

一、简介

- 1
- 2
- 3
1. Canal，译为水道/管道/沟渠，阿里开源的框架，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。
2. Canal的工作原理就是把自己伪装成MySQL slave，模拟MySQL slave的交互协议向MySQL Mater发送 dump协议，
3. MySQL master收到canal发送过来的dump请求，开始推送binary log给Canal，然后Canal解析binary log，再发送到存储目的地，比如MySQL，Kafka，Elastic Search等等。



二、准备MySQL

```
1 //支持版本 5.1.x , 5.5.x , 5.6.x , 5.7.x , 8.0.x
2 1.创建挂载目录
3 mkdir -vp /home/docker/mysql/{conf,data,logs}
4
5 2.挂载并启动容器
6 docker run --name mysql \
7 -e TZ=Asia/Shanghai \
8 -e MYSQL_ROOT_PASSWORD=root \
9 -v /home/docker/mysql/data:/var/lib/mysql \
10 -v /home/docker/mysql/conf:/etc/mysql/mysql.conf.d \
11 -v /home/docker/mysql/logs:/logs \
12 -p 3306:3306 \
13 -p 33060:33060 \
14 --privileged=true \
15 --restart=always \
16 -d mysql:5.7.21 \
17 --character-set-server=utf8mb4 \
18 --collation-server=utf8mb4_general_ci
19
20 3.配置mysqld.cnf
21 touch /home/docker/mysql/conf/mysqld.cnf
22 //写入配置
23 cat > /home/docker/mysql/conf/mysqld.cnf << EOF
24 [mysqld]
25 log-bin=mysql-bin
26 binlog-format=row
27 server-id=3306
28 log_timestamps=SYSTEM
29 default-time-zone='+8:00'
30 EOF
31 //重启容器生效
32 docker restart mysql;
33
34 4.创建一个用户并授权
35 docker exec -it mysql bash
36 > mysql -uroot -proot;
37 //如果执行报已经存在该用户 drop user canal@'%'; flush privileges;
38 > create user canal identified by 'canal';
39 > grant SELECT, REPLICATION SLAVE, REPLICATION CLIENT on *.* to 'canal'@'%';
40 > flush privileges;
41
42 5.检查状态
43 //是否为ON，为OFF重启容器
44 > show variables like 'log_bin';
45 //查看binlog，记录这两个参数 mysql-bin.000001 | 154
46 > show master status;
47
```

