**Canal&Maxwell**

**实时数据同步工具**

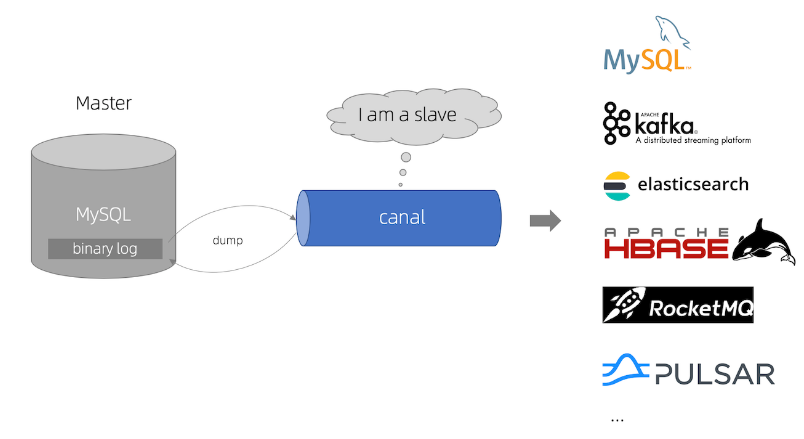
## Canal介绍

### **什么是Canal**

Canal [kə'næl]，译意为水道/管道/沟渠，主要用途是基于 MySQL 数据库增量日志解析，提供增量数据订阅和消费。

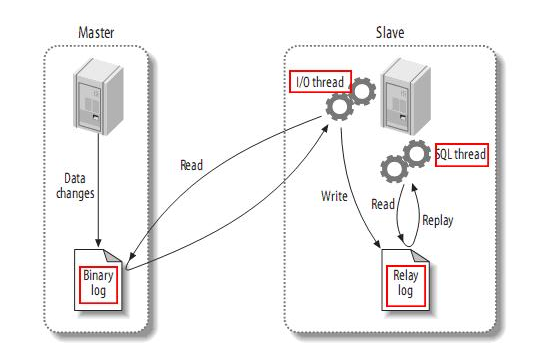
早期阿里巴巴因为杭州和美国双机房部署，存在跨机房同步的业务需求，实现方式主要是基于业务 trigger 获取增量变更。从 2010 年开始，业务逐步尝试数据库日志解析获取增量变更进行同步，由此衍生出了大量的数据库增量订阅和消费业务。

当前的 canal 支持源端 MySQL 版本包括 5.1.x , 5.5.x , 5.6.x , 5.7.x , 8.0.x。



Canal目前没有独立的官网，可以在GitHub上下载和查看Canal文档，地址如下：https://github.com/alibaba/canal/wiki

### **Canal工作原理**



* **MySQL主备复制原理**

1. MySQL master 将数据变更写入二进制日志( binary log, 其中记录叫做二进制日志事件binary log events，可以通过 show binlog events 进行查看)
2. MySQL slave 将 master 的 binary log events 拷贝到它的中继日志(relay log)

注意：中继日志是从服务器I/O线程将主服务器的二进制日志读取过来，记录到从服务器本地文件，然后从服务器SQL线程会读取relay-log日志的内容并应用到从服务器，从而使从服务器和主服务器的数据保持一致。

1. MySQL slave 重放 relay log 中事件，将数据变更反映它自己的数据

* **canal 工作原理**

1. canal 模拟 MySQL slave 的交互协议，伪装自己为 MySQL slave ，向 MySQL master 发送dump 协议
2. MySQL master 收到 dump 请求，开始推送 binary log 给 slave (即 canal )
3. canal 解析 binary log 对象(原始为 byte 流)

注意：mysql-binlog是MySQL数据库的二进制日志，记录了所有的DDL和DML(除了数据查询语句)语句信息。一般来说开启二进制日志大概会有1%的性能损耗。

## Canal1.1.4安装

### **安装前准备-开启MySQL binlog**

对于自建 MySQL , 需要先开启 Binlog 写入功能，配置 binlog-format 为 ROW 模式，开启Mysql binlog日志步骤如下：

1. **登录mysql查看MySQL是否开启binlog日志**

|  |
| --- |
| [root@node2 ~]# mysql -u root -p123456  mysql> show variables like 'log\_%'; |

