Red Hat build of Debezium 2.5.4

在 OpenShift 上安装 Debezium

用于 OpenShift Container Platform 上的红帽构建的 Debezium 2.5.4

Last Updated: 2024-04-19

Red Hat build of Debezium 2.5.4 在 OpenShift 上安装 Debezium

用于 OpenShift Container Platform 上的红帽构建的 Debezium 2.5.4

法律通告

摘要

本指南论述了如何使用 AMQ Streams 在 OpenShift Container Platform 上安装 Debezium。

目录

前言

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 DEBEZIUM 概述

Red Hat Integration 的 Debezium 是一个用来捕获数据库操作的分布式平台，为行级操作创建数据更改事件记录，并将事件记录流传输到 Apache Kafka 主题。Debezium 基于 Apache Kafka 构建，已部署并与 AMQ Streams 集成。

Debezium 捕获数据库表的行级更改，并将对应的更改事件传递给 AMQ Streams。应用程序可以读取 *这些更改事件流*，并按发生更改事件的顺序访问更改事件。

Debezium 是 Debezium 用于 Red Hat Integration 的上游社区项目。

Debezium 具有多个用途，包括：

- 数据复制
- 更新缓存和搜索索引
- 简化单体式应用程序
- 数据集成
- 启用流查询

Debezium 为以下通用数据库提供 Apache Kafka 连接连接器：

- Db2
- MySQL
- MongoDB
- Oracle
- PostgreSQL
- SQL Server

第 2 章 安装 DEBEZIUM 连接器

通过使用连接器插件扩展 Kafka Connect，通过 AMQ Streams 安装 Debezium 连接器。部署 AMQ Streams 后，您可以通过 Kafka Connect 将 Debezium 部署为连接器配置。

2.1. KAFKA 主题创建建议

Debezium 将数据存储在多个 Apache Kafka 主题中。主题必须由管理员提前创建，或者您可以配置 Kafka Connect 以 自动配置主题。

以下列表描述了在创建主题时要考虑的限制和建议：

Debezium Db2、MySQL、Oracle 和 SQL Server 连接器的数据库架构历史记录主题

对于前面的每个连接器，都需要一个数据库架构历史记录主题。无论您手动创建数据库 schema 历史记录主题，请使用 Kafka 代理自动创建主题，或使用 Kafka Connect 创建主题，请确保主题配置了以下设置：

- 无限或非常长的保留。
- 在生产环境中至少为 3 个复制因素。
- 单个分区。

其他主题

- 当您启用 Kafka 日志压缩 以便只保存给定记录 *的最后* 更改事件时，在 Apache Kafka 中设置以下主题属性：
  - **min.compaction.lag.ms**
  - **delete.retention.ms**
    为确保主题消费者有足够的时间接收所有事件和删除标记，请指定超过接收器连接器期望的最大停机时间的值。例如，请考虑将更新应用到接收器连接器时可能会出现的停机时间。
- 在生产环境中复制。
- 单个分区。
  您可以放宽单个磁盘分区，但应用程序必须为数据库中的不同行处理超出顺序的事件。一行的事件仍然被完全排序。如果您使用多个分区，则默认行为是 Kafka 通过哈希密钥来确定分区。其他分区策略需要使用单一消息转换(SMT)来为每个记录设置分区号。

2.2. AMQ STREAMS 上的 DEBEZIUM 部署

要在 Red Hat OpenShift Container Platform 上为 Debezium 设置连接器，您可以使用 AMQ Streams 构建 Kafka Connect 容器镜像，其中包含您要使用的每个连接器的连接器插件。连接器启动后，它会连接到配置的数据库，并为每个插入、更新和删除行或文档生成更改事件记录。

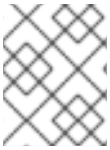
从 Debezium 1.7 开始，部署 Debezium 连接器的首选方法是使用 AMQ Streams 构建包含连接器插件的 Kafka Connect 容器镜像。

