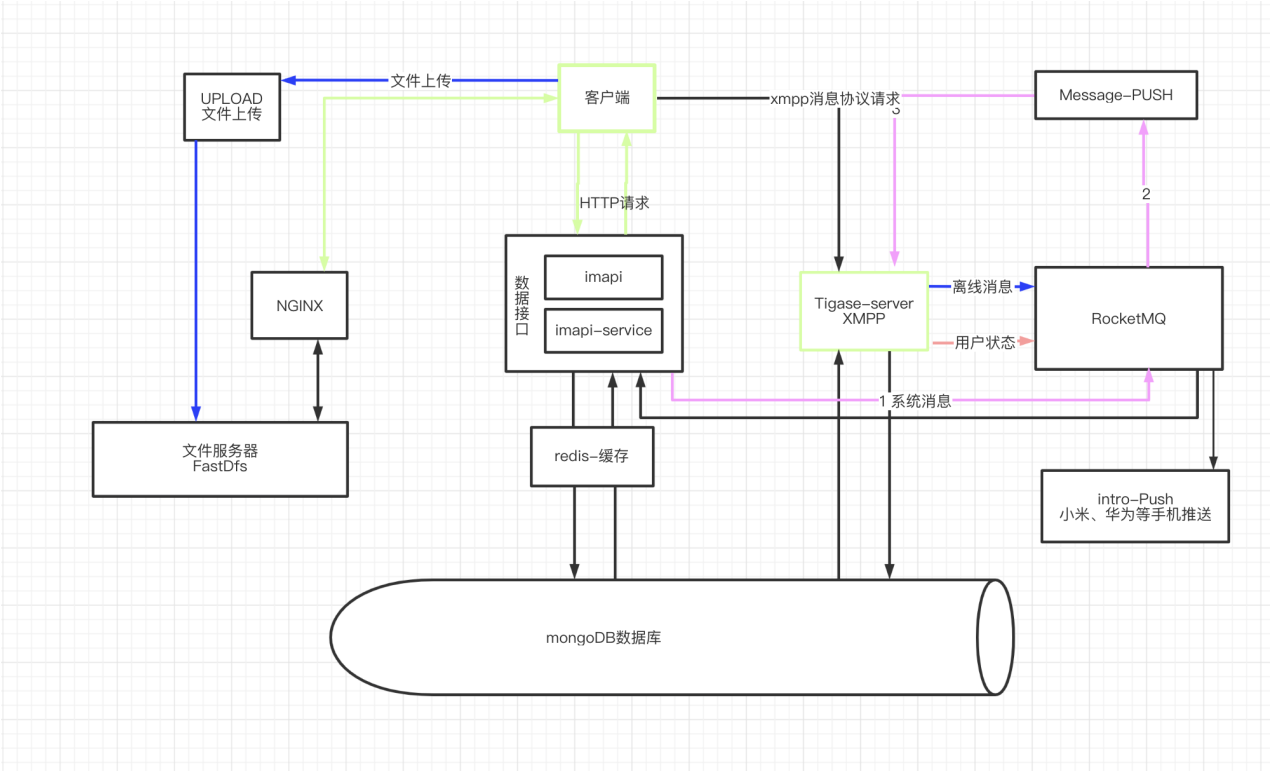
服务端安装部署文档

> 服务端安装方式分为两种，一是linux上面直接部署，二是使用docker容器化部署方式，安装部署文档涉及到的软件包，都在压缩包soft.zip下，软件包找相关项目负责人获取。

> 1、端口开放情况

服务	用途	使用端口	说明
Mongodb	数据库	27017	不用对外开放
imapi	Api接口	8092	需要开放
Tigase-server	通讯服务	Web端使用5260端口，手机端使用5666端口	需要开放
Upload	文件上传	8088	需要开放
Nginx	静态文件访问	8080	需要开放
Redis	缓存	6379	不用对外开放

> 2、产品架构图



一、LINUX 直接安装

> 软件包跟目录放在/opt 下

1、源码安装mongoDB

1.1 解压源码

```
[root@mac-pro~]# cd  /opt
[root@mac-pro~]# mv /soft/mongodb-linux-x86_64-3.4.0.tgz ./
[root@mac-pro~]# tar  -zxvf  mongodb-linux-x86_64-3.4.0.tgz
[root@mac-pro~]# mv  mongodb-linux-x86_64-3.4.0  mongodb-3.4.0
```

1.2 在/opt/mongodb-3.4.0目录下创建mongo.conf文件内容如下：

```
[root@mac-pro~]# cd   mongodb-3.4.0
[root@mac-pro~]# vim  mongo.conf
#录入以下内容
systemLog:
   destination: file
   path: "/opt/mongodb-3.4.0/logs/mongodb.log"
   logAppend: true
storage:
   dbPath: "/data/mongodb"
   journal:
      enabled: true
   mmapv1:
     smallFiles: true
   wiredTiger:
      engineConfig:
        configString: cache_size=1G
processManagement:
      fork: true
net:
   # bindIp: 127.0.0.1
   port: 27017
setParameter:
   enableLocalhostAuthBypass: false
```

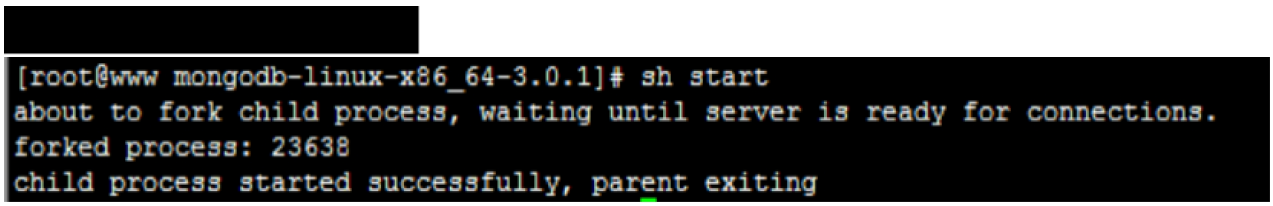
1.3 然后创建mongodb数据目录，和日志目录

```
[root@mac-pro~]# mkdir  -p  /data/mongodb
[root@mac-pro~]# mkdir  logs
```

1.4在/opt/mongodb-3.4.0目录下创建start启动脚本内容如下：

`/opt/mongodb-3.4.0/bin/mongod --config=/opt/mongodb-3.4.0/mongo.conf`

执行start脚本，出现如下图所示内容则启动成功




```
Stop脚本：
ps -ef|grep mongo.conf|grep -v grep|awk '{printf $2}'|xargs kill -9
```

2、安装Redis

2.1 解压、安装

```
[root@mac-pro~]# cd  /opt
[root@mac-pro~]# tar  -zxvf  redis-4.0.1.tar.gz
[root@mac-pro~]# cd  redis-4.0.1
[root@mac-pro~]# make && make  install
```

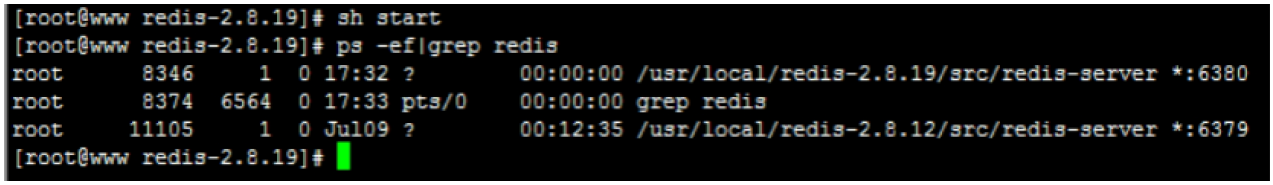
2.2 修改/opt/redis-4.0.1目录下redis.conf文件中配置项：

`daemonize yes（进程后台运行）`

2.3 在/opt/redis-4.0.1目录下创建start启动脚本内容如下：

`/opt/redis-4.0.1/src/redis-server  /opt/redis-4.0.1/redis.conf`

2.4 执行sh start 命令启动脚本，查看redis是否启动成功



2.5 Stop脚本

```
ps -ef|grep  /opt/redis-4.0.1/src/redis-server|grep -v grep|awk '{printf $2}'|xargs
kill -9
ps -ef|grep  /opt/redis-4.0.1/src/redis-server
```

3、安装Jdk1.8+

```
[root@mac-pro~]# cd  /opt
[root@mac-pro~]# tar -zxvf  jdk-8u131-linux-x64.tar.gz
[root@mac-pro~]# mkdir  java
[root@mac-pro~]# mv  jdk1.8.0_131  ./java
```

3.1 设置jdk环境变量

这里采用全局设置方法，就是修改etc/profile，它是是所有用户的共用的环境变量

`[root@mac-pro~]# vim  /etc/profile`

打开之后在末尾添加

```
JAVA_HOME=/opt/java/jdk1.8.0_131
JRE_HOME=/opt/java/jdk1.8.0_131/jre
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
PATH=$JAVA_HOME/bin:$PATH
export PATH JAVA_HOME CLASSPATH
```

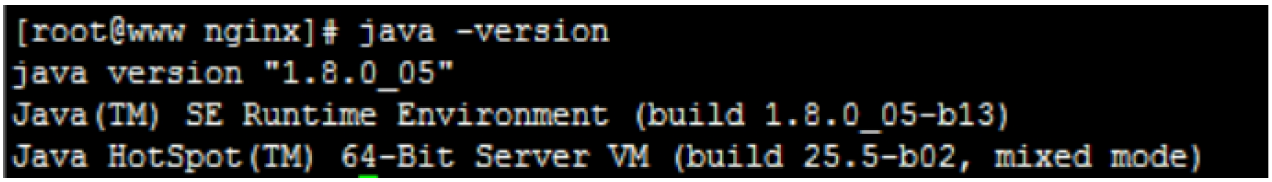
3.2 使环境变量生效

`[root@mac-pro~]# source /etc/profile`

3.3 检验是否安装成功

在终端执行：`java  -version` 命令，看看是否安装成功，成功则显示如下:

```
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b17)
Java HotSpot(TM) 64-Bit Server VM (build 25.65-b01, mixed mode)
```



3.4 可能出现的问题：

若出现

`bash: /usr/bin/java: /lib/ld-linux.so.2: bad ELF interpreter: No such file or directory`

执行：`sudo yum install glibc.i686` 命令安装glibc

4、安装Rocket MQ 队列

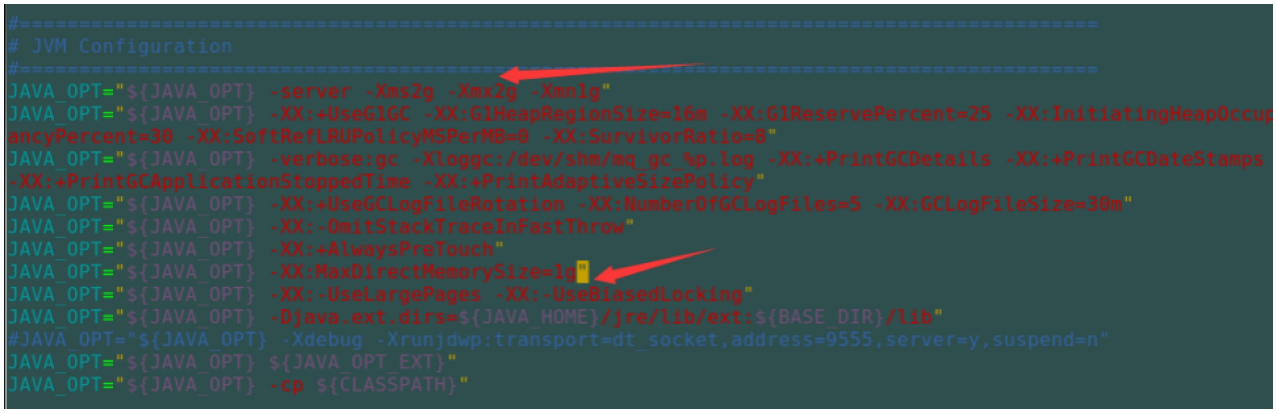
4.1 解压源码

```
[root@mac-pro~]# cd  /opt
[root@mac-pro~]# unzip  rocketmq-all-4.3.2-bin-release.zip
[root@mac-pro~]# mv  rocketmq-all-4.3.2-bin-release rocketmq-4.3.2
[root@mac-pro~]# cd  rocketmq-4.3.2
```

