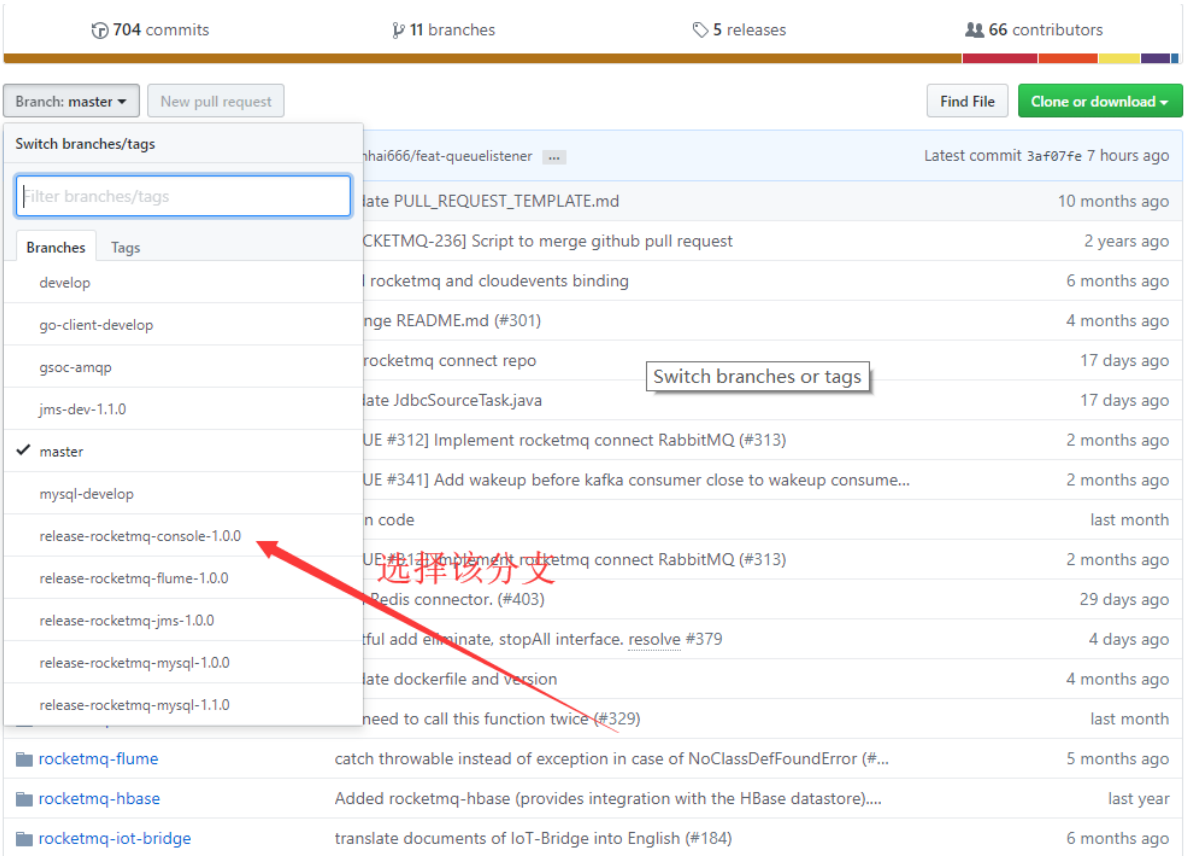
环境搭建

下载地址: https://github.com/apache/rocketmq-externals

console下载地址:https://github.com/apache/rocketmq-externals/tree/release-rocketmq-console-1.0.0

JDK版本：>7



选择分支然后下载，下载的时候建议开启VPN，否则特别慢。

下载成功后，将rocketmq-console项目导入idea，在idea中进行编译

1、启动配置

首先需要配置console的启动端口及nameserver地址，编辑application.properties配置文件

```
rocketmq-console/src/main/resources/application.properties
```

编辑application.properties文件:

```
#项目路径
server.contextPath=
server.port=8080
#spring.application.index=true
spring.application.name=rocketmq-console
```

```
spring.http.encoding.charset=UTF-8
spring.http.encoding.enabled=true
spring.http.encoding.force=true
logging.config=classpath:logback.xml
 #配置rocketmq nameserver的ip及端口，如有多个则用逗号间隔
 rocketmq.config.namesrvAddr=192.168.241.198:9876
 #rocketmq 版本< 3.5.8,设置 false 不设置默认为true
 rocketmq.config.isVIPChannel=
 #数据存放地址,本机地址
 rocketmq.config.dataPath=C:\\temp\\rocketmq-console\\data
 #set it false if you don't want use dashboard.default true
 rocketmq.config.enableDashBoardCollect=true
```

