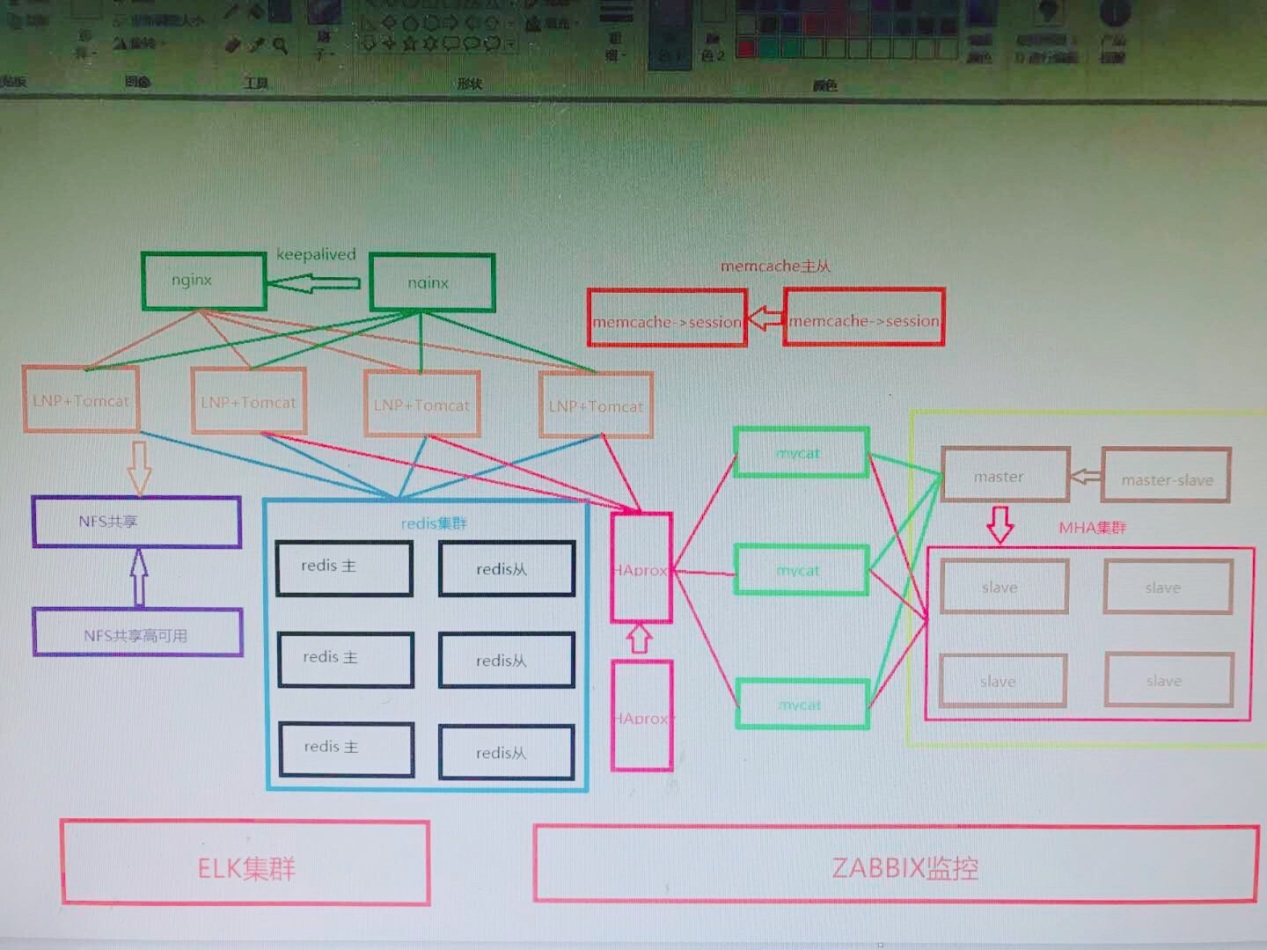
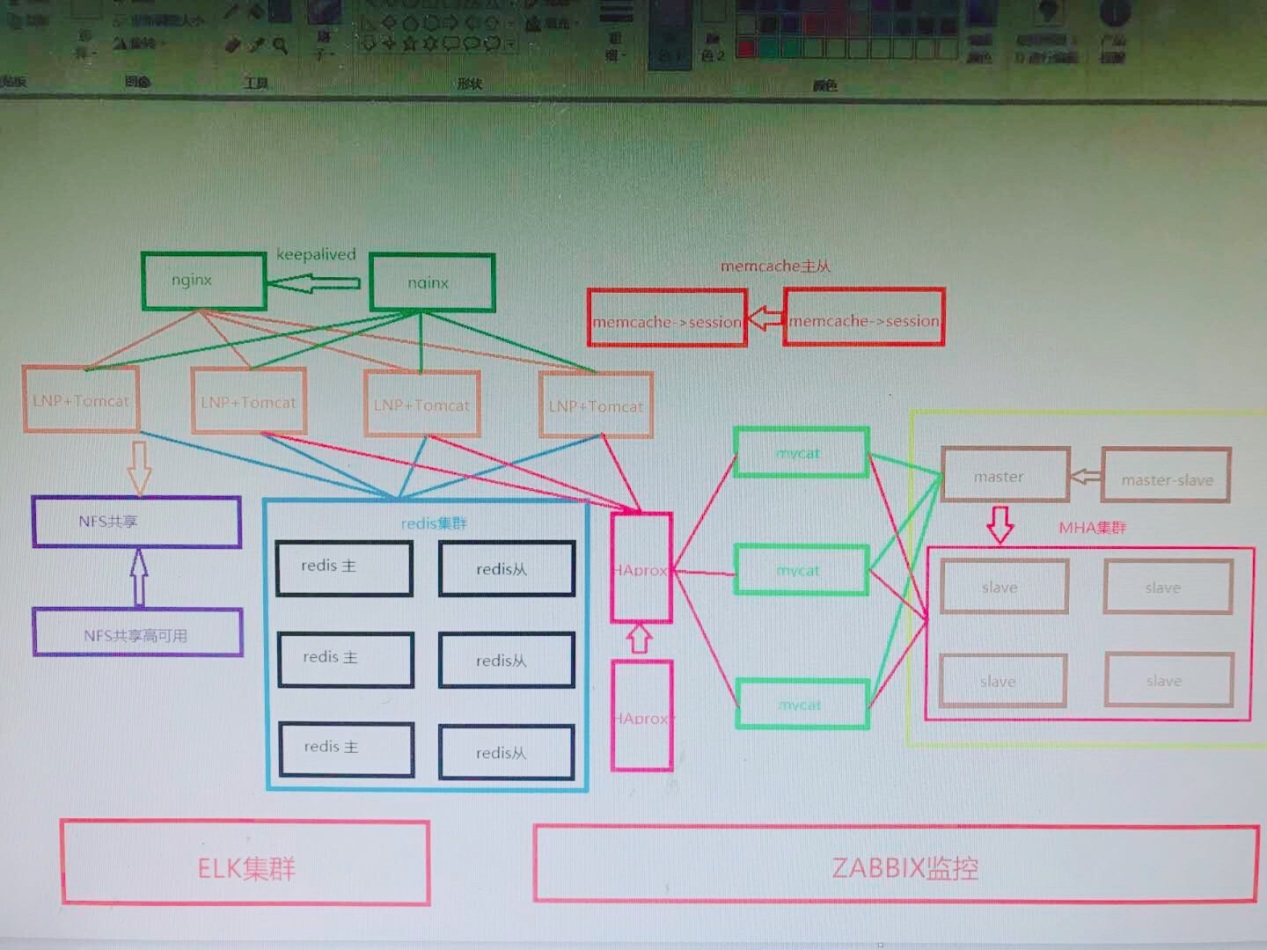
**架构图**

