目录

[1. Keepalived高可用部署步骤 2](#_Toc49522477)

[1.1安装keepalived 2](#_Toc49522478)

[1.2设置开机启动 2](#_Toc49522479)

[1.3日志设置 2](#_Toc49522480)

[1.4防火墙设置 2](#_Toc49522481)

[1.5 Keepalived配置与启动 3](#_Toc49522482)

[1.6 测试keepalived 5](#_Toc49522483)

[1.7 keepalive高可用FAQ 6](#_Toc49522484)

[1.8 keepalived虚拟MAC 6](#_Toc49522485)

[2. 数据库主主热备部署步骤 8](#_Toc49522486)

[2.1安装数据库 8](#_Toc49522487)

[2.2配置文件修改 10](#_Toc49522488)

[2.3创建目录与文件 12](#_Toc49522489)

[2.4设置防火墙并启动数据库 12](#_Toc49522490)

[2.5数据库主主复制配置 13](#_Toc49522491)

[2.5数据库主主复制错误处理 13](#_Toc49522492)

[3. 网络磁盘阵列挂载 15](#_Toc49522493)

[3.1 NFS sever配置 15](#_Toc49522494)

[3.2防火墙设置 16](#_Toc49522495)

[3.3 NFS client配置 17](#_Toc49522496)

[4. 数据库单机多实例部署步骤 17](#_Toc49522497)

[5. Redis高可用部署步骤 19](#_Toc49522498)

[5.1 redis主从配置步骤 19](#_Toc49522499)

[5.2 redis高可用配置步骤 21](#_Toc49522500)

[5.3 redis数据持久化配置步骤 23](#_Toc49522501)

[6. kafka高可用部署方案步骤 26](#_Toc49522502)

[前言 26](#_Toc49522503)

[6.1环境准备 27](#_Toc49522504)

[6.2 zookeeper集群安装 28](#_Toc49522505)

[6.3、kafka集群安装部署 30](#_Toc49522506)

[6.4测验 31](#_Toc49522507)

# Keepalived高可用部署步骤

**说明：**

假设A、B服务器均部署有mysql数据库，且互为主主。此处为A、B服务器部署mysql数据库高可用，当其中一台mysql宕机后，VIP可自动切换至另一台mysql提供服务，实现故障的自动迁移。以下配置仅作为参考，具体IP、端口等相关参数需要根据实际情况修改。

服务器A:10.66.221.211

服务器B:10.66.221.212

虚拟IP:10.66.221.217

注：数据库主主搭建请见第2章节，此处省略

## 1.1安装keepalived

yum install keepalived -y

## 1.2设置开机启动

systemctl enable keepalived

## 1.3日志设置

keepalived部署时，默认将日志打印到系统日志文件/var/log/messages，为了排查问题方便，此处单独为keepalived设置日志文件，配置如下：

(1) 修改/etc/sysconfig/keepalived，把KEEPALIVED\_OPTIONS="-D" 修改为：KEEPALIVED\_OPTIONS="-D -d -S 0"

(2) 在/etc/rsyslog.conf 的末尾添加:local0.\* /var/log/keepalived.log

(3) 重启日志记录服务

systemctl restart rsyslog

## 1.4防火墙设置

在防火墙上开启富策略，放通keepalived广播消息发送地址224.0.0.18(这里放通整个/24网段)，同时开通直连策略，放通vrrp协议

#开通富策略，放通广播消息

firewall-cmd --permanent --add-rich-rule='rule family=ipv4 destination address=224.0.0.18/24 protocol value=ip accept'

#在直连策略中放通vrrp协议

firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 --in-interface ens224 --destination 224.0.0.18 --protocol vrrp -j ACCEPT

firewall-cmd --reload

firewall-cmd --list-all

## 1.5 Keepalived配置与启动

注: keepalived官方手册：<https://www.keepalived.org/manpage.html>

下面以track\_script的方式，实现服务挂起后的vip自动切换

**（1）配置211的keepalived配置文件/etc/keepalived/keepalived.conf（注意标红的部分，需要根据实际情况做相应修改）**

! Configuration File for keepalived

global\_defs {

router\_id 211 #路由器ID,建议通过服务器IP地址最后一位数字进行标识

}

vrrp\_script track\_mysql {

script "/etc/keepalived/shell/mysql\_check.sh" #检测脚本，将返回非0时keepalived会将自身状态置为fault，并迁移vip到备机

interval 1 #脚本执行间隔

timeout 1 #等待脚本返回的时间，超过该时间则认为超时

rise 1 #脚本返回一次0即认为节点恢复正常

fall 1 #脚本返回一次非0即认为节点故障

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0 #网卡名，需要根据实际网卡名进行修改

virtual\_router\_id 217 #虚拟路由器IP，建议使用虚拟IP最后一位数字进行标识

priority 100 #keepalived优先级设置，优先级大的服务器将被选举为master

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.66.221.217 #此处配置虚拟IP地址

}

track\_script {

track\_mysql #此处调用vrrp\_script中定义的track\_mysql

}

}

**（2）配置212的keepalived配置文件/etc/keepalived/keepalived.conf（注意标红的部分，需要根据实际情况做相应修改）**

! Configuration File for keepalived

global\_defs {

router\_id 212 #路由器ID,建议通过服务器IP地址最后一位数字进行标识

}

vrrp\_script track\_mysql {

script "/etc/keepalived/shell/mysql\_check.sh" #检测脚本，将返回非0时keepalived会将自身状态置为fault，并迁移vip到备机