2、修改依赖

修改**pom.xml**文件，修改Rocketmq依赖版本

找到

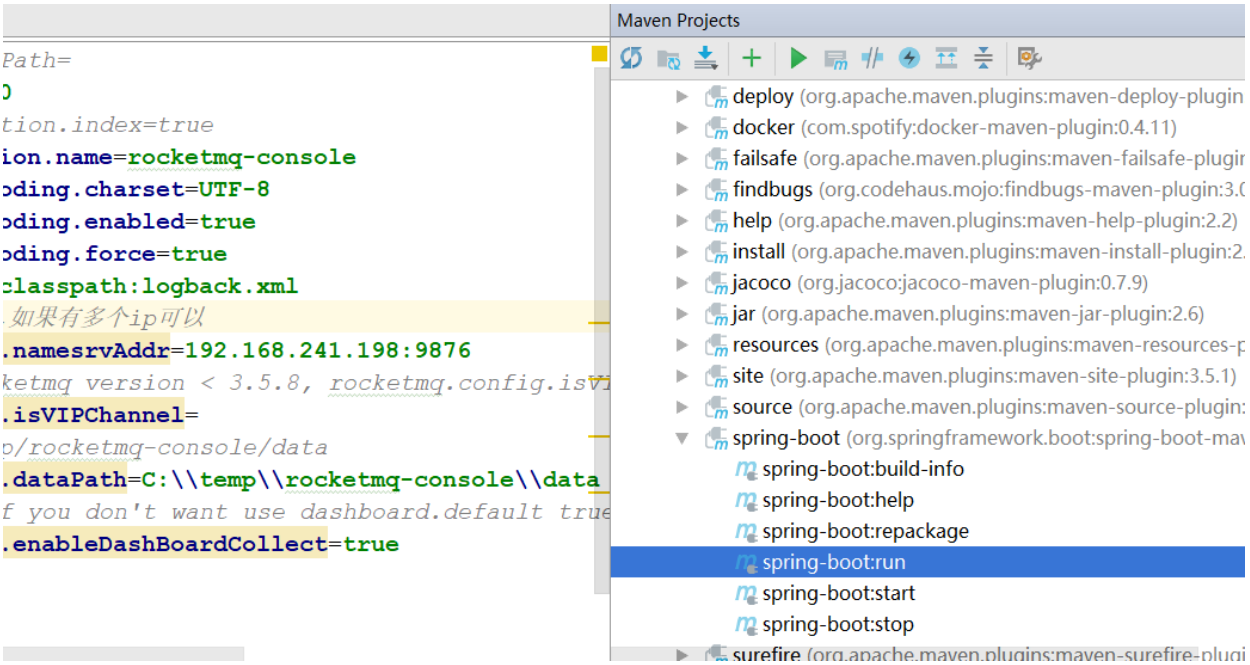
```
<rocketmq.version>4.0.0-incubating</rocketmq.version>
```

修改为与服务端对应的版本，这里我的服务端版本为4.3.2

```
<rocketmq.version>4.3.2</rocketmq.version>
```

如果不是在idea中打开项目，则直接cd回项目目录：cd ${path}/rocketmq-console
执行编译命令：mvn clean package -Dmaven.test.skip=true，此时target文件夹下会生成rocketmq-console-ng-1.0.0.jar，在target目录下，执行java -jar rocketmq-console-ng-1.0.0.jar启动console。

如果是在idea中打开的项目，直接点击**spring-boot:run**



使用文档说明

运维页面

- 你可以修改这个服务使用的navesvr的地址
- 你可以修改这个服务是否使用VIPChannel(如果你的mq server版本小于3.5.8，请设置不使用)

驾驶舱

- 查看broker的消息量（总量/5分钟图）
- 查看单一主题的消息量（总量/趋势图）

集群页面

- 查看集群的分布情况
  - cluster与broker关系
  - broker
- 查看broker具体信息/运行信息
- 查看broker配置信息

主题页面

- 展示所有的主题，可以通过搜索框进行过滤
- 筛选 普通/重试/死信 主题
- 添加/更新主题
  - clusterName 创建在哪几个cluster上
  - brokerName 创建在哪几个broker上
  - topicName 主题名

    - writeQueueNums 写队列数量
    - readQueueNums 读队列数量
    - perm //2是写 4是读 6是读写
- 状态 查询消息投递状态（投递到哪些broker/哪些queue/多少量等）
- 路由 查看消息的路由（现在你发这个主题的消息会发往哪些broker，对应broker的queue信息）
- CONSUMER管理（这个topic都被哪些group消费了，消费情况何如）
- topic配置（查看变更当前的配置）
- 发送消息（向这个主题发送一个测试消息）
- 重置消费位点(分为在线和不在线两种情况，不过都需要检查重置是否成功)
- 删除主题 （会删除掉所有broker以及namesvr上的主题配置和路由信息）

消费者页面

- 展示所有的消费组，可以通过搜索框进行过滤
- 刷新页面/每隔五秒定时刷新页面
- 按照订阅组/数量/TPS/延迟 进行排序
- 添加/更新消费组
    - clusterName 创建在哪几个集群上
    - brokerName 创建在哪几个broker上
    - groupName 消费组名字
    - consumeEnable //是否可以消费 FALSE的话将无法进行消费
    - consumeBroadcastEnable //是否可以广播消费
    - retryQueueNums //重试队列的大小
    - brokerId //正常情况从哪消费
    - whichBrokerWhenConsumeSlowly//出问题了从哪消费
- 终端在线的消费客户端查看，包括版本订阅信息和消费模式
- 消费详情对应消费组的消费明细查看，这个消费组订阅的所有Topic的消费情况，每个queue对应的消费client查看（包括Retry消息）
- 配置 查看变更消费组的配置
- 删除 在指定的broker上删除消费组

发布管理页面

- 通过Topic和Group查询在线的消息生产者客户端
    - 信息包含客户端主机 版本

消息查询页面

- 根据Topic和时间区间查询 *由于数据量大 最多只会展示2000条，多的会被忽略
- 根据Topic和Key进行查询
    - 最多只会展示64条
- 根据消息主题和消息Id进行消息的查询
- 消息详情可以展示这条消息的详细信息，查看消息对应到具体消费组的消费情况（如果异常，可以查看具体的异常信息）。可以向指定的消费组重发消息。