在部署过程中，您可以创建并使用以下自定义资源(CR)：

- 定义 Kafka Connect 实例的 **KafkaConnect** CR，并包含有关镜像中需要包含连接器工件的信息。
- **KafkaConnector** CR，提供包括连接器用来访问源数据库的信息。在 AMQ Streams 启动 Kafka Connect pod 后，您可以通过应用 **KafkaConnector** CR 来启动连接器。

在 Kafka Connect 镜像的构建规格中，您可以指定可用于部署的连接器。对于每个连接器插件，您还可以指定您的部署可以使用的其他组件。例如，您可以添加 Apicurio Registry 工件或 Debezium 脚本组件。当 AMQ Streams 构建 Kafka Connect 镜像时，它会下载指定的工件，并将其合并到镜像中。

KafkaConnect CR 中的 **spec.build.output** 参数指定存储生成的 Kafka Connect 容器镜像的位置。容器镜像可以存储在 Docker registry 中，也可以存储在 OpenShift ImageStream 中。要将镜像存储在 ImageStream 中，您必须在部署 Kafka Connect 前创建 ImageStream。镜像流不会被自动创建。



注意

如果使用 **KafkaConnect** 资源来创建集群，之后无法使用 Kafka Connect REST API 创建或更新连接器。您仍然可以使用 REST API 来检索信息。

其他资源

- 在 OpenShift 中部署和管理 AMQ Streams 中的配置 Kafka 连接。
- 在 OpenShift 中部署和管理 AMQ Streams 中自动构建新容器镜像 。

2.2.1. 使用 AMQ Streams 部署 Debezium

按照相同的步骤部署每种 Debezium 连接器。下面的部分论述了如何部署 Debezium MySQL 连接器。

使用早期版本的 AMQ Streams 时，要在 OpenShift 上部署 Debezium 连接器，您需要首先为连接器构建 Kafka Connect 镜像。在 OpenShift 上部署连接器的当前首选方法是使用 AMQ Streams 中的构建配置来构建 Kafka Connect 容器镜像，其中包含您要使用的 Debezium 连接器插件。

在构建过程中，AMQ Streams Operator 将 **KafkaConnect** 自定义资源（包括 Debezium 连接器定义）中的输入参数转换为 Kafka Connect 容器镜像。构建会从 Red Hat Maven 存储库或其他配置的 HTTP 服务器下载必要的工件。

新创建的容器被推送到在 **.spec.build.output** 中指定的容器 registry，用于部署 Kafka Connect 集群。在 AMQ Streams 构建 Kafka Connect 镜像后，您可以创建 **KafkaConnector** 自定义资源来启动构建中包含的连接器。

先决条件

- 您可以访问安装了集群 Operator 的 OpenShift 集群。
- AMQ Streams Operator 正在运行。
- 在 OpenShift 中部署和管理 AMQ Streams 所述，Apache Kafka 集群会被部署 。
- Kafka Connect 在 AMQ Streams 上部署
- 您有红帽构建的 Debezium 许可证。
- 已安装 OpenShift **oc** CLI 客户端，或者您可以访问 OpenShift Container Platform Web 控制台。

- 根据您要存储 Kafka Connect 构建镜像的方式，您需要 registry 权限，或者您必须创建 ImageStream 资源：

  将构建镜像存储在镜像 registry 中，如 Red Hat Quay.io 或 Docker Hub

  - 在 registry 中创建和管理镜像的帐户和权限。

  将构建镜像存储为原生 OpenShift ImageStream

  - ImageStream 资源已部署到集群中，以存储新的容器镜像。您必须为集群显式创建 ImageStream。默认无法使用镜像流。如需有关 ImageStreams 的更多信息，请参阅 OpenShift Container Platform 文档中的管理镜像流。

步骤

1. 登录 OpenShift 集群。
2. 为连接器创建 Debezium **KafkaConnect** 自定义资源(CR)，或修改现有的资源。例如，创建一个名为 **dbz-connect.yaml** 的 **KafkaConnect** CR，用于指定 **metadata.annotations** 和 **spec.build** 属性。以下示例显示了一个 **dbz-connect.yaml** 文件的摘录，该文件描述了 **KafkaConnect** 自定义资源。