****

**Nginx配置**

**Nginx调度**

**Nginx 的Keepalived高可用**

****

# Nfs共享+drbd磁盘共享+keepalived高可用

环境：

nfs-0001:192.168.1.103 nfs-0002:192.168.1.104 vip:192.168.1.200

大致思路：

* 在华为云上新购买两块云硬盘，分别挂载在两台nfs服务器上，
* 使用drbd技术将两块磁盘做数据热备，将其mount到nfs共享目录/var/webroot上，
* 使用keepalived技术，将两台nfs实现高可用
* 所有的web主机，将nfs共享目录(/var/webroot)挂载到/usr/local/nginx/html目录，
* 只要在nfs主服务器上/var/webroot目录准备好html文件，即可给所有web服务器对外发布

## drbd 安装配置

|-- 内核模块 （实现数据同步的具体功能）

`|- 内核模块安装

|- 官网下载软件包 drbd-8.4.11-1.tar.gz

|- 上传到要安装的主机，解包 tar -zxf drbd-8.4.11-1.tar.gz

|- 安装编译依赖软件包 yum install -y gcc make automake pkgconfig flex git kernel-devel kernel-headers

|- 编译安装 make && make install

|- 验证 modprobe drbd

|- lsmod |grep -i drbd

|-- 命令行工具，实现指令交换

`|- 官网下载软件包 drbd-utils.tar.gz

|- 上传到要安装的主机，解包 tar -zxf drbd-utils.tar.gz

|- 编译安装 ./configure --prefix=/usr/local/drbd --without-83support --with-udev \

--with-initscripttype=systemd --without-manual

|- 编译安装 make && make install

|- 编写配置文件 /usr/local/drbd/etc/drbd.d/drbd.res (详见配置文件)

drbdadm create-md r1 # 初始化 drbd (两台)

systemctl restart drbd # 两台都需要重启

drbdadm primary r1 - -force # - -force强制把其中一台设置为主

drbdadm secondary r1 # 设置为从

cat /proc/drbd # 查看状态

### 安装操作

yum install -y gcc make automake pkgconfig flex git kernel-devel kernel-headers

tar caf drbd-8.4.11-1.tar.gz

tar -xf drbd-8.4.11-1.tar.gz

cd drbd-8.4.11-1

make

make install

modprobe drbd

lsmod |grep -i dibd

tar -zxf drbd-utils.tar.gz

drbd-utils

sh autogen.sh #安装不了得先执行这个脚本

./configure --prefix=/usr/local/drbd --without-83support --with-udev --with-initscripttype=systemd --without-manual

make

make install

配置文件：

vim /usr/local/drbd/etc/drbd.d/drbd.res

[root@nfs-0001 ~]# cat /usr/local/drbd/etc/drbd.d/drbd.res

resource r1 { #这个r1是定义资源的名字

protocol C; #同步协议：C为完全同步，还有A和 B

on nfs-0001 { #on开头，后面是主机名称

device /dev/drbd0; #drbd设备名称

disk /dev/vdb; #drbd0使用的磁盘分区为新买的的磁盘

address 192.168.1.103:7789; #设置drbd监听地址与端口

meta-disk internal;

}

on nfs-0002 { #第二台主机配置

device /dev/drbd0;

disk /dev/vdb;

address 192.168.1.104:7789;

meta-disk internal;

}

}

在nfs1和nfs2上同时进行以上的安装步骤，安装配置完毕后同时初始化和启动

drbdadm create-md r1 #初始化drbd

systemctl restart drbd #启动drbd

drbdadm primary r1 --force #将其中一台设置为primary(主),--force为强制

cat /proc/drbd #查看状态，第一此两边会有数据同步

drbdadm secondary r1 # 设置为secondary（从）

mkfs.xfs /dev/drbd0

mount /dev/drbd0 /var/webroot

### 脑裂故障解决

　脑裂修复：在一台机器重启后出现了脑裂现象；node2和node1都unkown状况了，node2启动drbd异常

　　下面先将node2进行了primary

　　在drbd2处理方法：

　　[root@drbd2 /]# drbdadm disconnect r1

　　[root@drbd2 /]# drbdadm secondary r1

　　[root@drbd2 /]# drbdadm -- --discard-my-data connect r0

　　#该命令告诉drbd，secondary上的数据不正确，以primary上的数据为准。

　　需要在drbd1上重连接资源：

　　[root@drbd1 ~]# drbdadm connect r1

　　再次启动drbd2上的drbd服务，OK

　　这样drbd1就能和drbd2开始连接上了，并且保证数据不会丢失：

如果异常connect，可以先disconnect下。然后发现启动也正常了

## 配置nfs共享

(两台nfs服务器相同操作)

yum -y install rpcbind nfs-utils

vim /etc/exports #配置文件

配置内容为：

/var/webroot \*(rw)

