

咕喃服务端安装部署文档

---更新于 2021-06-06

目录

咕喃服务端安装部署文档.....	1
端口说明.....	2
部署方式-直接安装.....	2
Linux 系统下直接安装 IM 相关服务.....	2
1、 安装 MongoDB.....	2
2、 安装 Redis.....	3
3、 安装 Jdk1.8+.....	4
4、安装 RocketMQ 队列.....	5
5、安装 Spring-boot-imapi api 接口服务.....	8
6、 安装 IM-Server socketIM 通讯服务.....	10
7、安装 shiku-push 推送服务.....	13
8、安装 message-push 服务.....	16
9、安装 Upload 文件上传服务.....	17
10、安装 Nginx, 配置文件访问.....	19

端口说明

服务	用途	使用端口	说明
Mongodb	数据库	28018	不用对外开放
Redis	缓存	6379	不用对外开放
Spring-boot-ima pi	Api 接口	8092	需要开放
Im-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	一般使用 8089	需要开放

部署方式-直接安装

1) Linux 系统下直接安装 IM 相关服务

1、安装 MongoDB

① 下载并解压缩 mongodb 安装包：

```
[root@shiku~]# cd /opt
[root@shiku~]# wget http://xxx/soft/mongodb-linux-x86_64-3.4.0.tgz
[root@shiku~]# tar -zxvf mongodb-linux-x86_64-3.4.0.tgz
[root@shiku~]# mv mongodb-linux-x86_64-3.4.0 mongodb-3.4.0
```

② 在/opt/mongodb-3.4.0 目录下创建 mongo.conf 文件内容如下：

```
[root@shiku~]# cd mongodb-3.4.0
[root@shiku~]# vim mongo.conf
```

配置模板：

```
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: 28018
setParameter:
  enableLocalhostAuthBypass: false

```

#注 加上下面的配置必须设置密码

```

security:
  authorization: enabled

```

③然后创建 mongodb 数据目录，和日志目录

```

[root@shiku~]# mkdir -p /data/mongodb
[root@shiku~]# mkdir logs

```

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

```

[root@www mongodb .]# sh start
about to fork child process, waiting until server is ready for connections.
forked process: 23638
child process started successfully, parent exiting

```

Stop 脚本:

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

2、安装 Redis

① 下载、解压、安装

```

[root@shiku~]# cd /opt
[root@shiku~]# wget http://xxx/soft/redis-4.0.1.tar.gz
[root@shiku~]# tar -zxvf redis-4.0.1.tar.gz
[root@shiku~]# cd redis-4.0.1

```

```
[root@shiku~]# make && make install
```

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)

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
```

④ 执行 sh start 命令启动脚本, 查看 redis 是否启动成功

```
[root@www redis-2.8.19]# sh start
[root@www redis-2.8.19]# ps -ef|grep redis
root      8346      1    0 17:32 ?        00:00:00 /usr/local/redis-2.8.19/src/redis-server *:6380
root      8374    6564    0 17:33 pts/0    00:00:00 grep redis
root     11105      1    0 Jul09 ?        00:12:35 /usr/local/redis-2.8.12/src/redis-server *:6379
[root@www redis-2.8.19]#
```

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@shiku~]# cd /opt
[root@shiku~]# wget http://xxx/soft/jdk-8u131-linux-x64.tar.gz
[root@shiku~]# tar -zxvf jdk-8u131-linux-x64.tar.gz
[root@shiku~]# mkdir java
[root@shiku~]# mv jdk1.8.0_131 ./java
```

设置 jdk 环境变量

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

```
[root@shiku~]# 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
```

使环境变量生效

```
[root@shiku~]# source /etc/profile
```

⑤ 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)
```

```
[root@www nginx]# java -version
java version "1.8.0_05"
Java(TM) SE Runtime Environment (build 1.8.0_05-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.5-b02, mixed mode)
```

⑥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 RocketMQ 队列

下载地址：
<http://mirror.bit.edu.cn/apache/rocketmq/4.3.2/rocketmq-all-4.3.2-bin-release.zip>

```
cd /opt
wget
http://mirror.bit.edu.cn/apache/rocketmq/4.3.2/rocketmq-all-4.3.2-bin-release.zip
unzip rocketmq-all-4.3.2-bin-release.zip
mv rocketmq-all-4.3.2-bin-release rocketmq-4.3.2
cd rocketmq-4.3.2
```

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

vim bin/runbroker.sh