三、搭建 Canal 服务端

数据库初始化脚本

create_canal_admini....sql

3.91KB

```
1 //创建网桥，避免容器ip变化
2 docker network create --subnet=172.172.0.0/24 docker-canal
```

```
1  /**
2   * 配置说明: /home/admin/canal-server/conf/example/instance.properties
3   * example 为实例名称, 一个canal服务可以对应多个实例
4   * 在Mysql中使用 select password('123456') 生成密码密文 *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9
5   * 参数说明如下:
6   */
7  # 指定外部数据库地址
8  canal.instance.master.address=10.207.0.169:3306
9  # binlog日志名称(可省略)
10 canal.instance.master.journal.name=mysql-bin.000001
11 # mysql主库链接时起始的binlog偏移量(可省略)
12 canal.instance.master.position=154
13 # 在MySQL服务器授权的账号密码
14 canal.instance.dbUsername=root
15 canal.instance.dbPassword=root
16 # 字符集
17 canal.instance.connectionCharset = UTF-8
18 canal.instance.enableDruid=false
19 # 监听所有表, 也可以写具体的表名用逗号隔开
20 canal.instance.filter.regex=.*\\..*
21 # 数据解析表的黑名单, 多个表用逗号隔开
22 canal.instance.filter.black.regex=mysql\\.slave_.*
23 # MQ绑定的路由key名称
24 canal.mq.topic=canal_key
25
26 /**
27  * 配置说明: /home/admin/canal-server/conf/canal.properties
28  * 参数说明如下:
29  */
30 # canal配置
31 canal.ip = 绑定本机不然随机选取一个本地IP
32 canal.register.ip = zk地址
33 canal.port = 服务端口, 默认11111
34 canal.metrics.pull.port = 11112
35 # canal-admin页面管理配置
36 canal.admin.manager = 10.207.0.167:8089
37 canal.admin.port = 11110
38 canal.admin.user = admin
39 canal.admin.passwd = 6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9
40 canal.admin.register.auto = true
41 canal.admin.register.cluster = 集群名称
42 canal.admin.register.name = server名称
43 canal.destinations = example      指定实例名称, 可在/conf目录下新建其他实例目录
44 canal.serverMode = rabbitMQ      指定连接类型: tcp(代码监听), kafka, rocketMQ, rabbitMQ, pulsarMQ
45 rabbitmq.host = 10.207.0.164      MQ地址
46 rabbitmq.virtual.host = /SFJC      虚拟主机
47 rabbitmq.exchange = canal.exchange 交换机名称
48 rabbitmq.username = guest
49 rabbitmq.password = guest
50 rabbitmq.deliveryMode = 2          投递模式, 2-持久化
51
52 -----
53 1.启动服务并复制挂载文件
54 docker run --name canal-server -p 11111:11111 -d canal/canal-server
55 mkdir -vp /home/docker/canal/server/{conf/example,logs}
56 //docker exec -it canal-server bash
57 docker cp canal-server:/home/admin/canal-server/conf/canal.properties /home/docker/canal/server/conf/
58 docker cp canal-server:/home/admin/canal-server/conf/example/instance.properties /home/docker/canal/server/conf/example/
59
60 2.修改配置文件
61 【canal.properties】
62 canal.serverMode = rabbitMQ
63 rabbitmq.host = 10.207.0.164
64 rabbitmq.virtual.host = /SFJC
65 rabbitmq.exchange = canal.exchange
66 rabbitmq.username = guest
67 rabbitmq.password = guest
68 rabbitmq.deliveryMode = 2
69
70 【instance.properties】
71 canal.instance.master.address=10.207.0.169:3306
72 canal.instance.dbUsername=root
73 canal.instance.dbPassword=root
74 canal.mq.topic=canal.key
75
76 3.登录rabbitMQ并设置
77 > http://10.207.0.164:15672/
78 > 新增虚拟主机 /SFJC      新增队列 canal.queue      新增交换机 canal.exchange
79 > 绑定队列与交换机      路由键为 canal.key
80
81 4.移除容器, 重新挂载启动
82 docker stop canal-server;docker rm canal-server
```



```
83
84 docker run --name canal-server \
85 -p 11111:11111 \
86 -p 11110:11110 \
87 -p 11112:11112 \
88 -p 9100:9100 \
89 -v /home/docker/canal/server/conf/canal.properties:/home/admin/canal-server/conf/canal.properties \
90 -v /home/docker/canal/server/conf/example/instance.properties:/home/admin/canal-server/conf/example/instance.properties \
91 -v /home/docker/canal/server/logs:/home/admin/canal-server/logs/ \
92 --privileged=true \
93 --restart=always \
94 --net docker-canal --ip 172.172.0.2 \
95 -d canal/canal-server
```

```
2022-06-06 10:31:58.070 [Thread-6] INFO com.alibaba.otter.canal.deployer.CanalStarter - ## stop the canal server
2022-06-06 10:31:58.265 [Thread-6] INFO com.alibaba.otter.canal.deployer.CanalController - ## stop the canal server[172.17.0.2(172.17.0.2):11111]
2022-06-06 10:31:58.274 [Thread-6] INFO com.alibaba.otter.canal.deployer.CanalStarter - ## canal server is down.
2022-06-06 10:32:00.770 [main] INFO com.alibaba.otter.canal.deployer.Canallauncher - ## set default uncaught exception handler
2022-06-06 10:32:00.785 [main] INFO com.alibaba.otter.canal.deployer.Canallauncher - ## load canal configurations
2022-06-06 10:32:00.986 [main] INFO com.alibaba.otter.canal.deployer.CanalStarter - ## start the canal server.
2022-06-06 10:32:01.025 [main] INFO com.alibaba.otter.canal.deployer.CanalController - ## start the canal server[172.17.0.2(172.17.0.2):11111]
2022-06-06 10:32:02.411 [main] INFO com.alibaba.otter.canal.deployer.CanalStarter - ## the canal server is running now .....
[root@64936d248a5a canal]#
```

canal.log

```
2022-06-06 10:32:02.406 [main] INFO c.a.otter.canal.instance.core.AbstractCanalInstance - subscribe filter change to .*\\.
2022-06-06 10:32:02.406 [main] WARN c.a.o.canal.parse.inbound.mysql.dbsync.LogEventConvert - --> init table filter : ^.*\\..*$
2022-06-06 10:32:02.406 [main] INFO c.a.otter.canal.instance.core.AbstractCanalInstance - start successful....
2022-06-06 10:32:02.438 [destination = example , address = /10.207.0.169:3306 , EventParser] WARN c.a.o.c.p.inbound.mysql.rds.RdsBinlogEventParserProxy - prepare to find start position just last position
{"identity":{"slaveId":-1,"sourceAddress":{"address":"10.207.0.169","port":3306}},"postion":{"gtid":"","included":false,"journalName":"mysql-bin.000001","position":31037,"serverId":3306,"timestamp":1654481957000}}
2022-06-06 10:32:02.844 [destination = example , address = /10.207.0.169:3306 , EventParser] WARN c.a.o.c.p.inbound.mysql.rds.RdsBinlogEventParserProxy - ---> find start position successfully, EntryPosition[included=false,journalName=mysql-bin.000001,position=31037,serverId=3306,gtid=,timestamp=1654481957000] cost : 481ms , the next step is binlog dump
[root@64936d248a5a example]# ls
```

example.log

```
1 5.数据库修改一条数据
2 > update t_order set count = 7 where order_no = '202205270004';
```

Overview					Messages			Message rates			
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
/SFJC	canal.queue	classic	<div><div>D</div> Args</div>	<div><div>idle</div></div>	1	0	1	0.00/s	0.00/s	0.00/s	
/SFJC	queue.receipt	classic	<div><div>D</div> Args</div>	<div><div>idle</div></div>	3	0	3				

Get Message(s)

Message 1

The server reported 0 messages remaining.