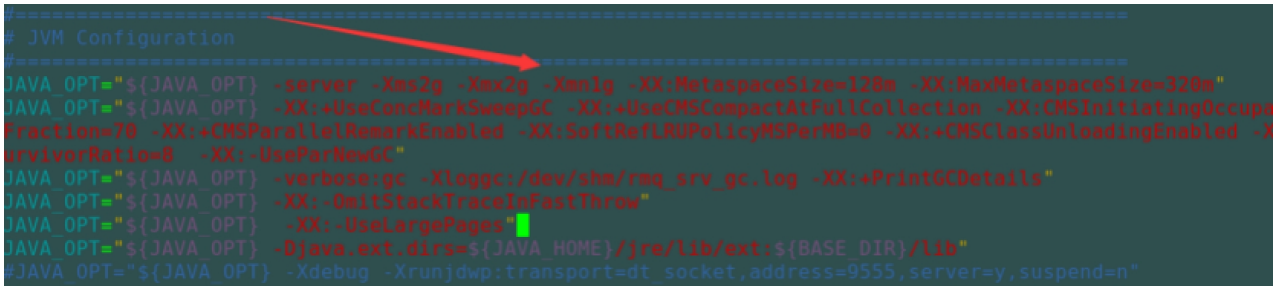
4.2 调整rocketMq 的内存值

注意：这里请根据机器的实际情况进行调整，如机器内存较大可适当配置高一点

`[root@mac-pro~]# vim  bin/runbroker.sh`



`[root@mac-pro~]# vim bin/runserver.sh`



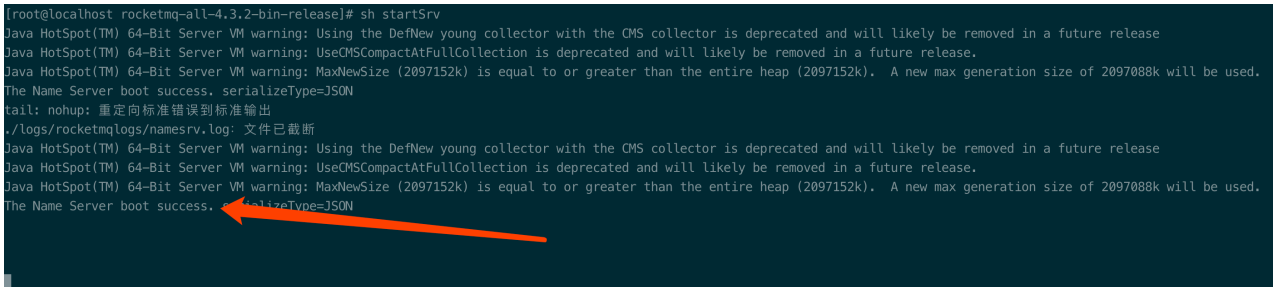
4.3 执行编写启动脚本 startSrv 并启动nameServer

`[root@mac-pro~]# vim startSrv`

写入如下内容：

```
nohup sh ./bin/mqnamesrv  >  ./logs/rocketmqlogs/namesrv.log  &
tail  -f  ./logs/rocketmqlogs/namesrv.log
```

`sh  startSrv`



4.4 执行编写启动脚本 startBroker 并启动Broker

`vim startBroker`

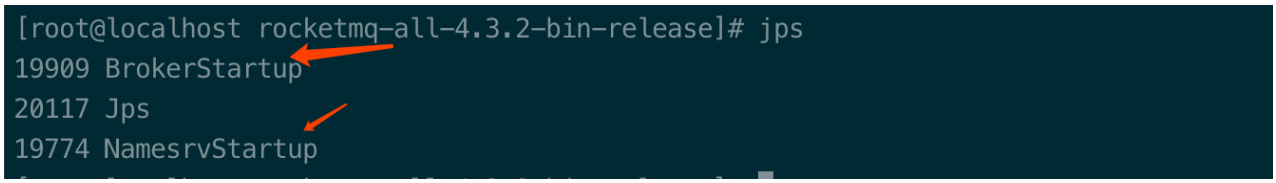
写入如下内容：

```
nohup sh bin/mqbroker -n  localhost:9876  >  ./logs/rocketmqlogs/broker.log  &
tail  -f  ./logs/rocketmqlogs/broker.log
```

执行startBroker脚本启动borker

`sh  startBroker`

执行jps 命令 查看正常应该能看到NamesrvStaup 和 BrokerStartup



4.4 注册推送消息、用户状态话题

```
sh  bin/mqadmin updateTopic -n localhost:9876  -c DefaultCluster  -t  pushMessage
sh  bin/mqadmin updateTopic -n localhost:9876  -c DefaultCluster  -t  xmppMessage
sh  bin/mqadmin updateTopic -n localhost:9876  -c DefaultCluster  -t
userStatusMessage
sh  bin/mqadmin updateTopic -n localhost:9876  -c DefaultCluster  -t HWPushMessage
sh  bin/mqadmin updateTopic -n localhost:9876  -c DefaultCluster -t  fullPushMessage
```

附： 单个脚本启动

`vim vim mqStart`

写入如下内容：

```
#!/bin/sh
export JAVA_HOME=/opt/java/jdk1.8.0_131/
nohup sh /opt/rocketmq-4.3.2/bin/mqnamesrv > /opt/rocketmq-
4.3.2/logs/rocketmqlogs/namesrv.log  2>&1 &
echo "Start Name Server End"
nohup sh /opt/rocketmq-4.3.2/bin/mqbroker -n localhost:9876 > /opt/rocketmq-
4.3.2/logs/rocketmqlogs/broker.log  2>&1  &
echo "Start  Broker End"
```

附： 单个脚本停止

`vim mqStop`

```
#!/bin/sh
sh /opt/rocketmq-4.3.2/bin/mqshutdown broker &
sh /opt/rocketmq-4.3.2/bin/mqshutdown namesrv &
echo "Please wait process to exit! check it type jps"
```

