配置详见：https://github.com/alibaba/canal/wiki/AdminGuide

同步实例配置

```
{canal-path}/example/instance.properties
#################################################
## mysql serverId , v1.0.26+ will autoGen
# canal.instance.mysql.slaveId=0

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

# position info 同步地址与同步点位设置
canal.instance.master.address=127.0.0.1:3306
#binlog名
canal.instance.master.journal.name=
#同步主库的binlog起始点位
canal.instance.master.position=
#时间点
canal.instance.master.timestamp=
#gtid
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=

# 配置你的mysql数据库的 username/password/encode
canal.instance.dbUsername=root
canal.instance.dbPassword=root
canal.instance.connectionCharset = UTF-8
#你的数据库
canal.instance.defaultDatabaseName = micromall
# enable druid Decrypt database password
canal.instance.enableDruid=false
#canal.instance.pwdPublicKey=MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALK4BUxdDltRRE5/zXpVEVPUgunvscYFtEip3pmLlhrWpacX7y7GCMo2/JM6LeHmiiNcH1FWgGCpUfircSwlWKUCAwEAAQ==

# table regex,订阅哪些表
canal.instance.filter.regex=.*\\..*
# table black regex,黑名单表不订阅信息
canal.instance.filter.black.regex=
#以下设置订阅表的哪些字段
# table field filter(format: schema1.tableName1:field1/field2,schema2.tableName2:field1/field2)
#canal.instance.filter.field=test1.t_product:id/subject/keywords,test2.t_company:id/name/contact/ch
# table field black filter(format: schema1.tableName1:field1/field2,schema2.tableName2:field1/field2)
#canal.instance.filter.black.field=test1.t_product:subject/product_image,test2.t_company:id/name/contact/ch

# mq config,如果不配置表同步topic,默认使用example
canal.mq.topic=example
# dynamic topic route by schema or table regex
#canal.mq.dynamicTopic=mytest1.user,mytest2\\..*,.*\\..*
#发送到mq-topic下的哪个分区当中,默认rockemq是4个分区,
#这里表示固定投递到0分区
canal.mq.partition=0
```

```
# hash partition config,采用hash分区投递
#canal.mq.partitionsNum=3
#canal.mq.partitionHash=test.table:id^name,.*\\..*
#################################################
```

canal.instance.filter.regex

监控哪些表的正则配置 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)
```

instance.xml配置文件

目前默认支持的instance.xml有以下几种:

1. spring/memory-instance.xml
2. spring/file-instance.xml
3. spring/default-instance.xml
4. spring/group-instance.xml

memory-instance.xml介绍:

所有的组件(parser , sink , store)都选择了内存版模式，记录位点的都选择了memory模式，重启后又会回到初始位点进行解析

特点：速度最快，依赖最少(不需要zookeeper)

场景：一般应用在quickstart，或者是出现问题后，进行数据分析的场景，不应该将其应用于生产环境

file-instance.xml介绍:

所有的组件(parser , sink , store)都选择了基于file持久化模式，注意，不支持HA机制.

特点：支持单机持久化

场景：生产环境，无HA需求，简单可用.

default-instance.xml介绍：
所有的组件(parser，sink，store)都选择了持久化模式，目前持久化的方式主要是写入zookeeper，保证数据集群共享.
特点：支持HA
场景：生产环境，集群化部署.

group-instance.xml介绍：
主要针对需要进行多库合并时，可以将多个物理instance合并为一个逻辑instance，提供客户端访问。
场景：分库业务。 比如产品数据拆分了4个库，每个库会有一个instance，如果不用group，业务上要消费数据时，需要启动4个客户端，分别链接4个instance实例。使用group后，可以在canal server上合并为一个逻辑instance，只需要启动1个客户端，链接这个逻辑instance即可.

canal启动步骤

```
第1步:
vim conf/canal.properties
# tcp, kafka, RocketMQ,编辑服务模式,同步数据发送到rocketmq
canal.serverMode = RocketMQ
# parallel parser config, 多核cpu设置为true
canal.instance.parser.parallel = true
###########MQ设置#############
canal.mq.servers = 127.0.0.1:9876
canal.mq.retries = 3
 canal.mq.batchSize = 16384
 canal.mq.maxRequestSize = 1048576
 canal.mq.lingerMs = 100
 canal.mq.bufferMemory = 33554432
 canal.mq.canalBatchSize = 50
 canal.mq.canalGetTimeout = 100
 #rocketmq消费者收到的DTO对象,设置为true,则接收FlatMessage
 canal.mq.flatMessage = true
 canal.mq.compressionType = none
 canal.mq.acks = all
```