Exchange

canal.exchange

Routing Key

canal.key

Redelivered

0

Properties

Payload

583 bytes
Encoding: string

{ "data": [{"id": "4", "order_no": "202205270004", "amount": "1000", "count": "7", "addr": "广州市番禺区星海学院", "phone": null, "create_date": "2022-06-01 16:52:09"}], "database": "sf_mall", "es": 1654482521000, "id": 1, "isDdl": false, "mysqlType": {"id": "int(11)", "order_no": "varchar(20)", "amount": "decimal(10,0)", "count": "int(11)", "addr": "varchar(80)", "phone": "varchar(20)", "create_date": "datetime"}}

get消息

格式化校验

```
1 {
2   "data": [{
3     "id": "4",
4     "order_no": "202205270004",
5     "amount": "1000",
6     "count": "7",
7     "addr": "广州市番禺区星海学院",
8     "phone": null,
9     "create_date": "2022-06-01 16:52:09"
10  }],
11  "database": "sf_mall",
12  "es": 1654482521000,
13  "id": 1,
14  "isDdl": false,
15  "mysqlType": {
16    "id": "int(11)",
17    "order_no": "varchar(20)",
18    "amount": "decimal(10,0)",
19    "count": "int(11)",
20    "addr": "varchar(80)",
21    "phone": "varchar(20)",
22    "create_date": "datetime"
23  },
24  "old": {
25    "count": "6"
26  },
}
```

新值

旧值

```
1 6.再修改一条数据
2 > update t_order set count = 5 where order_no = '202205270004';
```

Overview					Messages			Message rates			
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
/SFJC	canal.queue	classic	<div><div>D</div> Args</div>	<div><div>idle</div></div>	2	0	2	0.00/s	0.00/s	0.00/s	
/SFJC	queue.receipt	classic	<div><div>D</div> Args</div>	<div><div>idle</div></div>	3	0	3				

四、客户端连接

```
1 1.依赖
2 <!-- canal -->
3 <dependency>
```

```
4         <groupId>com.alibaba.otter</groupId>
5         <artifactId>canal.client</artifactId>
6         <version>1.1.4</version>
7     </dependency>
8
9 2. 工具类
10 import com.alibaba.otter.canal.client.CanalConnector;
11 import com.alibaba.otter.canal.client.CanalConnectors;
12 import com.alibaba.otter.canal.protocol.CanalEntry;
13 import com.alibaba.otter.canal.protocol.Message;
14 import org.springframework.beans.factory.InitializingBean;
15 import org.springframework.stereotype.Component;
16
17 import java.net.InetSocketAddress;
18 import java.util.List;
19
20 import static com.alibaba.otter.canal.protocol.CanalEntry.*;
21
22 /**
23  * canal 拦截 mysql binlog
24  */
25 @Component
26 public class CannalClient implements InitializingBean {
27
28     private final static int BATCH_SIZE = 1000;
29
30     @Override
31     public void afterPropertiesSet() throws Exception {
32         // 创建链接
33         CanalConnector connector = CanalConnectors.newSingleConnector(new InetSocketAddress("10.207.0.167", 11111), "example", "canal", "canal");
34         try {
35             //打开连接
36             connector.connect();
37             //订阅数据库表,全部表
38             connector.subscribe(".*\\..*");
39             //回滚到未进行ack的地方，下次fetch的时候，可以从最后一个没有ack的地方开始拿
40             connector.rollback();
41             while (true) {
42                 // 获取指定数量的数据
43                 Message message = connector.getWithoutAck(BATCH_SIZE);
44
45                 //获取批量ID
46                 long batchId = message.getId();
47                 //获取批量的数量
48                 int size = message.getEntries().size();
49                 //如果没有数据
50                 if (batchId == -1 || size == 0) {
51                     try {
52                         //线程休眠2秒
53                         Thread.sleep(2000);
54                     } catch (InterruptedException e) {
55                         e.printStackTrace();
56                     }
57                 } else {
58                     //如果有数据,处理数据
59                     printEntry(message.getEntries());
60                 }
61                 //进行 batch id 的确认。确认之后，小于等于此 batchId 的 Message 都会被确认。
62                 connector.ack(batchId);
63             }
64         } catch (Exception e) {
65             e.printStackTrace();
66         } finally {
67             connector.disconnect();
68         }
69     }
70
71     /**
72      * 打印canal server解析binlog获得的实体类信息
73      */
74     private static void printEntry(List<Entry> entrys) {
75         for (Entry entry : entrys) {
76             if (entry.getEntryType() == EntryType.TRANSACTIONBEGIN || entry.getEntryType() == EntryType.TRANSACTIONEND) {
77                 //开启/关闭事务的实体类型，跳过
78                 continue;
79             }
80             //RowChange对象，包含了一行数据变化的所有特征
81             //比如isDdl 是否是ddl变更操作 sql 具体的ddl sql beforeColumns afterColumns 变更前后的数据字段等等
82             RowChange rowChange;
83             try {
84                 rowChange = RowChange.parseFrom(entry.getStoreValue());
85             } catch (Exception e) {
86                 throw new RuntimeException("ERROR ## parser of eromanga-event has an error , data:" + entry.toString(), e);
87             }
88             //获取操作类型：insert/update/delete类型
89             EventType eventType = rowChange.getEventType();
```



