# Redis

## 预生产

192.168.51.241映射到外网 redis156.16.32.37 6379 在这个地址可以访问156.16.42.199

[weblogic@ysc-mhjms-app ~]$ ps -ef | grep redis

root 802 1 0 2017 ? 01:26:23 redis-server \*:6379

root 903 1 0 2017 ? 01:20:02 redis-server \*:6380

root 1027 1 0 2017 ? 01:20:08 redis-server \*:6381

root 2552 1 0 2017 ? 02:11:45 redis-sentinel \*:26379 [sentinel]

root 2716 1 0 2017 ? 02:11:45 redis-sentinel \*:26380 [sentinel]

weblogic 25724 25693 0 11:05 pts/3 00:00:00 grep redis

三主两哨兵

## 生产

192.169.51.2 映射到外网 redis156.16.32.35 6379

[sxlt@sc-ydwx-app1 ~]$ ps -ef | grep redis

sxlt 5492 5460 0 11:08 pts/2 00:00:00 grep redis

root 25472 1 0 2017 ? 02:32:53 redis-server \*:6379

root 25974 1 0 2017 ? 01:34:47 redis-server \*:6380

root 26085 1 0 2017 ? 01:38:21 redis-server \*:6381

root 27487 1 0 2017 ? 02:32:20 redis-sentinel \*:26379 [sentinel]

root 27927 1 0 2017 ? 02:31:57 redis-sentinel \*:26380 [sentinel]

三主两哨兵

## Redis清空

Root 进入192.168.51.35/software

redis-cli

auth Sxds@2017

flushall

# Weblogic

## 安装

### 1关闭防火墙

**/etc/init.d/iptables stop**

**chkconfig iptables off**

### 12设置服务器名

**hostname sxstscmhwwap01**

**Vim /etc/sysconfig/network**

**HOSTNAME=test01**

### 增加服务器名与IP 对应关系。

vi /etc/hosts

**192.168.51.35 test01**

### 检查服务器系统参数

**使用命令： ulimit -a ，查看open files，max user processes等参数是否已经优化。**

**如未优化，则根据\trunk\engineering\doc\5\_system\_platform\操作系统安装配制\操作系统检查与参数优化.txt**

**文档进行操作。**

### 创建weblogic用户与weblogic组

1）创建weblogic用户与weblogic组。

**用root用户登录服务器，创建weblogic用户与weblogic组，使用如下命令：**

**groupadd weblogic**

**useradd -g weblogic weblogic**

**passwd weblogic**

**mkdir -p /weblogic**

**chown -R weblogic:weblogic /weblogic**

**chmod -R a+xrw /weblogic**

### 2）使用weblogic用户上传weblogicsetup安装包

### /home/weblogic/目录。

如果没有使用weblogic用户执行

**chown -R weblogic.weblogic weblogicsetup**

### Jdk卸载安装

rpm -qa | grpe jdk

rpm -e java-1.7.0-openjdk-1.7.0.79-2.5.5.4.el6.x86\_64 --nodeps

vim /etc/profile

**#set java environment**

**JAVA\_HOME=/usr/java/sunjava6\_64**

**JRE\_HOME=$JAVA\_HOME/jre**

**CLASS\_PATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib**

**PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin**

**export JAVA\_HOME**

**export JRE\_HOME**

**export CLASS\_PATH**

**export PATH**

source /etc/profile

java -version

### 正式安装

**chmod +x /home/weblogic/weblogicsetup/\*.sh**

**用root用户登录服务器（用root用户安装）。运行以下命令：**

**cd /home/weblogic/weblogicsetup/**

**./Install.sh**

注：脚本会创建一个测试domain,并启动一个测试节点，端口是7001，

需要停止此测试节点。kill -9 jav进程号。

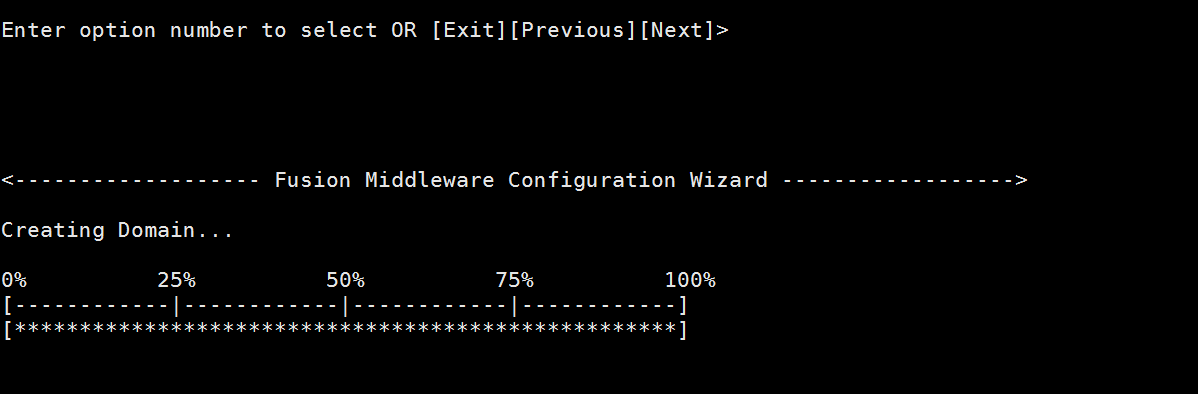
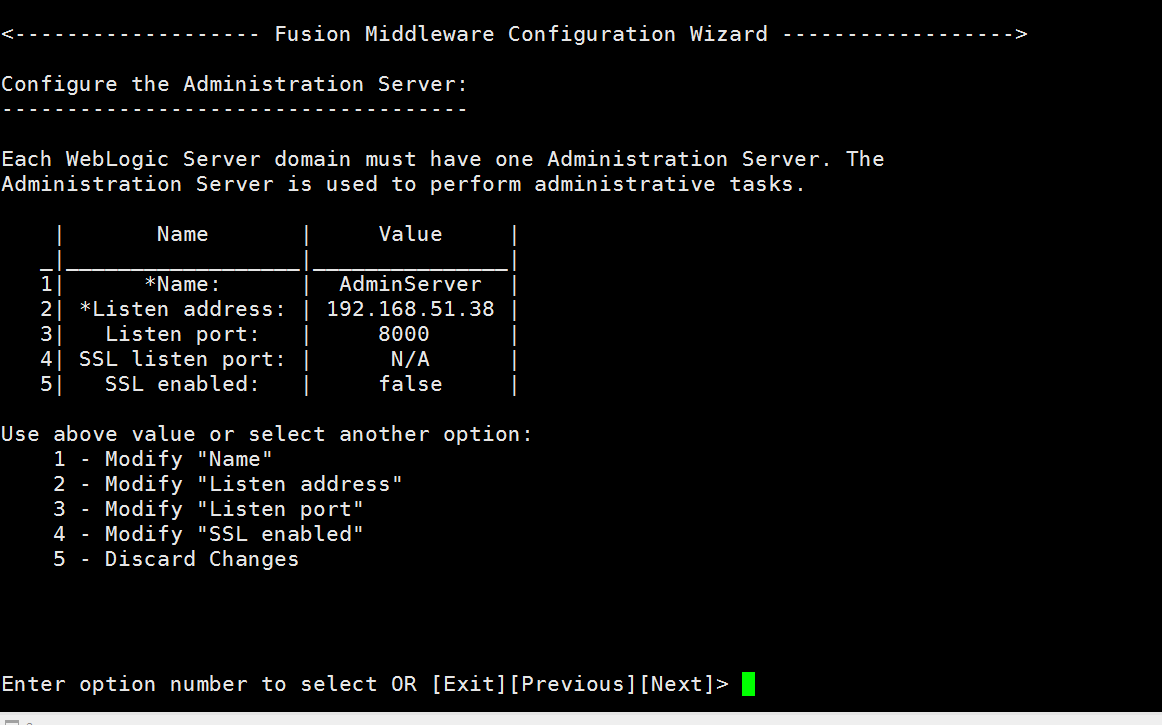
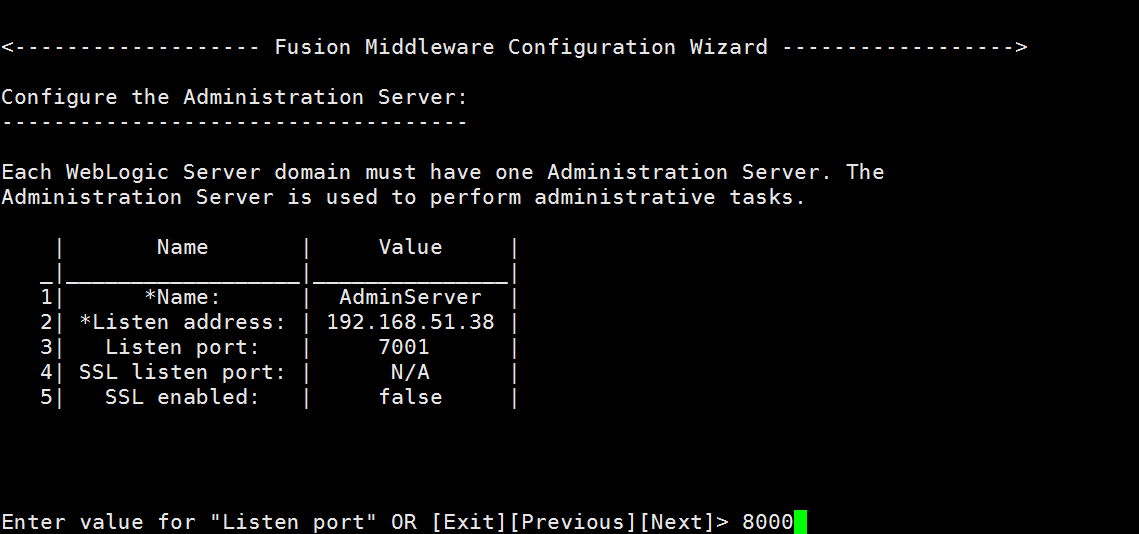
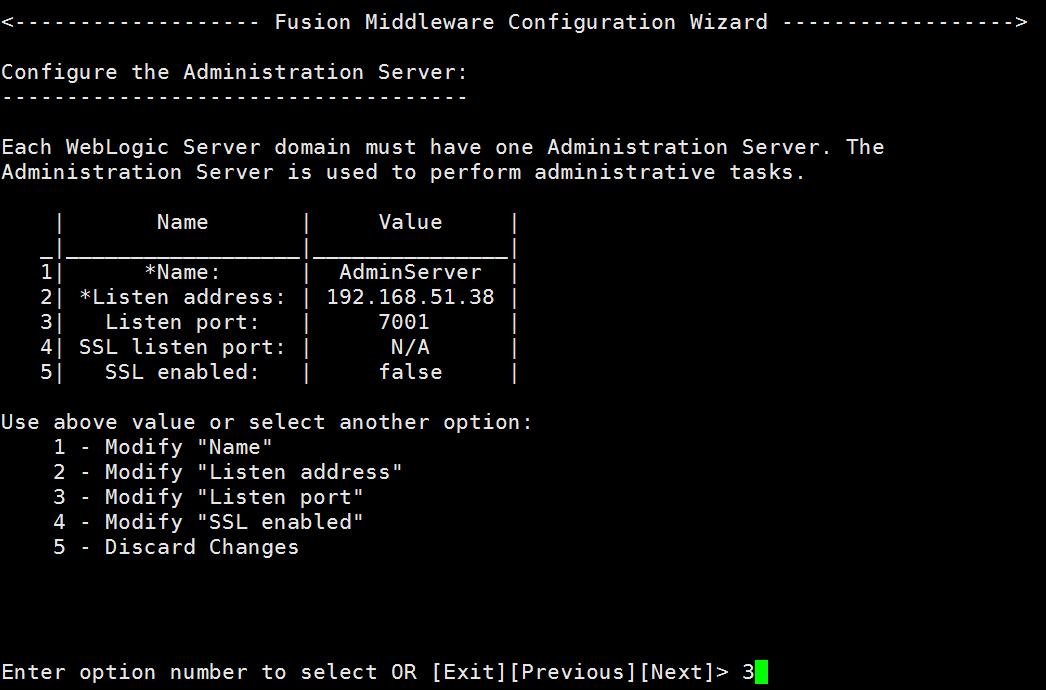
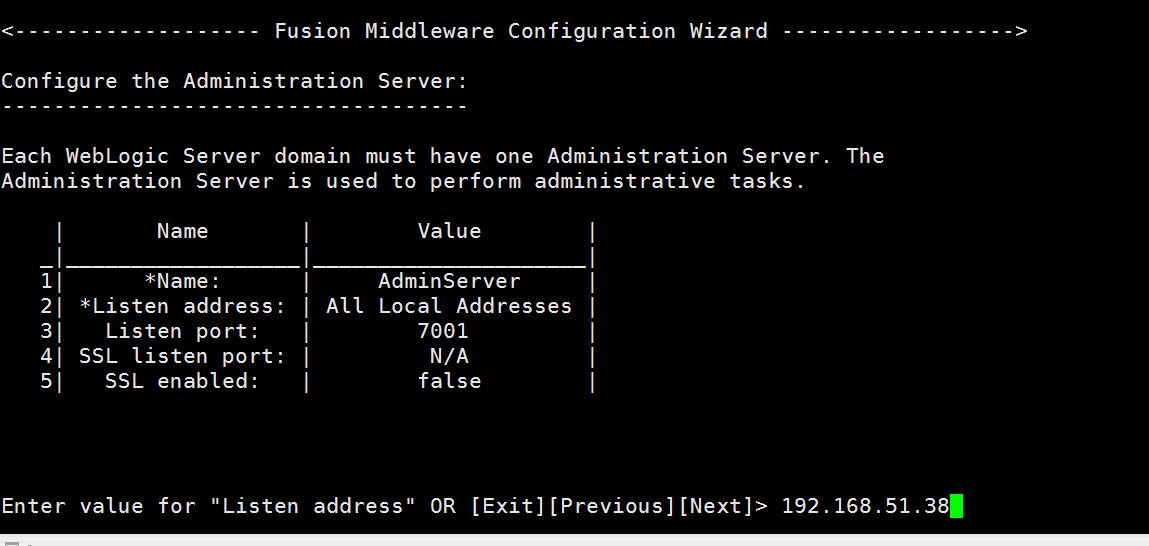
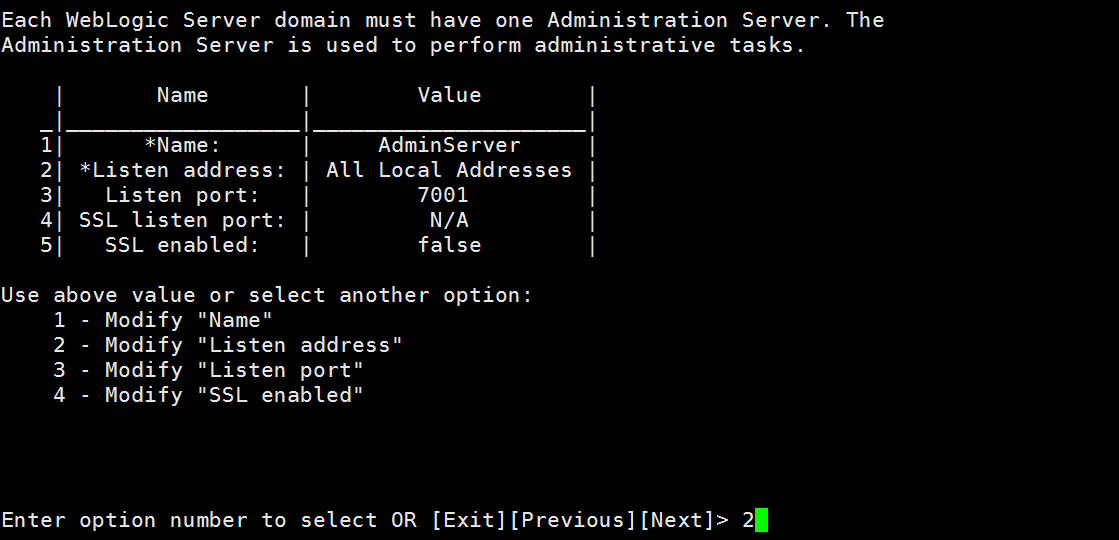
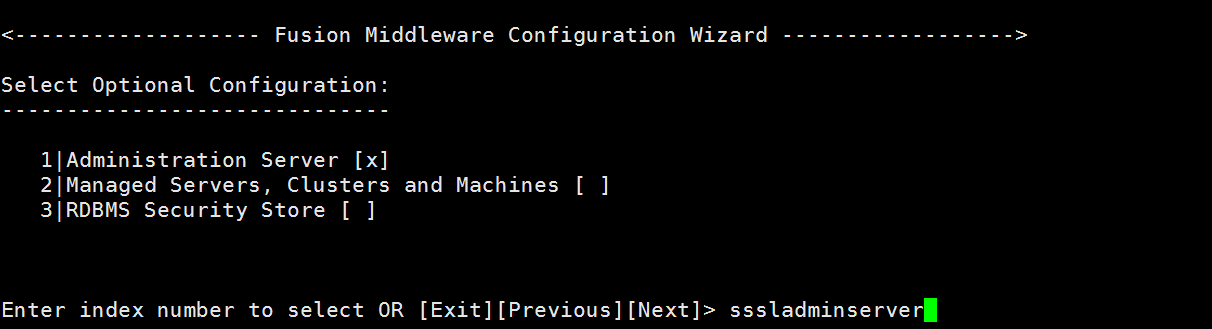
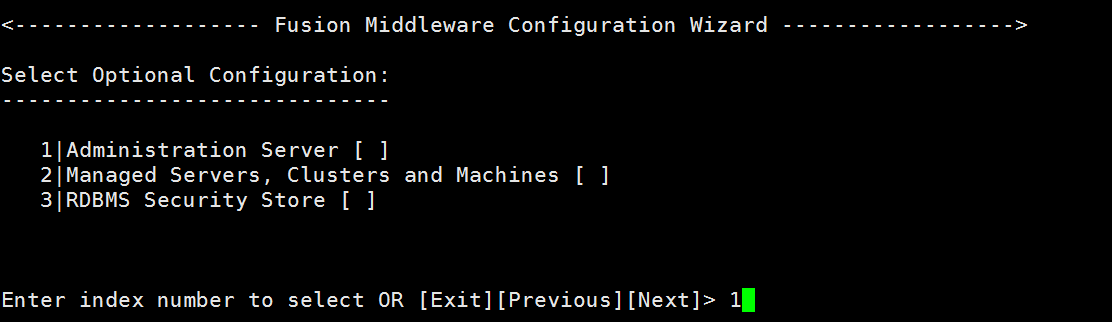
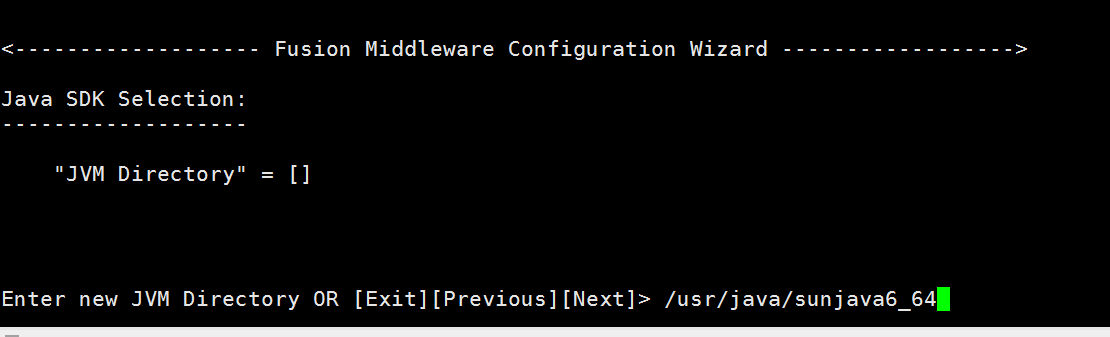
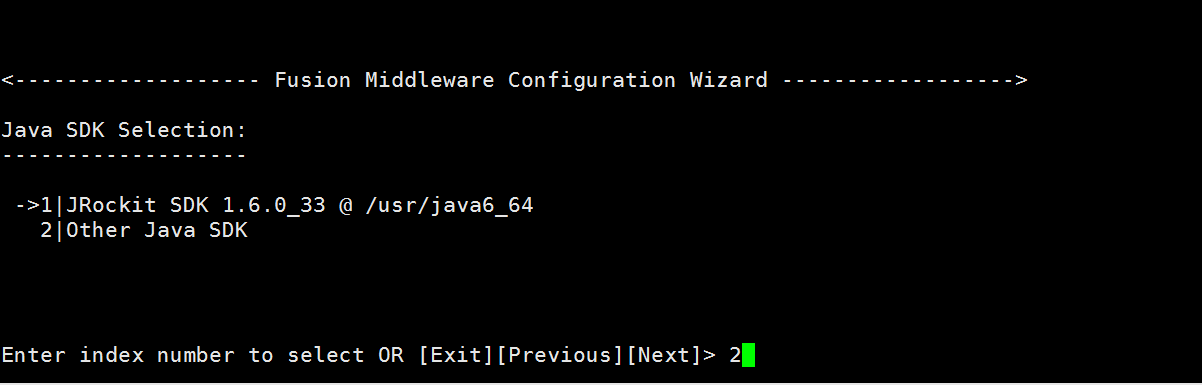
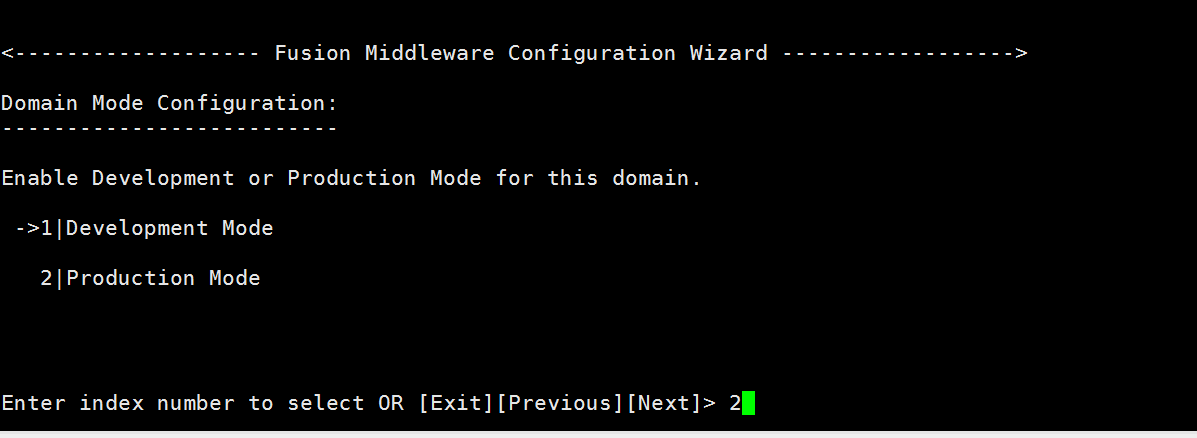
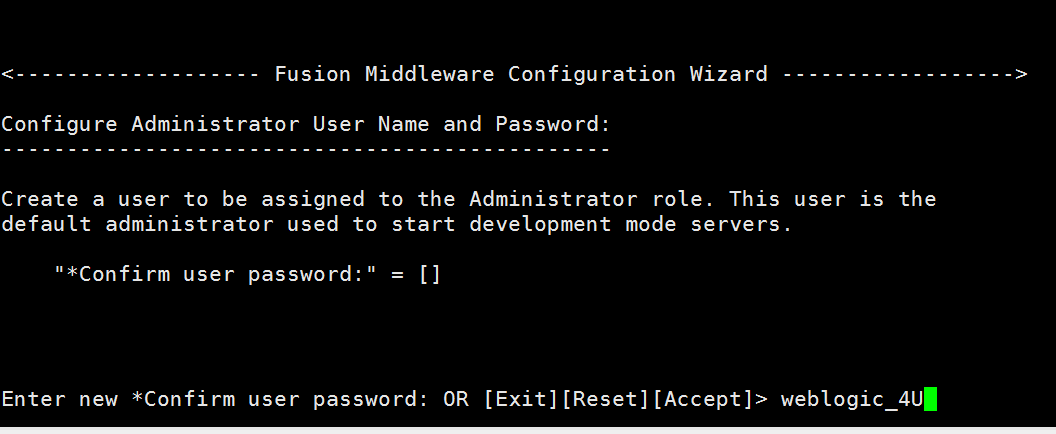
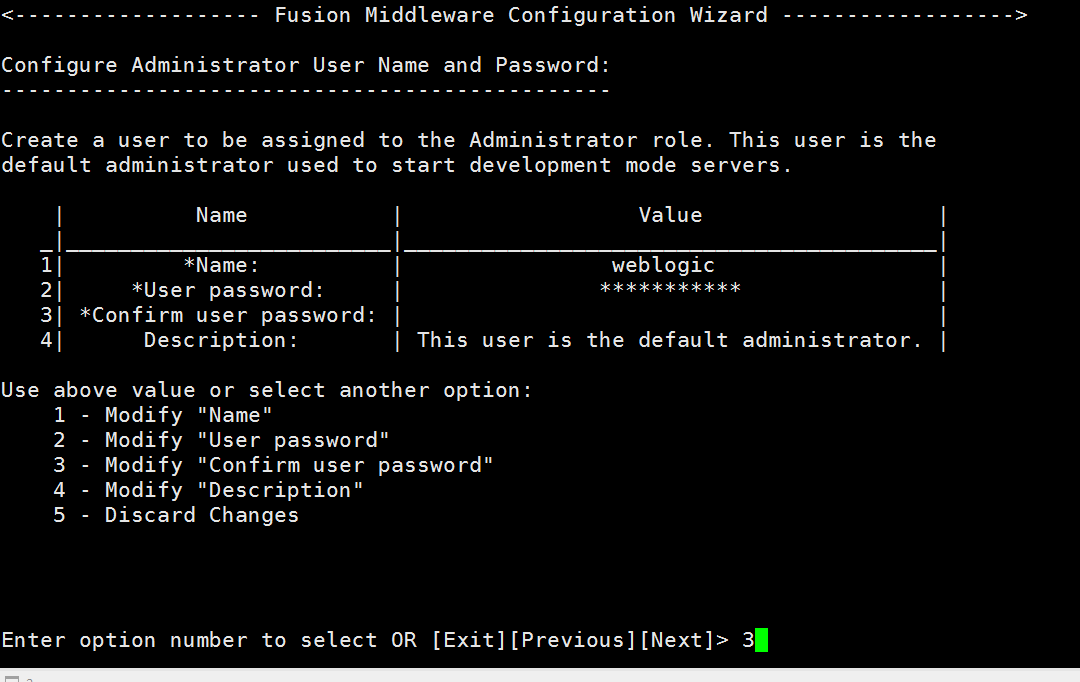
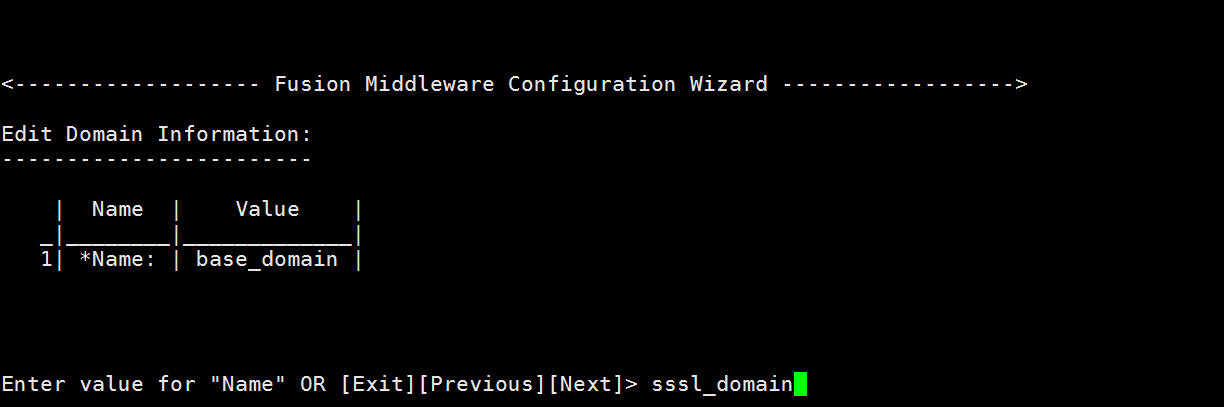
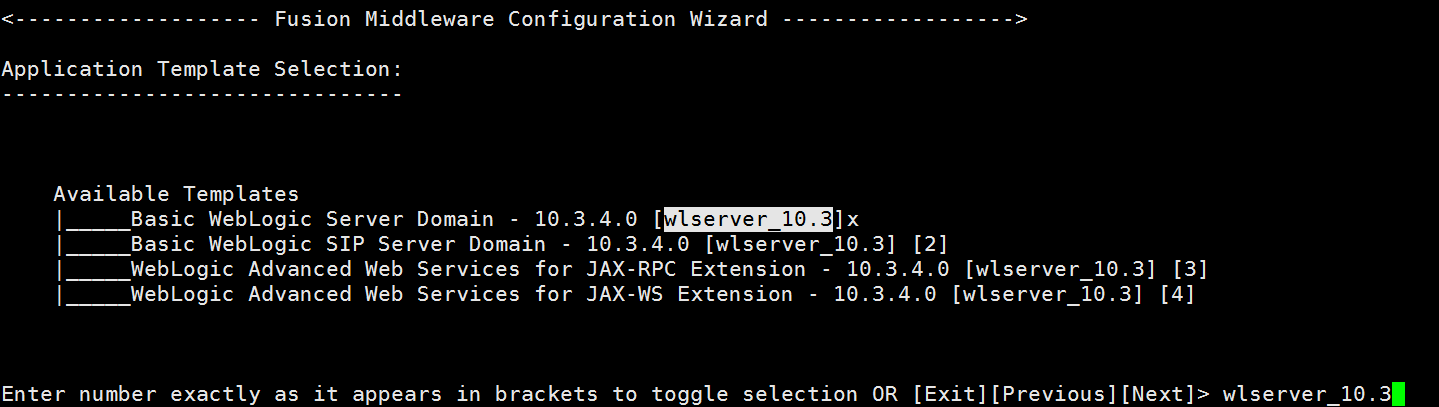
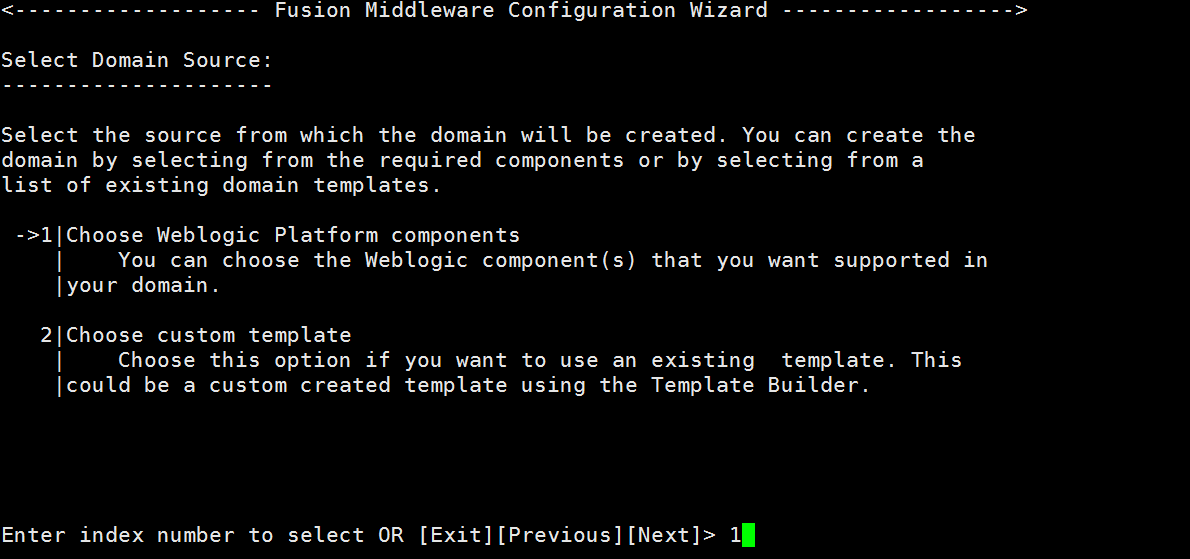
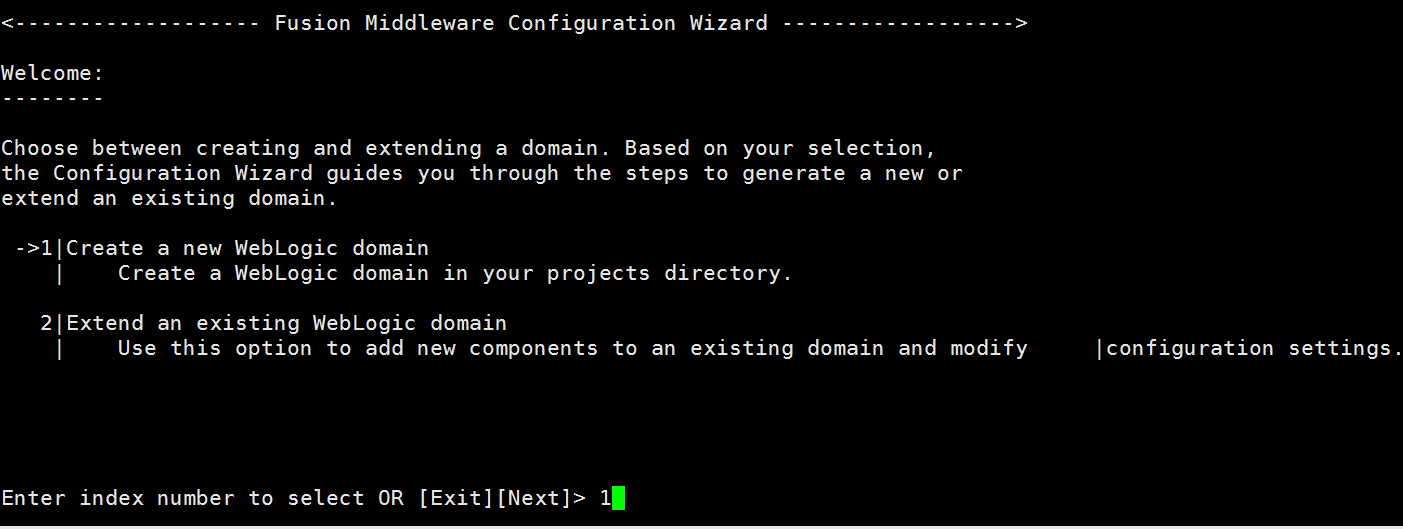
### 创建domain与管理节点