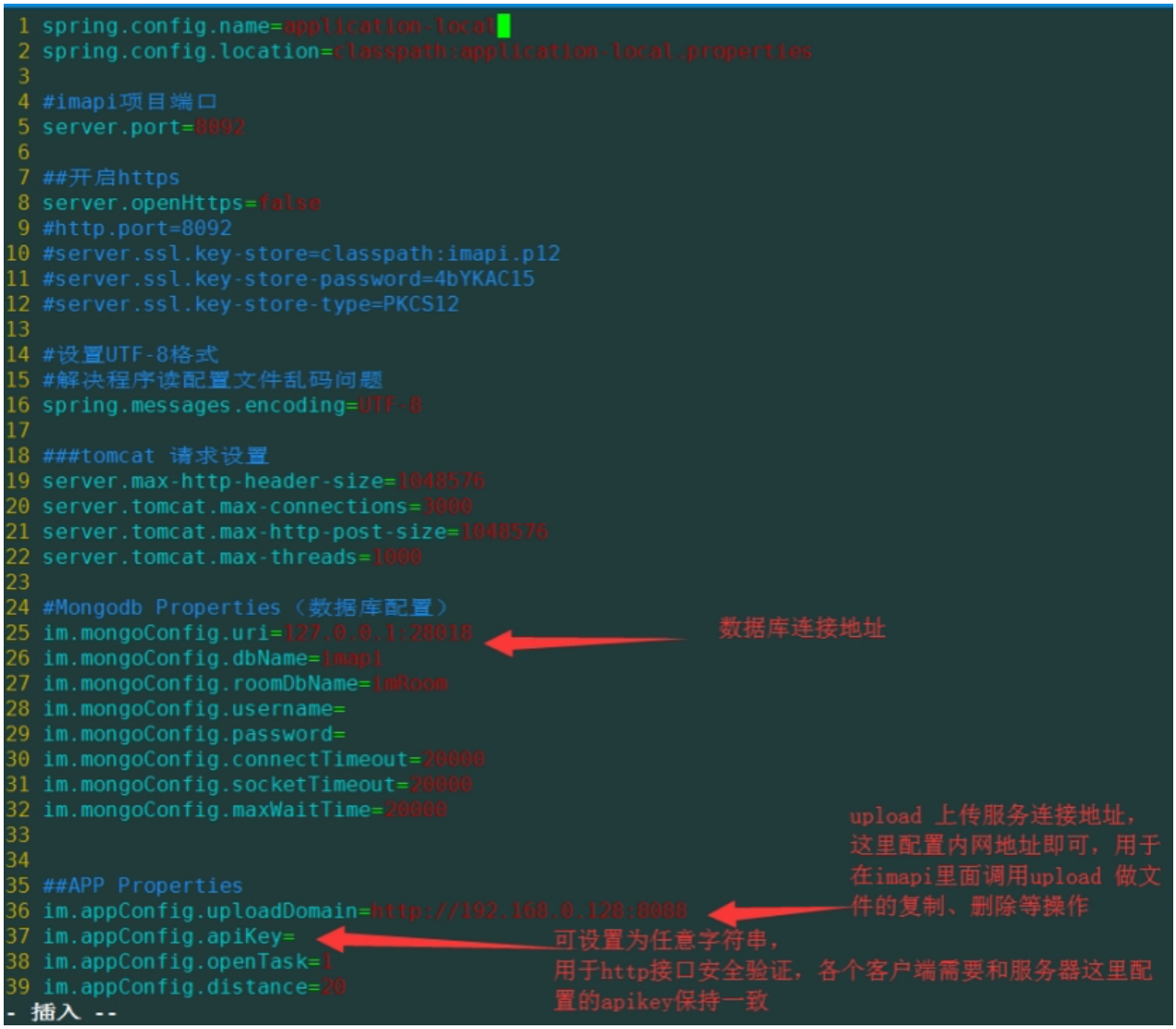
附：停止命令

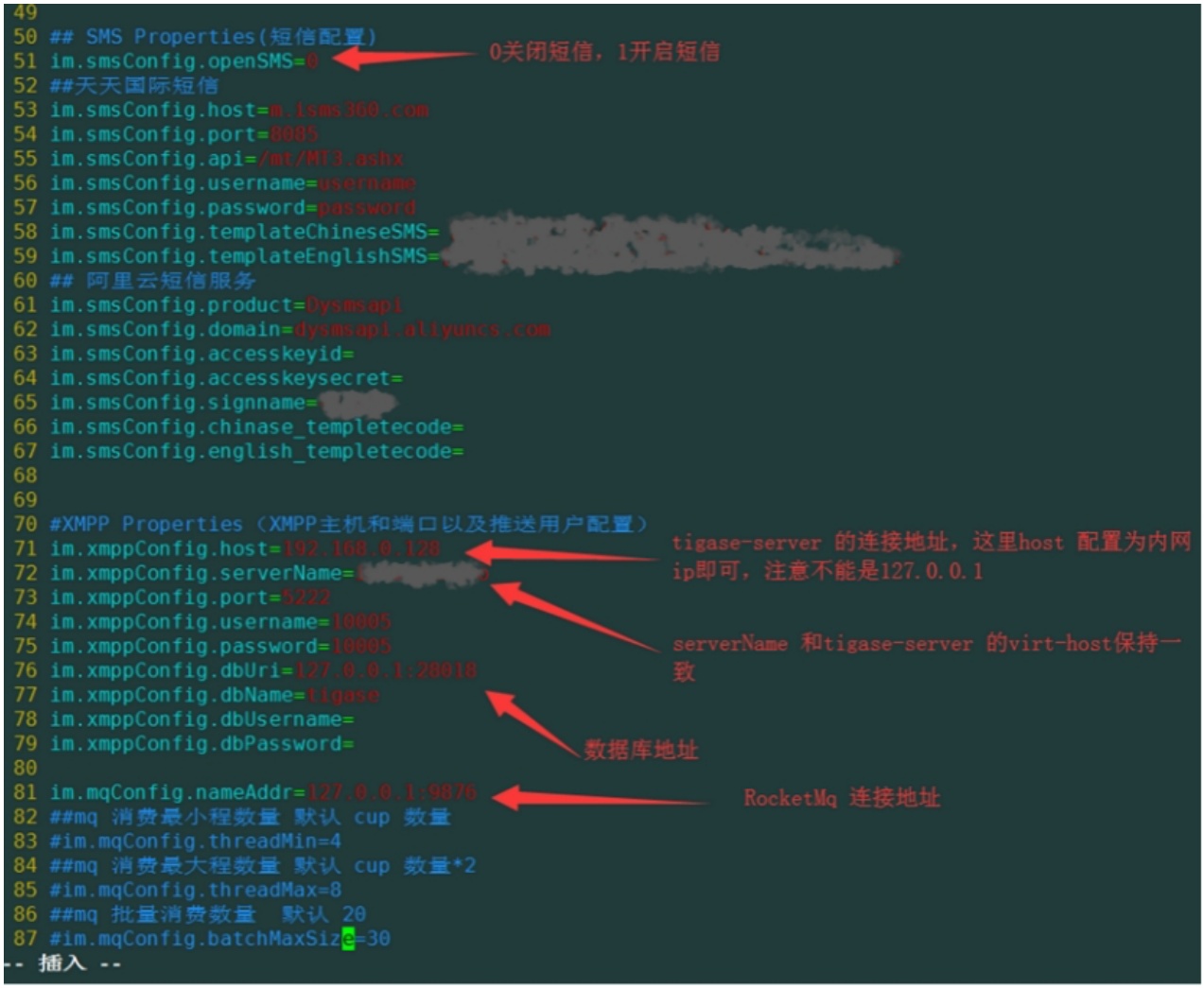
```
sh bin/mqshutdown namesrv
sh bin/mqshutdown broker
```

5 安装imapi api 接口服务

```
[root@mac-pro~]# cd  /opt
[root@mac-pro~]# tar  -xvf  imapi.tar
[root@mac-pro~]# cd  imapi
[root@mac-pro~]# vim  application.properties
```

5.1 修改application.properties配置文件


spring.config.name=application-local
spring.config.location=classpath:application-local.properties
#imapi项目端口
server.port=8092
##开启https
server.openHttps=false
#http.port=8092
#server.ssl.key-store=classpath:imapi.p12
#server.ssl.key-store-password=4bYKAC15
#server.ssl.key-store-type=PKCS12
#设置UTF-8格式
#解决程序读配置文件乱码问题
spring.messages.encoding=UTF-8
###tomcat 请求设置
server.max-http-header-size=1048576
server.tomcat.max-connections=3000
server.tomcat.max-http-post-size=1048576
server.tomcat.max-threads=1000
#Mongodb Properties（数据库配置）
im.mongoConfig.uri=127.0.0.1:28018
im.mongoConfig.dbName=imapi
im.mongoConfig.roomDbName=imRoom
im.mongoConfig.username=
im.mongoConfig.password=
im.mongoConfig.connectTimeout=20000
im.mongoConfig.socketTimeout=20000
im.mongoConfig.maxWaitTime=20000
##APP Properties
im.appConfig.uploadDomain=http://192.168.0.128:8088
im.appConfig.apiKey=
im.appConfig.openTask=1
im.appConfig.distance=20
-- 插入 --
数据库连接地址
upload 上传服务连接地址，这里配置内网地址即可，用于在imapi里面调用upload 做文件的复制、删除等操作
可设置为任意字符串，用于http接口安全验证，各个客户端需要和服务器这里配置的apikey保持一致



5.2 在imapi目录下执行sh start命令运行imapi接口服务

```
sh start
```

5.3 修改后台配置

启动完成后，在浏览器打开链接"http://localhost:8092/console/login",出现如下图所示内容即酷信接口部署成功

注：超级管理员账号：1000 初始密码：1000

6 安装tigase-server 通讯服务

6.1 包目录说明

Tigase-Server-聊天服务器

> lib 目录为 jar包存放目录 etc/init.properties 为服务器配置文件

```
[root@mac-pro~]# cd  /opt
[root@mac-pro~]# tar -zxvf tigase-server-7.1.3-b4482.tar
[root@mac-pro~]# cd tigase-server-7.1.3-b4482
[root@mac-pro~]# vim etc/init.properties
```

6.2 修改init.properties 配置文件

将172.18.1.232 替换成相应服务的内网地址。

```
config-type=--gen-config-def
# --admins=admin@172.26.46.123,10005@172.26.46.123
--admins=admin@172.18.1.232
--virt-hosts=172.18.1.232
#--debug=server,xmpp,sdy,db,net

# 集群配置
#--cluster-mode=true
#--cluster-nodes=tigase-1:5277,tigase-2:5277
#--cluster-connect-all=true
#--c2s-ports=5223

--user-db=tigase.mongodb.MongoRepository
--auth-db=tigase.mongodb.MongoRepository
--user-db-uri=mongodb://172.18.1.232/tigase?authSource=admin

--redis-uri=redis://172.18.1.232:6379
--redis-password=ycim.redis
--redis-database=0

--net-buff-high-throughput = 256k
#--max-queue-size=15000
--cm-ht-traffic-throttling = xmpp:0:0:disc,bin:0:0:disc
--cm-traffic-throttling = xmpp:0:0:disc,bin:0:0:disc
--net-buff-standard = 64k
--new-connections-throttling = 5222:1000
--elements-number-limit = 100000
sess-man/auth-timeout[L]=45

##imapi 的数据库链接 修改用户在线状态是用到
##mongodb://sysop:moon@localhost
--api-db-uri=mongodb://172.18.1.232:27017/imapi?authSource=admin

#同步用户在线状态
--sm-cluster-strategy-class=tigase.server.cluster.strategy.OnlineUsersCachingStrategy
##关键词过滤功能   1：打开  0：关闭
--confirm-open-keyword =1
"init.properties" [dos] 113L, 4104C
```

> serverName是用户域，有几个地方都要保持一致 需要跟访问的地址一样（即是外网地址）。
>
> tigase的--virt-hosts
>
> imapi的im.xmppConfig.serverName
>
> xmpp-push的im.xmppConfig.serverName
>
> 后台管理的xmpp虚拟域名

6.3 JVM、GC设置和建议

6.3.1 修改 etc/tigase.conf 配置文件

> 对于非生产部署（开发或说明环境），我们建议使用JVM的默认内存设置（这取决于底层操作系统），这会导致自动内存分配，并且根据经验，这是最安全的。环境。对于生产环境，我们建议使用固定大小的HEAP - 初始和最大大小，可以分别设置（JVM ）-Xms和-XmxJVM标志

6.3.2 服务器类机器（非VM）> 16GB > = 8核CPU

建议启用CMS垃圾收集器。根据实际物理内存大小调整Xms和Xms大小以获得实际可用内存 使用以下内容：

```
GC="-XX:+UseBiasedLocking -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=2 -
XX:+CMSIncrementalMode -XX:-ReduceInitialCardMarks -
XX:CMSInitiatingOccupancyFraction=70 -XX:+UseCMSInitiatingOccupancyOnly"
EX="-XX:+OptimizeStringConcat -XX:+DoEscapeAnalysis -XX:+UseNUMA"
HEAP=" -Xms10G -Xmx10G"
JAVA_OPTIONS="${GC} ${GC_DEBUG} ${EX} ${HEAP}"
```

6.3.3 对于具有大量可用内存的服务器，使用G1GC收集器可能是一个更好的主意

```
OSGI=${osgiEnabled}
ENC="-Dfile.encoding=UTF-8 -Dsun.jnu.encoding=UTF-8"
GC="-XX:+UseG1GC -XX:ConcGCThreads=4 -XX:G1HeapRegionSize=2 -
XX:InitiatingHeapOccupancyPercent=35 -XX:MaxGCPauseMillis=100"
EX="-XX:+OptimizeStringConcat -XX:+DoEscapeAnalysis -XX:+UseNUMA"
HEAP="-Xms60G -Xmx60G"
JAVA_OPTIONS="${GC} ${GC_DEBUG} ${EX} ${HEAP}"
```

6.3.4 8GB RAM，4核CPU等效

建议启用CMS垃圾收集器。必须配置NewRatio。需要调整Xms和Xms大小以获得实际可用内存。应该使用以下内容：

```
GC="-XX:+UseBiasedLocking -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=2 -
XX:+CMSIncrementalMode -XX:-ReduceInitialCardMarks -
XX:CMSInitiatingOccupancyFraction=70 -XX:+UseCMSInitiatingOccupancyOnly"
EX="-XX:+OptimizeStringConcat -XX:+DoEscapeAnalysis -XX:+UseNUMA"
HEAP="-Xms7G -Xmx7G" # heap memory settings must be adjusted on per deployment-base!
JAVA_OPTIONS="${GC} ${GC_DEBUG} ${EX} ${HEAP}"
nohup java -Djava.ext.dirs="lib" ${JAVA_OPTIONS}  com.mac-pro.imserver.IMServerStarter
imserver.properties > log.log &
```