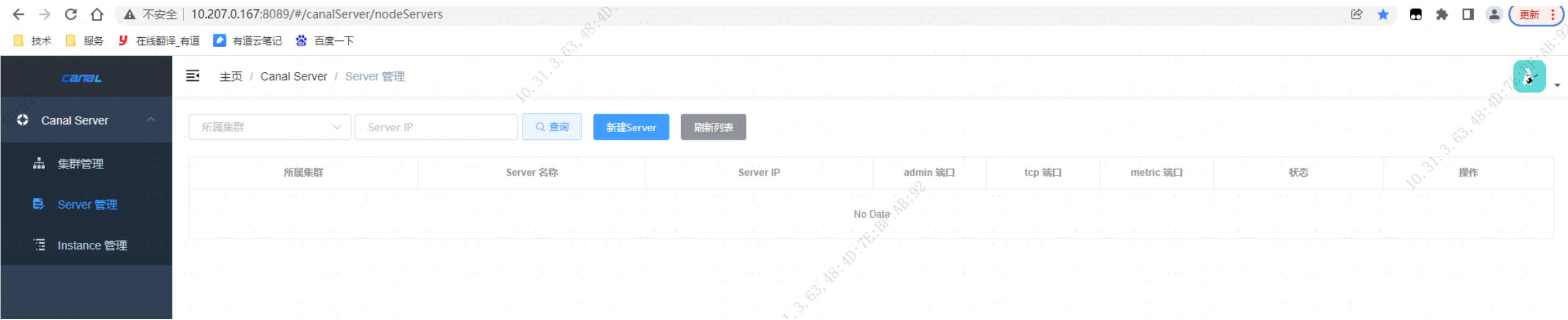
```
90 //打印Header信息
91 System.out.println();
92 System.out.println(String.format("> binlog及偏移量[%s:%s] , 库/表[%s,%s] , 操作类型[%s]",
93     entry.getHeader().getLogfileName(), entry.getHeader().getLogfileOffset(),
94     entry.getHeader().getSchemaName(), entry.getHeader().getTableName(),
95     eventType));
96 //判断是否是DDL语句
97 if (rowChage.getIsDdl()) {
98     System.out.println("> DDL sql: " + rowChage.getSql());
99 }
100 //获取RowChange对象里的每一行数据，打印出来
101 for (CanalEntry.RowData rowData : rowChage.getRowDatasList()) {
102     //删除语句
103     if (eventType == EventType.DELETE) {
104         printColumn(rowData.getBeforeColumnsList());
105     } else if (eventType == EventType.INSERT) {
106         printColumn(rowData.getAfterColumnsList());
107     } else {
108         //变更前的数据
109         System.out.println(">> 变更前数据");
110         printColumn(rowData.getBeforeColumnsList());
111         //变更后的数据
112         System.out.println(">> 变更后数据");
113         printColumn(rowData.getAfterColumnsList());
114     }
115 }
116 }
117 }
118
119 private static void printColumn(List<Column> columns) {
120     for (Column column : columns) {
121         System.out.println(column.getName() + " : " + column.getValue() + "    update=" + column.getUpdated());
122     }
123 }
124 }
125
126 3.测试
127 //将 canal.properties 中的 canal.serverMode = rabbitMQ 改为 canal.serverMode = tcp 后重启服务，不然代码层面监听会报异常
128 > update t_order set count = 4 where order_no = '202205270004';
129
```

```
> binlog及偏移量[mysql-bin.000001:32078] , 库/表[sf_mall,t_order] , 操作类型[UPDATE]
>> 变更前数据
id : 4      update=false
order_no : 202205270004      update=false
amount : 1000      update=false
count : 5      update=false
addr : 广州市番禺区星海学院      update=false
phone :      update=false
create_date : 2022-06-01 16:52:09      update=false
>> 变更后数据
id : 4      update=false
order_no : 202205270004      update=false
amount : 1000      update=false
count : 4      update=true
addr : 广州市番禺区星海学院      update=false
phone :      update=false
create_date : 2022-06-01 16:52:09      update=false
```