1. **开启mysql binlog日志**

在/etc/my.cnf文件中[mysqld]下写入以下内容：

|  |
| --- |
| [mysqld]  # 随机指定一个不能和其他集群中机器重名的字符串，配置 MySQL replaction 需要定#义，不要和 canal 的 slaveId 重复  server-id=123  #配置binlog日志目录，配置后会自动开启binlog日志，并写入该目录  log-bin=/var/lib/mysql/mysql-bin  # 选择 ROW 模式  binlog-format=ROW |

MySQL binlog-format有三种模式：Row、Statement 和 Mixed 。

* **Row:不记录sql语句上下文相关信息，仅保存哪条记录被修改。**

优点： binlog中可以不记录执行的sql语句的上下文相关的信息，仅需要记录那一条记录被修改成什么了。所以row level的日志内容会非常清楚的记录下每一行数据修改的细节。

缺点:所有的执行的语句当记录到日志中的时候，都将以每行记录的修改来记录，这样可能会产生大量的日志内容,比如一条update语句，修改多条记录，则binlog中每一条修改都会有记录，这样造成binlog日志量会很大，特别是当执行alter table之类的语句的时候，由于表结构修改，每条记录都发生改变，那么该表每一条记录都会记录到日志中。

* **Statement(默认)：每一条会修改数据的sql都会记录在binlog中**。

这种模式下，slave在复制的时候sql进程会解析成和原来master端执行过的相同的sql来再次执行。

优点：不需要记录每一行的变化，减少了binlog日志量，节约了IO，提高性能。

缺点：由于只记录语句，所以，在statement level下 已经发现了有不少情况会造成MySQL的复制出现问题，主要是修改数据的时候使用了某些定的函数或者功能的时候会出现。 例如：update 语句中含有uuid() ,now() 这种函数时，Statement模式就会有问题（update t1 set xx = now() where xx = xx）

* **Mixed: 混合模式**

在Mixed模式下，MySQL会根据执行的每一条具体的sql语句来区分对待记录的日志格式，也就是在Statement和Row之间选择一种。如果sql语句确实就是update或者delete等修改数据的语句，那么还是会记录所有行的变更。

1. **重启mysql 服务，重新查看binlog日志情况**

|  |
| --- |
| [root@node2 ~]# service mysqld restart  [root@node2 ~]# mysql -u root -p123456  mysql> show variables like 'log\_%'; |

### **下载安装Canal**

1. **下载Canal**

Cannal下载地址如下：<https://github.com/alibaba/canal/releases>,这里选择Canal 1.1.4版本下载。

1. **上传解压**

将下载好的Canal安装包上传到node3节点上，解压

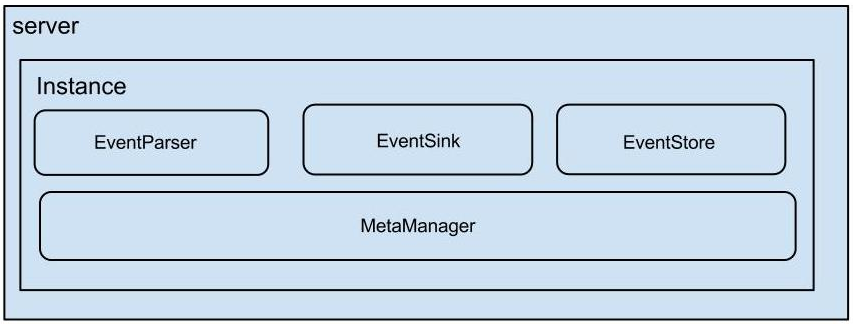
|  |
| --- |
| #首先创建目录 “/software/canal”  [root@node3 ~]# mkdir -p /software/canal  #将Canal安装包解压到创建的canal目录中  [root@node3 ~]# tar -zxvf /software/canal.deployer-1.1.4.tar.gz -C /software/canal/ |

## 使用Canal同步MySQL数据

使用Canal同步MySQL的数据可以直接使用Canal客户端API方式消费Canal同步的数据，详细api参照：<https://github.com/alibaba/canal/wiki/ClientAPI> ，也可以直接通过Canal将数据写入Kafka。下面我们使用Canal同步MySQL数据到Kafka为例，学习下Canal如何同步MySQL数据。

