**尚硅谷大数据之Debezium**

(作者：尚硅谷研究院)

版本：V1.7

# Debezium介绍

Debezium是用于捕获变更数据的开源工具。可以响应数据库的所有插入，更新和删除操作.

它是一种CDC（Change Data Capture）工具，工作原理与大家所熟知的Canal, Maxwell一样，均是通过抽取数据库日志来获取变更的。

Debezium构建在Apache Kafka之上，并提供Kafka连接器来监视特定的数据库管理系统。

目前Debezium最新稳定版是:1.7

更多介绍参考官网: <https://debezium.io/>

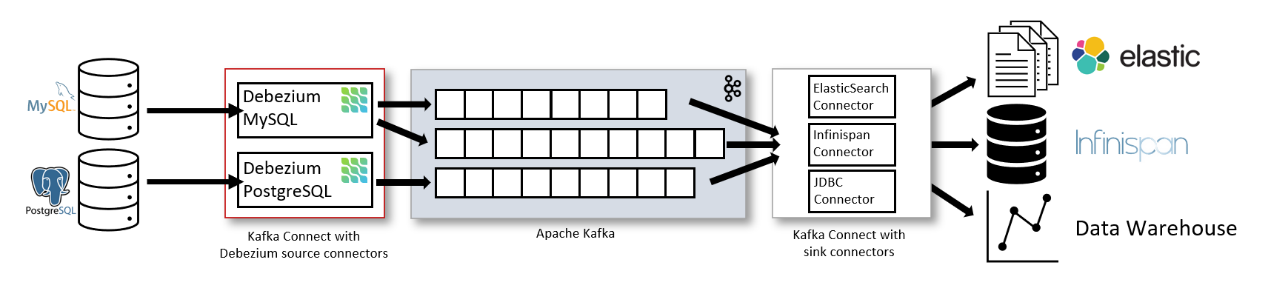
整个课程基于Centos7.5操作

# Debezium特点

1. 简单. 使用简单, 容易上手
2. 快速稳定. 构建于Kafka之上, 可扩展，经官方验证可处理大容量的数据。
3. 能监控多种数据库. 目前支持的数据库包括Mysql, Oracle, MongoDB, PostgreSQL, SQL Server, Db2, Cassandra, Vitess.

# Debezium架构

## 基于Kafka Connector 部署Debezium

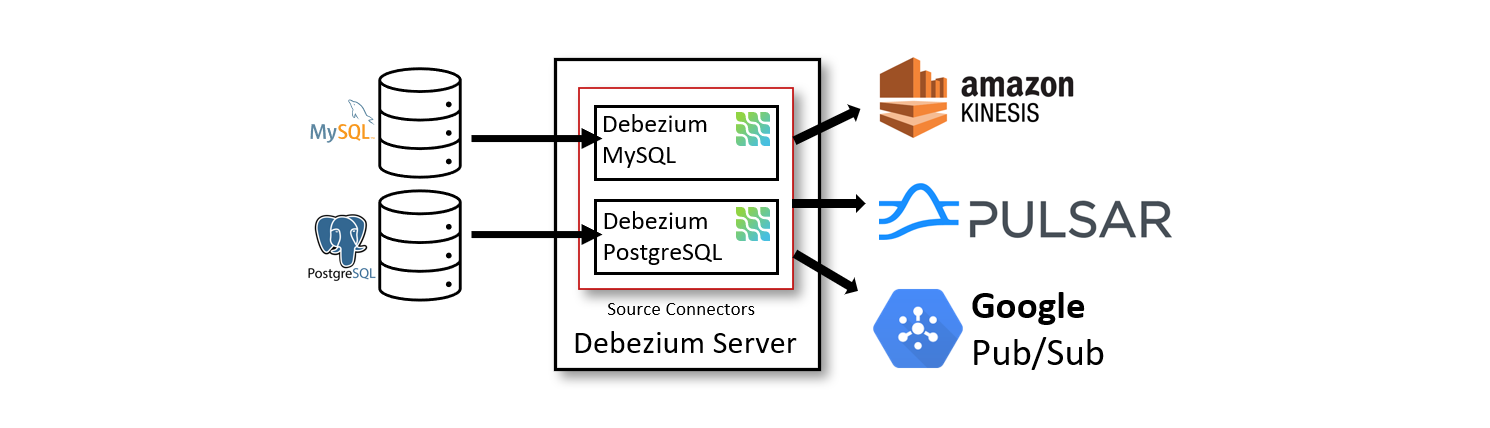


Kafka连接器包含两种:

1. source connector: 负责把数据库中的记录发送到Kafka(Debezium)
2. sink connector: 负责把kafka topic中的数据发送到其他系统,如es等(flume等)

## 使用Debezium Server部署Debezium

Debezium 服务器是一个可配置的、随时可用的应用程序，它将变化数据从源数据库流式传输到其他系统.(目前只支持sink到kinesis)



## 嵌入式引擎

另外一种使用方式:Debezium作为嵌入式引擎, 内置到其他应用中, 然后就可以实时读取数据库的变化数据了.

目前比较火的 Flink的CDC, 就是内置了Debezium.

学习Flink的同学可以参考: <https://ververica.github.io/flink-cdc-connectors/master/>

# 案例1:监控Mysql

## mysql, zookeeper,Kafka安装和配置

略. 具体操作可参考尚硅谷以前发布的视频和文档.

## Mysql数据库准备

### Mysql开启binlog

sudo vim /etc/my.cnf

[mysqld]

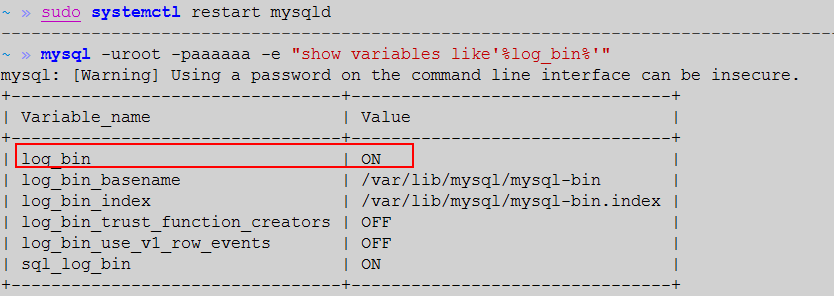
server-id= 1

log-bin=mysql-bin

binlog\_format=row

注意:

1. 万万不可修改/etc/my.cnf的权限, 否则msyql就会忽略这个配置文件.
2. 修改成功之后重启mysql: sudo systemctl restart mysqld
3. 检测binlog是否开启成功: mysql -uroot -paaaaaa -e "show variables like'%log\_bin%'"



### 准备测试数据库和表

启动Mysql客户端, 创建数据库和表用于测试

CREATE DATABASE `atguigu2022` CHARACTER SET utf8 COLLATE utf8\_general\_ci;

use atguigu2022;

create table stu(id int primary key, name varchar(255), age int);

## 安装Mysql Connector

### 下载Mysql Connector

wget https://repo1.maven.org/maven2/io/debezium/debezium-connector-mysql/1.7.1.Final/debezium-connector-mysql-1.7.1.Final-plu

gin.tar.gz

### 解压Mysql Connector

mkdir -p /opt/module/debezium/connector

tar -zxvf debezium-connector-mysql-1.7.1.Final-plugin.tar.gz -C /opt/module/debezium/connector/

### 配置Mysql Connector插件

打开kafka配置文件connect-distributed.properties 进行配置

vim /opt/module/kafka-2.4.1/config/connect-distributed.properties

bootstrap.servers=hadoop162:9092,hadoop163:9092,hadoop164:9092

group.id=connect-mysql

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

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

key.converter.schemas.enable=false

value.converter.schemas.enable=false

status.storage.topic=connect-mysql-status

status.storage.replication.factor=2

offset.flush.interval.ms=10000

plugin.path=/opt/module/debezium/connector

注意: 不要忘记分发配置文件到Kafka其他节点

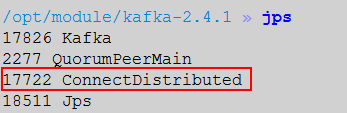
## 启动各个组件

### 启动mysql, zookeeper,kafka

### 启动Kafka connector

在hadoop162启动Kafka connector

/opt/module/kafka-2.4.1/bin/connect-distributed.sh -daemon /opt/module/kafka-2.4.1/config/connect-distributed.properties