```
#####
# JVM Configuration
#####
JAVA_OPT="${JAVA_OPT} -server -Xms3g -Xmx3g -Xenid"
JAVA_OPT="${JAVA_OPT} -XX:+UseG1GC -XX:G1HeapRegionSize=10m -XX:G1ReservePercent=25 -XX:InitiatingHeapOccupancyPercent=10 -XX:SoftRefLRUPolicyMSPerMB=0 -XX:SurvivorRatio=8"
JAVA_OPT="${JAVA_OPT} -verbose:gc -Xloggc:/dev/shm/mq_gc.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCApplicationStoppedTime -XX:+PrintAdaptiveSizePolicy"
JAVA_OPT="${JAVA_OPT} -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=30m"
JAVA_OPT="${JAVA_OPT} -XX:-ExitStackTraceInFastThrow"
JAVA_OPT="${JAVA_OPT} -XX:+AlwaysPreTouch"
JAVA_OPT="${JAVA_OPT} -XX:MaxDirectMemorySize=1g"
JAVA_OPT="${JAVA_OPT} -XX:-UseLargePages -XX:-UseBiasedLocking"
JAVA_OPT="${JAVA_OPT} -Djava.ext.dirs=${JAVA_HOME}/jre/lib/ext:${BASE_DIR}/lib"
#JAVA_OPT="${JAVA_OPT} -Xdebug -Xrunjdwp:transport=dt_socket,address=9555,server=y,suspend=n"
JAVA_OPT="${JAVA_OPT} ${JAVA_OPT_EXT}"
JAVA_OPT="${JAVA_OPT} -cp ${CLASSPATH}"
```

vim bin/runserver.sh

```
#####
# JVM Configuration
#####
JAVA_OPT="${JAVA_OPT} -server -Xms1g -Xmx1g -Xenid -XX:MetaspaceSize=120m -XX:MaxMetaspaceSize=120m"
JAVA_OPT="${JAVA_OPT} -XX:+UseConcMarkSweepGC -XX:+UseCMSCompactAtFullCollection -XX:CMSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled -XX:SoftRefLRUPolicyMSPerMB=0 -XX:+CMSClassUnloadingEnabled -XX:SurvivorRatio=8 -XX:-UseParNewGC"
JAVA_OPT="${JAVA_OPT} -verbose:gc -Xloggc:/dev/shm/mq_srv_gc.log -XX:+PrintGCDetails"
JAVA_OPT="${JAVA_OPT} -XX:-ExitStackTraceInFastThrow"
JAVA_OPT="${JAVA_OPT} -XX:-UseLargePages"
JAVA_OPT="${JAVA_OPT} -Djava.ext.dirs=${JAVA_HOME}/jre/lib/ext:${BASE_DIR}/lib"
#JAVA_OPT="${JAVA_OPT} -Xdebug -Xrunjdwp:transport=dt_socket,address=9555,server=y,suspend=n"
```

vim startSrv

写入如下内容:

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

执行 startSrv 脚本启动 nameServer

sh startSrv

```
2018-11-21 16:29:45 INFO main - tls.server.trustCertPath = null
2018-11-21 16:29:45 INFO main - tls.client.keyPassword = null
2018-11-21 16:29:45 INFO main - tls.client.certPath = null
2018-11-21 16:29:45 INFO main - tls.client.authServer = false
2018-11-21 16:29:45 INFO main - tls.client.trustCertPath = null
2018-11-21 16:29:46 INFO main - Using OpenSSL provider
2018-11-21 16:29:46 INFO main - SSLContext created for server
2018-11-21 16:29:46 INFO NettyEventExecutor - NettyEventExecutor service started
2018-11-21 16:29:46 INFO main - The Name Server boot success. serializeType=JSON
2018-11-21 16:29:46 INFO FileWatchService - FileWatchService service started
```

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
```

```
2018-11-21 17:25:35 WARN main - Load default transaction message hook service: TransactionalMessageService
mpl
2018-11-21 17:25:35 WARN main - Load default discard message hook service: DefaultTransactionalMessageCheck
Listener
2018-11-21 17:25:35 INFO FileWatchService - FileWatchService service started
2018-11-21 17:25:35 INFO PullRequestHoldService - PullRequestHoldService service started
2018-11-21 17:25:35 INFO brokerOutApi_thread_1 - register broker to name server 192.168.0.155:9876 OK
2018-11-21 17:25:35 INFO main - Start transaction service!
2018-11-21 17:25:35 INFO main - The broker[imtest1, 192.168.0.155:10911] boot success. serializeType=JSON
and name server is 192.168.0.155:9876
2018-11-21 17:25:45 INFO BrokerControllerScheduledThread1 - dispatch behind commit log 0 bytes
2018-11-21 17:25:45 INFO BrokerControllerScheduledThread1 - Slave fall behind master: 0 bytes
2018-11-21 17:25:45 INFO brokerOutApi_thread_2 - register broker to name server 192.168.0.155:9876 OK
```

