Red Hat Integration 2019-12

Installing Change Data Capture on OpenShift

For use with Change Data Capture 1.0 on OpenShift Container Platform

Last Updated: 2019-12-17

Red Hat Integration 2019-12 Installing Change Data Capture on OpenShift

For use with Change Data Capture 1.0 on OpenShift Container Platform

Legal Notice

Copyright © 2019 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 ® is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js ® 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 ® 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 Change Data Capture on OpenShift Container Platform with AMQ Streams.

Table of Contents

CHAPTER 1. CHANGE DATA CAPTURE OVERVIEW

IMPORTANT

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.

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

Change Data Capture 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.

Change Data Capture has multiple uses, including:

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

Change Data Capture 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 open source project for Change Data Capture.

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 CHANGE DATA CAPTURE CONNECTORS

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

2.1. PREREQUISITES

A Change Data Capture 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 Change Data Capture connectors.

To install Change Data Capture, 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 1.3 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

Change Data Capture 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:

- Replication, a factor of at least 3 in production
- Single partition
- Infinite (or very long) retention if topic compaction is disabled

- Log compaction enabled, if you wish to only keep the *last* change event for a given record

WARNING

Do not enable topic compaction for the database history topics used by the MySQL and SQL Server connectors.

If you relax the single partition rule, your application must be able to handle out-of-order events for different rows in the database (events for a single row are still fully 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 key for each record.

For log compaction, configure the **min.compaction.lag.ms** and **delete.retention.ms** topic-level settings in Apache Kafka 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, such as when you are updating them.

2.3. DEPLOYING CHANGE DATA CAPTURE WITH AMQ STREAMS

This procedure describes how to set up connectors for Change Data Capture 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-23:1.3.0** 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.

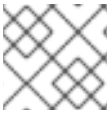
See Chapter 3, *Creating a Docker image from the Kafka Connect base image* .

In this procedure, we:

- Deploy a Kafka cluster to OpenShift
- Download and configure the Change Data Capture 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 AMQ Streams documentation.
   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 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 AMQ Streams download site.

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

```
$ tree ./my-plugin/
./my-plugin/
├── debezium-connector-mongodb
│   ├── CHANGELOG.md
│   ├── CONTRIBUTE.md
│   ├── COPYRIGHT.txt
│   ├── LICENSE-3rd-PARTIES.txt
│   ├── LICENSE.txt
│   ├── README.md
│   ├── bson-3.10.1.redhat-00001.jar
│   ├── debezium-connector-mongodb-1.0.0.Beta2-redhat-00001.jar
│   ├── debezium-core-1.0.0.Beta2-redhat-00001.jar
│   ├── mongodb-driver-3.10.1.redhat-00001.jar
│   ├── mongodb-driver-core-3.10.1.redhat-00001.jar
│   └── util-3.10.1.redhat-00001.jar
├── debezium-connector-mysql
│   ├── CHANGELOG.md
│   ├── CONTRIBUTE.md
```

```
│   ├── COPYRIGHT.txt
│   ├── LICENSE-3rd-PARTIES.txt
│   ├── LICENSE.txt
│   ├── README.md
│   ├── antlr4-runtime-4.7.0.redhat-00013.jar
│   ├── debezium-connector-mysql-1.0.0.Beta2-redhat-00001.jar
│   ├── debezium-core-1.0.0.Beta2-redhat-00001.jar
│   ├── debezium-ddl-parser-1.0.0.Beta2-redhat-00001.jar
│   ├── mysql-binlog-connector-java-0.19.1.redhat-00002.jar
│   └── mysql-connector-java-8.0.16.redhat-00001.jar
├── debezium-connector-postgres
│   ├── CHANGELOG.md
│   ├── CONTRIBUTE.md
│   ├── COPYRIGHT.txt
│   ├── LICENSE-3rd-PARTIES.txt
│   ├── LICENSE.txt
│   ├── README.md
│   ├── debezium-connector-postgres-1.0.0.Beta2-redhat-00001.jar
│   ├── debezium-core-1.0.0.Beta2-redhat-00001.jar
│   ├── postgresql-42.2.8.redhat-00001.jar
│   └── protobuf-java-3.8.0.redhat-00001.jar
└── debezium-connector-sqlserver
    ├── CHANGELOG.md
    ├── CONTRIBUTE.md
    ├── COPYRIGHT.txt
    ├── LICENSE-3rd-PARTIES.txt
    ├── LICENSE.txt
    ├── README.md
    ├── debezium-connector-sqlserver-1.0.0.Beta2-redhat-00001.jar
    ├── debezium-core-1.0.0.Beta2-redhat-00001.jar
    └── mssql-jdbc-7.2.2.jre8-redhat-00001.jar
```