interval 1 #脚本执行间隔

timeout 1 #等待脚本返回的时间，超过该时间则认为超时

rise 1 #脚本返回一次0即认为节点恢复正常

fall 1 #脚本返回一次非0即认为节点故障

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0 #网卡名，需要根据实际网卡名进行修改

virtual\_router\_id 217 #虚拟路由器IP，建议使用虚拟IP最后一位数字进行标识

priority 50 #keepalived优先级设置，优先级小的服务器将成为backup

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.66.221.217 #此处配置虚拟IP地址

}

track\_script {

track\_mysql #此处调用vrrp\_script中定义的track\_mysql

}

}

1. **创建/etc/keepalived/shell/mysql\_check.sh，该脚本内容如下：**

#!/bin/bash

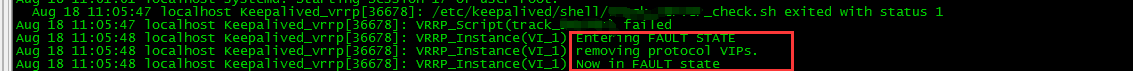
netstat -anop | grep -w 3306 | grep -ai listen > /dev/null 2>&1

1. **修改完配置文件以后，启动keepalived：**

systemctl start keepalived

## 1.6 测试keepalived

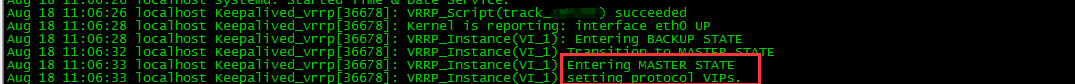
在211服务器上，手动停止mysql，模拟mysql数据库宕机，查看keepalived日志，可见keepalived检测到mysql\_check.sh返回非0，将keepalived置为fault，并将vip转移到212



在备用服务器212上，可以看到keepalived状态变为master:



手动将211上的mysql启动，可看到211的keepalived又切换为原来的master状态



## 1.7 keepalive高可用FAQ

1. 虚拟IP在每次重启以后可以正常访问服务，但是一段时间后，通过vip（虚拟IP）访问服务会出现异常。直接使用rip（真实IP）访问正常。同时在服务器本机通过vip访问服务也正常。重启keepalive后，访问业务也正常。

分析：出现这种情况可能是由于内网存在一个和vip相同的服务器，由于ip冲突导致。默认情况下，vip使用真实可通过查看arp路由表

## 1.8 keepalived虚拟MAC

个别网络环境，必须保证ip和mac地址一一对应，这种情况下，vip将不能共享物理网卡的mac地址。针对这种情况，可以启用vmac，配置很简单，在配置文件中加入如下命令（标红）即可：

! Configuration File for keepalived

global\_defs {

router\_id 211 #路由器ID,建议通过服务器IP地址最后一位数字进行标识

}

vrrp\_script track\_mysql {

script "/etc/keepalived/shell/mysql\_check.sh" #检测脚本，将返回非0时keepalived会将自身状态置为fault，并迁移vip到备机

interval 1 #脚本执行间隔

timeout 1 #等待脚本返回的时间，超过该时间则认为超时

rise 1 #脚本返回一次0即认为节点恢复正常

fall 1 #脚本返回一次非0即认为节点故障

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0 #网卡名，需要根据实际网卡名进行修改

virtual\_router\_id 217 #虚拟路由器IP，建议使用虚拟IP最后一位数字进行标识

priority 100 #keepalived优先级设置，优先级大的服务器将被选举为master

use\_vmac

vmac\_xmit\_base

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.66.221.217 #此处配置虚拟IP地址

}

track\_script {

track\_mysql #此处调用vrrp\_script中定义的track\_mysql

}

}

说明：

1. The 'use\_vmac' keyword will drive keepalived code to create a macvlan interface named 'vrrp.217' (default internal paradigm is vrrp.{virtual\_router\_id}, you can override this naming by giving an argument to 'use\_vmac' keyword, eg: use\_vmac vrrp217).
2. The ‘vmac\_xmit\_base’ keyword will transmit VRRP adverts over physical interface

# 数据库主主热备部署步骤

数据库A ：192.168.1.1  
数据库B ：192.168.1.2

## 2.1安装数据库

下载Mysql 5.6到/myapp目录下

cd /myapp

wget https://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.31-linux-glibc2.5-x86\_64.tar.gz

解压、删除安装包并重命名解压后的文件夹为mysql

tar -zxvf mysql-5.6.31-linux-glibc2.5-x86\_64.tar.gz

rm -f mysql-5.6.31-linux-glibc2.5-x86\_64.tar.gz

mv mysql-5.6.31-linux-glibc2.5-x86\_64 mysql

添加用户组

groupadd mysql

useradd -r -g mysql mysql

进入mysql目录更改权限

cd mysql/

chown -R mysql:mysql ./

先备份/etc/my.cnf，再执行安装脚本初始化数据库

mv /etc/my.cnf /etc/my\_bak.cnf

./scripts/mysql\_install\_db --user=mysql

注意：

如果提示缺少perl模块,则需要使用yum -y install perl perl-devel安装该模块； 如果缺少Data:Dumper，则使用yum -y install autoconf 进行安装