使用weblogic用户登录服务器，使用以下命令：

**cd /weblogic/wlserver\_10.3/common/bin/**

**./config.sh**

### 创建过程



### 设置管理节点

**vim startadmin.sh**

**nohup ./bin/startWebLogic.sh > ./logs/adminServer.log 2>&1 &**

**Cd /weblogic/user\_projects/domains/sssl\_domain/servers/AdminServer/**

**mkdir security**

**Vim boot.properties**

**username=weblogic**

**password=weblogic\_4U**

**/weblogic/user\_projects/domains/sssl\_domaincd/bin/**

**vim startManagedWebLogic.sh**

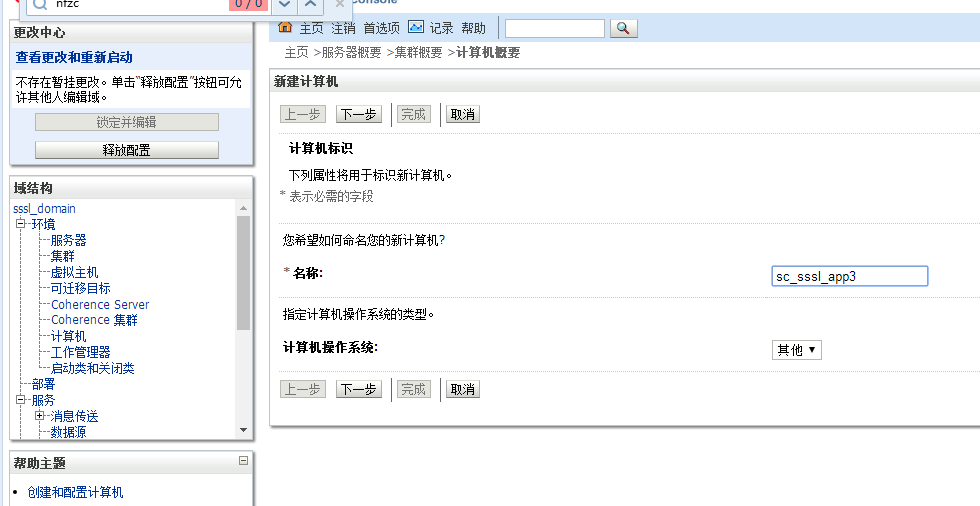
**WLS\_USER=”weblogic”**

**WLS\_PW=”weblogic\_4U”**

### 创建集群

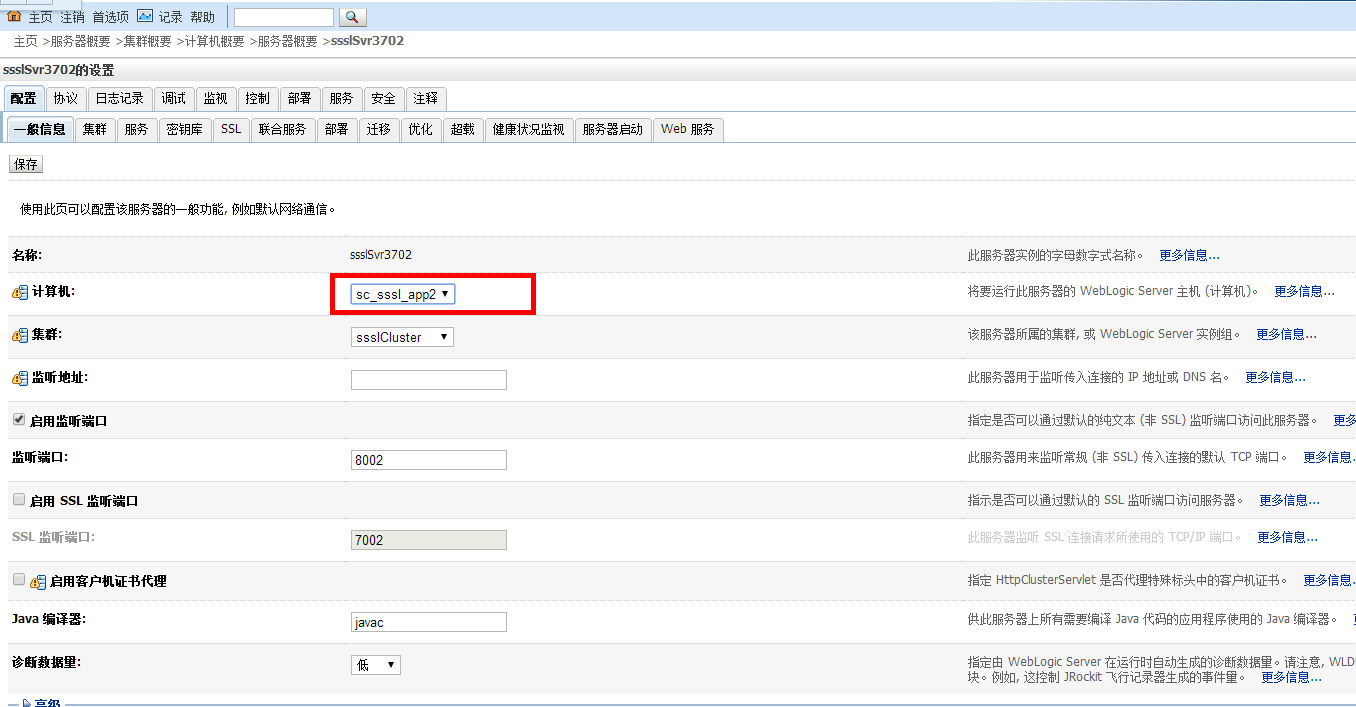
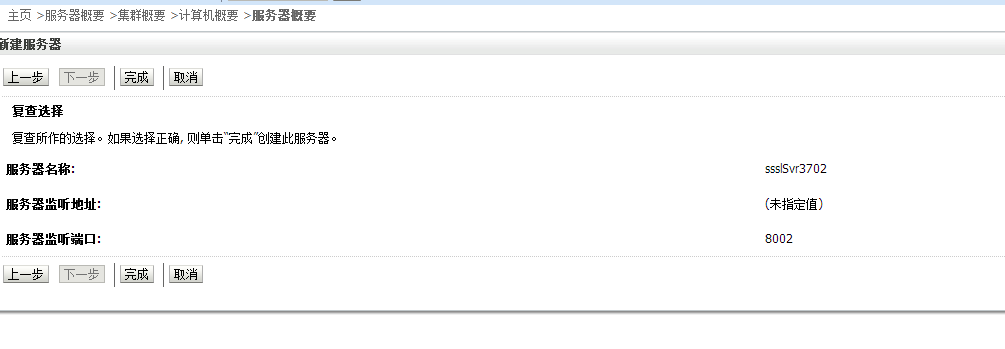
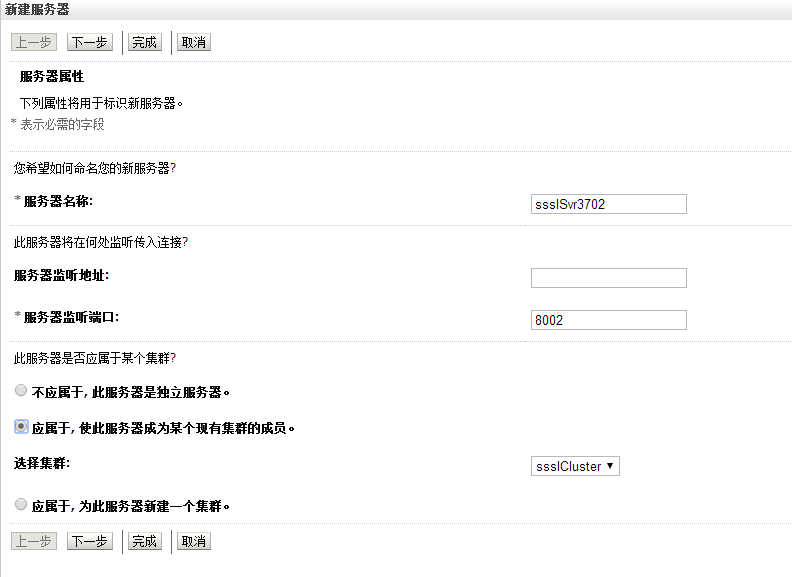


### 创建计算机





### 创建被管理节点



### 设置被管理节点

**进入/weblogic/user\_projects/domains/sssl\_domaincd/bin/**

**vim startManagedWebLogic.sh**

**WLS\_USER=”weblogic”**

**WLS\_PW=”weblogic\_4U”**

**vim startsssl3701.sh**

**nohup ./bin/startManagedWebLogic.sh ssslSvr3701 http://192.168.51.38:8000 >> ./logs/ssslSvr3701.log 2>&1 &**

**chmod +x \*.sh**

**mkdir logs**

### Could not obtain the localhsot address

启动报错

**vr3701/logs/ssslSvr3701.log is opened. All server side log events will be written to this file.>**

**<Jul 4, 2018 3:40:47 PM CST> <Critical> <WebLogicServer> <BEA-000386> <Server subsystem failed. Reason: java.lang.AssertionError: Could not obtain the localhost address. The most likely cause is an error in the network configuration of this machine.**

**java.lang.AssertionError: Could not obtain the localhost address. The most likely cause is an error in the network configuration of this machine.**

**at weblogic.server.channels.AddressUtils$AddressMaker.getLocalHost(AddressUtils.java:38)**

**at weblogic.server.channels.AddressUtils$AddressMaker.<clinit>(AddressUtils.java:33)**

**at weblogic.server.channels.AddressUtils.getIPAny(AddressUtils.java:154)**

**at weblogic.protocol.configuration.ChannelHelper.checkConsistency(ChannelHelper.java:61)**

**at weblogic.server.channels.ChannelService.start(ChannelService.java:207)**

**Truncated. see log file for complete stacktrace**

**Caused By: java.net.UnknownHostException: testmysql: testmysql**

**at java.net.InetAddress.getLocalHost(InetAddress.java:1360)**

**at weblogic.server.channels.AddressUtils$AddressMaker.getLocalHost(AddressUtils.java:36)**

**at weblogic.server.channels.AddressUtils$AddressMaker.<clinit>(AddressUtils.java:33)**

**at weblogic.server.channels.AddressUtils.getIPAny(AddressUtils.java:154)**

**at weblogic.protocol.configuration.ChannelHelper.checkConsistency(ChannelHelper.java:62)**

**Truncated. see log file for complete stacktrace**

解决：

**Hostname**

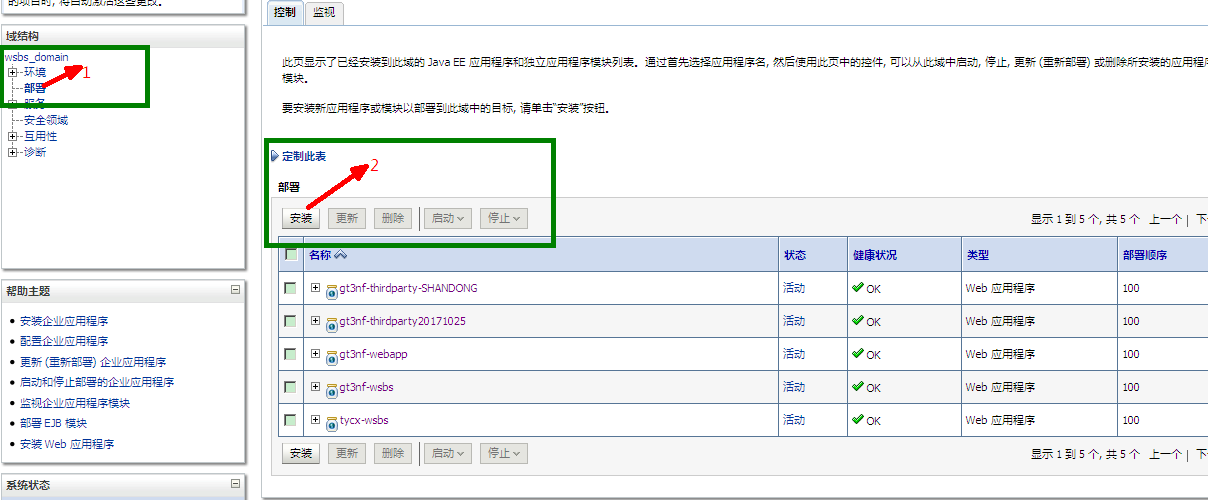
**Hoboson**

**Vim /etc/hsots**

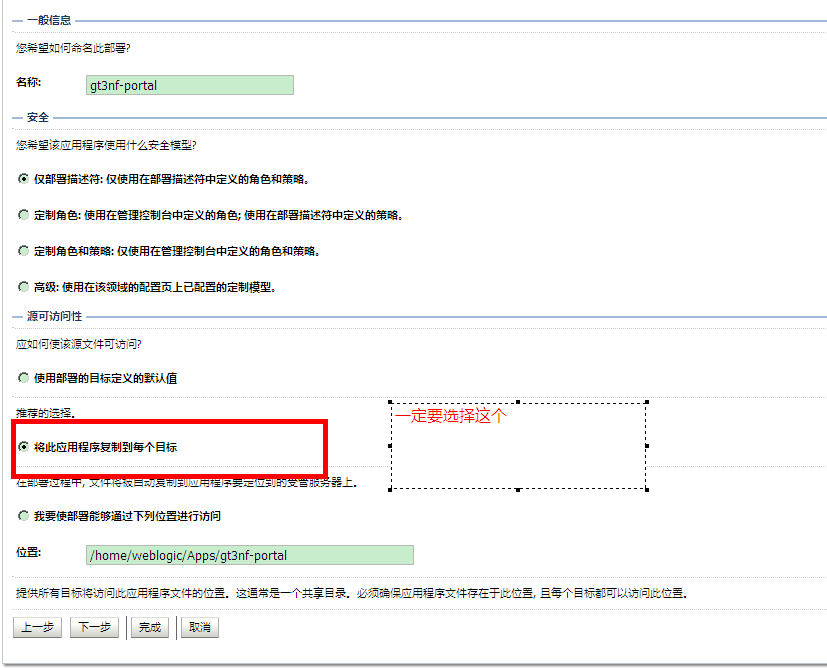
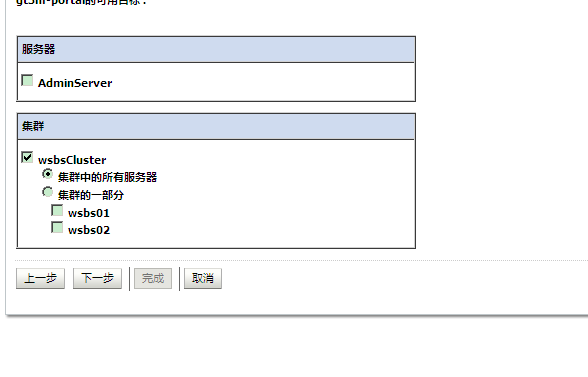
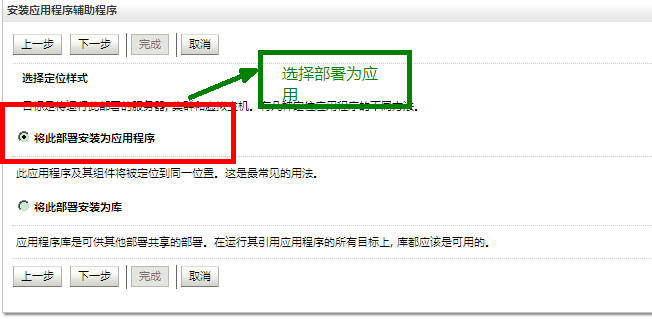
**192.168.51.37 Hoboson**

### test

## 部署应用







Start之后stage下面才会更新过去

总结：完成后再stage下面查看有没有同步过去

# Oracle

## 同义词 dblink

*select \* from hx\_dj.dj\_nsrxx*

*select \* from dba\_tables t where t.TABLE\_NAME=upper('dj\_nsrxx');*

*select \* from dba\_synonyms f where f.table\_name=upper('dj\_nsrxx');*

## DBLINK

*select \* from dba\_objects o where o.OBJECT\_TYPE = 'DATABASE LINK'*

*select \* from dba\_db\_links;*

## 权限 同义词

Grant select on table to NF\_NFZC 赋予跨越用户查询权限

Garnt select any table to NF\_NFZC 赋予跨用户查询该用户下所有表权限

Grant create synonym to nf\_nfzc 赋予创建同义词权限 system

Create or replace synonym dj\_nsrxx for hx\_dj.dj\_nsrxx 创建同义词

## 归档常用语句及监控

*--1.查询每日归档备份大小情况：*

SELECT TRUNC(FIRST\_TIME) "日期", TRUNC(SUM(BLOCKS\*BLOCK\_SIZE)/1024/1024/1024,2) "大小（GB/DAY）" FROM V$ARCHIVED\_LOG GROUP BY TRUNC(FIRST\_TIME) ORDER BY 1 DESC;

select NAME,SPACE\_LIMIT/1024/1024/1024,SPACE\_USED/1024/1024/1024,SPACE\_LIMIT/1024/1024/1024- SPACE\_USED/1024/1024/1024 as sy，NUMBER\_OF\_FILES from V$RECOVERY\_FILE\_DEST;

*--查看空间占用率*

select \* from v$flash\_recovery\_area\_usage

*--查看归档地址*

select \* from v$recovery\_file\_dest;

*--查看运行模式*

select name , log\_mode from v$database;

*--显示日志进程*

select\* from v$archive\_processes;

*--归档日志*

select t.NAME,t.\* from v$archived\_log t where t.NAME is not null;

*--查看正在使用的的组所对应的日志文件*

select \* from v$logfile;

## 归档

**java.sql.SQLException: ORA-00257: archiver error. Connect internal only, until freed.**

**归档已满**

**archive log stop;**

**archive log start;**

**archive log list;**

**show parameter RECOVERY\_FILE\_DEST;**

**show parameter RECOVERY\_FILE\_DEST\_SIZE;**

查看闪回区域设置大小

show parameter db\_recovery\_file\_dest\_size;

修改闪回区域设置大小

ALTER system set db\_recovery\_file\_dest\_size=8g;

删除所有归档

Delete expired archivelog all

删除三天之前的所有数据（保留最近三天数据）

Oracel用户登录

**rman target /**

**delete noprompt archivelog until time 'sysdate-3';**

**用数据泵导出比用exp 要快**

**################### 导出步骤：#####################**

**1.--创建directory**

**create or replace directory expdp\_dir as '/oracle/backup'; #改成相应的路径**

**grant read,write on directory expdp\_dir to flt;**

**2.--备份脚本autobak.sh内容**

**a=`date "+%y\_%m%d\_%H%M"`**

**expdp flt/flt123 schemas=flt dumpfile=fltdb%U.dmp logfile=fltdb$a.log directory=expdp\_dir filesize=20G**

**用filesize指定导出的每个文件大小**

**################### 导入步骤：#####################**

**1.在导入前也要在主库创建directory**

**create or replace directory expdp\_dir as '/oracle/backup'; #改成相应的路径**

**grant read,write on directory expdp\_dir to flt;**

**2.导入**

**impdp flt/flt123 schemas=flt dumpfile=文件名 logfile=impdp.log directory=expdp\_dir TABLE\_EXISTS\_ACTION=REPLACE**

**########################################**

**从视图 V$SESSION\_LONGOPS 中可以获得其它的有用信息来预测完成作业将花费的时间。**

**select sid, serial#, sofar, totalwork**

**from v$session\_longops**

**where sofar != totalwork**

**and sid =**

**(select sid**

**from v$session**

**where paddr in (select addr from v$process where spid = 30320))**

**设置非归档模式**

Archive log list

Alter system set log\_archive\_start=false scope=spfile

Shutdown immediate;

**不允许新的连接，不等待会话结束，不等待事物结束，做一个检查点并关闭数据库文 件没有结束的事物自动的rollback,启动时候不需要实例会服**

**Ps -ef | grep ora\_**

**Immediate 对应的进程是ora\_pmon\_nfzcdb**

Startup mount;//检查事物是否提交完整

Alter database noarchivelog

Alter database open;

Archive log list;

## 修改数据库语言

Alter session set NLS\_DATE\_LANGUAGE;

## Oracle时间操作

select \*

from hx\_dj.dj\_nsrxx dj

where dj.lrrq > to\_date('2018-03-21 00:00:00', 'YYYY-MM-DD hh24:mi:ss')

and dj.lrrq < to\_date('2018-03-21 8:39:16', 'YYYY-MM-DD hh24:mi:ss');

## 修改用户密码

alter user NF\_NFZC identified by 123456

## 用户解锁

select \* from dba\_users u where u.account\_status ='OPEN';

select \* from dba\_users u where u.username='NF\_NFZC';

alter user NF\_NFZC account unlock;

select count(1) from v$session;

select count(1) from v$session cn where cn.STATUS='ACTIVE';

select \* from v$parameter where name='processes';

## 全表扫描

*--查看全表扫描的语句*

select \*

from v$sql\_plan v

where v.OPERATION = 'TABLE ACCESS'

and v.OPTIONS = 'FULL'

and v.OBJECT\_OWNER = 'NF\_NFZC';

select \*

from v$sqltext t

where t.SQL\_ID = '72s278k2js0nr'

order by to\_number(piece);

处理结果

**申报清册表NF\_JCSJ\_DQYSB问题处理说明**

1. **现状说明**

ORACLE数据库一台服务器Cpu使用率达到95%，通过数据库快照发现申报清册表查询慢。具体sql如下：

q0javab0jsy2

select uuid, substr(t.zgswskfj\_dm, 0, 1) AS GDSBZ, t.kjzdzz\_dm, t.kjzdzzmc, t.zlbsxl\_dm, t.zlbsxlmc, t.sbqx, t.skssqq, t.skssqz, t.sbrq, t.nsqx\_dm, t.sbzt\_dm,

t.djxh, t.zgswskfj\_dm

from nf\_jcsj\_dqysb t where t.yxbz='Y' and t.sfcbbz='Y' and t.nd=:1 and t.djxh=:2 and (t.yf=:3 or t.yf='00')

5hr5zq9vrd408

select substr(t.zgswskfj\_dm, 0, 1) AS GDSBZ, t.kjzdzz\_dm, t.kjzdzzmc, t.zlbsxl\_dm, t.sbqx, t.skssqq, t.skssqz, t.sbrq, t.nsqx\_dm, t.sbzt\_dm, t.djxh,

t.zgswskfj\_dm

from nf\_jcsj\_dqysb t where t.yxbz='Y' and t.sfcbbz='Y' and t.nd=:1 and t.djxh=:2 and t.yf='00'

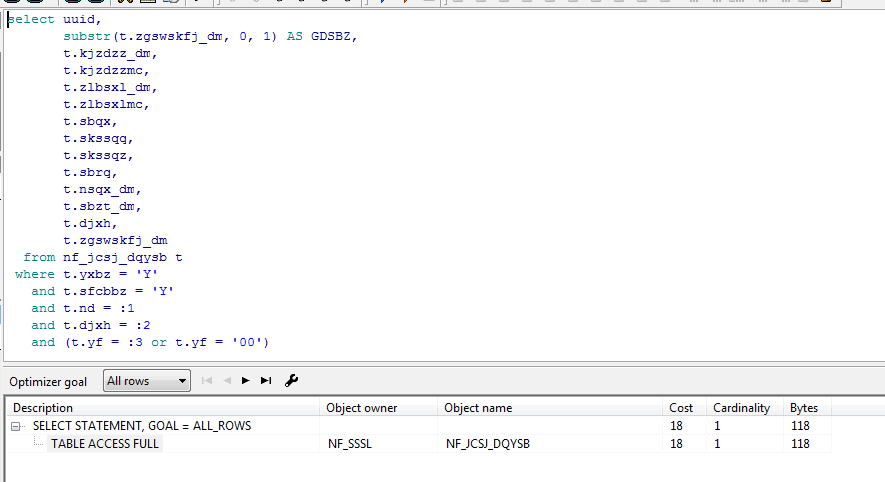
cn4n5pr2vftq1

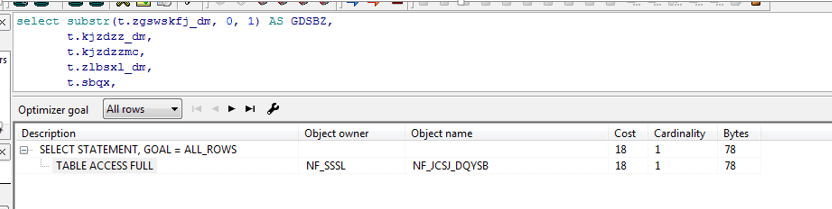
select substr(t.zgswskfj\_dm, 0, 1) AS GDSBZ, t.kjzdzz\_dm, t.kjzdzzmc, t.zlbsxl\_dm, t.sbqx, t.skssqq, t.skssqz, t.sbrq, t.nsqx\_dm, t.sbzt\_dm, t.djxh, t.zgswskfj\_dm

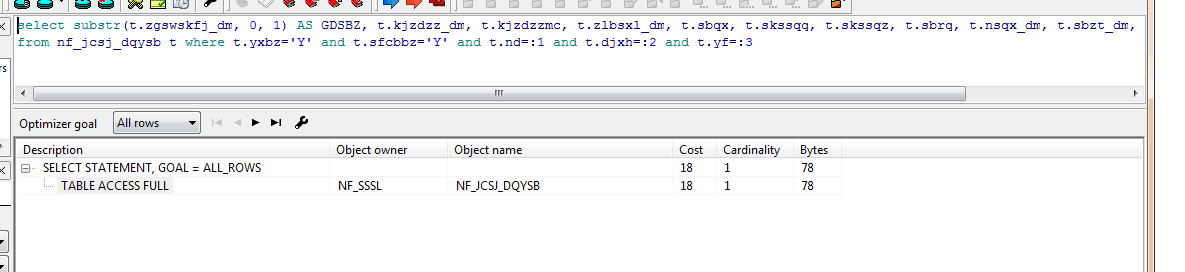
from nf\_jcsj\_dqysb t where t.yxbz='Y' and t.sfcbbz='Y' and t.nd=:1 and t.djxh=:2 and t.yf=:3

1. **问题分析**

**首先对执行效率进行了分析**







通过对日志报告分析，发现是全表扫描，初步看基本上定义是索引问题，但是再仔细看。从执行计划看总量很小，但是实际数据库表已经超过一百万条数据。

然后再对该表的统计信息进行分析，执行如下语句，查询当前表的统计信息状态：

select t.OWNER,

t.TABLE\_NAME,

t.NUM\_ROWS,

blocks,

avg\_row\_len,

t.LAST\_ANALYZED,

round(nvl(t.SAMPLE\_SIZE,1)/nvl(t.NUM\_ROWS,1),2)\*100||'%' "ESTIMATE\_PERCENT%",

t.PARTITIONED,

(select nvl(m.column\_name, 'null')

from dba\_part\_key\_columns m

where m.owner = t.OWNER

and m.name = t.TABLE\_NAME) "par\_key",

(select nvl(sm.column\_name, 'null')

from dba\_subpart\_key\_columns sm

where sm.owner = t.OWNER

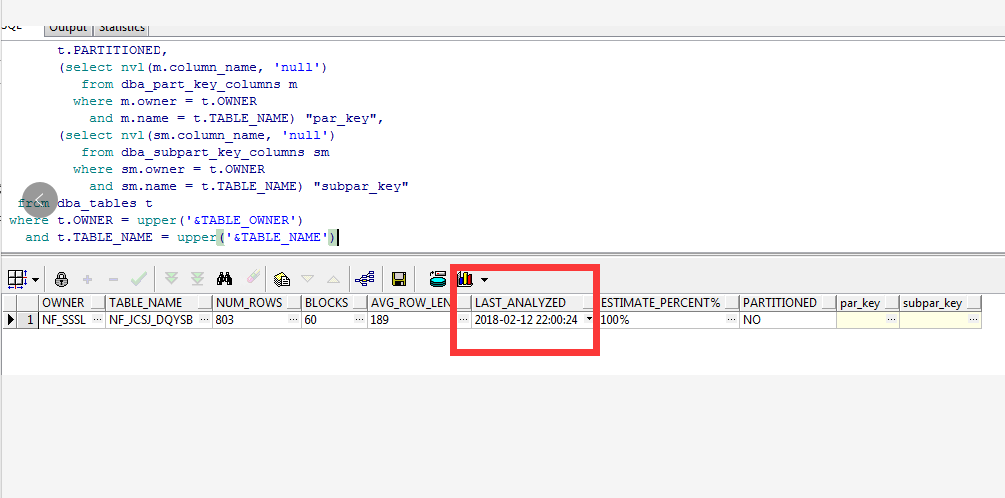
and sm.name = t.TABLE\_NAME) "subpar\_key"

from dba\_tables t

where t.OWNER = upper('&TABLE\_OWNER')

and t.TABLE\_NAME = upper('&TABLE\_NAME')。

通过结果得到统计信息如下：



时间是2018年2月12日，也就是oracle记录的这个表数据条数数2018年2月12日的，而这个时间是才装的数据库，条数自然很少，oracle认为数据库条数太少，就不通过索引执行查询，而且全表查询。结果就导致数据量大的时候，直接查询缓慢。