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 Change Data Capture directory:

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

NOTE

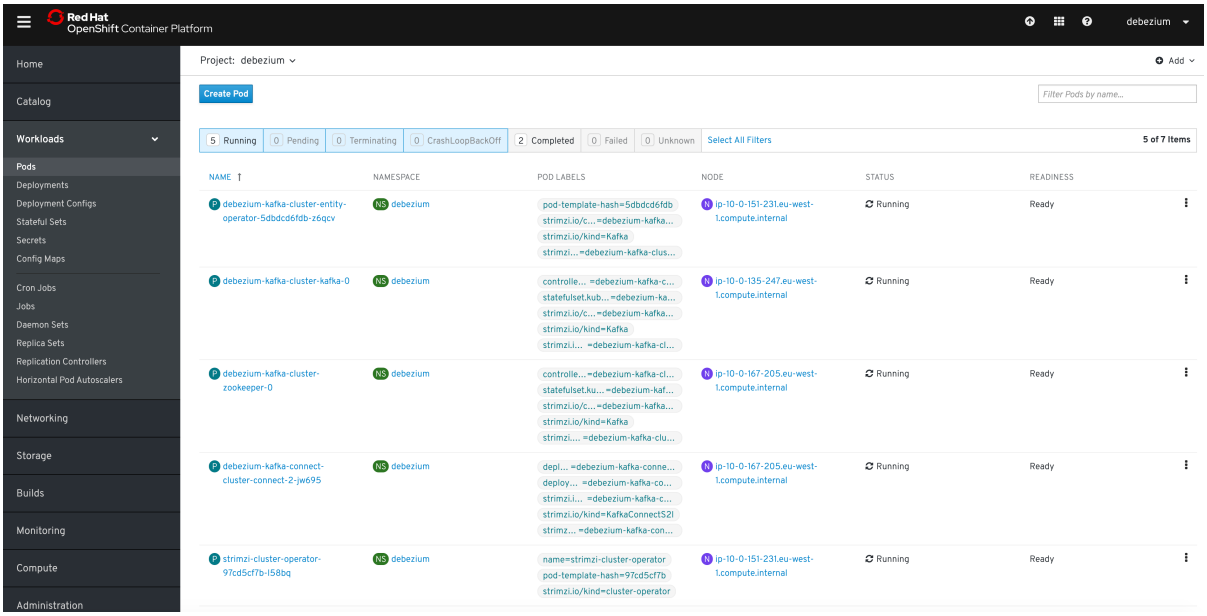
The name of the build is the same as the name of the deployed Kafka Connect cluster.

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 Change Data Capture 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 whether the deployment is correct by emulating the Debezium tutorial by following the steps to verify the deployment.

CHAPTER 3. CREATING A DOCKER 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 Change Data Capture 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-23:1.3.0** as the base image:

```
FROM registry.redhat.io/amq7/amq-streams-kafka-23:1.3.0
USER root:root
COPY ./my-plugins/ /opt/kafka/plugins/
USER jboss:jboss
```

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

Change Data Capture 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. Locate the **Red Hat Change Data Capture** entries in the **JBOSS INTEGRATION AND AUTOMATION** category.
3. Select the desired Change Data Capture product. The **Software Downloads** page opens.
4. Click the **Download** link for your component.

Revised on 2019-12-17 21:40:31 UTC