mkdir /var/webroot

chmod 777 /var/webroot

systemctl restart nfs-utils

systemctl enable nfs-utils

## 安装keepalived

yum -y install keepalived #两台都安装

vim /etc/keepalived/keepalived.conf

MASTER配置：nfs-0001主机 192.168.1.103

/bin/bash: Configuration: command not found

bal\_defs {

router\_id nfs #两台nfs服务器相同

}

vrrp\_script chk\_drbd {

script "/etc/keepalived/check\_nfs.sh" # 调用检查脚本，请看后面检查脚本内容

interval 5 # check every 2 seconds

}

vrrp\_instance NFS {

state MASTER #设置为MASTER主

interface eth0

track\_interface {

eth0

}

virtual\_router\_id 170 #两台一样

priority 100 #优先权，比从大就行

advert\_int 2

authentication {

auth\_type PASS #两边一样就行

auth\_pass drbd\_checknfs #keepalived匹配密码，两边一样就行

}

virtual\_ipaddress {

192.168.1.202/24 #vip地址

}

notify\_stop "/etc/keepalived/notify\_stop.sh" #调用停止脚本

notify\_master "/etc/keepalived/notify\_master.sh" #变成MASTER状态之后，调用脚本

track\_script {

chk\_drbd weight=0 # +2 if process is present

}

}

BACKUP配置:nfs-0002主机 192.168.1.104

/bin/bash: Configuration: command not found

bal\_defs {

router\_id nfs

}

vrrp\_instance NFS {

state BACKUP

interface eth0

track\_interface {

eth0

}

virtual\_router\_id 170

priority 50

advert\_int 2

authentication {

auth\_type PASS

auth\_pass drbd\_checknfs

}

virtual\_ipaddress {

192.168.1.202/24

}

notify\_master "/etc/keepalived/notify\_master.sh" #master状态调用的脚本

notify\_backup "/etc/keepalived/notify\_backup.sh" #backup状态调用的脚本

track\_script {

chk\_drbd weight=0 # +2 if process is present

}

}

总共有四个脚本：

check\_nfs.sh #master主机的检测脚本

notify\_stop.sh #master主机的停止drbd和umount脚本

notify\_master.sh #两台都要用，变为master后调用的脚本

notify\_backup.sh #backup主机的drbd恢復脚本

### check\_nfs.sh脚本内容：

[root@nfs-0001 ~]# vim /etc/keepalived/check\_nfs.sh

#!/bin/sh

###检查nfs可用性：进程和是否能够挂载

/sbin/service nfs status &>/dev/null

if [ $? -ne 0 ];then

###如果服务状态不正常，先尝试重启服务

/sbin/service nfs restart

/sbin/service nfs status &>/dev/null

if [ $? -ne 0 ];then

###若重启nfs服务后，仍不正常

###卸载drbd设备

umount /dev/drbd0

###将drbd主降级为备

drbdadm secondary r1

#关闭keepalived

/sbin/service keepalived stop

fi

fi

### notify\_stop.sh 脚本内容

[root@nfs-0001 ~]# vim /etc/keepalived/notify\_stop.sh

#!/bin/bash

time=`date "+%F %H:%M:%S"`

echo -e "$time ------notify\_stop------\n" >> /etc/keepalived/logs/notify\_stop.log

/sbin/service nfs stop &>> /etc/keepalived/logs/notify\_stop.log

/bin/umount /var/webroot &>> /etc/keepalived/logs/notify\_stop.log

/sbin/drbdadm secondary r1 &>> /etc/keepalived/logs/notify\_stop.log

echo -e "\n" >> /etc/keepalived/logs/notify\_stop.log

### notify\_master.sh脚本内容

[root@nfs-0001 ~]# vim /etc/keepalived/notify\_master.sh #两台都用

#!/bin/bash

time=`date "+%F %H:%M:%S"`

echo -e "$time ------notify\_master------\n" >> /etc/keepalived/logs/notify\_master.log

/sbin/drbdadm primary r1 &>> /etc/keepalived/logs/notify\_master.log

/bin/mount /dev/drbd0 /var/webroot &>> /etc/keepalived/logs/notify\_master.log

/sbin/service nfs restart &>> /etc/keepalived/logs/notify\_master.log

echo -e "\n" >> /etc/keepalived/logs/notify\_master.log

### notify\_backup.sh 脚本内容

[root@nfs-0002 ~]# vim /etc/keepalived/notify\_backup.sh #只在backup上用

#!/bin/bash

time=`date "+%F %H:%M:%S"`

echo -e "$time ------notify\_backup------\n" >> /etc/keepalived/logs/notify\_backup.log

/sbin/service nfs stop &>> /etc/keepalived/logs/notify\_backup.log

/bin/umount /dev/drbd0 &>> /etc/keepalived/logs/notify\_backup.log

/sbin/drbdadm secondary r1 &>> /etc/keepalived/logs/notify\_backup.log

echo -e "\n" >> /etc/keepalived/logs/notify\_backup.log

在华为云上申请192.168.1.202的地址作为vip

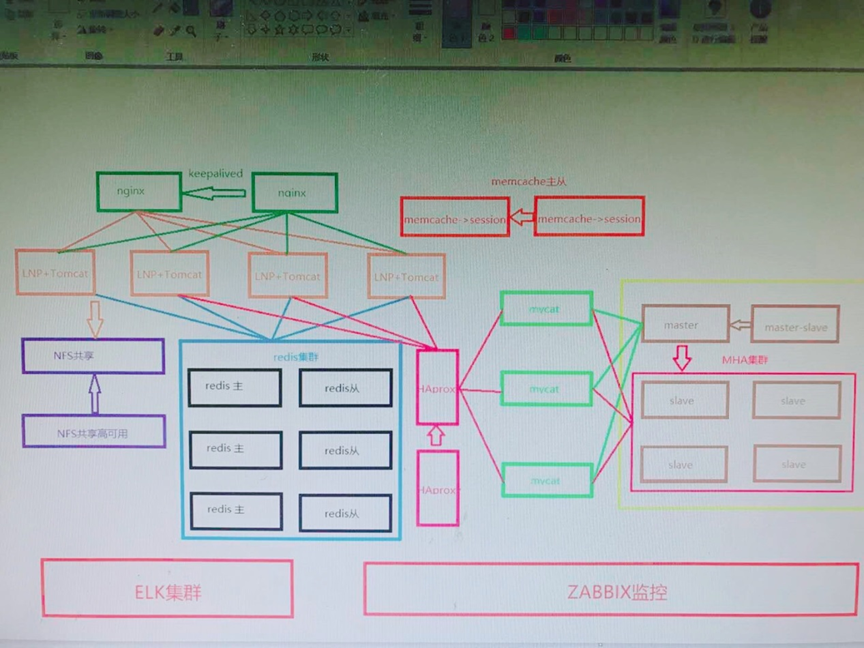
并且绑定192.168.1.103 和 192.168.1.104两台主机服务器

所有的nginx主机，都安装 nfs-utils

所有的nginx主机，都用vip192.168.1.202挂载，在这用ansible批量挂载共享目录：

ansible web -m shell -a 'umount /usr/local/nginx/html/'

ansible web -m shell -a 'mount -t nfs 192.168.1.202:/var/webroot /usr/local/nginx/html/'



# Mysql数据库主从

主：192.168.1.21 从：192.168.1.22--192.168.1.26

## 准备软件包

[root@jumpserver 数据库软件包]# tar -xf mysql-5.7.17.tar -C /var/ftp/localrepo/

[root@jumpserver 数据库软件包]# cd /var/ftp/localrepo/

[root@jumpserver localrepo]# ls

mysql-community-client-5.7.17-1.el7.x86\_64.rpm mysql-community-libs-compat-5.7.17-1.el7.x86\_64.rpm

mysql-community-common-5.7.17-1.el7.x86\_64.rpm mysql-community-minimal-debuginfo-5.7.17-1.el7.x86\_64.rpm

mysql-community-devel-5.7.17-1.el7.x86\_64.rpm mysql-community-server-5.7.17-1.el7.x86\_64.rpm

mysql-community-embedded-5.7.17-1.el7.x86\_64.rpm mysql-community-test-5.7.17-1.el7.x86\_64.rpm

mysql-community-embedded-compat-5.7.17-1.el7.x86\_64.rpm nginx-1.15.10-1.el7.centos.x86\_64.rpm

mysql-community-embedded-devel-5.7.17-1.el7.x86\_64.rpm repodata

mysql-community-libs-5.7.17-1.el7.x86\_64.rpm

[root@jumpserver localrepo]# createrepo --update .