6.4 在IMServer 执行 start 启动 服务

`sh start`

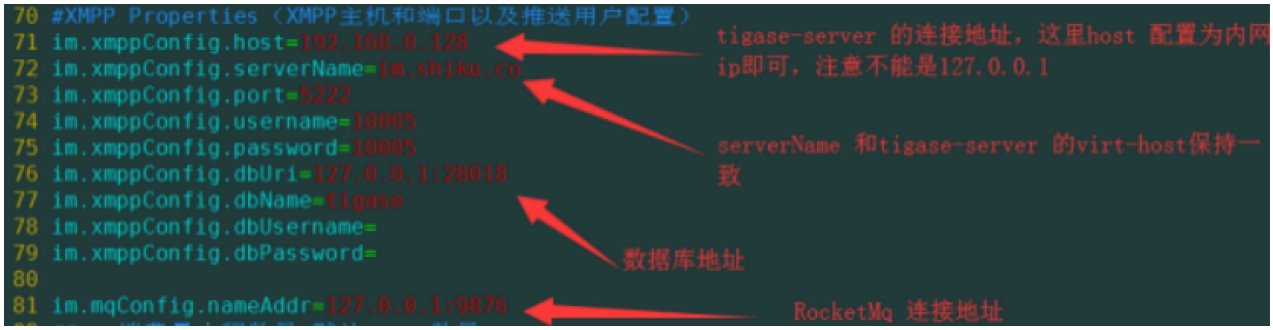
7、push 推送服务

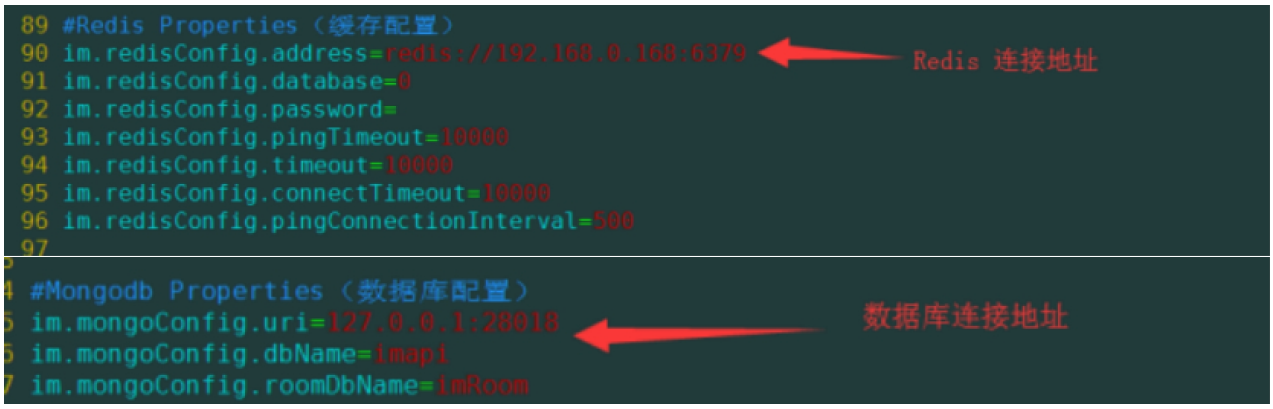
push服务为离线消息推送服务，去消费tigase-server 放到RocketMq 队列里的离线消息

7.1 安装push 推送服务

```
[root@mac-pro~]# tar  -xvf  push.tar
[root@mac-pro~]# cd  push
[root@mac-pro~]# vim  application.properties
```

7.2 修改application.properties 配置文件





> 目前ios 支持anps和极光推送 ，安卓目前集成了华为、小米、魅族、vovo、oppo，谷歌六种推送，如果手机有谷歌框架且能访问外网，会使用谷歌推送，否则根据机型来，非以上机型使用小米推送如已经申请好相关配置，将申请好的小米、华为、极光 等平台的key secret 配置到对应的位置。也可以后续在配置，加上配置重启push 服务即可

```
# 消息推送相关配置（小米、华为、魅族等）
im.pushConfig.packageName=com.sk.weichat

# 小米推送
im.pushConfig.xm_appSecret=wet...

# 华为推送
im.pushConfig.hw_appSecret=3b6...
im.pushConfig.hw_appId=100...
im.pushConfig.hw_tokenUrl=https://login.cloud.huawei.com/oauth2/v2/token
im.pushConfig.hw_apiUrl=https://api.push.hicloud.com/pushsend.do
im.pushConfig.hw_iconUrl=http://pic.qiantucdn.com/58pic/12/38/18/13758PIC4GV.jpg

##ios 推送
im.pushConfig.betaAppId=com.shiku.coolim.push1
im.pushConfig.betaApnsPk=/opt/shiku-push/apns_push1_pro.p12

im.pushConfig.appStoreAppId=com.shiku.im.push
im.pushConfig.appStoreApnsPk=/opt/shiku-push/apns_pro.p12

im.pushConfig.voipPk=/opt/shiku-push/voippush.p12

im.pushConfig.pkPassword=123...

im.pushConfig.isApnsSandbox=0

im.pushConfig.isDebug=1


# 极光推送
im.pushConfig.jPush_appkey=691...
im.pushConfig.jPush_masterSecret=e59...

# google FCM 推送
im.pushConfig.FCM_dataBaseUrl=https://sixth-hawk-164509.firebaseio.com
im.pushConfig.FCM_keyJson=/opt/shiku-push/sixth-hawk-164509-firebase-adminsdk-342gn-465351f0ef.json

#魅族推送
im.pushConfig.mz_appId=118...
im.pushConfig.mz_appSecret=15c...

# VIVO推送
im.pushConfig.vivo_appId=109...
im.pushConfig.vivo_appKey=472...
im.pushConfig.vivo_appSecret=127...
# OPPO推送
im.pushConfig.oppo_appKey=dIH...
im.pushConfig.oppo_masterSecret=Bb3...
```

> ios apns 推送配置
> 替换ios apns推送证书，个人版和企业版选择一个使用即可，需要注意在申请ios apns 证书的时候*必须要设置密码*。

7.3.修改完配置后使用 sh start 命令启动push服务

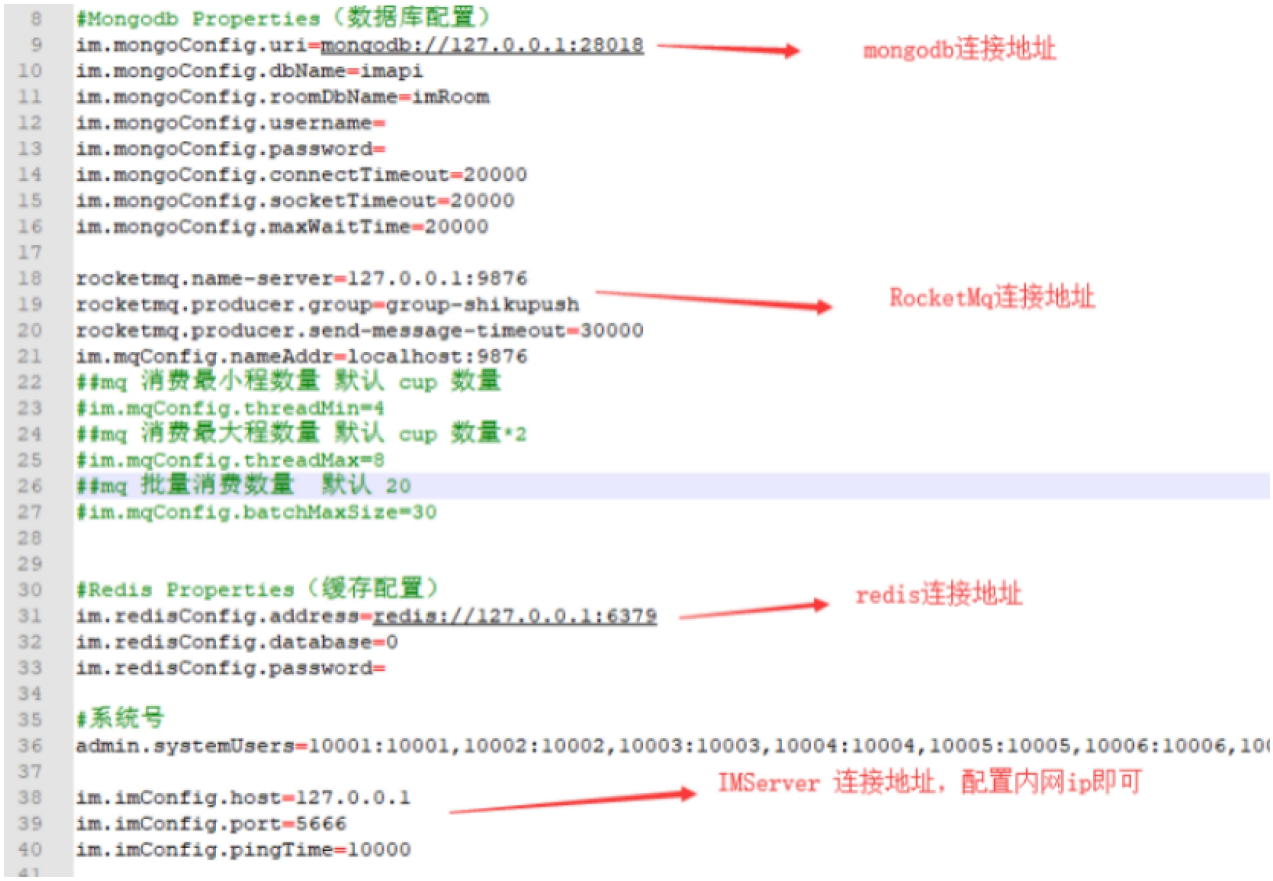
`sh start`

8、安装xmpp-push服务

> 说明：xmpp-push 是从 imapi 中独立出来的服务，用于发送系统Xmpp 消息Imapi 服务里面群组等部分接口操作后，需要发一条xmpp 消息通知用户，此时imapi 将需要发送的xmpp 消息放到rocketMq 队列下，由xmpp-push 服务获取队列里的消息然后把消息经过tigage-server 发送到设备端

8.1 安装 xmpp-push 服务并 参照如下示例，修改配置文件

```
[root@mac-pro~]# cd  /opt
[root@mac-pro~]# tar -xvf xmpp-push.tar
[root@mac-pro~]# cd  xmpp-push
[root@mac-pro~]# vim  application.properties
```



8.2 启动 xmpp-push 服务

`[root@mac-pro~]#sh start`

9、安装Upload文件上传服务

9.1 安装Upload服务

```
[root@mac-pro~]# cd  /opt
[root@mac-pro~]# tar -xvf upload.tar
[root@mac-pro~]# cd  upload
[root@mac-pro~]# vim  application.properties
```

9.2 参照如下示例修改配置：

说明：/data/www/resources：文件上传成功后存储的根目录 beginIndex：根目录“/data/www/resources”的字符串长度（从0开始）用于分隔字符串生成文件链接用（需要注意，如果修改了相关路径，beginIndex 的数值也要相应修改）

```
spring.mvc.view.prefix=/
spring.mvc.view.suffix=.jsp
server.context-path=/upload

server.port=8088

##https
server.openHttps=false
#http.port=8080
#server.ssl.key-store=/opt/upload/imshiku.pfx
#server.ssl.key-store-password=
#server.ssl.key-store-type=PKCS12

domain=http://im.server.com:8089
isBackDomain=1

##是否 把文件上传到 fastdfs  # 文件系统中
isOpenfastDFS=0
fastdfsDomain=http://im.server.com:8089
fastdfsBasePath=/data/fastdfs
##保存文件的数据库 url
## 只修改   mongodb://127.0.0.1:28018
##  /resources  不需要修改
dbUri=mongodb://127.0.0.1:28018/resources

##开启定时任务   删除文件
##  0  关闭     1 开启
## 在部署 多个 upload 项目的情况下
##只需要 一个 项目 执行定时任务就可以了
openTask=1

basePath=/data/www/resources
nTemp=/data/www/resources/%1$s/%2$s/
oTemp=/data/www/resources/%1$s/%2$s/
tTemp=/data/www/resources/%1$s/%2$s/
uTemp=/data/www/resources/temp/
beginIndex=19
#是否将amr编码为mp3: 0=否; 1=是
amr2mp3=0
```