1. **问题处理**

执行sql语句，将表状态更新到当前，具体如下：

begin

DBMS\_STATS.GATHER\_TABLE\_STATS(ownname => '大写用户名',

tabname => '大写表名',

estimate\_percent => 100,

method\_opt => 'for all columns size repeat',

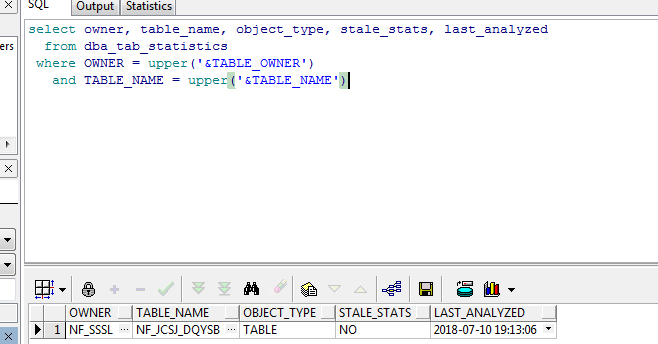
no\_invalidate => FALSE,

degree => 8,

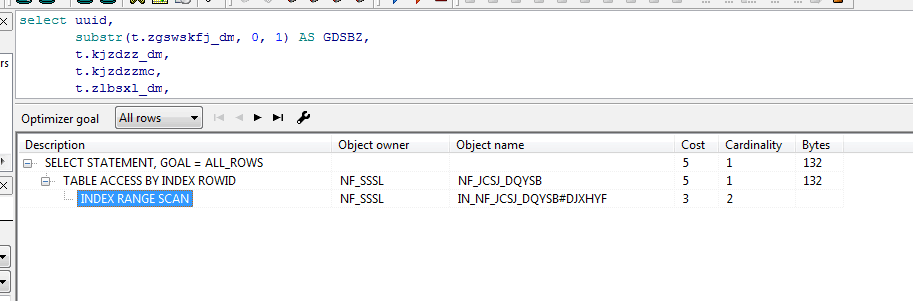
cascade => TRUE);

end;

**执行完成后，再查询状态，已经更新到最新状态了：**

****

**再看索引也生效了：**



1. **后续运维管理**

针对这个表，不要每天全部删除，每天只需要把无效的删除即可，然后当月月初删除上月的即可。

## 表锁

*--1、查看被锁住的表*

SELECT S.SID,

S.USER#,

s.SQL\_ID,

S.USERNAME,

OBJECT\_NAME,

MACHINE,

S.SID,

S.SERIAL#

FROM V$LOCKED\_OBJECT L, DBA\_OBJECTS O, V$SESSION S

WHERE L.OBJECT\_ID = O.OBJECT\_ID

AND L.SESSION\_ID = S.SID;

*--表解锁 alter system kill session 'sid,serial#';*

alter system kill session '3553,68';

*--根据sql\_id查询sql语句*

select \*

from v$sqltext t

where t.SQL\_ID = '7zmmadz89u547'

order by to\_number(piece);

## 表空间使用率查看

-**-查看表空间使用率**

**select ff.\*, ff.maxsize\_gb-ff.used\_gb as sygb from (**

**select tablespace\_name,**

**round(used\_space\*(select value from v$parameter where name='db\_block\_size')/power(2,30),2) USED\_GB,**

**round(tablespace\_size\*(select value from v$parameter where name='db\_block\_size')/power(2,30)) MAXSIZE\_GB,**

**round(used\_percent,2) as "PCT%"**

**from dba\_tablespace\_usage\_metrics**

**) ff where ff."PCT%" >10 order by ff."PCT%" desc**

**--查看表空间剩余大小排序**

**select ff.\*, ff.maxsize\_gb-ff.used\_gb as sygb from (**

**select tablespace\_name,**

**round(used\_space\*(select value from v$parameter where name='db\_block\_size')/power(2,30),2) USED\_GB,**

**round(tablespace\_size\*(select value from v$parameter where name='db\_block\_size')/power(2,30)) MAXSIZE\_GB,**

**round(used\_percent,2) as "PCT%"**

**from dba\_tablespace\_usage\_metrics**

**) ff where ff."PCT%" >10 order by sygb asc**

## 表空间

*--查看表空间*

select \* from dba\_tablespaces

*--查看用户和默认表空间的关系*

select tablespace\_name,file\_id,bytes/1024/1024,file\_name

　　 from dba\_data\_files order by file\_id;

*--查看用户和默认表空间的关系*

select username,default\_tablespace from dba\_users;

*--1创建表空间*

create tablespace Hoboson\_space

datafile 'H:\APP\HOBOSON\PRODUCT\11.2.0\TABLESPACE\HobosonDream.dbf'

size 50m autoextend on;

*--2创建用户*

CREATE USER Hoboson

IDENTIFIED BY Hoboosn0316

DEFAULT TABLESPACE Hoboson\_space;

*--3创建好表空间,也可以手动将表空间分配给用户.*

alter user zzg default tablespace ts\_zzg;

*--4然后，我们需要给自己的用户赋予权限来管理自己的表空间*

GRANT CONNECT TO Hoboson;

GRANT RESOURCE TO Hoboson;

GRANT dba TO Hoboson;

*--5.创建好用户我们接着就可以修改用户的密码.*

alter user Hobosonidentified by Hoboson\_2016;

*--查看表空间使用率*

select ff.\*, ff.maxsize\_gb-ff.used\_gb as sygb from (

select tablespace\_name,

round(used\_space\*(select value from v$parameter where name='db\_block\_size')/power(2,30),2) USED\_GB,

round(tablespace\_size\*(select value from v$parameter where name='db\_block\_size')/power(2,30)) MAXSIZE\_GB,

round(used\_percent,2) as "PCT%"

from dba\_tablespace\_usage\_metrics

) ff where ff."PCT%" >50 order by ff."PCT%" desc;

查看表空间路径

select \* from dba\_data\_files f where f.TABLESPACE\_NAME='TS\_SSSL\_DAT\_2018';

扩展表空间

ALTER TABLESPACE TS\_SSSL\_DAT\_2018 ADD DATAFILE '+DSDATA/snltnfzc/datafile/ts\_sssl\_dat\_35.dbf' SIZE 30G AUTOEXTEND ON NEXT 100M MAXSIZE UNLIMITED;

## 全表扫描监控

*--用SYS用户运行以下SQL:*

select sql\_text

from v$sqltext t, v$sql\_plan p

where t.hash\_value = p.hash\_value

and p.operation = 'TABLE ACCESS'

and p.options = 'FULL'

order by p.hash\_value, t.piece;

## 根据字段类型查找表

select distinct owner || '.' || table\_name

from all\_tab\_columns

where data\_type = 'BLOB'

and owner like 'NF%';

## TOP10Sql

select \* from

(select parsing\_user\_id,executions,sorts,command\_type,disk\_reads,sql\_text from v$sqlarea

order by disk\_reads desc) where rownum < 10;

## CUP消耗高的

Select a.sid,

spid,

status,

substr(a.program, 1, 40) prog,

a.terminal,

osuser,

value / 60 / 100 value,

d.SQL\_TEXT

from v$session a, v$process b, v$sesstat c, v$sqltext d

where c.statistic# = 12

and c.sid = a.sid

and a.paddr = b.addr

and a.SQL\_ADDRESS = d.ADDRESS

order by value desc;

## 查看asm磁盘空间

select name, /\*state,type\*/

total\_mb / 1024 总的G,

cold\_used\_mb / 1024 使用G,

free\_mb / 1024 剩余G

from v$asm\_diskgroup /\* WHERE NAME LIKE '%NF%'\*/

--查看磁盘状态

Select \* from v$asm\_disk;

## 查看用户密码是否过期

select \* from dba\_users where username like 'NF\_%' OR username like 'IS\_%';

SELECT \* FROM SYS.USER$ WHERE NAME LIKE 'NF\_%' OR NAME like 'IS\_%';

## 查看表空间包含临时表空间

-- 查询表空间(包含临时表空间)

SELECT D.TABLESPACE\_NAME,

SPACE "SUM\_SPACE(M)",

BLOCKS SUM\_BLOCKS,

SPACE - NVL(FREE\_SPACE, 0) "USED\_SPACE(M)",

ROUND((1 - NVL(FREE\_SPACE, 0) / SPACE) \* 100, 2) "USED\_RATE(%)",

FREE\_SPACE "FREE\_SPACE(M)",

sysdate

FROM (SELECT TABLESPACE\_NAME,

ROUND(SUM(BYTES) / (1024 \* 1024), 2) SPACE,

SUM(BLOCKS) BLOCKS

FROM DBA\_DATA\_FILES

GROUP BY TABLESPACE\_NAME) D,

(SELECT TABLESPACE\_NAME,

ROUND(SUM(BYTES) / (1024 \* 1024), 2) FREE\_SPACE

FROM DBA\_FREE\_SPACE

GROUP BY TABLESPACE\_NAME) F

WHERE D.TABLESPACE\_NAME = F.TABLESPACE\_NAME(+)

UNION ALL --if have tempfile

SELECT D.TABLESPACE\_NAME,

SPACE "SUM\_SPACE(M)",

BLOCKS SUM\_BLOCKS,

USED\_SPACE "USED\_SPACE(M)",

ROUND(NVL(USED\_SPACE, 0) / SPACE \* 100, 2) "USED\_RATE(%)",

NVL(FREE\_SPACE, 0) "FREE\_SPACE(M)",

sysdate

FROM (SELECT TABLESPACE\_NAME,

ROUND(SUM(BYTES) / (1024 \* 1024), 2) SPACE,

SUM(BLOCKS) BLOCKS

FROM DBA\_TEMP\_FILES

GROUP BY TABLESPACE\_NAME) D,

(SELECT TABLESPACE\_NAME,

ROUND(SUM(BYTES\_USED) / (1024 \* 1024), 2) USED\_SPACE,

ROUND(SUM(BYTES\_FREE) / (1024 \* 1024), 2) FREE\_SPACE

FROM V$TEMP\_SPACE\_HEADER

GROUP BY TABLESPACE\_NAME) F

WHERE D.TABLESPACE\_NAME = F.TABLESPACE\_NAME(+);

## sqlplus中查看语句执行计划

方法一、

SQL> set autot trace exp;

SQL>

SQL>

SQL> select \* from dual;

Elapsed: 00:00:00.00

Execution Plan

----------------------------------------------------------

Plan hash value: 272002086

--------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

--------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 1 | 2 | 2 (0)| 00:00:01 |

| 1 | TABLE ACCESS FULL| DUAL | 1 | 2 | 2 (0)| 00:00:01 |

--------------------------------------------------------------------------

## 正在执行的sql

select a.username, a.sid,b.SQL\_TEXT, b.SQL\_FULLTEXT

from v$session a, v$sqlarea b

where a.sql\_address = b.address

## 已经执行的sql

select b.SQL\_TEXT,b.FIRST\_LOAD\_TIME,b.SQL\_FULLTEXT

from v$sqlarea b

where b.FIRST\_LOAD\_TIME between '2009-10-15/09:24:47' and

'2009-10-15/09:24:47' order by b.FIRST\_LOAD\_TIME

（此方法好处可以查看某一时间段执行过的sql，并且 SQL\_FULLTEXT 包含了完整的 sql 语句）

## 细致，查系统各维度规律

select s.snap\_date,

decode(s.redosize, null, '--shutdown or end--', s.currtime) "TIME",

to\_char(round(s.seconds/60,2)) "elapse(min)",

round(t.db\_time / 1000000 / 60, 2) "DB time(min)",

s.redosize redo,

round(s.redosize / s.seconds, 2) "redo/s",

s.logicalreads logical,

round(s.logicalreads / s.seconds, 2) "logical/s",

physicalreads physical,

round(s.physicalreads / s.seconds, 2) "phy/s",

s.executes execs,

round(s.executes / s.seconds, 2) "execs/s",

s.parse,

round(s.parse / s.seconds, 2) "parse/s",

s.hardparse,

round(s.hardparse / s.seconds, 2) "hardparse/s",

s.transactions trans,

round(s.transactions / s.seconds, 2) "trans/s"

from (select curr\_redo - last\_redo redosize,

curr\_logicalreads - last\_logicalreads logicalreads,

curr\_physicalreads - last\_physicalreads physicalreads,

curr\_executes - last\_executes executes,

curr\_parse - last\_parse parse,

curr\_hardparse - last\_hardparse hardparse,

curr\_transactions - last\_transactions transactions,

round(((currtime + 0) - (lasttime + 0)) \* 3600 \* 24, 0) seconds,

to\_char(currtime, 'yy/mm/dd') snap\_date,

to\_char(currtime, 'hh24:mi') currtime,

currsnap\_id endsnap\_id,

to\_char(startup\_time, 'yyyy-mm-dd hh24:mi:ss') startup\_time

from (select a.redo last\_redo,

a.logicalreads last\_logicalreads,

a.physicalreads last\_physicalreads,

a.executes last\_executes,

a.parse last\_parse,

a.hardparse last\_hardparse,

a.transactions last\_transactions,

lead(a.redo, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) curr\_redo,

lead(a.logicalreads, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) curr\_logicalreads,

lead(a.physicalreads, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) curr\_physicalreads,

lead(a.executes, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) curr\_executes,

lead(a.parse, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) curr\_parse,

lead(a.hardparse, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) curr\_hardparse,

lead(a.transactions, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) curr\_transactions,

b.end\_interval\_time lasttime,

lead(b.end\_interval\_time, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) currtime,

lead(b.snap\_id, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) currsnap\_id,

b.startup\_time

from (select snap\_id,

dbid,

instance\_number,

sum(decode(stat\_name, 'redo size', value, 0)) redo,

sum(decode(stat\_name,

'session logical reads',

value,

0)) logicalreads,

sum(decode(stat\_name,

'physical reads',

value,

0)) physicalreads,

sum(decode(stat\_name, 'execute count', value, 0)) executes,

sum(decode(stat\_name,

'parse count (total)',

value,

0)) parse,

sum(decode(stat\_name,

'parse count (hard)',

value,

0)) hardparse,

sum(decode(stat\_name,

'user rollbacks',

value,

'user commits',

value,

0)) transactions

from dba\_hist\_sysstat

where stat\_name in

('redo size',

'session logical reads',

'physical reads',

'execute count',

'user rollbacks',

'user commits',

'parse count (hard)',

'parse count (total)')

group by snap\_id, dbid, instance\_number) a,

dba\_hist\_snapshot b

where a.snap\_id = b.snap\_id

and a.dbid = b.dbid

and a.instance\_number = b.instance\_number

order by end\_interval\_time)) s,

(select lead(a.value, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) - a.value db\_time,

lead(b.snap\_id, 1, null) over(partition by b.startup\_time order by b.end\_interval\_time) endsnap\_id

from dba\_hist\_sys\_time\_model a, dba\_hist\_snapshot b

where a.snap\_id = b.snap\_id

and a.dbid = b.dbid

and a.instance\_number = b.instance\_number

and a.stat\_name = 'DB time') t

where s.endsnap\_id = t.endsnap\_id

order by s.snap\_date ,time desc;

## 查找频繁提交的语句

*--提交次数最多的SESSION*

select t1.sid, t1.value, t2.name

from v$sesstat t1, v$statname t2

where t2.name like '%user commits%'

and t1.STATISTIC# = t2.STATISTIC#

and value >= 10000

order by value desc;

*--取得SID既可以代入到V$SESSION 和V$SQL中去分析*

*--得出SQL\_ID*

select t.SID,

t.PROGRAM,

t.EVENT,

t.LOGON\_TIME,

t.WAIT\_TIME,

t.SECONDS\_IN\_WAIT,

t.SQL\_ID,

t.PREV\_SQL\_ID

from v$session t

where sid in ('3037', '379', '3018', '4150', '2264');

*--根据sql\_id或prev\_sql\_id代入得到SQL*

select t.sql\_id,

t.sql\_text,

t.EXECUTIONS,

t.FIRST\_LOAD\_TIME,

t.LAST\_LOAD\_TIME

from v$sqlarea t

where sql\_id in ('ccpn5c32bmfmf');

*--也请关注一下这个：*

select \* from v$active\_session\_history where session\_id=194

## 分页

规律总结

1 0 6

2 5 11

3 10 16

4 15 21

5 20 26

*--公式= (page Now-1)\* pageSize ,Pagnow\*PageSize+1*

select \*

from (select rownum as nu, s.\* from hx\_dj.dj\_nsrxx s where rownum < 6) fin

where fin.nu > 0

Mysql

/\*

list=(pageNow-1)\*pageSize,pageSize

\*/

SELECT \* FROM `dj\_nsrxx` dj LIMIT 0,10

## Oracle快照

### oracle用户

### 查看oracle安装目录 echo $ORACLE\_HOME

### 进入awrrpt.sql目录

cd /u01/app/oracle/product/11.2.0/db\_1/rdbms/admin/

等同于：

$ORACLE\_HOME/rdbms/admin/awrrpt.sql

### sqlplus / as sysdba;

### @awrrpt.sql

参数1 report\_type 报表类型

参数2 report\_name 报表名称

参数3 begin\_snap 查快照时间起

参数4 end\_snap 快照时间止

### 下载快照路径如下

/u01/app/oracle/product/11.2.0/db\_1/rdbms/admin/

## Rac管理

### Crs\_stat -t

ASM 共享存储----》共享文件系统 只能放和数据库的文件 不能放常规的文件 扩展性比较好

OCFS 可以放其他的文件

**当一个节点挂掉时候 这个节点的vip会漂到另一个正常的节点 用户可以连接到，但是正常的节点没有监听这个vip 所以无法连接数据库 将错误快速返回给用户 而不是等待超时**

### crs\_stat -ls

## Exp导出

在oracle 用户下导出

**exp NF\_NFZC/Foresee\_2018@snltnfzc file=/home/oracle/yh\_mmxxds\_tmp.dmp tables=yh\_mmxxds\_tmp log=/home/oracle/yh\_mmxxds\_tmp.log**

## 知识科普

**1两个实例操作一个数据库**

**2健忘症**

一号机修改配置二号机不知道

二号机修改配置一号机不知道

公用的共享磁盘存放集群配置

**3脑裂**

心跳数据（节点通讯）协同工作 当双击不能协同工作，oracle驱逐节点（投票算法 偶数集群（谁先和表决盘通讯谁就可以保留，驱逐另一个没有抢到的），奇数集群，自动重启）

1发包探测2 3

2 发包探测1 3

3 发包探测1 2

**4中间件**

Oracle 9之前

主机高可用 收费

数据库集群

Oracle 10 自己的高可用中间件 rac

Oracle 11 gridInfrastructure ASM(自动存储管理)

## 集群配置：

1 关闭防火墙

2主机必须有两个网卡

3配置高可用vip

4 ntp服务配置

5 openfile共享磁盘建立

6建立信任关系

7安装grid中间件

**instance：实例时客户端和db之间通讯的桥梁**

# OGG

### OracleSession会话

数据库位置156.16.33.81 ,166.16.33.82

登录81、82

Id 查看用户

Su – oracle

Sqlplus / as sysdba

Select count(\*) from v$session;

总结：主要看81和82会话差距大不大，判断是否负载均衡

修改参数配置

Edit param r\_dj

Alter r\_sb6,extseqno 7311,extrba 0

Stop r\_dj

Start r\_dj

Info r\_dj

### Ogg查看

Id 查看用户

Su – oracle

Cd /ogg

./ggsci

info all 查看所有进程

info r\_dj 查看进程信息

info r\_dj detail 查看进程详细信息

view params r\_dj 查看参数文件

edit params r\_dj 编辑参数文件

view report rd\_jd 查看错误日志

### Ogg拆分

拆分前根据info r\_dj 获取列队文件信息及RBA号

Stop r\_dj

Info r\_dj

edit param r\_dj\_new

编写参数文件对应的map对象

复制要拆分的内容 edit param r\_dj

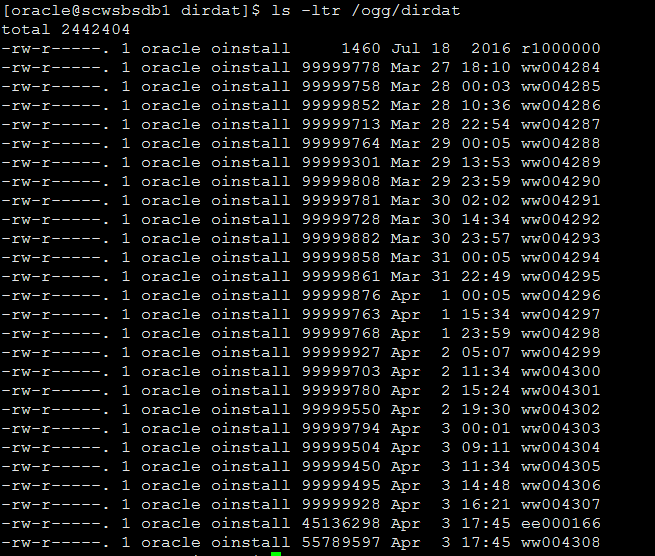
启动拆分复制进程，指定读取的checkpoint

Alter extract r\_dj extseqno sn007489 ,extrba 138762665

Start r\_dj

Start r\_djnew

ls -ltr /ogg/dirdat



**Summer:**

**在拆分过程中拆分job和被拆分进程处于stop状态**

**被拆分的表被排除**

## 查看Vfs路径

测试

<http://192.168.50.249/config/conf?action=confTableEdit>

搜索vfs查看路径

生产

<http://192.168.51.2:8081/config/conf?action=confTableEdit>

搜索vfs查看路径

# Linux

## 常用命令

### 时间格式化

export TIME\_STYLE='+%Y-%m-%d %H:%M:%S'

### 查看以字符编码

echo $LANG

env | grep LANG

export | grep LANG

locale

### 设置字符编码

export LANG=zh\_ZN.UFT-8

设置当前会话字符编码 会话失效设置失效

Vim /etc/sysconfig/i18n

Source /etc/sysconfig/i18n 强制生效

### 根据端口号查看服务

netstat –anp | grep 8080

### 查看文件大小

du –h /file 查看明细以及总大小

du –sh /file 查看总大小

nginx软启动

./nginx -s reload

### 定时任务

crontab -e 添加要设置的定时任务

/etc/init.d/crond status 查看状态

Ps -ef|grep crond 查看进程

定时任务：cd /etc

**cron.d/ #系统定期需要做的任务**

cron.deny #用于控制是那些用户不能使用Crontab功能

crontab #设定定时任务执行文件

cron.hourly/ #每小时执行一次的job

cron.daily/ #每天执行一次的job

cron.weekly/ #每星期执行一次的job

cron.monthly/ #每月执行一次的job

30 10 \* \* 1,2,3,4,5 /weblogic/user\_projects/domains/sssl\_domain/clearAlllogs.sh

**注意：**

**脚本要写绝对路径**

**crond 服务是否启动**

**是否有权限执行**

**时间格式是否正确**

**service crond status**

**service crond start**

**查看开始是否启动crond**

**chkconfig|grep crond 2 3 4 5 为on则为开机启动**

**crond 0:off 1:off 2:on 3:on 4:on 5:on 6:off**

### Ntp

Check ntp on

1查看ntp服务状态

service ntpd status

2ntp启动

service ntpd start restart stop

3查看

watch ntpq -p

### Alias

用户家目录下 .bashrc

Ls /etcprofile.d

Unalias ls

### 通配符

Ls \*.txt 多个字符

Ls ?.txt 一个字符

Ls [123].txt

Ls [0-9].tx

Cat 1.txx>2.txt 覆盖

Cat 1.txt>>2.txt 追加

ls aaa2>a.txt 正确的结果

ls aaa 2>>a.txt错误的结果

ls aaa2&>a.txt 正确的结果

ls aaa 2&>>a.txt错误的结果

### 暂停

Ctrl +z 暂停

Bg

Fg

### Cut

cat /etc/passwd |head -2|cut -d ":" -f 1,2,3

### sort

sort /etc/passwd

### tee

sort ab.txt |uniq -c |tee -a 22.txt

### 正则表达式

**Grep ‘ro\*r’ passwd**

**零个或者多个前面的字符o**

**Grep ‘ro.r’ passwd**

**表示任意一个字符**

**Grep ‘ro?t’ passwd**

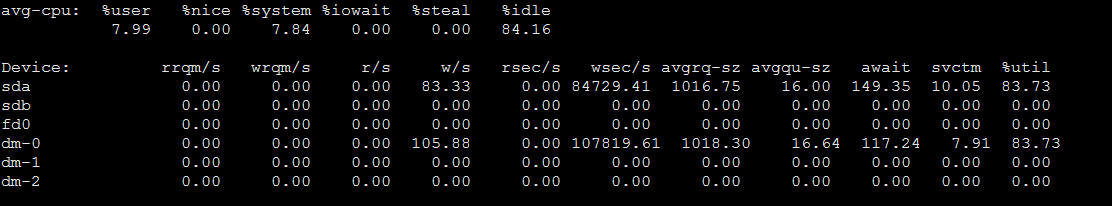
**零个或者一个o**

**Grep ‘ro+t’**

**1个或者多个前面的字符**

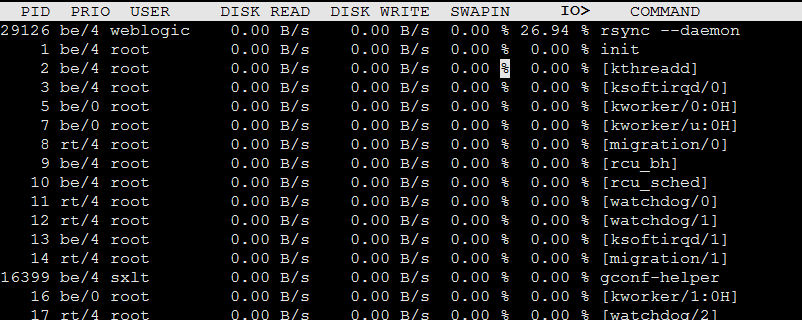
### Iostat

查看iostat -x 1 一秒刷新一次



### Iotop

Iotop -o 只显示有io操作的进程



### Wc

统计有多少个httpd请求

ps -ef|grep httpd |grep -v grep | wc -l

-l 统计行数

-c 统计字节数

-m 统计字符数

### Tcpdump

tcpdump tcp port 80 -i eth2 -w tcpdump2018.pcap

### 查询关键字并输出关键字所在行号

grep -n ERROR sssSvr5501.log

Head -100 (Tail -f 行号-100 )

### 统计文件包含某个字符

grep -o E022登录会话已经超时 wsbsSvr601.log |wc -l

Grep -o 只输出文件中匹配的部分

Wc -o 统计行数

### 测试系统io

iostat -d -x -k 1 100

### Scp

Scp weblogic@192.168.51.2:/home/weblogic/Apps/app2/clearlogs.sh /home/weblogic/Apps/app2/

### Nmon

D

Io读写情况查看

### Jdk卸载安装

rpm -qa | grpe jdk

rpm -e java-1.7.0-openjdk-1.7.0.79-2.5.5.4.el6.x86\_64 --nodeps

vim /etc/profile

**#set java environment**

**JAVA\_HOME=/usr/java/sunjava6\_64**

**JRE\_HOME=$JAVA\_HOME/jre**

**CLASS\_PATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib**

**PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin**

**export JAVA\_HOME**

**export JRE\_HOME**

**export CLASS\_PATH**

**export PATH**

source /etc/profile

java -version

### Mount

mount -t nfs 192.168.50.2:/software/appstatic /software/appstatic

vim /etc/exports

**/software/appstatic \*(rw,no\_root\_squash)**

exportfs –a

service nfs restart 有肯能没有启动服务

挂载镜像文件

mount -t iso9660 -o,loop /home/sxlt/rhel-server-6.7-x86\_64-dvd.iso /mnt/

### 解锁

**pam\_tally2 --user=weblogic --reset**

### Yum源配置

**1上传ios文件到home/sxlt/下**

**2 mount -o loop /home/sxlt/rhel-server-6.7-x86\_64-dvd.iso /mnt/**

**3 vim vim /etc/yum.repos.d/local.repo**

**[local.repo]**

**name=local.repo**

**baseurl=file:///mnt**

**enabled=1**

**gpgcheck=0**

**yum -y install tree**

### 修改密码

**chattr -i /etc/passwd /etc/group /etc/shadow /etc/gshadow**

**echo 密码 | passwd --stdin weblogic**

**chattr +i /etc/passwd /etc/group /etc/shadow /etc/gshadow**

### ln 源端 目标端

ln -s /weblogic/user\_projects/domains/sssl\_domain/ ln\_weblogic

正确删除软连接 rm -rf ln\_weblogic

错误删除软连接 rm -rf ln\_weblogic/ 连文件都删除了 比较苦逼 小心

### Ls

**ls -ltr --time-style="+%y-%m-%d %H:%M:%S"**

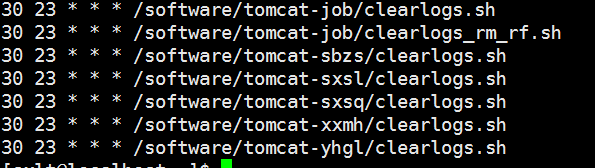
## 脚本

### 清空当前目录下所有日志

**清空当前目录下所有以log和out结尾的文件 结合定时任务**

**例如每天晚上十点钟清空日志**

**Crontab -l**



for i in `find . -name "\*.log" -o -name "\*.out" ` ;do cat /dev/null>$i;done;

**Summery:**

**定时任务查看命令 crontab -l**

**定时任务编辑命令 crontab -e**

**定时任务格式：**

**\* \* \* \* \* commond**

**分时日月周 命令**

### Tomcat启动脚本

#!/bin/bash

source ~/.bash\_profile

cd `dirname $0`

TOMCAT\_HOME=`pwd`

TOMCAT\_DIR='tomcat-sbzs'

echo "TOMCAT\_HOME:$TOMCAT\_HOME"

echo "TOMCAT\_DIR:$TOMCAT\_DIR"

