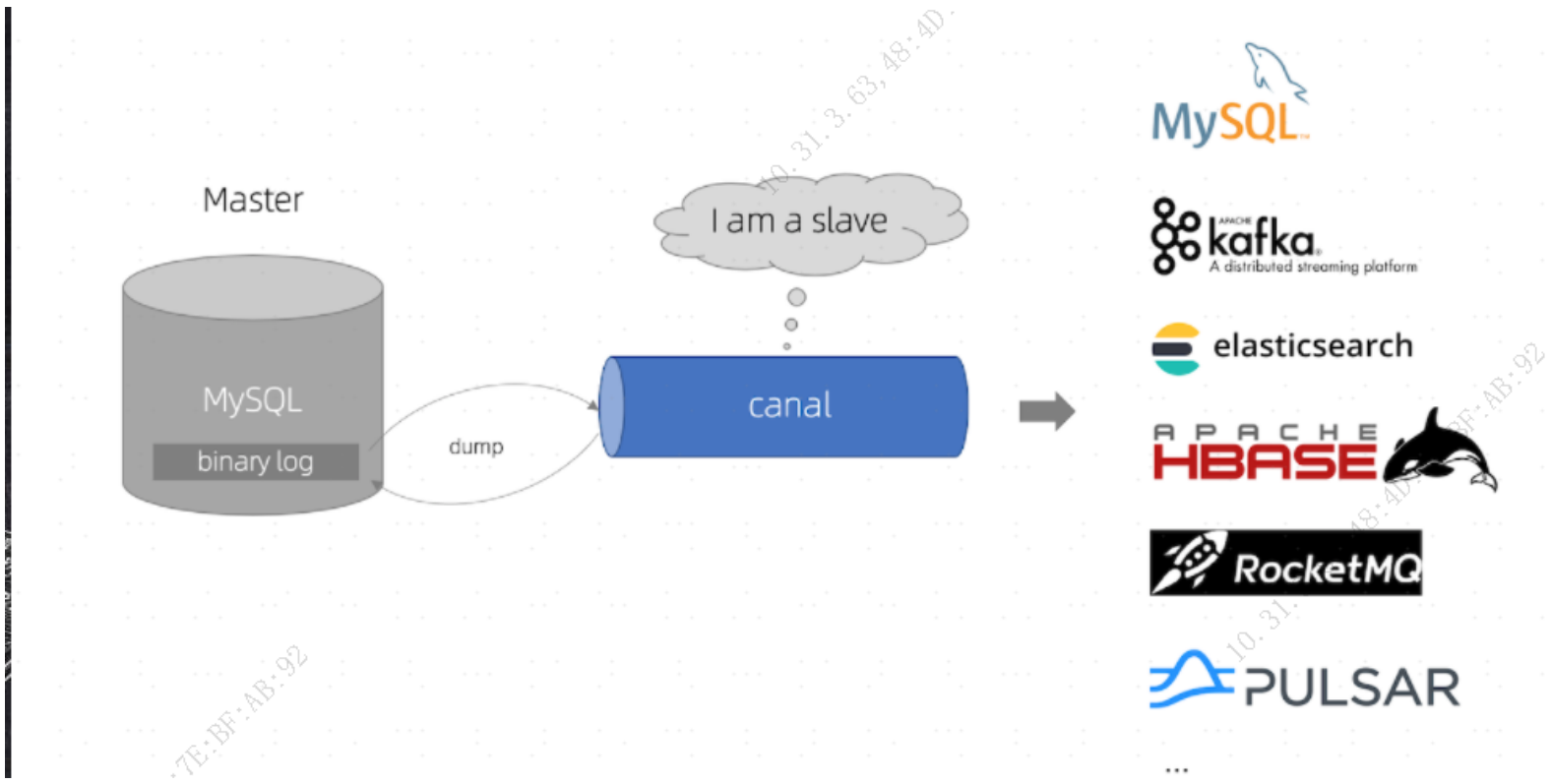
Canal

一、简介

```
1.canal，译意为水道/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，
  MySQL mater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，等等。
```