### **Canal架构原理**

* **Canal Server结构**

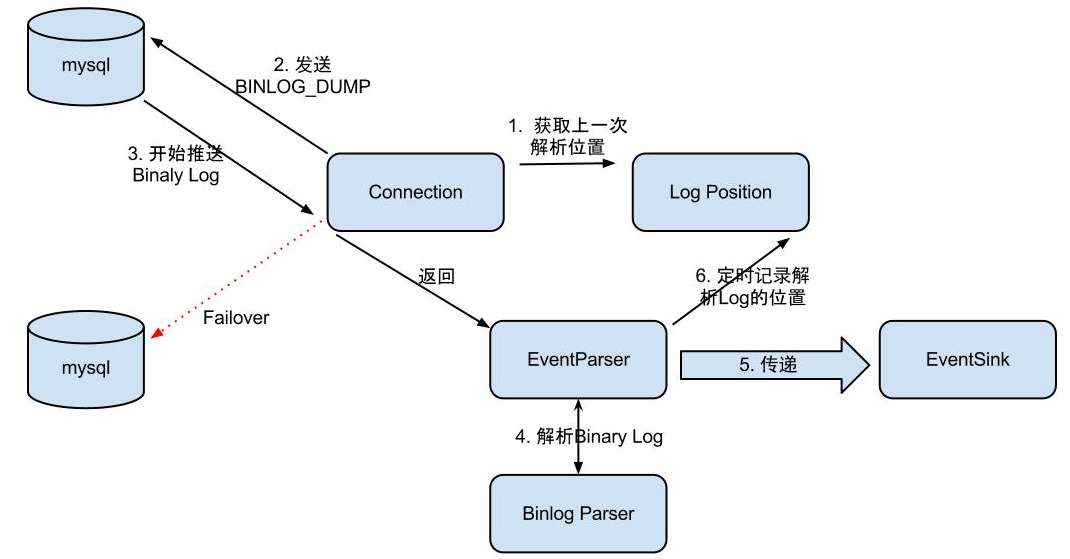


* server 代表一个 canal 运行实例，对应于一个 jvm。
* instance 对应于一个数据队列 （1个 canal server 对应 1..n 个 instance )
* instance 下的子模块

1. eventParser: 数据源接入，模拟 slave 协议和 master 进行交互，协议解析
2. eventSink: Parser 和 Store 链接器，进行数据过滤，加工，分发的工作
3. eventStore: 数据存储
4. metaManager: 增量订阅 & 消费信息管理器

* **Canal同步MySQL数据原理**

EventParser在向mysql发送dump命令之前会先从Log Position中获取上次解析成功的位置(如果是第一次启动，则获取初始指定位置或者当前数据段binlog位点)。mysql接受到dump命令后，由EventParser从mysql上pull binlog数据进行解析并传递给EventSink(传递给EventSink模块进行数据存储，是一个阻塞操作，直到存储成功)，传送成功之后更新Log Position。流程图如下：



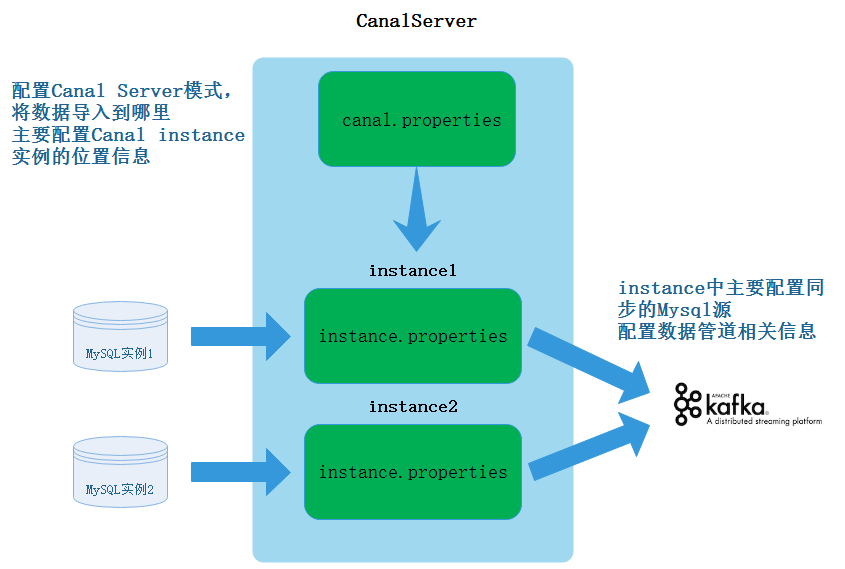
EventSink起到一个类似channel的功能，可以对数据进行过滤、分发/路由(1:n)、归并(n:1)和加工。EventSink是连接EventParser和EventStore的桥梁。