## ansible 批量传送/etc/my.cnf文件

[root@jumpserver localrepo]# vim /etc/ansible/hosts #hosts文件添加了所有的mysql主机

[mysql]

192.168.1.21

192.168.1.22

192.168.1.23

192.168.1.24

192.168.1.25

192.168.1.26

[root@jumpserver localrepo]# ansible mysql -m ping

**所有机器安装mysql**

[root@jumpserver localrepo]# ansible mysql -m yum -a 'name=mysql-server state=installed'

用ansible批量传配置文件，以及修改server-id用ip地址

在 mysql-master 上修改 my.cnf：

[mysqld]

bind-address=0.0.0.0 #监听地址，有些版本差异，添上不会出问题

server-id= {{id.stdout}} #用ansible批量传送，传参方式获取

binlog-format=mixed #binglog日志格式

log\_bin=/var/log/mysql/mysql-bin

relay-log=/var/log/mysql/relay-log #relay-log中继日志

relay-log-index=/var/log/mysql/relay-log.info #relay-log中继日志

relay-log-info-file=/var/log/mysql/relay-log.info

relay\_log\_purge=0 #不删除日志，MHA用得上

#以下是主从半同步复制的配置（semi sync master/slave）

plugin-load=rpl\_semi\_sync\_master=semisync\_master.so #加载主半同步复制模块

plugin-load=rpl\_semi\_sync\_slave=semisync\_slave.so #加载从半同步复制模块

rpl\_semi\_sync\_slave\_enabled=1 #启用从半同步复制模式

rpl\_semi\_sync\_master\_enabled=1 #启用主半同步复制模式

rpl\_semi\_sync\_master\_timeout=3000

[root@jumpserver nsd1811]# ifconfig eth0 | grep -Po "(?<=inet 192.168.1.)\d+"

250 #本机地址为：192.168.1.250，获取到了250

Yum文件内容如下：

[root@jumpserver nsd1811]# vim mysqlcnf.yml

---

- hosts: mysql

remote\_user: root

tasks:

- shell: ifconfig eth0 | grep -Po "(?<=inet 192.168.0.)\d+"

register: id

- template:

src: my.cnf

dest: /etc/my.cnf

owner: root

group: root

mode: 0644

[root@jumpserver nsd1811]# ansible-playbook mysqlcnf.yml

**Ansible 批量操作**

[root@jumpserver nsd1811]# ansible mysql -m shell -a 'mkdir /var/log/mysql'

[root@jumpserver nsd1811]# ansible mysql -m shell -a 'chown 27.27 /var/log/mysql'

[root@jumpserver nsd1811]# ansible mysql -m shell -a 'chown 27.27 /etc/my.cnf'

[root@jumpserver nsd1811]# ansible mysql -m shell -a 'mysqld --initialize-insecure --user mysql'

## #------master------------------------#

重启服务 systemctl restart mysqld

### 初始化master

开启binlog日志（之前配置文件中已批量配置好）

reset master;

### 添加同步用户

create user 'repluser '@'%' identified by '123456';

### 使用innobackup备份数据库

安装备份工具 xtrabackup

yum install -y percona-xtrabackup-24

案例:将所有库完全备份到/allbak (目录不需事先创建,备份程序自动创建)

innobackupex --user root --password 123456 /allbak -no-timestamp

## #------slave-------------------------#

### 使用 innobackup 恢复备份

yum install -y percona-xtrabackup-24

备份的数据中有以下文件:

Xtra backup\_logfile已提交日志

Ibdata1文件是未提交数据日志

**1 准备恢复**

[root@sql51 ~]# cat /allbak/xtrabackup\_checkpoints

backup\_type = full-backuped #备份内容完全备

from\_lsn = 0

to\_lsn = 3038691 #备注：from\_lsn = 0 to\_lsn = 3038691 日志范围

last\_lsn = 3038700 #数据库下次备份开始序列号

compact = 0

recover\_binlog\_info = 0

[root@sql51 ~]# innobackupex --apply-log /allbak #准备还原

190219 10:12:44 completed OK! #看到OK就正常

**2 开始还原**

[root@sql51 ~]# systemctl stop mysqld.service

[root@sql51 ~]# rm -rf /var/lib/mysql

[root@sql51 ~]# ls /var/lib/mysql

ls: 无法访问/var/lib/mysql: 没有那个文件或目录

[root@sql51 ~]# innobackupex --copy-back /allbak #开始还原

190219 10:19:35 innobackupex: Starting the copy-back operation

190219 10:19:44 completed OK!

[root@sql51 ~]# ls /var/lib/mysql -l #查看数据库目录里面的所有文件权限为root

[root@sql51 ~]# chown -R mysql:mysql /var/lib/mysql

[root@sql51 ~]# systemctl start mysqld

[root@sql51 ~]# ls -l /var/lib/mysql

[root@sql51 ~]# mysql -uroot -p654321

mysql>

mysql> show databases;

### 配置主从

启动 mysql 设置主从，binlog文件（master\_log\_file）及其执行位置（master\_log\_pos）在 /var/lib/mysql/xtrabackup\_info 可查找到，也可在主库上执行：show master status\G;语句

change master to master\_host='192.168.1.18',\

master\_user='repl',master\_password='lper',\

master\_log\_file="mysql-bin.000001", master\_log\_pos=615;

start slave;

### 检查验证

show slave status\G

## Mysql数据库主从半同步复制模式

### #-------semi sync master----------------#

show plugins; #查看 mysql 插件

install plugin rpl\_semi\_sync\_master soname 'semisync\_master.so'; #安装半同步插件

set global rpl\_semi\_sync\_master\_enabled=1; #开启半同步

set global rpl\_semi\_sync\_master\_timeout=1000; #等待超时时间

#等待超时时间：设置此参数值（ms）,为了防止半同步复制在没有收到确认的情况下发生堵塞，如果Master在超时之前没有收到任何确认，将恢复到正常的异步复制，并继续执行没有半同步的复制操作。

