

# Debezium Connector 配置说明文档

## 文件修订记录

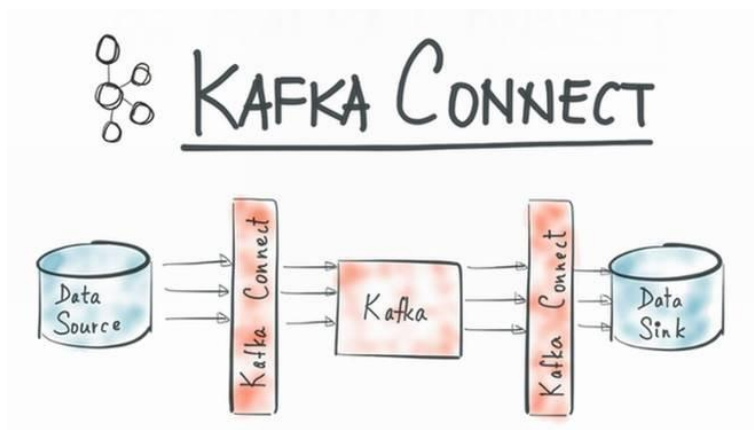
[illegible]

## 一、Kafka Connect

### 1 简介

Kafka Connect 是一个为 Kafka 和外部系统提供可靠的、可扩展的连接的框架。比如可连接的外部系统有数据库、键-值存储、搜索索引、文件系统等等。

### 2 整体框架



### 3 整体框架说明

围绕 Kafka，Kafka Connect 旨在建造一个可伸缩、可靠的数据流通道，通过 Kafka Connect 可以实现一个数据源和目标数据源之间的进行大量数据进出的数据管道。

### 4 Debezium Connector

Debezium 是基于 Kafka Connect 建立的开源连接器。选择 MySQL 作为数据源，通过记录数据库历史操作，监测数据库中增加、删除、修改的这几类操作。通过消费 Kafka 的特定主题，就可以实时监测数据库的变化。并且即使连接器关闭或重启，也不会丢失任一条消息。

## 二、配置说明

### 1. 前期准备工作

1) 开启 zookeeper，默认端口为 2181

# 开启 zookeeper （此处使用 confluent 内置 zookeeper）

```
./bin/zookeeper-server-  
start ./etc/kafka/zookeeper.properties >> ../zookeeper/zookeeper.log &  
(设置后台运行，输出日志文件位于 zookeeper/下)
```

2) 开启 Kafka broker，默认端口为 9092

# 开启 Kafka broker （此处使用 confluent 内置 kafka）

```
./bin/kafka-server-start ./etc/kafka/server.properties
```

3) 配置 MySQL 开启 binlog

# 修改 MySQL 配置文件

```
server-id=2331  
log_bin=mysql-bin  
binlog_format=row
```

server-id 与 connector 中配置一致即可

# 重启 MySQL

# 检查是否开启 binlog，值为 OFF 代表关闭，值为 ON 代表开启

```
mysql> show variables like 'log_bin';  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| log_bin      | ON    |  
+-----+-----+  
1 row in set (0.01 sec)
```

### 4) 开启 Kafka Connect

（代码包中已包含 confluent 文件，可跳过下载配置过程，直接开启 Kafka Connect）

## # 下载 Confluent

<https://www.confluent.io/download/>

Go to the Download Center

Download Current Release (version 3.2.2)

Email (required)

Your Information (optional):

First Name Last Name

Company Name Phone Number

Go

输入邮箱，点击 Go

Confluent Open Source 3.2.2

Download ZIP Download TAR

Install DEB Install RPM

选择 Confluent Open Source 3.2.2, 下载压缩包

## # 解压

```
tar -xzf confluent-oss-3.2.1-2.11.tar.gz
```

## # 解压后目录下文件

bin/	可执行脚本文件
etc/	配置文件
share/	jar 包等额外配置文件
src/	其他资源压缩包
logs/	日志文件（运行后产生）

## # 修改 Kafka Connect 配置文件

etc/schema-registry/connect-avro-distributed.properties

```
key.converter = org.apache.kafka.connect.json.JsonConverter
value.converter = org.apache.kafka.connect.json.JsonConverter
```

