# 测试环境

## VPN命令

测试172.16.1.204 /etc/openvpn

线上10.10.10.5 rpm包、/etc/openvpn/easy-

中电线上 192.168.150.62 /etc/openvpn/

vpn签发

source vars

./build-key guancheng

cd key

拉下来guancheng.crt guancheng.key

VPN撤销

.source vars

./revoke-full client2

有error23需要重启vpn，重启线上vpn时，先去防护墙把05的22端口打开

搭建openVPN 最后的路由配置

iptables -t nat -A POSTROUTING -s 192.168.5.0/24 -j SNAT --to-source 10.10.10.15

iptables -t nat -A POSTROUTING -s 192.168.5.0/24 -j MASQUERADE

172.16.1.231

svn

创建库

svnadmin create /opt/svnroot/RiskManagement

启动svn

svnserve -d -r /opt/svnroot

添加权限

vi /data/svnroot/authz.conf

添加密码

htpasswd /data/svnroot/passwd jiawei

关闭库

killall svnserve

svn测试地址

http://172.16.1.231/svn/RiskManagement

## tomcat日志切割

查看cronolog安装后所在目录（验证安装是否成功）

# which cronolog

正常情况下显示：

/usr/local/sbin/cronolog

要想分割tomcat的catalina.out，需作如下工作：

Tomcat7以前的版本：

（1）注释掉（#）

touch “$CATALINA\_BASE”/logs/catalina.out

（2）修改tomcat bin目录下的catalina.sh文件中的两处

org.apache.catalina.startup.Bootstrap “$@” start  \

>> “$CATALINA\_BASE”/logs/catalina.out 2>&1 &

为

org.apache.catalina.startup.Bootstrap "$@" start  2>&1 \

| /usr/local/sbin/cronolog "$CATALINA\_BASE"/logs/catalina.%Y-%m-%d.out >> /dev/null &

完成之后重起Tomcat就可以了。

隔天看logs文件中是否有catalina.2015-09-13.out样式的日志。

Tomcat7以后的版本：

1  第一步

将

if [ -z "$CATALINA\_OUT" ] ; then

CATALINA\_OUT="$CATALINA\_BASE"/logs/catalina.out

fi

修改为

if [ -z "$CATALINA\_OUT" ] ; then

CATALINA\_OUT="$CATALINA\_BASE"/logs/catalina-%Y-%m-%d.out

fi

2  第二步

将

touch "$CATALINA\_OUT"

改为

#touch "$CATALINA\_OUT"

3  第三步

将

org.apache.catalina.startup.Bootstrap "$@" start \

>> "$CATALINA\_OUT"   2>&1 &

修改为

org.apache.catalina.startup.Bootstrap "$@" start 2>&1 \

| /usr/local/sbin/cronolog "$CATALINA\_OUT" >> /dev/null &

完成之后重起Tomcat就可以了。

隔天看logs文件中是否有catalina.out.2015-09-13样式的日志。

## 181服务器外网端口映射

命令一：

iptables -t nat -A PREROUTING -d 192.168.1.181 -p tcp --dport 2001 -j DNAT --to-destination 192.168.1.167:80

命令二：

iptables -t nat -A POSTROUTING -p tcp --dport 80 -j MASQUERADE

## Linux下查看tcp连接数及状态命令

netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'

## 测试环境服务位置

Java环境

后台管理项目 /data/guancheng\_tomcat/tyq-back/

前端pc段 /data/guancheng\_tomcat/tyq-front/

## 字节转换

1MB(兆字节)=1024KB(千字节)  
1KB(千字节)=1024B(字节)   
1MB(兆字节)=1024×1024=1048576B(字节

## Zookeeper伪集群搭建

创建三个目录分别放入三个zk.tar包，解压

修改配置文件，vim conf/zoo.cfg

dataDir=/usr/local/zookeeper/zookeeper2/data

dataLogDir=/usr/local/zookeeper/zookeeper2/log

clientPort=2182

server.0=192.168.1.213:2287:3387

server.1=192.168.1.213:2288:3388

server.2=192.168.1.213:2289:3389

echo 1 > /usr/local/zookeeper/zookeeper2/data/myid 导入id号

开启服务 ./bin/zkServer.sh start

查看集群状态 ./bin/zkServer.sh status

客户端连入 ./bin/zkCli.sh –server 192.168.2.213:2182

## Redis伪集群搭建

创建三个redis目录

解压redis.tar 到目录

yum install gcc-c++ libstdc++-devel zlib-deve gcc

make

make PREFIX=/usr/local/redis install

mkdir conf   
cp /usr/local/redis-3.0.0/redis.conf /usr/local/redis

cd /usr/local/redis   
./bin/redis-server ./redis.conf   
redis默认使用6379端口。

## redis 集群手工搭建及配置

1. 目录结构规划：

redis@192.168.9.1$ mkdir -p /home/redis/6379/{config,data,logs} && mkdir -p /home/redis/6380/{config,data,logs}