文件访问地址
上传后用该地址拼接文件路径
得到完整url 返回给客户端

是否启用fastDfs

fastDfs 模式下文件访问地址

fastDfs 基础存储路径

数据库连接地址

```
imageFilter = gif|jpg|jpeg|bmp|png|
videoFilter = mp4|flv|wmv|
audioFilter = acc|amr|mp3|

##fastdfs 的配置
########
connect_timeout = 2
network_timeout = 30
charset = UTF-8
http.tracker_http_port = 80
http.anti_steal_token = no
http.secret_key = FastDFS1234567890
tracker_server= 192.168.0.155:22122,192.168.0.156:22122
```

fastDfs tracker_server 连接地址

9.3 在文件上传服务所在机器创建存储目录（例如“/data/www/resources”）并初始化目录结构

可以直接将以下命令全部拷贝到Linux服务器一次性执行

```
mkdir  -p  /data/www/resources
cd /data/www/resources
mkdir audio
mkdir avatar
mkdir avatar/o
mkdir avatar/t
mkdir avatar_r
mkdir avatar_r/o
mkdir avatar_r/t
mkdir gift
mkdir image
mkdir image/o
mkdir image/t
mkdir other
mkdir preview
mkdir temp
mkdir u
mkdir video
```

9.4 启动 upload

```
[root@mac-pro~]# cd  /opt/upload
[root@mac-pro~]# sh  start
```

```
 FastDFS config ======================>
{
  g_connect_timeout(ms) = 2000
  g_network_timeout(ms) = 30000
  g_charset = UTF-8
  g_anti_steal_token = false
  g_secret_key = FastDFS1234567890
  g_tracker_http_port = 80
  trackerServers = 172.31.77.83:22122
}

============================> 上传服务启动成功 请放心使用  ===========>
```

10、安装FastDFS分布式文件存储服务

10、安装Nginx，配置文件访问

10.1 首先安装 nginx 所需的依赖

```
 [root@mac-pro~]# yum install -y  pcre  pcre-devel  zlib  zlib-devel openssl  openssl-devel
```

10.2 安装 Nginx-1.9.11

```
[root@mac-pro~]# cd /opt
[root@mac-pro~]# tar  -zxvf  nginx-1.9.11.tar.gz
[root@mac-pro~]# mv  nginx-1.9.11  nginx-install
[root@mac-pro~]# cd nginx-install
[root@mac-pro~]# ./configure --prefix=/opt/nginx-1.9.11  --with-http_ssl_module  --with-stream
[root@mac-pro~]# make && make install
[root@mac-pro~]# rm  -rf  ../nginx-install
```

10.3在/opt/nginx-1.9.11目录下创建start、stop脚本：

vim start , 写入如下内容

```
./sbin/nginx
ps -ef|grep nginx
```

vim stop , 写入如下内容

```
./sbin/nginx -s stop
ps -ef|grep nginx
```

10.4 在upload 所在的机器上 配置头像访问。首先请确保已经安装 Nginx，如没有安装请参考上面的 安装 Nginx-1.9.11中的步骤进行安装。

说明：由于FastDfs 不支持对上传文件的自定义命名，而目前用户头像是根据userId 命名的，所以目前用户头像没有使用FastDfs 储存。

10.5 修改Nginx 配置文件 nginx.conf ，添加如下内容

```
#用户头像访问
server{
  listen       80;
  server_name    head.mac-pro.co;
  #拒接访问html 等类型的文件避免受到脚本攻击
```

```
  location ~* /\.(html|htm|jsp|php|js)$ {
     deny  all;
  }

  location /{
      root     /data/www/resources;
      expires   4d;
  }
}
#ip 配置示例
server{
     listen       8080;
     #本机外网ip
     server_name    192.168.0.168;
     #拒接访问html 等类型的文件避免受到脚本攻击
     location ~ /\.(html|htm|jsp) {
        deny  all;
   }

    location ~* /{
      root   /data/www/resources;
      expires    4d;
    }
}
```

如按IP方式配置即得到文件访问、下载路径 为 ip:8080

3) 执行start、stop脚本，查看nginx是否启动、停止成功：

```
[root@www nginx]# sh start
root      9803     1  0 17:48 ?        00:00:00 nginx: master process ./sbin/nginx
root      9805  9801  0 17:48 pts/0    00:00:00 grep nginx
nobody    9806  9803  0 17:48 ?        00:00:00 nginx: worker process
nobody    9807  9803  0 17:48 ?        00:00:00 nginx: worker process
nobody    9808  9803  0 17:48 ?        00:00:00 nginx: worker process
nobody    9809  9803  0 17:48 ?        00:00:00 nginx: worker process
nobody    9810  9803  0 17:48 ?        00:00:00 nginx: worker process
nobody    9811  9803  0 17:48 ?        00:00:00 nginx: worker process
[root@www nginx]# sh stop
root      9851  9848  0 17:48 pts/0    00:00:00 grep nginx
```

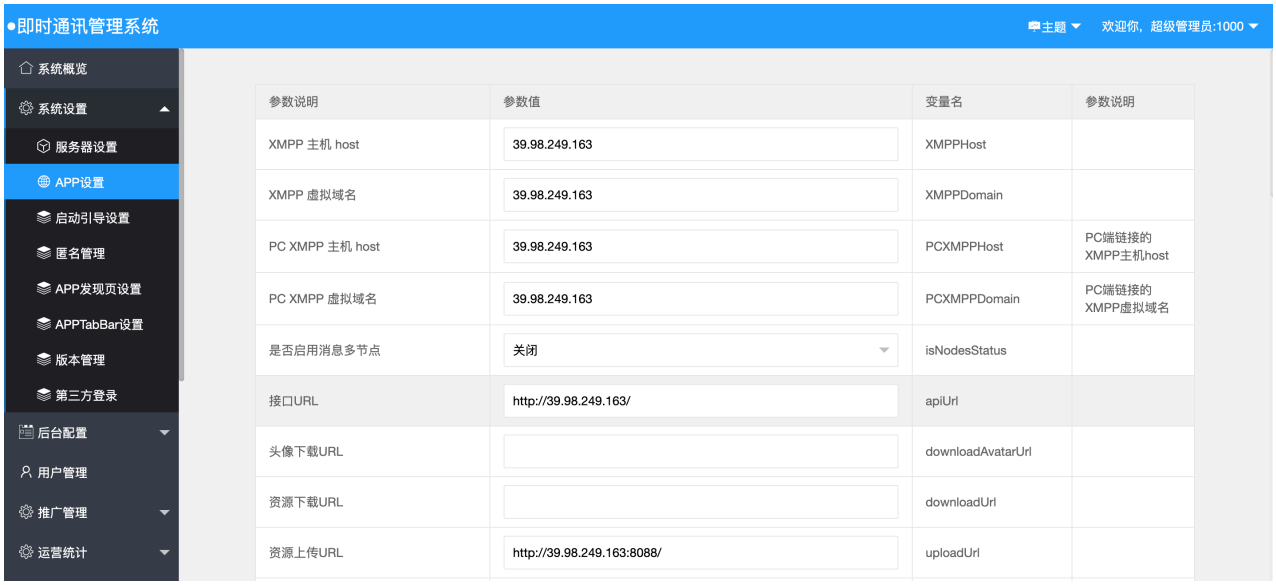
至此服务端相关服务已经部署完成

最后徐修改后台配置

Imapi服务启动完成后，在浏览器打开链接“http://localhost:8092/console/login”

账号密码 默认1000

将系统配置---> 客户端配置里的地址修改为部署的服务器的地址，这些地址用于在客户端启动时返回给客户端使用，请参考下图



修改配置保存后访问

http://localhost:8092/config

Raw Parsed


```
{"currentTime":1602308853904,"data":
{"PCXMPPDomain":"39.98.249.163","PCXMPPHost":"39.98.249.163","XMPPDomain":"39.98.249.163","XMPPHost":"39.98.249.163","XMPPTimeout":15,"address":"CN","aliLoginStatus":2,"aliPayStatus":
2,"aliWithdrawStatus":2,"androidAppUrl":"","androidDisable":"","androidExplain":"","androidVersion":0,"apiUrl":"http://39.98.249.163/","appleId":"","audioLen":"20","chatRecordTimeOut"
:-1,"displayRedPacket":1,"distance":20,"downloadAvatarUrl":"","downloadUrl":"","fileValidTime":-1,"guideWebsite":"","headBackgroundImg":"","helpUrl":"","hideSearchByFriends":1,"invisi
bleList":
[],"iosAppUrl":"","iosDisable":"","iosExplain":"","iosVersion":0,"ipAddress":"127.0.0.1","isCommonCreateGroup":0,"isCommonFindFriends":0,"isDiscoverStatus":1,"isNodesStatus":0,"isOpen
Cluster":0,"isOpenGoogleFCM":0,"isOpenOSStatus":0,"isOpenPositionService":0,"isOpenReadReceipt":1,"isOpenReceipt":1,"isOpenRegister":1,"isOpenSMSCode":1,"isOpenTelnum":1,"isOpenTwoBar
Code":1,"isQestionOpen":0,"isTabBarStatus":1,"isUserSignRedPacket":0,"isWithdrawToAdmin":0,"jitsiServer":"","liveUrl":"","macAppUrl":"","macDisable":"","macExplain":"","macVersion":0,
"minTransferAmount":0.0,"minWithdrawToAdmin":0.0,"nicknameSearchUser":2,"pCXMPPDomain":"39.98.249.163","pCXMPPHost":"39.98.249.163","pcAppUrl":"","pcDisable":"","pcExplain":"","pcVers
ion":0,"popularAPP":"
{\"lifeCircle\":1,\"videoMeeting\":1,\"liveVideo\":1,\"shortVideo\":1,\"peopleNearby\":1,\"scan\":1}","projectIco":"","projectLogo":"http://39.98.249.163:8087/group1/M00/00/00/rBoue18
xIPyAK4QOAAAdXTsxnRE548.png","projectName":"","qqLoginStatus":2,"regeditPhoneOrName":0,"registerInviteCode":0,"shareUrl":"","showContactsUser":1,"softUrl":"","tabBarConfigList":
{"tabBarNum":0,"tabBarStatus":0},"tlPayStatus":2,"tlWithdrawStatus":2,"transferExtraFee":0.0,"transferRate":0.0,"uploadMaxSize":0,"uploadUrl":"http://39.98.249.163:8088/","videoLen":"
20","website":"","wechatLoginStatus":2,"wechatPayStatus":2,"wechatWithdrawStatus":2,"xMPPDomain":"39.98.249.163","xMPPHost":"39.98.249.163","xMPPTimeout":15,"xmppPingTime":6},"resultC
ode":1}
```


最后

http://localhost:8092/config 为客户端配置的apiurl 将 apiurl和 apikey（在imapi服务配置文件中配置） 提供 给客户端，客户端打包需要的，至此大功告成。

11、将服务设置为开机自启动

注意：这里的启、停本质上也是执行对应服务的安装目录下的start 与stop脚本

如果之前start脚本里配置的是相对路径，那么就需要改为绝对路径

11.1 Mongodb

在/lib/systemd/system/目录下新建mongodb.service文件，内容如下：

```
[Unit]
Description=mongodb
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
ExecStart=/opt/mongodb-3.2.4/bin/mongod --config /opt/mongodb-3.2.4/mongo.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/opt/mongodb-3.2.4/bin/mongod  --shutdown  --config /opt/mongodb-
3.2.4/mongo.conf
PrivateTmp=true


[Install]
WantedBy=multi-user.target
```