## # 开启 Kafka Connect (distributed 模式)

```
./bin/connect-distributed ./etc/schema-registry/connect-avro-
distributed.properties
```

```
ENGINE=InnoDB DEFAULT CHARSET=latin1 (io.debezium.connector.mysql.SnapshotReader:703)
[2017-06-14 22:10:31,564] INFO CREATE TABLE `user` (
  `uid` varchar(5) NOT NULL,
  `name` varchar(10) NOT NULL,
  `addr` varchar(20) NOT NULL,
  `updated_time` timestamp(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE CURRENT_TIMESTAMP(6),
  PRIMARY KEY (`uid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 (io.debezium.connector.mysql.SnapshotReader:703)
[2017-06-14 22:10:31,568] INFO Step 7: releasing global read lock to enable MySQL writes (io.debezium.connector.mysql.SnapshotReader:362)
[2017-06-14 22:10:31,570] INFO Step 7: blocked writes to MySQL for a total of 00:00:00.333 (io.debezium.connector.mysql.SnapshotReader:368)
[2017-06-14 22:10:31,571] INFO Step 8: scanning contents of 5 tables while still in transaction (io.debezium.connector.mysql.SnapshotReader:383)
[2017-06-14 22:10:31,573] INFO Step 8: - scanning table 'kafka.device' (1 of 5 tables) (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,583] INFO Step 8: - Completed scanning a total of 4 rows from table 'kafka.device' after 00:00:00.009 (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,584] INFO Step 8: - scanning table 'kafka.kafkatable' (2 of 5 tables) (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,584] INFO Step 8: - Completed scanning a total of 3 rows from table 'kafka.kafkatable' after 00:00:00.0 (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,585] INFO Step 8: - scanning table 'kafka.testtable' (3 of 5 tables) (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,586] INFO Step 8: - Completed scanning a total of 4 rows from table 'kafka.testtable' after 00:00:00.001 (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,587] INFO Step 8: - scanning table 'kafka.tmst' (4 of 5 tables) (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,590] INFO Step 8: - Completed scanning a total of 8 rows from table 'kafka.tmst' after 00:00:00.002 (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,591] INFO Step 8: - scanning table 'kafka.user' (5 of 5 tables) (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,592] INFO Step 8: - Completed scanning a total of 4 rows from table 'kafka.user' after 00:00:00.001 (io.debezium.connector.mysql.SnapshotReader:428)
[2017-06-14 22:10:31,593] INFO Step 8: scanned 23 rows in 5 tables in 00:00:00.021 (io.debezium.connector.mysql.SnapshotReader:488)
[2017-06-14 22:10:31,593] INFO Step 9: committing transaction (io.debezium.connector.mysql.SnapshotReader:520)
[2017-06-14 22:10:31,593] INFO Completed snapshot in 00:00:00.383 (io.debezium.connector.mysql.SnapshotReader:570)
[2017-06-14 22:10:32,242] WARN Error while fetching metadata with correlation id 1 : {dbserver1=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient:707)
[2017-06-14 22:10:32,392] WARN Error while fetching metadata with correlation id 6 : {dbserver1.kafka.device=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient:707)
[2017-06-14 22:10:32,534] WARN Error while fetching metadata with correlation id 11 : {dbserver1.kafka.kafkatable=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient:707)
[2017-06-14 22:10:32,680] WARN Error while fetching metadata with correlation id 15 : {dbserver1.kafka.testtable=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient:707)
[2017-06-14 22:10:32,817] WARN Error while fetching metadata with correlation id 19 : {dbserver1.kafka.tmst=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient:707)
[2017-06-14 22:10:32,975] WARN Error while fetching metadata with correlation id 23 : {dbserver1.kafka.user=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient:707)
Jun 14, 2017 10:10:33 PM com.github.shyiko.mysql.binlog.BinaryLogClient connect
INFO: Connected to localhost:3306 at mysql-bin.000001/9436 (sid:184054, cid:6)
[2017-06-14 22:10:33,174] INFO Connected to MySQL binlog at localhost:3306, starting at binlog file 'mysql-bin.000001', pos=9436, skipping 0 events plus 0 rows (io.debezium.connector.mysql.SnapshotReader:570)
[2017-06-14 22:10:44,610] INFO Reflections took 14550 ms to scan 568 urls, producing 13198 keys and 86314 values (org.reflections.Reflections:229)
^[[A[2017-06-14 22:11:30,552] INFO Finished WorkerSourceTask{id=inventory-connector-0} commitOffsets successfully in 17 ms (org.apache.kafka.connect.runtime.WorkerSourceTask:100)
```