查看状态

show global variables like '%rpl\_semi%';

show global status like '%rpl\_semi%';

show variables like 'have\_dynamic\_loading'; #查看是否允许动态加载模块

select plugin\_name,plugin\_status from information\_schema.plugins where plugin\_name like '%semi%'; #查看模块是否被加载

show variables like "rpl\_semi\_sync\_%\_enabled"; #查看是否启用

### #-------semi sync slave-----------------#

show plugins; #查看 mysql 插件

install plugin rpl\_semi\_sync\_slave soname 'semisync\_slave.so'; #手动安装半同步插件

set global rpl\_semi\_sync\_slave\_enabled=1; #手动开启半同步

show global variables like '%rpl\_semi%'; #查看状态

重启 IO 线程

stop slave io\_thread;

start slave io\_thread;

### #-------主从半同步my.cnf 永久配置内容---------------------#

plugin-load="rpl\_semi\_sync\_master=semisync\_master.so" #加载主配置模块

plugin-load="rpl\_semi\_sync\_slave=semisync\_slave.so" #加载从配置模块

rpl\_semi\_sync\_slave\_enabled = 1 #启用从配置半同步复制模式

rpl\_semi\_sync\_master\_enabled = 1 #启用主配置半同步复制模式

rpl\_semi\_sync\_master\_timeout = 3000

#等待超时时间 设置此参数值（ms）,为了防止半同步复制在没有收到确认的情况下发生堵塞，如果Master在超时之前没有收到任何确认，将恢复到正常的异步复制，并继续执行没有半同步的复制操作。

# Mysql-MHA集群配置

MHAmanager服务器：192.168.1.20 其他节点：192.168.1.21-26 vip 192.168.1.203

## MHA node节点配置：

192.16.1.21--192.168.1.26

安装依赖包：yum install gcc pcre-devel pkgconfig autoconf automake perl-ExtUtils-MakeMaker perl-CPAN perl-DBI perl-DBD-MySQL

安装mha node节点包软件：mha4mysql-node-0.56-0.el6.noarch

安装方法：将所有包放入跳板机，做成私有源（createrepo –update .）后ansible批量安装

## Mysql主库绑定VIP

把vip 192.168.1.203配置当前主数据库服务器192.168.4.21上

[root@sql51 ~]# ifconfig eth0:1 192.168.1.203

[root@sql51 ~]# ifconfig eth0:1

eth0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.1.203 netmask 255.255.255.0 broadcast 192.168.1.255

ether 52:54:00:57:34:54 txqueuelen 1000 (Ethernet)

所有服务器之间ssh免密登陆

所有的数据库my.cnf都设置了不自动删除本机的中继日志文件

[mysqld]

server\_id=55

relay\_log\_purge=off #不自动删除本机的中继日志文件

mysql> set global relay\_log\_purge=off;

## 登陆主库的mysql,给root授权

1授权监控用户,在21上做,22-26自动同步

mysql> grant all on \*.\* to root@"%" identified by "123456";

Query OK, 0 rows affected, 1 warning (0.03 sec)

## MHA manager管理主机配置

管理主机地址：192.168.1.20

### 必须先装数据节点安装包:

mha4mysql-node-0.56-0.el6.noarch.rpm

[root@sql56 mha-soft-student]# rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm

### 再安装管理包:

mha4mysql-manager-0.56.tar.gz

[root@sql56 mha-soft-student]# tar -xvf mha4mysql-manager-0.56.tar.gz

[root@sql56 mha-soft-student]# ls

mha4mysql-manager-0.56

[root@sql56 mha-soft-student]# cd mha4mysql-manager-0.56/

[root@sql56 mha4mysql-manager-0.56]# ls

AUTHORS COPYING inc Makefile.PL META.yml rpm t

bin debian lib MANIFEST README samples tests

[root@sql20 mha4mysql-manager-0.56]# which perl #查看perl命令

/usr/bin/perl

[root@sql20 mha4mysql-manager-0.56]# perl Makefile.PL #先检测

\*\*\* Module::AutoInstall version 1.03

\*\*\* Checking for Perl dependencies...

[Core Features]

- DBI ...loaded. (1.627) #括号内不为0,表示依赖包被安装

- DBD::mysql ...loaded. (4.023) #括号内不为0,表示依赖包被安装

- Time::HiRes ...loaded. (1.9725) #括号内不为0,表示依赖包被安装

- Config::Tiny ...loaded. (2.14) #括号内不为0,表示依赖包被安装

- Log::Dispatch ...loaded. (2.41) #括号内不为0,表示依赖包被安装

- Parallel::ForkManager ...loaded. (1.18) #括号内不为0,表示依赖包被安装

- MHA::NodeConst ...loaded. (0.56) #括号内不为0,表示依赖包被安装

\*\*\* Module::AutoInstall configuration finished.

Checking if your kit is complete...

Looks good

Writing Makefile for mha4mysql::manager

Writing MYMETA.yml and MYMETA.json

[root@sql20 mha4mysql-manager-0.56]# make

[root@sql20 mha4mysql-manager-0.56]# make install

[root@sql20 mha4mysql-manager-0.56]# masterha\_ #连续tab,显示全部命令

### 创建服务的主配置文件

参考模板app1.cnf

[root@sql20~]#cp /root/mha-soft-student/mha4mysql-manager-0.56/samples/conf/app1.cnf /etc/mha\_manager/ #拷贝模板到这个目录(目录需要手动创建)

[root@sql56 ~t]# ls /etc/mha\_manager

[root@mha ~]# vim /etc/mha\_manager/app1.cnf

[server default]

manager\_log=/etc/mha\_manager/manager.log

manager\_workdir=/etc/mha\_manager #工作目录

master\_ip\_failover\_script=/usr/local/bin/master\_ip\_failover #切换脚本路径

password=123456 #监控数据库用户的授权密码

ping\_interval=1

remote\_workdir=/etc/mha\_manager

repl\_password=123456 #主从同步的用户的密码

repl\_user=repluser #mysqL主从同步的用户名，由mysql创建并授权

ssh\_port=22 #ssh免密端口

ssh\_user=root #ssh免密用户名

user=root #监控数据库的用户名，由mysql创建并授权

[server1]

candidate\_master=1 #竞选主库

hostname=192.168.1.21

[server2]

candidate\_master=1 #竞选主库

hostname=192.168.1.22

[server3]

candidate\_master=1 #竞选主库

hostname=192.168.1.23

[server4]

hostname=192.168.1.24

no\_master=1 #不竞选主库

[server5]

hostname=192.168.1.25

no\_master=1

[server6]

hostname=192.168.1.26

no\_master=1

部署故障切换脚本文件

在管理机主配置文件中如下语句里指定了故障切换脚本的目录与名称:

master\_ip\_failover\_script=/usr/local/bin/master\_ip\_failover #加载故障切换脚本

将脚本放置/usr/local/bin/目录中.并且命名为:master\_ip\_failover

mha切换vip是靠脚本实现，

vim编辑脚本master\_ip\_failover设置vip 在（line:35）

my $vip = '192.168.1.203/24'; # Virtual IP

cp master\_ip\_failover /usr/local/bin/

chmod 755 /usr/local/bin/master\_ip\_failover #赋予执行权限

### 检测ssh免密

masterha\_check\_ssh --conf=/etc/mha\_manager/app1.cnf

### 验证数据节点的主从同步配置

[root@sql56 mha\_manager]# cd /usr/local/bin/

[root@sql56 bin]# masterha\_check\_repl --conf=/etc/mha\_manager/app1.cnf

Fri Feb 22 19:31:00 2019 - [info] Alive Servers: #数据库服务器列表

Fri Feb 22 19:31:00 2019 - [info] 192.168.4.51(192.168.4.51:3306)