   例 2.1. 定义包含 Debezium 连接器的 KafkaConnect 自定义资源的 dbz-connect.yaml 文件

   在以下示例中，自定义资源被配置为下载以下工件：

   - Debezium 连接器存档。
   - 红帽构建的 Apicurio Registry 存档。Apicurio Registry 是一个可选组件。只有在打算将 Avro 序列化与连接器搭配使用时，才添加 Apicurio Registry 组件。
   - Debezium 脚本 SMT 归档以及您要与 Debezium 连接器一起使用的关联脚本引擎。SMT 归档和脚本语言依赖项是可选组件。只有在打算使用 Debezium 的基于内容的路由 SMT 或 过滤 SMT 时，才添加这些组件。

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: debezium-kafka-connect-cluster
  annotations:
    strimzi.io/use-connector-resources: "true" 1
spec:
  version: 3.6.0
  build: 2
    output: 3
      type: imagestream 4
      image: debezium-streams-connect:latest
    plugins: 5
      - name: debezium-connector-mysql
        artifacts:
          - type: zip 6
            url: https://maven.repository.redhat.com/ga/io/debezium/debezium-connector-mysql/2.5.4.Final-redhat-00001/debezium-connector-mysql-2.5.4.Final-redhat-00001-plugin.zip 7
          - type: zip
```

```
          url: https://maven.repository.redhat.com/ga/io/apicurio/apicurio-registry-distro-connect-converter/2.4.4.Final-redhat-<build-number>/apicurio-registry-distro-connect-converter-2.4.4.Final-redhat-<build-number>.zip 8
        - type: zip
          url: https://maven.repository.redhat.com/ga/io/debezium/debezium-scripting/2.5.4.Final-redhat-00001/debezium-scripting-2.5.4.Final-redhat-00001.zip 9
        - type: jar
          url: https://repo1.maven.org/maven2/org/apache/groovy/groovy/3.0.11/groovy-3.0.11.jar 10
        - type: jar
          url: https://repo1.maven.org/maven2/org/apache/groovy/groovy-jsr223/3.0.11/groovy-jsr223-3.0.11.jar
        - type: jar
          url: https://repo1.maven.org/maven2/org/apache/groovy/groovy-json3.0.11/groovy-json-3.0.11.jar