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

```
[root@izbp14lat0trtf309x95roz shiku-push]# jps
23960 Jps
15401 NamesrvStartup
15402 BrokerStartup
```

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
```

```
[root@imtest1 rocketmq-all-4.3.2-bin-release]# sh bin/mqadmin updateTopic -n localhost:9876 -c DefaultC
ter -t chatMessage
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
create topic to 192.168.0.155:10911 success.
```

附： 单个脚本启动

```
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 Spring-boot-imapi api 接口服务

```
[root@shiku~]# cd /opt
```

```
[root@shiku~]# wget http://xxx/soft/kuxin/spring-boot-imapi.tar
```

```
[root@shiku~]# tar -xvf spring-boot-imapi.tar
```

```
[root@shiku~]# cd spring-boot-imapi
```

```
[root@shiku~]# vim application.properties
```

2)修改 [application.properties](#) 配置文件


```

23 ##APP Properties
24 im.appConfig.uploadDomain=http://test.shiku.co:8088
25 im.appConfig.apiKey=
26 im.appConfig.openTask=1
27 im.appConfig.distance=50
28 im.appConfig.isBeta=1
29 im.appConfig.qqzengPath=/opt/spring-boot-imapi/qqzeng-ip-3.0-ultimate.dat
30 im.appConfig.languages[0].key=zh
31 im.appConfig.languages[0].name=\u4E2D\u6587
32 im.appConfig.languages[0].value=\u7B80\u4F53\u4E2D\u6587
33 im.appConfig.languages[1].key=en
34 im.appConfig.languages[1].name=\u82F1\u6587
35 im.appConfig.languages[1].value=English
36 im.appConfig.languages[2].key=big5
37 im.appConfig.languages[2].name=\u7E41\u4F53
38 im.appConfig.languages[2].value=\u7E41\u4F53\u4E2D\u6587
39
40
41 ## SMS Properties
42 im.smsConfig.openSMS=1
43 im.smsConfig.host=m.isms360.com
44 im.smsConfig.port=8085
45 im.smsConfig.api=/mt/MT3.ashx
46 im.smsConfig.username=
47 im.smsConfig.password=
48 im.smsConfig.templateChineseSMS=【视酷IM】，您的验证码为：
49 im.smsConfig.templateEnglishSMS=[SHIKU IM], Your verification code is:
50
51 ## 阿里云短信服务
52 im.smsConfig.product=Dysmsapi
53 im.smsConfig.domain=dysmsapi.aliyuncs.com
54 im.smsConfig.accessKeyId=LT..
55 im.smsConfig.accessKeySecret=ldT...
56 im.smsConfig.signname=\u89c6\u9177IM
57 im.smsConfig.chinese_templatecode=SMS_168..
58 im.smsConfig.english_templatecode=SMS_168..
59
60
61 #Mongodb Properties (数据库配置)
62 im.mongoConfig.uri=mongodb://127.0.0.1:28018
63 im.mongoConfig.dbName=imapi
64 im.mongoConfig.roomDbName=imRoom
65 im.mongoConfig.username=
66 im.mongoConfig.password=
67 im.mongoConfig.connectTimeout=20000
68 im.mongoConfig.socketTimeout=20000
69 im.mongoConfig.maxWaitTime=20000
70
71
72 #mq 配置
73 im.mqConfig.nameAddr=127.0.0.1:9876
74 ##mq 消费最小程数量 默认 cup 数量
75 im.mqConfig.threadMin=4
76 ##mq 消费最大程数量 默认 cup 数量*2
77 im.mqConfig.threadMax=8
78 ##mq 批量消费数量 默认 20
79 im.mqConfig.batchMaxSize=30
80
81
82 #Redis Properties (缓存配置)
83 im.redisConfig.address=redis://127.0.0.1:6379
84 im.redisConfig.database=0
85 im.redisConfig.password=