Fri Feb 22 19:31:00 2019 - [info] 192.168.4.52(192.168.4.52:3306)

Fri Feb 22 19:31:00 2019 - [info] 192.168.4.53(192.168.4.53:3306)

Fri Feb 22 19:31:00 2019 - [info] 192.168.4.54(192.168.4.54:3306)

Fri Feb 22 19:31:00 2019 - [info] 192.168.4.55(192.168.4.55:3306)

Fri Feb 22 19:31:00 2019 - [info] Alive Slaves: #存活的从服务器

Fri Feb 22 19:31:00 2019 - [info] 192.168.4.52(192.168.4.52:3306) Version=5.7.17-log (oldest major version between slaves) log-bin:enabled

Fri Feb 22 19:31:00 2019 - [info] Replicating from 192.168.4.51(192.168.4.51:3306)

验证数据节点时可以看到有哪些数据库服务器,以及目前主库服务器是192.168.4.51,

MySQL Replication Health is OK. #此结果显示成功

#如果显示 MySQL Replication Health is NOT OK! 不成功

启动管理服务

[root@sql56 ~]# masterha\_manager --conf=/etc/mha\_manager/app1.cnf --remove\_dead\_master\_conf --ignoer\_last\_failover

#启动信息会占用一个终端显示,,,,此时不能ctrl + c，另开一个终端查看运行状态

# --remove\_dead\_master\_conf //删除宕机主库配置

# --ignore\_last\_failover //忽略xxx.health文件默认8个小时内主库连续宕机,不切换主库,如果添加 --ignore\_last\_failover,则8小时内主库宕机了,就会切换

### 新开一个终端查看运行状态

[root@sql56 ~]# masterha\_check\_status --conf=/etc/mha\_manager/app1.cnf

app1 (pid:13953) is running(0:PING\_OK), master:192.168.1.21 #当前主库

### 测试高可用:目前主库21故障

模拟主库服务器21宕机后访问VIP继续使用数据库

[root@sql51 bin]# systemctl stop mysqld.service

**启动未关的终端将会有信息显示：**

[root@sql56 mha\_manager]# masterha\_manager --conf=/etc/mha\_manager/app1.cnf --remove\_dead\_master\_conf --igno\_last\_failover

Fri Feb 22 17:00:45 2019 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Fri Feb 22 17:00:45 2019 - [info] Reading application default configuration from /etc/mha\_manager/app1.cnf..

Fri Feb 22 17:00:45 2019 - [info] Reading server configuration from /etc/mha\_manager/app1.cnf..

Creating /var/tmp if not exists.. Ok. #切换开始

Checking output directory is accessible or not..

ok.

Binlog found at /var/lib/mysql, up to 51log.000007

Fri Feb 22 17:03:01 2019 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Fri Feb 22 17:03:01 2019 - [info] Reading application default configuration from /etc/mha\_manager/app1.cnf..

Fri Feb 22 17:03:01 2019 - [info] Reading server configuration from /etc/mha\_manager/app1.cnf..

[root@sql56 mha\_manager]# #切换中以上状态会自动结束,到此处就切换完成

**验证**

在其他数据库上查看主从 show slave status\G 192.168.1.22或者192.168.1.23变成了主库

## 主库21的恢复

此时主库飘到了22,将21恢复并其加入集群,

### 恢复数据

21恢复宕机期间的数据,从新主库22上备份数据库,放到21上恢复,

### 将21数据库设置为22的从库

mysql> change master to

-> master\_host="192.168.4.22",

-> master\_user="repluser",

-> master\_password="123456",

-> master\_log\_file="52log.000002",

-> master\_log\_pos=654;

Query OK, 0 rows affected, 2 warnings (0.29 sec

mysql> start slave;

Query OK, 0 rows affected (0.04 sec)

mysql> show slave status \G;

### 配置app1.cnf 文件重新添加server1的配置

将之前21的server配置添加上

[server1]

candidate\_master=1

hostname=192.168.1.21

### MHA manager上重新验证数据节点

[root@sql20 bin]# masterha\_check\_repl --conf=/etc/mha\_manager/app1.cnf

MySQL Replication Health is OK.

# Mycat

Mycat 服务器：192.168.1.27-192.168.1.29

[root@jumpserver ~]# scp .ssh/id\_rsa 192.168.1.27:/root/.ssh/

[root@jumpserver ~]# scp .ssh/id\_rsa 192.168.1.28:/root/.ssh/

[root@jumpserver ~]# scp .ssh/id\_rsa 192.168.1.29:/root/.ssh/

主库上授权查询用户：

create user 'zdd'@'%' IDENTIFIED BY '123456'; #创建用户

grant select on \*.\* to zdd@'%'; #授权用户查询权限

select host,user from mysql.user; #查看所有用户

show grants for zdd@"%"; #查看root用户的权限

## 安装jdk环境

[root@jumpserver ~]# ansible mycat -m shell -a 'yum -y install java-1.8.0'

## 安装mycat

[root@jumpserver ~]# scp Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz 192.168.1.27:/root

Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz 100% 15MB 107.9MB/s 00:00

[root@jumpserver ~]# scp Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz 192.168.1.28:/root

Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz 100% 15MB 88.5MB/s 00:00

[root@jumpserver ~]# ansible mycat -m shell -a 'tar -xf /root/Mycat-server-1.6-RELEASE-20161028210-linux.tar.gz'

[root@jumpserver ~]# ansible mycat -m shell -a 'mv /root/mycat /usr/local/' #放置mycat目录