## # 检查 Kafka Connect 状态 (默认本地开启, 端口 8083)

```
curl -H "Accept:application/json" localhost:8083/
```

若未开启, 则连接失败

```
curl: (7) Failed to connect to localhost port 8083: Connection refused
```

若成功开启, 则可查看 Kafka Connect 版本信息

```
{"version":"0.10.2.1-cp1","commit":"078e7dc02a100018"}
```

## # 列出当前所有注册的 connector

```
curl -H "Accept:application/json" localhost:8083/connectors/
```

或者

```
curl -X GET localhost:8083/connectors
```

结果显示所有 connector 的名称列表, 若无, 则为 "[]" 空列表

## 5) 注册 Debezium Connector

(代码包中 `confluent` 文件夹下已包含 `debezium` 插件，可跳过下载拷贝过程，直接开启 `Kafka Connect`，并注册连接器)

# 下载 Debezium Connector plugin archives

<http://debezium.io/docs/install/>

### Installing a Debezium connector

If you've already installed Zookeeper, Kafka, and Kafka Connect, then using one of Debezium's connectors is easy. Simply download one or more connector plugin archives (see below), extract its files into your Kafka Connect environment, and add the directory with the JARs to Kafka Connect's classpath. Restart your Kafka Connect process to pick up the new JARs.

The connector plugins are available from Maven:

- [MySQL Connector plugin archive](#)
- [MongoDB Connector plugin archive](#)

If immutable containers are your thing, then check out Debezium's [Docker images](#) for Zookeeper, Kafka, and Kafka Connect with the MySQL and MongoDB connectors already pre-installed and ready to go. Our [tutorial](#) even walks you through using these images, and this is a great way to learn what Debezium is all about. You can even [run Debezium on Kubernetes and OpenShift](#).

选择 MySQL Connector plugin archive 下载

# 解压

```
tar -xzvf debezium-connector-mysql-0.5.0-plugin.tar.gz
```

将解压后所有 jar 包：

```
debezium-connector-mysql-0.5.0.jar
debezium-core-0.5.0.jar
LICENSE.txt
mysql-binlog-connector-java-0.9.0.jar
mysql-connector-java-5.1.40.jar
```

复制至 `confluent` 目录，`share/java/ kafka-connect-jdbc` 位置下

# 重启 Kafka Connect

否则在注册 Debezium Connector 时，将无法找到相关类

(也可先进行上述拷贝操作，再直接开启 `Kafka Connect`)

# 注册新的 Debezium Connector，其中 JSON 格式配置按下方配置信息设置

```
curl -i -X POST -H "Accept:application/json" -H "Content-Type:application/json" localhost:8083/connectors/ -d '{ "name": "debe-connector", "config": { "connector.class":
```

```
"io.debezium.connector.mysql.MySqlConnector", "tasks.max": "1",
"database.hostname": "localhost", "database.port": "3306",
"database.user": "root", "database.password": "xxx",
"database.server.id": "2331", "database.server.name": "dbserver",
"database.whitelist": "kafka",
"database.history.kafka.bootstrap.servers": "localhost:9092",
"database.history.kafka.topic": "dbhistory.debetopic" } }'
```

```
HTTP/1.1 201 Created
Date: Thu, 13 Jul 2017 08:50:50 GMT
Location: http://localhost:8083/connectors/debe-connector
Content-Type: application/json
Content-Length: 501
Server: Jetty(9.2.15.v20160210)
```