# kill taxcp

TAXCP=`ps -ef|grep java|grep ${TOMCAT\_DIR}|grep ${TOMCAT\_HOME}|grep -v grep|awk '{print $2}'|xargs`

echo `ps -ef|grep java|grep ${TOMCAT\_DIR}|grep ${TOMCAT\_HOME}`

echo "[Kill Old's Pid:$TAXCP]"

if [[ "$TAXCP" ]] ;then

kill -9 $TAXCP

fi

cd $TOMCAT\_HOME

rm -rf logs/\*

rm -rf work/\*

rm -rf temp/\*

rm -rf \*.log

curl -s "http://192.168.51.32:6789/confCenter/conf?action=reload&confTable=conf\_default" > /dev/null

export TAXCP\_OPTS="-DsubGroup=sxds -DconfCenterAddr=http://192.168.51.32:6789/confCenter/conf -DconfTable=conf\_default -Ddubbo.registry.file=/tmp/.dubbo/dubbo-registry-sbzs.cache"

cd $TOMCAT\_HOME

/software/${TOMCAT\_DIR}/bin/catalina.sh start

tail -f ./logs/catalina.out

**Summery:**

**temp用户缓存目录**

**Work 是jsp转换成class的工作目录**

**重启删除temp和work防止缓存导致发布项目不生效**

**User.xml里面有tomca配置了tomcat的manger用户名和密码**

**Service.xml里面配置了端口号**

### Zookeeper启动脚本

port1=3881;

port2=3882;

port3=3883;

#根据端口号查询对应的pid

pid1=$(netstat -nlp | grep :$port1 | awk '{print $7}' | awk -F"/" '{ print $1 }');

pid2=$(netstat -nlp | grep :$port2 | awk '{print $7}' | awk -F"/" '{ print $1 }');

pid3=$(netstat -nlp | grep :$port3 | awk '{print $7}' | awk -F"/" '{ print $1 }');

#杀掉对应的进程，如果pid不存在，则不执行

if [ -n "$port1" ]; then

kill -9 $pid1;

fi

#杀掉对应的进程，如果pid不存在，则不执行

if [ -n "$port2" ]; then

kill -9 $pid2;

fi

#杀掉对应的进程，如果pid不存在，则不执行

if [ -n "$port3" ]; then

kill -9 $pid3;

Fi

**Summery：**

**netstat -lnp | grep 3881**

**netstat -nlp | grep 3881 | awk '{print $7}' | awk -F"/" '{ print $1 }'**

**根据端口号查询pid 然后根据pid kill 掉进程**

注意awk的使用

## Linux Kernel or Directory

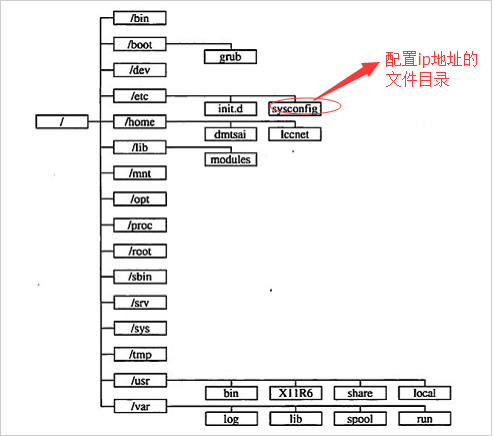
### Linux简介

Linux是一套免费使用和自由传播的类Unix操作系统，是一个基于POSIX和UNIX的多用户、多任务、支持多线程和多CPU的操作系统。它能运行主要的UNIX工具软件、应用程序和网络协议。它支持32位和64位硬件。Linux继承了Unix以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。Redhat的图形界面被称为X-Window 所以切换到图形界面命令为stratxsss

### Linux结构

**内核、用户界面、文件结构、实用工具**

#### 文件图





#### 文件结构

**/**

根目录,所有的目录、文件、设备都在/之下，/就是Linux文件系统的组织者，也是最上级的领导者。

**/bin**

bin就是二进制（binary）英文缩写。在一般的系统当中，你都可以在这个目录下找到linux常用的命令。系统所需要的那些命令位于此目录，比如ls、cp、mkdir等命令；功能和/usr/bin类似，这个目录中的文件都是可执行的、普通用户都可以使用的命令。作为基础系统所需要的最基础的命令就是放在这里。

**/boot**

Linux的内核及引导系统程序所需要的文件目录，比如vmlinuzinitrd.img文件都位于这个目录中。 在一般情 况下，GRUB或LILO系统引导管理器也位于这个目录。

**/cdrom**

这个目录在你刚刚安装系统的时候是空的。你可以将光驱文件系统挂在这个目录下。

例如：mount/dev/cdrom /cdrom

**/dev**

dev是设备(device)的英文缩写。这个目录对所有的用户都十分重要。因为在这个目录中包含了所有linux系统中 使用的外部设备。但是这里并不是放的外部设备的驱动程序。 这一点和我们常用的windows,dos操作系统不一样。 它实际上是一个访问这些外部设备的端口。我们可以非常方便地去访问这些外部设备，和访问一个文件，一个目录没有任 何区别。

**/etc**

etc这个目录是linux系统中最重要的目录之一。在这个目录下存放了系统管理时要用到的各种配 置文件和子目 录。我们要用到的网络配置文件，文件系统，x系统配置文件， 设备配置信息，设置 用户信息等都在这个目录下。

一般发行版的/etc目录下通常有

rc0.d-rc6.d目录，里面对应包含了从init0-init6不同运行级所要启动的命令，

rc.d目录下有 8个目录和3个文件，其中目录为rc0.d-rc6.d，分别代表了各个run-level的脚本，

目录中文件名称均以S及K开头，后接两位 数字与该服务名称，这些文件都是link文件，

指向/etc/rc.d/init.d目录。在rc3.d目录中，以S开头的文件为执行该服务，以K开头的文件则是杀掉该服务的意思。

数字代表启动顺序。

**/home**

如果我们建立一个用户，用户名是"xx",那么在/home目录下就有一个对应的/home/xx路径，用来 存放用户的主 目录。

**/lib**

lib是库（library）英文缩写。这个目录是用来存放系统动态连接共享库的。几乎所有的应用程 序都会用到这 个目录下的共享库。因此，千万不要轻易对这个目录进行什么操 作，一旦发生问题， 你的系统就不能工作了。

**/lost+found**

在ext2或ext3文件系统中，当系统意外崩溃或机器意外关机，而产生一些文件碎片放在这里。 当系统启动的 过程中fsck工具会检查这里，并修复已经损坏的文件系统。有时系 统发生问题，有很 多的文件被移到这个目 录中，可能会用手工的方式来修复，或移到文件到原来的位置上。

**/mnt**

这个目录一般是用于存放挂载储存设备的挂载目录的，比如有cdrom等目录。可以参看/etc/fstab 的定义。有时我 们可以把让系统开机自动挂载文件系统，把挂载点放在这里也是可以的。主要看/etc/fstab 中怎么定义了；比如光驱可以 挂载到/mnt/cdrom。

**/media**

有些linux的发行版使用这个目录来挂载那些usb接口的移动硬盘(包括U盘)、CD/DVD

驱动器等等。

**/opt**

这里主要存放那些可选的程序。你想尝试最新的firefox测试版吗?那就装到/opt目录下吧，这样， 当你尝试 完，想删掉firefox的时候，你就可以直接删除它，而不影响系统 其他任何设置。安装到 /opt目录下的程序， 它所有的数据、库文件等等都是放在同个目录下面。

**/proc**

可以在这个目录下获取系统信息。这些信息是在内存中，由系统自己产生的。操作系统运行时， 进程信息 及内核信息（比如cpu、硬盘分区、内存信息等）存放在这里。/proc 目录伪装的文件系统 proc的挂载目录，proc 并不是真正的文件系统，它的定义可以参见/etc/fstab

**/root**

Linux超级权限用户root的家目录。

**/sbin**

这个目录是用来存放系统管理员的系统管理程序。大多是涉及系统管理的命令的存放，是超级权 限用户 root 的可执行命令存放地，普通用户无权限执行这个目录下的命令，这个目录和 /usr/sbin;/usr/X11R6/sbin或 /usr/local/sbin 目录是相似的；我们记住就行了，凡是目录sbin中包含 的都是root权限才能执行的。

**/selinux**

好像是对SElinux的一些配置文件目录，SElinux可以让你的linux更加安全。

**/srv**

服务启动后，所需访问的数据目录，举个例子来说，www服务启动读取的网页数据就可以

放在/srv/www中

**/sys**

Linux内核中设计较新的一种虚拟的基于内存的文件系统，它的作用与proc有些类似，但除了与 proc相同的具 有查看和设定内核参数功能之外，还有为Linux统一设备模型作为 管理之用。

**/tmp**

临时文件目录，用来存放不同程序执行时产生的临时文件。有时用户运行程序的时候，会产生临 时文件。 /tmp就用来存放临时文件的。/var/tmp目录和这个目录相似。

**/usr**

这是linux系统中占用硬盘空间最大的目录。用户的很多应用程序和文件都存放在这个目录下。 在这个目录下，

你可以找到那些不适合放在/bin或/etc目录下的额外的工具。比如像游戏阿，一些打印 工具拉等等。/usr目录包含了 许多子目录：/usr/bin目录用于存放程序;/usr/share用于存放一些共享的 数据，比如音乐文件或者图标等等;

**/usr/lib**

目录用于存放那些不能直接运行的，但却是许多程序运行所必需的一些函数库文件。你的软件包 管理器(应 该是“新立得”吧)会自动帮你管理好/usr目录的。

**/usr/local**

这里主要存放那些手动安装的软件，即不是通过“新立得”或apt-get安装的软件。它和/usr目 录具有相类似 的目录结构。让软件包管理器来管理/usr目录，

而把自定义的脚 本(scripts)放到/usr/local目录下面，我想这应该是个不错的主意。

**/usr/share**

系统共用的东西存放地，比如/usr/share/fonts是字体目录，

/usr/share/doc和/usr/share/man帮助文件。

**/var**

这个目录的内容是经常变动的，看名字就知道，我们可以理解为vary的缩写，/var下有/var/log 这是用来存放 系统日志的目录。

/var/www目录是定义Apache服务器站点存放目录；

**/var/lib**

用来存放一些库文件，比如MySQL的，以及MySQL数据库的的存放地。

配置系统  .bash\_profile

环境变量  配置用户的环境变量

环境变量：配置.bashrc文件可以指定某些程序在用户登录的时候就自动启动

 　　　　系统环境变量设置：在root目录下  Vi .bash\_profile

修改PATH=$PATH:$HOME/bin :/安装目录/bin

需要退出（quit），重新登陆

当希望临时加入某个环境变量：用export PATH =$PATH : $HOME /bin:/root/test/t1

已定义好的环境变量：

SHELL:默认的shell

PATH:路径

USER：当前登录用户的用户名

显示变量内容：

echo $PATH

echo $USER

echo $SHELL

也可以使用env命令显示环境变量

通配符：

\* 代表多个字母或数字

？  一个

#### 重启

shutdown - h now 立刻关机

shutdown - r now 重启

reboot 重启

#### 设备硬盘操作

挂载设备

查看设备：使用命令“fdisk –l”可以查看系统的存储设备

挂载设备 ：首先使用mkdir命令建立挂载点目录，然后再使用mount命令挂载相关设备

mkfs [选项][-t <文件系统类型>] [设备名称] [区块数]

说明：把指定的设备格式为指定的文件系统。

查看磁盘使用情况

df [-参数] 比如， df-l

查看某个目录是在哪个分区

df [目录全路径]

查看Linux系统分区具体情况

fdish -l

#### 网络配置

配置网络接口可以使用三种不同的工具来完成：

使用网络接口配置程序netconfig

使用图形配置工具

使用终端命令ifconfig

监控网络状态信息:

显示网络统计信息的命令netstat

此命令用来显示整个系统目前的网络情况。例如目前的连接、数据包传递数据 、或路由表的内容。

如 netstat -an? netstat -anp|more (查看网络端口的使用的情况)

显示数据包经过历程命令：traceroute

route 查看路由表

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.追踪路由 ：tracert 目标ip/域名

2.测试两个ip是否畅通：ping 目标IP

3.window下查看IP情况：ipconfig

4.Linux/unix下查看IP情况：ifconfig

5.linux 网络环境配置：

第一种：

(1) 用root身份登陆，运行setup命令进入到text mode setup utility

对网络进行配置，这里可以进行ip、子网掩码、默认网管、dns的设置

(2) 这时网卡的配置没有立即生效，运行

/etc/rc.d/init.d/network restart命令我们刚才作的配置生效。

第二种：

(1)ifconfig eth0 x.x.x.x对网卡进行设置

(2)infconfig eth0 network x.x.x.x对子网掩码设置

对广播地址和dns使用默认的

注意：这样配置网络会立即生效，但是是临时生效

?

第三种：

(1) 修改/etc/sysconfig/network-scripts/ifcfg-eth0

这个文件的各个属性可以修改，包括ip，子网掩码、广播地址、默认网关

(2)这是网卡的配置没有生效

运行/etc/rc.d/init.d/network restart 命令我们刚才做的设置才生效

Linux中的所有设备都是文件，这种方法是最底层的方法，永久性修改

VMware下Linux和window ping

(1) windows下ipconfig查看VM网卡IP

(2) 配置Linux的IP使两者处于同一个子网的 即可以ping通

ifconfig <设备名> <IP地址> netmask <掩码>

例如：

ifconfig eth0 192.168.6.11 netmask 255.255.255.0

在网络配置界面中，通过“激活”或者“解除”按钮可以启动或者禁用网络接口，

网络控制程序network

/etc/rc.d/init.d/network? start|stop|restart

命令ifconfig :

ifconfig <设备名> [up|down]

命令ifup/ifdown

ifup eth0

ifdown eth0

配置文件名

功能

/etc/sysconfig/network

最基本的网络信息，系统启动时读取该文件

/etc/sysconfig/network-scripts/\*

此目录下的文件是系统启动时用来初始化网络的一些信息，例如：第一块以太网卡对应的文件为ifcfg-eth0

/etc/host.conf

域名解析的控制文件

/etc/hosts

域名或主机名与IP地址的映射文件

/etc/resolv.conf

域名服务器设置文件

/etc/protocols

定义使用的网络互联协议及协议号

/etc/services

设定主机的不同端口的网络服务

可以在 vim /etc/rc.d/rc.local 中加入 ifconfig eth0 192.168.35.0

每次启动的时候都会重新将ip设置为 192.168.35.0

#### 安装软件

获取最新版本的Apache源代码

将源代码解压缩

./configure –- prefix=/home/myapache

执行编译命令：make

执行安装命令：make install?

账号和用户管理

用户和组的配置信息保存在以下三个文件中：

/etc/passwd

/etc/shadow

/etc/group

2.10 更改文件所有者命令chown

chown [选项] user[:group] <file>...

更改文件访问权限命令chmod

chmod [选项] <mode> <file>...

mode：[ugoa][[+-=][rwxX]...][,…]，

mode也可以用数字来表示权限：

chmod abc file

a,b,c各为一个数字，分别表示User、Group、及Other的权限。权限是关于可读（r）、可写(w)、可执行(r)三个属性设置 值的和，其中r=4，w=2，x=1，

例如：

若要rwx属性，则4+2+1=7；

若要rw-属性，则4+2=6；

若要r-x属性，则4+1=7

#### 进程管理

Linux操作系统包括三种不同类型的进程，每种进程都有自己的特点和属性：

交互进程:由shell启动的进程。

批处理进程:这种进程和终端没有联系，是一个进程序列。

守护进程:在后台持续运行的进程。

前台启动:一般地，用户键入一个命令，就已经启动了一个前台的进程。

后台启动 :对于非常耗时进程，可以然进程在后台运行。从后台启动进程其实就是在命令结尾加上一个“&”号

每个进程，都会对应一个父进程，而这个父进程可以复制多个子进程

每个进程都可能以两种方式存在：后台或前台

进程就是正在执行的程序

显示系统执行的进程：

ps命令，可以不加任何参数

1. ps -a :显示当前终端的所有进程信息

2：ps -u :以用户的格式显示进程信息

3. ps -x :显示后台进程运行的参数

ps -aux

终止进程：kill/killall

终止某个进程：kill 进程号? 如：kill 6251

kill -q 5222 ：因为某些进程会捕捉某些信息，如果直接不能结束进程，可以使用“? -q”传送信息

动态监控进程：top

top和ps类似，top在执行一段时间可以更新正在进行的进程

1.监视特定用户

top：输入此命令,按回车键，查看执行的进程

u：然后输入“u” 回车，在输入用户名即可

2.终止指定的进程

top：

k: 输入k回车，再输入要结束的进程ID

3.top -d 10 :指定系统更新进程的时间为10秒

按小q退出

2.16：软件安装

1.linux JDK的安装

a.把\*\*.iso 文件挂载在虚拟机上做好配置

mount /mnt/cdrom

unmount /mnt/cdrom

b.把安装文件拷贝到 /home

cp 文件 /home

c.cd /home

d.安装

./\*\*.bin

e.看看文件 /etc/profile[环境配置文件]

f.配置刚才自己安装的JDK

2.eclipse

a.安装文件拷贝到 /home

b.安装

? tar -zxvf \*\*.tar.gz

c.启动eclipse[进入图形界面]

startx ./eclipse

./eclipse & 后台方式运行

#### RPM管理

介绍：一种用于互联网下载包的打包及安装工具，它包含在某些Linux分发版中。它生成 具有.RPM扩展名的文件。

RPM是RedHat Package Manager（RedHat软件包管理工具）的缩写

RPM包的名称格式：apache-1.3.23-11.i386.rpm

\*\*RPM 常用命令

rpm -qa:查询所安装的所有rpm软件包

rpm -qa | more

rpm -pa | grep x

rpm -q 软件包名 （查询软件包是否安装）

rpm -q foo

rpm -qi 软件包名：查询软件包信息

rpm -ql file

rpm -ql 软件包名 ：查询软件包中的文件

rpm -ql file

rpm -qf文件全路径名：查询文件所属的软件包

rpm -qf /etc/passwd

rpm -qp包文件名：查询包的信息、对这个软件包的介绍

rpm -qp jdk-1-5\_linux -i586.rpm

安装rpm包：

rpm -i RPM包路径名称：安装包到当前系统

i = install

rpm -ivh RPM包全路径名称：安装包到当前系统有提示信息

参数说明：i = install 安装

v = verbase 提示

h = hash 进度条

删除rpm包：rpm -e jdk

如果其它软件包依赖于您要卸载的软件包，卸载时则会产生错误的信息

如果忽略这个错误信息继续卸载，使用 --nodeps命令行选项

升级rpm包：

rpm -u RPM包全路径包

作为命令语言互

#### Xshell连接Liunx

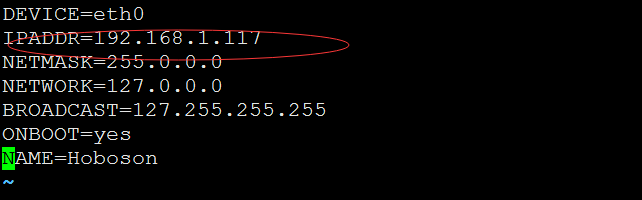
刚刚安装好liunx操作系统 ifconfig时候可能没有配置eth0，无法远程连接，

1设置ip:

cd /etc/sysconfig/network-scripts

cat ifcfg-lo>ifcfg-eht0

vim eth0



*临时设置IP*

*ifconfig eth0 192.168.1.155 netmask 255.255.255.0*

*2 在客户端ping所设置的ip ping 192.168.1.115*

*3 在服务器端telnet 192.168.1.115 22*

*4 关闭防火墙*

*1) 重启后生效*

*开启： chkconfig iptables on*

*关闭： chkconfig iptables off*

*2) 即时生效，重启后失效*

*开启： service iptables start*

*关闭： service iptables stop*

#### 内核

内核是操作系统的核心，具有很多最基本功能，它负责管理系统的进程、内存、设备驱动程序、

文件和网络系统，决定着系统的性能和稳定性。

Linux 内核由如下几部分组成：内存管理、进程管理、设备驱动程序、文件系统和网络管理等

#### 安装

打开vt否则无法安装

安装完毕后 设置ip 关闭防火墙

#### 启动过程

**启动第一步：加载bios(Basic Input Output System)**

当你打开计算机电源，计算机会首先加载BIOS信息，BIOS信息是如此的重要，以至于计算机必须在最开始就找到它。这是因为BIOS中包含了CPU的相关信息、设备启动顺序信息、硬盘信息、内存信息、时钟信息、PnP特性等等。在此之后，计算机心里就有谱了，知道应该去读取哪个硬件设备了

**启动第二部：读取MBR(Master Boot Record)**

众所周知，硬盘上第0磁道第一个扇区被称为MBR，也就是Master Boot Record，即主引导记录，它的大小是512字节，别看地方不大，可里面却存放了预启动信息、分区表信息。

系统找到BIOS所指定的硬盘的MBR后，就会将其复制到0×7c00地址所在的物理内存中。其实被复制到物理内存的内容就是Boot Loader，而具体到你的电脑，那就是lilo或者grub了

**启动第三部：Boot Loader**

Boot Loader 就是在操作系统内核运行之前运行的一段小程序。通过这段小程序，我们可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核做好一切准备。 Boot Loader有若干种，其中Grub、Lilo和spfdisk是常见的Loader。 我们以Grub为例来讲解吧，毕竟用lilo和spfdisk的人并不多。 系统读取内存中的grub配置信息（一般为menu.lst或grub.lst），并依照此配置信息来启动不同的操作系统。

**启动第四步－－加载内核**根据grub设定的内核映像所在路径，系统读取内存映像，并进行解压缩操作。此时，屏幕一般会输出“Uncompressing Linux”的提示。当解压缩内核完成后，屏幕输出“OK, booting the kernel”。 系统将解压后的内核放置在内存之中，并调用start\_kernel()函数来启动一系列的初始化函数并初始化各种设备，完成Linux核心环境的建立。至此，Linux内核已经建立起来了，基于Linux的程序应该可以正常运行了**。 启动第五步－－用户层init依据inittab文件来设定运**行等级 内核被加载后，第一个运行的程序便是/sbin/init，该文件会读取/etc/inittab文件，并依据此文件来进行初始化工作。 其实/etc/inittab文件最主要的作用就是设定Linux的运行等级，其设定形式是“：id:5:initdefault:”，这就表明Linux需要运行在等级5上。Linux的运行等级设定如下： 0：关机 1：单用户模式 2：无网络支持的多用户模式 3：有网络支持的多用户模式 4：保留，未使用 5：有网络支持有X-Window支持的多用户模式 6：重新引导系统，即重启 关于/etc/inittab**文件的学问，其实还有很多 启动第六步－－init**进程执行rc.sysinit 在设定了运行等级后，Linux系统执行的第一个用户层文件就是/etc/rc.d/rc.sysinit脚本程序，它做的工作非常多，包括设定PATH、设定网络配置（/etc/sysconfig/network）、启动swap分区、设定/proc等等。如果你有兴趣，可以到/etc/rc.d中查看一下rc.sysinit**文件，里面的脚本够你看几天**的 启动第七步－－启动内核模块 具体是依据/etc/modules.conf文件或/etc/modules**.d目录下的文件来装载内核模块。 启动**第八步－－执行不同运行级别的脚本程序 根据运行级别的不同，系统会运行rc0.d到rc6.d中的相应的脚本程序，来完**成相应的初始化工作和启动相应的服务。 启动第九步－－**执行/etc/rc.d/rc.local 你如果打开了此文件，里面有一句话，读过之后，你就会对此命令的作用一目了然： # This script will be executed \*after\* all the other init scripts. # You can put your own initialization stuff in here if you don’t # want to do the full Sys V style init stuff. rc.local就是在一切初始化工作后，Linux留给用户进行个性**化的地方。你可以把你想设置和启动的东西放到这里。 启动**第十步－－执行/bin/login程序，进入登录状态 此时，系统已经进入到了等待用户输入username和password的时候了，你已经可以用自己的帐号登入系统了。:) 漫长的启动过程结束了，一切都清静了… 其实在这背后，还有着更加复杂的底层函数调用，等待着你去研究…本文就算抛砖引玉了:) 本文参考了如下文章，精炼荟萃而成： http://bbs.chinaunix.net/thread-835918-1-1.html http://hi.baidu.com/fembed/blog/item/b9f0881f51145866f624e4be.html http://baike.baidu.com/view/9485.htm

*Linux系统有7个运行级别(runlevel)：*

1. *运行级别0：系统停机状态，系统默认运行级别不能设为0，否则不能正常启动*
2. *运行级别1：单用户工作状态，root权限，用于系统维护，禁止远程登陆*
3. *运行级别2：多用户状态(没有NFS)*
4. *运行级别3：完全的多用户状态(有NFS)，登陆后进入控制台命令行模式*
5. *运行级别4：系统未使用，保留*
6. *运行级别5：X11控制台，登陆后进入图形GUI模式*
7. *运行级别6：系统正常关闭并重启，默认运行级别不能设为6，否则不能正常启动*

## Linux Common Commands

### 命令的操作

more more [选项]? <file>…分屏显示命令

less less [选项] <filename> 按页显示命令

clear 清除屏幕命令

history 查看历史命令记录

### 目录相关命令

cd .. 可进入上一层目录

cd - 进入上一个进入的目录

cd ~ ?可进入用户的home目录

pwd? 显示当前在哪个路径

ls 列出文件和目录

ls -a? 显示隐藏文件

ls -l 显示常列表格式

mkdir 建立目录

rmdir? 删除空目录

touch? 建立空文件

### 文件用户权限

2.4.1文件操作

cp 复制命令

将档案 aaa 复制(已存在),并命名为 bbb:

cp aaa bbb

将所有的C语言程式拷贝至 Finished 子目录中:

cp \*.c Finished

cp -r dir1 dir2? 递归复制（复制子目录信息）

mv ?移动文件和改文件 将档案 aaa 更名为 bbb:

mv aaa bbb

将所有的C语言程式移至 Finished 子目录中:　　mv -i \*.c

rm 删除文件和目录

***删除目录时，必须具有对其父目录的写权限。***

rm -rf \* 删除所有内容（包括目录和文件）

如何修改文件的访问权限

chmod 777 along

范例 :将档案 file1.txt 设为所有人皆可读取:

chmod ugo+r file1.txt

将档案 file1.txt 设为所有人皆可读取:

chmod a+r file1.txt

将档案 file1.txt 与 file2.txt 设为该档案拥有者,与其所属同一个群体者可写入,但其他以外的人则不可写入:

chmod ug+w,o-w file1.txt file2.txt

将 ex1.py 设定为只有该档案拥有者可以执行:

chmod u+x ex1.py

将目前目录下的所有档案与子目录皆设为任何人可读取:

chmod -R a+r \*

### 用户组操作

?su-? 切换成系统管理员

su 用户之间的切换

logout 用户注销

useradd? xiaoming? 添加用户（root下）

passwd xiaoming? 设置xiaoming的密码

userdel? xiaoming? 删除用户

userdel -r xiaoming删除用户及其主目录

每个用户必须属于某一个组，不能独立于组外。

每个文件有所有者、所在组、其它组的概念

\*1，所有者：一般指创建者

用ls -ahl 命令可以查看文件的所有者

用chown 用户名 文件名 来修改文件的所有者

\*2，文件所在组 ----用户所在的组

ls -ahl? 可以看见文件的所有组

chgrp 组名 文件名 修改文件所在组

\*3，其它组---除开文件的所有者和所在组的用户外，

? 系统其他用户都是文件的其他组

添加组 ---groupadd policeman

查看Linux中所有组 ： vi /etc/group 查看该文件或cat /etc/group | more

创建用户并指定将该用户分配到哪个组

useradd -g 组名 用户名

查看Linux中所有用户信息：vi /etc/passwd 或者cat /etc/passwd

文件权限：-----分为三种：r-可读，用四表示

?| | | w-可写，用2表示

-|rw-|r--|r--? x-可执行，用1表示

?| | |

上图中由右向左：

第1部分：其它组的用户对该文件的权限

第2部分：文件所在组对该文件的权限

第3部分：文件的所有者对该文件的权限

第4部分：文件类型 ---普通文件(-),目录(d),链接(l)

-d <dirName>：指定用户主目录，默认情况下，将会在/home目录下新建一个与用户名相同的用户主目录

删除用户的命令为userdel，该命令的格式为：userdel <用户名>

修改用户属性

usermod –g<主组名> -G <组名> -d <用户主目录> -s <用户shell>

在添加用户时，可以指定将该用户添加到哪个组中，同样的用root的管理权限可以改变

某个用户所在的组：usermod -g 组名 用户名

可以用 usermod -d 目录名 用户名 改变该用户登录的初始目录?