[root@jumpserver ~]# ansible mycat -m shell -a 'ls /usr/local/mycat' #查看目录

## 配置server.xml

[root@mycat-0001 ~]# vim /usr/local/mycat/conf/server.xml

80 <user name="root"> #访问mycat 的用户名，与数据库用户无关

81 <property name="password">123456</property> #root访问的密码

82 <property name="schemas">mydb</property> #访问的虚拟IP

83

95 <user name="zdd"> #新增访问mycat用户，与数据库用户无关

96 <property name="password">123456</property>

97 <property name="schemas">mydb</property>

98 <property name="readOnly">true</property>

99 </user>

## 配置schema.xml

<?xml version="1.0"?>

<!DOCTYPE mycat:schema SYSTEM "schema.dtd">

<mycat:schema xmlns:mycat="http://io.mycat/">

<schema name="mydb" checkSQLschema="false" sqlMaxLimit="100" dataNode='dn1'>

/> -->

<dataNode name="dn1" dataHost="localhost1" database="db1" />

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="3"

writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">

<heartbeat>select user()</heartbeat>

<writeHost host="hostM1" url="192.168.1.203:3306" user="admin" password="123456">

<!-- can have multi read hosts -->

<readHost host="hostS2" url="192.168.1.24:3306" user="zdd" password="123456" />

<readHost host="hostS2" url="192.168.1.25:3306" user="zdd" password="123456" />

<readHost host="hostS2" url="192.168.1.26:3306" user="zdd" password="123456" />

</writeHost>

</dataHost>

</mycat:schema>

以上设置中：hostM1的用户admin是数据库授权用户 ，至少有插入权限

hostS\* 的用户为数据库授权查询用户，有查询权限

[root@mycat-0001 ~]# /usr/local/mycat/bin/mycat start

Usage: /usr/local/mycat/bin/mycat { console | start | stop | restart | status | dump }

## 测试Mycat：

在其他任意一台数据库服务器上执行不断查询操作，如果不行请重启所有mysql

[root@mysql-0005 ~]# mysql -uroot -p123456 -h192.168.1.27 -P8066 -e'select @@hostname;'

mysql: [Warning] Using a password on the command line interface can be insecure.

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

| @@hostname |

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

| mysql-0006 | #查询到是192.168.1.26的mysql-0006主机

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

[root@mysql-0005 ~]# mysql -uroot -p123456 -h192.168.1.27 -P8066 -e'select @@hostname;'

mysql: [Warning] Using a password on the command line interface can be insecure.

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

| @@hostname |

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

| mysql-0005 | #再次查询到是192.168.1.25的mysql-0005主机

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

[root@mysql-0005 ~]# mysql -uroot -p123456 -h192.168.1.27 -P8066 -e'select @@hostname;'

mysql: [Warning] Using a password on the command line interface can be insecure.

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

| @@hostname |

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

| mysql-0004 | #再次查询到是192.168.1.24的mysql-0004主机

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

# Haproxy

Haproxy主机地址：192.168.1.31---192.168.1.32

## 安装haproxy

Yum –y install haproxy

Vim /etc/haproxy/haproxy.cfg

## 配置

修改 /etc/haproxy/haproxy.cfg （两台服务器相同，配置完可传送到另一台）

…….

#---------------------------------------------------------------------

# main frontend which proxys to the backends

#---------------------------------------------------------------------

listen mycat\_3306 \*:3306

mode tcp # mysql 得使用 tcp 协议

option tcpka # 使用长连接

balance leastconn # 最小连接调度算法

server mycat\_01 192.168.1.31:8066 check inter 3000 rise 1 maxconn 1000 fall 3

server mycat\_02 192.168.1.32:8066 check inter 3000 rise 1 maxconn 1000 fall 3

## 启动haproxy

Systemctl start haproxy

## 验证haproxy

在装有mysql 的数据库服务器上

Mysql –uroot –p123456 –h192.168.1.31 –e ‘select @@hostname’

#反复执行，得到与测试mycat相同结果

## Haproxy+keepalived

Haproxy：192.168.1.31---192.168.1.32

Vip:192.168.1.204 (华为云上购买虚拟IP,并绑定到两台haproxy服务器上)

## 安装keepalived

Yum –y install keepalived

## 配置keepalived

Haproxy-0001上配置：

! Configuration File for keepalived

global\_defs {

router\_id haproxy1 #可相同可不同

}

vrrp\_strict chk\_haproxy{

script "killall -0 haproxy"

interval 2

}

vrrp\_instance mycat1 { #两边都是mycat1

state BACKUP

interface eth0

virtual\_router\_id 160 #不要用默认的

priority 200

advert\_int 2

authentication {

auth\_type PASS #验证用户名：一定要相同

auth\_pass 1111 #验证密码：一定要相同

}

virtual\_ipaddress {

192.168.1.204/24

}

track\_script{

chk\_haproxy weight=0

}

}

Haporxy-0002上配置

! Configuration File for keepalived

global\_defs {

router\_id haproxy1

}

vrrp\_strict chk\_haproxy{

script "killall -0 haproxy"

interval 2

}

vrrp\_instance mycat1 {

state BACKUP

interface eth0

virtual\_router\_id 160 #不要用默认的

priority 100

advert\_int 2

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.1.204/24

}

track\_script{

chk\_haproxy weight=0

}

}

## 验证keepalive

在任意一台数据库服务器上

1、mysql –uroot -p123456 –h192.168.1.204 –e’select @@hostname’ #可看到轮询效果

2、ip a s 查看vip地址是否在MASTER上

3、停止haproxy-0001上的keepalived ，在haproxy-0002上ip a s 查看vip和查看轮询

4、第3步验证成功，恢复haproxy-0001上的keepalived，并ip a s 查看vip和查看轮询

# Redis集群

集群必须至少要有三台主库,

--replicate 1 定义每个主库有1个从库,3台主库,3台从库,总共6台

--replicate 2 定义每个主库有2个从库,3台主库,6台从库,总共9台 以此类推

集群地址：192.168.1.41---192.168.1.46

管理主机：192.168.1.41

部署id\_rsa文件：

ssh[root@jumpserver ~]# for i in 192.168.1.{41..46}

> do

> scp /root/.ssh/id\_rsa $i:/root/.ssh/

> done

拷贝要用的安装包到所有主机：

[root@jumpserver ~]# for i in 192.168.1.{41..46}; do scp /root/数据库软件包/\*redis\* $i:/roo

php-redis-2.2.4.tar.gz

redis-3.2.1.gem

redis-4.0.8.tar.gz

php-redis-2.2.4.tar.gz

## 安装redis

[root@jumpserver ~]# ansible redis -m shell -a 'tar -xf /root/redis-4.0.8.tar.gz'

[root@jumpserver ~]# ansible redis -m shell -a 'cd /root/redis-4.0.8/; make && make install'

## 初始化redis

[root@jumpserver ~]# ansible redis -m shell -a '/root/redis-4.0.8/utils/install\_server.sh'

## 配置redis

[root@jumpserver ~]# ansible redis -m shell -a '/etc/init.d/redis\_6379 stop'

