数据传输服务

数据订阅（旧版）

产品文档

腾讯云

文档目录

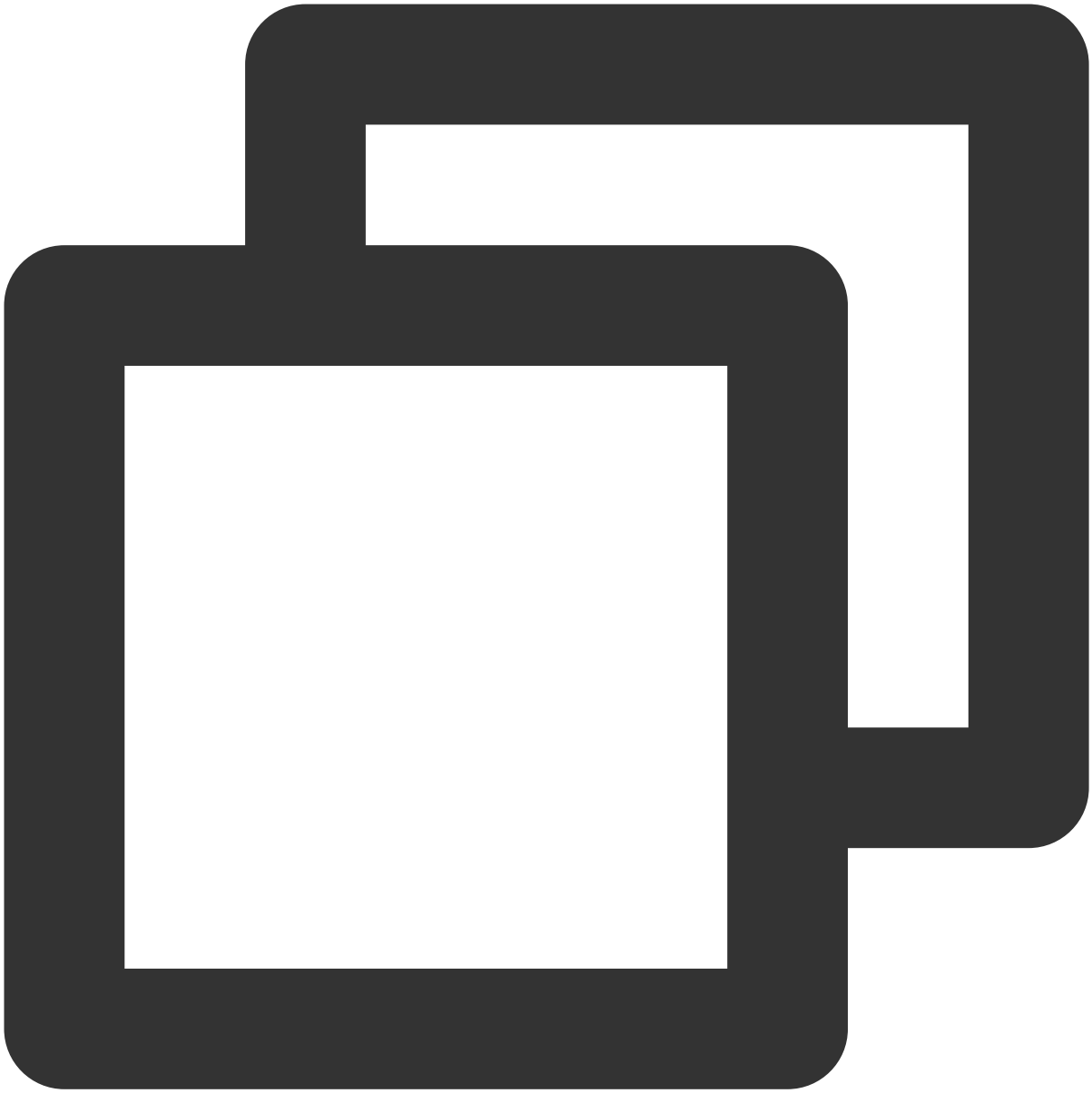
数据订阅（旧版）
数据订阅库表到本地

最近更新时间：2020-12-22 18:04:50

本文将以一个简单案例来说明数据订阅中拉取对应表到本地文件的功能，并且提供简易 LocalDemo 下载。以下操作将在 CentOS 操作系统中完成。

配置环境

Java 环境配置



```
yum install java-1.8.0-openjdk-devel
```

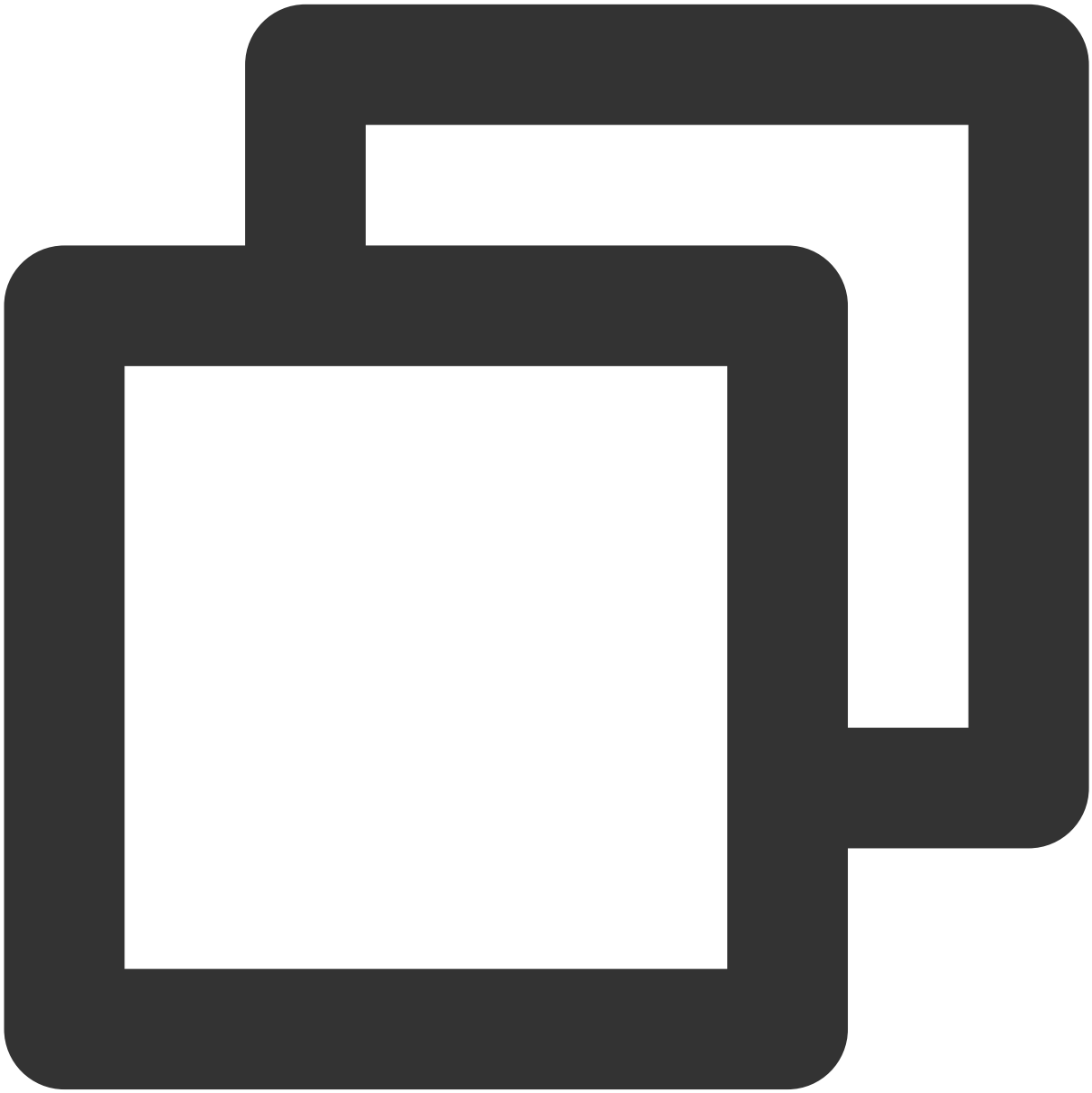
数据订阅 SDK 下载

获取密钥

登录 访问管理控制台 获取密钥。

选择数据订阅

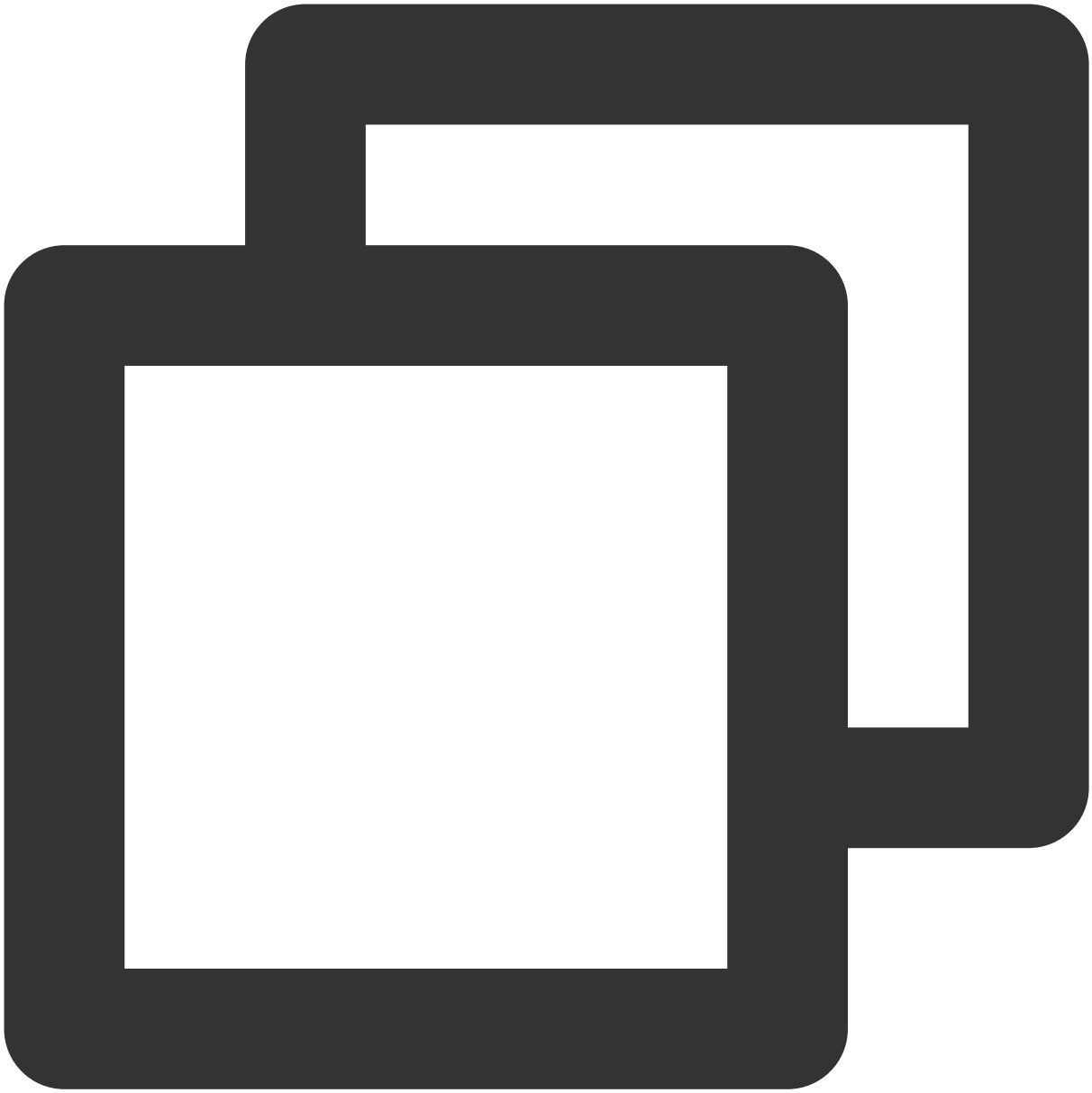
1. 登录 DTS 控制台，选择左侧的**数据订阅**，进入数据订阅页面。
2. 选择需同步的 TencentDB 实例名，然后单击启动，再返回数据订阅，单击您所创建的数据订阅。详细介绍请参考如何获取数据订阅。
3. 查看对应的 DTS 通道、IP 和 port，然后结合之前的密钥填写到对应 LocalDemo.java 里面。



```
// 从云API获取密钥,填写到此处
   context.setSecretId("AKIDfdsfdsfsdt1331431sdfds"); 请填写您从云API获取的secretID
   context.setSecretKey("test111usdfsdfsddsfRkeT"); 请填写