五、Admin管理界面

```
1 //登录密码不能少于6位，故改为123456    如果指定了数据库，以数据库里的canal_user表为准
2 //在Mysql中使用 select password('123456') 生成密码密文 *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9
3 1.启动服务并复制挂载文件
4 docker run --name=canal-admin -p 8089:8089 -d canal/canal-admin
5 mkdir -vp /home/docker/canal/admin/{conf,logs}
6 //docker exec -it canal-admin bash
7 docker cp canal-admin:/home/admin/canal-admin/conf/application.yml /home/docker/canal/admin/conf/
8
9 2.配置
10 【application.yml】
11 server:
12     port: 8089 # docker启动需要单独指定，否则不生效
13 spring:
14     jackson:
15         date-format: yyyy-MM-dd HH:mm:ss
16         time-zone: GMT+8
17
18 spring.datasource:
19     address: 10.207.0.169:3306
20     database: canal_manager
21     username: root
22     password: root
23     driver-class-name: com.mysql.jdbc.Driver
24
25     url: jdbc:mysql://${spring.datasource.address}/${spring.datasource.database}?useUnicode=true&characterEncoding=UTF-8&useSSL=false
```

```
25     hikari:
26         maximum-pool-size: 30
27         minimum-idle: 1
28
29     canal:
30         adminUser: admin
31         adminPasswd: 123456
32
33 3. 移除旧容器
34 docker stop canal-admin ; docker rm canal-admin
35
36 4. 重新挂载启动
37 docker run --name=canal-admin \
38 -p 8089:8089 \
39 -e server.port=8089 \
40 -v /home/docker/canal/admin/conf/application.yml:/home/admin/canal-admin/conf/application.yml \
41 -v /home/docker/canal/admin/logs:/home/admin/canal-admin/logs/ \
42 --privileged=true \
43 --restart=always \
44 --net docker-canal --ip 172.172.0.4 \
45 -d canal/canal-admin
46
47 5. 访问admin
48 http://10.207.0.167:8089 // admin/123456 登录
```

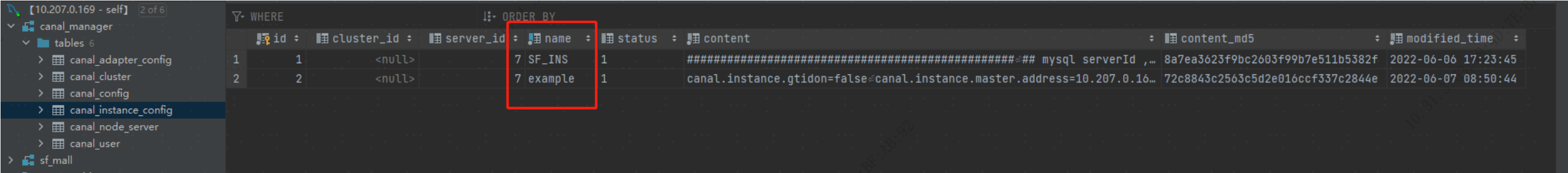


```
1 6. server关联admin
2 > vi /home/docker/canal/server/conf/canal.properties
3 canal.admin.register.auto = true
4 canal.admin.register.cluster =
5 canal.admin.register.name = 167-canal-server
6
7 > docker restart canal-server
```