EventStore实现模式是内存模式，内存结构为环形队列，由三个指针(Put、Get和Ack)标识数据存储和读取的位置。

MetaManager是增量订阅&消费信息管理器，增量订阅和消费之间的协议包括get/ack/rollback，分别为：

1. Message getWithoutAck(int batchSize)，允许指定batchSize，一次可以获取多条，每次返回的对象为Message，包含的内容为：batch id[唯一标识]和entries[具体的数据对象]。
2. void rollback(long batchId)，顾名思义，回滚上次的get请求，重新获取数据。基于get获取的batchId进行提交，避免误操作。
3. void ack(long batchId)，顾名思义，确认已经消费成功，通知server删除数据。基于get获取的batchId进行提交，避免误操作。

* **关于同步MySQL数据配置信息**



首先Canal可以是一个集群，这里以Canal单机为例解释Canal同步MySQL数据配置文件配置原理。

首先需要在Canal中配置CanalServer 对应的canal.properties，这个文件中主要配置Canal对应的同步数据实例(Canal Instance)位置信息及数据导出的模式，例如：我们需要将某个mysql中的数据同步到Kafka中，那么就可以创建一个“数据同步实例”，导出到Kafka就是一种模式。

其次，需要配置Canal Instance 实例中的instance.properties文件，指定同步到MySQL数据源及管道信息。

### **配置步骤**

1. **配置“canal.properties”**

进入“/software/canal/conf”目录下，编辑“canal.properties”文件：

|  |
| --- |
| #canal将数据写入Kafka，可配：tcp, kafka, RocketMQ，tcp就是使用canal代码接收  canal.serverMode = kafka  #配置canal写入Kafka地址  canal.mq.servers = node1:9092,node2:9092,node3:9092 |

**关于canal.properties更多参数参照：<https://github.com/alibaba/canal/wiki/AdminGuide>**

1. **配置mysql slave的权限**

Canal的原理是模拟自己为mysql slave，所以这里一定需要做为mysql slave的相关权限 ，授权Canal连接MySQL具有作为MySQL slave的权限：

|  |
| --- |
| mysql> CREATE USER canal IDENTIFIED BY 'canal';  mysql> GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO 'canal'@'%';  mysql> FLUSH PRIVILEGES;  mysql> show grants for 'canal' ; |

1. **配置“instance.properties”**

进入“/software/canal/conf/example/”下，编辑“instance.properties”文件：

|  |
| --- |
| #canal伪装为一个mysql的salve，配置其id，不要和真正mysql server-id冲突，这里也可以不配置，会自动生成  canal.instance.mysql.slaveId=123456  #配置mysql master 节点及端口  canal.instance.master.address=node2:3306  #配置连接mysql的用户名和密码，就是前面复制权限的用户名和密码  canal.instance.dbUsername=canal  canal.instance.dbPassword=canal  #配置Canal将数据导入到Kafka topic  canal.mq.topic=canal\_topic |

配置参照：

<https://github.com/alibaba/canal/wiki/Canal-Kafka-RocketMQ-QuickStart>

关于“instance.properties”更多参数介绍如下：<https://github.com/alibaba/canal/wiki/AdminGuide#instanceproperties%E4%BB%8B%E7%BB%8D>

1. **启动Canal**

进入“/software/canal/bin”,执行“startup.sh”脚本启动Canal。

|  |
| --- |
| #启动Canal  [root@node3 ~]# cd /software/canal/bin/  [root@node3 bin]# ./startup.sh  [root@node3 bin]# jps  68675 CanalLauncher #启动成功 |

1. **启动zookeeper和Kafka，并监控Kafka中“canal\_topic”的数据**

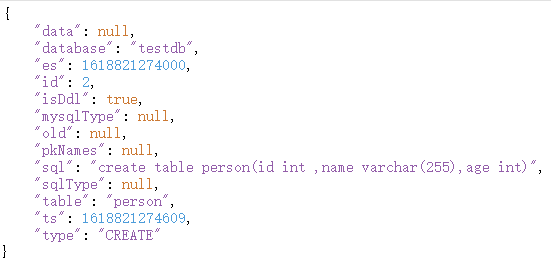
注意：“canal\_topic”不需要提前创建，默认创建就是1个分区。

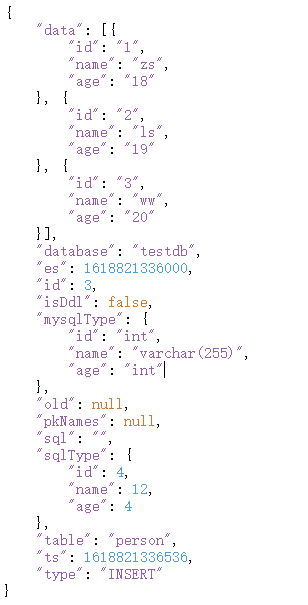
|  |
| --- |
| [root@node2 bin]# ./kafka-console-consumer.sh --bootstrap-server node1:9092,node2:9092,node3:9092 --topic canal\_topic |