1. redis cluster 配置
2. 在 192.168.9.1 服务器中编辑 /home/redis/6379/config/redis-6379.conf，并保存如下内容：

redis@192.168.9.1$ vi /home/redis/6379/config/redis-6379.conf  
####### redis server 配置信息  
port 6379  
maxmemory 4gb  
protected-mode no  
daemonize yes  
dir /home/redis/6379/data  
pidfile /home/redis/6379/data/redis-6379.pid

logfile /home/redis/6379/logs/redis-6379.log  
tcp-backlog 511  
timeout 300  
tcp-keepalive 60  
loglevel notice  
cluster-enabled yes  
cluster-node-timeout 15000  
cluster-config-file "/home/redis/6379/nodes-6379.conf"

1. 在 192.168.9.1 服务器中编辑 /home/redis/6380/config/redis-6380.conf，并保存如下内容：

redis@192.168.9.1$ vi /home/redis/6380/config/redis-6380.conf  
####### redis server 配置信息

port 6380  
maxmemory 4gb  
protected-mode no  
daemonize yes  
dir /home/redis/6380/data  
pidfile /home/redis/6380/data/redis-6380.pid  
logfile /home/redis/6380/logs/redis-6380.log  
tcp-backlog 511  
timeout 300  
tcp-keepalive 60  
loglevel notice  
cluster-enabled yes  
cluster-node-timeout 15000  
cluster-config-file "/home/redis/6380/nodes-6380.conf"

1. 将 192.168.9.1 服务器上 /home/redis/bin、/home/redis/6379、/home/redis/6380，分发到另外 2 台服务器中

redis@192.168.9.1$ scp -r /home/redis/bin /home/redis/6379 /home/redis/6380 redis@192.168.9.2:/home/redis  
redis@192.168.9.1$ scp -r /home/redis/bin /home/redis/6379 /home/redis/6380 redis@192.168.9.3:/home/redis

1. 启动所有 redis 节点

redis@192.168.9.1$ cd ~/bin && ./redis-server ../6379/config/redis-6379.conf  
redis@192.168.9.1$ cd ~/bin && ./redis-server ../6380/config/redis-6380.conf

redis@192.168.9.2$ cd ~/bin && ./redis-server ../6379/config/redis-6379.conf  
redis@192.168.9.2$ cd ~/bin && ./redis-server ../6380/config/redis-6380.conf  
redis@192.168.9.3$ cd ~/bin && ./redis-server ../6379/config/redis-6379.conf  
redis@192.168.9.3$ cd ~/bin && ./redis-server ../6380/config/redis-6380.conf

1. 节点间握手：

这一步可以在任意节点上进行，这里以 192.168.9.1:6379 为例

1. redis@192.168.9.1$ ./redis-cli -h 192.168.9.1 -p 6379
2. 192.168.9.1:6379> cluster meet 192.168.9.1 6380
3. 192.168.9.1:6379> cluster meet 192.168.9.2 6379
4. 192.168.9.1:6379> cluster meet 192.168.9.2 6380
5. 192.168.9.1:6379> cluster meet 192.168.9.3 6379
6. 192.168.9.1:6379> cluster meet 192.168.9.3 6380
7. 分配 hash 槽：

这里我们规划三台服务器上端口奇数节点为 master 节点，端口偶数节点为 slave 节点

1. redis@192.168.9.1$ cd ~/bin && ./redis-cli -h 192.168.9.1 -p 6379 cluster addslots {0..5461}
2. redis@192.168.9.2$ cd ~/bin && ./redis-cli -h 192.168.9.2 -p 6379 cluster addslots {5462..10922}
3. redis@192.168.9.3$ cd ~/bin && ./redis-cli -h 192.168.9.3 -p 6379 cluster addslots {10923..16383}

查看集群节点信息：

1. redis@192.168.9.1$ ./redis-cli -h 192.168.9.1 -p 6379

1. 192.168.9.1:6379> cluster nodes

登陆各个从节点对主节点进行复制：\*\* 这里我们规划 192.168.9.1:6380 复制 192.168.9.2:6379，192.168.9.2:6380 复制 192.168.9.3:6379，192.168.9.3:6380 复制 192.168.9.1:6379\*\*

1. redis@192.168.9.1$ ./redis-cli -h 192.168.9.1 -p 6380
2. 192.168.9.1:6380> cluster replicate NODEID **注意这里的 NODEID 并不是 redis 节点的 ip:port，而是从节点要复制的 redis master 节点的 hash 值即 NODEID**
3. redis@192.168.9.2$ ./redis-cli -h 192.168.9.2 -p 6380
4. 192.168.9.2:6380> cluster replicate NODEID
5. redis@192.168.9.3$ ./redis-cli -h 192.168.9.3 -p 6380
6. 192.168.9.3:6380> cluster replicate NODEID