server自动注入进来



添加实例，一个默认，一个新增

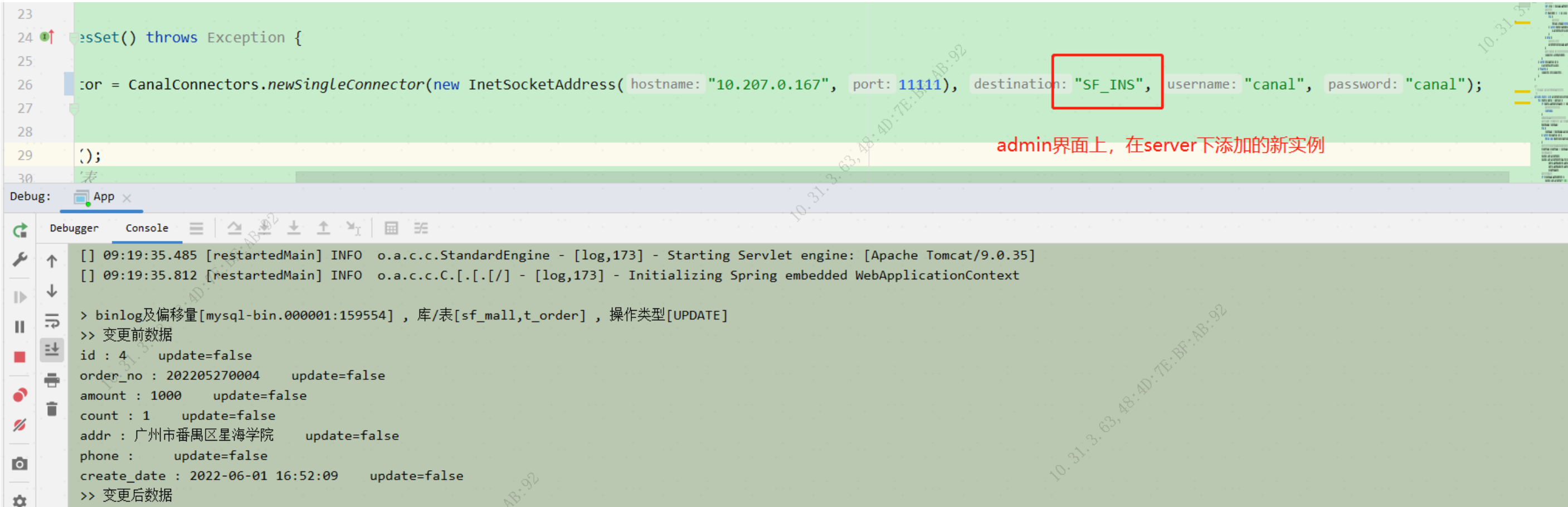


关闭example实例




```
【zero-test】
=====
[] 08:47:29.995 [restartedMain] INFO com.sf.App - [logStarting,55] - Starting App on DESKTOP-8DJKML with PID 7140 (D:\git\docker_springboot\target\classes started by FORMSSI in D:\git\docker_springboot\target\classes)
[] 08:47:30.011 [restartedMain] DEBUG com.sf.App - [logStarting,56] - Running with Spring Boot v2.3.0.RELEASE, Spring v5.2.6.RELEASE
[] 08:47:30.011 [restartedMain] INFO com.sf.App - [logStartupProfileInfo,651] - No active profile set, falling back to default profiles: default
[] 08:47:33.631 [restartedMain] INFO o.a.c.h.Http11NioProtocol - [log,173] - Initializing ProtocolHandler ["http-nio-8088"]
[] 08:47:33.632 [restartedMain] INFO o.a.c.c.StandardService - [log,173] - Starting service [Tomcat]
[] 08:47:33.632 [restartedMain] INFO o.a.c.c.StandardEngine - [log,173] - Starting Servlet engine: [Apache Tomcat/9.0.35]
[] 08:47:33.976 [restartedMain] INFO o.a.c.c.C.[.][/] - [log,173] - Initializing Spring embedded WebApplicationContext
com.alibaba.otter.canal.protocol.exception.CanalClientException: failed to subscribe with reason: something goes wrong with channel:[id: 0x73831cf2, /10.31.3.63:50427 => /172.172.0.2:11111], exception: java.lang.IllegalArgumentException: Invalid channel name: SF_INS
    at com.alibaba.otter.canal.client.impl.SimpleCanalConnector.subscribe(SimpleCanalConnector.java:249)
    at com.sf.config.CannalClient.afterPropertiesSet(CannalClient.java:31)
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.invokeInitMethods(AbstractAutowireCapableBeanFactory.java:1855)
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1792)
```

监听新增的SF_INS实例



六、多server容器

```
1 1.新增server-2容器
2 docker run --name canal-server-2 -p 10011:11111 --net docker-canal --ip 172.172.0.3 -d canal/canal-server
3 //docker stop canal-server-2 ; docker rm canal-server-2
4 //docker exec -it canal-server-2 bash
5 //cd /home/admin/canal-server/conf
6 //docker restart canal-server-2
```

```
canal.metrics.pull.port = 11112
# canal instance user/passwd
# canal.user = canal
# canal.passwd = E3619321C1A937C46A0D8BD1DAC39F93B27D4458

# canal admin config
canal.admin.manager = 10.207.0.167:8089
canal.admin.port = 11110
canal.admin.user = admin
canal.admin.passwd = 6BB4837EB74329105EE45680DA7DC67ED2CA2AD9
# admin auto register
canal.admin.register.auto = true
canal.admin.register.cluster =
canal.admin.register.name = 167-canal-server-2

canal.zkServers =
# flush data to zk
```



```
1 2.登录rabbitMQ并设置
2 > 在虚拟主机/SFJC下 新增队列 canal-2.queue 新增交换机 canal-2.exchange
3 > 绑定队列与交换机 路由键为 canal-2.key
4
5 3.在admin管理界面配置server及实例，添加MQ信息
```

```
162 rocketmq.tag =
163
164 #####
165 ##### RabbitMQ #####
166 #####
167 # tcp, kafka, rocketMQ, rabbitMQ
168 canal.serverMode = rabbitMQ
169 rabbitmq.host = /SFJC
170 rabbitmq.virtual.host = 10.207.0.164
171 rabbitmq.exchange = canal-2.exchange
172 rabbitmq.username = guest
173 rabbitmq.password = guest
174 rabbitmq.deliveryMode = 2
```


example-2

167-canal-server-2

保存

载入模板

返回

```
1 canal.instance.gtidon=false
2 canal.instance.master.address=10.207.0.169:3306
3 canal.instance.master.journal.name=
4 canal.instance.master.position=
5 canal.instance.master.timestamp=
6 canal.instance.master.gtid=
7 canal.instance.rds.accesskey=
8 canal.instance.rds.secretkey=
9 canal.instance.rds.instanceId=
10 canal.instance.tsdb.enable=true
11 canal.instance.dbUsername=root
12 canal.instance.dbPassword=root
13 canal.instance.connectionCharset = UTF-8
14 canal.instance.enableDruid=false
15 canal.instance.filter.regex=.*\\.*
16 canal.instance.filter.black.regex=
17 canal.mq.topic=canal-2.key
18 canal.mq.partition=0
```