  bootstrapServers: debezium-kafka-cluster-kafka-bootstrap:9093

  ...
```

表 2.1. Kafka Connect 配置设置的描述

项	描述
1	将 **strimzi.io/use-connector-resources** 注解设置为 **"true"**，使 Cluster Operator 使用 **KafkaConnector** 资源在此 Kafka Connect 集群中配置连接器。
2	**spec.build** 配置指定在镜像中存储构建镜像的位置，并列出要在镜像中包含的插件，以及插件工件的位置。
3	**build.output** 指定存储新构建镜像的 registry。
4	指定镜像输出的名称和镜像名称。**output.type** 的有效值是 **要推送到** 容器 registry（如 Docker Hub 或 Quay）或 **镜像流** 的有效值，以将镜像推送到内部 OpenShift ImageStream。要使用 ImageStream，必须将 ImageStream 资源部署到集群中。有关在 KafkaConnect 配置中指定 **build.output** 的更多信息，请参阅 {Name configuration StreamsOpenShift} 中的 AMQ Streams Build schema 参考。
5	**plugins** 配置列出了您要包含在 Kafka Connect 镜像中的所有连接器。对于列表中的每个条目，指定一个插件**名称**，以及有关构建连接器所需的工件的信息。另外，对于每个连接器插件，您还可以包含可用于连接器的其他组件。例如，您可以添加 Service Registry 工件或 Debezium 脚本组件。
6	**artifacts.type** 的值指定在 **artifacts.url** 中指定的工件类型。有效类型为 **zip**、**tgz** 或 **jar**。Debezium 连接器存档以 **.zip** 文件格式提供。**类型** 值必须与 **url** 字段中引用的文件类型匹配。
7	**artifacts.url** 的值指定 HTTP 服务器的地址，如 Maven 存储库，用于存储连接器工件的文件。Debezium 连接器工件在 Red Hat Maven 存储库中提供。OpenShift 集群必须有权访问指定的服务器。

项	描述
8	（可选）指定用于下载 Apicurio Registry 组件的工件 **类型和 url**。包含 Apicurio Registry 工件，只有在您希望连接器使用 Apache Avro 来序列化带有红帽构建的 Apicurio Registry 的值，而不是使用默认的 JSON 转换程序时。
9	（可选）指定 Debezium 脚本 SMT 归档的工件 **类型和 url**，以用于 Debezium 连接器。只有在打算使用 Debezium 的基于内容的路由 SMT 或 过滤 SMT 时才包括脚本 SMT。要使用脚本 SMT，您必须部署 JSR 223 兼容脚本实现，如 groovy。
10	（可选）指定 JSR 223 兼容脚本实施的 JAR 文件的工件 **类型和 url**，这是 Debezium 脚本 SMT 所需的。  **重要** 如果使用 AMQ Streams 将连接器插件合并到 Kafka Connect 镜像中，每个所需的脚本语言 **工件。url** 必须指定 JAR 文件的位置，并且 **artifacts.type** 的值也必须设置为 **jar**。无效的值会导致连接器在运行时失败。 要启用带有脚本 SMT 的 Apache Groovy 语言，示例中的自定义资源会为以下库检索 JAR 文件： • **groovy** • **Groovy-jsr223**（指定代理） • **groovy-json**（解析 JSON 字符串的模块） 作为替代方案，Debebe Debezium 脚本 SMT 也支持使用 JSR 223 实现 GraalVM JavaScript。

3. 输入以下命令将 **KafkaConnect** 构建规格应用到 OpenShift 集群：

```
oc create -f dbz-connect.yaml
```

根据自定义资源中指定的配置，Streams Operator 准备要部署的 Kafka Connect 镜像。构建完成后，Operator 将镜像推送到指定的 registry 或 ImageStream，并启动 Kafka Connect 集群。集群中提供了您在配置中列出的连接器工件。

4. 创建一个 **KafkaConnector** 资源来定义您要部署的每个连接器的实例。
例如，创建以下 **KafkaConnector** CR，并将它保存为 **mysql-inventory-connector.yaml**

例 2.2. 为 Debezium 连接器定义 KafkaConnector 自定义资源的 mysql-inventory-connector.yaml 文件

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
  labels:
```

```
    strimzi.io/cluster: debezium-kafka-connect-cluster
  name: inventory-connector-mysql (1)
spec:
  class: io.debezium.connector.mysql.MySqlConnector (2)
  tasksMax: 1 (3)
  config: (4)
    schema.history.internal.kafka.bootstrap.servers: debezium-kafka-cluster-kafka-
bootstrap.debezium.svc.cluster.local:9092
    schema.history.internal.kafka.topic: schema-changes.inventory
    database.hostname: mysql.debezium-mysql.svc.cluster.local (5)
    database.port: 3306 (6)
    database.user: debezium (7)
    database.password: dbz (8)
    database.server.id: 184054 (9)
    topic.prefix: inventory-connector-mysql (10)
    table.include.list: inventory.* (11)

    ...
```

表 2.2. 连接器配置设置的描述

项	描述
1	使用 Kafka Connect 集群注册的连接器名称。
2	连接器类的名称。
3	可以同时操作的任务数量。
4	连接器的配置。
5	主机数据库实例的地址。
6	数据库实例的端口号。
7	Debezium 用于连接到数据库的帐户名称。
8	Debezium 用于连接到数据库用户帐户的密码。
9	连接器的唯一数字 ID。
10	数据库实例或集群的主题前缀。 指定的名称只能由字母数字字符或下划线组成。 因为主题前缀被用作从这个连接器接收更改事件的任何 Kafka 主题的前缀，所以该名称在集群中的连接器之间必须是唯一的。 如果连接器与 Avro 连接器集成，则此命名空间也用于相关 Kafka Connect 模式的名称，以及相应 Avro 模式的命名空间。