安装完成后修改当前目录拥有者为root用户，修改data目录拥有者为mysql

chown -R root:root ./

chown -R mysql:mysql data

打开启动脚本，修改basedir为/myapp/mysql

vi ./support-files/mysql.server

basedir=/myapp/mysql

启动mysql

./support-files/mysql.server start

备注：

如果提示启动失败，则在my.cnf配置文件中的 [mysqld\_safe] 下添加log-error = /myapp/mysql/error.log，使能mysql错误日志。手动创建该日志并添加权限。然后使用上述命令重新启动mysql服务，查看error.log文件，修改相应的错误。

如果提示日志中提示找不到mysql.user表，则使用./scripts/mysql\_install\_db --user=mysql重新初始化一次数据库

更改密码、登录mysql并修改其他用户的密码

./bin/mysqladmin -u root -h localhost.localdomain password '123456789'

./bin/mysql -h127.0.0.1 -uroot -p123456789

mysql> update mysql.user set password=password('123456789') where user='root';

mysql> flush privileges;

将mysql加入service系统服务

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

chkconfig --add mysqld

chkconfig mysqld on

修改my.cnf，在最后面添加如下内容

character-set-server = utf8 #修改数据库编码为UTF8

lower\_case\_table\_names = 1 #设置大小写不敏感

max\_allowed\_packet = 100M #调整Server接受的数据包大小为100M

重启服务并确认状态

service mysqld restart

service mysqld status

## 2.2配置文件修改

修改数据库的配置文件如下：

[mysqld]

port=13306

#设置客户端最大连接数。

max\_connections=1024

character\_set\_server=utf8

max\_allowed\_packet=100M

#关闭大小写敏感

lower\_case\_table\_names=1

character-set-server=utf8

open\_files\_limit=65536

tmp\_table\_size=1024M

max\_heap\_table\_size=1024M

init\_connect='SET NAMES utf8'

basedir=/myapp/mariadb

datadir=/myapp/data/mariadb

socket=/myapp/data/mariadb/mariadb\_13306.sock

log-error=/myapp/mariadb/log/mysqld.log

#自增值偏移量设置，避免双机热备时出现自增ID冲突。

#需要将数据库A的偏移设置为1，数据库B的偏移设置为2

auto\_increment\_offset=1

auto\_increment\_increment=2

#主主热备中服务器ID用于唯一标识一台数据库。两台数据库必须设置不同

server\_id=117

log\_bin=mysql\_bin.log

#binlog保存时间根据服务写数据库速率以及磁盘大小做设置。此处设置为7天

expire\_logs\_days=7

binlog-ignore-db=mysql

relay-log=relay-log.log

binlog\_format=ROW

sync\_master\_info=1

master\_verify\_checksum=1

slave\_sql\_verify\_checksum=1

binlog\_rows\_query\_log\_events=1

#设置备用数据库在bin-log发生错误时默认跳过的错误类型

slave\_skip\_errors=1146,1008,1062,1032

#以下两个参数控制备数据库的同步速度

#innodb\_flush\_log\_at\_trx\_commit=2，表示MySQL会每秒执行一次flush操作

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=0

sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES

[mysqld\_safe]

log-error=/myapp/mariadb/log/error.log

[mysqladmin]

socket=/myapp/data/mariadb/mariadb\_13306.sock

将数据库的配置文件复制到另外一台数据库上并修改如下配置：

auto\_increment\_offset = 2

server\_id=118

## 2.3创建目录与文件

**根据2.2中的配置文件，需要在数据库中创建如下目录与文件：**

1. 创建log目录以及log文件（如果2.1中已创建，可忽略）

cd /myapp/mysql

mkdir log

cd log

touch error.log

touch mysqld.log

1. 创建套接字目录并修改目录所有者

cd /myapp/mysql

mkdir var

chown mysql:mysql ./var

## 2.4设置防火墙并启动数据库

在两台数据库的防火墙上放通3306端口，使数据库之间可以互相访问

firewall-cmd --permanent --add-rich-rule="rule family="ipv4" source address="10.66.221.211" port protocol="tcp" port="3306" accept"

firewall-cmd --reload

设置好防火墙后，重启数据库：

systemctl restart mysqld

## 2.5数据库主主复制配置

数据库A与B互相授权：

（1）在数据库A（192.168.1.1）上

GRANT REPLICATION SLAVE ON \*.\* TO 'backup'@'192.168.1.2' IDENTIFIED BY '123456';  
flush privileges;

（2）在数据库B（192.168.1.2）上

GRANT REPLICATION SLAVE ON \*.\* TO 'backup'@'192.168.1.1' IDENTIFIED BY '123456';  
flush privileges;

设置主数据库同步信息：

1. 在数据库A（192.168.1.1）上

change master to master\_host='192.168.1.2',master\_user='backup',master\_port=13306,master\_password='123456',master\_log\_file='mysql\_bin.000001',master\_log\_pos=0;

（2）在数据库B（192.168.1.2）上

change master to master\_host='192.168.1.1',master\_user='backup',master\_port=13306,master\_password='123456',master\_log\_file='mysql\_bin.000001',master\_log\_pos=0;

在主备数据库A与B中启动备库同步

start slave;