如图所示：

设置权限

`[root@mac-pro~]# chmod 754 mongodb.service`

启动服务

`[root@mac-pro~]#systemctl start mongodb.service`

关闭服务

`[root@mac-pro~]#systemctl stop mongodb.service`

开机启动

`[root@mac-pro~]#systemctl enable mongodb.service`

11.2 Redis

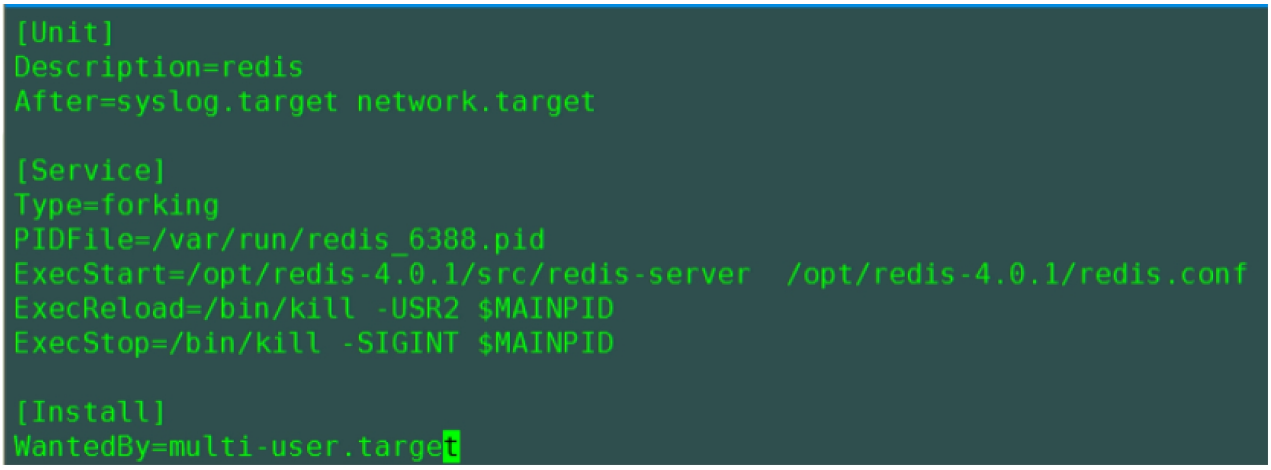
在/lib/systemd/system/目录下新建redis.service文件，内容如下：

```
[Unit]
Description=redis
After=syslog.target network.target

[Service]
Type=forking
PIDFile=/var/run/redis_6388.pid
ExecStart=/opt/redis-4.0.1/src/redis-server  /opt/redis-4.0.1/redis.conf
ExecReload=/bin/kill -USR2 $MAINPID
ExecStop=/bin/kill -SIGINT $MAINPID

[Install]
WantedBy=multi-user.target
```

如下图所示：



设置权限

`[root@mac-pro~]# chmod 754 redis.service`

启动服务

`[root@mac-pro~]#systemctl  start  redis.service`

关闭服务

`[root@mac-pro~]#systemctl  stop  redis.service`

开机启动

`[root@mac-pro~]#systemctl  enable redis.service`

11.3 Nginx

在/lib/systemd/system/目录下新建 nginx.service文件，内容如下：

```
[Unit]
Description=nginx
After=syslog.target network.target

[Service]
Type=forking
ExecStart=/opt/nginx_1.9.11/sbin/nginx
ExecReload=/opt/nginx_1.9.11/sbin/nginx -s reload
ExecStop=/opt/nginx_1.9.11/sbin/nginx -s quit
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

设置权限

`[root@mac-pro~]# chmod 754 nginx.service`

启动服务

`[root@mac-pro~]#systemctl  start  nginx.service`

关闭服务

`[root@mac-pro~]#systemctl  stop  nginx.service`

*开机启动 *

`[root@mac-pro~]#systemctl  enable nginx.service`

11.4 Imapi

注: 设置开机自启动服务启动脚本全部要求使用绝对路径 ， 和给予相应执行权限

如：

```
#!/bin/bash
GC="-XX:+UseBiasedLocking -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=2 -XX:+CMSIncrementalMode -XX:ParallelCMSThreads=2 -XX:-ReduceInitialCardMarks -XX:CMSInitiatingOc
cupancyFraction=70  -XX:+UseCMSInitiatingOccupancyOnly -XX:+HeapDumpOnOutOfMemoryError"
#EX="-XX:+OptimizeStringConcat -XX:+DoEscapeAnalysis -XX:+UseNUMA"

GC_DEBUG=" -XX:+PrintTenuringDistribution -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -Xloggc:logs/jvm.log -verbose:gc "

PRODUCTION_HEAP_SETTINGS=" -Xms2G -Xmx2G -Xss228k"

JAVA_OPTIONS="${GC} ${GC_DEBUG} -server ${PRODUCTION_HEAP_SETTINGS} ${DNS_RESOLVER} ${INTERNAL_IP} ${EXTERNAL_IP}  -XX:MaxDirectMemorySize=128m "

cd /opt/imapi/
nohup /opt/java/jdk1.8.0_131/bin/java ${JAVA_OPTIONS} -jar imserver-api-*.jar --spring.config.location=/opt/imapi/application.properties >> log.log &
tail -f log.log
```

在/lib/systemd/system/目录下新建 imapi.service文件，内容如下：

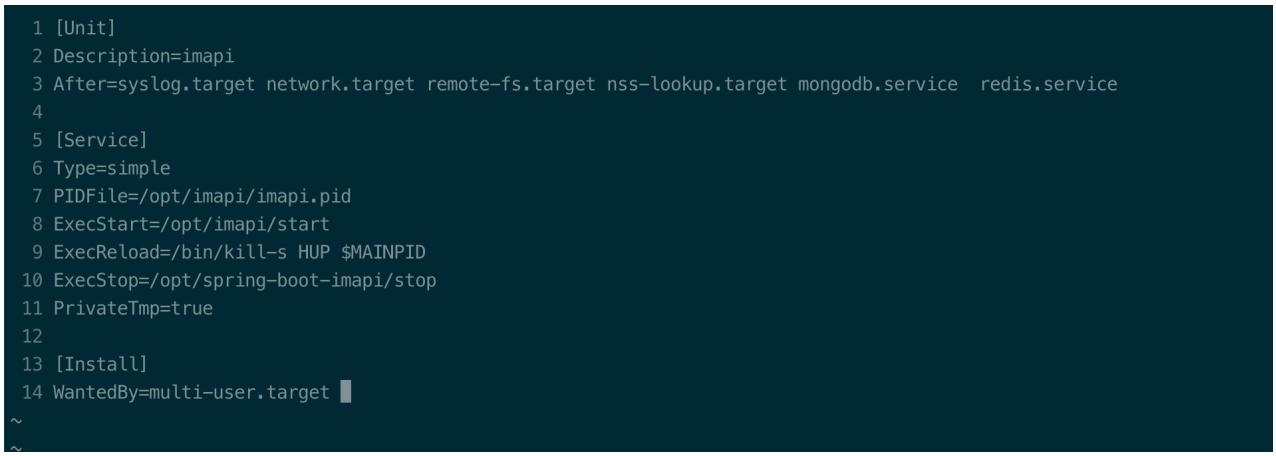
```
[Unit]
Description=imapi
```

```
After=syslog.target network.target remote-fs.target nss-lookup.target mongodb.service
redis.service

[Service]
Type=simple
PIDFile=/opt/imapi/imapi.pid
ExecStart=/opt/imapi/start
ExecReload=/bin/kill-s HUP $MAINPID
ExecStop=/opt/spring-boot-imapi/stop
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

如下图所示：



设置权限

`[root@mac-pro~]# chmod 754 imapi.service`

启动服务

`[root@mac-pro~]#systemctl start imapi.service`

关闭服务

`[root@mac-pro~]#systemctl stop imapi.service`

开机启动

`[root@mac-pro~]#systemctl enable imapi.service`

11.5 Tigase-Server

在/lib/systemd/system/目录下新建 tigase.service文件，内容如下：

```
[Unit]
Description=tigase-server
After=syslog.target network.target remote-fs.target nss-lookup.target mongodb.service
redis.service


[Service]

Type=forking
ExecStart=/opt/tigase-server-7.1.3-b4482/start
ExecReload=/bin/kill-s HUP $MAINPID
ExecStop=/opt/tigase-server-7.1.3-b4482/stop
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

设置权限

`[root@mac-pro~]# chmod 754 tigase-server.service`

启动服务

`[root@mac-pro~]#systemctl  start  tigase-server.service`

关闭服务

`[root@mac-pro~]#systemctl  stop  tigase-server.service`

开机启动

`[root@mac-pro~]#systemctl  enable tigase-server.service`

11.6 rocketmq

在/lib/systemd/system/目录下新建 rocketmq.service文件，内容如下：

```
[Unit]
Description=rocketmq
After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
ExecStart=/opt/rocketmq-all-4.3.2-bin-release/mqStart
ExecReload=/bin/kill-s HUP $MAINPID
ExecStop=/opt/rocketmq-all-4.3.2-bin-release/mqStop
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

设置权限

`[root@mac-pro~]# chmod 754 rocketmq.service`

启动服务

`[root@mac-pro~]#systemctl  start  rocketmq.service`

关闭服务

`[root@mac-pro~]#systemctl  stop  rocketmq.service`

开机启动

`[root@mac-pro~]#systemctl  enable rocketmq.service`

11.7 xmpp-push

在/lib/systemd/system/目录下新建 xmpp-push.service文件，内容如下：

```
[Unit]

iDescription=xmpp-push
After=syslog.target network.target remote-fs.target nss-lookup.target  mongodb.service
redis.service rocketmq.service

[Service]
ExecStart=/opt/xmpp-push/start
ExecReload=/bin/kill-s HUP $MAINPID
ExecStop=/opt/xmpp-push/stop
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

设置权限

```
[root@mac-pro~]# chmod 754 xmpp-push.service
```

启动服务

```
[root@mac-pro~]#systemctl  start  xmpp-push.service
```

关闭服务

```
[root@mac-pro~]#systemctl  stop  xmpp-push.service
```

开机启动

```
[root@mac-pro~]#systemctl  enable xmpp-push.service
```

11.8 push

在/lib/systemd/system/目录下新建 push.service文件，内容如下：

```
[Unit]

iDescription=push
After=syslog.target network.target remote-fs.target nss-lookup.target  mongodb.service
redis.service rocketmq.service

[Service]
ExecStart=/opt/push/start
ExecReload=/bin/kill-s HUP $MAINPID
ExecStop=/opt/push/stop
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

设置权限

```
[root@mac-pro~]# chmod 754 push.service
```

启动服务

```
[root@mac-pro~]#systemctl  start  push.service
```

关闭服务

```
[root@mac-pro~]#systemctl  stop  push.service
```

开机启动

```
[root@mac-pro~]#systemctl  enable push.service
```

11.9 upload

在/lib/systemd/system/目录下新建 upload.service文件，内容如下：

```
[Unit]

iDescription=push
After=syslog.target network.target remote-fs.target nss-lookup.target  mongodb.service
redis.service rocketmq.service

[Service]
ExecStart=/opt/upload/start
ExecReload=/bin/kill-s HUP $MAINPID
ExecStop=/opt/upload/stop
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

设置权限

`[root@mac-pro~]# chmod 754 upload.service`

启动服务

`[root@mac-pro~]#systemctl  start  upload.service`

关闭服务

`[root@mac-pro~]#systemctl  stop  upload.service`

开机启动

`[root@mac-pro~]#systemctl  enable upload.service`

二、部署方式二（通过docker部署）

注：如习惯使用Docker 容器部署项目，并具有相应的使用、维护经验，则可以使用Docker 容器的方式部署。

1. 安装docker（以centos7.X 为例）

`[root@mac-pro ~]# uname  -r`