```

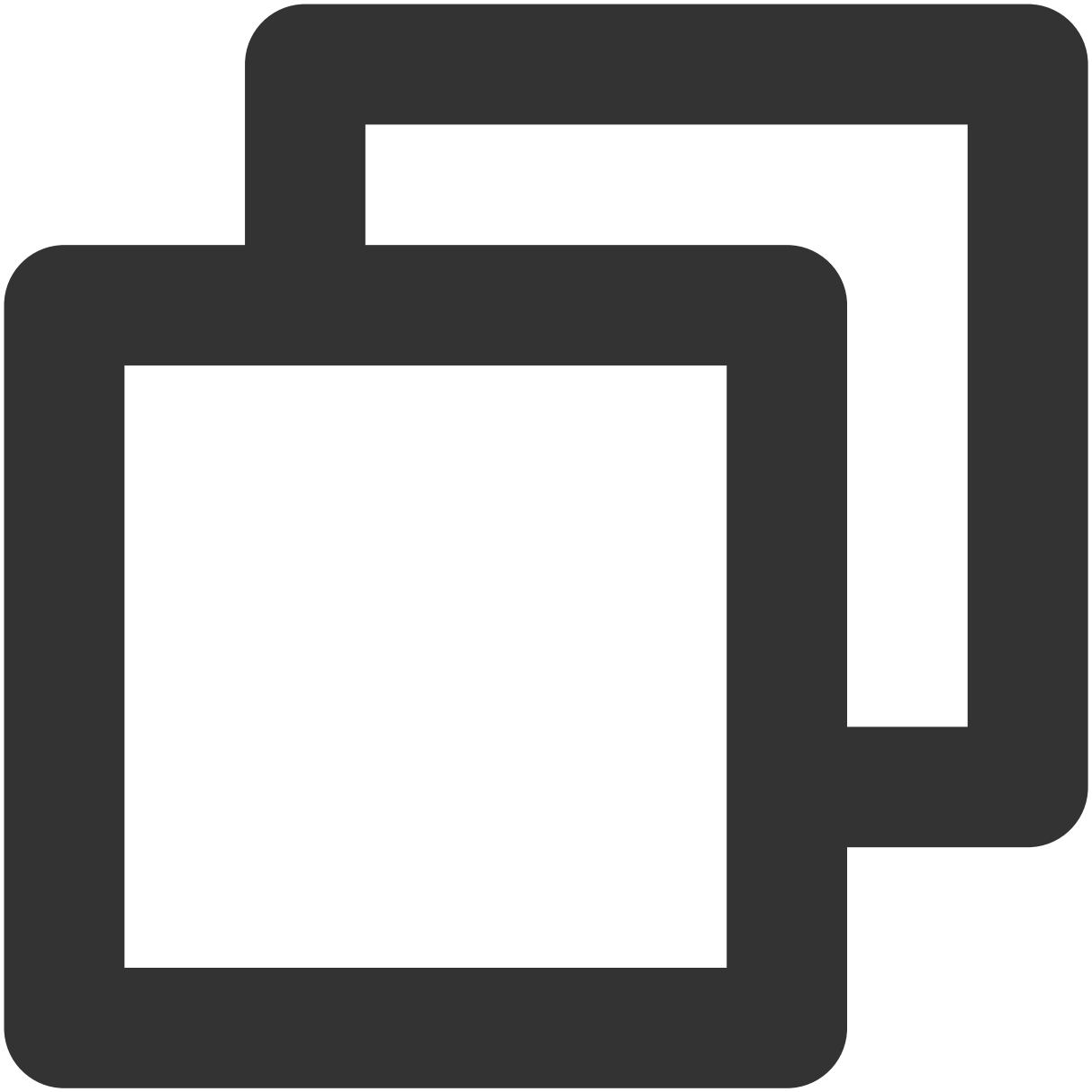
```
    您从云API获取的secretKey
// 在数据迁移服务里面通过数据订阅获取到对应的ip,port,填写到此处
    context.setServiceIp("10.66.112.181"); 请填写您从数据订阅配置获取到的IP
    context.setServicePort(7507);
    请填写您从数据订阅配置获取到的PORT
    final DefaultSubscribeClient client = new DefaultSubscribeClient(context);
// 填写对应要同步的数据库和表名,并修改对应要落地存储的文件名.
    final String targetDatabase = "test"; 填写您所要订阅的库名
    final String targetTable = "alantest"; 填写您所要订阅的表名

    final String fileName = "test.alan.txt"; 填写您希望落在本地的文件名

      client.addClusterListener(listener);
// 通过数据迁移订阅的配置选项获取到dts-channel的配置信息,填写到此处
client.askForGUID("dts-channel-e4FQxtYV3It4test"); 请填写您从数据订阅获取的通道dts的名称
    client.start();
```