### 检测Kafka connector是否正常工作

1. 检测kafka连接器的服务状态

curl -H "Accept:application/json" hadoop162:8083/



1. 检查向 Kafka Connect 注册的连接器列表

curl -H "Accept:application/json" hadoop162:8083/connectors/



返回空列表, 表示目前还没有注册的连接器

## 部署Debezium Mysql Connector

### 配置信息

{

"name": "atguigu-mysql-connector",

"config": {

"connector.class": "io.debezium.connector.mysql.MySqlConnector",

"database.hostname": "hadoop162",

"database.port": "3306",

"database.user": "root",

"database.password": "aaaaaa",

"database.server.id": "184054",

"database.server.name": "bigdata",

"database.include.list": "atguigu2022",

"database.history.kafka.bootstrap.servers": "hadoop162:9092",

"database.history.kafka.topic": "schema-changes.inventory"

}

}

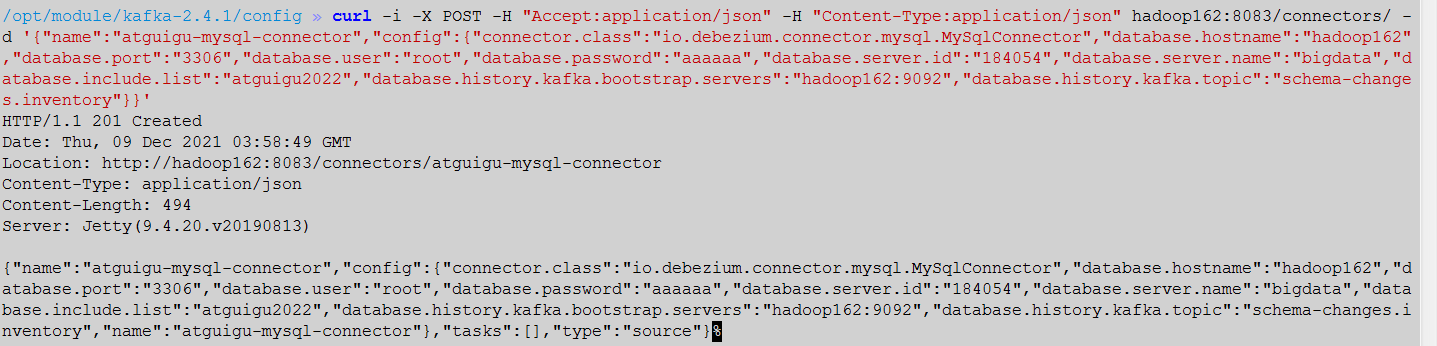
说明:

1. name: 连接器名字
2. database.include.list: 要监控的数据库列表
3. database.server.name: 服务器名. 会成为topic的前缀

### 注册连接器

Kafka Connect 服务的 API 提交POST针对/connectors资源的请求，其中包含描述新连接器（称为inventory-connector）的 JSON 文档。

curl -i -X POST -H "Accept:application/json" -H "Content-Type:application/json" hadoop162:8083/connectors/ -d '{"name":"atguigu-mysql-connector","config":{"connector.class":"io.debezium.connector.mysql.MySqlConnector","database.hostname":"hadoop162","database.port":"3306","database.user":"root","database.password":"aaaaaa","database.server.id":"184054","database.server.name":"bigdata","database.include.list":"atguigu2022","database.history.kafka.bootstrap.servers":"hadoop162:9092","database.history.kafka.topic":"schema-changes.inventory"}}'



### 测试

在表中插入或修改或删除确认是否在kafka中收到数据

注意: 默认情况一张表一个topic, topic名字就是: 服务器名.数据名.表名

1. 插入数据

insert into stu values(10, 'zs', 18);