```
[root@localhost ~]# uname -r
3.10.0-693.el7.x86_64
[root@localhost ~]#
```

安装 docker 服务

`[root@mac-pro ~]#  yum -y install docker-io`

```
Installed:
  docker.x86_64 2:1.13.1-63.git94f4240.el7.centos

Dependency Installed:
  container-selinux.noarch 2:2.55-1.el7
  container-storage-setup.noarch 0:0.9.0-1.rhel75.gite0997c3.el7
  docker-client.x86_64 2:1.13.1-63.git94f4240.el7.centos
  docker-common.x86_64 2:1.13.1-63.git94f4240.el7.centos
  oci-register-machine.x86_64 1:0-6.git2b44233.el7
  oci-systemd-hook.x86_64 1:0.1.15-2.gitc04483d.el7
  oci-umount.x86_64 2:2.3.3-3.gite3c9055.el7
  skopeo-containers.x86_64 1:0.1.29-3.dev.git7add6fc.el7

Dependency Updated:
  libselinux.x86_64 0:2.5-12.el7
  libselinux-python.x86_64 0:2.5-12.el7
  libselinux-utils.x86_64 0:2.5-12.el7
  libsemanage.x86_64 0:2.5-11.el7
  libsemanage-python.x86_64 0:2.5-11.el7
  libsepol.x86_64 0:2.5-8.1.el7
  policycoreutils.x86_64 0:2.5-22.el7
  policycoreutils-python.x86_64 0:2.5-22.el7
  selinux-policy.noarch 0:3.13.1-192.el7_5.3
  selinux-policy-targeted.noarch 0:3.13.1-192.el7_5.3
  setools-libs.x86_64 0:3.3.8-2.el7

Complete!
```

启动docker 服务

`[root@mac-pro ~]#service docker start`

```
[root@localhost ~]# service docker start
Redirecting to /bin/systemctl start docker.service
[root@localhost ~]# ps -ef|grep docker
root      3850     1  0 06:52 ?        00:00:00 /usr/bin/dockerd-current --add-runtime d
ocker-runc=/usr/libexec/docker/docker-runc-current --default-runtime=docker-runc --exec-
opt native.cgroupdriver=systemd --userland-proxy-path=/usr/libexec/docker/docker-proxy-c
urrent --init-path=/usr/libexec/docker/docker-init-current --seccomp-profile=/etc/docker
/seccomp.json --selinux-enabled --log-driver=journald --signature-verification=false --s
torage-driver overlay2
root      3855  3850  0 06:52 ?        00:00:00 /usr/bin/docker-containerd-current -l un
ix:///var/run/docker/libcontainerd/docker-containerd.sock --metrics-interval=0 --start-t
imeout 2m --state-dir /var/run/docker/libcontainerd/containerd --shim docker-containerd-
shim --runtime docker-runc --runtime-args --systemd-cgroup=true
root      3961  2366  0 06:53 pts/0    00:00:00 grep --color=auto docker
[root@localhost ~]#
```

2. Docker 创建Redis 容器

1、从官方仓库拉取redis 4.0.1 的镜像

```
[root@mac-pro ~]# docker pull redis:4.0.1
```

```
[root@izbp1a5bxfhn22cit0b97wz ~]# docker pull redis:4.0.1
4.0.1: Pulling from library/redis
065132d9f705: Pull complete
be9835c27852: Pull complete
f4a0d1212c38: Pull complete
ea1f878b621a: Pull complete
7a838393b4b9: Pull complete
9f48e489da12: Pull complete
Digest: sha256:8a54dcc711406447b3631a81ef929f500e6836b43e7d61005fa27057882159da
Status: Downloaded newer image for redis:4.0.1
```

2、在宿主机创建 /data/redis 目录

```
mkdir  /data/redis
```

3、通过镜像启动容器

```
docker run -p 6388:6379 --name redis-4.0.1  --hostname redis  --restart=always -v
/data/redis:/data  -d redis:4.0.1  redis-server --appendonly yes
```

```
[root@izbp1a5bxfhn22cit0b97wz docker]# docker run -p 6388:6379 --name redis-4.0.1  --hostname red
erver --appendonly yes
119e4c8bfef1bb56043b5ed0bc755e20f6bc92545640a42ed99430d5e4af9b58
[root@izbp1a5bxfhn22cit0b97wz docker]#
```

3. Docker 创建Mongodb容器

1、使用命令从官方仓库拉取3.4.0 版本的镜像

```
[root@mac-pro ~]# docker  pull  mongo:3.4.0
```

2、在宿主机创建目录

```
mkdir  -p  /data/mongodb/db
```

3、使用镜像启动容器

```
docker run -d -p 27107:27017  -v  /data/mongodb/db:/data/db  --restart=always --name mongodb  mongo:3.4.0
```

注：IP地址请根据实际部署机器的内网ip进行调整

4. Docker 创建imapi容器

1、在imapi 的部署文件imapi中创建 imapi_dockerfile文件 ， 写入如下内容

```
FROM java:8
VOLUME  /opt/logs
COPY application.properties /opt/application.properties
ADD imapi.war /opt/imapi.war
RUN bash -c 'touch /opt/imapi.war'
EXPOSE 8092

ENTRYPOINT ["java","-jar","/opt/imapi.war","--
spring.config.location=/opt/application.properties"]
```

说明：这里的application.properties 从 imapi 目录里拷贝mapi.war 为imapi 服务的部署包

2、通过dockerfile 文件构建镜像

```
docker build  -t  imapi  -f  imapi_dockerfile  ./
```

3、通过镜像启动容器

```
docker run -d -p 8092:8092 -e "UPLOAD_ADDR=192.168.0.152:8088" -e "MONGODB_ADDR=192.168.0.151:27107"   -e "NAMESRV_ADDR=192.168.0.155:9876"  -e "XMPP_HOST=192.168.0.155" -e "XMPP_SERVERNAME=im.mac-pro.co" -e "REDIS_ADDR=redis://192.168.0.155:6379" --name  imapi-server  imapi
```

5. Docker 创建Tigase-server容器

1、修改 tigase-server-7.1.3-b4482/etc/tigase.conf 配置文件

```
#DNS_RESOLVER=" -Dresolver-class=tigase.util.DNSResolverDefault "
#INTERNAL_IP=" -Dtigase-primary-address=hostname.local "
#EXTERNAL_IP=" -Dtigase-secondary-address=hostname "
Shiku="-Dtigase-configurator=tigase.shiku.conf.ShikuConfigurator"
JAVA_HOME="${JAVA_HOME}"
CLASSPATH=""
TIGASE_HOME="/opt/tigase-server"
#PRODUCTION_HEAP_SETTINGS=" -Xms5G -Xmx5G " # heap memory settings mus
JAVA_OPTIONS="${GC} ${Shiku}  ${GC_DEBUG} ${EX} ${ENC} ${DRV} ${JMX_RE
"
TIGASE_OPTIONS=" --property-file etc/init.properties "
```

修改 JAVA_HOME的值为"${JAVA_HOME}"，修改TIGASE_HOME的值为"/opt/tigase-server"

JAVA_HOME="${JAVA_HOME}"

TIGASE_HOME="/opt/tigase-server"

2、修改 tigase-server-7.1.3-b4482/etc/init.properties 配置文件

```
config-type=--gen-config-def
--admins=admin@oem.shiku.co,10005@oem.shiku.co
--virt-hosts=oem.shiku.co
--debug=server,xmpp,shiku,db,net

# 集群配置
#--cluster-mode=true
#--cluster-nodes=tigase-1:5277,tigase-2:5277
#--cluster-connect-all=true

--user-db=tigase.mongodb.MongoRepository
--auth-db=tigase.mongodb.MongoRepository
--user-db-uri=mongodb://172.16.6.117:28018/tigase


##imapi 的数据库链接 修改用户在线状态是用到
##mongodb://sysop:moon@localhost
--api-db-uri=mongodb://172.16.6.117:28018/imapi

##关键词过滤功能   1：打开  0：关闭
--confirm-open-keyword =1
#消息存储
--shiku-archive-jid = shiku-message-archive@oem.shiku.co
--shikuDeBug = 1
#离线通知
--shikuPush_mqAddr=192.168.0.139:9876

--sm-plugins=-starttls,shiku-auto-reply,shiku-offline-msg,shiku-message-archive-plugin,jabber:iq:register
##XEP-0198
c2s/processors[s]=urn:xmpp:sm:3
bosh/processors[s]=urn:xmpp:sm:3
# AMP Component
```

Tigase 所在机器的IP或域名，生产环境建议使用域名，注意，这里如配置ip不能是127.0.0.1，必须是实际外网ip，如：39.17.42.155

数据库链接地址

这里和顶部的 virt-hosts 保持一致

RocketMq 连接地址

```
# AMP Component
sess-man/plugins-conf/enabled-mechanisms=PLAIN
amp/amp-repo-uri=mongodb://172.16.6.117:28018/tigase
amp/amp-repo-class=tigase.mongodb.MongoMsgRepository
# AMP Plugin
sess-man/plugins-conf/amp/amp-repo-uri=mongodb://172.16.6.117:28018/tigase
sess-man/plugins-conf/amp/amp-repo-class=tigase.mongodb.MongoMsgRepository
amp/store-limit[L]=1000
#https
#bosh/connections/ports[i] = 5280,5281
#bosh/connections/5281/socket = ssl
#bosh/connections/5281/type = accept

#bosh/max-batch-size[I]=100
bosh/max-session-waiting-packets[I]=500

--comp-name-1=http
--comp-class-1=tigase.http.HttpMessageReceiver
http/api-keys[s]=open_access
http/http/port[I]=9090
http/http/server-class[S]=tigase.http.jetty.JettyStandaloneHttpServer

# 群聊组件
--comp-name-2=muc
--comp-class-2=tigase.muc.MUCComponent

## 遍历所有的连接，检查它们是否都真正的活着  间隔时间 毫秒   10000 即 10秒
--watchdog_delay=40000
--watchdog_ping_type=xmpp
## 毫秒  如果 客户端 超出该时间 未与服务端链接  即 离线
--watchdog_timeout=75000

muc/history-db=tigase.mongodb.muc.MongoHistoryProvider
muc/history-db-uri=mongodb://172.16.6.117:28018/tigase
muc/shiku-room-db=tigase.shiku.db.MongoShikuMucRoomRepository
muc/shiku-room-db-uri=mongodb://172.16.6.117:28018/imRoom
muc/default_room_config/muc#maxhistoryfetch=100
muc/muc-lock-new-room[B]=false
--comp-name-pubsub = pubsub
--comp-class-pubsub = tigase.pubsub.PubSubComponent

# 视酷消息归档组件
--comp-name-3=shiku-message-archive
--comp-class-3=tigase.shiku.ShikuMessageArchiveComponent
shiku-message-archive/archive-repo-uri=mongodb://172.16.6.117:28018/tigase
shiku-message-archive/archive-repo-class=tigase.shiku.db.MongoShikuMessageArchiveRepository
shiku-message-archive/muc-repo-uri=mongodb://172.16.6.117:28018/imRoom
shiku-message-archive/muc-msgs-split-method=year/month/day
shiku-message-archive/msgs-split-method=year/month/day


sess-man/incoming-filters=tigase.shiku.ShikuKeywordFilter
##message-router/incoming-filters=tigase.shiku.ShikuKeywordFilter
#c2s/incoming-filters=tigase.shiku.ShikuKeywordFilter
```


3、编辑tigase-dockerfile 文件，写入如下内容

> 注：tigase-server-7.1.3-b4482 为tigase-server 的部署文件的目录 5222 端口用于Android、Ios、PC 等客户端连接使用，5280 端口用于WEB 版连接使用，如没有 WEB 版5280 端口可不用