项	描述
11	连接器捕获更改事件的表列表。

5. 运行以下命令来创建连接器资源：

```
oc create -n <namespace> -f <kafkaConnector>.yaml
```

例如，

```
oc create -n debezium -f mysql-inventory-connector.yaml
```

连接器注册到 Kafka Connect 集群，并开始针对 **KafkaConnector** CR 中的 **spec.config.database.dbname** 指定的数据库运行。连接器 pod 就绪后，Debebe 正在运行。

现在，您已准备好 验证 Debezium 部署。

2.2.2. 验证 Debezium 连接器是否正在运行

如果连接器正确启动且没有错误，它会为每个连接器配置为捕获的表创建一个主题。下游应用程序可以订阅这些主题，以检索源数据库中发生的信息事件。

要验证连接器是否正在运行，您可以从 OpenShift Container Platform Web 控制台或 OpenShift CLI 工具 (oc)执行以下操作：

- 验证连接器状态。
- 验证连接器是否生成主题。
- 验证主题是否填充了读取操作("op":"r")的事件，连接器在每个表的初始快照中生成。

先决条件

- Debezium 连接器部署到 OpenShift 上的 AMQ Streams。
- 已安装 OpenShift **oc** CLI 客户端。
- 访问 OpenShift Container Platform web 控制台。

流程

1. 使用以下方法之一检查 **KafkaConnector** 资源的状态：
   - 在 OpenShift Container Platform Web 控制台中：
     a. 导航到 **Home → Search**。
     b. 在 **Search** 页面中，点 **Resources** 打开 **Select Resource** 框，然后键入 **KafkaConnector**。
     c. 在 **KafkaConnectors** 列表中，点您要检查的连接器的名称，如 **inventory-connector-mysql**。
     d. 在 **Conditions** 部分，验证 **Type** 和 **Status** 列中的值是否已设置为 **Ready** 和 **True**。

- 在终端窗口中：

  a. 使用以下命令：

  ```
  oc describe KafkaConnector <connector-name> -n <project>
  ```

  例如，

  ```
  oc describe KafkaConnector inventory-connector-mysql -n debezium
  ```

  该命令返回类似以下示例的状态信息：

  例 2.3. KafkaConnector 资源状态

  ```
  Name:         inventory-connector-mysql
  Namespace:    debezium
  Labels:       strimzi.io/cluster=debezium-kafka-connect-cluster
  Annotations:  <none>
  API Version:  kafka.strimzi.io/v1beta2
  Kind:         KafkaConnector

  ...

  Status:
    Conditions:
      Last Transition Time:  2021-12-08T17:41:34.897153Z
      Status:                True
      Type:                  Ready
    Connector Status:
      Connector:
        State:      RUNNING
        worker_id:  10.131.1.124:8083
      Name:         inventory-connector-mysql
      Tasks:
        Id:               0
        State:            RUNNING
        worker_id:        10.131.1.124:8083
      Type:               source
    Observed Generation:  1
    Tasks Max:            1
    Topics:
      inventory-connector-mysql.inventory
      inventory-connector-mysql.inventory.addresses
      inventory-connector-mysql.inventory.customers
      inventory-connector-mysql.inventory.geom
      inventory-connector-mysql.inventory.orders
      inventory-connector-mysql.inventory.products
      inventory-connector-mysql.inventory.products_on_hand
  Events:  <none>
  ```