1. **在MySQL中建表，插入语句**

|  |
| --- |
| mysql> create database testdb;  mysql> use testdb;  mysql> create table person(id int ,name varchar(255),age int);  mysql> insert into person values (1,"zs",18),(2,"ls",19),(3,"ww",20);  #对应的在Kafka中有对应的数据日志写入 |

以上写入Kafka中json格式如下：





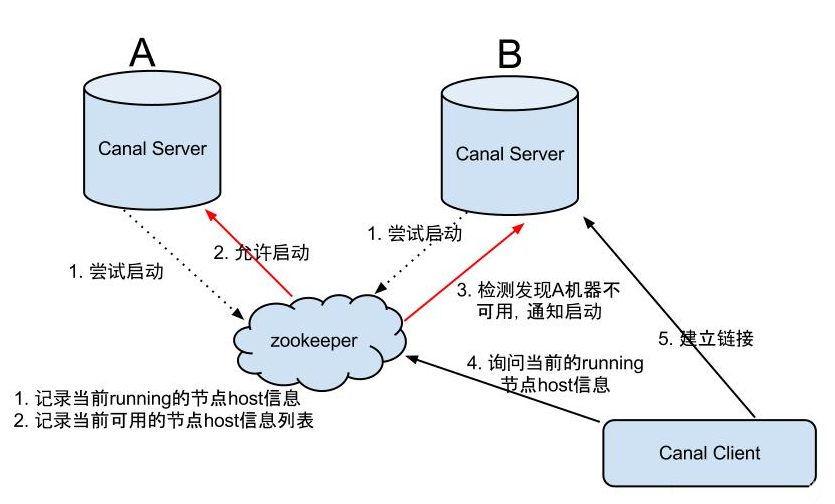
关于以上json字段解析如下：

* data：最新的数据，为JSON数组，如果是插入则表示最新插入的数据，如果是更新，则表示更新后的最新数据，如果是删除，则表示被删除的数据。
* database：数据库名称。
* es：事件时间，13位的时间戳。
* id：事件操作的序列号，1,2,3...
* isDdl：是否是DDL操作。
* mysqlType：字段类型。
* old：旧数据。
* pkNames：主键名称。
* sql：SQL语句。
* sqlType：是经过canal转换处理的，比如unsigned int会被转化为Long，unsigned long会被转换为BigDecimal。
* table：表名。
* ts：日志时间。
* type：操作类型，比如DELETE，UPDATE，INSERT。

## Canal HA原理及安装

### **Canal HA原理**

Canal一般用于实时同步数据场景，那么对于实时场景HA显得尤为重要，Canal支持HA搭建，canal的HA分为两部分，canal server和canal client分别有对应的HA实现。大数据中使用Canal同步数据一般同步到Kafka中，这里Kafka相当于是Canal Client，Kafka集群自带HA属性，所以这里我们 只关注Canal Server HA。Canal Server HA主要是为了减少对mysql dump的请求，不同server上的instance(不同server上的相同instance)要求同一时间只能有一个处于running，其他的处于standby状态(standby是instance的状态)，Canal Server HA原理如下：



Canal HA 保证步骤如下：

1. canal server要启动某个canal instance时都先向zookeeper\_进行一次尝试启动判断。
2. 创建zookeeper节点成功后，对应的canal server就启动对应的canal instance，没有创建成功的canal instance就会处于standby状态。
3. 一旦zookeeper发现canal server A创建的instance节点消失后，立即通知其他的canal server再次进行步骤1的操作，重新选出一个canal server启动instance。
4. canal client每次进行connect时，会首先向zookeeper询问当前是谁启动了canal instance，然后和其建立链接，一旦链接不可用，会重新尝试connect。