二、准备MySQL

```
//支持版本 5.1.x , 5.5.x , 5.6.x , 5.7.x , 8.0.x
1.创建挂载目录
mkdir -vp /home/docker/mysql/{conf,data,logs}

2.挂载并启动容器
docker run --name mysql \
-e TZ=Asia/Shanghai \
-e MYSQL_ROOT_PASSWORD=root \
-v /home/docker/mysql/data:/var/lib/mysql \
-v /home/docker/mysql/conf:/etc/mysql/mysql.conf.d \
-v /home/docker/mysql/logs:/logs \
-p 3306:3306 \
-p 33060:33060 \
--privileged=true \
--restart=always \
-d mysql:5.7.21 \
--character-set-server=utf8mb4 \
--collation-server=utf8mb4_general_ci

3.配置mysqld.cnf
touch /home/docker/mysql/conf/mysqld.cnf
//写入配置
cat > /home/docker/mysql/conf/mysqld.cnf << EOF
[mysqld]
log-bin=mysql-bin
binlog-format=row
server_id=3306
log_timestamps=SYSTEM
default-time-zone='+8:00'
EOF
```

```
//重启容器生效
docker restart mysql;

4.创建一个用户并授权
docker exec -it mysql bash
> mysql -uroot -proot;
//如果执行报已经存在该用户 drop user canal@'%'; flush privileges;
> create user canal identified by 'canal';
> grant SELECT, REPLICATION SLAVE, REPLICATION CLIENT on *.* to 'canal'@'%';
> flush privileges;

5.检查状态
//是否为ON，为OFF重启容器
> show variables like 'log_bin';
//查看binlog
//记录这两个参数 mysql-bin.000001 | 154
> show master status;

```

三、搭建 Canal 服务端

```
1.创建挂载目录
mkdir -p /home/docker/canal/{conf,logs}
touch /home/docker/canal/conf/{instance.properties,canal.properties}

2.写入配置
【instance.properties】
cat > /home/docker/canal/conf/instance.properties << EOF
# 数据库地址
canal.instance.master.address=10.207.0.169:3306
# binlog日志名称
canal.instance.master.journal.name=mysql-bin.000001
# mysql主库链接时起始的binlog偏移量
canal.instance.master.position=154
# mysql主库链接时起始的binlog的时间戳
canal.instance.master.timestamp=
canal.instance.master.gtid=
# 在MySQL服务器授权的账号密码
canal.instance.dbUsername=canal
canal.instance.dbPassword=canal
# 字符集
canal.instance.connectionCharset = UTF-8
# enable druid Decrypt database password
canal.instance.enableDruid=false
# 监听所有表，也可以写具体的表名用逗号隔开
canal.instance.filter.regex=.*\\..*
# 数据解析表的黑名单，多个表用逗号隔开
canal.instance.filter.black.regex=
EOF

【canal.properties】
//select password('123456') 生成密码密文 *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9
cat > /home/docker/canal/conf/canal.properties << EOF
# canal admin config
canal.register.ip = 10.207.0.169
canal.admin.manager = 10.207.0.169:8090
canal.admin.port = 11110

canal.admin.user = admin
```

```
canal.admin.passwd = 6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9
canal.admin.register.auto = true
canal.admin.register.cluster =
EOF

3.挂载并启动容器
// docker stop canal;docker rm canal
docker run --name canal -p 10010:11111 -d \
-v /home/docker/canal/conf/instance.properties:/home/admin/canal-server/conf/example/instance.properties \
-v /home/docker/canal/logs/:/home/admin/canal-server/logs/ \
--privileged=true \
canal/canal-server:v1.1.5

//没生效，暂时注释掉
//-v /home/docker/canal/conf/canal.properties:/home/admin/canal-server/conf/canal.properties \