查看备库同步情况：

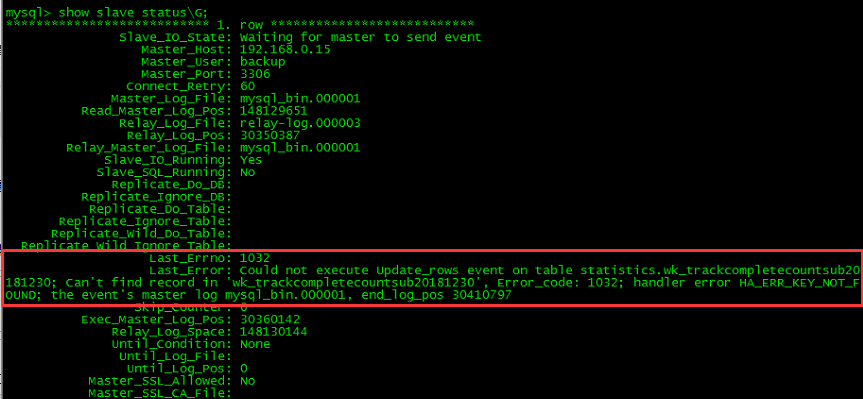
show slave status\G

如果主从复制建立成功，可看到如下关键字：

 Slave\_IO\_Running: Yes  
 Slave\_SQL\_Running: Yes

## 2.5数据库主主复制错误处理

1. 查看复制状态：show slave status\G;



发现数据库提示出现1032错误（在主主同步的测试环境，由于业务侧没有遵循同一时间只写一个点的原则，造成A库上删除了一条数据，B库上在同时更新这条数据。

由于异步和网络延时，B的更新event先到达A端执行，造成A端找不到这条记录，故SQL\_THREAD报错1032，主从同步停止。）

1. 解决方法1：跳过错误event，命令如下：

stop slave;

set global sql\_slave\_skip\_counter=1;

start slave;

1. 解决方法2：跳过所有1032错误，直接修改my.cnf配置文件后重启数据库：

[mysqld]

slave-skip-errors = 1032

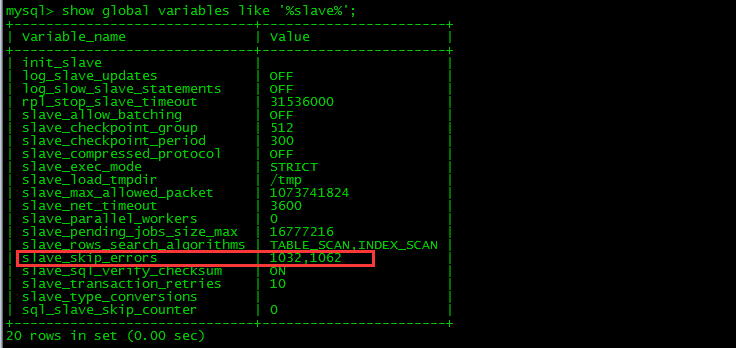
注意：

如需添加跳过错误的类型，直接在该参数后添加错误代码并以逗号分隔。如：

slave-skip-errors = 1032,1062

重启数据库后，需要确认设置是否生效：

show global variables like ‘%slave%’;



# 网络磁盘阵列挂载

目前常用的磁盘阵列主要分为两种：

1. NAS专业存储服务器
2. 挂载在linux系统下的磁盘阵列服务器

对于第一种NAS服务器，配置有定制的操作系统，NFS server一般已经配置完成。只需在NFS client端执行相关的挂载操作即可

对于第二种挂载在linux下的磁盘阵列，需要单独配置NFS server后，再在NFS client端进行挂载使用

下面分别说明NFS server和NFS client的配置方法

## NFS sever配置

说明：

NFS 服务器IP：192.168.1.1

NFS 客户端IP：192.168.1.2

1. 安装NFS服务器

（1）查看系统是否已安装NFS（如果已安装，可略过本步骤）

rpm -qa | grep nfs

rpm -qa | grep rpcbind

（2）安装NFS

yum -y install nfs-utils rpcbind

1. 服务端配置
2. 在NFS服务端上创建共享目录/myapp/data并设置权限

mkdir -p /myapp/data

chmod 666 /myapp/data

1. 编辑export文件，设置共享目录

vi /etc/exports

/myapp/data 192.168.0.\*(rw,no\_root\_squash,async)

1. 使配置生效

exportfs -r

1. 启动NFS服务器

systemctl start rpcbind

systemctl start nfs

1. 查看目录是否共享成功

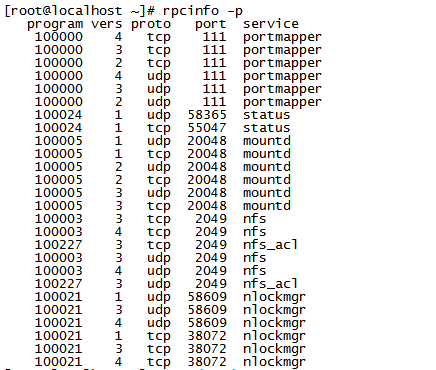
showmount -e localhost

## 3.2防火墙设置

1. 查看NFS server使用的端口

rpcinfo -p

/\*8+963

.774

从以上返回可以看出，nfs server使用了多个端口。除了nfs和portmapper使用固定端口以外，其他端口均为随机端口，每次重启nfs server以后，都会发生变化。如果防火墙使用端口来配置访问策略，则需要固定这部分端口：