编译操作和检验



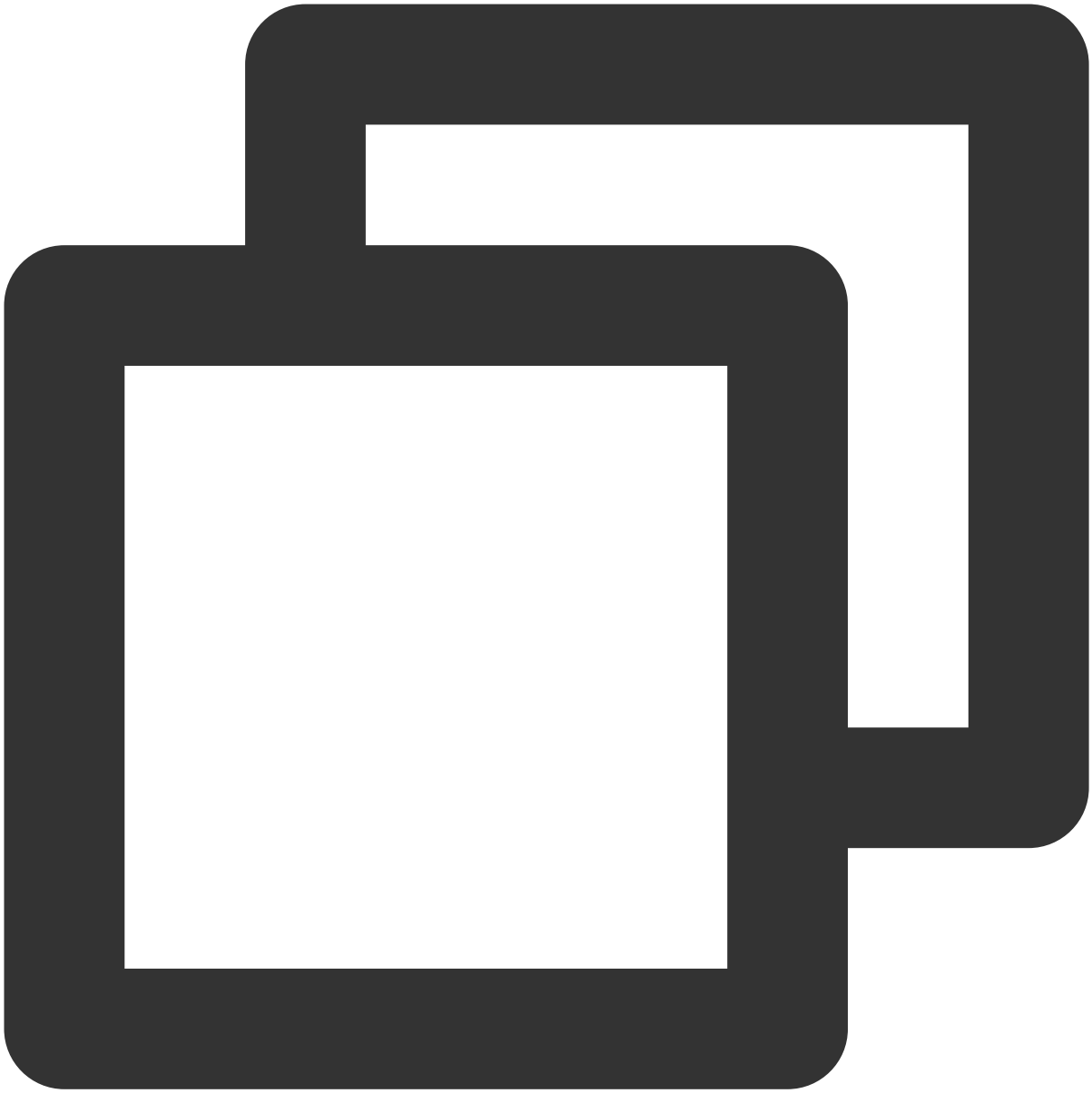
```
javac -classpath binlogsdk-2.6.0-release.jar -encoding UTF-8 LocalDemo.java
```

1. 然后执行启动，如果没有异常报错就是正常在服务了，然后查看对应之前设置的落地文件。



```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g -classpath .:binlogsdk-2.6.0-release.jar
```

2. 查看对应之前我们设置的 test.alan.txt 能看到已经有数据拉取到了本地。



```
[root@VM_71_10_centos ~]# cat test.alan.txt
checkpoint:144251@3@357589@317713
record_id:000001000000000004D9110000000000000001
record_encoding:utf8
fields_enc:latin1,utf8
gtid:4f21864b-3bed-11e8-a44c-5cb901896188:1562
source_category:full_recorded
source_type:mysql
table_name:alantest
record_type:INSERT
db:test
```

```
timestamp:1523356039
primary:
Field name: id
Field type: 3
Field length: 2
Field value: 26
Field name: name
Field type: 253
Field length: 4
Field value: alan
```