```

```
[root@node2 canal]# docker logs canal
DOCKER_DEPLOY_TYPE=VM
==> INIT /alidata/init/02init-sshd.sh
==> EXIT CODE: 0
==> INIT /alidata/init/fix-hosts.py
==> EXIT CODE: 0
==> INIT DEFAULT
Generating SSH1 RSA host key: [  OK  ]
Starting sshd: [  OK  ]
Starting crond: [  OK  ]
==> INIT DONE
==> RUN /home/admin/app.sh
==> START ...
start canal ...
start canal successful
==> START SUCCESSFUL ...
[root@node2 canal]#
```

四、客户端连接

```
1.依赖
<!-- canal -->
<dependency>
    <groupId>com.alibaba.otter</groupId>
    <artifactId>canal.client</artifactId>
    <version>1.1.4</version>
</dependency>

2.工具类
import com.alibaba.otter.canal.client.CanalConnector;
import com.alibaba.otter.canal.client.CanalConnectors;
import com.alibaba.otter.canal.protocol.CanalEntry;
import com.alibaba.otter.canal.protocol.Message;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.stereotype.Component;

import java.net.InetSocketAddress;
import java.util.List;

import static com.alibaba.otter.canal.protocol.CanalEntry.*;

/**
 * canal 拦截 mysql binlog
 */
@Component
public class CannalClient implements InitializingBean {

