canal数据库数据同步

概述

canal可以用来监控数据库数据的变化，从而获得新增数据，或者修改的数据。

canal是应阿里巴巴存在杭州和美国的双机房部署，存在跨机房同步的业务需求而提出的。

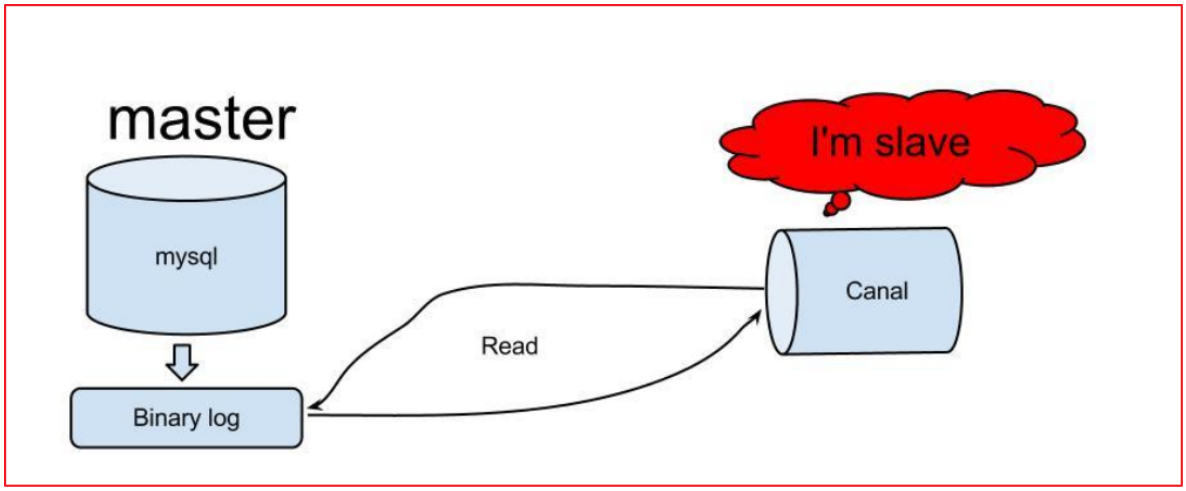
阿里系公司开始逐步的尝试基于数据库的日志解析，获取增量变更进行同步，由此衍生出了增量订阅&消费的业务。

优点

Canal应用场景非常多，如下：

1. 数据库镜像
2. 数据库实时备份
3. 多级索引 (卖家和买家各自分库索引)
4. search build
5. 业务cache刷新
6. 价格变化等重要业务消息

Canal工作原理



原理相对比较简单：

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(也就是canal)
3. canal解析binary log对象(原始为byte流)

canal需要使用到mysql，我们需要先安装mysql,给大家发的虚拟机中已经安装了mysql容器，但canal是基于mysql的主从模式实现的，所以必须先开启binlog.

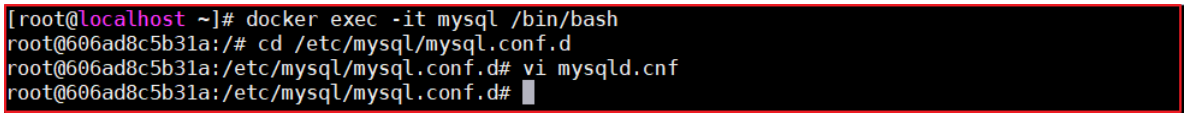
Canal安装

开启binlog模式

linux上安装mysql容器.此处不在演示。

(1) 修改/etc/my.cnf 需要开启主 从模式，开启binlog模式。

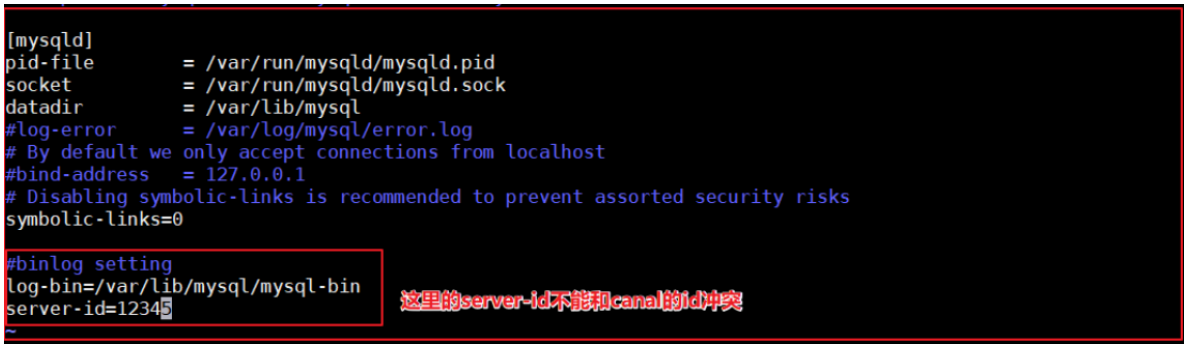
执行如下命令，编辑mysql配置文件



命令行如下：

```
docker exec -it mysql /bin/bash
cd /etc/mysql/mysql.conf.d
vi mysqld.cnf
```