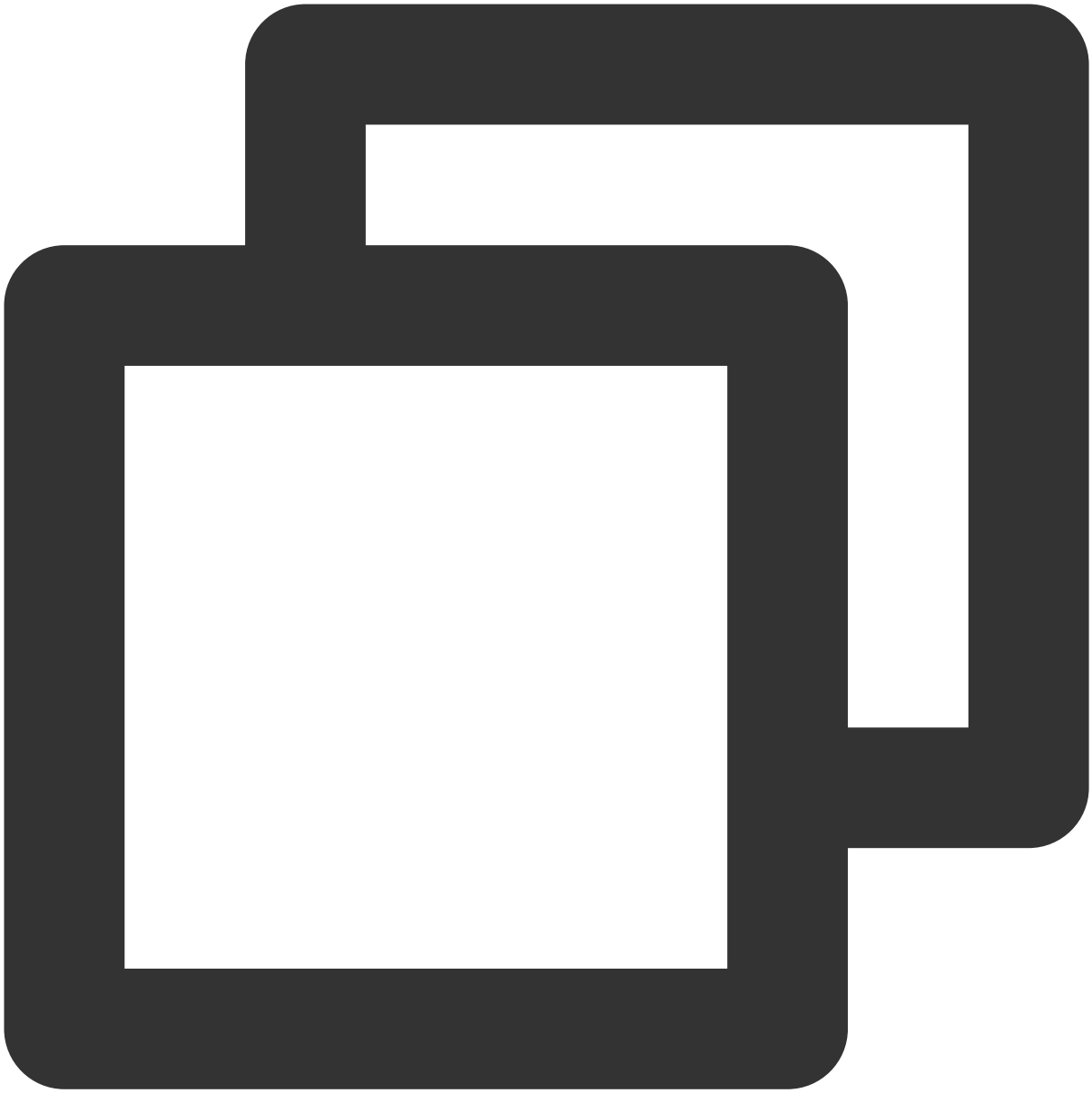
数据订阅库表到 Redis

最近更新时间：2024-02-19 09:59:34

本文将以一个简单案例来说明数据订阅中拉取对应表到 Redis 的功能，并且提供简易 RedisDemo 下载 。以下操作将在 CentOS 操作系统中完成。

配置环境

Java 环境配置



```
yum install java-1.8.0-openjdk-devel
```

数据订阅 SDK 下载

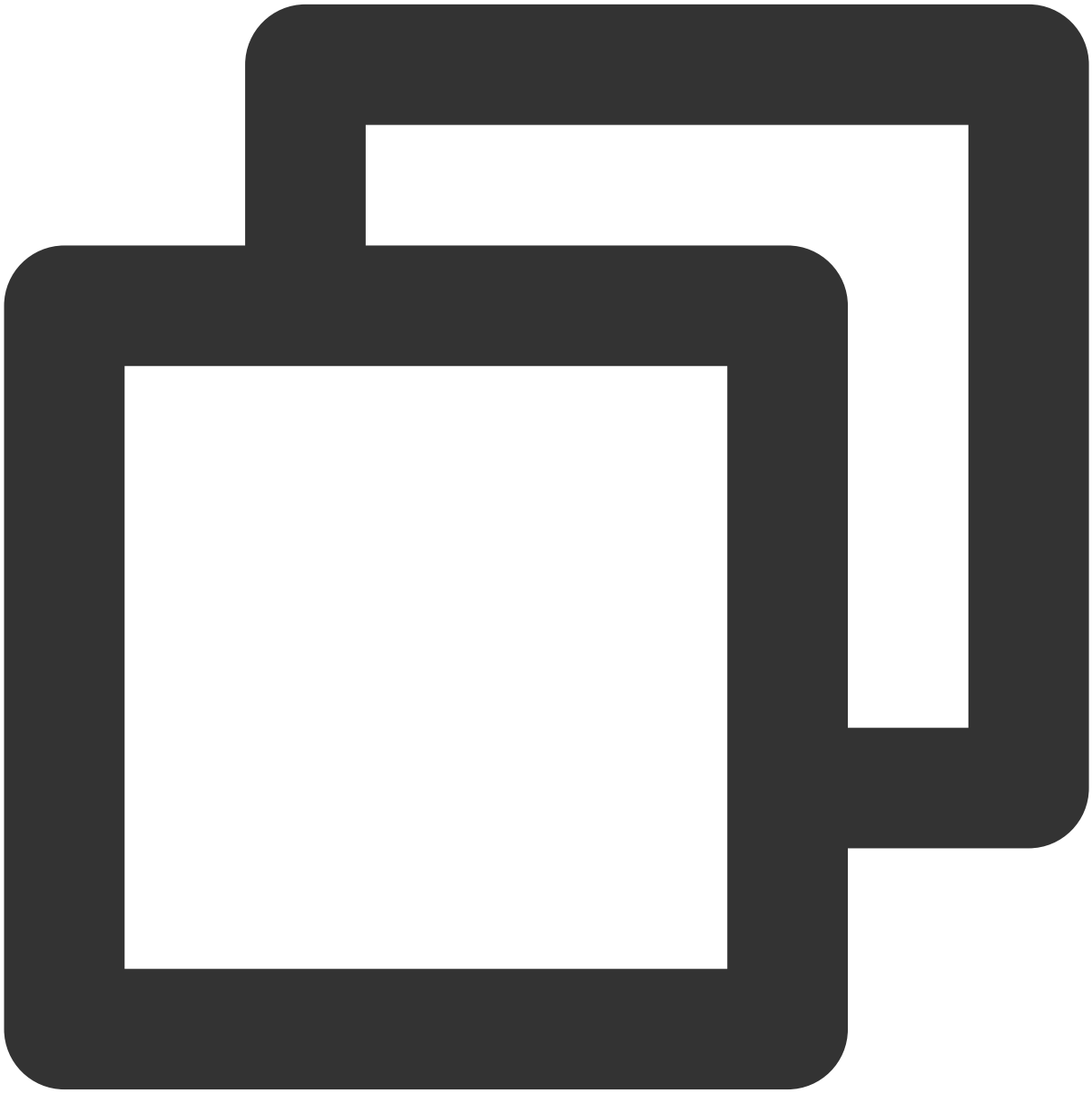
jedis-2.9.0.jar 下载

获取密钥

登录 访问管理控制台 获取密钥。

选择数据订阅

1. 登录 DTS 控制台，选择左侧的**数据订阅**，进入数据订阅页面。
2. 选择需同步的 TencentDB 实例名，然后单击启动，再返回数据订阅，单击您所创建的数据订阅。 详细介绍请参考如何获取数据订阅。
3. 查看对应的 DTS 通道、IP 和 Port，然后结合之前的密钥填写到对应 RedisDemo.java 里面。

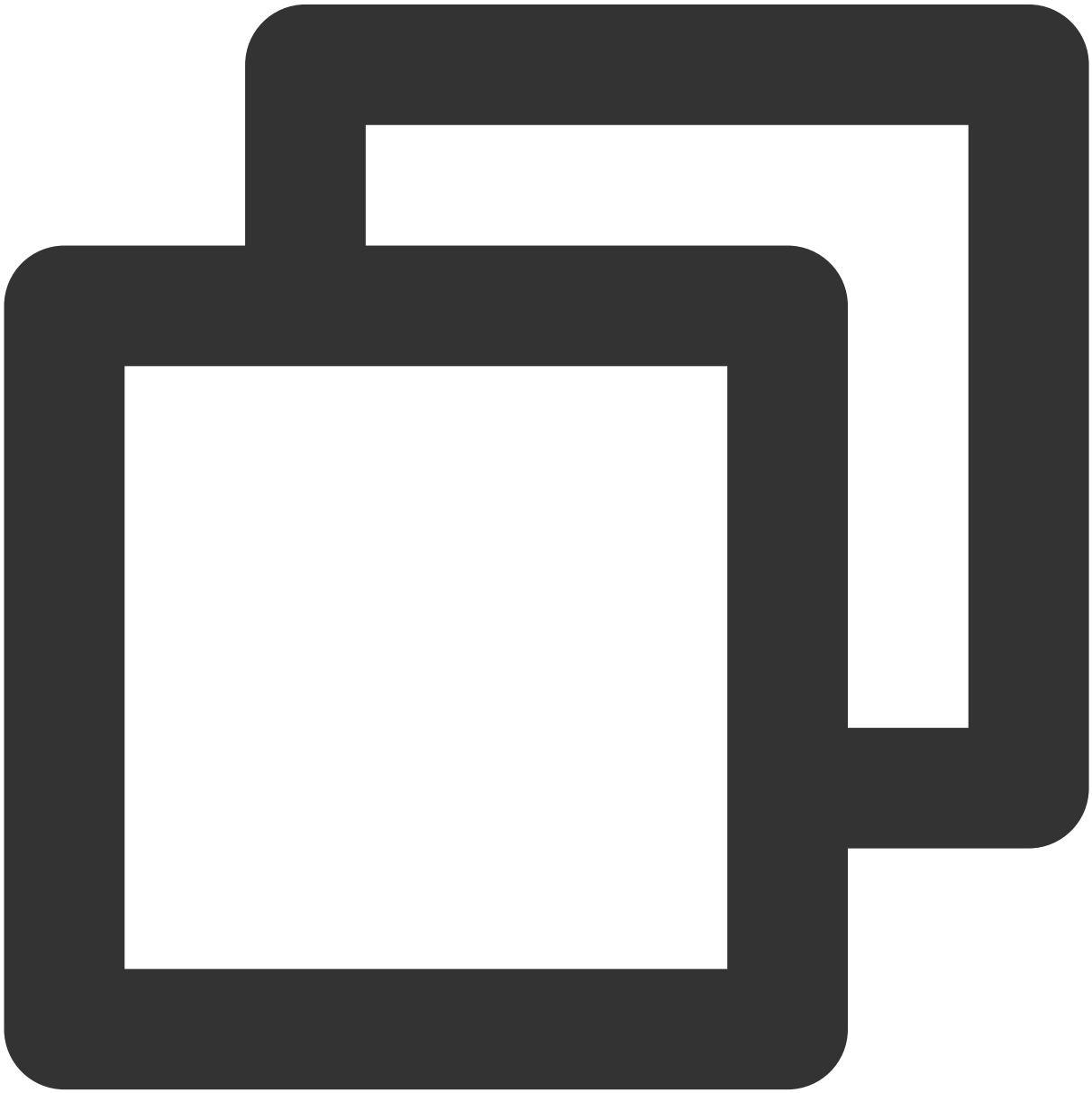


```
context.setSecretId("AKIDfdsfdsfsdt1331431sdfds"); 请填写您从云API获取的secretID。
        context.setSecretKey("test111usdfsdfsddsfRkeT"); 请填写您从云API获取的secretKe
    // 在数据迁移服务里面通过数据P阅获取到对应的ip,port,填写到此处
```

```
        context.setServiceIp("10.66.112.181"); 请填写您从数据订阅配置获取到的IP
        context.setServicePort(7507); 请填写您从数据订阅配置获取到的PORT

        // 创建消费者
        //SubscribeClient client=new DefaultSubscribeClient(context,true);
        final DefaultSubscribeClient client = new DefaultSubscribeClient(context);

        final Jedis jedis = new Jedis("127.0.0.1", 6379); 请填写您对应的redis主机和端口

        final String targetDatabase = "test"; 填写您所要订阅的库名
        final String targetTable = "alantest"; 填写您所要订阅的表名，表有2个字段分别是id,r

        // 创建订阅监听者listener
        ClusterListener listener = new ClusterListener() {
            @Override
            public void notify(List<ClusterMessage> messages) throws Exception {
        //                System.out.println("--------------------:" + messages.siz
                for(ClusterMessage m:messages){
                    DataMessage.Record record = m.getRecord();
                    //过滤不感兴趣的订阅信息
                if(!record.getDbName().equalsIgnoreCase(targetDatabase) || !record.
                        //注意：对于不感兴趣的信息也必须Ack
                        m.ackAsConsumed();
                        continue;
                    }

                    if(record.getOpt() != DataMessage.Record.Type.BEGIN && record.g
                        List<DataMessage.Record.Field> fields = record.getFieldList

                        //INSERT RECORD
                        //String pk = record.getPrimaryKeys();

                        if(record.getOpt() == DataMessage.Record.Type.INSERT){

                 String keyid="";
                 String value="";
                             for (DataMessage.Record.Field field : fields) {

                                 //先获取id值,需要有primary key,然后找到名为name的列，赋值
                 if(field.getFieldname().equalsIgnoreCase("id")){
                                     keyid=field.getValue();
                continue;
                                 }
                 if(field.getFieldname().equalsIgnoreCase("name")){
                     value=field.getValue();

                                 }
```