[root@redis-0001 ~]# vim /etc/redis/6379.conf #已192.168.1.41为例，所有都修改

70 bind 192.168.4.41 #不用回环地址

93 port 6341 #修改端口

137 daemonize yes #守护进程方式运行

501 #requirepass 123456 #501 注释掉密码

815 cluster-enabled yes #815 行开启集群

823 cluster-config-file nodes-6341.conf #指定集群信息文件

829 cluster-node-timeout 5000： #829 当前主机与其他主机通讯超时时间5s

[root@jumpserver ~]# ansible redis -m shell -a 'rm -rf /var/lib/redis/6379/dump.rdb'

#不删除此文件无法做集成

[root@jumpserver ~]# ansible redis -m shell -a '/etc/init.d/redis\_6379 restart'

## 配置redis 管理机

[root@jumpserver 数据库软件包]# scp ruby-devel-2.0.0.648-30.el7.x86\_64.rpm 192.168.1.41:/root

### 安装redis-3.2.1.gem

[root@redis-0001 ~]# gem install redis-3.2.1.gem #未安装无法使用

-bash: gem: command not found

[root@redis-0001 ~]# ls

php-redis-2.2.4.tar.gz redis-4.0.8.tar.gz

redis-3.2.1.gem ruby-devel-2.0.0.648-30.el7.x86\_64.rpm

redis-4.0.8

[root@redis-0001 ~]# yum -y install ruby rubygems

##脚本文件是ruby语言编写的,所以要安装解释器

[root@redis-0001 ~]# yum -y install ruby-devel-2.0.0.648-30.el7.x86\_64.rpm

[root@redis-0001 ~]# gem install redis-3.2.1.gem

Successfully installed redis-3.2.1

Parsing documentation for redis-3.2.1

Installing ri documentation for redis-3.2.1

1 gem installed

### 创建命令工具

[root@redis-0001 ~]# cd /root/redis-4.0.8/src/

[root@redis-0001 src]# echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

[root@redis-0001 src]# mkdir /root/bin

[root@redis-0001 src]# cp redis

redisassert.h redis-check-aof.o redis-cli.o

redis-benchmark redis-check-rdb redismodule.h

redis-benchmark.c redis-check-rdb.c redis-sentinel

redis-benchmark.o redis-check-rdb.o redis-server

redis-check-aof redis-cli redis-trib.rb

redis-check-aof.c redis-cli.c

[root@redis-0001 src]# cp redis-trib.rb /root/bin/

[root@redis-0001 src]# chmod +x /root/bin/redis-trib.rb

[root@redis-0001 src]# redis-trib.rb help

## 创建集群

[root@redis-0001 src]# redis-trib.rb create --replicas 1 192.168.1.41:6341 192.168.1.42:6342 192.168.1.43:6343 192.168.1.44:6344 192.168.1.45:6345 192.168.1.46:6346

>>> Creating cluster

>>> Performing hash slots allocation on 6 nodes...

Using 3 masters:

192.168.1.41:6341

192.168.1.42:6342

192.168.1.43:6343

Adding replica 192.168.1.45:6345 to 192.168.1.41:6341

Adding replica 192.168.1.46:6346 to 192.168.1.42:6342

Adding replica 192.168.1.44:6344 to 192.168.1.43:6343

M: b26c7c0a752a07c882c66fd6670f92e961df544e 192.168.1.41:6341

slots:0-5460 (5461 slots) master

M: 7b2f6899ea1f335ef31a7ec4a71a4e87bdede2ef 192.168.1.42:6342

slots:5461-10922 (5462 slots) master

M: f8fc30d368b924543682a62889e21066598b85cb 192.168.1.43:6343

slots:10923-16383 (5461 slots) master

S: e5b5e0b7343c9f497bbc1a8a93423e766e65496f 192.168.1.44:6344

replicates f8fc30d368b924543682a62889e21066598b85cb

S: 58cba5137e8a825dcba21734d6d731fdcda7aa0c 192.168.1.45:6345

replicates b26c7c0a752a07c882c66fd6670f92e961df544e

S: cf099ea2c6872f0f55f067278c4b8271958d906e 192.168.1.46:6346

replicates 7b2f6899ea1f335ef31a7ec4a71a4e87bdede2ef

Can I set the above configuration? (type 'yes' to accept): yes #此处输入yes

……….

[OK] All nodes agree about slots configuration.

>>> Check for open slots...

>>> Check slots coverage...

[OK] All 16384 slots covered.

## 查看集群状态

[root@redis-0001 src]# redis-cli -h 192.168.1.41 -p 6341 #登陆

192.168.1.41:6341> cluster info #登陆集群中任意节点都可查询

cluster\_state:ok

cluster\_slots\_assigned:16384

cluster\_slots\_ok:16384

cluster\_slots\_pfail:0

cluster\_slots\_fail:0

cluster\_known\_nodes:6

cluster\_size:3

cluster\_current\_epoch:6

cluster\_my\_epoch:1

cluster\_stats\_messages\_ping\_sent:324

cluster\_stats\_messages\_pong\_sent:328

cluster\_stats\_messages\_sent:652

cluster\_stats\_messages\_ping\_received:323

cluster\_stats\_messages\_pong\_received:324

cluster\_stats\_messages\_meet\_received:5

cluster\_stats\_messages\_received:652

## 查看集群节点

192.168.1.41:6341> cluster nodes

b26c7c0a752a07c882c66fd6670f92e961df544e 192.168.1.41:6341@16341 myself,master - 0 1554789248000 1 connected 0-5460

f8fc30d368b924543682a62889e21066598b85cb 192.168.1.43:6343@16343 master - 0 1554789248000 3 connected 10923-16383

7b2f6899ea1f335ef31a7ec4a71a4e87bdede2ef 192.168.1.42:6342@16342 master - 0 1554789248700 2 connected 5461-10922

cf099ea2c6872f0f55f067278c4b8271958d906e 192.168.1.46:6346@16346 slave 7b2f6899ea1f335ef31a7ec4a71a4e87bdede2ef 0 1554789247597 6 connected

58cba5137e8a825dcba21734d6d731fdcda7aa0c 192.168.1.45:6345@16345 slave b26c7c0a752a07c882c66fd6670f92e961df544e 0 1554789247597 5 connected

e5b5e0b7343c9f497bbc1a8a93423e766e65496f 192.168.1.44:6344@16344 slave f8fc30d368b924543682a62889e21066598b85cb 0 1554789247000 4 connected

192.168.1.41:6341> info replication

# Replication

role:master

connected\_slaves:1

slave0:ip=192.168.1.45,port=6345,state=online,offset=14196,lag=1

master\_replid:e07854fb685789f799a209e42b951f03df414dd7

master\_replid2:0000000000000000000000000000000000000000

master\_repl\_offset:14196

second\_repl\_offset:-1

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:1

repl\_backlog\_histlen:14196