修改mysqld.cnf配置文件，添加如下配置：



上图配置如下：

```
log-bin/var/lib/mysql/mysql-bin
server-id=12345
```

(2) 创建账号 用于测试使用,

使用root账号创建用户并授予权限

```
create user canal@'%' IDENTIFIED by 'canal';
GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT,SUPER ON *.* TO
'canal'@'%';
FLUSH PRIVILEGES;
```

(3)重启mysql容器

```
docker restart mysql
```

canal容器安装

下载镜像：

```
docker pull docker.io/canal/canal-server
```

容器安装

```
docker run -p 11111:11111 --name canal -d docker.io/canal/canal-server
```

进入容器,修改核心配置canal.properties 和instance.properties，canal.properties 是canal自身的配置，instance.properties是需要同步数据的数据库连接配置。

执行代码如下:

```
docker exec -it canal /bin/bash
cd canal-server/conf/
vi canal.properties
cd example/
vi instance.properties
```

修改canal.properties的id，不能和mysql的server-id重复，如下图：

```
#canal.manager.jdbc.password=121212
canal.id = 1001
canal.ip =
canal.port = 11111
canal.metrics.pull.port = 11112
```

修改instance.properties,配置数据库连接地址:

```
#################################################
## mysql serverId , v1.0.26+ will autoGen
# canal.instance.mysql.slaveId=0

# enable gtid use true/false
canal.instance.gtidon=false

# position info
canal.instance.master.address=172.17.0.2:3306    需要监听的数据库连接配置
canal.instance.master.journal.name=
canal.instance.master.position=
canal.instance.master.timestamp=
canal.instance.master.gtid=

# rds oss binlog
canal.instance.rds.accesskey=
canal.instance.rds.secretkey=
canal.instance.rds.instanceId=

# table meta tsdb info
canal.instance.tsdb.enable=true
#canal.instance.tsdb.url=jdbc:mysql://127.0.0.1:3306/canal_tsdb
#canal.instance.tsdb.dbUsername=canal
#canal.instance.tsdb.dbPassword=canal

#canal.instance.standby.address =
#canal.instance.standby.journal.name =
#canal.instance.standby.position =
#canal.instance.standby.timestamp =
#canal.instance.standby.gtid=

# username/password
canal.instance.dbUsername=canal
canal.instance.dbPassword=canal
canal.instance.connectionCharset = UTF-8
# enable druid Decrypt database password
canal.instance.enableDruid=false
#canal.instance.pwdPublicKey=MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALK4BUxdDltRRE5/zXpVEVPUgunvscYFtEip3pmLlhrWpa
EAAQ==

# table regex
canal.instance.filter.regex=.*\\..*    需要监听的表正则表达式
# table black regex
canal.instance.filter.black.regex=
```

这里的`canal.instance.filter.regex`有多种配置，如下：

```
mysql 数据解析关注的表，Perl正则表达式.
多个正则之间以逗号(,)分隔，转义符需要双斜杠(\\)
常见例子：
1.  所有表：.*   or  .*\\..*
2.  canal schema下所有表： canal\\..*
3.  canal下的以canal打头的表：canal\\.canal.*
4.  canal schema下的一张表：canal.test1
5.  多个规则组合使用：canal\\..*,mysql.test1,mysql.test2 (逗号分隔)
注意：此过滤条件只针对row模式的数据有效(ps. mixed/statement因为不解析sql，所以无法准确提取tableName进行过滤)
```

配置完成后，设置开机启动，并记得重启canal。

```
docker update --restart=always canal
docker restart canal
```

canal务搭建

当用户执行 数据库的操作的时候，binlog 日志会被canal捕获到，并解析出数据。我们就可以将解析出来的数据进行同步到redis中即可。

思路：创建一个独立的程序，并监控canal服务器，获取binlog日志，解析数据，将数据更新到redis中。这样广告的数据就更新了。

(1)安装辅助jar包

在`canal\spring-boot-starter-canal-master`中有一个工程`starter-canal`，它主要提供了SpringBoot环境下`canal`的支持，我们需要先安装该工程，在`starter-canal`目录下执行`mvn install`，如下图：

```
Microsoft Windows [版本 10.0.17134.829]
(c) 2018 Microsoft Corporation。保留所有权利。

D:\documents\畅购\讲义\资源\canal\spring-boot-starter-canal-master\starter-canal>mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building starter-canal 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.1:resources (default-resources) @ starter-canal ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ starter-canal ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 23 source files to D:\documents\畅购\讲义\资源\canal\spring-boot-starter-canal-master\starter-canal\tar
get\classes
[INFO]
[INFO] --- maven-resources-plugin:3.0.1:testResources (default-testResources) @ starter-canal ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\documents\畅购\讲义\资源\canal\spring-boot-starter-canal-master\starter-ca
nal\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:testCompile (default-testCompile) @ starter-canal ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.20.1:test (default-test) @ starter-canal ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ starter-canal ---
[INFO] Building jar: D:\documents\畅购\讲义\资源\canal\spring-boot-starter-canal-master\starter-canal\target\starter-car
al-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ starter-canal ---
[INFO] Installing D:\documents\畅购\讲义\资源\canal\spring-boot-starter-canal-master\starter-canal\target\starter-canal-
0.0.1-SNAPSHOT.jar to D:\dev\install\apache-maven-3.5.2\conf\repo\com\xpand\starter-canal\0.0.1-SNAPSHOT\starter-canal-0
.0.1-SNAPSHOT.jar
[INFO] Installing D:\documents\畅购\讲义\资源\canal\spring-boot-starter-canal-master\starter-canal\pom.xml to D:\dev\ins
tall\apache-maven-3.5.2\conf\repo\com\xpand\starter-canal\0.0.1-SNAPSHOT\starter-canal-0.0.1-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.379 s
[INFO] Finished at: 2019-06-18T07:55:45+08:00
[INFO] Final Memory: 32M/461M
[INFO] ------------------------------------------------------------------------
```