vim /etc/sysconfig/nfs

编辑端口：

RQUOTAD\_PORT=30001

LOCKD\_TCPPORT=30002

LOCKD\_UDPPORT=30002

MOUNTD\_PORT=30003

STATD\_PORT=30004

如果不通过端口配置策略，则可以忽略上述配置

1. 防火墙开启nfs相关服务，供nfs client连接：

firewall-cmd --zone=public --add-service mountd --permanent

firewall-cmd --zone=public --add-service rpc-bind --permanent

firewall-cmd --zone=public --add-service nfs --permanent

firewall-cmd --reload

注意：如果nfs-server是云服务器，必须开启UDP 32768 - 65535段。因为数据是通过UDP传输的，光开启TCP无用

## 3.3 NFS client配置

1.安装NFS客户端

yum -y install nfs-utils

1. 查看nfs server的共享目录信息

showmount -e 192.168.1.1

1. 挂载网络磁盘阵列

mount -t nfs 192.168.1.1:/myapp/data /myapp/data

注意：

在挂载nfs的时候，如果挂载不上，可以尝试指定nfs版本为version3，并设置为软挂载，具体命令为：  
mount -t nfs -o vers=3,rw,soft,timeo=30,retry=3 192.168.1.1:/myapp/data /myapp/data

1. 查看挂载是否成功

df -h

# 数据库单机多实例部署步骤

为了充分利用服务器资源，可在单个物理服务器上部署多个mysql实例。

注意：如果不是必须，不建议部署单服务器多mysql实例。因为mysql之间可能由于磁盘IO和CPU、内存等资源占用而互相影响，反而降低mysql的性能

部署步骤：

1. 将已部署的mysql复制一份，修改目录名为mysql\_3307，同时将权限改为755

cp -r /myapp/mysql /myapp/mysql\_3307

chmod -R 755 /myapp/mysql\_3307

1. 将全局配置文件备份:

mv /etc/my.cnf /etc/my.cnf\_YYYYMMDD

注意：mysql5.6 读取配置文件的顺序为

1）/etc/my.cnf

2）basedir/my.cnf

3）datadir/my.cnf

4）--defaults-extra-file    #在读取全局配置文件之后，读取用户配置文件(~/.my.cnf)之前，读取extra指定的参数文件

5）~/.my.cnf      #家目录下面的隐藏文件，my.cnf前面的点，说明my.cnf是隐藏文件

假设4个配置文件都存在，同时使用--defaults-extra-file指定了参数文件，如果这时有一个 "参数变量"在5个配置文件中都出现了，那么后面的配置文件中的参数变量值会覆盖前面配置文件中的参数变量值，就是说会使用~/.my.cnf中设置的值。

1. 修改mysql\_3307的配置文件

vi /myapp/mysql\_3307/my.cnf

[client]

port = 3307

socket = /myapp/mysql\_3307/var/mysql\_3307.sock

[mysqld]

port = 3307

socket = /myapp/mysql\_3307/var/mysql\_3307.sock

[mysqld\_safe]

log-error = /myapp/mysql\_3307/error.log

注意：如果需要开启主备数据库，则还需要修改server\_id等必改参数

1. 修改mysql.server文件中的基础目录和数据目录

vi /myapp/mysql\_3307/support-files/mysql.server

basedir=/myapp/mysql\_3307

datadir=/myapp/mysql\_3307/data

1. 修改data和var目录用户为mysql

chown -R mysql:mysql /myapp/mysql\_3307/data

chown -R mysql:mysql /myapp/mysql\_3307/var

1. 启动mysql\_3307

/myapp/mysql\_3307/support-files/mysql.server start

注意：如果无法启动，可通过错误日志 /myapp/mysql\_3307/error.log进行定位。

1. 访问mysql\_3307

/myapp/mysql\_3307/bin/mysql -uroot -p123456789 -S /myapp/mysql\_3307/var/mysql\_3307.sock

# Redis高可用部署步骤

## redis主从配置步骤

### 环境准备

这里使用三台服务器，每台服务器上开启一个redis-server服务，redis-server端口为8000（修改默认端口是安全的第一步）。

redis-server说明服务器规划如下：

192.168.68.110:8000 //主redis

192.168.68.111:8000 //从redis

192.168.68.112:8000 //从redis

### 搭建redis主从

先在三台机器上安装redis，具体步骤可参考环境安装包使用说明：<http://192.168.150.95:3118/article/5f18fc9778059e2299bc5088>

安装完成后复制redis提供的默认配置文件

cp /etc/redis.conf  /usr/local/redis

修改配置文件

cd /usr/local/redis

vim redis.conf

主节点192.168.68.110上的配置

port  8000

daemonize  yes

bind  192.168.68.110

requirepass 123456

pidfile   /var/run/redis-8000.pid

logfile   /var/log/redis/redis-8000.log

从节点192.168.68.111，192.168.68.112上的配置

port  8000

daemonize  yes

bind  192.168.68.111   #另一台机器配置为192.168.68.112

requirepass 123456

masterauth 123456

pidfile   /var/run/redis-8000.pid

logfile   /var/log/redis/redis-8000.log

slaveof  192.168.68.110 8000

注意：redis不会自动创建目录，所以在启动之前需要手动创建目录/var/log/redis

在三台服务器的防火墙上放通8000端口：

firewall-cmd --zone=public --add-port=8000/tcp --permanent

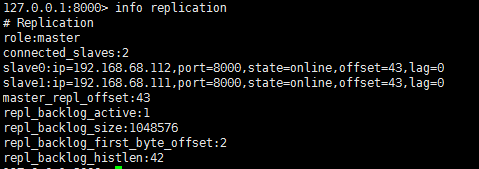
firewall-cmd --reload

说明：如防火墙已关闭，可不用执行上述命令

启动三台机器上的redis

redis-server  /usr/local/redis/redis.conf

三个redis服务启动完毕后，进入主redis的命令行，执行info replication查看当前主从配置



### 主从复制测试

主节点添加数据

IMG_256

从节点获取数据

IMG_257

## redis高可用配置步骤

按照5.1配置完主从同步，当主服务器宕机后，需要手动把一台从服务器切换为主服务器，这就需要人工干预，相对较耗时，而且还会造成一段时间内服务不可用。为解决此问题，redis提供哨兵模式（redis sentinel），实现故障发现、故障自动转移、配置中心、客户端通知等功能。