### **Canal HA 搭建**

1. **机器准备**

运行Canal的机器：node3,node4

zookeeper地址:node3:2181,node4:2181,node5:2181

mysql地址：node2:3306

1. **在node3，node4上单独部署配置Canal**

将Canal安装包上传到node3，node4，并解压到“/software/canal”目录下，修改“/software/canal/conf”下的canal.properties文件，加上zookeeper配置

|  |
| --- |
| #指定zookeeper集群地址  canal.zkServers = node3:2181,node4:2181,node5:2181  #配置spring的xml配置文件  canal.instance.global.spring.xml = classpath:spring/default-instance.xml  #canal将数据写入Kafka，可配：tcp, kafka, RocketMQ，tcp就是使用canal代码接收  canal.serverMode = kafka  #配置canal写入Kafka地址  canal.mq.servers = node1:9092,node2:9092,node3:9092 |

进入“/software/canal/conf/example”目录，修改“instance.properties”文件：

|  |
| --- |
| #另外一台机器改成123457，保证slaveId不重复即可  canal.instance.mysql.slaveId=123456  #配置mysql master 节点及端口  canal.instance.master.address=node2:3306  #配置连接mysql的用户名和密码，就是前面复制权限的用户名和密码  canal.instance.dbUsername=canal  canal.instance.dbPassword=canal  #配置Canal将数据导入到Kafka topic  canal.mq.topic=canal\_topic |

注意：两台机器上的instance目录的名字需要保证完全一致，HA模式是依赖于instance name进行管理，同时必须都选择default-instance.xml配置，此配置中才有关于zookeeper的设置信息。

1. **启动两台机器的Canal**

|  |
| --- |
| #在node3上启动Canal  [root@node3 ~]# cd /software/canal/bin  [root@node3 bin]# ./startup.sh  #在node4上启动Canal  [root@node4 ~]# cd /software/canal/bin  [root@node4 bin]# ./startup.sh |

启动完成后，可以查看zookeeper中对应的路径信息：



### **Canal HA 测试**

默认搭建好的Canal HA 后可以通过查看Zookeeper中的“/otter/canal/destinations/examples/running”来查看Active的Canal节点：



测试Canal HA 如下：

1. **向Mysql中“testdb.person”表中写入数据**

|  |
| --- |
| mysql> insert into person values (4,"s1",21),(5,"s2",22),(6,"s3",23); |

可以观察到Kafka canal\_topic中有监控到的数据如下：

|  |
| --- |
| {"data":[{"id":"4","name":"s1","age":"21"},{"id":"5","name":"s2","age":"22"},{"id":"6","name":"s3","age":"23"}],"database":"testdb","es":1618849974000,"id":2,"isDdl":false,"mysqlType":{"id":"int","name":"varchar(255)","age":"int"},"old":null,"pkNames":null,"sql":"","sqlType":{"id":4,"name":12,"age":4},"table":"person","ts":1618849975203,"type":"INSERT"} |

1. **关闭active Canal Server节点，继续向Mysql表中写入数据**

关闭node3 Canal Server:

|  |
| --- |
| [root@node3 ~]# cd /software/canal/bin  [root@node3 bin]# ./stop.sh |

查看zookeeper “/otter/canal/destinations/examples/running”路径Active的Canal节点:



继续向MySQL中“testdb.person”表中写入数据：

|  |
| --- |
| mysql> insert into person values (7,"x1",24),(8,"x2",25),(9,"x3",26); |

可以观察写入到Kafka “canal\_topic”中数据如下：

|  |
| --- |
| {"data":[{"id":"7","name":"x1","age":"24"},{"id":"8","name":"x2","age":"25"},{"id":"9","name":"x3","age":"26"}],"database":"testdb","es":1618850233000,"id":2,"isDdl":false,"mysqlType":{"id":"int","name":"varchar(255)","age":"int"},"old":null,"pkNames":null,"sql":"","sqlType":{"id":4,"name":12,"age":4},"table":"person","ts":1618850234136,"type":"INSERT"} |

经过以上测试，Canal HA 生效。

注意：经过测试Canal HA 在使用zookeeper存储binlog position时，当有一个Canal Server重新启动并切换成Active节点时，每次都会重复读取最后一条数据。使用非HA 本地存储binlog position时，没有此问题。