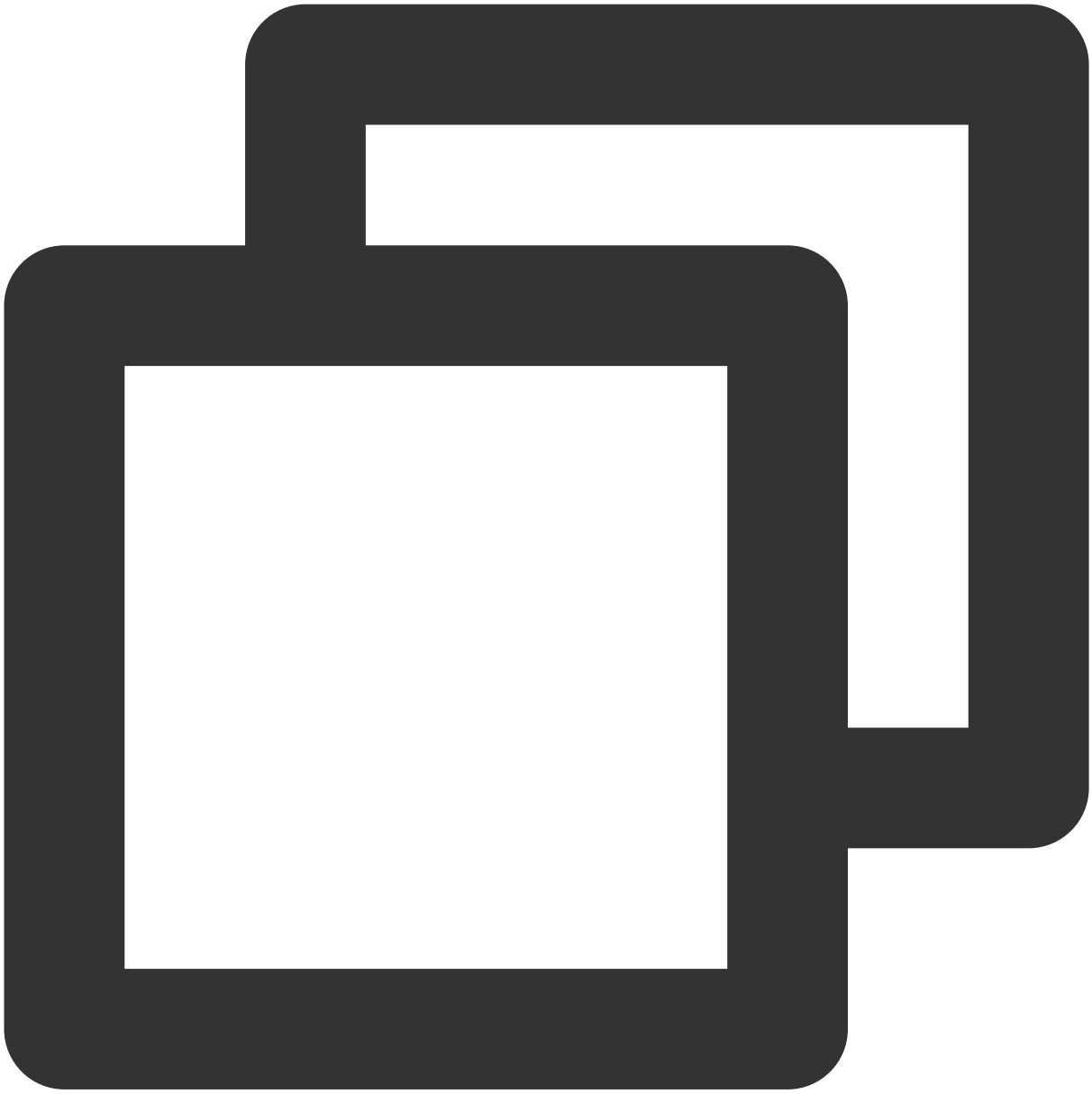
```
                    jedis.set(keyid, value);
                                }


                            }
```

编译操作与检验



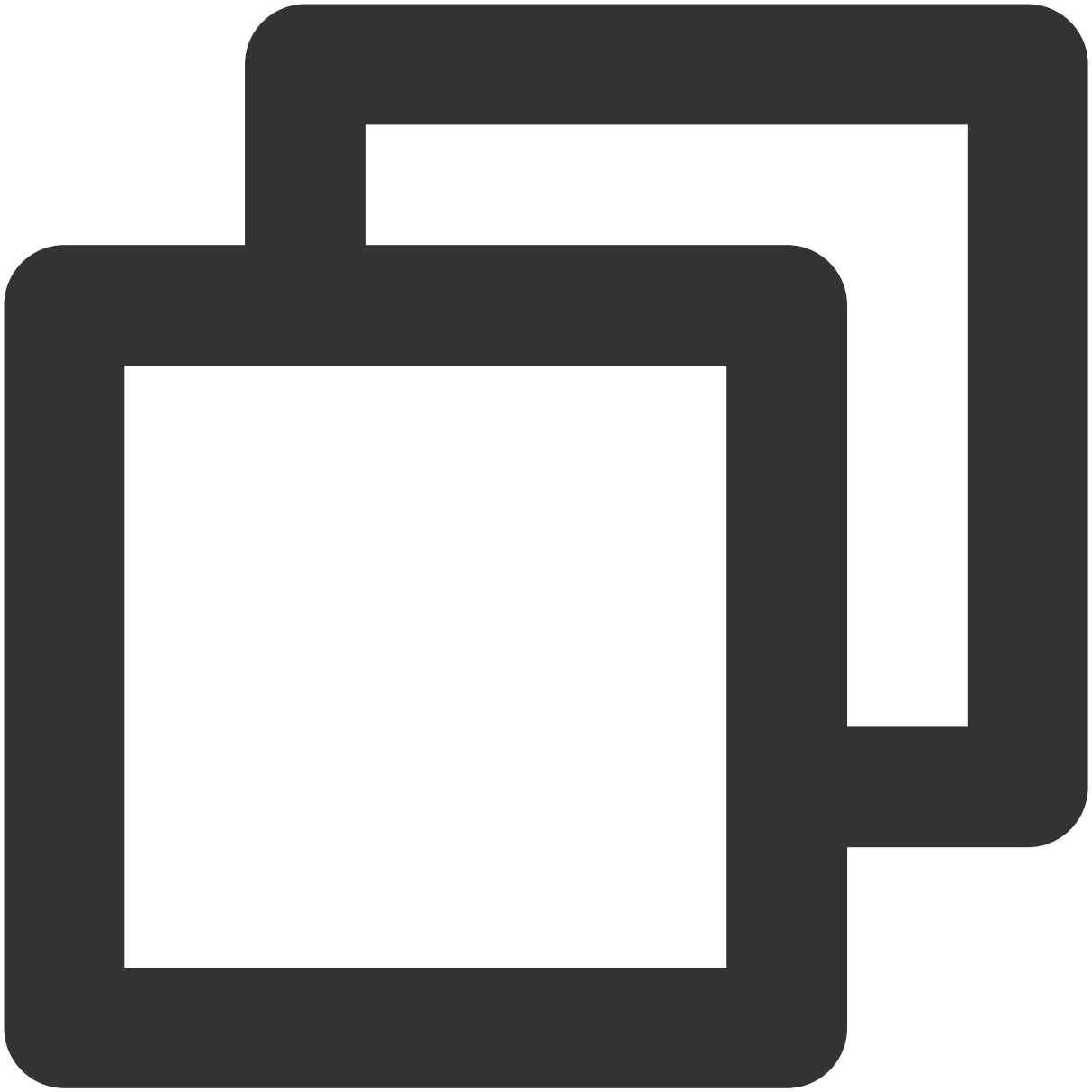
```
[root@VM_71_10_centos ~]# javac -classpath binlogsdk-2.6.0-release.jar:jedis-2.9.0.
```

1. 执行启动，如果没有异常报错就是正常在服务了，然后查看对应之前设置的落地文件。



```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g -classpath .:binlogsdk-2.6.0-release.ja
```

2. 查看进行数据库插入和 update 操作，并从 redis 观察发现确实插入并修改成功了,最后进行 delete 操作，redis 对应的数据也被删除掉了。



```
MySQL [test]> insert into alantest values(1001,'alan1');
Query OK, 1 row affected (0.00 sec)

MySQL [test]> update alantest set name='alan2' where id=1001;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

-------------------------
127.0.0.1:6379> get 1001
"alan2"
```

```
MySQL [test]> update alantest set name='alan3' where id=1001;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

------------------------
127.0.0.1:6379> get 1001
"alan3"

MySQL [test]> delete from alantest where id=1001;
Query OK, 1 row affected (0.00 sec)

------------------------

127.0.0.1:6379> get 1001
(nil)
```