### 环境准备

redis sentinel随redis实例一起搭建，因此直接在5.1.1的环境基础上，为每台服务器在开启一个redis sentinel实例即可，端口采用6800。规划如下：

192.168.68.110:6800

192.168.68.111:6800

192.168.68.112:6800

### redis sentinel搭建

说明：三台服务器均需要按如下说明配置

复制默认的配置文件并进行编辑

cp /etc/redis-sentinel.conf  /usr/local/redis/

vim /usr/local/redis/redis-sentinel.conf

修改配置文件内容如下

protected-mode no

daemonize yes

port  6800

sentinel monitor mymaster 192.168.68.110 8000 1

sentinel auth-pass mymaster 123456

#5秒内master6800没有响应，就认为SDOWN

sentinel down-after-milliseconds mymaster 5000

sentinel failover-timeout mymaster 15000

logfile  /var/log/redis/redis-sentinel-6800.log

pidfile  /var/run/redis-sentinel-6800.pid

在三台服务器的防火墙上放通6800端口：

firewall-cmd --zone=public --add-port=6800/tcp --permanent

firewall-cmd --reload

说明：如防火墙已关闭，可不用执行上述命令

启动redis-sentinel

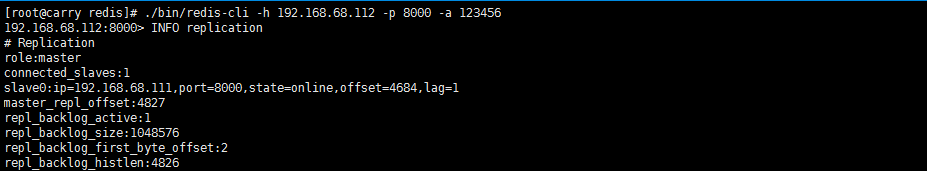
redis-sentinel  /usr/local/redis/redis-sentinel.conf

### 故障转移测试

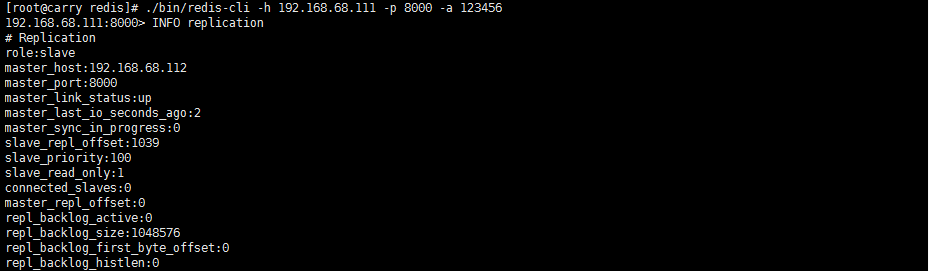
停掉主节点

IMG_256

查看redis集群信息，可以看到，从节点192.168.68.112升级为主节点



在从节点192.168.68.111上，可以看到主节点变更为192.168.68.112



## redis数据持久化配置步骤

### redis持久化模式

redis提供了两种持久化方式，分别是：

1）RDB定时快照方式(snapshot)： RDB 将数据库的快照（snapshot）以二进制的方式保存到磁盘中。

2）AOF基于语句追加文件的方式：以协议文本的方式，将所有对数据库进行过写入的命令（及其参数）记录到 AOF 文件，以此达到记录数据库状态的目的。

### 持久化模式的优缺点对比

1. **RDB的优缺点**

**1）优点：**

I. RDB是一个非常紧凑的文件，它保存了某个时间点的数据集，非常适用于数据集的备份，这样即使出了问题你也可以根据需求恢复到不同版本的数据集。

Ii. RDB是一个紧凑的单一文件，很方便传送到另一个远端数据中心或者亚马逊的S3（可能加密），非常适用于灾难恢复。

Iii. RDB在保存RDB文件时父进程唯一需要做的就是fork出一个子进程，接下来的工作全部由子进程来做，父进程不需要再做其他IO操作，所以RDB持久化方式可以最大化redis的性能。

与AOF相比，在恢复大的数据集的时候，RDB方式会更快一些。

**2）缺点：**

I. 如果希望在redis意外停止工作的情况下丢失的数据最少的话，那么RDB不适合

Ii. RDB需要经常fork子进程来保存数据集到硬盘上，当数据集比较大的时候，fork的过程是非常耗时的，可能会导致redis在一些毫秒级内不能相应客户端的请求。

1. **AOF的优缺点**

**1）优点：**

I. 使用AOF会让你的redis更加耐久

Ii. AOF文件是一个只进行追加的日志文件，所以不需要写入seek，即使由于某些原因（磁盘空间已满，写的过程中宕机等）未执行完整的写入命令，你也可使用redis-check-aof工具修复这些问题。