```

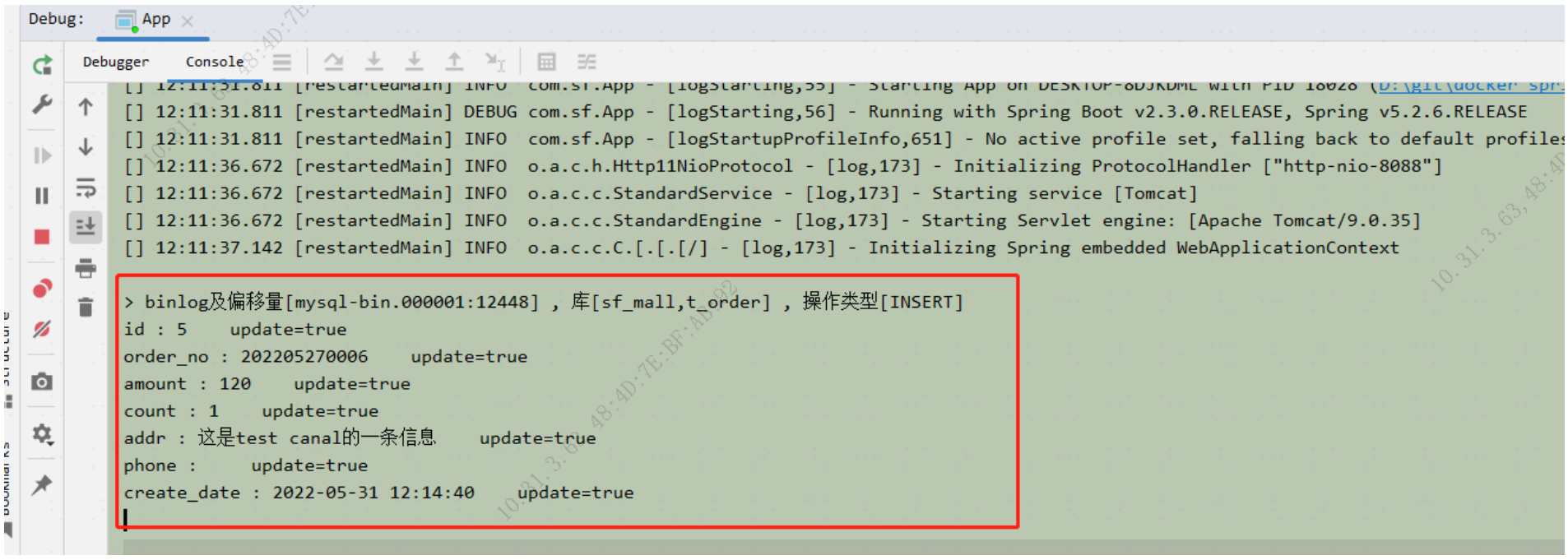
```
    private final static int BATCH_SIZE = 1000;

    @Override
    public void afterPropertiesSet() throws Exception {
        // 创建链接
        CanalConnector connector = CanalConnectors.newSingleConnector(new InetSocketAddress("10.207.0.169", 10010), "e
"canal");
        try {
            //打开连接
            connector.connect();
            //订阅数据库表,全部表
            connector.subscribe(".*\\..*");
            //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
            connector.rollback();
            while (true) {
                // 获取指定数量的数据
                Message message = connector.getWithoutAck(BATCH_SIZE);

                //获取批量ID
                long batchId = message.getId();
                //获取批量的数量
                int size = message.getEntries().size();
                //如果没有数据
                if (batchId == -1 || size == 0) {
                    try {
                        //线程休眠2秒
                        Thread.sleep(2000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                } else {
                    //如果有数据,处理数据
                    printEntry(message.getEntries());
                }
                //进行 batch id 的确认。确认之后，小于等于此 batchId 的 Message 都会被确认。
                connector.ack(batchId);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            connector.disconnect();
        }
    }

    /**
     * 打印canal server解析binlog获得的实体类信息
     */
    private static void printEntry(List<Entry> entrys) {
        for (Entry entry : entrys) {
            if (entry.getEntryType() == EntryType.TRANSACTIONBEGIN || entry.getEntryType() == EntryType.TRANSACTIONEND
                //开启/关闭事务的实体类型，跳过
                continue;
            }
            //RowChange对象，包含了一行数据变化的所有特征
            //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
            RowChange rowChage;
            try {
                rowChage = RowChange.parseFrom(entry.getStoreValue());
            } catch (Exception e) {
                throw new RuntimeException("ERROR ## parser of eromanga-event has an error , data:" + entry.toString()
            }
```

```
            //获取操作类型：insert/update/delete类型
            EventType eventType = rowChage.getEventType();
            //打印Header信息
            System.out.println();
            System.out.println(String.format("> binlog及偏移量[%s:%s] , 库/表[%s,%s] , 操作类型[%s]",
                    entry.getHeader().getLogfileName(), entry.getHeader().getLogfileOffset(),
                    entry.getHeader().getSchemaName(), entry.getHeader().getTableName(),
                    eventType));
            //判断是否是DDL语句
            if (rowChage.getIsDdl()) {
                System.out.println("> DDL sql: " + rowChage.getSql());
            }
            //获取RowChange对象里的每一行数据，打印出来
            for (CanalEntry.RowData rowData : rowChage.getRowDatasList()) {
                //删除语句
                if (eventType == EventType.DELETE) {
                    printColumn(rowData.getBeforeColumnsList());
                } else if (eventType == EventType.INSERT) {
                    printColumn(rowData.getAfterColumnsList());
                } else {
                    //变更前的数据
                    System.out.println(">> 变更前数据");
                    printColumn(rowData.getBeforeColumnsList());
                    //变更后的数据
                    System.out.println(">> 变更后数据");
                    printColumn(rowData.getAfterColumnsList());
                }
            }
        }
    }

    private static void printColumn(List<Column> columns) {
        for (Column column : columns) {
            System.out.println(column.getName() + " : " + column.getValue() + "    update=" + column.getUpdated());
        }
    }
}

```

五、测试

```
use sf_mall;

-- mysql-bin.000001,12232,"","",""
show master status;

-- 1.insert
insert into t_order(order_no, amount, count, addr) values
('202205270006', 120, 1, '这是test canal的一条信息');

-- mysql-bin.000001,12578,"","",""
show master status;

-- 2.update
update t_order set count = 3 where order_no = '202205270006';

-- mysql-bin.000001,12981,"","",""
```

```
show master status;
```




```
> binlog及偏移量[mysql-bin.000001:12786] , 库[sf_mall,t_order] , 操作类型[UPDATE]
>> 变更前数据
id : 5 update=false
order_no : 202205270006 update=false
amount : 120 update=false
count : 1 update=false
addr : 这是test canal的一条信息 update=false
phone : update=false
create_date : 2022-05-31 12:14:40 update=false
>> 变更后数据
id : 5 update=false
order_no : 202205270006 update=false
amount : 120 update=false
count : 3 update=true
addr : 这是test canal的一条信息 update=false
phone : update=false
create_date : 2022-05-31 12:14:40 update=false
```