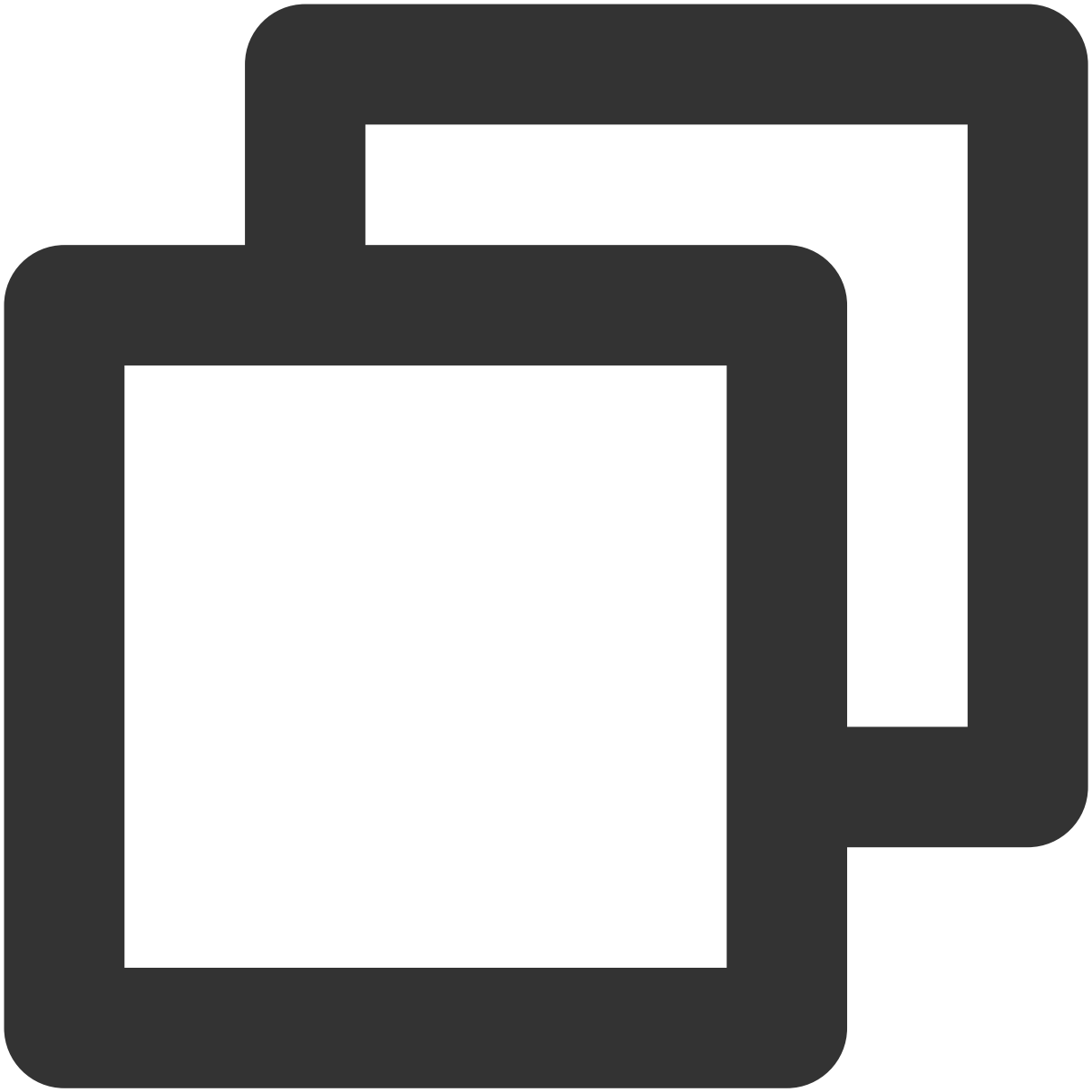
数据订阅库表到 Kafka

最近更新时间：2024-03-05 11:15:59

本文以一个简单案例来说明数据订阅中拉取对应表到 Kafka 的功能，并提供简易 KafkaDemo 下载。

配置环境

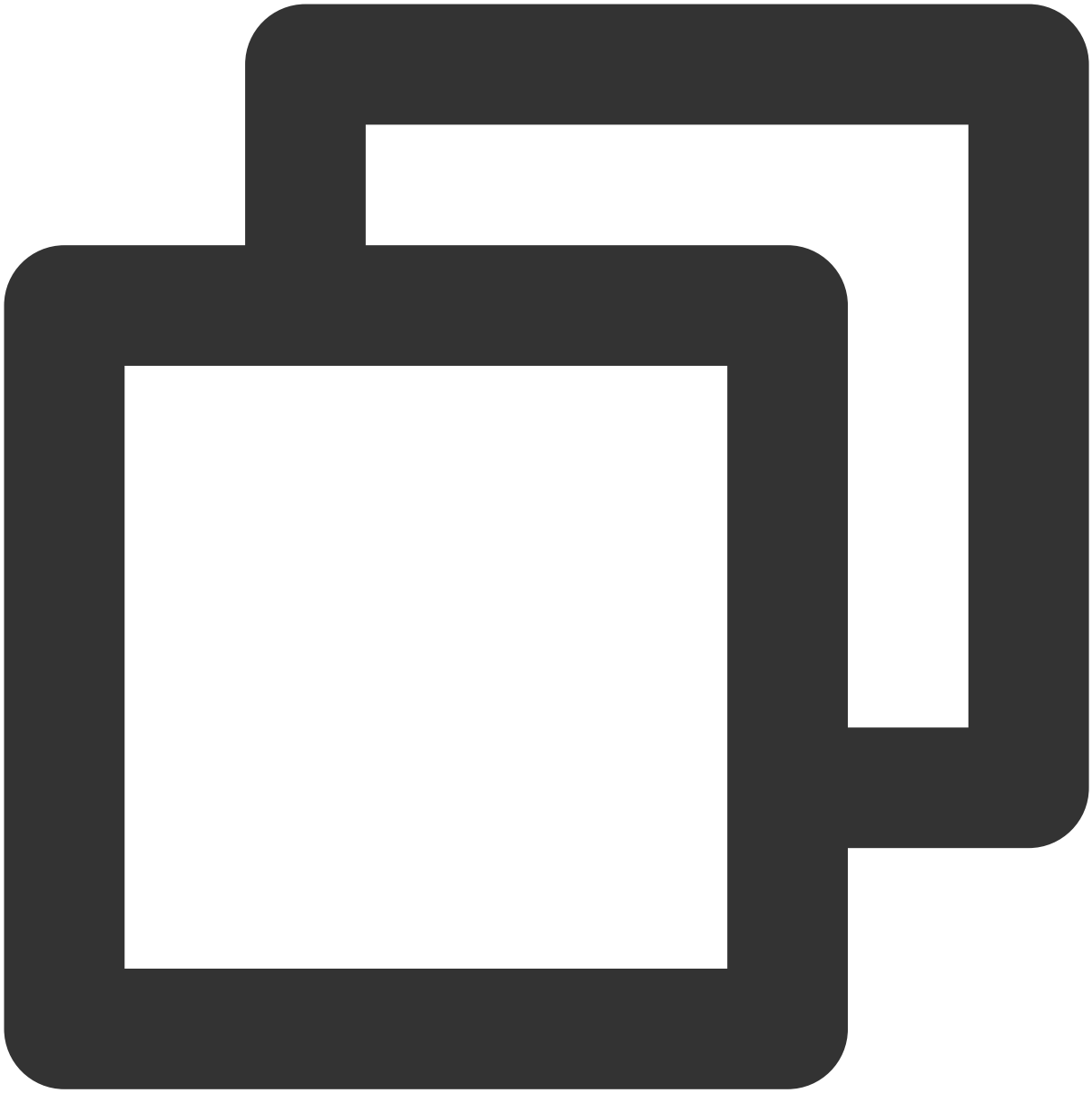
操作系统：CentOS
相关下载
数据订阅 SDK
SLF4J 组件
Kafka-clients
Java 环境配置



```
yum install java-1.8.0-openjdk-devel
```