```
#canal.mq.properties. =
canal.mq.producerGroup = Canal-Producer
# Set this value to "cloud", if you want open message trace feature in aliyun.
canal.mq.accessChannel = LOCAL
# aliyun mq namespace
#canal.mq.namespace =

编辑实例配置
vi conf/example/instance.properties
编辑MySQL用户名与密码
canal.instance.dbUsername=canal
canal.instance.dbPassword=canal
canal.instance.connectionCharset = UTF-8
canal.instance.defaultDatabaseName = micromall

第2步:
配置MySQL，开启binlog
vi /etc/my.cnf #根据自己环境配置
#binlog文件存储位置
log-bin=/usr/local/mysql/data/binlog/mysql-bin
#binlog格式
binlog-format=ROW
#[必须]服务器唯一ID，默认是1
server-id=1

第3步:
#创建用户并赋权
MariaDB [(none)]> CREATE USER 'canal'@'%' IDENTIFIED BY 'root';
#授权
MariaDB [(none)]> grant SELECT, REPLICATION SLAVE, REPLICATION CLIENT on *.* to canal@'%' identified by 'canal';
#刷新权限
MariaDB [(none)]> FLUSH PRIVILEGES;

第4步:
```

```
重启MySQL
service mysqld restart; #mysql需要被注册为服务

第5步：
启动canal
bin/startup.sh

第6步：
检查是否启动成功，查看logs下的日志
canal/canal.log
2020-03-02 22:54:20.874 [main] INFO com.alibaba.otter.canal.deployer.CanalLauncher - ## set default uncaught exception handler
2020-03-02 22:54:20.906 [main] INFO com.alibaba.otter.canal.deployer.CanalLauncher - ## load canal configurations
2020-03-02 22:54:20.924 [main] INFO com.alibaba.otter.canal.deployer.CanalStarter - ## start the canal server.
2020-03-02 22:54:20.960 [main] INFO com.alibaba.otter.canal.deployer.CanalController - ## start the canal server[192.168.122.1(192.168.122.1):11111]
2020-03-02 22:54:22.088 [main] INFO com.alibaba.otter.canal.deployer.CanalStarter - ## the canal server is running now ......
example/example.log
2020-03-02 22:54:21.356 [main] INFO c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer - Loading properties file from class path resource [canal.properties]
2020-03-02 22:54:21.359 [main] INFO c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer - Loading properties file from class path resource [example/instance.properties]
2020-03-02 22:54:21.515 [main] WARN o.s.beans.GenericTypeAwarePropertyDescriptor - Invalid JavaBean property 'connectionCharset' being accessed! Ambiguous write methods found next to actually used [public void com.alibaba.otter.canal.parse.inbound.mysql.AbstractMysqlEventParser.setConnectionCharset(java.nio.charset.Charset)]: [public void com.alibaba.otter.canal.parse.inbound.mysql.AbstractMysqlEventParser.setConnectionCharset(java.lang.String)]
2020-03-02 22:54:21.571 [main] INFO c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer - Loading properties file from class path resource [canal.properties]
2020-03-02 22:54:21.572 [main] INFO c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer - Loading properties file from class path resource [example/instance.properties]
```

```
2020-03-02 22:54:22.034 [main] INFO c.a.otter.canal.instance.spring.CanalInstanceWithSpring - start CannalInstance for 1-example
2020-03-02 22:54:22.049 [main] WARN c.a.o.canal.parse.inbound.mysql.dbsync.LogEventConvert - --> init table filter : ^micromall.ums_order$
2020-03-02 22:54:22.049 [main] WARN c.a.o.canal.parse.inbound.mysql.dbsync.LogEventConvert - --> init table black filter :
2020-03-02 22:54:22.058 [main] INFO c.a.otter.canal.instance.core.AbstractCanalInstance - start successful....
2020-03-02 22:54:22.275 [destination = example , address = /192.168.232.198:3306 , 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
2020-03-02 22:54:22.275 [destination = example , address = /192.168.232.198:3306 , EventParser] WARN c.a.o.c.p.inbound.mysql.rds.RdsBinlogEventParserProxy - prepare to find start position just show master status
2020-03-02 22:54:22.604 [destination = example , address = /192.168.232.198:3306 , EventParser] WARN c.a.o.c.p.inbound.mysql.rds.RdsBinlogEventParserProxy - ---> find start position successfully, EntryPosition[included=false,journalName=mysql-bin.000043,position=4,serverId=1,gtid=<null>,timestamp=1583218298000] cost : 317ms , the next step is binlog dump
日志如上，表示启动成功
```

canal集成MQ

canal 1.1.1版本之后, 默认支持将canal server接收到的binlog数据直接投递到MQ，目前支持kafka、Rocketmq。

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