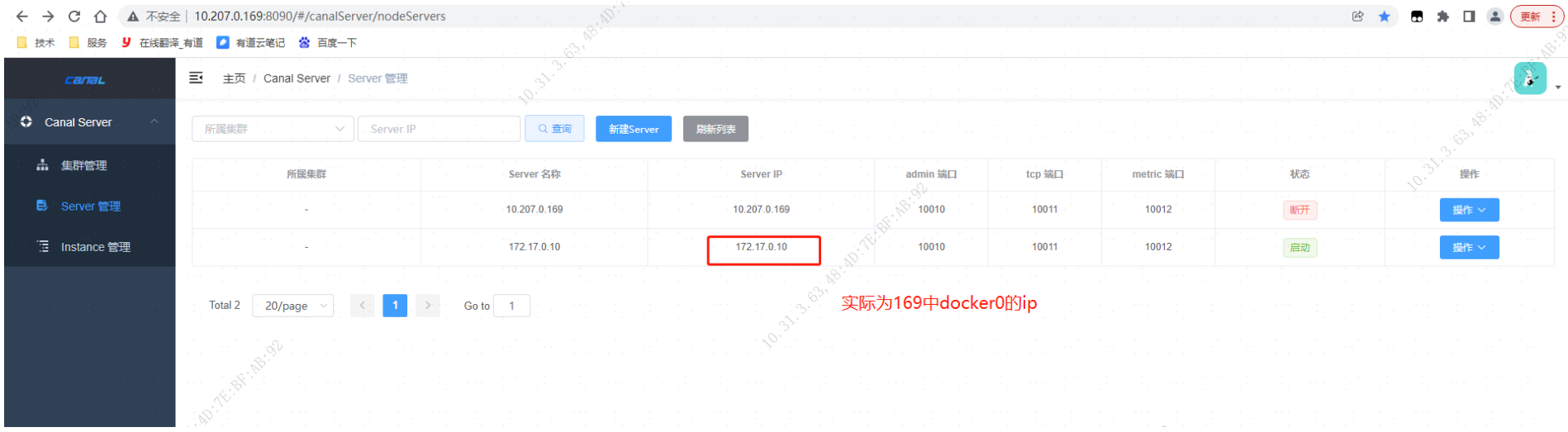
六、web界面

// 如需要指定外部mysql，可以使用下面脚本初始化

V1.0.0_2__create_….sql
3.92KB

```
// docker stop canal-admin;docker rm canal-admin; docker stop canal-server;docker rm canal-server
//WEB
docker run -d -it \
-p 8090:8089 \
-e server.port=8089 \
-e canal.adminUser=admin \
-e canal.adminPasswd=admin \
--name=canal-admin \
-m 1024m canal/canal-admin


//SERVER  11110-通信端口 11111-服务端口
docker run -d -it \
-p 10010:11110 \
-p 10011:11111 \
-p 10012:11112 \
-p 9100:9100 \
-e canal.admin.manager=10.207.0.169:8090 \
-e canal.admin.port=10010 \
-e canal.admin.user=admin \
-e canal.admin.passwd=4ACFE3202A5FF5CF467898FC58AAB1D615029441 \
```