增加用户组

groupadd <新组名>

删除用户组

? groupdel <组名>

修改组成员：直接编辑/etc/group文件，将用户名写到对应的组名的后面

whoami命令的功能在于显示用户自身的用户名。

who [选项]：该命令主要用于查看当前在线的用户情况

w命令 ：用于显示登录到系统的用户情况

finger命令可用于查找和显示用户信息，并且在查找后显示指定账号的相关信息

chfn命令能够改变系统存储的用户信息

?切换用户身份：su [用户名]

### 帮助命令

man? <command> info <command> help [command]

**alias**

命令：alias显示系统当前定义的所有alias

alias cp ='cp-i'

alias ll='ls -l --color = tty'

### 文件查看连接命令

cat cat [选项] <file1> …

more? ?显示文件内容，带分页

less 显示文件内容，带分页

grep 在文本中查询指定内容

|管道命令[把上一个命令的结果给|后的命令处理]

grep "shunping" aaa.java

grep -n ......? 显示行数?

grep -n "shunping" aaa.java > kkk.bak 重定向命令

ls -l > a.txt? 列表的内容写入文件 a.txt（覆盖写）

ls -l >> a.txt 追加写到文件的末尾

find的使用：

在特定目录下搜索并显示指定名称的文件和目录，搜索一段时间内被存 取/变? 更的文件或目录。

find /home -amin -10 十分钟内存取的文件和目录

find /home -atime -10 十小时

find /home -cmin -10 十分钟内更改过的

find /home -size +10k 大小为10k的

将目前目录及其子目录下所有延伸档名是 c 的档案列出来。

# find . -name "\*.c"

# find. -name \\*.c

将目前目录其其下子目录中所有一般档案列出

# find . -type f

将目前目录及其子目录下所有最近 20 分钟内更新过的档案列出

# find . -ctime -20

### Vi编辑命令

vi有三种基本工作模式，分别是：命令模式（command mode）、插入模式（insert mode）和底行模式（last line mode）

进入vi：

命令“vi 文件名”

命令“vi”，在退出vi时再指定文件名

选项“+n”，表示希望在进入vi之后，光标处于文件中第n行上，

选项“+”表示希望在进入vi之后光标处于文件最末行。

**i o a I OＡ　分别为：插入在光标左边　插入在光标下一行　插入在光标右侧　　插入在该行行头　插入在该行上一行　插入在该行行尾**

保存文件:

1.在命令模式下,连按两次大写字母<Z>。

2.在末行模式下:

?:w vi保存当前编辑的文件，但并不退出vi，而是继续等待用户输入命令。 :w <newfile>

?:w! <newfile>? 把当前文件的内容保存到指定的文件newfile中，如果newfile已经存在，则覆盖原有内容。

在末行模式下，有四种方法可以退出vi返回到shell：

?:q 系统退出vi返回到shell。在用此命令时，若编辑的文件没有被保存，则vi在窗口的最末行给出提示信息。

?:q!? vi放弃所作修改而直接退到shell下。

?:wq 先保存文件，然后再退出vi返回到shell。

?:x? 该命令的功能与命令模式下的ZZ命令功能相同

0（数字0） 移到当前行的行首

$? 移到当前行的行尾

|  |  |  |
| --- | --- | --- |
| h或向左方向键 | 光标向左移动一个字符 | j或者向下方向键 |
| j或者向下方向键 | 光标向下移动一个字符 | K或者向上方向键 |
| K或者向上方向键 | 光标向上移动一个字符 | l或者向右方向键 |
| l或者向右方向键 | 光标向右移动一个字符 | Ctrl + f 或者pageUP键 |
| Ctrl + f 或者pageUP键 | 屏幕向前移动一页 | Ctrl + b 或者pageDOWN键 |
| Ctrl + b 或者pageDOWN键 | 屏幕向后移动一页 | Shift+6或者0（数字0） |
| Shift+6或者0（数字0） | 移动到本行行首 | Shift+4 |
| Shift+4 | 即’$’移动到本行行尾 | G |
| G | 光标移动到文本的最末行 | nG（n是数字） |
| nG（n是数字） | 移动到该文本的第n行 | gg |
| gg | 移动带该文本的首行 | **一般模式下查找与替换** |
| **一般模式下查找与替换** |  |  |

|  |  |  |
| --- | --- | --- |
| /word | 向光标之后寻找一个字符串名为word的字符串，当找到第一个word后，按”n”继续搜后一个 | ?word |
| ?word | 想光标之前寻找一个字符串名为word的字符串，当找到第一个word后，按”n”继续搜前一个 | :n1,n2s/word1/word2/g |
| :n1,n2s/word1/word2/g | 在n1和n2行间查找word1这个字符串并替换为word2，你也可以把”/”换成”#” | :1,$s/word1/word2/g |
| :1,$s/word1/word2/g | 从第一行到最末行，查找word1并替换成word2 | **一般模式下删除、复制粘贴** |
| **一般模式下删除、复制粘贴** |  |  |

|  |  |
| --- | --- |
| x,X | x为向后删除一个字符，X为向前删除一个字符 |
| nx（n为数字） | 向后删除n个字符 |
| dd | 删除光标所在的那一行 |
| ndd（n为数字） | 删除光标所在的向下n行 |
| d1G | 删除光标所在行到第一行的所有数据 |
| dG | 删除光标所在行到末行的所有数据 |
| yy | 复制光标所在的那行 |
| nyy | 复制从光标所在行起向下n行 |
| p,P | p复制的数据从光标下一行粘贴，P则从光标上一行粘贴 |
| y1G | 复制光标所在行到第一行的所有数据 |
| yG | 复制光标所在行到末行的所有数据 |
| J | 讲光标所在行与下一行的数据结合成同一行 |
| u | 还原过去的操作 |

|  |
| --- |
| **进入编辑模式** |

|  |  |
| --- | --- |
| i | 在当前字符前插入字符 |
| I | 在当前行行首插入字符 |
| a | 在当前字符后插入字符 |
| A | 在当前行行末插入字符 |
| o | 在当前行下插入新的一行 |
| O | 在当前行上插入新的一行 |
| r | 替换光标所在的字符，只替换一次 |
| R | 一直替换光标所在的字符，一直到按下ESC |

|  |
| --- |
| **命令模式** |

|  |  |
| --- | --- |
| :w | 将编辑过的文本保存 |
| :w! | 若文本属性为只读时，强制保存 |
| :q | 退出vim |
| :q! | 不管编辑或未编辑都不保存退出 |
| :wq | 保存，退出 |
| :e! | 将文档还原成最原始状态 |
| ZZ | 若文档没有改动，则不储存离开，若文档改动过，则储存后离开，等同于:wq |
| :w [filename] | 编辑后的文档另存为filename |
| :r [filename] | 在当前光标所在行的下面读入filename文档的内容 |
| :set nu | 在每行的行首显示行号 |
| :set nonu | 取消行号 |
| n1,n2 w [filename] | 将n1到n2的内容另存为filename这个文档 |
| :! command | 暂时离开vim运行某个linux命令，例如 :! ls /home 暂时列出/home目录下的文件，然后会提示按回车回到vim |

### More less

more命令功能：让画面在显示满一页时暂停，此时可按空格健继续显示下一个画面，或按Q键停止显示。

less命令功能：less命令的用法与more命令类似，也可以用来浏览超过一页的文件。所不同的是less命

令**除了可以按空格键向下显示文件外，还可以利用上下键来卷动文件。当要结束浏览时，只要在less命**

**令的提示符“: ”下按Q键即可。**

### 打包 压缩 解压

*打包： tar -cvf Hoboson.tar Hoboson*

*压缩： tar -zcvf Hoboson.tar.gz Hoboson*

*压缩： tar -jcvf Hoboson.tar.gz2 Hoboson*

*在选项f之后的文件档名是自己取的，我们习惯上都用 .tar 来作为辨识。*

*如果加z选项，则以.tar.gz或.tgz来代表gzip 压缩过的tar包；如果加j选项，则以.tar.bz2来作为tar包名*

*查看tar.gz有哪些内容*

*Tar -zxvf Hoboson.tar.gz*

*由于我们使用 gzip 压缩的log.tar.gz，所以要查阅log.tar.gz包内的文件时，就得要加上z这个选项了。*

### 防火墙

检查是否安装防火墙：rpm -qa|grep iptables

service iptables status

1）立即关闭，并非永久关闭 service iptables stop 2）永久关闭 iptables -F 关闭防火墙功能 chkconfig iptables off 禁止防火墙启动 另外，运行 setup 在界面，选择Firewall configuration，进入下一界面，选择 Security Level为Disabled，保存

### Ftp或者Sftp链接

*A: 文件传送协议FTP(File Transfer Protocol)是TCP/IP协议簇中的一个成员，也是现在因特网上最广泛的文件传送协议。FTP协议包括两个部分，一个是FTP客户端，另一个是FTP服务器。当然，FTP服务器是用来存储文件资源的，FTP客户端通过访问FTP服务器来获得资源的。*

*B: 安全文件传送协议SFTP(Secure File Transfer Protocol)可以为文件传送提供安全的加密/解密技术。基本语法和FTP差不多。SFTP是SSH的一部分，在SSH软件包中，已经包含了一个SFTP(Secure File Transfer Protocol)的安全文件传输子系统，SFTP本身没有单独的守护进程，它必须使用sshd守护进程（端口号默认是22）来完成相应的连接操作。由于这种传输方式使用了加密/解密技术，文件传送相对来说是很安全的，但是是有代价的，它的传输效率比FTP要低得多*

test

## hell Summary

### Shell

动式地解释和执行用户输入的命令只是shell功能的一个方面

另外shell还可以进行程序设计，他提供了定义变量和参数的手段以及丰富的程序控制结构。使用shell类似于DOS中的 批处理文件称为shell script

shell有很多，常用的有3种：/bin/sh? /bin/csh /bin/ksh

查看shell种类 ：ls -l /bin/\*sh

1.查看目前使用的是哪种shell

env [该命令可以显示当前操作系统的环境变量]

2.shell的修改:

直接输入：chsh -s 输入新的shell 如：/bin/csh

### At

在shell提示符下输入”at 时间”，然后按回车键。这时在下一行shell会等待用户继续输入要执行的命令。每一行输入一 个命令，所有命令都输入完毕后按Ctrl+d键结束。

将各个命令写入shell脚本中，然后使用下面格式设置在指定时间执行shell脚本中的命令：

at 时间 –f脚本文件。

### batch命令

crone命令在系统启动时由一个shell脚本自动启动，进入后台。

cron启动后搜索/var/spool/cron目录，寻找以/etc/passwd文件中的用户名命名的crontab文件，被找到的这种文件将载入内 存。

如果没有crontab文件，就转入“休眠”状态，释放系统资源。

cron每分钟“醒”过来一次，查看当前是否有需要运行的命令。

如果发现某个用户设置了crontab文件，它将以该用户的身份去运行文件中指定的命令。命令执行结束后，任何输出都将 作为邮件发送给crontab的所有者，或者/etc/crontab文件中MAILTO环境变量中指定的用户。

内存查看命令free

磁盘空间用量查看命令df

### 设置系统日期

1.date命令。可以直接输入date 来查看系统时间

2.利用date命令来更改系统时间

date MMDDHHMCCYY.SS:指定月月日日时时分分年年年年.秒秒

3.查看日历：cal 3 2002? ：查看2001年3月的日历

4.查看年历：cal 2008

### 任务调度命令

任务调度指系统在某个时间执行的特定的命令和程序

任务调度分类：

1.系统工作：有些重要的工作必须周而复始的执行，如病毒扫描.....

2.个别用户工作：个别用户可能希望执行某些程序

置任务调度文件：/etc/crontab

设置个人任务调度：执行crontab -e命令

接着输入任务到调度文件

如：5\*\*\*\* ls -l /etc/ > /tmp/to.txt

意思是说每小时的第五分钟执行ls -l /etc/ > /tmp/to.txt

任务调度的使用：

1.设置任务 crontab -e

2.每隔一定时间去执行 data > /home/mydata1

希望每天凌晨2：00执行 date >> /home/mydate

02\*\*\* date >> /home/mydate

调度文件的规则：

字段名称 ?说明? 范围

分钟 每小时第几分钟 0~59

小时 每日的第几个小时 0~23

日期 每月的第几天 1~31

日历 ?每年的第几个月 1~12

星期 每周的第几天 0~6

3.怎么样调度多个任务？

a.在crontab -e 中直接写

b.可以把所有的任务写入一个可执行文件（shell编程）

例子： vi mytask.sh

date >> /home/mydate

cp /home/mydate /root

chmod 744 mytask.sh

crontab -e

\*\*\*\*\* /root/mytask.sh

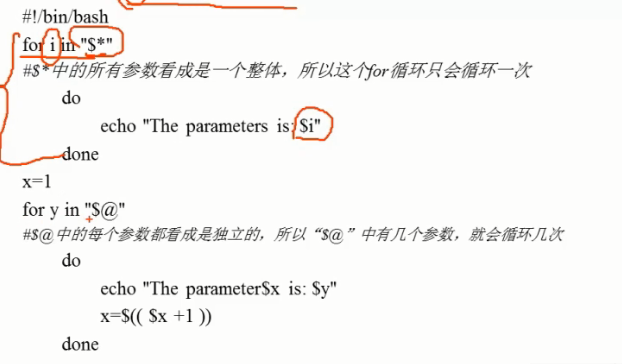
终止任务调度： crontab -r ：终止任务调度

? crontab -l ：列出当前有哪些任务调度

### 多命令执行



### $\*和$@区别



**Test**

## linux系统日常管理

### 命令w

查看当前负载情况

*第一行从左面开始显示的信息依次为：时间，系统运行时间，登录用户数，平均负载*

*第二行开始以及下面所有的行，告诉我们的信息是，当前登录的都有哪些用户，以及他们是从哪里登录的等等*

### 修改密码

Xshell登陆linux后，再次输入passwd 则出现：

Changing password for user root.

New password: root..

Retype new password: root..

passwd: all authentication tokens updated successfully.

密码修改成功

### 破解root密码

redhat忘记root密码的解决方法

redhat默认安装方式下用的是grub引导的，所以在出现启动菜单时（就是选择引导菜单的时候），

就可以按“e”键进入编辑状态，具体操作如下：

1.**在出现grub画面时，用上下键选中你平时启动Linux的那一项，然后按e键**

2. **再次用上下键选中你平时启动linux的那一项(类似于kernel /boot/vmlinuz-2.4.18-14 ro　root=LABEL=/)，**

**然后按e键**

3. **修改你现在见到的命令行，加入空格single，结果如下：kernel /boot/vmlinuz-2.4.18-14 roroot=LABEL=/ single**

**4. 回车返回，然后按b键启动，即可直接进入linux命令行**

**5. 在#提示符下直接输入命令：passwd root 然后写入你的新密码**

**6. #提示符下输入reboot 重启**

## Linux杂谈

### Unix特点

*可靠新强*

*伸缩性好*

*开放性好*

*植根与网络*

*面型数据库使用*

### 运行级别

***Linux系统有7个运行级别(runlevel)****运行级别0：系统停机状态，系统默认运行级别不能设为0，否则不能正常启动 运行级别1：单用户工作状态，root权限，用于系统维护，禁止远程登陆 运行级别2：多用户状态(没有NFS) 运行级别3：完全的多用户状态(有NFS)，登陆后进入控制台命令行模式 运行级别4：系统未使用，保留 运行级别5：X11控制台，登陆后进入图形GUI模式 运行级别6：系统正常关闭并重启，默认运行级别不能设为6，否*则不能正常启动

### 用户账号的添加、删除与修改

增加用户

Useradd hoboson

Passwd hoboson

修改用户名

Passwd 回车修改自己的密码

Passwd hoboson 回车修改hoboson用户的密码

usermod -l **hoboson0316** hoboson

删除用户

userdel hoboson0316

查看用户

cat /etc/passwd

显示内容如下

**〔用户名〕：〔密码〕：〔UID〕：〔GID〕：〔身份描述〕：〔主目录〕：〔登陆Shell〕**

**root:x:0:0:root:/root:/bin/bash**

**bin:x:1:1:bin:/bin:/sbin/nologin**

**daemon:x:2:2:daemon:/sbin:/sbin/nologin**

**adm:x:3:4:adm:/var/adm:/sbin/nologin**

**lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin**

**sync:x:5:0:sync:/sbin:/bin/sync**

**shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown**

**halt:x:7:0:halt:/sbin:/sbin/halt**

**mail:x:8:12:mail:/var/spool/mail:/sbin/nologin**

**news:x:9:13:news:/etc/news:**

**uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin**

**operator:x:11:0:operator:/root:/sbin/nologin**

# Mongodb

## 启动

./mongod --dbpath=/data/db/ --rest

## 增加

db.getCollection('CsZlpzSwsxZb').insert({

"\_id" : ObjectId("591eac0b25f81819f8cb5531"),

"swsxmc" : "中国税收居民身份证明申请表",

"swsxDm" : "SXA051005001",

"flzlDm" : "001382",

"\_class" : "com.foresee.dzswj.zlpz.api.vo.CsZlpzSwsxZb"

});

## 删除

db.getCollection('CsZlpzSwsxZb').remove({"\_id" : ObjectId("591eac0b25f81819f8cb5531")})

## 查询语句

查询整表

db.getCollection('CodeDataConfiguration').find({})

根据登记序号查询

db.getCollection('ysqxxZcbwVO').find({"djxh" : "10126101001036920970"});

根据登记序号和业务编码查询

db.getCollection('ysqxxZcbwVO').find({"$and" : [ { "djxh" : "10126101001036920970"},{"ywbm":"SZYSSB"}]});

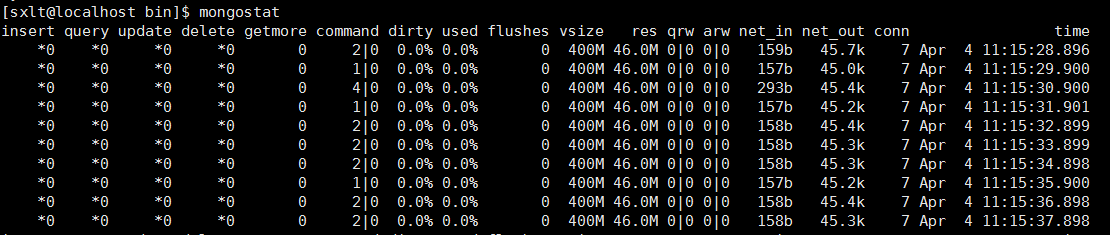
## 创建索引

db.getCollection('ysqxxVO').ensureIndex({"ysqxxSb.yzpzxh":1});

db.getCollection('ysqxxVO').getIndex();

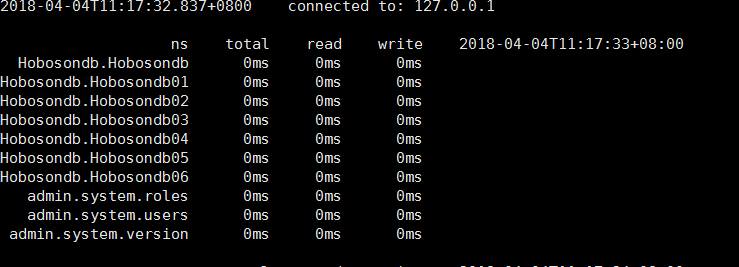
## 查看状态

mongostat



## 读写时间查看

Mongotop



## 数据库锁的使用情况

mongotop --locks

## 排序

db.getCollection('zlpzSxVO').find().sort({lrrq:-1})

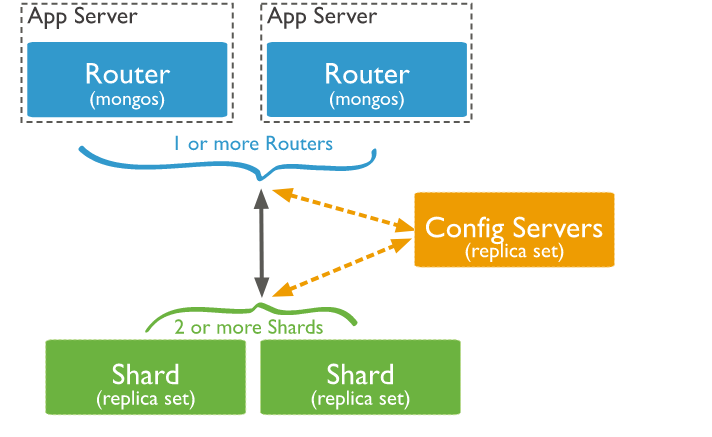
## 版本配置副本集+分片

### 一：概念

在搭建集群之前，需要首先了解几个概念：路由，分片，副本集，配置服务器等

相关概念

先看一张图



从图中可以看到有四个组件：mongos，config server，shard，replica set。

**Mongos：**数据库集群请求的入口，所有的请求都通过mongos进行协调，不需要在应用程勋添加一个路由选择器，mongos自己就是一个请求分发中心，它负责把对应的数据请求请求转发到对应的shard服务器上，在生产环境通常有多mongos作为请求的入口，防止其中一个挂掉所有的mongodb请求都没有办法操作。

**Config server：**顾名思义为配置服务器，存储所有数据库元信息（路由，分片）的配置。Mongos本身没有物理存储分片服务器和数据路由信息，只是缓存在内存里，配置服务器则实际存储这些数据。Mongos第一次启动或者关掉重启就会从config server 加载配置信息，以后如果配置服务器信息变化会通知到所有的mongos更新自己的状态，这样mongos就能继续准备路由，在生产环境通常有多个config server 配置服务器，因为它存储了分片路由的元数据，防止数据丢失！

**Shard：** 分片（sharding）是指将数据库拆分，将其分散在不同的机器上的过程。将数据分散到不同的机器上，不需要功能强大的服务器就可以存储更多的数据和处理更大的负载。基本思想就是将集合切成小块，这些块分散到诺干片里，每个偏只负责总数据的一部分，最后通过一个均衡器来对各个分片进行均衡（数据迁移）。

**Replica set:** 中文翻译副本集，其实就是shard的备份，防止shard挂掉之后数据丢失。复制提供了数据的冗余备份，并在多个服务器上存储数据副本，提高了数据的可用性，并可以保证数据的安全性。

**仲裁者（Arbiter）,是复制集中的一个MongoDB实例，它并不保存数据，仲裁节点使用最小的资源并且不要求硬件设备，不能将Arbiter部署在同一个数据集节点中，可以部署在其他应用服务器或者监视服务器中，也可以部署在单独的虚拟机中，为了确保复制集中有技术的投票成员（包括primary），需要添加仲裁节点作为投票，否则primary不能运行时不会自动切换primary。**

**简单了解之后，我们可以这样总结一下，应用请求mongos来操作mongodb的增删改查，配置服务器存储数据库元信息，并且和mongos做同步，数据最终存入在shard（分片）上，为了防止数据丢失同步在副本集中存储了一份，仲裁在数据存储到分片的时候决定存储到哪个节点**

### 二：准备环境

### Os：Red Hat Enterprise Linux Server release 6.5 (Santiago)

**Software**：mongodb-linux-x86\_64-rhel62-3.4.4.tgz

vim /etc/profile

export MONGODB\_HOME=/usr/local/mongodb

export PATH=$MONGODB\_HOME/bin:$PATH#

source /etc/profile

### 端口分配

**mongos： 27017**

**config： 21000**

**shard1： 27001**

**shard2： 27002**

**shard3： 27003**

### 三：服务器规划

| **192.168.51.36** | **192.168.51.37** | **192.168.51.38** |
| --- | --- | --- |
| mongos | mongos | mongos |
| config server | config server | config server |
| shard server1 主节点 | shard server1 副节点 | shard server1 仲裁 |
| shard server2 仲裁 | shard server2 主节点 | shard server2 副节点 |
| shard server3 副节点 | shard server3 仲裁 | shard server3 主节点 |

### 安装MongoDB

**#解压**

**tar -xzvf mongodb-linux-x86\_64-3.4.6.tgz -C /usr/local/**

**#改名**

**mv mongodb-linux-x86\_64-3.4.6 mongodb**

### 分别在每台机器建立conf、mongos、config、shard1、shard2、shard3六个目录，因为mongos不存储数据，只需要建立日志文件目录即可。

**mkdir -p /usr/local/mongodb/conf**

**mkdir -p /usr/local/mongodb/mongos/log**

**mkdir -p /usr/local/mongodb/config/data**

**mkdir -p /usr/local/mongodb/config/log**

**mkdir -p /usr/local/mongodb/shard1/data**

**mkdir -p /usr/local/mongodb/shard1/log**

**mkdir -p /usr/local/mongodb/shard2/data**

**mkdir -p /usr/local/mongodb/shard2/log**

**mkdir -p /usr/local/mongodb/shard3/data**

**mkdir -p /usr/local/mongodb/shard3/log**

### 配置环境变量

**vim /etc/profile**

**# 内容**

**export MONGODB\_HOME=/usr/local/mongodb**

**export PATH=$MONGODB\_HOME/bin:$PATH**

**# 使立即生效**

**source /etc/profile**

### Config server配置服务器（三台机器）

mongodb3.4以后要求配置服务器也创建副本集，不然集群搭建不成功，特别注意此处和3.4以前的版本有不一样

添加配置文件（三台服务器都要进行添加）

#### vim /software/mongodb/conf/config.conf

**## 配置文件内容**

**pidfilepath = /usr/local/mongodb/config/log/configsrv.pid**

**dbpath = /usr/local/mongodb/config/data**

**logpath = /usr/local/mongodb/config/log/congigsrv.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 21000**

**fork = true**

**#declare this is a config db of a cluster;**

**configsvr = true**

**#副本集名称**

**replSet=configs**

**#设置最大连接数**

**maxConns=20000**

启动三台服务器的config server

mongod -f /usr/local/mongodb/conf/config.conf

登录任意一台conf配置中心

mongo --port 20000

**config = {\_id : "configs", members : [{\_id : 0, host : "192.168.51.36:21000" },{\_id : 1, host : "192.168.51.37:21000" },{\_id : 2, host : "192.168.51.38:21000" }]};**

初始化副本集

rs.initiate(config);

### 五：配置分片副本集（三台机器）

#### 设置第一个副本集分片

**vi /usr/local/mongodb/conf/shard1.conf**

**#dfilepath = /usr/local/mongodb/shard1/log/shard1.pid**

**dbpath = /usr/local/mongodb/shard1/data**

**logpath = /usr/local/mongodb/shard1/log/shard1.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 27001**

**fork = true**

**#打开web监控**

**httpinterface=true**

**rest=true**

**#副本集名称**

**replSet=shard1**

**#declare this is a shard db of a cluster;**

**shardsvr = true**

**#设置最大连接数**

**maxConns=20000**

启动

mongod -f /usr/local/mongodb/conf/shard1.conf

登陆任意一台服务器，初始化副本集，不要登录到arbiterOnly这台机器

mongo --port 20000

use admin

**config = {\_id : "shard1",members : [{\_id : 0, host : "192.168.51.38:27001" },{\_id : 1, host : "192.168.51.37:27001" },{\_id : 2, host : "192.168.51.36:27001", arbiterOnly: true }]}**

初始化副本集

rs.initiate(config);

查看状态

rs.status();

#### 配置第二个副本集分片

**vi /usr/local/mongodb/conf/shard2.conf**

**#配置文件内容**

**#——————————————–**

**pidfilepath = /usr/local/mongodb/shard2/log/shard2.pid**

**dbpath = /usr/local/mongodb/shard2/data**

**logpath = /usr/local/mongodb/shard2/log/shard2.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 27002**

**fork = true**

**#打开web监控**

**httpinterface=true**

**rest=true**

**#副本集名称**

**replSet=shard2**

**#declare this is a shard db of a cluster;**

**shardsvr = true**

**#设置最大连接数**

**maxConns=20000**

启动

mongod -f /software/mongodb/conf/shard2.conf

登陆任意一台服务器，初始化副本集，不要登录到arbiterOnly这台机器

mongo --port 20000

use admin

**config = {\_id : "shard2",members : [{\_id : 0, host : "192.168.51.38:27002" },{\_id : 1, host : "192.168.51.36:27002" },{\_id : 2, host : "192.168.51.37:27002", arbiterOnly: true }]}**

初始化副本集

rs.initiate(config);

查看状态

rs.status();

#### 设置第三个副本集分片

**vi /usr/local/mongodb/conf/shard3.conf**

**#配置文件内容**

**#=======================**

**pidfilepath = /usr/local/mongodb/shard3/log/shard3.pid**

**dbpath = /usr/local/mongodb/shard3/data**

**logpath = /usr/local/mongodb/shard3/log/shard3.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 27003**

**fork = true**

**#打开web监控**

**httpinterface=true**

**rest=true**

**#副本集名称**

**replSet=shard3**

**#declare this is a shard db of a cluster;**

**shardsvr = true**

**#设置最大连接数**

**maxConns=20000**

启动

mongod -f /software/mongodb/conf/shard3.conf

登陆任意一台服务器，初始化副本集，不要登录到arbiterOnly这台机器

mongo --port 20000

use admin

**config = {\_id : "shard3",members : [{\_id : 0, host : "192.168.51.36:27003" },{\_id : 1, host : "192.168.51.37:27003" },{\_id : 2, host : "192.168.51.38:27003", arbiterOnly: true }]}**

初始化

rs.initiate(config);

查看状态

rs.status();

### 六：配置路由器服务器mongos

先启动配置服务器和分片服务器，后启动路由实例

启动路由实例

vi /software/mongodb/conf/mongos.conf

**#dfilepath = /usr/local/mongodb/mongos/log/mongos.pid**

**logpath = /usr/local/mongodb/mongos/log/mongos.log**

**logappend = true**

**bind\_ip = 0.0.0.0**

**port = 20000**

**fork = true**

**#监听的配置服务器,只能有1个或者3个 configs为配置服务器的副本集名字**

**configdb = configs/192.168.51.36:21000,192.168.51.37:21000,192.168.51.38:21000**

**#设置最大连接数**

**maxConns=20000**

启动服务

mongos -f /software/mongodb/conf/mongos.conf

启用分片

目前搭建了mongodb配置服务器、路由服务器，各个分片服务器，不过应用程序连接到mongos路由服务器并不能使用分片机制，还需要在程序里设置分片配置，让分片生效。

mongo --port 20000

use admin

**sh.addShard("shard1/192.168.51.36:27001,192.168.51.37:27001,192.168.51.38:27001")**

**sh.addShard("shard2/192.168.51.36:27002,192.168.51.37:27002,192.168.51.38:27002")**

**sh.addShard("shard3/192.168.51.36:27003,192.168.51.37:27003,192.168.51.38:27003")**

查看状态

sh.status();

### 七：测试

目前配置服务、路由服务、分片服务、副本集服务都已经串联起来了，但我们的目的是希望插入数据，数据能够自动分片。连接在mongos上，准备让指定的数据库、指定的集合分片生效。

#指定testdb分片生效

Use admin

db.runCommand( { enablesharding :"testdb"});

#指定数据库里需要分片的集合和片键

db.runCommand( { shardcollection : "testdb.table1",key : {id: 1} } )

我们设置testdb的 table1 表需要分片，根据 id 自动分片到 shard1 ，shard2，shard3 上面去。要这样设置是因为不是所有mongodb 的数据库和表 都需要分片！

测试分片配置结果

我们设置testdb的 table1 表需要分片，根据 id 自动分片到 shard1 ，shard2，shard3 上面去。要这样设置是因为不是所有mongodb 的数据库和表 都需要分片！

测试分片配置结果

mongo 127.0.0.1:27017

#使用testdb

use testdb1;

#插入测试数据

**db.table1.save({id:1111,"test1":"testval1"});**

**db.table1.save({id:2222,"test1":"testval2"});**

**db.table1.save({id:3333,"test1":"testval3"});**

**db.table1.save({id:4444,"test1":"testval4"});**

**db.table1.save({id:5555,"test1":"testval5"});**

#查看分片情况如下，部分无关信息省掉了

db.table1.stats();

{

"sharded" : true,

"ns" : "testdb.table1",

"count" : 100000,

"numExtents" : 13,

"size" : 5600000,

"storageSize" : 22372352,

"totalIndexSize" : 6213760,

"indexSizes" : {

"\_id\_" : 3335808,

"id\_1" : 2877952

},

"avgObjSize" : 56,

"nindexes" : 2,

"nchunks" : 3,

"shards" : {

"shard1" : {

"ns" : "testdb.table1",

"count" : 42183,

"size" : 0,

...

"ok" : 1

},

"shard2" : {

"ns" : "testdb.table1",

"count" : 38937,

"size" : 2180472,

...

"ok" : 1

},

"shard3" : {

"ns" : "testdb.table1",

"count" :18880,

"size" : 3419528,

...

"ok" : 1

}

},

"ok" : 1

}

### 八：启动关闭

mongodb的启动顺序是，先启动配置服务器，在启动分片，最后启动mongos.