```

upload 上传服务连接地址，这里可以配置内网地址，用于在imapi里面调用upload

可设置为任意字符串，用于http接口安全认证
各个客户端和服务端配置的保持一致

是否开启短信功能 1开启 0关闭

mongodb连接地址

rockermq连接地址

redis连接地址

3) 在 spring-boot-imapi 目录下执行 sh start 命令运行 imapi 接口服务

```

[root@www spring-boot-imapi]# pwd
/usr/local/spring-boot-imapi
[root@www spring-boot-imapi]# ll
总用量 32528
-rw-r--r--. 1 root root    3070 2月   6 17:07 application.properties
-rw-r--r--. 1 root root  4655681 6月   2 16:59 log.log
-rwx----- 1 root  401 28635059 5月  26 10:14 mianshi-im-api-0.0.1-SNAPSHOT.war
-rw-r--r--. 1 root root    227 5月   4 15:28 start
-rw-r--r--. 1 root root    91 5月   5 10:03 stop
[root@www spring-boot-imapi]#

```

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咕喃接口部署成功：



注：超级管理员账号：1000 初始密码：1000

6、安装 IM-Server socketIM 通讯服务

6.1 包目录说明

IMServer-聊天服务器



lib 目录为 jar 包存放目录

imserver.properties 为服务器配置文件

```
[root@shiku~]# cd /opt
[root@shiku~]# wget http://xxx/soft/kuxin/IMServer.zip
[root@shiku~]# unzip IMServer.zip
[root@shiku~]# cd IMServer
[root@shiku~]# vim imserver.properties
```

6.2 修改 imserver.properties 配置文件

```
bindIp=47.75.89.57 ← 绑定外网IP

#客户端接口链接端口
tcp.port=5666 ← Tcp 客户端端口

websocket.port=5260 ← websocket 客户端端口

isCluster=false ← 是否集群模式

isDebug=true

heartbeatTimeout=0

messageHandlerNameSpace=com.shiku.imserver.message.handler ← 消息处理包名

#Mongodb Properties (数据库配置)
mongoConfig.uri=mongodb://127.0.0.1:28018/imapi ← mongodb 数据库链接配置
mongoConfig.dbName=imapi
mongoConfig.roomDbName=imRoom
mongoConfig.username=
mongoConfig.password=
mongoConfig.connectTimeout=20000
mongoConfig.socketTimeout=20000
mongoConfig.maxWaitTime=20000

mqConfig.nameAddr=localhost:9876
##mq 消费最小程数量 默认 cup 数量
mqConfig.threadMin=4
##mq 消费最大程数量 默认 cup 数量*2
mqConfig.threadMax=8
##mq 批量消费数量 默认 20
mqConfig.batchMaxSize=30

#Redis Properties (缓存配置)
redisConfig.address=redis://127.0.0.1:6379
redisConfig.database=0
redisConfig.password=
redisConfig.pingTimeout=10000
redisConfig.timeout=10000
redisConfig.connectTimeout=10000
redisConfig.pingConnectionInterval=500 ← redis 缓存配置

zkConfig.connectStr=127.0.0.1:2181
zkConfig.sessionTimeoutMs=10000
zkConfig.connectionTimeoutMs=10000 ← zk 配置 集群模式下需要用到
zkConfig.namespace=shikuim
```

6.3 JVM、GC 设置和建议

修改 start 启动文件

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 `-Xmx` JVM 标志

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

HEAP="-Xms10G -Xmx10G"

JAVA_OPTIONS="${GC} ${GC_DEBUG} ${EX} ${HEAP}"

nohup java -Djava.ext.dirs="lib" ${JAVA_OPTIONS} com.shiku.imserver.IMServerStarter imserver.properties > log.log &
```

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：