2. 验证连接器是否创建了 Kafka 主题：

   - 通过 OpenShift Container Platform Web 控制台。

a. 导航到 **Home → Search**。

b. 在 **Search** 页面中，点 **Resources** 打开 **Select Resource** 框，然后键入 **KafkaTopic**。

c. 在 **KafkaTopics** 列表中，点您要检查的主题名称，例如 **inventory-connector-mysql.inventory.orders---ac5e98ac6a5d91e04d8ec0dc9078a1ece439081d**。

d. 在 **Conditions** 部分，验证 **Type** 和 **Status** 列中的值是否已设置为 **Ready** 和 **True**。

- 在终端窗口中：

a. 使用以下命令：

```
oc get kafkatopics
```

该命令返回类似以下示例的状态信息：

例 2.4. KafkaTopic 资源状态

```
NAME                                                 CLUSTER
PARTITIONS   REPLICATION FACTOR   READY
connect-cluster-configs                               debezium-kafka-cluster   1
1                 True
connect-cluster-offsets                               debezium-kafka-cluster   25
1                 True
connect-cluster-status                                debezium-kafka-cluster   5
1                 True
consumer-offsets---84e7a678d08f4bd226872e5cdd4eb527fadc1c6a
debezium-kafka-cluster   50          1                 True
inventory-connector-mysql--a96f69b23d6118ff415f772679da623fbbb99421
debezium-kafka-cluster   1           1                 True
inventory-connector-mysql.inventory.addresses---
1b6beaf7b2eb57d177d92be90ca2b210c9a56480         debezium-kafka-cluster
1           1                 True
inventory-connector-mysql.inventory.customers---
9931e04ec92ecc0924f4406af3fdace7545c483b         debezium-kafka-cluster   1
1                 True
inventory-connector-mysql.inventory.geom---
9f7e136091f071bf49ca59bf99e86c713ee58dd5           debezium-kafka-cluster
1           1                 True
inventory-connector-mysql.inventory.orders---
ac5e98ac6a5d91e04d8ec0dc9078a1ece439081d           debezium-kafka-cluster
1           1                 True
inventory-connector-mysql.inventory.products---
df0746db116844cee2297fab611c21b56f82dcef         debezium-kafka-cluster   1
1                 True
inventory-connector-mysql.inventory.products_on_hand---
8649e0f17ffcc9212e266e31a7aeea4585e5c6b5   debezium-kafka-cluster   1
1                 True
schema-changes.inventory                              debezium-kafka-cluster
1           1                 True
strimzi-store-topic---effb8e3e057afce1ecf67c3f5d8e4e3ff177fc55         debezium-
kafka-cluster   1           1                 True
strimzi-topic-operator-kstreams-topic-store-changelog---
b75e702040b99be8a9263134de3507fc0cc4017b  debezium-kafka-cluster  1   1
True
```

3. 检查主题内容。

- 在终端窗口中输入以下命令：

```
oc exec -n <project>  -it <kafka-cluster> -- /opt/kafka/bin/kafka-console-consumer.sh \
>     --bootstrap-server localhost:9092 \
>     --from-beginning \
>     --property print.key=true \
>     --topic=<topic-name>
```

例如，

```
oc exec -n debezium  -it debezium-kafka-cluster-kafka-0 -- /opt/kafka/bin/kafka-console-consumer.sh \
>     --bootstrap-server localhost:9092 \
>     --from-beginning \
>     --property print.key=true \
>     --topic=inventory-connector-mysql.inventory.products_on_hand
```

指定主题名称的格式与 **oc describe** 命令返回的格式与第 1 步中返回，例如 **inventory-connector-mysql.inventory.addresses**。