**mongod -f /software/mongodb/conf/config.conf**

**mongod -f /software/mongodb/conf/shard1.conf**

**mongod -f /software/mongodb/conf/shard2.conf**

**mongod -f /software/mongodb/conf/shard3.conf**

**mongos -f /software/mongodb/conf/mongos.conf**

关闭时，直接killall杀掉所有进程

**killall mongod**

**killall mongos**

注意分片的时候由于chunksize值默认是64MB，所以需要插入大量的数据才能够进行分片，此处将chunksize值设置小点，能够看到明显的分片效果

修改chunksize进行测试分片

use config

db.settings.save( { \_id:"chunksize", value: 10 } )

### 九：添加用户

mongos> use admin

switched to db admin

mongos> db.createUser({user:'admin',pwd:'123456', roles:[{role:'root', db:'admin'}]})

Successfully added user: {

"user" : "admin",

"roles" : [

{

"role" : "root",

"db" : "admin"

}

]

}

mongos> use dzswjdb

switched to db dzswjdb

mongos> db.createUser({user:'dzswjdb',pwd:'123456', roles:[{role:'readWrite', db:'dzswjdb'}]})

Successfully added user: {

"user" : "dzswjdb",

"roles" : [

{

"role" : "readWrite",

"db" : "dzswjdb"

}

]

}

mongos> use sbzsdb

switched to db sbzsdb

mongos> db.createUser({user:sbzsdb,pwd:'123456', roles:[{role:'readWrite', db:sbzsdb}]})

Successfully added user: {

"user" : "sbzsdb",

"roles" : [

{

"role" : "readWrite",

"db" : "sbzsdb"

}

]

}

mongos>

### 九：启动认证

将所有的服务先进行关闭

创建key存放的路径

mkdir /usr/local/mongodb/key

然后在所有的shard，config，mongos的配置文件中添加如下配置

clusterAuthMode=keyFile

keyFile=/usr/local/mongodb/key/mongdb-keyfile

生成密码配置文件

openssl rand -base64 100 > /usr/local/mongodb/key/mongdb-keyfile

chmod 600 /usr/local/mongodb/key/mongdb-keyfile

将密码文件复制到另外的两台机器上

启动mongo集群，先启动配置服务器---->分片服务器----> mongos服务器

验证认证是否生效

[root@mongodb1 mongos]# mongo --port 20000

MongoDB shell version v3.4.4

connecting to: mongodb://127.0.0.1:20000/

MongoDB server version: 3.4.4

mongos> use admin

switched to db admin

mongos> show users;

2017-11-14T11:04:48.072+0800 E QUERY [thread1] Error: not authorized on admin to execute command { usersInfo: 1.0 } :

\_getErrorWithCode@src/mongo/shell/utils.js:25:13

DB.prototype.getUsers@src/mongo/shell/db.js:1537:1

shellHelper.show@src/mongo/shell/utils.js:752:9

shellHelper@src/mongo/shell/utils.js:659:15

@(shellhelp2):1:1

mongos> db.auth('admin','123456')

1

mongos> show users;

{

"\_id" : "admin.admin",

"user" : "admin",

"db" : "admin",

"roles" : [

{

"role" : "root",

"db" : "admin"

}

]

}

mongos>

### 十：参考文档

<https://www.cnblogs.com/ityouknow/p/7344005.html>

<https://docs.mongodb.com/manual/core/index-creation/>

### 十一：常用命令

导入导出通过最高权限来进行导入导出，就不会报认证错误

导出命令参数

**-h IP**

**-u 用户**

**-p 端口**

**-d 数据库名字**

**-c 集合名字**

**-o 导出的文件名字**

**-q 导出过滤条件**

#### Mongodb导出sbzsdb库数据

**mongodump -h 192.168.51.38 -u admin -p 123456 -d dzswjdb -o /usr/local/src/ --authenticationDatabase admin**

#### mongodb导入sbzs库数据

**mongorestore -h 192.168.51.38 -u admin -p 123456 -d dzswjdb /usr/local/src/dzswjdb --authenticationDatabase admin**

导入数据

mongorestore /usr/local/src/

#### Mongodb导出单张表

**mongodump -h 156.16.42.133 -u admin -p 123456 -d dzswjdb -c CodeDataConfiguration -o /usr/local/src/codedataconfiguration --authenticationDatabase admin**

#### ---创建数据库 然后创建表

**use sbzsdb**

**db.sbzsdb.insert({'name':'sbzstest'})**

**db.sbzs.insert({'name':'sbzstest'})**

#### --删除数据库

**db**

**db.dropDatabase();**

#### 查看主节点

**./mongo**

**use amdin**

**db.auth('admin','System@!2018');**

**sh.status();**

**db.databases.find()**

#### 集合操作相当于rdbms中的table

**db.createCollection('CodeDataConfiguration')**

**db.getCollection('CodeDataConfiguration').find({})**

**db.getCollection('CodeDataConfiguration').find({'\_id':ObjectId("595f3159b8555378d7513b32")})**

**db.getCollection('CodeDataConfiguration').drop()**

## 创建索引

# Zookeeper

## 安装

1修改操作系统的/etc/hosts文件

#zookeeper server

192.168.85.211 zookeeper-server-1

2下载zookeeper

[http://apache.fayea.com/zookeeper/zookeeper-3.3.6/zookeeper-3.3.6.tar.gz](http://apache.fayea.com/zookeeper/zookeeper-3.3.6/)

3解压

tar -zxvf zookeeper-3.3.6.tar.gz

4 在解压目录下创建文件夹

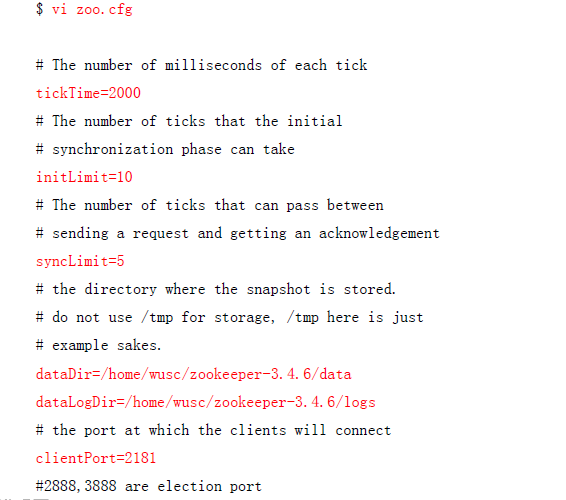
mkdir /zookeeper-3.3.6/data

mkdir /zookeeper-3.3.6/logs

5将conf下的zoo\_sample.cfg cp

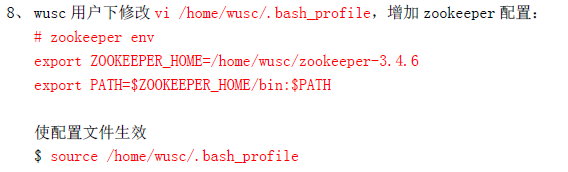
cp zoo\_sample.cfg zoo.cfg

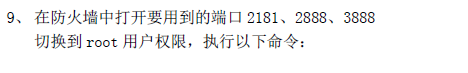
6修改配置文件

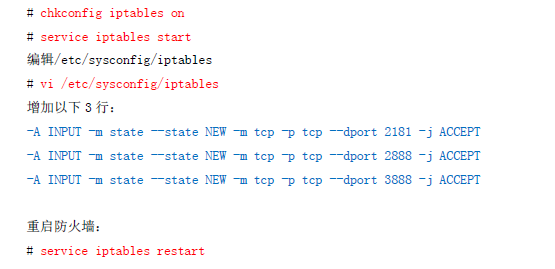


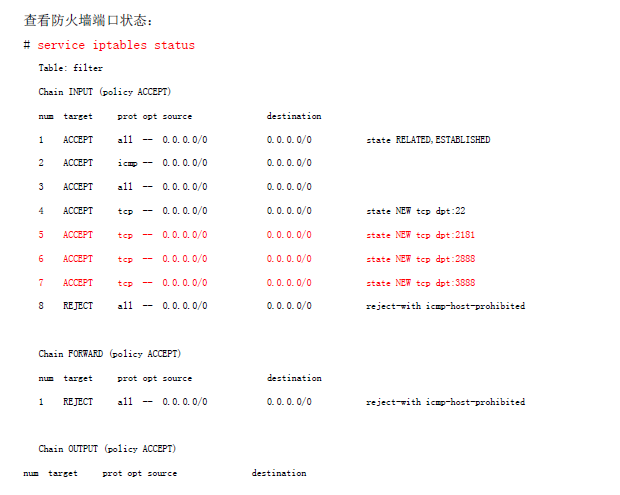
clipboard.png

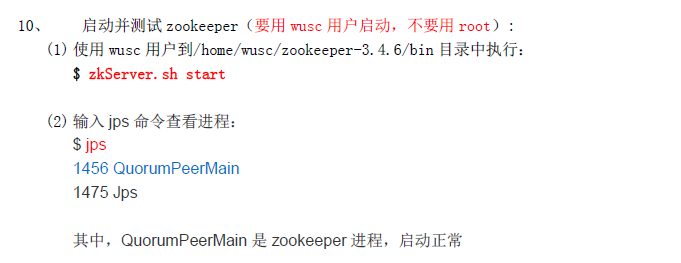




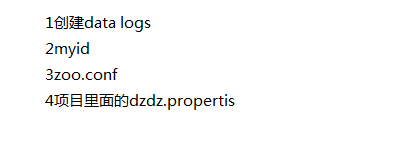








## 总结配置项如下



**Dzdz. 更新时候注意发布目录的dzdz.propertis连接的那个zookeeper**

**Fbrp.properties**

**连接zookeeper时候先找fbrp.propertis如果没有配置则去找dzda.properties fbrp.properties优先级高**

**现在生产zookeeper用的是156.12.19.11没有用10上的**

**做集群复制zookeeper需要修改以下内容**

**1 Myid**

**2 Zoo.cfg port path logpath**

**3 给别的节点添加增加节点**

## Mongodb查询速度慢

日志分析：92.12.67.70:root/Root@!6770

拷贝节点下shard1.log 日志到92.12.67.70：/software/mtools/log

分析日志

mloginfo before.log --queries --sort count

分析日志图形

mplotqueries before.log --group operation

# Mysql安装手册

下载地址：http://ftp.ntu.edu.tw/MySQL/Downloads/MySQL-5.6/

本次安装的mysql版本：mysql-5.6.35-linux-glibc2.5-x86\_64.tar.gz

服务器版本：Red Hat Enterprise Linux Server release 7.1 (Santiago)

版本包上传路径：/usr/local/src

安装之前先安装依赖包：

yum install -y perl-Module-Install.noarch

创建mysql安装目录

tar -zxvf mysql-5.6.35-linux-glibc2.5-x86\_64.tar.gz –C /usr/local/mysql

创建mysql组

groupadd mysql

添加msyql用户

useradd -r -g mysql mysql -d /usr/local/mysql

给/usr/local/mysql文件夹进行授权

chown -R mysql:mysql /usr/local/mysql/

将配置文件拷贝到/etc目录下

cp /usr/local/mysql/support-files/my-default.cnf /etc/my.cnf

修改my.cnf配置文件

# These are commonly set, remove the # and set as required.

basedir = /usr/local/mysql

datadir = /data

# port = .....

# server\_id = .....

# socket = .....

修改环境变量

echo 'export PATH=/usr/local/mysql/bin:$PATH'>> /etc/profile

source /etc/profile

配置mysql启动服务

cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld

chmod +x /etc/init.d/mysqld

修改mysqld服务配置文件

basedir=/usr/local/mysql

datadir=/data

添加mysqld服务

chkconfig --add mysqld

chkconfig mysqld on

初始化数据库

/usr/local/mysql/scripts/mysql\_install\_db --defaults-file=/etc/my.cnf --basedir=/usr/local/mysql --datadir=/data --user=mysql

启动数据库

service  mysqld  start

如果使用别的安装目录，启动会报如下错误

[root@gsds-wwsjk1 bin]# service mysqld start

Starting MySQL.Logging to '/data/mysql/gsds-wwsjk1.err'.

171016 17:12:35 mysqld\_safe The file /usr/local/mysql/bin/mysqld

does not exist or is not executable. Please cd to the mysql installation

directory and restart this script from there as follows:

./bin/mysqld\_safe&

See http://dev.mysql.com/doc/mysql/en/mysqld-safe.html for more information

ERROR! The server quit without updating PID file (/data/mysql/gsds-wwsjk1.pid).

解决方法：

ln -s /software/mysql/bin/mysqld /usr/local/mysql/bin/mysqld

切换用户

su - mysql

执行下面的安全初始化设置root密码

安全初始化数据库

/usr/local/mysql/bin/mysql\_secure\_installation

这可以设置密码

通过mysql用户进行启动数据库，得到的结果如下所示，都是通过mysql用户启动的进程

root@topsec mysql]# ps -ef | grep mysqld

mysql     1884     1  0 14:41 pts/1    00:00:00 /bin/sh /usr/local/mysql/bin/mysqld\_safe --datadir=/data --pid-file=/data/topsec.com. pid

mysql     2020  1884  2 14:41 pts/1    00:00:00 /usr/local/mysql/bin/mysqld --basedir=/usr/local/mysql --datadir=/data --plugin-dir=/

usr/local/mysql/lib/plugin --log-error=/data/topsec.com.err --pid-file=/data/topsec.com.pid

登录数据库

mysql -uroot -p123456

show databases 查看数据库

use mysql 进入数据库

show tables 查看数据库里面的表

如果数据库忘记root密码，则可以通过下面的方式重置root密码

service mysqld start --skip-grant-tables

>use mysql  
>update user set password=password("123456") where user="root";  
>flush privileges;

grant all privileges on \*.\* to system@'localhost' identified by '123456' with grant option; ---定义system用户，localhost访问

grant all privileges on \*.\* to root@'156.12.19.42' identified by '123456' with grant option; ---定义root用户，localhost访问

grant all privileges on \*.\* to system@'%' identified by '123456' with grant option; ---定义system用户，任何地方都能够进行访问

grant all privileges on gzdssbzs.\* to gzdssbzs@'%' identified by 'csgjRds168' with grant option; ---定义某个用户对某个库的超级权限

flush privileges; ---刷新保存配置

数据库导出操作

mysqldump -usystem -p123456 lhywzxdev > lhywzxdev-zyg.sql

数据库导入操作

mysql -usystem -p123456 lhywzxdev < taxcp-mysql-lhywzxdev20170105.sql

如果远程连接不上mysql的时候，需要查看端口在外网是否能够访问，主要检查防火墙的状态是怎么样的

Mysql主备配置，在做主备之前保证mysql主备的数据是一致的，可以将主库的数据导出来，然后导入到从库里面，这样就能够保证主备数据库内容一致

### Mysql主备配置

主机配置：

service mysqld stop

配置my.cnf:

vim /etc/my.cnf

log-bin=mysql-bin-master #启用二进制日志

server-id = 1 #本机数据库ID 标示

binlog-do-db=sdgsfpgl #可以被从服务器复制的库。二进制需要同步的数据库名

binlog-ignore-db=mysql #不可以被从服务器复制的库

重启mysql:

service mysqld restart

授权： 授权从服务器，让从服务器能够通过slave账号进行复制的操作

mysql> grant replication slave on \*.\* to [slave@192.168.1.64](mailto:slave@192.168.1.64) (从)identified by "123456";

查看状态信息：

mysql> show master status;

+-------------------------+----------+--------------+------------------+

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |

+-------------------------+----------+--------------+------------------+

| mysql-bin-master.000001 | 259 | HA | mysql |

+-------------------------+----------+--------------+------------------+

查看二进制日志：

ls /data/

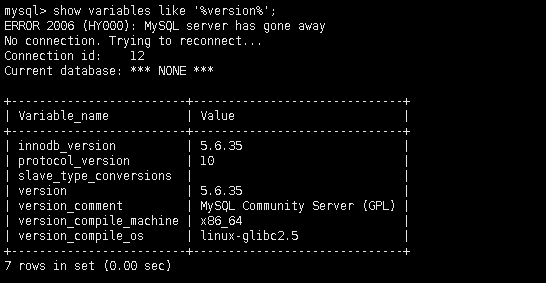


mysql> show binlog events\G; 主要是通过这些信息来进行数据库还原

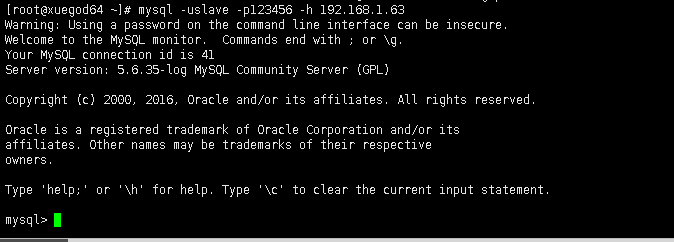


备机配置：

mysql> show variables like '%version%'; --主备机的版本需要一致



--验证定义的slave能否登录

[root@xuegod64 ~]# mysql -uslave -p123456 -h 192.168.1.63 

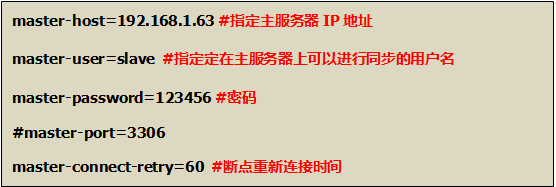
数据导入到备库，保证主备数据库一致性

service mysqld stop

vim /etc/my.cnf

server-id = 2 #从服务器ID号，不要和主ID相同 ，如果设置多个从服务器，每个从服务器必须有一个唯一的server-id值，必须与主服务器的以及其它从服务器的不相同。可以认为server-id值类似于IP地址：这些ID值能唯一识别复制服务器群集中的每个服务器实例。

如果是mysql5.1的配置，则添加如下内容

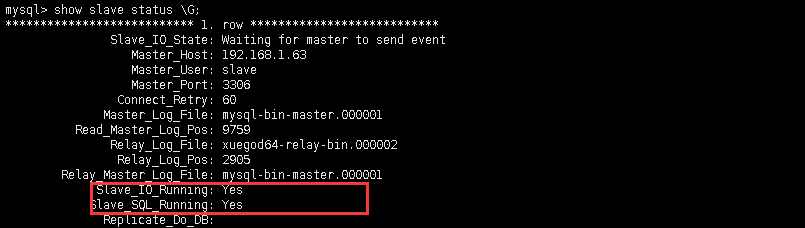


如果不是mysql5.1，则不用添加文本框里面的内容

mysql> change master to master\_host='192.168.1.63',master\_user='slave',master\_password='123456';

mysql> start slave;

mysql> show slave status \G;



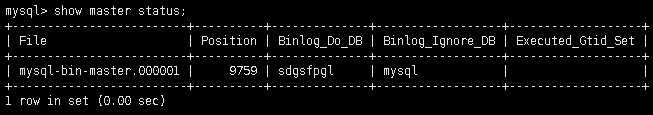
Slave\_IO\_Running ：一个负责与主机的io通信,如果这个故障，检查网络

Slave\_SQL\_Running：负责自己的slave mysql进程，如果这个故障，检查服务

## 排错：

如果遇到主从不同步，看一下主从bin-log的位置，然后再同步。

mysql> show master status; ---主上查看



从服务器执行MySQL命令下：

mysql> slave stop； #先停止slave服务

mysql> change master to master\_log\_file=' mysql-bin-master.000001' ,master\_log\_pos= 9759;

#根据上面主服务器的show master status的结果，进行从服务器的二进制数据库记录回归，达到同步的效果

mysql>slave start; #启动从服务器同步服务

mysql> show slave status\G; #用show slave status\G;看一下从服务器的同步情况

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

如果都是yes，那代表已经在同步

重启从服务器，再查看状态：

停止从服务器slave stop;

开启从服务器slave start;

排错思路：

1、二进制日志没有开启

2、IPTABLES 没有放开端口

3、对应的主机 IP地址写错了

SQL线程出错

1、主从服务器数据库结构不统一

出错后，数据少，可以手动解决创建插入，再更新slave状态。

注：如果主上误删除了。那么从上也就误删除了。 #因此主上要定期做mysqldump备份。

# Mysql 主备搭建

## mysql安装

[root@localhost software]# rpm -ivh mysql-community-common-5.7.12-1.el6.x86\_64.rpm

warning: mysql-community-common-5.7.12-1.el6.x86\_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY

Preparing... ########################################### [100%]

1:mysql-community-common ########################################### [100%]

[root@localhost software]#

[root@localhost software]#

[root@localhost software]# rpm -ivh mysql-community-libs-5.7.12-1.el6.x86\_64.rpm

warning: mysql-community-libs-5.7.12-1.el6.x86\_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY

Preparing... ########################################### [100%]

1:mysql-community-libs ########################################### [100%]

[root@localhost software]# rpm -ivh mysql-community-devel-5.7.12-1.el6.x86\_64.rpm

warning: mysql-community-devel-5.7.12-1.el6.x86\_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY

Preparing... ########################################### [100%]

1:mysql-community-devel ########################################### [100%]

[root@localhost software]# rpm -ivh mysql-community-client-5.7.12-1.el6.x86\_64.rpm

warning: mysql-community-client-5.7.12-1.el6.x86\_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY

Preparing... ########################################### [100%]

1:mysql-community-client ########################################### [100%]

[root@localhost software]# rpm -ivh mysql-community-server-5.7.12-1.el6.x86\_64.rpm

warning: mysql-community-server-5.7.12-1.el6.x86\_64.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY

Preparing... ########################################### [100%]

1:mysql-community-server ########################################### [100%]

## 添加密码

**vi /etc/my.conf 最后一行添加 skip-grant-tables 保存后启动**

**service mysqld start**

**或者service mysqld start --skip-grant-tables**

mysql -u root -p

输入密码直接敲击回车跳过

mysql> use mysql

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> update user set password\_expired='N' where user='root';

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> update user set authentication\_string=password('123456') where user='root';

Query OK, 1 row affected, 1 warning (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 1

mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)

然后注释掉/etc/my.conf 最后一行添加 skip-grant-tables

service mysqld restart

mysql -u root -p

输入密码123456

## 允许远程连接

mysql> update user set host='%' where user='root';

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> flush privileges;

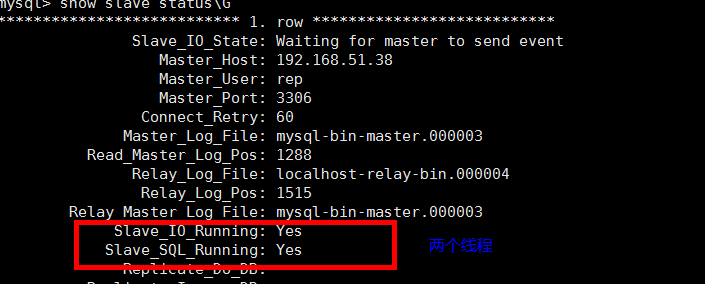
Query OK, 0 rows affected (0.00 sec)

## 主从库搭建

**科普知识**

Show master status\G;

Show slave status\G; //查看主从库配置状态



show binlog events; //查看主库开启日志状态

change master to master\_host='192.168.51.38',master\_user='rep' ,master\_password='123'

完整版如下：

CHANGE MASTER TO

MASTER\_HOST='192.168.51.38',

MASTER\_PORT=3306,

MASTER\_USER='rep',

MASTER\_PASSWORD='123456',

MASTER\_LOG\_FILE='mysql-bin-master.000001',

MASTER\_LOG\_POS=4296;

密码简单化

set global validate\_password\_policy=0;

set global validate\_password\_length=0;

#授权从服务器，让重服务器可以访问

create user rep identified by '123456';

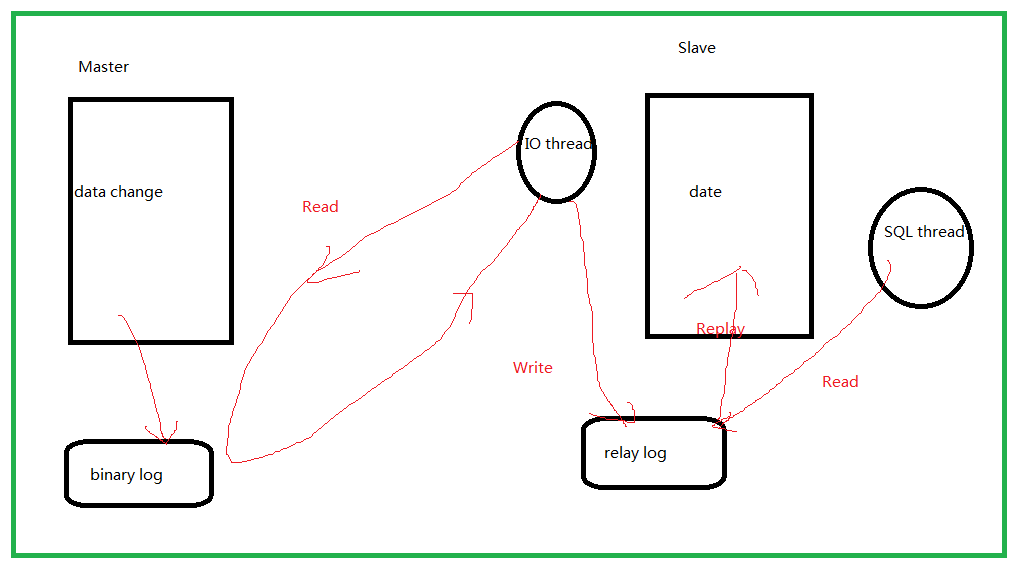
grant replication slave ,reload,super on \*.\* to rep@192.168.51.37 identified by "rep123456" ;

Mysql主备复制原理：

**A) Master将改变记录写到binary log 中（叫做二进制日志事件 binary log events,可以通过show binlog events 查看）**

**B) Slave讲Master的binary log events copy 到中继日志（relay log）**

**C) Slave 重做中继日志中的事件**



配置

主库

[mysqld]

log-bin=mysql-bin

server-id=1

从库

[mysqld]

log-bin=mysql-bin

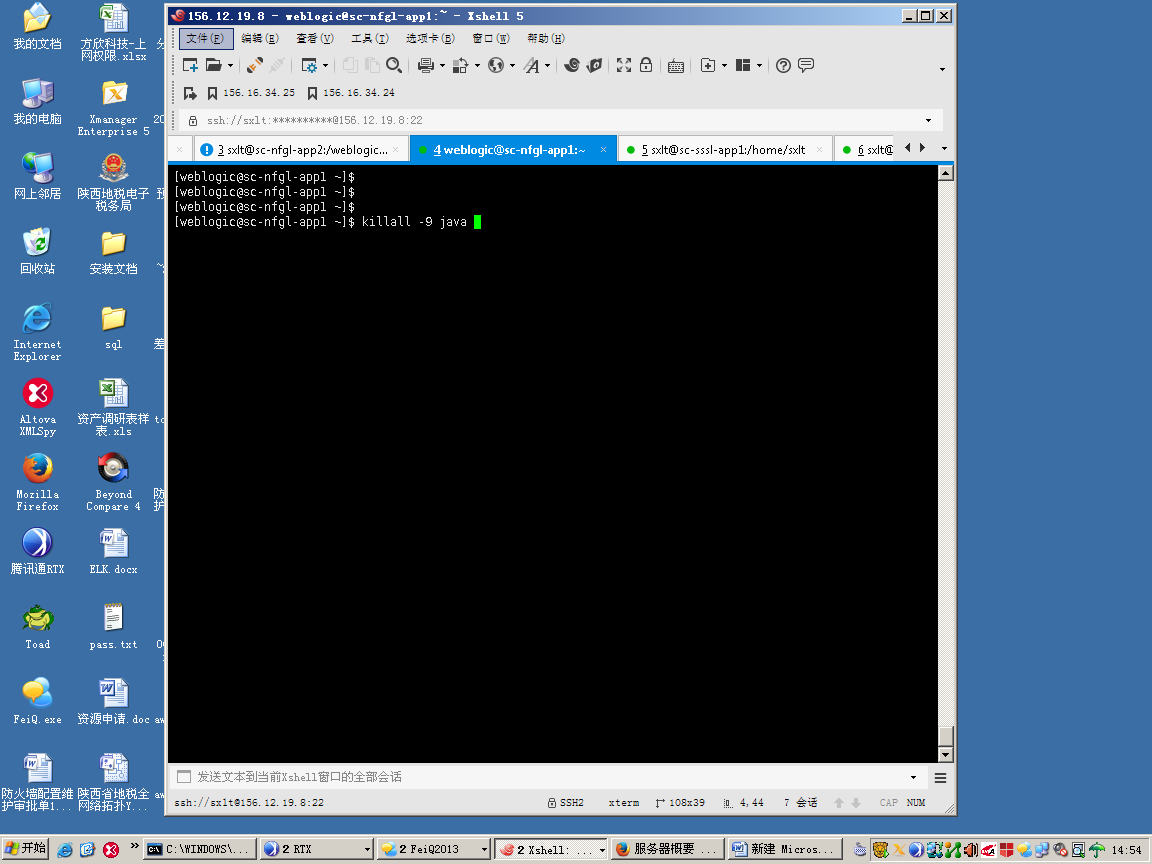
server-id=2

relay\_log=mysql-relay-bin

log\_slave\_updates=1

reda\_only=1

|  |  |
| --- | --- |
| 156.12.19.8（nfgl管理节点） | /weblogic/user\_projects/domains/ |
| 156.12.19.9 | /weblogic/user\_projects/domains/ |
| 156.12.19.10(dzda管理节点) | /weblogic/user\_projects/domains/ |
| 156.12.19.11（ydwx管理节点） | /weblogic/user\_projects/domains/ |
| 156.12.19.12(sssl管理节点) | /weblogic/user\_projects/domains/ |
| 156.12.19.13 | /weblogic/user\_projects/domains/ |
| 156.12.19.14 | /weblogic/user\_projects/domains/ |
| 156.12.19.23(wtdz管理节点) | /weblogic/user\_projects/domains/ |
| 156.12.19.24 | /weblogic/user\_projects/domains/ |
| 192.168.51.2（ydwx管理节点） | /weblogic/user\_projects/domains/ |
| 192.168.51.3（omni管理节点） | /weblogic/user\_projects/domains/ |
| 192.168.51.4（portal管理节点） | /weblogic/user\_projects/domains/ |
| 192.168.51.5（wwjms管理节点） | /weblogic/user\_projects/domains/ |
| 192.168.51.6(wsbs管理节点) | /weblogic/user\_projects/domains/ |
| 192.168.51.7 | /weblogic/user\_projects/domains/ |