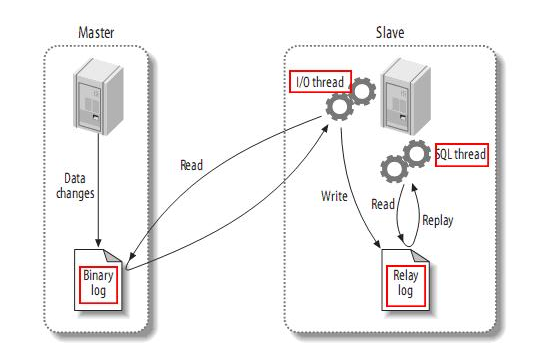
## Maxwell介绍

Maxwell是由美国Zendesk开源，使用Java编写的MySQL实时抓取工具，可以实时读取MySQL二进制日志binlog，并生成 JSON 格式的消息，作为生产者发送给 Kafka，Kinesis、RabbitMQ、Redis、Google Cloud Pub/Sub、文件或其它平台的应用程序。它设计的初衷是实时采集Mysql数据到Kafka。支持全表load数据，支持自动断点还原，支持按照列将数据发送到Kafka不同分区。

Maxwell官网：http://maxwells-daemon.io/

## Maxwell工作原理

Maxwell工作原理与Canal工作原理一样，都是把自己伪装成MySQL 的slave从库，同步binlog数据，来达到同步MySQL数据，与Canal相比，更加轻量。同样使用Maxwell也需要开启MySQL binlog日志。



## 使用Maxwell 同步MySQL数据

首先下载Maxwell,Maxwell下载地址：https://github.com/zendesk/maxwell/releases/tag/v1.28.2,现在版本1.30.0版本之上需要jdk11以上，建议下载1.30.0版本以下版本。下载完成后按照如下步骤进行配置，同步MySQL数据到Kafka，前提是MySQL需要开启Binlog日志，可以参考Canal章节设置。Maxwell不支持高可用搭建，但是支持断点还原，可以在执行失败时重新启动继续上次位置读取数据。

1. **将下载好的安装包上传到node3并解压**

|  |
| --- |
| [root@node3 ~]# cd /software/  [root@node3 software]# tar -zxvf ./maxwell-1.33.0.tar.gz |

1. **在MySQL中创建Maxwell的用户及赋权**

Maxwell同步mysql数据到Kafka中需要将读取的binlog位置文件及位置信息等数据存入MySQL，所以这里创建maxwell数据库，及给maxwell用户赋权访问其他所有数据库。

|  |
| --- |
| mysql> CREATE database maxwell;  mysql> CREATE USER 'maxwell'@'%' IDENTIFIED BY 'maxwell';  mysql> GRANT ALL ON maxwell.\* TO 'maxwell'@'%';  mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE ON \*.\* TO 'maxwell'@'%';  mysql> flush privileges; |

1. **修改配置“config.properties”文件**

进入“/software/maxwell-1.33.0”，修改“config.properties.example”为“config.properties”并配置：

|  |
| --- |
| producer=kafka  kafka.bootstrap.servers=node1:9092,node2:9092,node3:9092  kafka\_topic=maxwell\_topic  #设置根据表将binlog写入Kafka不同分区，还可指定：[database, table, primary\_key, transaction\_id, thread\_id, column]  producer\_partition\_by=table  #mysql 节点  host=node2  #连接mysql用户名和密码  user=maxwell  password=maxwell |

注意：以上参数也可以在后期启动maxwell时指定参数方式来设置。

1. **启动zookeeper及Kafka,并监控Kafka maxwell\_topic**

|  |
| --- |
| [root@node2 bin]# cd /software/kafka\_2.11-0.11/  [root@node2 bin]# ./kafka-console-consumer.sh --bootstrap-server node1:9092,node2:9092,node3:9092 --topic maxwell\_topic |

1. **启动Maxwell**

|  |
| --- |
| [root@node3 ~]# cd /software/maxwell-1.28.2/bin  [root@node3 bin]# maxwell --config ../config.properties |

注意以上启动也可以编写脚本：

|  |
| --- |
| #startMaxwell.sh 脚本内容：  /software/maxwell-1.28.2/bin/maxwell --config /software/maxwell-1.28.2/config.properties > ./log.txt 2>&1 & |

修改执行权限：

|  |
| --- |
| chmod +x ./start\_maxwell.sh |

1. **向MySQL中增删改查写入数据**

|  |
| --- |
| mysql> create database mysqldb;  mysql> use mysqldb;  mysql> create table info(id int,name varchar(255),age int);  mysql> insert into info values (10,"xx",20);  mysql> update info set age = 100 where id = 10;  mysql> delete from info where id = 10;  #对应Kafka中的消息如下：  {"database":"mysqldb","table":"info","type":"insert","ts":1619000098,"xid":3890,"commit":true,"data":{"id":100,"name":"aaa","age":10}}  {"database":"mysqldb","table":"info","type":"update","ts":1619000152,"xid":4142,"commit":true,"data":{"id":100,"name":"aaa","age":20},"old":{"age":10}}  {"database":"mysqldb","table":"info","type":"delete","ts":1619000183,"xid":4228,"commit":true,"data":{"id":100,"name":"aaa","age":20}} |