对于主题中的每个事件，命令会返回类似以下示例的信息：

例 2.5. Debezium 更改事件的内容


```
{"schema":{"type":"struct","fields":
[{"type":"int32","optional":false,"field":"product_id"}],"optional":false,"name":"inventory-connector-mysql.inventory.products_on_hand.Key"},"payload":{"product_id":101}}
{"schema":{"type":"struct","fields":[{"type":"struct","fields":
[{"type":"int32","optional":false,"field":"product_id"},
{"type":"int32","optional":false,"field":"quantity"}],"optional":true,"name":"inventory-connector-mysql.inventory.products_on_hand.Value","field":"before"},
{"type":"struct","fields":[{"type":"int32","optional":false,"field":"product_id"},
{"type":"int32","optional":false,"field":"quantity"}],"optional":true,"name":"inventory-connector-mysql.inventory.products_on_hand.Value","field":"after"},{"type":"struct","fields":
[{"type":"string","optional":false,"field":"version"},
{"type":"string","optional":false,"field":"connector"},
{"type":"string","optional":false,"field":"name"},
{"type":"int64","optional":false,"field":"ts_ms"},
{"type":"string","optional":true,"name":"io.debezium.data.Enum","version":1,"parameters":
{"allowed":"true,last,false"},"default":"false","field":"snapshot"},
{"type":"string","optional":false,"field":"db"},
{"type":"string","optional":true,"field":"sequence"},
{"type":"string","optional":true,"field":"table"},
{"type":"int64","optional":false,"field":"server_id"},
{"type":"string","optional":true,"field":"gtid"},{"type":"string","optional":false,"field":"file"},
{"type":"int64","optional":false,"field":"pos"},{"type":"int32","optional":false,"field":"row"},
{"type":"int64","optional":true,"field":"thread"},
{"type":"string","optional":true,"field":"query"}],"optional":false,"name":"io.debezium.connector.mysql.Source","field":"source"},{"type":"string","optional":false,"field":"op"},
{"type":"int64","optional":true,"field":"ts_ms"},{"type":"struct","fields":
[{"type":"string","optional":false,"field":"id"},
```

{"type":"int64","optional":false,"field":"total_order"},
{"type":"int64","optional":false,"field":"data_collection_order"}],"optional":true,"field":"transaction"}],"optional":false,"name":"inventory-connector-mysql.inventory.products_on_hand.Envelope"},"payload":{"before":null,"after":{"product_id":101,"quantity":3},"source":{"version":"2.5.4.Final-redhat-00001","connector":"mysql","name":"inventory-connector-mysql","ts_ms":1638985247805,"snapshot":"true","db":"inventory","sequence":null,"table":"products_on_hand","server_id":0,"gtid":null,"file":"mysql-bin.000003","pos":156,"row":0,"thread":null,"query":null},"op":"r","ts_ms":1638985247805,"transaction":null}}

在前面的示例中，**有效**负载 值显示连接器快照从表 **inventory.products_on_hand** 生成读取(**op" ="r"**)事件。**product_id** 记录的 **"before"** 状态为 **null**，表示该记录不存在之前的值。**"after"** 状态对于 **product_id** 为 **101** 的项目的 **quantity** 显示为 **3**。

您可以使用多个 Kafka Connect 服务集群和多个 Kafka 集群运行 Debezium。您可以部署到 Kafka Connect 集群的连接器数量取决于数据库事件的卷和速率。

后续步骤

有关部署特定连接器的更多信息，请参阅 Debezium 用户指南中的以下主题：

- 部署 Db2 连接器
- 部署 MongoDB 连接器
- 部署 MySQL 连接器
- 部署 Oracle 连接器
- 部署 PostgreSQL 连接器
- 部署 SQL Server 连接器

附录 A. 使用您的订阅

Debezium 通过软件订阅提供。要管理您的订阅，请访问红帽客户门户中的帐户。

访问您的帐户

1. 转至 access.redhat.com。
2. 如果您还没有帐户，请创建一个帐户。
3. 登录到您的帐户。

激活订阅

1. 转至 access.redhat.com。
2. 导航到 **Subscriptions**。
3. 导航到 **激活订阅** 并输入您的 16 位激活号。

下载 zip 和 tar 文件

要访问 zip 或 tar 文件，请使用客户门户网站查找下载的相关文件。如果您使用 RPM 软件包，则不需要这一步。

1. 打开浏览器并登录红帽客户门户网站 **产品下载页面**，网址为 access.redhat.com/downloads。
2. 向下滚动到 INTEGRATION **和 AUTOMATION**。
3. 点 **Red Hat Integration** 以显示 Red Hat Integration 下载页面。
4. 单击组件的 **Download** 链接。

更新于 2024-04-19