{  
    "before":null,  
    "after":{  
        "id":10,  
        "name":"zs",  
        "age":18  
    },  
    "source":{  
        "version":"1.7.1.Final",  
        "connector":"mysql",  
        "name":"bigdata",  
        "ts\_ms":1639022905000,  
        "snapshot":"false",  
        "db":"atguigu2022",  
        "sequence":null,  
        "table":"stu",  
        "server\_id":1,  
        "gtid":null,  
        "file":"mysql-bin.000001",  
        "pos":1590,  
        "row":0,  
        "thread":null,  
        "query":null  
    },  
    "op":"c",  
    "ts\_ms":1639022905284,  
    "transaction":null  
}

1. 更新数据

update stu set age=19 where id=10;

{  
    "before":{  
        "id":10,  
        "name":"zs",  
        "age":18  
    },  
    "after":{  
        "id":10,  
        "name":"zs",  
        "age":19  
    },  
    "source":{  
        "version":"1.7.1.Final",  
        "connector":"mysql",  
        "name":"bigdata",  
        "ts\_ms":1639023015000,  
        "snapshot":"false",  
        "db":"atguigu2022",  
        "sequence":null,  
        "table":"stu",  
        "server\_id":1,  
        "gtid":null,  
        "file":"mysql-bin.000001",  
        "pos":1870,  
        "row":0,  
        "thread":null,  
        "query":null  
    },  
    "op":"u",  
    "ts\_ms":1639023015525,  
    "transaction":null  
}

1. 删除数据

delete from stu where id=10;

{  
    "before":{  
        "id":10,  
        "name":"zs",  
        "age":19  
    },  
    "after":null,  
    "source":{  
        "version":"1.7.1.Final",  
        "connector":"mysql",  
        "name":"bigdata",  
        "ts\_ms":1639023085000,  
        "snapshot":"false",  
        "db":"atguigu2022",  
        "sequence":null,  
        "table":"stu",  
        "server\_id":1,  
        "gtid":null,  
        "file":"mysql-bin.000001",  
        "pos":2164,  
        "row":0,  
        "thread":null,  
        "query":null  
    },  
    "op":"d",  
    "ts\_ms":1639023085455,  
    "transaction":null  
}

# 案例2:监控PostgreSQL

## 安装PostgreSQL

# Install the repository RPM:

sudo yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86\_64/pgdg-redhat-repo-latest.noarch.rpm

# Install PostgreSQL:

sudo yum install -y postgresql14-server

# Optionally initialize the database and enable automatic start:

sudo /usr/pgsql-14/bin/postgresql-14-setup initdb

sudo systemctl enable postgresql-14

sudo systemctl start postgresql-14

## PostgreSQL配置

1. 开启逻辑复制允许监听所有ip

sudo vim /var/lib/pgsql/14/data/postgresql.conf

修改:

listen\_addresses = '\*'

添加:

wal\_level = logical

max\_wal\_senders = 2

max\_replication\_slots = 1

sudo vim /var/lib/pgsql/14/data/pg\_hba.conf

添加

host all all 0.0.0.0/0 scram-sha-256

host replication all 0.0.0.0.0 trust

重启数据库:

sudo systemctl restart postgresql-14

1. 创建用户和库

sudo -i -u postgres

psql

创建新用户

CREATE USER sz\_atguigu WITH PASSWORD 'aaaaaa';

创建用户数据库

CREATE DATABASE db\_gmall OWNER sz\_atguigu;

把库的所有权赋给用户

GRANT ALL PRIVILEGES ON DATABASE db\_gmall TO sz\_atguigu;

给用户添加application权限

ALTER USER sz\_atguigu REPLICATION superuser;

退出

exiut

删除postgres用户密码

sudo passwd -d postgres

重新设置密码

sudo passwd postgres

创建新用户与刚才pgsql中的用户同名

sudo adduser sz\_atguigu

sudo passwd sz\_atguigu

用刚才新创建的用户登录

su - sz\_atguigu

psql -d db\_gmall

1. 建库建表

create table stu(id int, name varchar(25));

insert into stu values(1, 'lisi')