1. **测试Maxwell断点续传**

停止Maxwell,向MySQL中插入新的数据，重启Maxwell观察是否从上次消费到的binlog位置继续消费。

|  |
| --- |
| #使用kill -9 xxx 命令在node3停止Maxwell  #向MySQL继续插入数据  mysql> insert into info values (200,"bbb",20);  mysql> update info set age = 30 where id = 200;  #重新在node3启动Maxwell，可以观察到Kafka中继续上次binlog位置写入数据  {"database":"mysqldb","table":"info","type":"insert","ts":1619000378,"xid":4565,"commit":true,"data"  :{"id":200,"name":"bbb","age":20}}{"database":"mysqldb","table":"info","type":"update","ts":1619000391,"xid":4566,"commit":true,"data"  :{"id":200,"name":"bbb","age":30},"old":{"age":20}} |

## 使用Maxwell Bootstrap全量同步MySQL数据

Maxwell Bootstrap可以将MySQL中已经存在的数据批量同步到Kafka中，操作步骤如下：

1. **修改“/software/maxwell-1.28.2/config.properties”配置文件**

停止maxwell进程，在当前config.properties配置文件最后一行添加配置“client\_id”,此配置项是指定当前maxwell启动后连接mysql的实例id,名字自取，在全量同步数据时需要使用到。

|  |
| --- |
| #指定maxwell 当前连接mysql的实例id,名字自取  client\_id=maxwell\_first |

1. **在mysql中创建库“mysqldb2”,并插入数据**

|  |
| --- |
| mysql> create database mysqldb2;  mysql> use mysqldb2;  mysql> create table t1 (id int ,name varchar(255),age int );  mysql> insert into t1 values (1,"zs",18),(2,"ls",19),(3,"ww",20); |

1. **重新启动Maxwell,然后启动maxwell-bootstrap全量导数据**

maxwell-bootstrap脚本可以指定MySQL数据库及表参数，同步MySQL指定库下对应表的全量数据，同时可以指定where条件。

|  |
| --- |
| [root@node3 ~]# cd /software/maxwell-1.28.2/bin  #重启maxwell  [root@node3 bin]# maxwell --config ../config.properties  #同步mysqldb2 .t1表的全量数据  [root@node3 bin]# ./maxwell-bootstrap --database mysqldb2 --table t1 --host node2 --user maxwell --password maxwell --client\_id maxwell\_first --where "id<=2" |

注意：maxwell-bootstrap命令执行后，可以在对应的topic中查看到数据，这里对应的topic是maxwell进程对应config.properties文件中配置的topic。同时maxwell-bootstrap命令指定的client\_id 需要与maxwell进行对应的config.properties配置文件中配置的一样。--where是指定条件，只会全量导入满足条件的数据，有了where条件可以使maxwell-bootstrap进程配合maxwell实时同步进程将一张表数据无缝同步到Kafka中。

## Maxwell和Canal对比

|  |  |  |
| --- | --- | --- |
|  | **Canal** | **Maxwell** |
| **公司** | 阿里 | zendesk |
| **开发语言** | Java | Java |
| **高可用** | 支持HA | 不支持，支持断点续传 |
| **数据格式** | 格式自由 | JSON |
| **Bootstrap（刷全量数据）** | 不支持 | 支持 |
| **数据落地** | 支持客户端，支持定制/kafka | Kafka,Redis等。 |

* Canal是阿里公司使用Java开发，Maxwell是zendesk公司使用Java开发。
* Canal支持高可用HA，支持断点续传。Maxwell不支持HA,但是支持断点续传，要想支持HA需要自己实现。
* Canal由于有Client消费数据，针对binlog数据可以使用Client自定义数据格式，Maxwell支持Json数据写出到Kafka或Redis。
* Canal只能获取MySQL最新数据，Maxwell支持Bootstrap，可以支持获取MySQL中历史数据。
* Canal采用Server+client模式，Maxwell没有采用这种模式，直接将数据发送到Kafka或者Redis等。

总体来看，Maxwell相对于Canal更加轻量级。