```
#osgiEnabled=(true|false)
#osgiEnabled=false
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"
#GC_DEBUG=" -XX:+PrintTenuringDistribution -XX:+PrintGCDetails -XX:+PrintGCDateStamps
-XX:+PrintGCTimeStamps -Xloggc:logs/jvm.log -verbose:gc "
HEAP="-Xms60G -Xmx60G"
JAVA_OPTIONS="${GC} ${GC_DEBUG} ${EX} ${HEAP}"
nohup java -Djava.ext.dirs="lib" ${JAVA_OPTIONS} com.shiku.imserver.IMServerStarter imserver.properties > log.log &
```

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 Xmx 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"
#GC_DEBUG=" -XX:+PrintTenuringDistribution -XX:+PrintGCDetails -XX:+PrintGCDateStamps
-XX:+PrintGCTimeStamps -Xloggc:logs/jvm.log -verbose:gc "
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.shiku.imserver.IMServerStarter imserver.properties > log.log &
```

④ 在 IMServer 执行 start 启动 服务

7、安装 shiku-push 推送服务

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

shiku-push 部署包各个文件说明

-  apns_dev.p12 个人版开发证书
-  apns_pro.p12 个人版生产证书
-  apns_push1_dev.p12 企业版开发证书
-  apns_push1_pro.p12 企业版生产证书
-  application.properties 配置文件
-  shiku-push-1.0.war 程序war包
-  start 启动脚本
-  stop 停止脚本
-  voippush.p12 voip 音视频推送证书

安装 shiku-push 推送服务

部署包下载地址: `wget http://xxx/soft/shiku-push.tar`

`[root@shiku~]# tar -xvf shiku-push.tar`

`[root@shiku~]# cd shiku-push`

`[root@shiku~]# vim application.properties`

修改 application.properties 配置文件


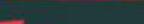


```

70 #XMPP Properties (XMPP主机和端口以及推送用户配置)
71 im.xmppConfig.host=192.168.0.128
72 im.xmppConfig.serverName=im.shiku.ro
73 im.xmppConfig.port=5222
74 im.xmppConfig.username=pusher
75 im.xmppConfig.password=10000
76 im.xmppConfig.dbUri=127.0.0.1:20010
77 im.xmppConfig.dbName=tigase
78 im.xmppConfig.dbUsername=
79 im.xmppConfig.dbPassword=
80
81 im.mqConfig.nameAddr=127.0.0.1:6379

89 #Redis Properties (缓存配置)
90 im.redisConfig.address=redis://192.168.0.100:6379
91 im.redisConfig.database=0
92 im.redisConfig.password=
93 im.redisConfig.pingTimeout=10000
94 im.redisConfig.timeout=10000
95 im.redisConfig.connectTimeout=10000
96 im.redisConfig.pingConnectionInterval=100
97

4 #Mongodb Properties (数据库配置)
5 im.mongoConfig.uri=127.0.0.1:20010
6 im.mongoConfig.dbName=impush
7 im.mongoConfig.roomDbName=imRoom

```

 tigase-server 的连接地址, 这里host 配置为内网ip即可, 注意不能是127.0.0.1
 serverName 和tigase-server 的virt-host保持一致
 数据库地址
 RocketMQ 连接地址
 Redis 连接地址
 数据库连接地址

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 shiku-push 服务即可

```
51 # 消息推送相关配置（小米、华为、魅族等）
52 im.pushConfig.packageName=com.sk.weichat
53
54 # 小米推送
55 im.pushConfig.xm_appSecret=wet...
56
57 # 华为推送
58 im.pushConfig.hw_appSecret=3b6...
59 im.pushConfig.hw_appId=100...
60 im.pushConfig.hw_tokenUrl=https://login.cloud.huawei.com/oauth2/v2/token
61 im.pushConfig.hw_apiUrl=https://api.push.hicloud.com/pushsend.do
62 im.pushConfig.hw_iconUrl=http://pic.qiantucdn.com/58pic/12/38/18/13758PIC4GV.jpg
63
64 ##ios 推送
65 im.pushConfig.betaAppId=com.shiku.coolim.push1
66 im.pushConfig.betaApnsPk=/opt/shiku-push/apns_push1_pro.pl2
67
68 im.pushConfig.appStoreAppId=com.shiku.im.push
69 im.pushConfig.appStoreApnsPk=/opt/shiku-push/apns_pro.pl2
70
71 im.pushConfig.voipPk=/opt/shiku-push/voippush.pl2
72
73 im.pushConfig.pkPassword=123...
74
75 im.pushConfig.isApnsSandbox=0
76
77 im.pushConfig.isDebugEnabled=1
78
79
80 # 极光推送
81 im.pushConfig.jPush_appkey=691...
82 im.pushConfig.jPush_masterSecret=e59...
83
84 # google FCM 推送
85 im.pushConfig.FCM_dataBaseUrl=https://sixth-hawk-164509.firebaseio.com
86 im.pushConfig.FCM_keyJson=/opt/shiku-push/sixth-hawk-164509-firebase-adminsdk-342gn-465351f0ef.json
87
88 #魅族推送
89 im.pushConfig.mz_appId=118...
90 im.pushConfig.mz_appSecret=15c...
91
92 # VIVO推送
93 im.pushConfig.vivo_appId=109...
94 im.pushConfig.vivo_appKey=472...
95 im.pushConfig.vivo_appSecret=127...
96
97 # OPPO推送
98 im.pushConfig.oppo_appKey=dIH...
99 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**。

3.修改完配置后使用 sh start 命令启动 shiku-push 服务

8、安装 message-push 服务

说明: Message-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 Message-push 服务获取
队列里的消息然后把消息经过 tigase-server 发送到设备端