注册成功！返回 connector 信息

## # 具体 connector 配置信息

```
{
  "name": "debe-connector", //connector 名称
  "config": {
    "connector.class": "io.debezium.connector.mysql.MySqlConnector",
    "tasks.max": "1",
    "database.hostname": "localhost",
    "database.port": "3306", // 数据库地址及端口
    "database.user": "root",
    "database.password": "xxx", // 数据库用户名及密码
    "database.server.id": "2331", // 数据库 id, 与 mysql 配置 id 一致
    "database.server.name": "dbserver", //设定 topic 前缀
    "database.whitelist": "kafka", //mysql 中监听的 database 白名单
    "database.history.kafka.bootstrap.servers": "localhost:9092", // 入口
    broker 地址
    "database.history.kafka.topic": "dbhistory.debetopic",
```



```
}
}
```

其他详细配置见 <http://debezium.io/docs/connectors/mysql/#configuration>

## # 涉及 topic

1. dbhistory.debetopic 由 database.history.kafka.topic 配置
2. dbserver 由 database.server.name 配置, 表示 schema change topic
3. dbserver.kafka.xxx 由 database.server.name 作为前缀, kafka 为 database name, 后缀加上 table name: xxx, 表示数据库中表有增、删、改变化的 topic

## # 查看 connector 信息或状态

```
curl -X GET localhost:8083/connectors/debe-connector
curl -X GET localhost:8083/connectors/debe-connector/status
```

```
{"name":"debe-connector","connector":{"state":"RUNNING","worker_id":"127.0.1.1:8083"},"tasks":[{"state":"RUNNING","id":0,"worker_id":"127.0.1.1:8083"}]}
```

## 6) 运行同步代码

### # 代码包 debezium-mysql-connector 文件目录

bin/	可执行脚本文件
config/	配置文件
lib/	核心代码
models/	数据库封装操作
node_modules/	node 模块
confluent-3.2.1/	confluent 文件, 包含 kafka connect
zookeeper/	zookeeper log 文件 (若单独开启 zookeeper, 则可忽略)
package.json	npm 文件
pm2.json	pm2 文件

### # 安装 node 模块

```
npm install
根据 package.json 自动安装所有依赖
```

# 根据所需监测的数据库表配置 config/config.json 文件

```
{
  "topics": {
    "serverName": "dbserver",
    "databaseName": "kafka",
    "tableWhiteList": ["Users", "Devices", "Groups",
      "User2Devices", "Group2Devices",
      "Developers", "Products", "OTAs",
      "Developer2Pros", "Product2OTAs"]
  },
  "options": {
    "host": "localhost:2181",
    "ssl": false,
    "groupId": "SyncGroup",
    "id": "consumer1",
    "protocol": ["roundrobin"],
    "fromOffset": "latest",
    "outOfRangeOffset": "earliest",
    "migrateHLC": false,
    "migrateRolling": true
  }
}
```

1. 配置数据库名称 databaseName
2. 配置监测的表格 tableWhiteList, 列表形式

# 运行同步程序

```
pm2 start pm2.json
```

```
[PM2][WARN] You are starting 1 processes in fork_mode without load balancing. To enable it remove -x o
[PM2] Applying action restartProcessId on app [debezium-mysql-connector](ids: 2)
[PM2] [debezium-mysql-connector](2) ✓
```

App name	id	mode	pid	status	restart	uptime	cpu	mem	watchin
debezium-mysql-connector	2	fork	30119	online	0	0s	0%	11.7 MB	disab

# 查看 log 文件

```
tail -f /tmp/debezium-mysql-connector.log
```

```
connected.

coming message...
consumer1 from Topic=dbserver.kafka.Users Partition=0 Offset=12
Operation: updateUsers.
connected.
```

(日志文件地址可在 pm2.json 中配置)

### 三、参考网址

<http://debezium.io/>

<http://docs.confluent.io/3.2.1/>

<http://docs.confluent.io/current/connect/quickstart.html>