## supervisor搭建时配置

### 监控tomcat配置

(主要的命令配置)

command=/opt/atlassian/confluence/bin/catalina.sh run

### 监控zookeeper配置

Vim /usr/local/zookeeper-3.4.10/bin/startZK.sh

#!/bin/bash

function shutdown()

{

date

echo "Shutting down ZK"

unset CATALINA\_PID # Necessary in some cases

/usr/local/zookeeper-3.4.10/bin/zkServer.sh stop

}

date

echo "Starting ZK"

#. /data/apps/website1/bin/catalina.sh start -security

/usr/local/zookeeper-3.4.10/bin/zkServer.sh start

# Allow any signal which would kill a process to stop Tomcat

trap shutdown HUP INT QUIT ABRT KILL ALRM TERM TSTP

echo "Waiting for `cat $CATALINA\_PID`"

wait `cat $CATALINA\_PID`

vim /etc/supervisor.d/zookeeper.ini

[program:zookeeper]

#directory=/usr/local/zookeeper-3.4.10

command=/usr/local/zookeeper-3.4.10/bin/startZK.sh #调用脚本启动

#stdout\_logfile=/opt/logs/zookeeper.log

autostart=true

autorestart=true

startsecs=5

priority=1

stopasgroup=true

killasgroup=true

systemctl restart supervisord.service

## 命令

ps axo psr，comm 查看那个进程运行在哪个cpu（核心）上

lsof

## tcpdump命令使用

tcpdump [ –q ] –XX –vvv -i em1 -nn tcp port 443 and src 1.180.206.192

# MySQL命令

## mysqldump命令

mysqldump --single-transaction --set-gtid-purged=off -uroot -p clb grade\_item grade\_template\_item\_score\_correlation > bk/clb\_20180412\_s.sql

mysql –u root –p clb < clb\_20180412\_s.sql

如果不想导出gtid信息可加入 --set-gtid-purged=off 参数(阿里云MySQL数据库)

mysqldump -h rm-2zeaxabm0b7pu8675rw.mysql.rds.aliyuncs.com -uqmzb -pQmzb2017 --opt --default-character-set=utf8 --extended-insert=false --triggers --hex-blob --set-gtid-purged=off qmzb\_activity > /tmp/qmzb\_activity.sql

## 怎样用命令查看Mysql中某个数据库的大小？

* **摘要：**要想知道每个数据库的大小的话,步骤如下:1、进入information\_schema数据库(存放了其他的数据库的信息)useinformation\_schema;2、查询所有数据的大小:selectconcat(round(sum(data\_length/1024/1024),2),'MB')asdatafromtables;3、查看指定数据库的大小:比如查看数据库home的大小selectconcat(round(sum(data\_length/1024/1024),2),
* 要想知道每个数据库的大小的话,步骤如下:  
    
  1、进入information\_schema 数据库(存放了其他的数据库的信息)  
  use information\_schema;  
    
  2、查询所有数据的大小:  
  select concat(round(sum(data\_length/1024/1024),2),'MB') as data from tables;  
    
  3、查看指定数据库的大小:  
  比如查看数据库home的大小  
  select concat(round(sum(data\_length/1024/1024),2),'MB') as data from tables where table\_schema='home';  
    
  4、查看指定数据库的某个表的大小  
  比如查看数据库home中 members 表的大小  
  select concat(round(sum(data\_length/1024/1024),2),'MB') as data from tables where table\_schema='home' and table\_name='members';

## CPU优化

taskset [options] [mask | cpu-list] [pid|cmd [args...]]

taskset –p –c 0，3 12212 把12212进程固定到第0和3颗CPU上，重启后失效

## yum拷贝缓存包命令

mkdir /opt/rpm

find /var/cache/yum/x86\_64/ -name \*.rpm -exec cp -a {} /opt/rpm \;

## flume-ng 搭建

#### 下载tar包解压

编辑配置文件：

Vim /usr/local/conf/example.conf

# example.conf: A single-node Flume configuration

# Name the components on this agent

a1.sources = r1

a1.sinks = k1

a1.channels = c1

# Describe/configure the source

a1.sources.r1.type = netcat

a1.sources.r1.bind = localhost

a1.sources.r1.port = 44444

# Describe the sink

a1.sinks.k1.type = logger

# Use a channel which buffers events in memory

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

#### 启动程序：

bin/flume-ng agent --conf conf --conf-file conf/single-node.conf --name a1 -Dflume.root.logger=INFO,console

JDK监控

Jmap [option] [pid] 查看此进程的各种内存使用情况

查看堆空间的详细信息：

jmap -heap <pid>

Jps –m –l 查看运行jvm的进程为哪个

Jstat：jvm监控统计工具

Jstat –options 查看帮助、获取可选参数

jstat -gc pid 查看此进程运用的内存状态

s0c、s1c、s0u、s1u：c表示容量、u表示已用量

EC，EU：eden区域的容量和已用量

pc、pu:

YGC：新生代的垃圾回收次数；

YGCT：新生代垃圾回收消耗的时长；

FGC：Full GC的次数；

FGCT：Full GC消耗的时长；

GCT：GC消耗的总时长；

图形化监控工具

Jvisualvam

jconsole

以-XX表示的非Stable参数，虽然在官方文档中是不确定的，不健壮的，各个公司的实现也各有不同，但往往非常实用，所以这部分参数对于GC非常重要。

请看一下一个时间的Java参数配置：（服务器：Linux 64Bit，8Core×16G）

export  **JAVA\_OPTS="$JAVA\_OPTS -server -Xms3G -Xmx3G -Xss256k -XX:PermSize=128m -XX:MaxPermSize=128m -XX:+UseParallelOldGC -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/usr/aaa/dump -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -Xloggc:/usr/aaa/dump/heap\_trace.txt -XX:NewSize=1G -XX:MaxNewSize=1G"**

vim ./bin/catalina.sh

JAVA\_OPTS="

-Xms6000M

-Xmx6000M

-XX:NewSize=2250M

-XX:MaxNewSize=2250M

-XX:PermSize=128M

-XX:MaxPermSize=256M

-XX:LargePageSizeInBytes=128m"

{ CATALINA\_OPTS="

-server

-Xms6000M

-Xmx6000M

-Xss512k

-XX:NewSize=2250M

-XX:MaxNewSize=2250M

-XX:PermSize=128M

-XX:MaxPermSize=256M

-XX:+AggressiveOpts

-XX:+UseBiasedLocking

-XX:+DisableExplicitGC

-XX:+UseParNewGC

-XX:+UseConcMarkSweepGC

-XX:MaxTenuringThreshold=31

-XX:+CMSParallelRemarkEnabled

-XX:+UseCMSCompactAtFullCollection

-XX:LargePageSizeInBytes=128m

-XX:+UseFastAccessorMethods

-XX:+UseCMSInitiatingOccupancyOnly

-Duser.timezone=Asia/Shanghai

-Djava.awt.headless=true" }

## jdk内存修改相关选项：（以下都可选择）

#### ****1 性能相关****

-XX:-UseBiasedLocking -XX:-UseCounterDecay -XX:AutoBoxCacheMax=20000 -XX:+PerfDisableSharedMem -XX:+AlwaysPreTouch -Djava.security.egd=file:/dev/./urandom

#### ****2 内存大小相关(JDK7)****

-Xms4096m -Xmx4096m -Xmn2048m -XX:MaxDirectMemorySize=4096m-XX: PermSize=256m -XX:MaxPermSize=512m -XX:ReservedCodeCacheSize=240M

#### ****3 CMS GC 相关****

-XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -XX:MaxTenuringThreshold=6 -XX:+ExplicitGCInvokesConcurrent -XX:+ParallelRefProcEnabled

#### ****4 GC 日志 相关****

-Xloggc:/dev/shm/app-gc.log -XX:+PrintGCApplicationStoppedTime -XX:+PrintGCDateStamps -XX:+PrintGCDetails

#### ****5 异常 日志 相关****

-XX:-OmitStackTraceInFastThrow -XX:ErrorFile=${LOGDIR}/hs\_err\_%p.log -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${LOGDIR}/

## 邮件服务发送设置

Vim /etc/mail.rc

#加入以下三行

set from=13597476155@139.com smtp=smtp.139.com #设定服务器

set smtp-auth-user=laogong666 smtp-auth-password=passwd #登录

set smtp-auth=login

发送邮件设置

mail –s “projeck” [gonglong@guanchenglianhe.com](mailto:gonglong@guanchenglianhe.com) 输入内容，Ctrl+D结束

# 线上服务器

一般应用在/usr/local下

一般项目都在/opt 下

一般的数据都在/data 下

投友圈Java项目

## 配置文件

## Zookeeper配置

Vim /usr/local/zookeeper/conf/zoo.cfg

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/data/zookeeper

# the port at which the clients will connect

clientPort=2181

# the maximum number of client connections.

# increase this if you need to handle more clients

maxClientCnxns=200

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

dataLogDir=/data/zookeeper/logs

preAllocSize=50M #为事务日志分配的磁盘大小

snapCount=5000 #每多少次事务进行一次快照操作

server.1=10.10.10.13:28888:38888

server.2=10.10.10.16:28888:38888

server.3=10.10.10.17:28888:38888

# The number of snapshots to retain in dataDir

autopurge.snapRetainCount=30

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

## MQ配置

Vim /usr/local/activemq/conf/activemq.xml

主从都注释此三行；

<!-- <persistenceAdapter>

<kahaDB directory="${activemq.data}/kahadb"/>

</persistenceAdapter>

-->

添加此几行；

<persistenceAdapter>

<replicatedLevelDB

directory="${activemq.data}/leveldb"

replicas="2"

bind="tcp://0.0.0.0:0"

zkAddress="10.10.10.13:2181,10.10.10.16:2181,10.10.10.17:2181"

hostname="10.10.10.17"

sync="local\_disk"

zkPath="/activemq/leveldb-stores"