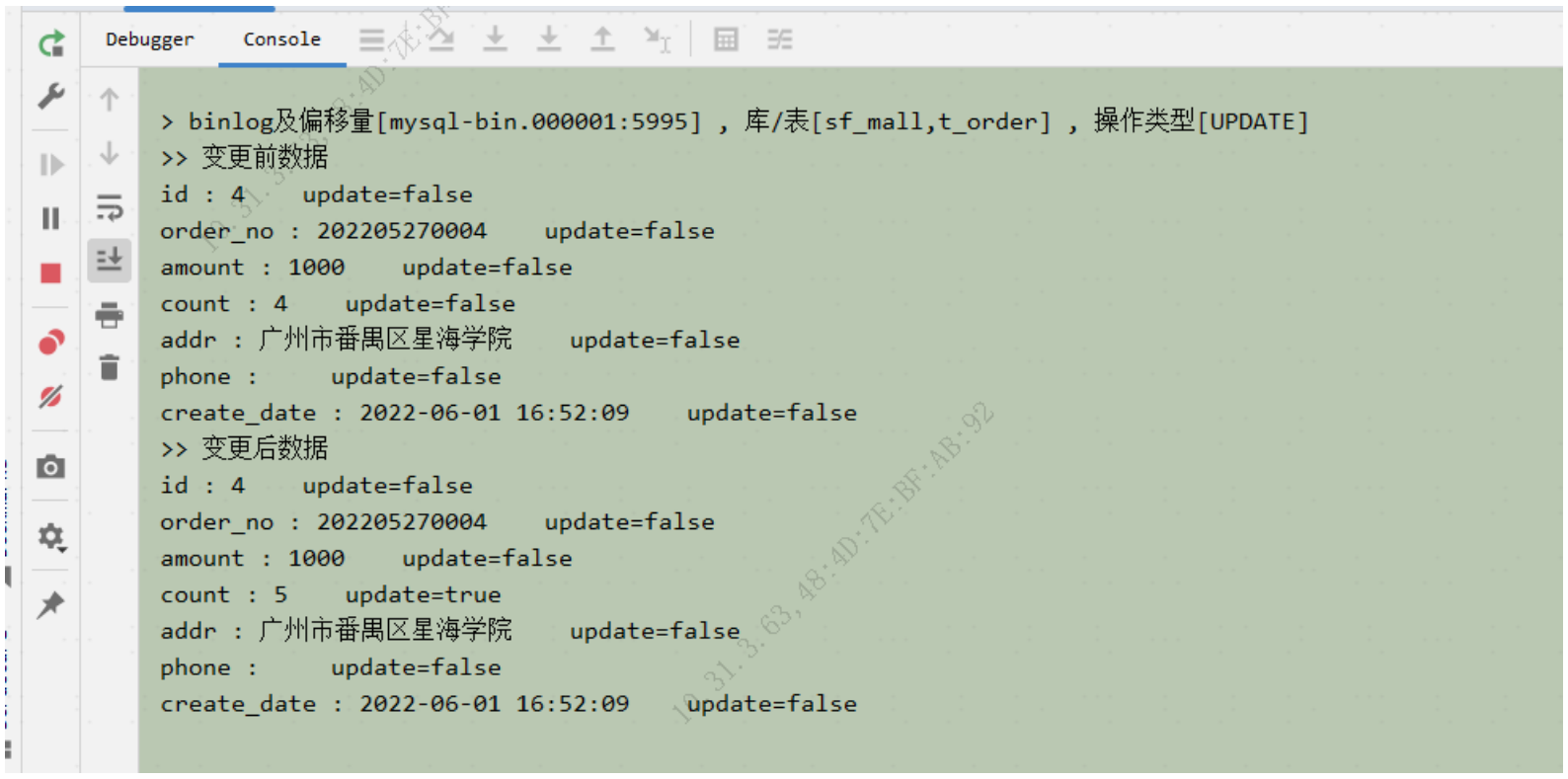
```
--name=canal-server \
-m 4096m canal/canal-server

2.访问，发现之前启动的server没有
http://10.207.0.169:8090   // admin/123456 登录

推送到mq
http://www.voycn.com/article/shiyong-docker-bushu-canalbingjiangxiaoxituisongdao-rabbitmq
```



Java客户端

```
Set() throws Exception {

r = CanalConnectors.newSingleConnector(new InetSocketAddress( hostname: "10.207.0.169", port: 10011), destination: "SF_INS", username: "canal", password: "canal");
```

修改一条数据



没日志

【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