## 部署PostgreSQL Connector

### 下载连接器

wget <https://repo1.maven.org/maven2/io/debezium/debezium-connector-postgres/1.7.1.Final/debezium-connector-postgres-1.7.1.Final-plugin.tar.gz>

tar -zxvf debezium-connector-postgres-1.7.1.Final-plugin.tar.gz -C /opt/module/debezium/connector

### 配置信息

{  
    "name":"atguigu-pgsql-connector",  
    "config":{  
        "connector.class":"io.debezium.connector.postgresql.PostgresConnector",  
        "database.hostname":"192.168.1813",  
        "database.port":"5432",  
        "database.user":"sz\_atguigu",  
        "database.password":"aaaaaa",  
        "database.dbname":"db\_gmall",  
        "database.server.name":"server5",  
        "plugin.name":"pgoutput"  
    }  
}

### 注册连接器

curl -X POST http://hadoop162:8083/connectors -H "Content-Type: application/json" -d '{"name":"atguigu-pgsql-connector","config":{"connector.class":"io.debezium.connector.postgresql.PostgresConnector","database.hostname":"192.168.18.163","database.port":"5432","database.user":"sz\_atguigu","database.password":"aaaaaa","database.dbname":"db\_gmall","database.server.name":"server5", "plugin.name": "pgoutput"}}'

# 案例3:监控Sql Server

## 安装SQL Server数据库

参考尚硅谷视频: <https://www.bilibili.com/video/BV1H44y1x76X?p=18>

## 准备数据库和表

sqlcmd -S localhost -U SA -P Huilove521

create database atguigudb

go

use atguigudb

go

create table stu(id int,name varchar(25))

go

insert into stu values(1,'zhangsan')

go

## 为sql server 开启CDC功能

### 开启 Sql Server agent

sudo /opt/mssql/bin/mssql-conf set sqlagent.enabled true

sudo systemctl restart mssql-server.service

### 给数据库开启cdc

use atguigudb

go

EXEC sys.sp\_cdc\_enable\_db

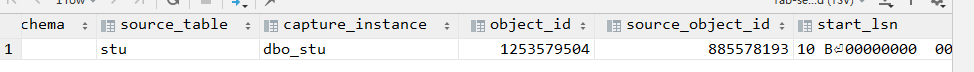
go

### 给表开启cdc

**use** atguigudb  
**go  
EXEC** sys.*sp\_cdc\_enable\_table* @source\_schema = **N'dbo'**,  
 @source\_name = **N'stu'**,  
 @role\_name = **NULL**,  
 @supports\_net\_changes = 0  
**GO**

### 验证是否开启成功

**USE** atguigudb;  
**GO  
EXEC** sys.sp\_cdc\_help\_change\_data\_capture  
**GO**



如果有记录表示开启成功, 否则需要查找原因并, 重新开启

## 部署SQL Server Connector

### 下载SQL Server Connector

wget <https://repo1.maven.org/maven2/io/debezium/debezium-connector-sqlserver/1.7.1.Final/debezium-connector-sqlserver-1.7.1.Final-plugin.tar.gz>

tar -zxvf debezium-connector-sqlserver-1.7.1.Final-plugin.tar.gz -C /opt/module/debezium/connector

### 配置信息

{

"name": "atguigu-sql-server-connector",

"config": {

"connector.class": "io.debezium.connector.sqlserver.SqlServerConnector",

"database.hostname": "192.168.18.162",

"database.port": "1433",

"database.user": "sa",

"database.password": "Huilove621",

"database.dbname": "atguigudb",

"database.server.name": "server2",

"table.include.list": "dbo.stu",

"database.history.kafka.bootstrap.servers": "hadoop162:9092",

"database.history.kafka.topic": "dbhistory.fullfillment"