主页 / Canal Server / Instance 管理

Instance 名称

167-canal-server-2

查询

新建 Instance

刷新列表

Instance 名称	所属集群	所属主机	状态	修改时间	操作
example-2	-	167-canal-server-2	启动	2022-06-07 10:21:09	操作

Total 1

20/page

<1>

Go to

1

admin配置server似乎没生效，还是tcp模式

Overview					Messages			Message rates		
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/SFJC	canal-2.queue	classic	D Args	idle	0	0	0			
/SFJC	canal.queue	classic	D Args	idle	5	0	5	0.00/s	0.00/s	0.00/s
/SFJC	queue.receivept	classic	D Args	idle	3	0	3			

```
= CanalConnectors.newSingleConnector(new InetSocketAddress( hostname: "10.207.0.167", port: 10011), destination: "example-2", username: "canal", password: "canal");
```

Debugger Console

> binlog及偏移量[mysql-bin.000001:246123] , 库/表[sf_mall,t_order] , 操作类型[UPDATE]
>> 变更前数据
id : 4 update=false
order_no : 202205270004 update=false
amount : 1000 update=false
count : 4 update=false
addr : 广州市番禺区星海学院 update=false
phone : update=false
create_date : 2022-06-01 16:52:09 update=false
>> 变更后数据

启动暂停ok

所属集群	Server 名称	Server IP	admin 端口	tcp 端口	metric 端口	状态	操作
-	167-canal-server	172.172.0.2	11110	11111	11112	启动	操作
-	167-canal-add	172.172.0.6	11110	11111	11112	断开	操作
-	167-canal-server-2	172.172.0.3	11110	10011	11112	停止	操作

```
[ ] 10:37:10.999 [restartedMain] INFO o.a.c.c.StandardEngine - [log,173] - Starting Servlet engine: [Apache Tomcat/9.0.35]
[ ] 10:37:11.234 [restartedMain] INFO o.a.c.c.C.[.[./] - [log,173] - Initializing Spring embedded WebApplicationContext
com.alibaba.otter.canal.protocol.exception.CanalClientException Create breakpoint : java.net.ConnectException: Connection refused: connect
at com.alibaba.otter.canal.client.impl.SimpleCanalConnector.doConnect(SimpleCanalConnector.java:198)
at com.alibaba.otter.canal.client.impl.SimpleCanalConnector.connect(SimpleCanalConnector.java:115)
at com.sf.config.CannalClient.afterPropertiesSet(CannalClient.java:29)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.invokeInitMethods(AbstractAutowireCapableBeanFactory.java:1855)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1792)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:595)
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:517)
at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:323)
```

既然admin上修改canal.properties没用，那就去server容器改吧，然后在admin添加一个实例 SF_INS_2

```
#####
##### RabbitMQ #####
#####
rabbitmq.host = 10.207.0.164
rabbitmq.virtual.host = /SFJC
rabbitmq.exchange = canal-2.exchange
rabbitmq.username = guest
rabbitmq.password = guest
rabbitmq.deliveryMode = 2
```


SF_INS_2/instance.properties167-canal-server-2修改重置

1

canal.instance.gtidon=false

2

canal.instance.master.address=10.207.0.169:3306

3

canal.instance.master.journal.name=

4

canal.instance.master.position=

5

canal.instance.master.timestamp=

6

canal.instance.master.gtid=

7

canal.instance.rds.accesskey=

8

canal.instance.rds.secretkey=

9

canal.instance.rds.instanceId=

10

canal.instance.tsdb.enable=true

11

canal.instance.dbUsername=root

12

canal.instance.dbPassword=root

13

canal.instance.connectionCharset = UTF-8

14

canal.instance.enableDruid=false

15

canal.instance.filter.regex=.*\\..*

16

canal.instance.filter.black.regex=

17

canal.mq.topic=canal-2.key

18

canal.mq.partition=0

Overview					Messages			Message rates		
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/SFJC	canal-2.queue	classic	D Args	idle	3	0	3	0.00/s		
/SFJC	canal.queue	classic	D Args	idle	11	0	11	0.00/s	0.00/s	0.00/s
/SFJC	queue.receipt	classic	D Args	idle	3	0	3			

SF_INS_2.log刷新返回

2022-06-07 11:07:03.627 [canal-instance-scan-0] INFO c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer - Properties resource not found: class path resource [SF_INS_2/instance.properties] cannot be opened because it does not exist

2022-06-07 11:07:03.669 [canal-instance-scan-0] INFO c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer - Properties resource not found: class path resource [SF_INS_2/instance.properties] cannot be opened because it does not exist