安装 Kafka

1. 请参考 Kafka 介绍 安装 Kafka。
2. 启动之后创建一个 testtop 主题。



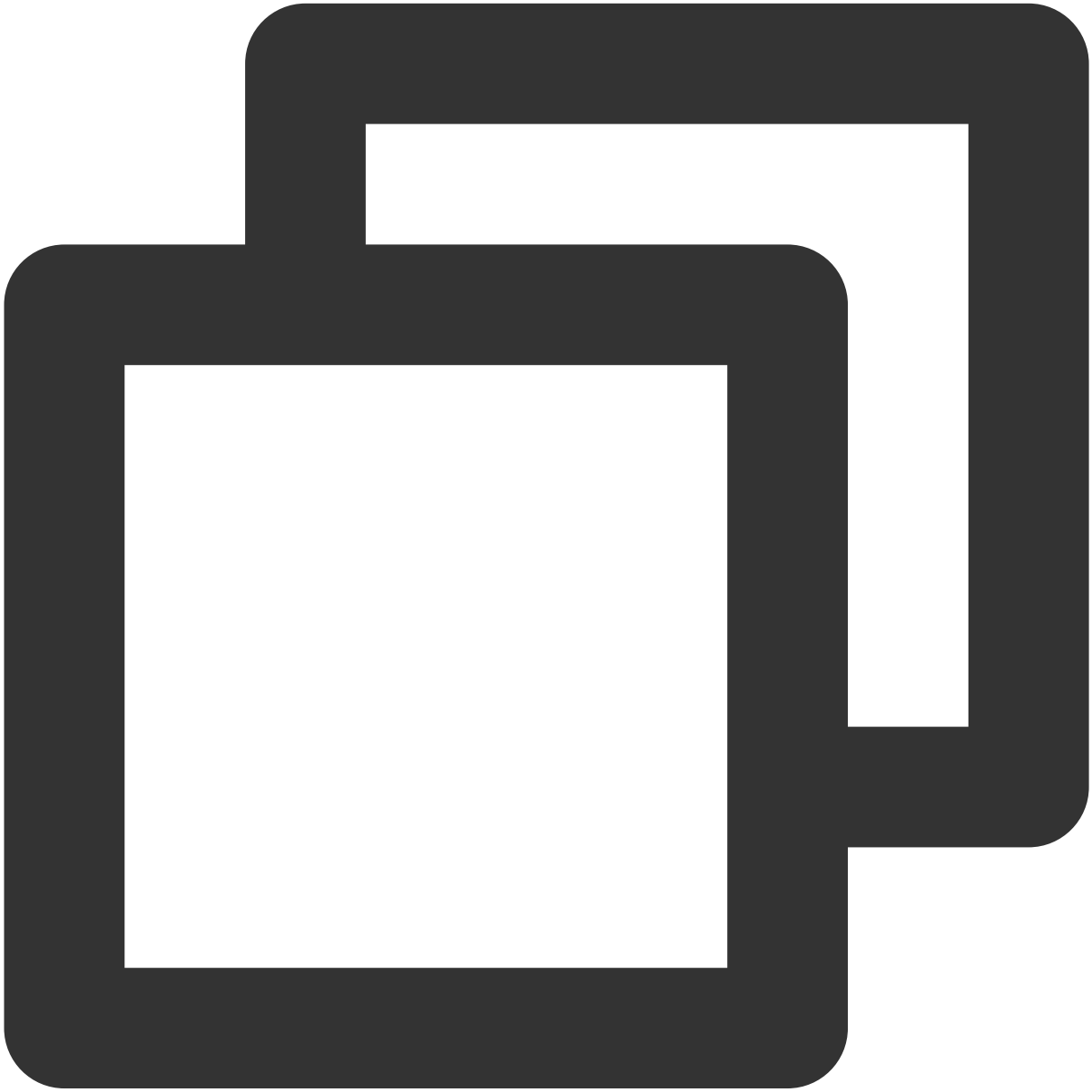
```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --pa
Created topic "testtop".
```

获取密钥

登录 访问管理控制台 获取密钥。

选择数据订阅

1. 登录 DTS 控制台，选择左侧的**数据订阅**，进入数据订阅页面。
2. 在订阅列表，单击订阅名，进入订阅详情页，查看对应的通道 ID、服务 IP 和服务端口。
3. 结合之前的密钥填写到对应 KafkaDemo.java 里。



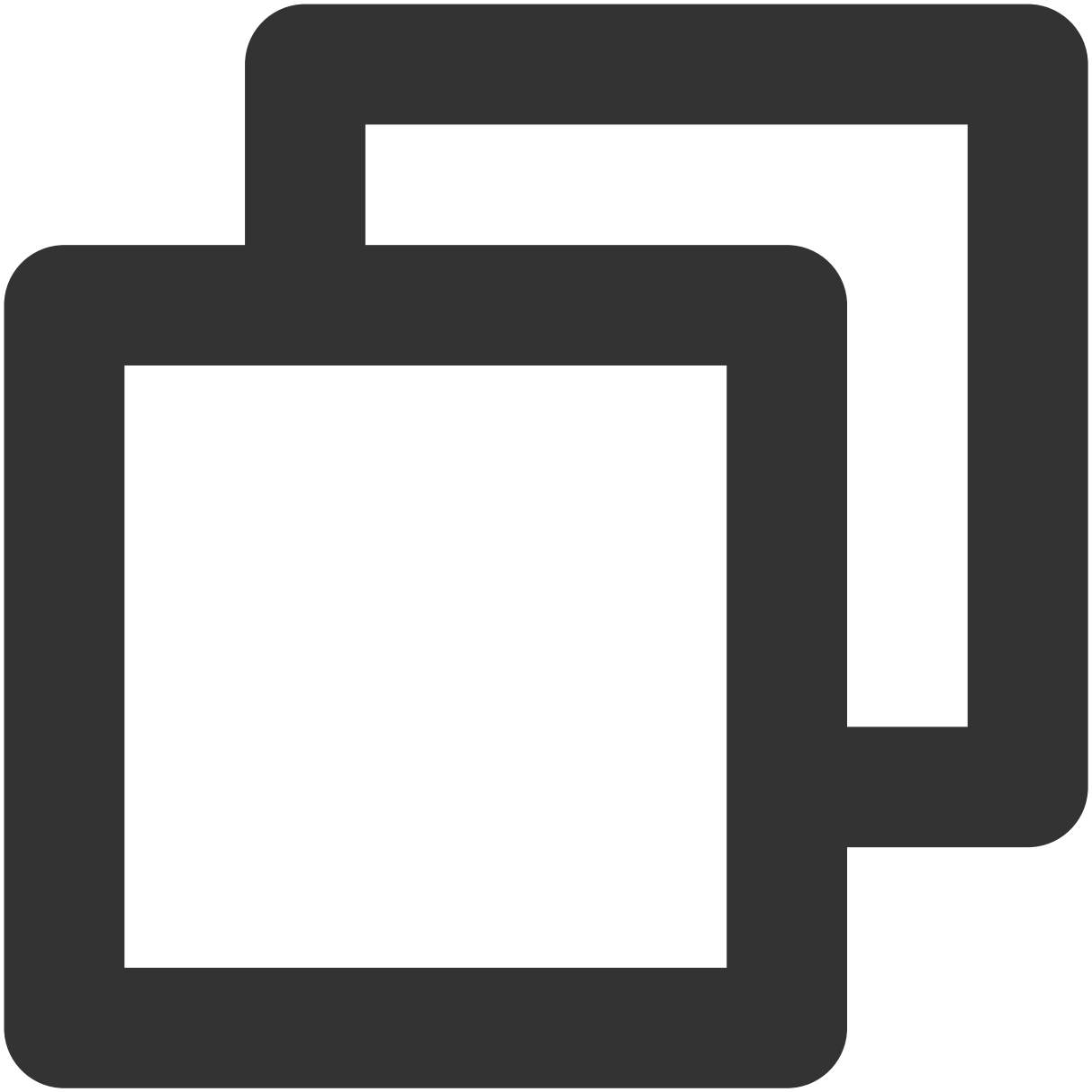
```
import com.qcloud.dts.context.SubscribeContext;
import com.qcloud.dts.message.ClusterMessage;
import com.qcloud.dts.message.DataMessage;
import com.qcloud.dts.subscribe.ClusterListener;
```

```
import com.qcloud.dts.subscribe.DefaultSubscribeClient;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.StringSerializer;
import org.apache.log4j.Logger;

import java.util.List;
import java.util.Properties;

public class KafkaDemo {
    public static void main(String[] args) throws Exception {
        //初始化一个 kafka producer
        final String TOPIC = "testtop";
        Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "10.168.1.6:9092");
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.cl
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.clas

        final Producer<String, String> producer = new KafkaProducer<String, String>

        // 创建一个 context
        SubscribeContext context = new SubscribeContext();
        context.setSecretId("AKIDPko5fVtvTDE0WffffkCwd4NzKcdePt79uauy");
        context.setSecretKey("ECtY8F5e2QqtdXAe18yX0EBqK");
        // 订阅通道所在 region
        context.setRegion("ap-beijing");
        final DefaultSubscribeClient client = new DefaultSubscribeClient(context);

        // 创建订阅监听者 listener
        ClusterListener listener = new ClusterListener() {
            @Override
            public void notify(List<ClusterMessage> messages) throws Exception {
                System.out.println("--------------------:" + messages.size());
                for(ClusterMessage m:messages){
                    DataMessage.Record record = m.getRecord();


                    if(record.getOpt() != DataMessage.Record.Type.BEGIN && record.g
                        List<DataMessage.Record.Field> fields = record.getFieldList

                        //打印每个列的信息
                        for (int i = 0; i < fields.size(); i++) {
                            DataMessage.Record.Field field = fields.get(i);
                            System.out.println("Database Name:" + record.getDbName(
                            System.out.println("Table Name:" + record.getTablename(
```

```
                            System.out.println("Field Value:" + field.getValue());
                            System.out.println("Field Value:" + field.getValue().le
                            System.out.println("Field Encoding:" + field.getFieldEn
                        }

                        //将整个 record 发送到指定的 kafka topic 中
                        System.out.println("Record++++++++++++++++++++++++++++++++
                        producer.send(new ProducerRecord<String, String>(TOPIC, rec
                    }

                    m.ackAsConsumed();
                }
            }
            @Override
            public void onException(Exception e){
                System.out.println("listen exception" + e);
            }
        };
        // 添加监听者
        client.addClusterListener(listener);
        client.askForGUID("dts-channel-p15e9eW9rn8hA68K");
        client.start();
    }

}
```