/>

</persistenceAdapter>

## Redis主从配置

主配置文件

Vim redis.conf

bind 10.10.10.17

protected-mode yes

port 6379

tcp-backlog 511

timeout 0

tcp-keepalive 300

daemonize yes

supervised no

pidfile /var/run/redis\_6379.pid

loglevel notice

logfile "./redis-log"

databases 16

#RDB config

save 900 1

save 300 10

save 60 10000

stop-writes-on-bgsave-error yes

rdbcompression yes

rdbchecksum yes

dbfilename dump.rdb

dir ./

slave-serve-stale-data yes

slave-read-only yes

repl-diskless-sync no

repl-diskless-sync-delay 5

repl-disable-tcp-nodelay no

slave-priority 10

#requirepass 111111

maxclients 100000

maxmemory 16G

#AOF config

appendonly no

appendfilename "appendonly.aof"

#appendfsync everysec

appendfsync always

no-appendfsync-on-rewrite no

auto-aof-rewrite-percentage 100

auto-aof-rewrite-min-size 64mb

aof-load-truncated yes

lua-time-limit 5000

slowlog-log-slower-than 10000

slowlog-max-len 128

latency-monitor-threshold 0

notify-keyspace-events ""

hash-max-ziplist-entries 512

hash-max-ziplist-value 64

list-max-ziplist-size -2

list-compress-depth 0

set-max-intset-entries 512

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

hll-sparse-max-bytes 3000

activerehashing yes

client-output-buffer-limit normal 0 0 0

client-output-buffer-limit slave 256mb 64mb 60

client-output-buffer-limit pubsub 32mb 8mb 60

hz 10

aof-rewrite-incremental-fsync yes

从配置文件

Vim redis.conf

bind 10.10.10.16

protected-mode yes

port 6379

tcp-backlog 511

timeout 0

tcp-keepalive 300

daemonize yes

supervised no

pidfile /var/run/redis\_6379.pid

loglevel notice

logfile "./redis-log"

slaveof 10.10.10.17 6379

#masterauth 111111

save 900 1

save 300 10

rdbcompression yes

rdbchecksum yes

dbfilename dump.rdb

dir ./

slave-serve-stale-data yes

slave-read-only yes

repl-diskless-sync no

repl-diskless-sync-delay 5

repl-disable-tcp-nodelay no

slave-priority 100

#requirepass 111111

maxclients 100000

maxmemory 16G

#AOF config

appendonly no

appendfilename "appendonly.aof"

appendfsync everysec

no-appendfsync-on-rewrite no

auto-aof-rewrite-percentage 100

auto-aof-rewrite-min-size 64mb

aof-load-truncated yes

lua-time-limit 5000

slowlog-log-slower-than 10000

slowlog-max-len 128

latency-monitor-threshold 0

notify-keyspace-events ""

hash-max-ziplist-entries 512

hash-max-ziplist-value 64

list-max-ziplist-size -2

list-compress-depth 0

set-max-intset-entries 512

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

hll-sparse-max-bytes 3000

activerehashing yes

client-output-buffer-limit normal 0 0 0

client-output-buffer-limit slave 256mb 64mb 60

client-output-buffer-limit pubsub 32mb 8mb 60

hz 10

aof-rewrite-incremental-fsync yes

"redis.conf" 66L, 1311C written

[root@tyq16 redis-slave]# cat redis.conf

bind 10.10.10.16

protected-mode yes

port 6379

tcp-backlog 511

timeout 0

tcp-keepalive 300

daemonize yes

supervised no

pidfile /var/run/redis\_6379.pid

loglevel notice

logfile "./redis-log"

databases 16

slaveof 10.10.10.17 6379

#masterauth 111111

#RDB config

save 900 1

save 300 10

save 60 10000

stop-writes-on-bgsave-error yes

rdbcompression yes

rdbchecksum yes

dbfilename dump.rdb

dir ./

slave-serve-stale-data yes

slave-read-only yes

repl-diskless-sync no

repl-diskless-sync-delay 5

repl-disable-tcp-nodelay no

slave-priority 100

#requirepass 111111

maxclients 100000

maxmemory 16G

#AOF config

appendonly no

appendfilename "appendonly.aof"

appendfsync everysec

no-appendfsync-on-rewrite no

auto-aof-rewrite-percentage 100

auto-aof-rewrite-min-size 64mb

aof-load-truncated yes

lua-time-limit 5000

slowlog-log-slower-than 10000

slowlog-max-len 128

latency-monitor-threshold 0

notify-keyspace-events ""