```
FROM java:8
VOLUME  /opt/logs
ADD ./tigase-server-7.1.3-b4482/  /opt/tigase-server
RUN chmod  -R 755 /opt
EXPOSE 5222
EXPOSE 5280
ENTRYPOINT cd /opt/tigase-server; java -version; ./scripts/tigase.sh run
etc/tigase.conf; wait $!
```

4、通过dockerfile 文件构建镜像

```
docker  build  -t  tigase-server -f  tigase-dockerfile ./
```

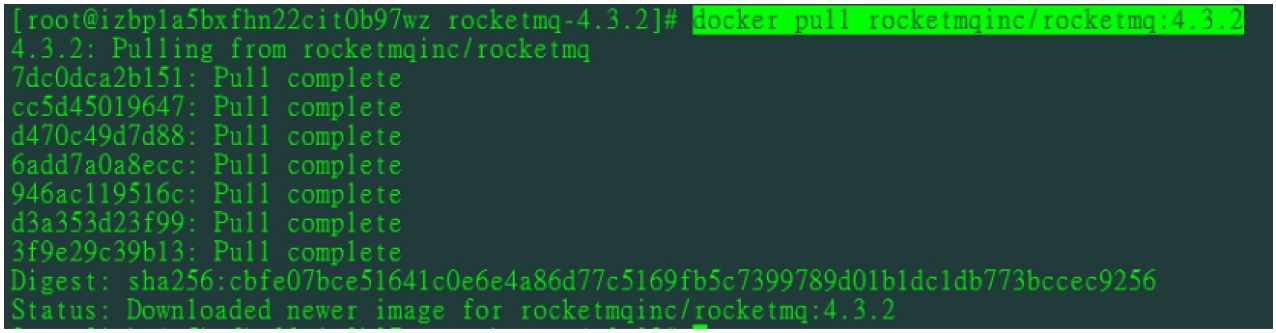
5、通过镜像启动容器

```
docker run -d -p 5222:5222  -p 5280:5280  --name  tigase-server  tigase-server
```

6. Docker 创建RocketMq容器

1、拉取rocketmq-4.3.2 的镜像

```
docker pull rocketmqinc/rocketmq:4.3.2
```



2、在宿主机创建目录

```
mkdir  -p  /data/namesrv/logs
mkdir  -p  /data/namesrv/store
mkdir  -p  /data/broker/logs
mkdir  -p  /data/broker/store
```

3、使用镜像启动 NameServer 容器

```
docker run -d -p 9876:9876 -v /data/namesrv/logs:/root/logs -v
/data/namesrv/store:/root/store --name rmqnamesrv  rocketmqinc/rocketmq:4.3.2 sh
mqnamesrv
```

4、使用镜像启动broker 容器（说明：***-Xms512m 等内存值可根据机器实际内存值进行调整***）

```
docker run -d -p 10911:10911 -p 10909:10909 -v /data/broker/logs:/root/logs -v
/data/broker/store:/root/store --name rmqbroker --link rmqnamesrv:namesrv -e
"NAMESRV_ADDR=namesrv:9876"  -e "JAVA_OPT=${JAVA_OPT} -server -Xms512m -Xmx512m -
Xmn256m -XX:PermSize=128m -XX:MaxPermSize=128m" rocketmqinc/rocketmq:4.3.2  sh
mqbroker
```

7. Docker 创建push容器

1、在push的部署文件push中创建push_dockerfile文件， 写入如下内容

```
FROM java:8
VOLUME  /opt/push/logs
COPY application.properties /opt/push/application.properties
COPY *.p12  /opt/push/
COPY sixth-hawk-164509-firebase-adminsdk-342gn-465351f0ef.json /opt/mac-pro-
push/sixth-hawk-164509-firebase-adminsdk-342gn-465351f0ef.json
ADD push.war /opt/push/push.war
RUN bash -c 'touch /opt/push/push.war'
ENTRYPOINT ["java","-jar","/opt/push/push-1.0.war","--
spring.config.location=/opt/push/application.properties"]
```

说明：这里的application.properties 从 push目录里拷贝

> .p12 文件为ios 推送证书文件
>
> sixth-hawk-164509-firebase-adminsdk-342gn-465351f0ef.json 为Google 推送需要的文件
>
> push.war 为 push 服务部署包

2、通过dockerfile 文件构建镜像

```
docker build -t mac-pro-push  -f push_dockerfile  ./
```

3、通过镜像启动容器

```
docker run -d -e "MONGODB_ADDR=192.168.0.151:27107"   -e
"NAMESRV_ADDR=192.168.0.155:9876"  -e "XMPP_HOST=192.168.0.155" -e
"XMPP_SERVERNAME=192.168.0.155" -e "REDIS_ADDR=redis://192.168.0.155:6379" --name  push-
server  push
```

8.Docker 创建xmpp-push 容器

1、在xmpp-push的部署文件xmpp-push中创建 xmpp-push_dockerfile文件， 写入如下内容

```
FROM java:8
VOLUME  /opt/xmpp-push/logs
COPY application.properties /opt/xmpp-push/application.properties
ADD  xmpp-push-0.0.1-SNAPSHOT.war /opt/xmpp-push/xmpp-push-0.0.1-SNAPSHOT.war
RUN bash -c 'touch /opt/xmpp-push/xmpp-push-0.0.1-SNAPSHOT.war'
ENTRYPOINT ["java","-jar","/opt/xmpp-push/xmpp-push-0.0.1-SNAPSHOT.war","--
spring.config.location=/opt/xmpp-push/application.properties"]
```

> 说明：这里的application.properties 从 conf_file/message-push目录里拷贝

> xmpp-push-0.0.1-SNAPSHOT.war 为 xmpp-push 服务的部署包

2、通过dockerfile 文件构建镜像

```
docker  build  -t  message-push  -f  message-push_dockerfile  ./
```

3、通过镜像启动容器

```
docker run -d -e "MONGODB_ADDR=192.168.0.151:27107"   -e
"NAMESRV_ADDR=192.168.0.151:9876"  -e "XMPP_HOST=192.168.0.151" -e
"XMPP_SERVERNAME=im.mac-pro.co" -e "REDIS_ADDR=redis://192.168.0.151:6379" --name  xmpp-
push-server  xmpp-push
```

9.Docker 创建Upload容器

1、在upload的部署文件upload中创建 upload_dockerfile文件 ， 写入如下内容

```
FROM java:8
VOLUME  /opt/upload/logs
COPY application.properties /opt/upload/application.properties
ADD  upload-2.0.war /opt/upload/upload-2.0.war
RUN bash -c 'touch /opt/upload/upload-2.0.war'
RUN mkdir  -p  /data/www/resources
ENTRYPOINT ["java","-jar","/opt/upload/upload-2.0.war","--
spring.config.location=/opt/upload/application.properties"]
```

> 说明：这里的application.properties 从 conf_file/upload目录里拷贝
>
> upload-2.0.war 为 upload 服务的部署包

2、通过dockerfile 文件构建镜像

```
docker build -t upload  -f upload_dockerfile  ./
```

3、在宿主机创建文件存储目录，挂载到容器(可将下面命令复制后一次执行)

```
mkdir  -p  /data/www/resources
cd  /data/www/resources
mkdir audio
mkdir avatar
mkdir avatar/o
mkdir avatar/t
mkdir avatar_r
```

```
mkdir avatar_r/o
mkdir avatar_r/t
mkdir gift
mkdir image
mkdir image/o
mkdir image/t
mkdir other
mkdir preview
mkdir temp
mkdir u
mkdir video
```

4、通过镜像启动容器

`docker run -d  -p 8088:8088 -v /data/www/resources:/data/www/resources" -e "MONGODB_URL=mongodb://192.168.0.151:27107" --name upload-server  upload`

10.Docker 创建Nginx容器

说明：这里Nginx 主要提供文件访问服务，需要和 Upload 放在同一宿主机，以便能读取到上传的文件

1、拉取官方镜像

`docker  pull  nginx`

2、在宿主机创建目录，编辑配置文件

```
mkdir  -p  /data/nginx/logs
mkdir  -p  /data/nginx/conf
cd  /data/nginx/conf
```

vim nginx.conf

写入如下内容：

```
user  root;
worker_processes  1;
pid     /run/nginx.pid;

events {
  worker_connections  1024;
}


http {
```

```
    include    mime.types;
    default_type  application/octet-stream;
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    sendfile     on;
    keepalive_timeout  65;
  server{
    listen     8080;
    charset utf-8;
    location / {
     if ($http_referer ~* ".php") {
         return 403;
     }
      root   /www/resources;
      expires  5d;
    }
  }
  }
```

3、通过镜像启动容器

`docker run -p 8080:8080 --name nginx -v /data/www/resources:/www/resources -v /data/nginx/conf/nginx.conf:/etc/nginx/nginx.conf -v /data/nginx/logs:/var/log/nginx  -d nginx`

三、服务器管理、维护

1.修改调整服务器最大连接数

使用以下命令查看当前最大连接数：（默认为1024，需要改大）

`[root@mac-pro ~]# ulimit -n`

1024

修改以下配置文件：

1.1 编辑 /etc/security/limits.conf

`[root@mac-pro~]# vim  /etc/security/limits.conf`

```
*    soft   nofile   204800
*    hard   nofile   204800
*    soft   nproc   204800
*    hard   nproc   204800
```

在配置文件中添加以上内容

1.2 编辑 /etc/pam.d/login

`[root@mac-pro~]# vim /etc/pam.d/login`

session required pam_limits.so

在配置文件中添加以上内容

将以上保存好，然后重启服务器，再使用ulimit -n

`[root@mac-pro~]# ulimit -n`

204800

2 查看监听的端口

当某个服务出现不能正常访问，使用如下命令会查看对应的端口是否处于监听状态

#[root@mac-pro~]#netstat -lntp

```
[root@337b81dcc75c resources]# netstat -lntp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      50/nginx: master pr
tcp        0      0 0.0.0.0:28018           0.0.0.0:*               LISTEN      25/mongod
tcp        0      0 127.0.0.1:6388          0.0.0.0:*               LISTEN      45/redis-server 127
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      41/sshd
tcp        0      0 0.0.0.0:8089            0.0.0.0:*               LISTEN      50/nginx: master pr
tcp6       0      0 :::8080                 :::*                    LISTEN      425/java
tcp6       0      0 :::5269                 :::*                    LISTEN      75/java
tcp6       0      0 :::22                   :::*                    LISTEN      41/sshd
tcp6       0      0 :::8092                 :::*                    LISTEN      55/java
tcp6       0      0 :::5280                 :::*                    LISTEN      75/java
tcp6       0      0 :::9090                 :::*                    LISTEN      75/java
tcp6       0      0 :::5222                 :::*                    LISTEN      75/java
tcp6       0      0 :::5223                 :::*                    LISTEN      75/java
tcp6       0      0 :::8009                 :::*                    LISTEN      425/java
tcp6       0      0 :::5290                 :::*                    LISTEN      75/java
```

各个服务对应端口说明：

服务名称	描述	端口号使用情况
Tigase-server	xmpp 通讯服务	客户端使用5222，Web版使用5280端口
upload	文件上传服务	8088
mongodb	数据库	27017
Redis	缓存服务	6379
Nginx	主要用于文件访问时的目录映射	8080
fastDfs	分布式文件存储服务	tracker 服务使用 22122 端口 Storage 服务使用 23000端口