```
[root@shiku~]# cd /opt
[root@shiku~]# wget http://xxx/soft/kuxin/message-push.tar
[root@shiku~]# tar -xvf message-push.tar
[root@shiku~]# cd message-push
[root@shiku~]# vim application.properties
```

参照如下示例, 修改配置文件

```
8  #Mongodb Properties (数据库配置)
9  im.mongoConfig.uri=mongodb://127.0.0.1:28018
10 im.mongoConfig.dbName=imapi
11 im.mongoConfig.roomDbName=imRoom
12 im.mongoConfig.username=
13 im.mongoConfig.password=
14 im.mongoConfig.connectTimeout=20000
15 im.mongoConfig.socketTimeout=20000
16 im.mongoConfig.maxWaitTime=20000
17
18 rocketmq.name-server=127.0.0.1:9876
19 rocketmq.producer.group=group-shikupush
20 rocketmq.producer.send-message-timeout=30000
21 im.mqConfig.nameAddr=localhost:9876
22 ##mq 消费最小程数量 默认 cup 数量
23 #im.mqConfig.threadMin=4
24 ##mq 消费最大程数量 默认 cup 数量*2
25 #im.mqConfig.threadMax=8
26 ##mq 批量消费数量 默认 20
27 #im.mqConfig.batchMaxSize=30
28
29
30 #Redis Properties (缓存配置)
31 im.redisConfig.address=redis://127.0.0.1:6379
32 im.redisConfig.database=0
33 im.redisConfig.password=
34
35 #系统号
36 admin.systemUsers=10001:10001,10002:10002,10003:10003,10004:10004,10005:10005,10006:10006,10007:10007,10008:10008,10009:10009,10010:10010
37
38 im.imConfig.host=127.0.0.1
39 im.imConfig.port=5666
40 im.imConfig.pingTime=10000
41
```

mongodb连接地址

RocketMq连接地址

redis连接地址

IMServer 连接地址, 配置内网ip即可

启动 message-push 服务

```
[root@shiku~]# sh start
```

9、安装 Upload 文件上传服务

```
[root@shiku~]# cd /opt
[root@shiku~]# wget http://xxx/soft/upload.tar
[root@shiku~]# tar -xvf upload.tar
[root@shiku~]# cd upload
[root@shiku~]# vim application.properties
```

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

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

```

4) 启动 upload

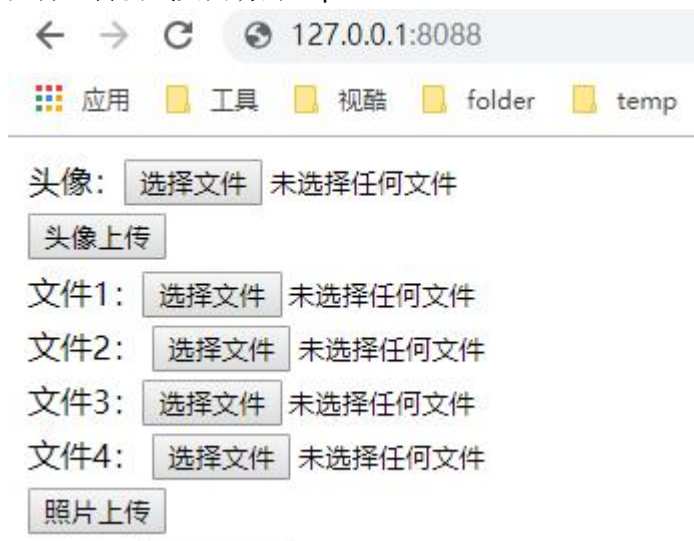
```
[root@shiku~]# Cd /opt/upload
```

```
[root@shiku~]# 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 =====>
```