2022-06-07 11:07:03.759 [canal-instance-scan-0] INFO c.a.otter.canal.instance.spring.CanalInstanceWithSpring - start CannalInstance for 1-SF_INS_2

2022-06-07 11:07:03.760 [canal-instance-scan-0] WARN c.a.o.canal.parse.inbound.mysql.dbsync.LogEventConvert - --> init table filter : ^.*\\..*\$

2022-06-07 11:07:03.760 [canal-instance-scan-0] WARN c.a.o.canal.parse.inbound.mysql.dbsync.LogEventConvert - --> init table black filter :

2022-06-07 11:07:03.761 [canal-instance-scan-0] INFO c.a.otter.canal.instance.core.AbstractCanalInstance - start successful....

2022-06-07 11:07:03.777 [destination = SF_INS_2 , address = /10.207.0.169:3306 , EventParser] WARN c.a.o.c.p.inbound.mysql.rds.RdsBinlogEventParserProxy - ----> begin to find start position, it will be long time for reset or first position

2022-06-07 11:07:03.777 [destination = SF_INS_2 , address = /10.207.0.169:3306 , EventParser] WARN c.a.o.c.p.inbound.mysql.rds.RdsBinlogEventParserProxy - prepare to find start position just show master status

2022-06-07 11:07:04.102 [destination = SF_INS_2 , address = /10.207.0.169:3306 , EventParser] WARN c.a.o.c.p.inbound.mysql.rds.RdsBinlogEventParserProxy - ----> find start position successfully, EntryPosition[included=false,journalName=mysql-bin.000001,position=280215,serverId=3306,gtid=,timestamp=1654571219000] cost : 325ms , the next step is binlog dump

七、Canal目录结构

```
└─ canal
  │   └─ admin
  │       │   └─ conf
  │       │       └─ application.yml
  │       │       └─ logs
  │       │           └─ admin
  │       │               └─ 2022-06-06
  │       │                   └─ admin-2022-06-06-0.log.gz
  │       │                   └─ admin.log
  │   └─ server
  │       │   └─ conf
  │       │       └─ canal.properties
  │       │       └─ example
  │       │       └─ instance.properties
  │       │       └─ logs
  │       │           └─ canal
  │       │               │   └─ 2022-06-06
  │       │               │       └─ canal-2022-06-06-0.log.gz
  │       │               │       └─ canal.log
  │       │               │       └─ canal_stdout.log
  │       │               │       └─ rocketmq_client.log
  │       │               └─ example
  │       │                   │   └─ 2022-06-06
  │       │                   │       └─ example-2022-06-06-0.log.gz
  │       │                   │       └─ meta-2022-06-06-0.log.gz
  │       │                   └─ example.log
  │       │                   └─ meta.log
  │   └─ SF_INS
  │       │   └─ 2022-06-06
  │       │       └─ SF_INS-2022-06-06-0.log.gz
  │       │       └─ meta.log
  │       └─ SF_INS.log
```

【最佳实践】

1. Server 可以单独启动，并配置实例
2. Server 配置 admin 相关参数，启动后自动注入到 admin 管理界面
3. Admin 新增 server 以及对自动注入的 server 修改配置都不会生效，新增实例配置没问题
4. Admin 查询 server 及实例状态、启动暂停都没问题
5. 如果对某个DB实例既要代码监听又要MQ监听，可以启动两个 server 在各自配置里指定 tcp 或 rabbitMQ 模式

【异常】：

问题：canal容器启动失败，配置文件属于挂载外部文件，容器内没有权限读取

原因：centos7 安全子系统 Selinux 禁止了一些安全权限，导致进行挂载目录时出现这个错误

解决：可以在 docker run 命令中加入 --privileged=true 设置，给容器加上特定权限

问题：启动canal，报canal-admin密码错误

```
2022-05-31 15:19:06.842 [main] INFO com.alibaba.otter.canal.deployer.CanalLauncher - ## load canal configurations
2022-05-31 15:19:07.323 [main] ERROR com.alibaba.otter.canal.deployer.CanalLauncher - ## Something goes wrong when starting up the canal Server:
com.alibaba.otter.canal.common.CanalException: load manager config failed.
Caused by: com.alibaba.otter.canal.common.CanalException: requestGet for canal config error: auth :admin is failed
```

原因：配置需要用 select password('xx') 生成的密文，123456 -> 6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9

问题： com.alibaba.otter.canal.protocol.exception.CanalClientException: java.net.ConnectException: Connection refused: connect

原因： MQ的连接需要将 canal.properties 中的 canal.serverMode = rabbitMQ, 如需代码监听 改为 canal.serverMode = tcp

问题： Caused by: java.net.BindException: Cannot assign requested address

原因： canal.ip = 10.207.0.167 绑定本机真实ip, 由于docker默认网络类型为bridge, 不特殊指定的话, 容器内部分配的ip为172.17.0.x

解决： docker创建网桥, 启动时固定容器ip, 参考docker