1. 外网应用服务器停止步骤：

用weblogic用户登陆，然后运行

killall -9 java

1. 内网应用服务器停止步骤：

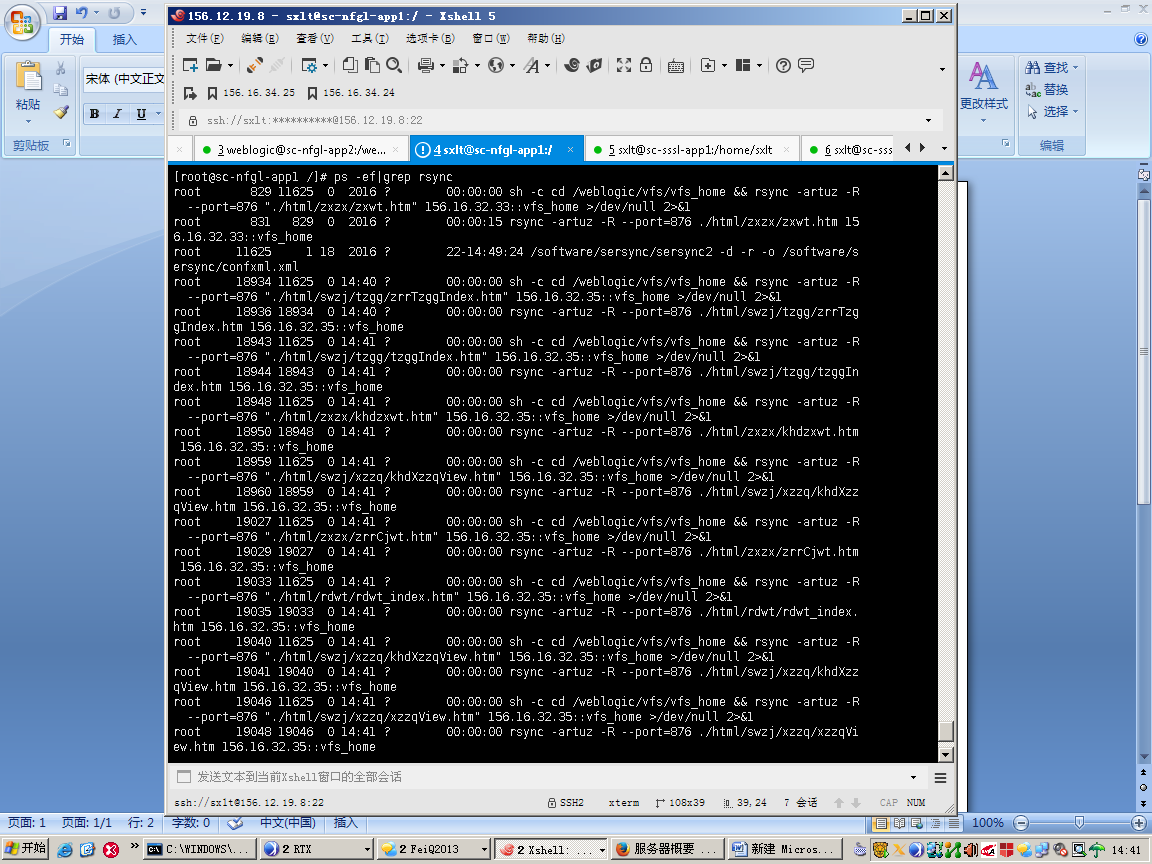
用weblogic用户登陆，然后运行

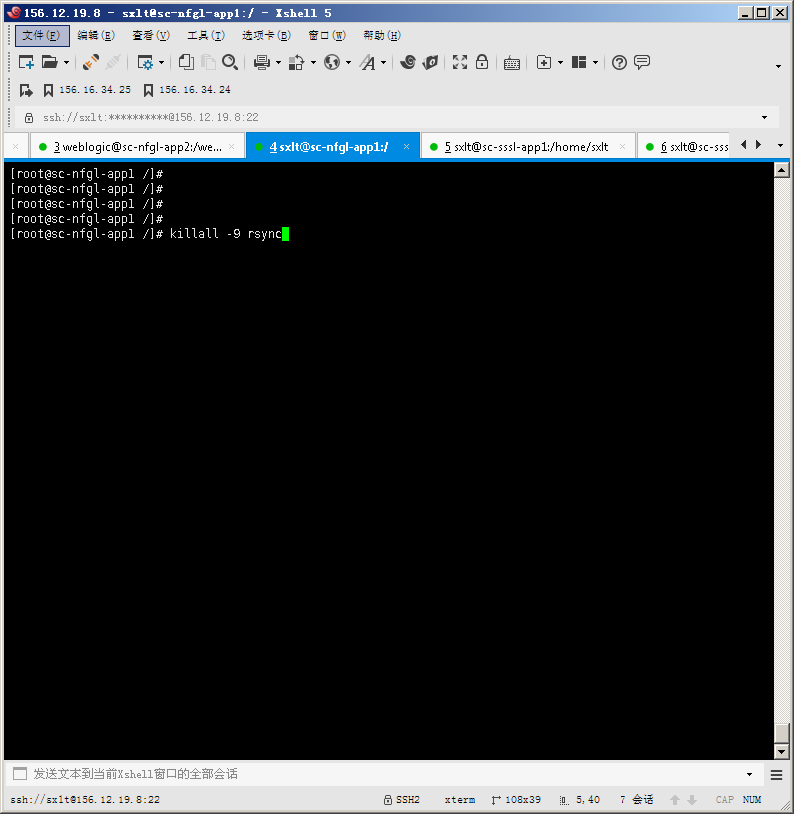
killall -9 java

### 3.停止rsync服务

用root用户登陆156.12.19.8这台服务器，然后执行以下命令

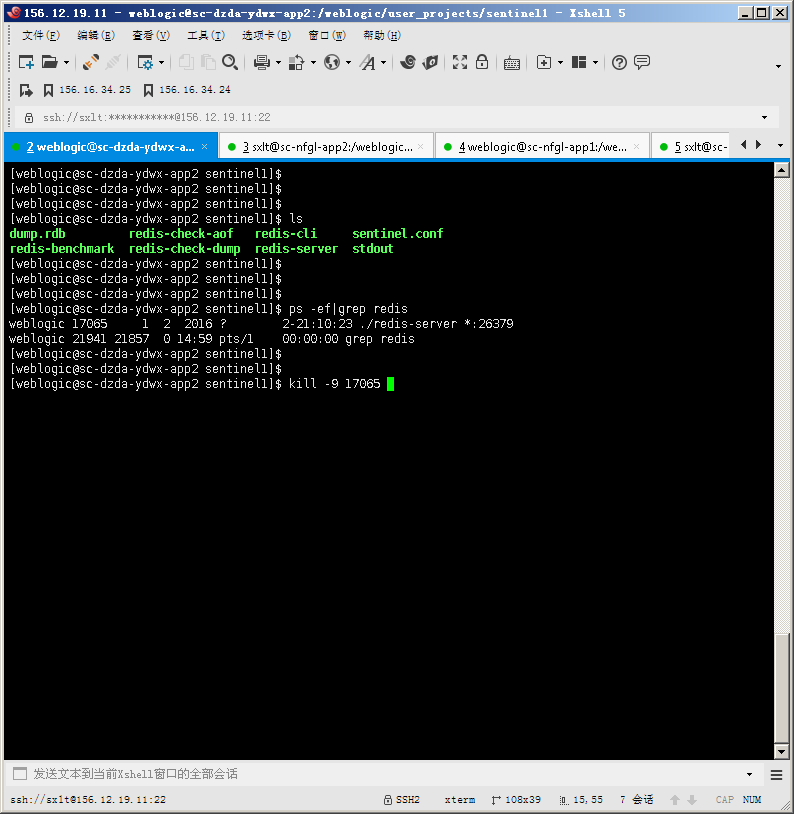
killall -9 rsync





### 4.停止redis服务

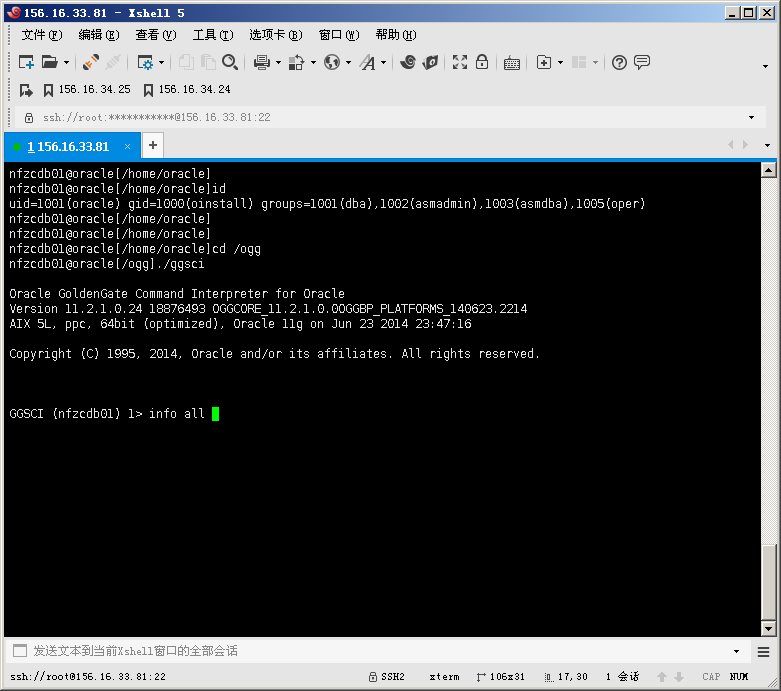
用weblogic用户登陆156.12.19.11和192.168.51.2这2台服务器，然后执行以下命令，找出redis的进程，然后将其全部kill掉

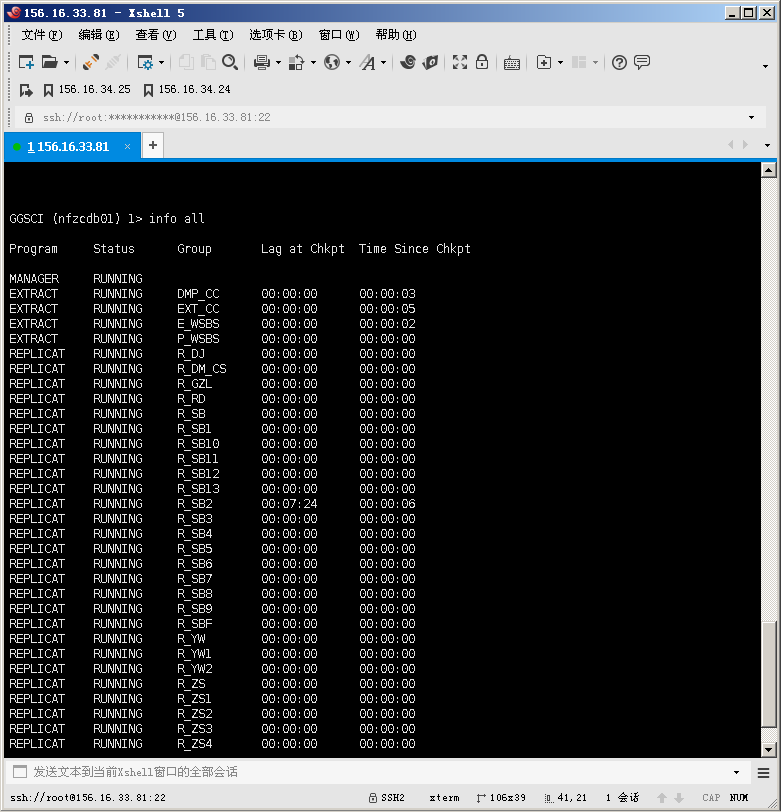


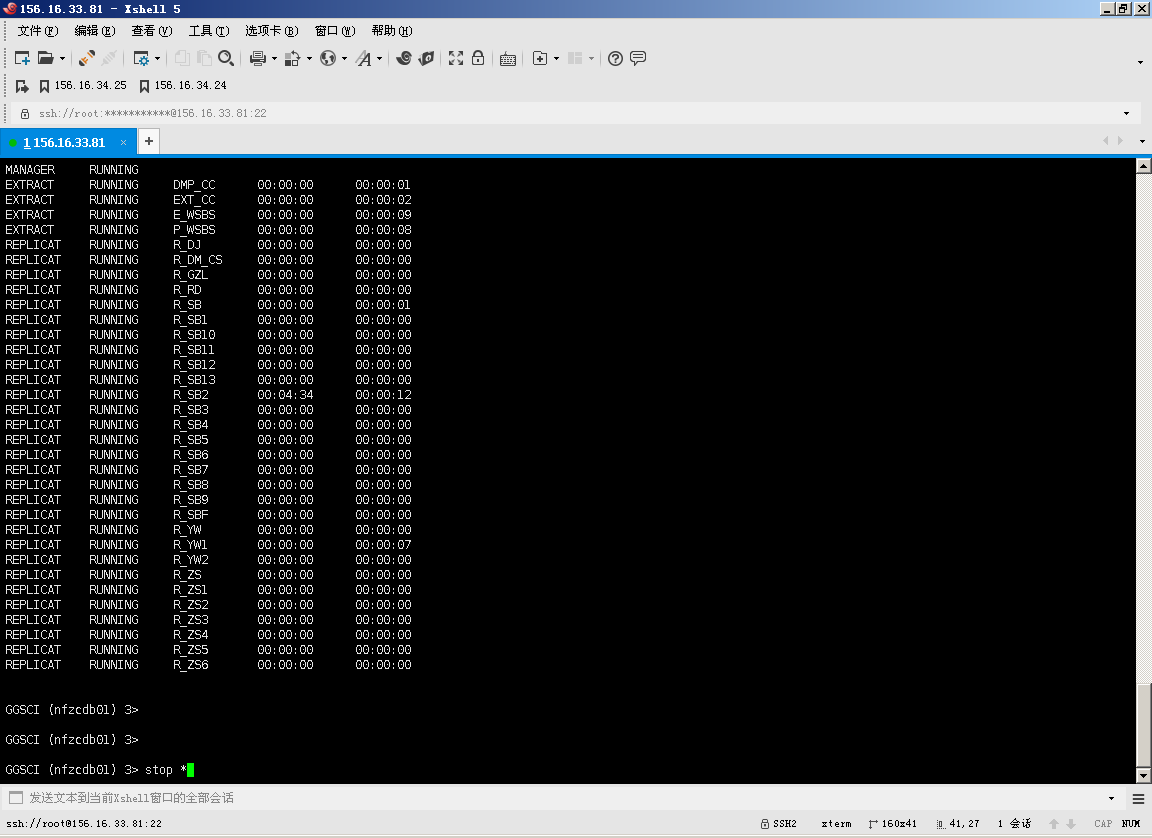
### 5.停止ogg

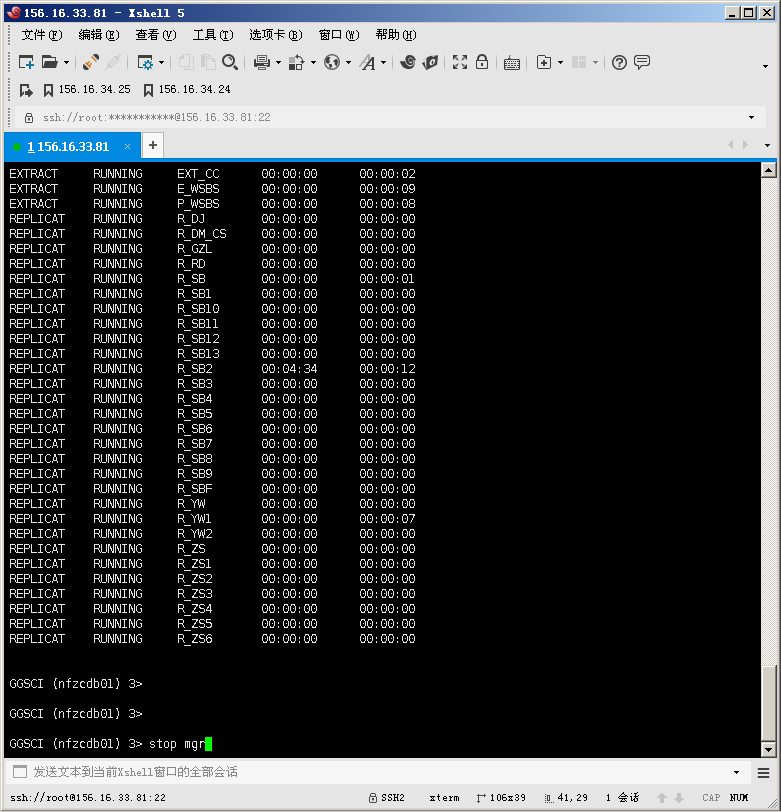
（1）应先停止源端，然后再停止目标端

用oracle用户登陆156.16.33.81（源端）这台服务器，然后进入/ogg目录，执行以下命令**：**



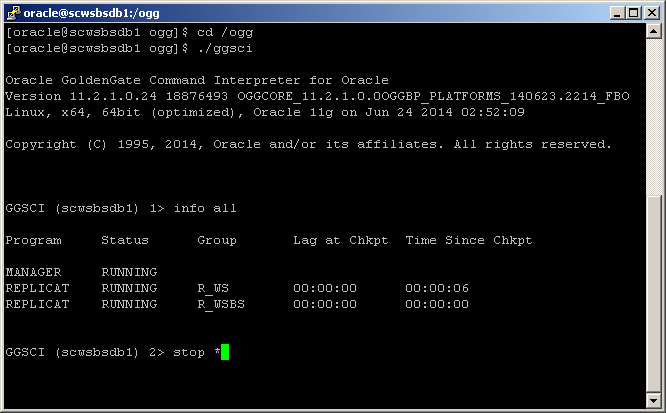


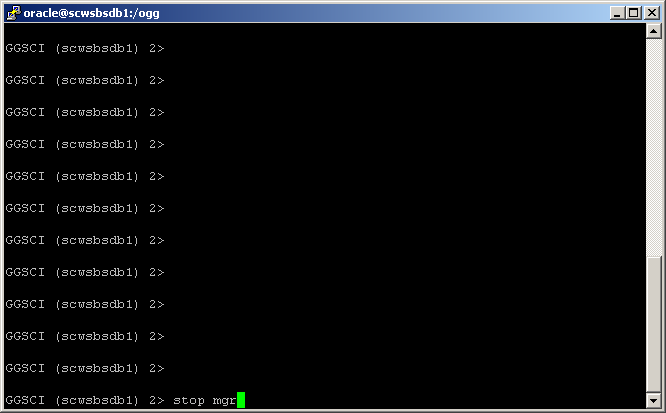




（2）停止目标端

用oracle用户登陆192.168.52.2（目标端）这台服务器，然后进入/ogg目录，执行以下命令：

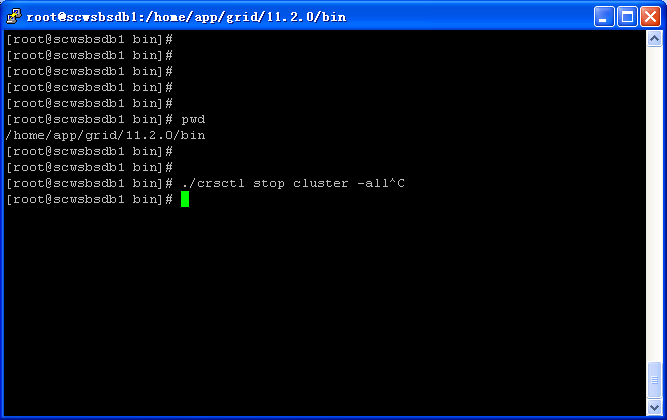




### 6.停止数据库

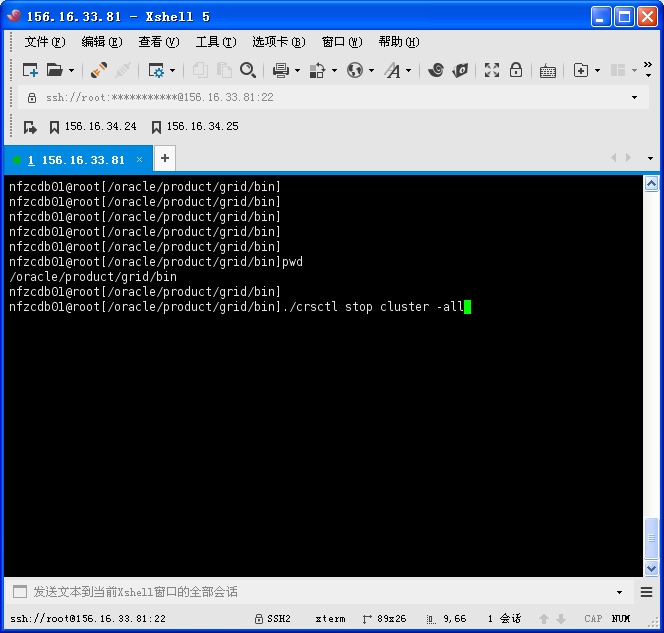
用root用户登陆192.168.52.2/4，然后运行以下命令（1个节点操作）：

crsctl stop cluster -all



用root用户登陆156.16.33.81/82，然后运行以下命令：

crsctl stop cluster -all



**自此电子税务局系统停止完毕。**

## 四、电子税务局的启动顺序



### 1.启动数据库

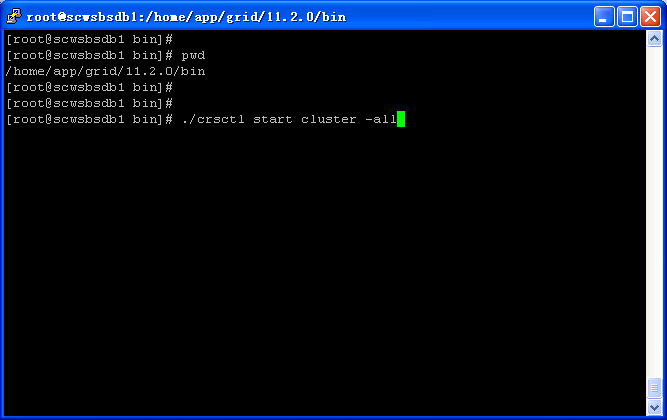
用root用户登陆192.168.52.2/4，然后运行以下命令：

cd /home/app/grid/11.2.0/bin

./crsctl start cluster –all

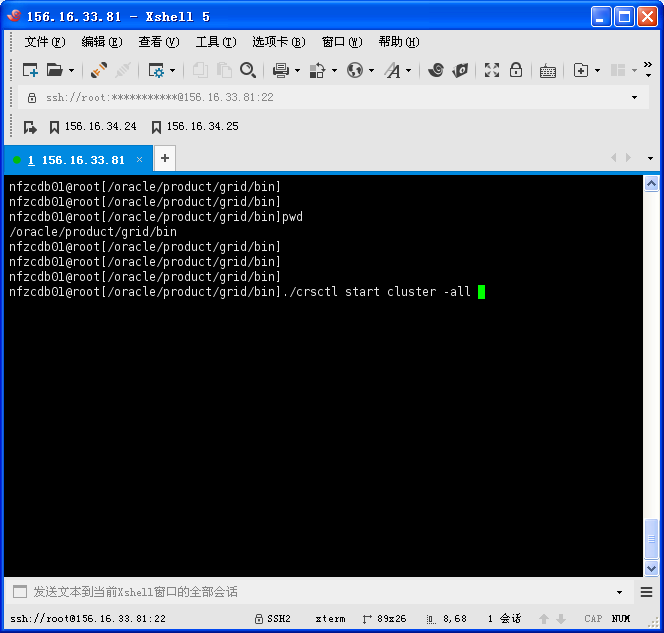
crsctl check cluster

crs\_stat -t -v



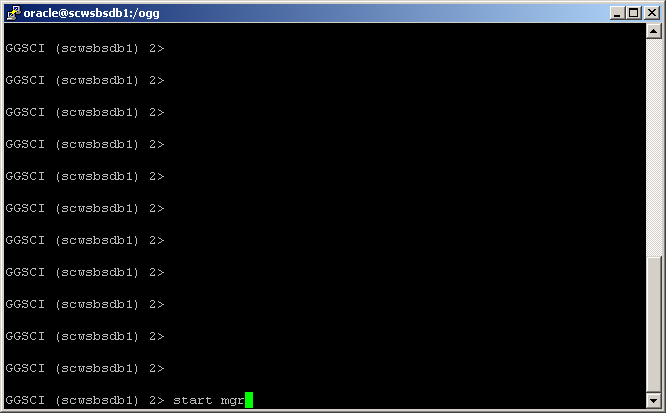
用root用户登陆156.16.33.81/82，然后运行以下命令：

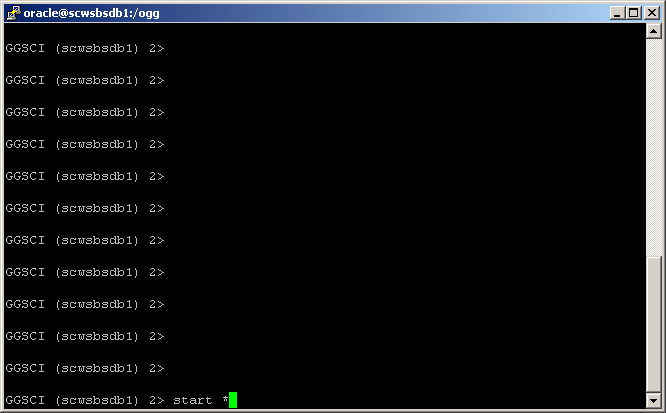
crsctl start cluster –all



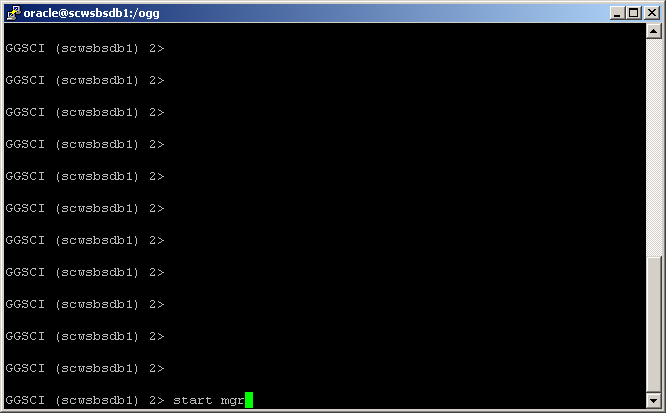
### 2.启动ogg

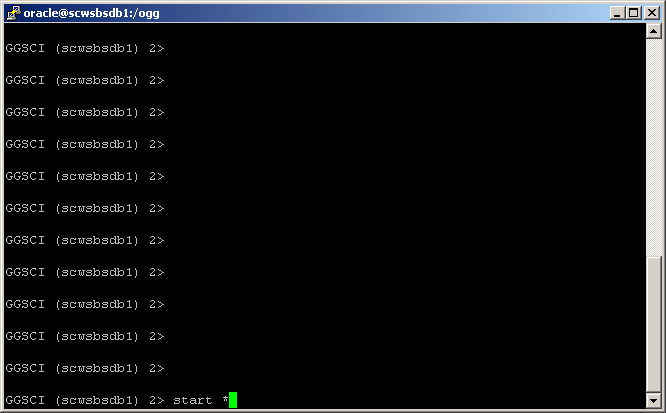
1. 先启动目标的（192.168.52.2）





（2）启动源端（156.16.33.81）



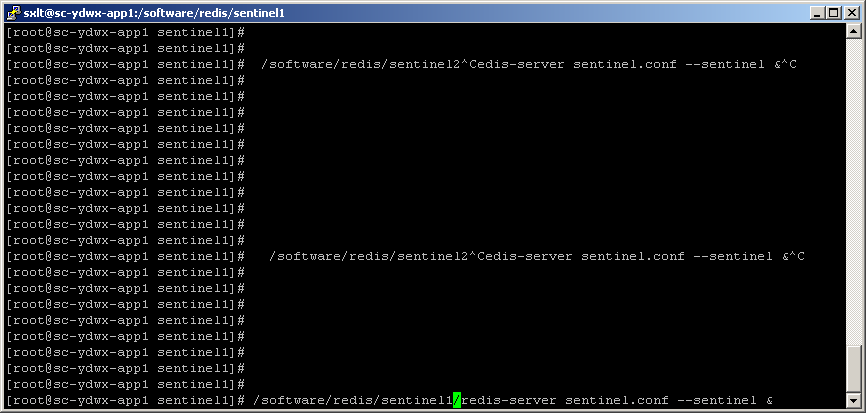


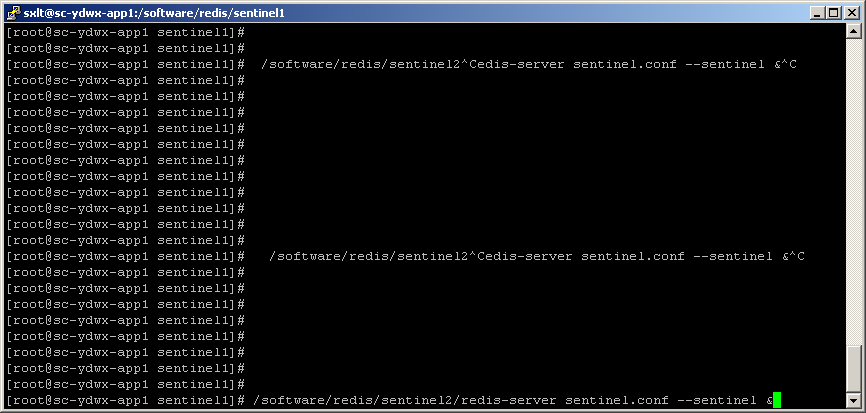
### 3.启动redis服务

（1）用root用户登陆192.168.51.2并执行以下命令，

先启动master，然后在启动哨兵



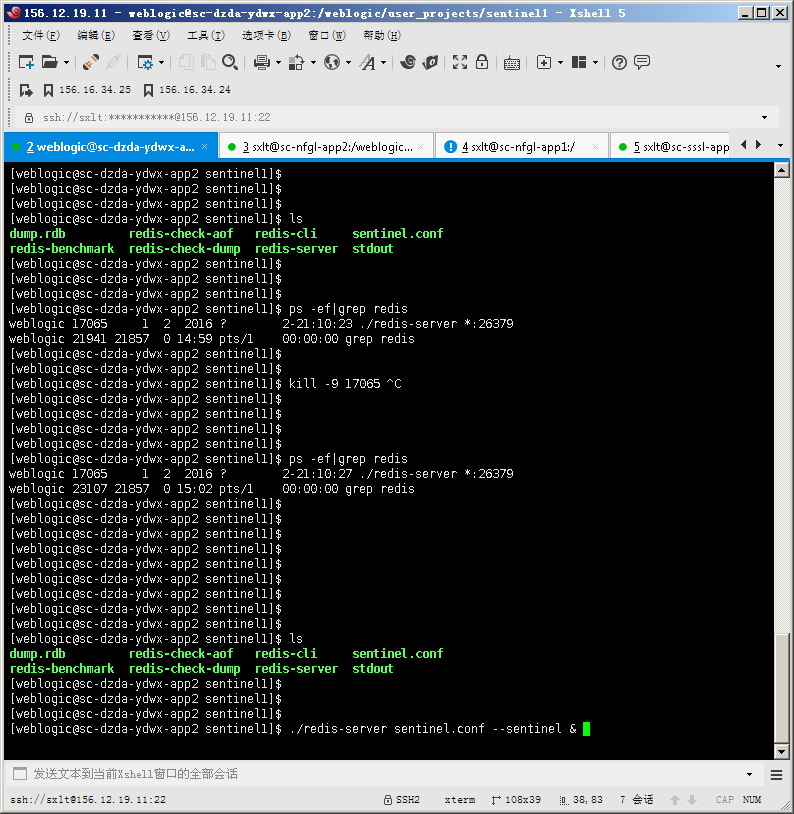




1. 用root用户登陆156.12.19.11并执行以下命令

/weblogic/user\_projects/sentinel1

/weblogic/user\_projects/sentinel2

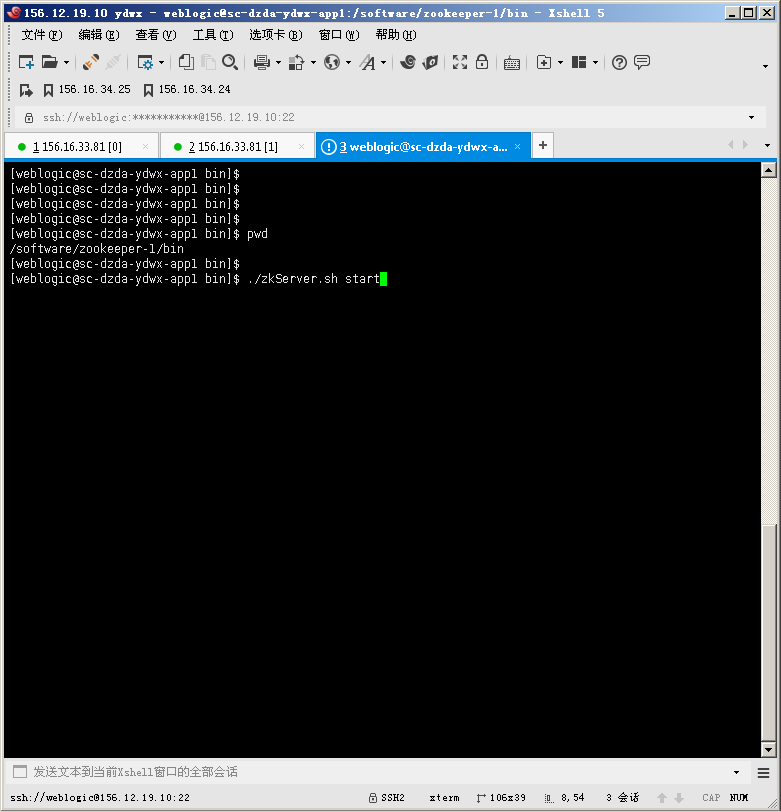


### 4.启动zookeeper服务

用weblogic用户登陆156.12.19.10和156.12.19.11这两台服务其，然后启动zookeeper服务。按顺序将其全部启动。总共有6个zookeeper服务。(依次)