(2)canal工程搭建

基于springboot创建canal-service工程，并引入相关配置。

pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.itheima</groupId>
    <artifactId>canal-service</artifactId>
    <version>1.0-SNAPSHOT</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.4.RELEASE</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

```

```
        <!--canal依赖-->
        <dependency>
            <groupId>com.xpand</groupId>
            <artifactId>starter-canal</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
    </dependencies>

</project>
```

application.yml配置

```
server:
  port: 18081
spring:
  application:
    name: canal
#canal配置
canal:
  client:
    instances:
      example:
        host: 192.168.211.132
        port: 11111
```

(3)监听创建

创建一个CanalDataEventListener类，实现对表增删改操作的监听，代码如下：

```
package com.itheima.canal.listener;
import com.alibaba.otter.canal.protocol.CanalEntry;
import com.xpand.starter.canal.annotation.*;
@CanalEventListener
public class CanalDataEventListener {

    /***
     * 增加数据监听
     * @param eventType
     * @param rowData
     */
    @InsertListenPoint
    public void onEventInsert(CanalEntry.EventType eventType,
CanalEntry.RowData rowData) {
        rowData.getAfterColumnsList().forEach((c) -> System.out.println("By-
-Annotation: " + c.getName() + " ::   " + c.getValue()));
    }

    /***
     * 修改数据监听
     * @param rowData
     */
    @UpdateListenPoint
    public void onEventUpdate(CanalEntry.RowData rowData) {
```

```
        System.out.println("UpdateListenPoint");
        rowData.getAfterColumnsList().forEach((c) -> System.out.println("By-
-Annotation: " + c.getName() + " ::   " + c.getValue()));
    }

    /***
     * 删除数据监听
     * @param eventType
     */
    @DeleteListenPoint
    public void onEventDelete(CanalEntry.EventType eventType) {
        System.out.println("DeleteListenPoint");
    }

    /***
     * 自定义数据修改监听
     * @param eventType
     * @param rowData
     */
    @ListenPoint(destination = "example", schema = "changgou_content", table
= {"tb_content_category", "tb_content"}, eventType =
CanalEntry.EventType.UPDATE)
    public void onEventCustomUpdate(CanalEntry.EventType eventType,
CanalEntry.RowData rowData) {
        System.err.println("DeleteListenPoint");
        rowData.getAfterColumnsList().forEach((c) -> System.out.println("By-
-Annotation: " + c.getName() + " ::   " + c.getValue()));
    }
}
```

(4)启动类创建

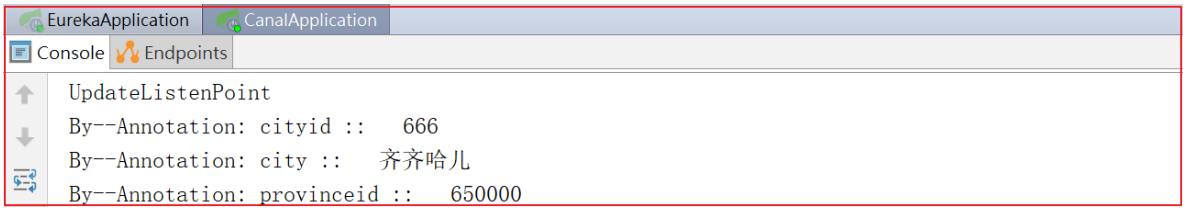
在com.changgou中创建启动类，代码如下：

```
package com.itheima;
import com.xpand.starter.canal.annotation.EnableCanalClient;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
@SpringBootApplication(exclude={DataSourceAutoConfiguration.class})
@EnableCanalClient
public class CanalApplication {

    public static void main(String[] args) {
        SpringApplication.run(CanalApplication.class,args);
    }
}
```

测试

启动canal微服务，然后修改任意数据库的表数据，canal微服务后台输出如下：



总结

通过上面的案例，我们可以看到利用Canal可以实时监控MySQL数据的变化，并且可以指定对应的表，这样可以很方便做数据整合或者做数据同步、刷新缓存等操作，Canal在工作中的应用场景非常多，作用非常强大。