hash-max-ziplist-entries 512

hash-max-ziplist-value 64

list-max-ziplist-size -2

list-compress-depth 0

set-max-intset-entries 512

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

hll-sparse-max-bytes 3000

activerehashing yes

client-output-buffer-limit normal 0 0 0

client-output-buffer-limit slave 256mb 64mb 60

client-output-buffer-limit pubsub 32mb 8mb 60

hz 10

aof-rewrite-incremental-fsync yes

redis 启动停止

启动 nohup ./src/redis-server redis.conf &

停止 kill -15 pid

## Gitlab搭建

#centos6

#-------安装gitlab-------

yum install curl openssh-server openssh-clients postfix cronie

service postfix start

chkconfig postfix on

lokkit -s http -s ssh

添加源

[gitlab-ce]

name=gitlab-ce

baseurl=http://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el6

repo\_gpgcheck=0

gpgcheck=0

enabled=1

gpgkey=https://packages.gitlab.com/gpg.key

#----安装----

yum install gitlab-ce

更改项目的IP

vi /var/opt/gitlab/gitlab-rails/etc/gitlab.yml

gitlab:

## Web server settings (note: host is the FQDN, do not include http://)

host: #改为服务器IP或域名

port: 80

https: false

启动

gitlab-ctl reconfigure

重启

gitlab-ctl restart

## MySQL配置

Vim /etc/my.cnf

笔记

## Tomcat常用配置

tomcat整理文档

1、常用操作

启动: ./bin/startup.sh

停止： ps -ef | grep cfl-app[项目名]

日志：tomcat默认日志为logs/catalina.out，有的做了日志切割，logs/catalina.[日期].out

查看日志: tail -f logs/catalina.out

2、日志切割

更改 vim bin/catalina.sh

/CATALINA\_OUT 搜索

#touch "$CATALINA\_OUT"

org.apache.catalina.startup.Bootstrap "$@" start 2>&1 | /usr/local/cronolog/sbin/cronolog "$CATALINA\_BASE"/logs/catalina.%Y-%m-%d.out >> /dev/null &

#org.apache.catalina.startup.Bootstrap "$@" start \

#>> "$CATALINA\_OUT" 2>&1 "&"

3、日志切割脚本

#!/bin/bash

# tomcat log cutting

TOMCATDIR=""

SERVICE=""

TOMCATPID=`ps -ef | grep [j]ava | grep $SERVICE | awk '{print $2}'`

if [ ! -d /opt/$SERVICE/logs\_bak ]

then

mkdir /opt/$SERVICE/logs\_bak

fi

cp ca ca-`date +%Y%m%d`.log

>ca.out

cp $TOMCATDIR/logs/ca-`date +%Y%m%d`.log $TOMCATDIR/logs\_bak/

#ping脚本

for i in {1..254}

do

(

ping -c 1 192.168.1.$i &> /dev/null;

if [ $? -eq 0 ]; then

echo $i

fi

) &

done

wait

4、关闭默认日志

vim conf/server.xml

<!--<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="localhost\_access\_log" suffix=".txt"

pattern="%h %l %u %t &quot;%r&quot; %s %b" /> -->

vim conf/logging.properties

level = WARRING

其他全部OFF

5、部署新tomcat

vim conf/server.xml

用《阿里云线上tomcat》的配置文件

更改端口号

更改项目路径

做日志切割、关闭默认日志

## 搭建VPN配置

[root@bogon config]# cat server.conf

port 1163 #指定监听的本机端口号

proto tcp #指定采用的传输协议，可以选择tcp或udp

dev tun #指定创建的通信隧道类型，可选tun或tap

ca ca.crt #指定CA证书的文件路径

cert app\_server.crt #指定服务器端的证书文件路径

key app\_server.key #指定服务器端的私钥文件路径

dh dh1024.pem #指定迪菲赫尔曼参数的文件路径

server 192.168.2.0 255.255.255.0 #指定虚拟局域网占用的IP地址段和子网掩码，此处配置的服务器自身占用10.0.0.1。

push "route 192.168.2.0 255.255.255.0"

push "route 0.0.0.0 0.0.0.0"

push "route 192.168.1.0 255.255.255.0" ifconfig-pool-persist ipp.txt #服务器自动给客户端分配IP后，客户端下次连接时，仍然采用上次的IP地址(第一次分配的IP保存在ipp.txt中，下一次分配其中保存的IP)。

keepalive 10 120 #每10秒ping一次，连接超时时间设为120秒。

comp-lzo #开启VPN连接压缩，如果服务器端开启，客户端也必须开启

client-to-client #允许客户端与客户端相连接，默认情况下客户端只能与服务器相连接

persist-key

persist-tun #持久化选项可以尽量避免访问在重启时由于用户权限降低而无法访问的某些资源。

status openvpn-status.log #指定记录OpenVPN状态的日志文件路径

verb 3

#auth-user-pass-verify /usr/local/openvpn/config/checkpsw.sh via-env

#client-cert-not-required

#username-as-common-name

duplicate-cn

[root@bogon 2.0]# egrep -v "^$|^#|^;" vars

export EASY\_RSA="`pwd`"