}

}

### 注册连接器

curl -X POST http://hadoop162:8083/connectors -H "Content-Type: application/json" -d '{"name":"atguigu-sql-server-connector","config":{"connector.class":"io.debezium.connector.sqlserver.SqlServerConnector","database.hostname":"192.168.18.162","database.port":"1433","database.user":"sa","database.password":"Huilove521","database.dbname":"atguigudb","database.server.name":"server2","table.include.list":"dbo.stu","database.history.kafka.bootstrap.servers":"hadoop162:9092","database.history.kafka.topic":"dbhistory.fullfillment"}}'

# 案例4:监控MongoDB

## 安装MongoDB数据库

tar -zxvf mongodb-linux-x86\_64-rhel70-5.0.5.tgz -C /opt/module

mv mongodb-linux-x86\_64-rhel70-5.0.5 mongodb

## 设置MongoDB

MongoDB 连接器使用 MongoDB 的 oplog 来捕获更改，因此该连接器仅适用于 MongoDB 副本集或分片集群.

为方便操作, 我们在一个节点上启动3个mongoDB进程, 来模拟三个副本

1. 先创建三个目录来分别存放这三个节点的数据

sudo mkdir -p /data/mongodb

sudo chmod 777 /data/mongodb

mkdir -p /data/mongodb/a /data/mongodb/b /data/mongodb/c

1. 分别启动三个mongod 进程，端口分别为：27017，27018，27019

bin/mongod --replSet rs0 --port 27017 --bind\_ip hadoop162 --dbpath /data/mongodb/a --oplogSize 128

bin/mongod --replSet rs0 --port 27018 --bind\_ip hadoop162 --dbpath /data/mongodb/b --oplogSize 128

bin/mongod --replSet rs0 --port 27019 --bind\_ip hadoop162 --dbpath /data/mongodb/c --oplogSize 128

1. 登录其中一个, 搭建副本集

bin/mongo --host hadoop162 --port 27017

rsconf = {

\_id: "rs0",

members: [

{

\_id: 0,

host: "hadoop162:27017"

},

{

\_id: 1,

host: "hadoop162:27018"

},

{

\_id: 2,

host: "hadoop162:27019"

}

]

}

rs.initiate( rsconf )

1. 准备数据

use test

db.createCollection("stu")

db.stu.insert({"name": "zs", "age":20})

## 部署MongoDB Connector

### 下载连接器

wget <https://repo1.maven.org/maven2/io/debezium/debezium-connector-mongodb/1.7.1.Final/debezium-connector-mongodb-1.7.1.Final-plugin.tar.gz>

tar zxvf debezium-connector-mongodb-1.7.1.Final-plugin.tar.gz -C /opt/module/debezium/connector

### 配置

{

"name": "atguigu-mongodb-connector",

"config": {

"connector.class": "io.debezium.connector.mongodb.MongoDbConnector",

"mongodb.hosts": "rs0/192.168.18.162:27017",

"mongodb.name": "server3",

"collection.include.list": "\*"

}

}

### 注册连接器

curl -X POST http://hadoop162:8083/connectors -H "Content-Type: application/json" -d '{"name":"atguigu-mongodb-connector","config":{"connector.class":"io.debezium.connector.mongodb.MongoDbConnector","mongodb.hosts":"rs0/192.168.18.162:27017","mongodb.name":"server3"}'

# 常见问题

1. Debezium能不能同步历史数据?

可以. 使用参数: snapshot.mode= initial 来控制. 当第一次监控某个表的时候会先同步所有历史数据, 然后再通过mysql的binlog监控所有更新数据

参考: https://debezium.io/documentation/reference/1.7/connectors/mysql.html#mysql-property-snapshot-mode

1. Debezium同步历史数据(做快照)的时候, 这个时候如果有向Mysql对应的表写数据怎么办?

给表加锁. [snapshot.locking.mode](https://debezium.io/documentation/reference/1.7/connectors/mysql.html#mysql-property-snapshot-locking-mode)= minimal

1. 默认情况是一个表一个topic, 是否支持某个库下所有的表进入同一个topic?

支持.使用 topic routing. 但是有个前提条件, 这些表要有相同的schema, 也就是表结构.

具体参考: https://debezium.io/documentation/reference/1.7/transformations/topic-routing.html