文件上传测试页面访问 ip:8088



10、安装 Nginx，配置文件访问

首先安装 **nginx** 所需的依赖

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

安装 Nginx-1.9.11

```
[root@shiku~]# cd /opt
[root@shiku~]# wget http://xxx/soft/nginx-1.9.11.tar.gz
[root@shiku~]# tar -zxvf nginx-1.9.11.tar.gz
[root@shiku~]# mv nginx-1.9.11 nginx-install
[root@shiku~]# cd nginx-install
[root@shiku~]# ./configure --prefix=/opt/nginx-1.9.11 --with-http_ssl_module --with-stream
```

```
[root@shiku~]# make && make install
[root@shiku~]# rm -rf ../nginx-install
```

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

start 脚本: vim start , 写入如下内容

```
#!/sbin/nginx

ps -ef|grep nginx
```

stop 脚本: vim stop 写入如下内容

```
#!/sbin/nginx -s stop

ps -ef|grep nginx
```

1) 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。

2) 在 **upload 所在的机器上** 配置头像访问

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。

修改 Nginx 配置文件 nginx.conf , 添加如下内容 (**域名配置示例**)

```
#用户头像访问
server{
    listen          80;
    server_name     head.ouchcloud.com;
    #拒接访问 html 等类型的文件避免受到脚本攻击
    location ~* \.(html|html|jsp|php|js)$ {
        deny all;
    }

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

```
}
```

(**ip 配置示例**)

```
server{
    listen            8089;
    #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:8089

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>”

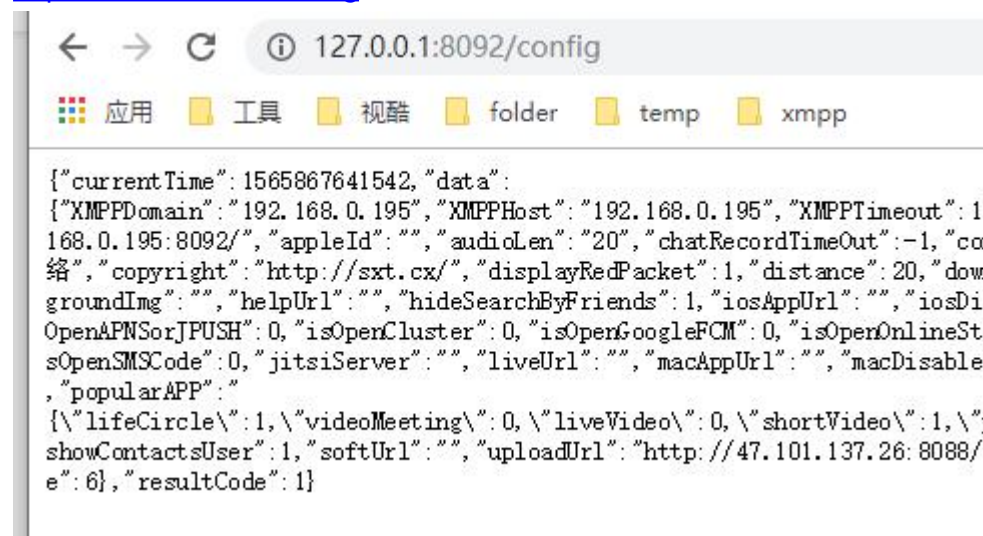
账号密码 默认 1000

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

咕晴管理系统				欢迎你, 超级管理员-系统管理员	
参数说明	参数值	变量名	参数说明		
XMPP 主机 host	127.0.0.1	XMPPHost			
XMPP 虚拟域名	127.0.0.1	XMPPDomain			
接口URL	http://127.0.0.1:9000/	apiUrl			
头像下载URL	http://127.0.0.1:9000/	downloadAvatarUrl			
资源下载URL	http://127.0.0.1:9000/	downloadUrl			
资源上传URL	http://127.0.0.1:9000/	uploadUrl			
视频服务器URL	https://meet.jit.si/	jitsiServer			
短视频拍摄上时长	25	videoLength	秒为单位		
IOS应用AppleId		appleId			

修改配置保存后访问

<http://localhost:8092/config>



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。