./zkServer.sh start

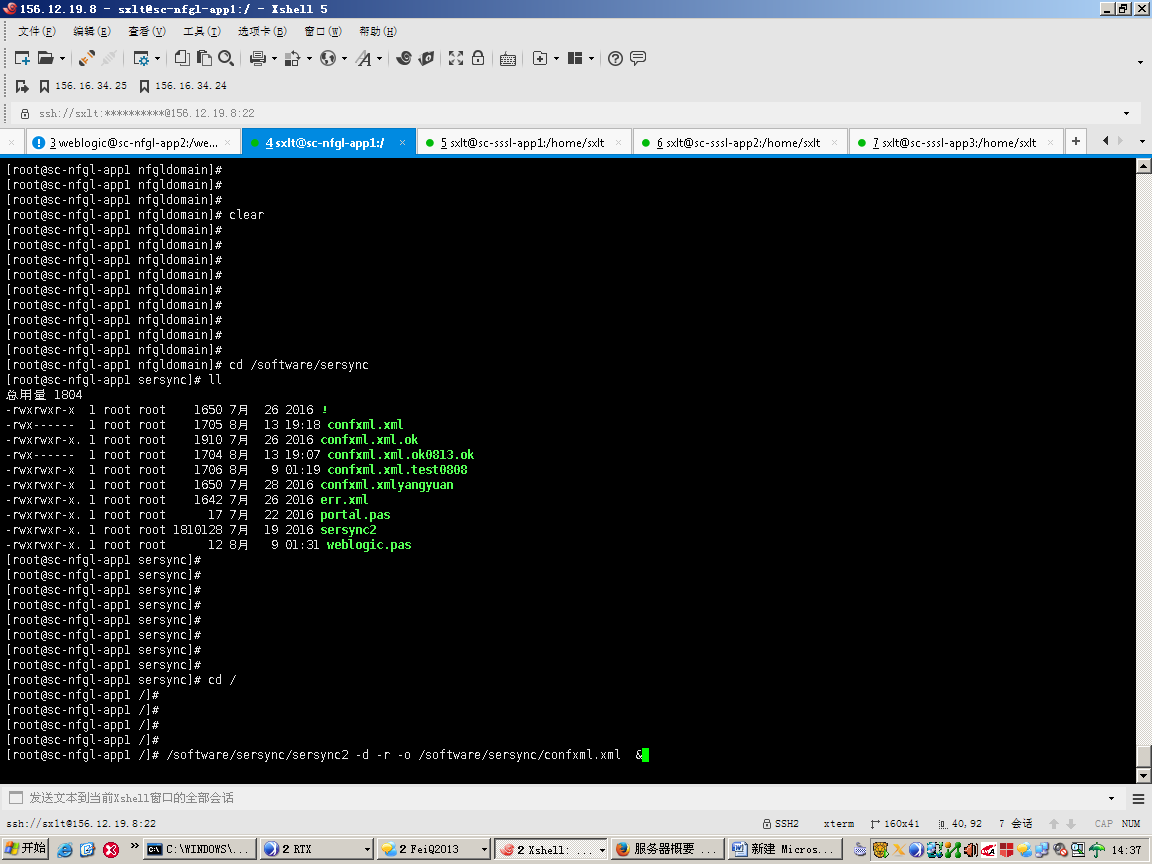
|  |  |
| --- | --- |
| **ip** | **zookeeper安装路径** |
| 156.12.19.10 | /software/zookeeper-1/bin |
| /software/zookeeper-2/bin |
| /software/zookeeper-3/bin |
| 156.12.19.11 | /software/zookeeper-1/bin |
| /software/zookeeper-2/bin |
| /software/zookeeper-3/bin |



### 5.启动rsync服务

用root用户登陆156.12.19.8这台服务器，然后执行以下命令

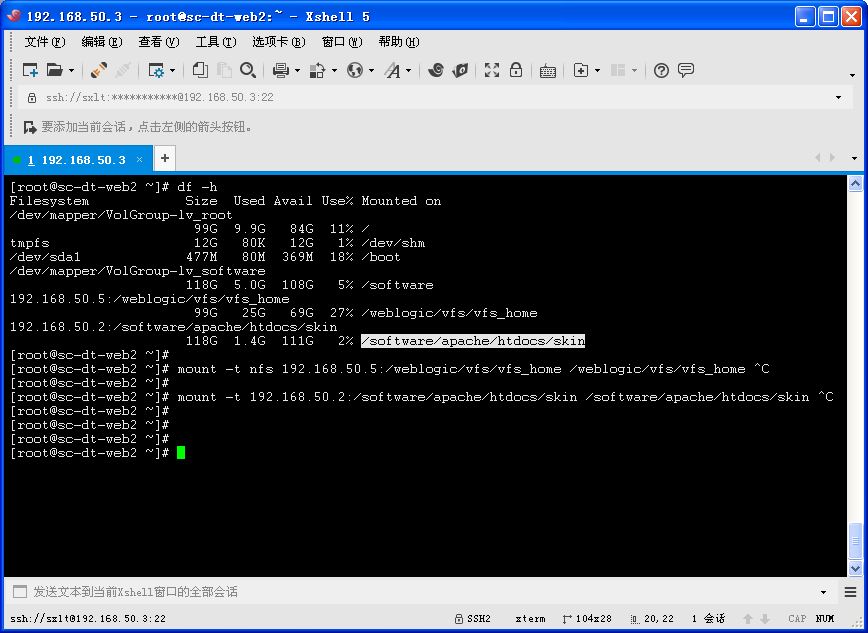
/software/sersync/sersync2 -d -r -o /software/sersync/confxml.xml &

6.mount vfs\_home、skin共享目录（cat /etc/fstab查看共享目录路径）

**6.1**首先挂载外网apache服务器（192.168.50.2/3/4/5）的vfs\_home、skin目录,skin的源端在：192.168.50.2这台服务器上，vfs\_home的源在：192.168.50.5。

故此192.168.50.2上只需挂载vfs\_home目录, 192.168.50.5只需要挂载skin目录。而192.168.50.3/4需要挂载vfs\_home和skin这两个目录。（内网skin和vfs\_home共享目录在 156.12.19.8 weblogic/vfs/vfs\_home /software/apache/htdocs/skin

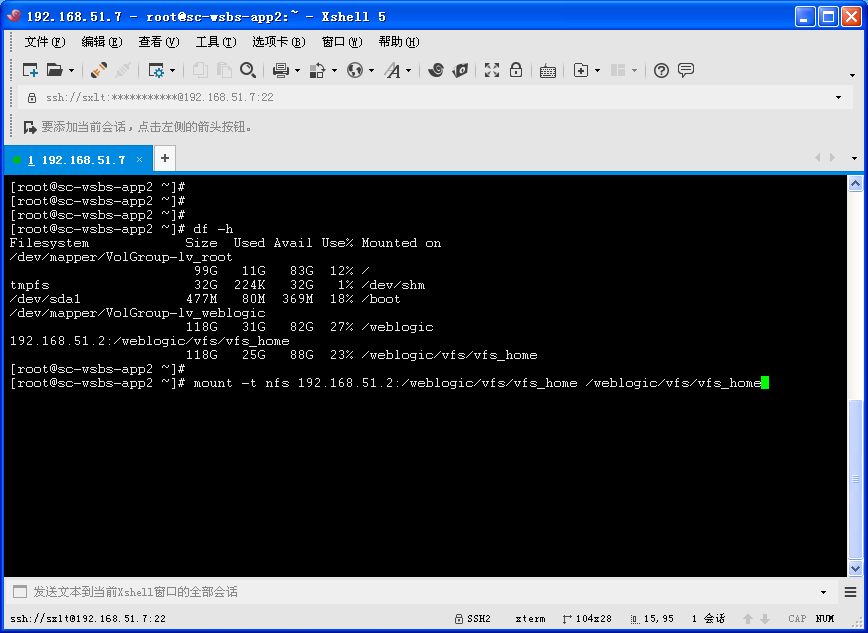
）



mount -t nfs 192.168.50.5:/weblogic/vfs/vfs\_home /weblogic/vfs/vfs\_home

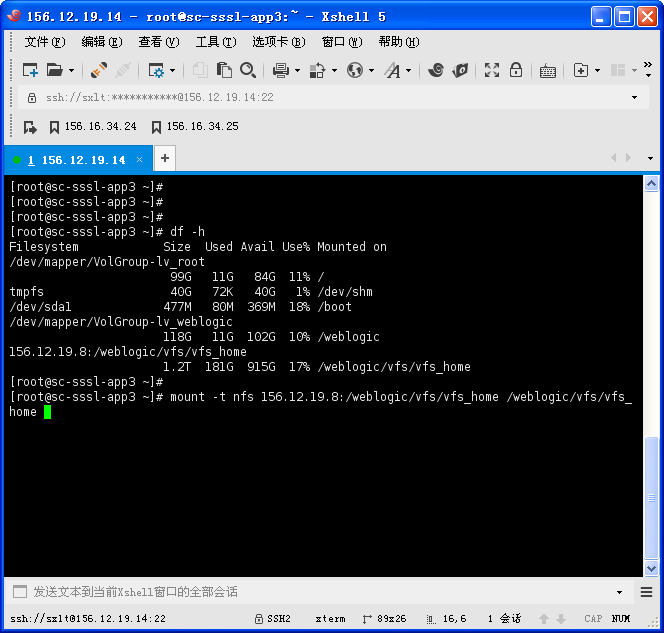
mount -t nfs 192.168.50.2:/software/apache/htdocs/skin /software/apache/htdocs/skin

**6.2**挂载外网应用服务器（192.168.51.3/4/5/6/7）的vfs\_home目录，vfs\_home源端在192.168.51.2这台服务器上。

****

mount -t nfs 192.168.51.2:/weblogic/vfs/vfs\_home /weblogic/vfs/vfs\_home

**6.3**挂载内网应用服务器（156.12.19.8/9/10/11/12/13/14）的vfs\_home目录，vfs\_home源端在156.12.19.8这台服务器上。

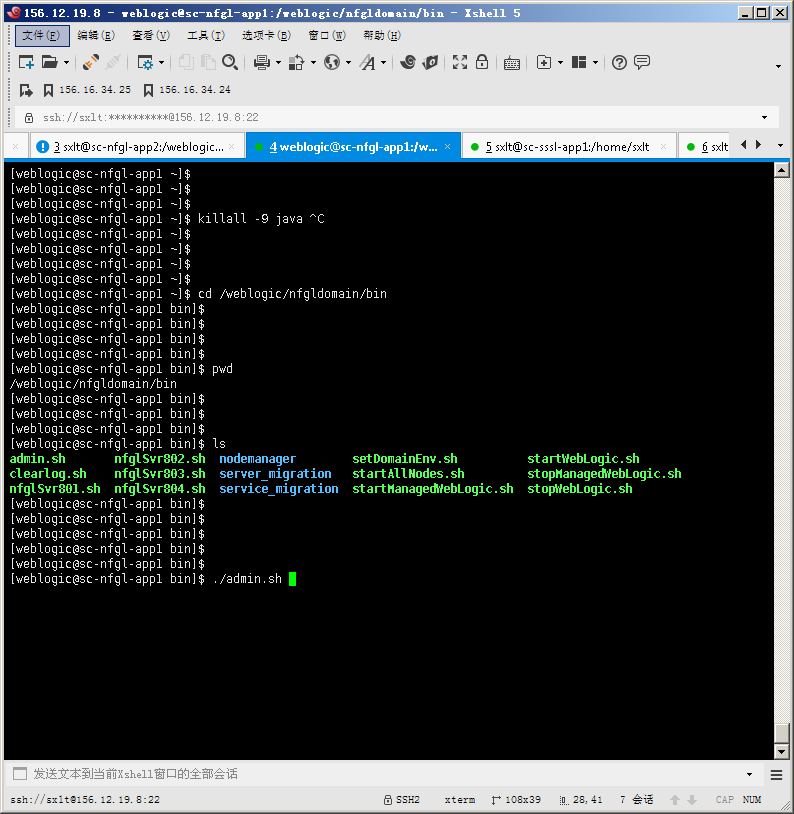
****

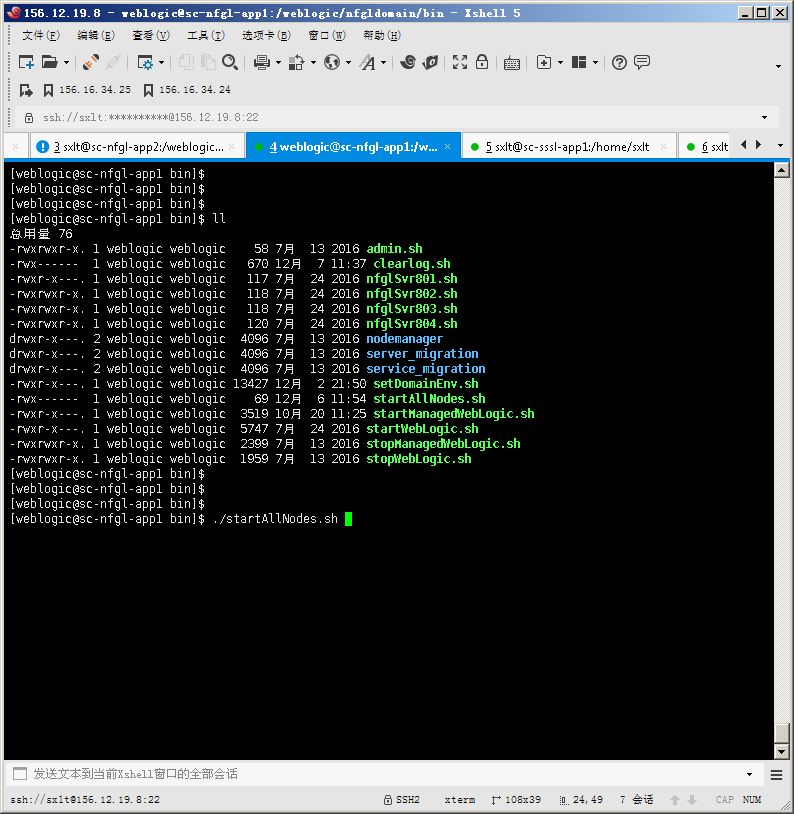
mount -t nfs 156.12.19.8:/weblogic/vfs/vfs\_home /weblogic/vfs/vfs\_home

### 7.启动应用服务器(先后顺序)

|  |  |
| --- | --- |
| **weblogic应用服务器ip地址** | **域安装路径** |
| 156.12.19.8（nfgl管理节点） | /weblogic/user\_projects/domains/ |
| 156.12.19.9 | /weblogic/user\_projects/domains/ |
| 156.12.19.10(dzda管理节点) | /weblogic/user\_projects/domains/ |
| 156.12.19.11（ydwx管理节点） | /weblogic/user\_projects/domains/ |
| 156.12.19.12(sssl管理节点) | /weblogic/user\_projects/domains/ |
| 156.12.19.13 | /weblogic/user\_projects/domains/ |
| 156.12.19.14 | /weblogic/user\_projects/domains/ |
| 156.12.19.23（wtdz管理节点） | /weblogic/user\_projects/domains/ |
| 156.12.19.24 | /weblogic/user\_projects/domains/ |
| 192.168.51.2（ydwx管理节点） | /weblogic/user\_projects/domains/ |
| 192.168.51.3（omni管理节点） | /weblogic/user\_projects/domains/ |
| 192.168.51.4（portal管理节点） | /weblogic/user\_projects/domains/ |
| 192.168.51.5（wwjms管理节点） | /weblogic/user\_projects/domains/ |
| 192.168.51.6(wsbs管理节点) | /weblogic/user\_projects/domains/ |
| 192.168.51.7 | /weblogic/user\_projects/domains/ |

1. 首先我们需要启动JMS应用服务器（192.168.51.4/5），先启动管理节点然后在启动被管理节点。（JMS服务器如果重启 其他应用服务器必须全部重启）

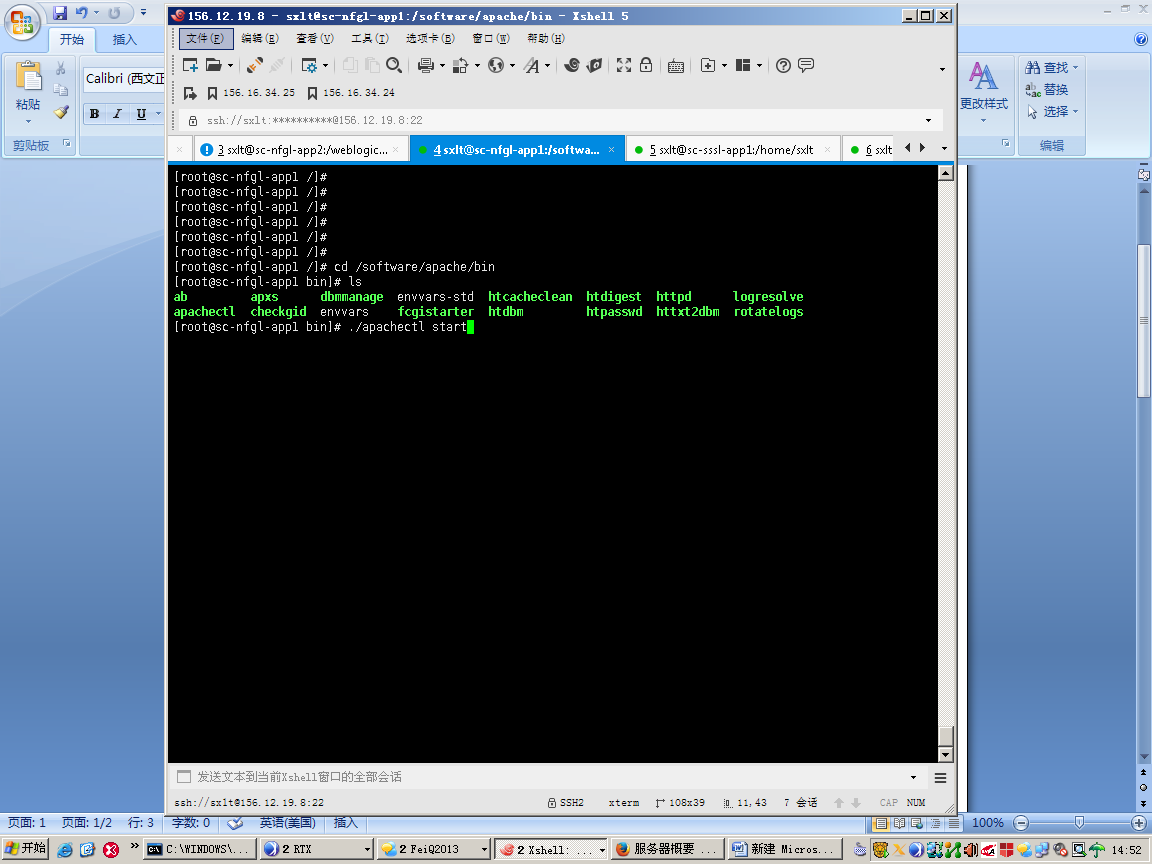




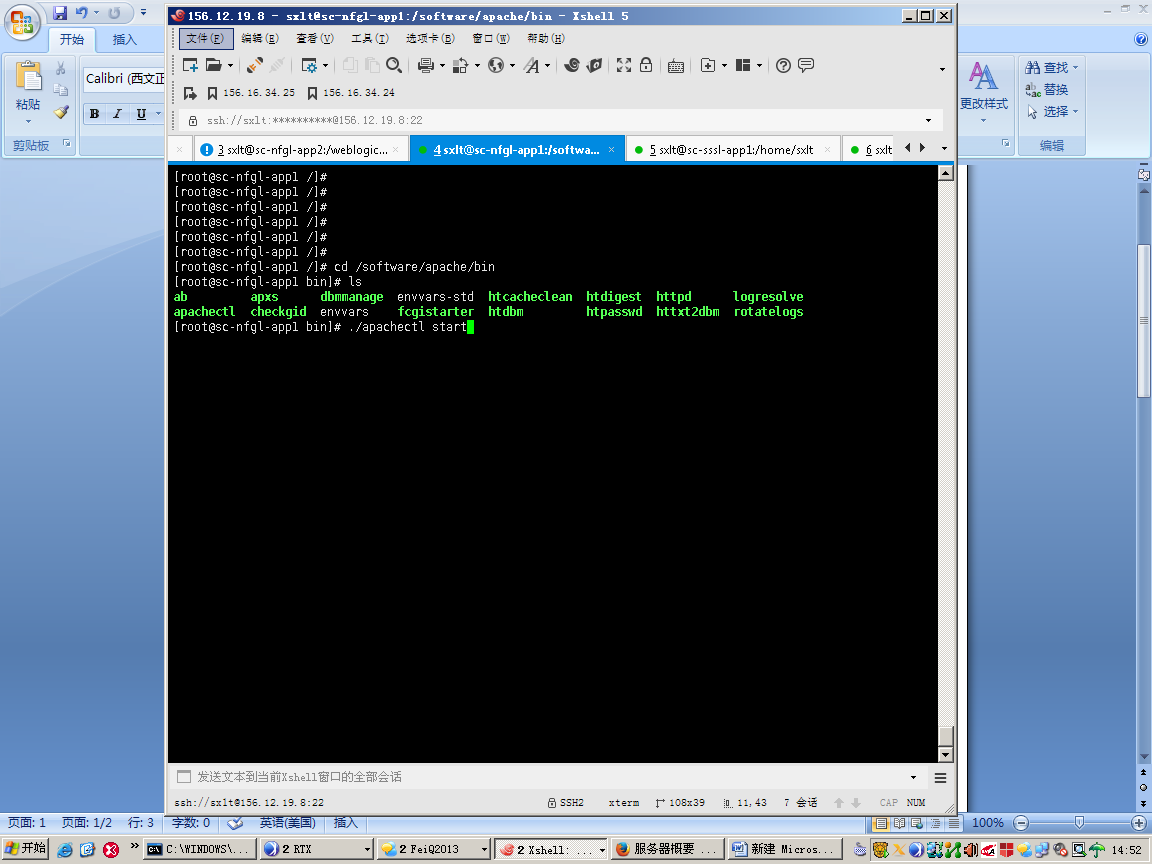
1. 启动nfgl应用服务器（156.12.19.8/9）
2. 启动sssl应用服务器（156.12.19.12/13/14）
3. 启动dzda应用服务器（156.12.19.10/11）
4. 启动移动微信内网应用服务器（156.12.19.10/11）
5. 启动移动微信外网应用服务器（192.168.51.2/3）
6. 启动wsbs应用服务器（192.168.51.6/7）
7. 启动portal应用服务器（192.168.51.4/5）
8. 启动omni应用服务器（192.168.51.2/3）
9. 启动wtdz应用服务器（156.12.19.23/24）

### 8.启动apache服务器（依次）

（1）首先启动内网apache服务器（156.12.19.8/9）



（2）然后启动外网apache服务器（192.168.50.2/3/4/5）



**自此电子税务局系统启动完毕。**

# Docker

## Docker的八种用途

　　Docker 提供轻量的虚拟化，你能够从Docker获得一个额外抽象层，你能够在单台机器上运行多个Docker微容器，而每个微容器里都有一个微服务或独立应用，例如你可以将Tomcat运行在一个Docker，而MySQL运行在另外一个Docker，两者可以运行在同一个服务器，或多个服务器上。未来可能每个应用都要Docker化。

容器的启动和关系是非常快速的。Docker目前能够有以下八种用途：

## 1 简化配置

这是Docker初始目的，虚拟机VM最大的好处是基于你的应用配置能够无缝运行在任何平台上。Docker提供同样类似VM的能力，但是没有任何副作用，它能让你将环境和配置放入代码然后部署，同样的Docker配置能够在各种环境中使用，这实际是将应用环境和底层环境实现了解耦。

## 2 代码管道化管理

能够对代码以流式pipeline管道化进行管理，从开发者的机器到生产环境机器这个流程中都能有效管理。因为在这个流程中会有各种不同的环境，每个都可能有微小的区别，Docker提供了跨越这些异构环境以一致性的微环境，从开发到部署实现流畅发布。

## 3开发人员的生产化

在一个开发环境，我们希望我们的开发环境能更加接近于生产环境，我们会让每个服务运行在自己的VM中，这样能模拟生产环境，比如有时我们并不总是需要跨越网络连接，这样我们可以将多个Docker装载一系列服务运行在单机上最大程度模拟生产分布式部署的环境。

## 4应用隔离

有很多理由你需要在一台机器上运行多个应用，这就需要将原来铁板一块monolithic的应用切分为很多微服务。实现应用之间的解耦，将多个应用服务部署在多个Docker中能轻松达到这个目的。

## 5服务合并

使用Docker也能合并多个服务以降低费用，不多的操作系统内存占用，跨实例共享多个空闲的内存，这些技术Docker能以更加紧密资源提供更有效的服务合并。

## 6多租户

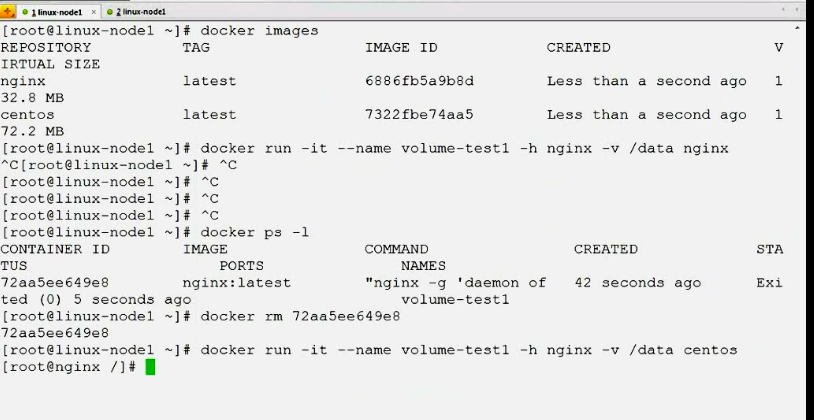
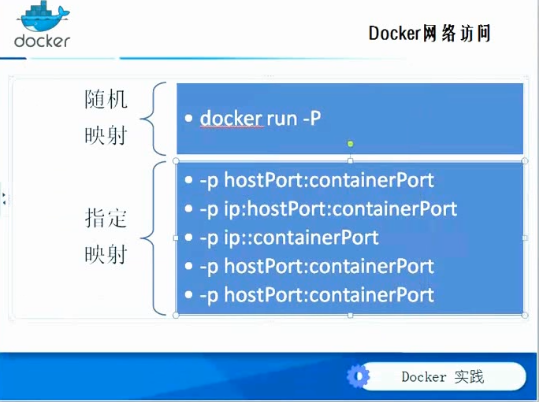
Docker能够作为云计算的多租户容器，使用Docker能容易为每个租户创建运行应该多个实例，这得益其灵活的快速环境以及有效diff命令。

## 7快速部署

Docker通过创建流程的容器，不必重新启动操作系统，几秒内能关闭，你可以在数据中心创建或销毁资源，不用担心额外消耗。典型的数据中心利用率是30%，通过更积极的资源分配，以低成本方式对一个新的实例实现一个更聚合的资源分配，我们很容易超过这个利用率，大大提高数据中心的利用效率。

## 8Docker+微服务覆盖传统的软件架构

## Docuer端口映射



数据卷使用



## Test

# Readme

## UE快捷键

**%[ ^t]++^p 空行**

**^p 换行**

**% 行首替换**

**$ 行末尾替换**

**[^t]+$ 行末尾空格**

**Ctrl + E 删除当前行**

**Ctrl + W 自动换行**

**F7 插入当前日期 2018-07-12 15:08:10**

**Ctrl + F3 查找前一个配置**

**F3查找后一个匹配**

## Tomcat

**查看端口 和访问路径**

/software/tomcat-confcenter/conf/server.xml

<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">

http://192.168.51.32:6789/confCenter/conf?action=reload

## 配置中心连接数据库

/software/tomcat-confcenter/webapps/confCenter/WEB-INF/classes/properties

## 核心发布DDL执行

内网156.16.33.81

数据库156.16.33.81 :1521/nfzcdb

Passwd:Foresee\_4U2016

SJSC\_HXCX

外网192.168.51.24

数据库192.168.52.2:1521/snltwsbs

Passwd:Foresee\_4U2016

SJSC\_HXCX

Summery:执行hxcx脚本先执行sql,在执行索引

## 交易处理时间查询

*--交易用时五秒以上的占比*

select

round(sum(case when r.jyclsj>5000 then 1 else 0 end )\*1.0/count(\*)\*100,2) ||'%' p

from

nf\_sssl.nf\_sssl\_jyrz\_20180814 r;

*--五秒以上*

select r.sid, r.jyclsj, r.\*

from nf\_sssl.nf\_sssl\_jyrz\_20180814 r

where r.jyclsj > 5000

and r.sid in ('Nfzh.SWZJ.HXZG.DJ.CXNSRXXWBMBGSWDJXXDJXHNSRMC',

'ETax.SB.enterSbxx.SbcxService',

'ETax.SB.sbInit2.910610090023sxds')

*--总交易*

select count(\*) from nf\_sssl.nf\_sssl\_jyrz\_20180112;

*--交易扩展表*

select count(\*) from nf\_sssl.nf\_sssl\_jyrz\_kz\_20180112;

select \* from nf\_sssl.nf\_sssl\_jyrz\_kz\_20180112;

*---三方接入表空间*

select \* from nf\_sfjr\_jyrz ff order by ff.lrrq asc

select \* from nf\_sfjr\_jyrz\_mx

## Oralce2mysql

Mysql :str\_to\_date('01-10-1994', '%d-%m-%Y')

## Confluence

知识库 Confluence访问地址：http://156.12.19.61:8090/admin

管理账号admin/admin

数据库安装目录 :/usr/local/mysql root/123456

## Pdf查询

db.getCollection('ysqRzVO').find({dsdjxh:'20126100000011438941'}).sort({lrrq:-1});

db.getCollection('ysqxxVO').find({djxh:'10126101001037550903'})

db.getCollection('zlpzSxVO').find({ysqxxId:''})

db.getCollection('zlpzxxVO').find({\_id:''})

db.getCollection('fs.files').find({filename:''})

db.getCollection('fs.chunks').find({files\_id:ObjectId("")})