Iii. redis可以在aof文件体积变得过大时，自动的在后台对aof进行重写。

Iv. aof文件有序地保存了对数据库执行的所有写入操作，这些写入操作以redis协议的格式保存，因此aof文件的内容非常容易被人读懂，对文件进行分析（parse）也很轻松。

**2）缺点：**

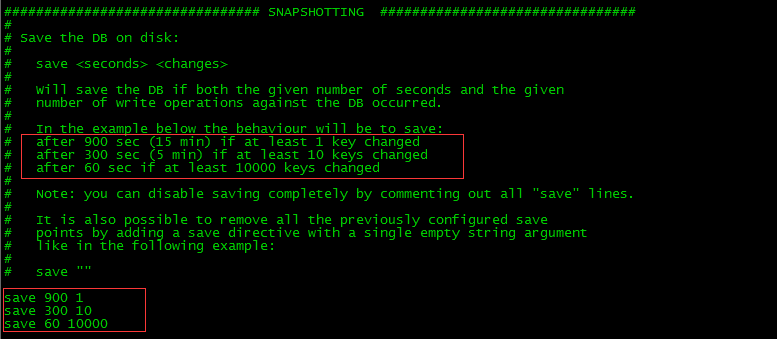
I. 对于相同的数据集来说，aof文件的体积通常要大于rdb文件的体积。

Ii. 根据所使用的fsync策略，aof的速度可能会慢于rdb

1. **如何选择使用哪种持久化方式**
2. 一般来说，如果想达到足以媲美PostgreSQL的数据安全性，你应该同时使用两种持久化功能。
3. 如果非常关心你的数据，但仍然可以承受数分钟以内的数据丢失，那么你可以只使用RDB持久化。
4. 不推荐只是用AOF持久化，因为定时生成RDB快照（snapshot）非常便于进行数据库备份，并且RDB恢复数据集的速度也要比AOF恢复的速度要快。

### RDB配置步骤

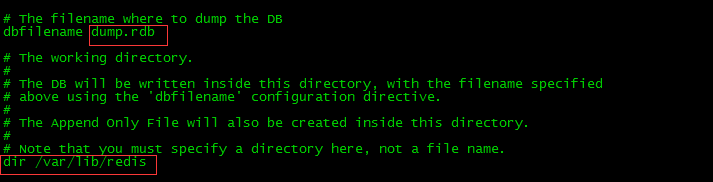
默认情况下，redis 将数据库快照保存在名字为dump.rdb的二进制文件中，可以在redis.conf配置文件中修改持久化信息。



save 900 1 //表示在900秒内，至少更新了1条数据。Redis就将数据持久化到硬盘  
save 300 10 //表示在300秒内，至少更新了10条数据，Redis就会触发将数据持久化到硬盘  
save 60 10000 //表示60秒内，至少更新了10000条数据，Redis就会触发将数据持久化到硬盘

以上默认配置可根据需要修改，修改完并重启redis生效，redis将按照配置策略自动生成RDB备份文件。

默认的rdb文件名是dump.rdb，文件路径是/var/lib/redis，可以根据需要修改



RDB文件是默认是压缩过的，可以通过配置rdbcompression参数来禁用压缩。

如果没有触发自动快照，需要对redis执行手动快照操作，save和bgsave都可以执行手动快照，但是两者有区别：前者是由主进程进行快照，会阻塞其他请求，后者是通过fork子进程进行快照。

redis-cli -h<IP> -p<port> //登录redis

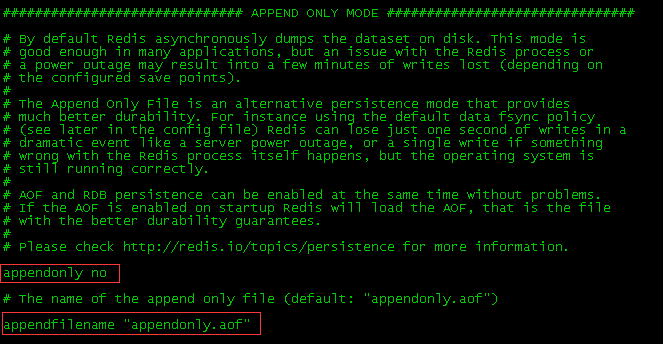
IP:port> auth <password> //进行密码认证

IP:port> save //save方式进行手动快照

IP:port> bgsave //bgsave方式进行手动快照

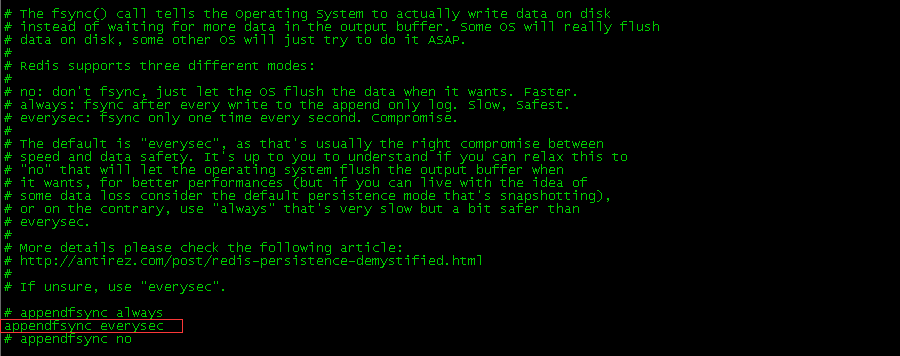
### AOF配置步骤

在redis.conf配置文件中设置appendonly为yes并重启redis即可。默认AOF未打开。



AOF默认的存储路径与RDB相同（dir /var/lib/redis），文件改名默认为appendonly.aof，可通过修改appendfilename参数进行调整