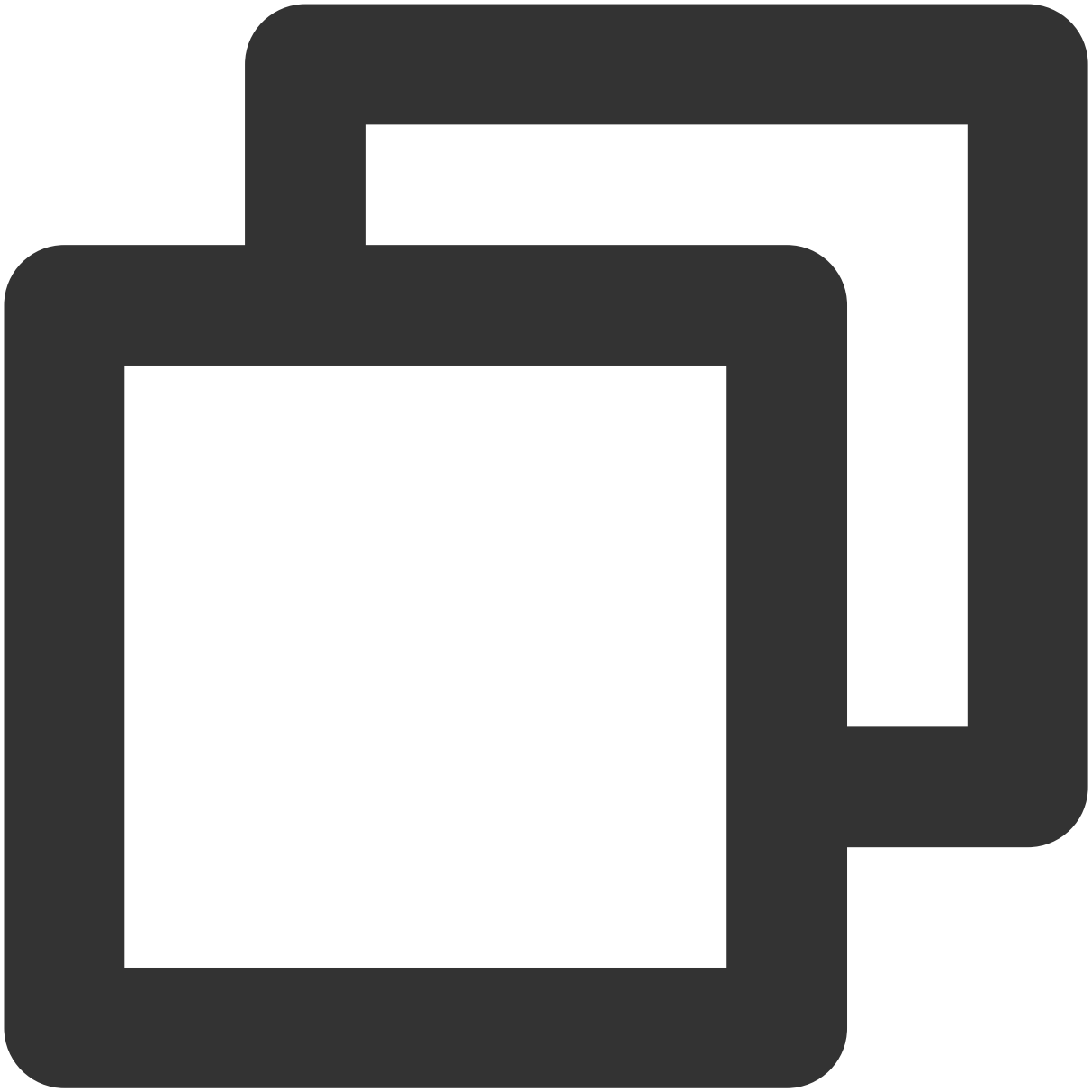
编译操作与检验

1. 编译客户端程序 KafkaDemo.java。



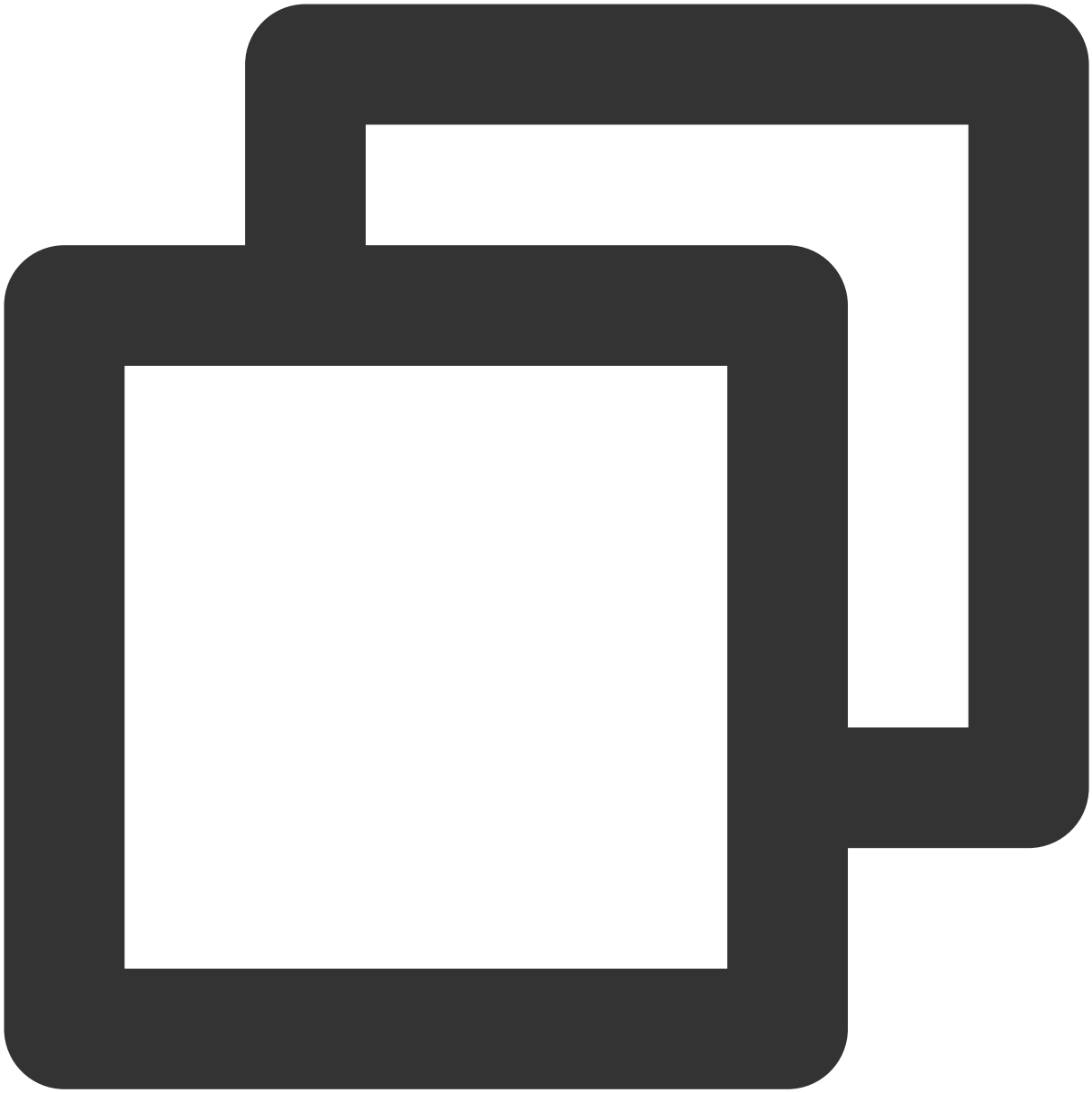
```
javac -classpath binlogsdk-2.9.1-jar-with-dependencies.jar:slf4j-api-1.7.25.jar:slf
```

2. 执行启动，如无异常报错即为正常在服务。



```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g  -classpath .:binlogsdk-2.9.1-jar-with-d
```

3. 通过对表 alantest 插入一条数据，发现在 Kafka 订阅的 testtop 里面能看到已经有数据过来了。



```
MySQL [test]> insert into alantest values(123456,'alan');
Query OK, 1 row affected (0.02 sec)
[root@VM_71_10_centos kafka_2.11-1.1.0]#  bin/kafka-console-consumer.sh --bootstrap
checkpoint:144251@3@1275254@1153089
record_id:00000100000000001198410000000000000001
record_encoding:utf8
fields_enc:latin1,utf8
gtid:4f21864b-3bed-11e8-a44c-5cb901896188:5552
source_category:full_recorded
source_type:mysql
table_name:alantest
```

```
record_type:INSERT
db:test
timestamp:1524649133
primary:id
Field name: id
Field type: 3
Field length: 6
Field value: 123456
Field name: name
Field type: 253
Field length: 4
Field value: alan
```