export OPENSSL="openssl"

export PKCS11TOOL="pkcs11-tool"

export GREP="grep"

export KEY\_CONFIG=`$EASY\_RSA/whichopensslcnf $EASY\_RSA`

export KEY\_DIR="$EASY\_RSA/keys"

echo NOTE: If you run ./clean-all, I will be doing a rm -rf on $KEY\_DIR

export KEY\_SIZE=1024

export CA\_EXPIRE=3650

export KEY\_EXPIRE=3650

export KEY\_COUNTRY="CN"

export KEY\_PROVINCE="BJ"

export KEY\_CITY="Beijing"

export KEY\_ORG="zcb\_admin"

export KEY\_EMAIL="hexinlei@zhongchoubao.com"

[root@bogon ~]# cat /etc/sysconfig/iptables

# Generated by iptables-save v1.4.7 on Fri Jul 21 14:01:47 2017

\*nat

:PREROUTING ACCEPT [3:513]

:POSTROUTING ACCEPT [0:0]

:OUTPUT ACCEPT [0:0]

-A POSTROUTING -s 192.168.2.0/24 -o eth0 -j MASQUERADE

COMMIT

# Completed on Fri Jul 21 14:01:47 2017

# Generated by iptables-save v1.4.7 on Fri Jul 21 14:01:47 2017

\*filter

:INPUT ACCEPT [0:0]

:FORWARD ACCEPT [0:0]

:OUTPUT ACCEPT [44:5128]

-A INPUT -j ACCEPT

COMMIT

# Completed on Fri Jul 21 14:01:47 2017

openvpn参数配置详解

http://blog.chinaunix.net/uid-21385796-id-139273.html

openvpn /etc/openvpn/jenkins\_online.ovpn >> /var/log/openvpn/`date +%Y%m%d`.log &

scp -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key /var/lib/jenkins/workspace/${JOB\_NAME}/cfl-admin/target/cfl-admin.war 10.10.10.2:/home/online\_data/

ssh -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key -t root@10.10.10.2 'sh ~/scripts/cfl-admin.sh '

sleep 5 && ps -ef | grep [o]penvpn | awk '{print $2}' | xargs kill -9

#!/bin/bash

# build cfl-admin

source /etc/profile

ps -ef | grep [j]ava | grep cfl-admin | awk '{print $2}' | xargs kill -9

rm -rf /opt/cfl-admin/webapps/\*

rm -rf /opt/cfl-admin/work/\*

cp /home/online\_data/cfl-admin.war /opt/cfl-admin/webapps/

echo "启动tomcat"

/opt/cfl-admin/bin/startup.sh && [ $? -eq 0 ] && echo "tomcat启动成功"

## jinkens登录免秘钥登录

scp -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key /var/lib/jenkins/jobs/${JOB\_NAME}/workspace/target/creditloan-web.war 192.168.1.181:/home/test\_data/

ssh -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key -t root@192.168.1.181 'sh ~/scripts/deproside.sh'

vim deproside.sh

#!/bin/bash

# build deproside

source /etc/profile

dir=depository #jar包的项目名

ls /home/$dir >& /dev/null #判断项目的家目录是否生成

[ $? -eq 0 ] || mkdir /home/$dir

mkdir -pv /data/guancheng\_tomcat/"$dir"/webapps\_bak/"$dir"`date +%F%T` | grep -o "$dir"2.\*[0-9] &> /tmp/$dir #生成备份文件目录

bak\_dir\_web=`cat /tmp/"$dir"` #最后一个备份文件目录查看

ps -ef | grep $dir |sed -n '1p' | awk '{print $2}' | xargs kill -9 >& /dev/null #杀掉项目进程

cp -rf /data/guancheng\_tomcat/$dir/$dir.jar /data/guancheng\_tomcat/$dir/webapps\_bak/$bak\_dir\_web/ #备份原项目jar包

if [ $? -eq 0 ] #判断此次备份成功

then

rm -rf /data/guancheng\_tomcat/$dir/$dir.jar #删除原jar包

cp /home/$dir/$dir.jar /data/guancheng\_tomcat/$dir/ #复制jenkins打包的文件到项目目录下

if [ $? -ne 0 ] #判断此次备份不成功

then

cp -rf /data/guancheng\_tomcat/$dir/webapps\_bak/$bak\_dir\_web/\* /data/guancheng\_tomcat/$dir/ #回滚

fi

echo "启动tomcat"

# /data/guancheng\_tomcat/$dir/bin/startup.sh && [ $? -eq 0 ] && echo "tomcat启动成功"

cd /data/guancheng\_tomcat/$dir

nohup java -jar $dir.jar > nohup.out 2>&1 &

[ $? -eq 0 ] && echo "项目启动成功1"

else

echo "备份失败，重新启动tomcat，项目未部署"

