMAGE** variable in the Cluster Operator. For example:

```
apiVersion: kafka.strimzi.io/v1beta1
kind: KafkaConnector
metadata:
  name: my-connect-cluster
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](https://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](https://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](https://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 2020-10-02 21:45:37 UTC*