# /data/guancheng\_tomcat/$dir/bin/startup.sh && [ $? -eq 0 ] && echo "tomcat启动成功"

cd /data/guancheng\_tomcat/$dir

nohup java -jar $dir.jar > nohup.out 2>&1 &

[ $? -eq 0 ] && echo "项目回滚启动成功2"

fi

scp -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key /var/lib/jenkins/jobs/${JOB\_NAME}/workspace/target/moxie-service.war 192.168.1.181:/home/test\_data/

ssh -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key -t root@192.168.1.181 'sh ~/scripts/tyq-front.sh '

vim tyq-front.sh

#!/bin/bash

# build cfl-app

source /etc/profile

dir=tyq-front #项目目录名

ls /home/$dir &> /dev/null #判断项目的家目录是否生成

[ $? -eq 0 ] || mkdir /home/$dir

mkdir -pv /data/guancheng\_tomcat/$dir/webapps\_bak/$dir`date +%F%T` | grep -o "$dir"2.\*[0-9] &> /tmp/$dir #生成备份目录

bak\_dir\_web=`cat /tmp/$dir` #查看最后一次备份的目录名

ps -ef | grep [j]ava | grep $dir | awk '{print $2}' | xargs kill -9 #杀掉项目进程

cp -rf /data/guancheng\_tomcat/$dir/webapps/\*.war /data/guancheng\_tomcat/$dir/webapps\_bak/$bak\_dir\_web #备份项目的war包

if [ $? -eq 0 ] #判断此次备份成功

then

rm -rf /data/guancheng\_tomcat/$dir/webapps/\*

rm -rf /data/guancheng\_tomcat/$dir/work/\*

cp /home/$dir/\*.war /data/guancheng\_tomcat/$dir/webapps/ROOT.war #复制jenkins打包好的文件到项目目录下，并改名为ROOT.war

if [ $? -ne 0 ] #判断此次备份不成功

then

cp -rf /data/guancheng\_tomcat/$dir/webapps\_bak/$bak\_dir\_web/\*.war /data/guancheng\_tomcat/$dir/webapps/ #回滚

fi

echo "启动tomcat"

/data/guancheng\_tomcat/$dir/bin/startup.sh && [ $? -eq 0 ] && echo "tomcat启动成功1"

else

echo "备份失败，重新启动tomcat，项目未部署"

/data/guancheng\_tomcat/$dir/bin/startup.sh && [ $? -eq 0 ] && echo "tomcat启动成功2"

fi

线上web-api

openvpn /etc/openvpn/moxie/jenkins\_online.ovpn >> /var/log/openvpn/`date +%Y%m%d`.log &

scp -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key /var/lib/jenkins/jobs/${JOB\_NAME}/workspace/target/creditloan-web.war 10.10.10.2:/home/online\_data/

ssh -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key -t root@10.10.10.2 'sh ~/scripts/creditloan-web-api.sh '

sleep 5 && ps -ef | grep [o]penvpn | grep jenkins | awk '{print $2}' | xargs kill -9

线上moxie\_service

#openvpn /etc/openvpn/moxie/jenkins\_online.ovpn >> /var/log/openvpn/`date +%Y%m%d`.log &

scp -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key /var/lib/jenkins/jobs/${JOB\_NAME}/workspace/target/moxie-service.war 10.10.10.2:/home/online\_data/

ssh -o "StrictHostKeyChecking no" -i ~/.ssh/ssh\_key -t root@10.10.10.2 'sh ~/scripts/moxie-service.sh '

#sleep 5 && ps -ef | grep [o]penvpn | grep jenkins | awk '{print $2}' | xargs kill -9

# Nginx脚本提取pv和uv

Nginx日志格式

192.165.158.238 - - 2017-03-06T20:47:04+08:00 "GET http://download.helloworld.com/ HTTP/1.1" 200 851 425 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36" "-" 0.000 -

## 用shell脚本提取

# awk 'END{print "PV is:",NR}' access.log

PV is: 1881955

# awk '{s[$1]+=1} END{for(i in s){sum+=1}} END{print "UV is:",sum}' access.log

UV is: 64953

# awk '{s[$6]+=1} END{for(i in s){print s[i],i}}' access.log  | sort -rn | head -10

# 只打印出访问次数最多的10条记录

92838 http://download.helloworld.com/hello/hello

88873 http://download.helloworld.com/world/hi/

57711 http://appy.helloworld.com/world/js/jquery-1.10.1.min.js

46980 http://download.helloworld.com/favicon.ico

38759 http://appy.helloworld.com/world/css/style.css?t=00001

38684 http://appy.helloworld.com/world/css/base.css

35404 http://appy.helloworld.com/favicon.ico

34907 http://download.helloworld.com/world/js/jquery-1.10.1.min.js

34882 http://appy.helloworld.com/world/img/hi.jpg

34445 <http://download.helloworld.com/world/css/base.css>