开启了AOF以后，默认情况下会每秒进行同步，可根据需求进行调整：



appendfsync always每次都同步（最安全但是最慢）

appendfsync everysec每秒同步（默认的同步策略）

appendfsync no 不主动同步，由操作系统来决定（最快但是不安全）

# kafka高可用部署方案步骤

## 前言

**Kafka高可用方案实现原理**：kafka 在0.8的版本以后，提供了HA机制，就是replication副本机制。每个partition的数据都会同步到其他机器上，形成自己的多个replica副本。然后所有replica会选举一个leader出来，那么生产和消费都跟这个leader打交道，然后其他replica就是follower。写的时候，leader会负责把数据同步到所有follower上去，读的时候就直接读leader上数据即可。只能读写leader！！！很简单，要是你可以随意读写每个follower，那么就要care数据一致性的问题，系统复杂度太高，很容易出问题。kafka会均匀的将一个partition的所有replica分布在不同的机器上，这样才可以提高容错性。

kafka的这种机制，就是所谓的高可用性了，因为如果某个broker宕机了，也没事儿，因为那个broker上面的partition在其他机器上都有副本的，那么此时会重新选举一个新的leader出来，大家继续读写那个新的leader即可。这就是所谓的高可用性了。

## 6.1环境准备

### 6.1.1集群环境

|  |  |  |  |
| --- | --- | --- | --- |
| 系统版本 | Hostname | IP | 角色 |
| Centos7.8 | Kafka1 | 192.168.154.48 | Zookeeper+kafka |
| Centos7.8 | Kafka2 | 192.168.154.58 | Zookeeper+kafka |
| Centos7.8 | Kafka3 | 192.168.154.68 | Zookeeper+kafka |

### 6.1.2软件版本

|  |  |
| --- | --- |
| Jdk | openjdk version "1.8.0\_131" |
| zookeeper | zookeeper-3.4.14 |
| kafka | kafka\_2.11-2.3.0 |

### 6.1.3 部署前的准备

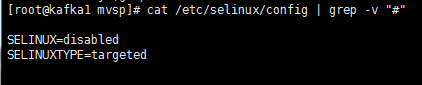
关闭防火墙或开放端口，关闭selinux(生产环境按需关闭或打开)

Vim /etc/selinux/config

selinux=disabled

临时关闭

setenforce 0



防火墙开启富策略或者开放端口访问

firewall-cmd --permanent --add-port=12181/tcp

firewall-cmd --permanent --add-port=2888/tcp

firewall-cmd --permanent --add-port=3888/tcp

firewall-cmd --permanent --add-port=19092/tcp

firewall-cmd --reload

firewall-cmd --list-all

PS：后期三台机器上装的kafka和zookeeper必须保持一致

## 6.2 zookeeper集群安装

### 6.2.1安装jvmjava依赖环境(三台)

Java环境安装可参考S17业务环境安装说明，具体步骤可参考环境安装包使用说明：<http://192.168.150.95:3118/article/5f18fc9778059e2299bc5088>

也可自行安装，自行安装如下：

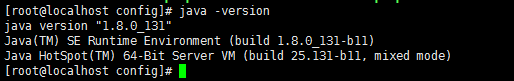
2. 进入解压目录的script目录下， 执行jdk的安装默认不输出参数安装Java：

host-a# cd env\_dependency\_normal\_mariadb/script

host-a# ./install\_env.sh

查看当前Java版本

java –version



### 6.2.2安装配置zookeeper集群

需要使用单独的zookeeper安装包，不要使用kafka自带安装包下载zookeeper包到指定目录

cd /myapp

wget <https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/zookeeper-3.4.14/zookeeper-3.4.14.tar.gz>

解压包到指定目录

tar xf zookeeper-3.4.14.tar.gz -C /myapp

cd /myapp

ln -sv zookeeper-3.4.14 zookeeper #软连接关联zookeeper

拷贝配置文件至配置目录

cd /myapp/zookeeper/conf

cp  zoo\_sample.cfg zoo.cfg

修改配置文件

vim zoo.cfg

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/myapp/data/zk\_data

dataLogDir=/myapp/zookeeper/logs

clientPort=12181

maxClientCnxns=60

autopurge.snapRetainCount=3

autopurge.purgeInterval=1

server.1=192.168.154.48:2888:3888

server.2=192.168.154.58:2888:3888

server.3=192.168.154.68:2888:3888

mkdir -p /myapp/data/zk\_data

mkdir /myapp/zookeeper/logs

创建ServerID标识

在ZooKeeper集群中除配置文件外，还需要配置一个myid文件，以唯一标志zookeeper，三台zookeeper的标识必须不一样。

Kafka1

[root@kafka1 /]# echo '1' > /myapp/data/zk\_data/myid

Kafka2

[root@kafka2 /]# echo '2' > /myapp/data/zk\_data/myid

Kafka3

[root@kafka3 /]# echo '3' > /myapp/data/zk\_data/myid

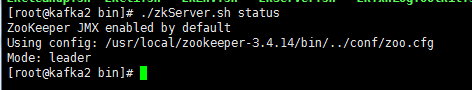
启动ZK集群并查看 三台ZK节点全部启动

cd /myapp/zookeeper/bin

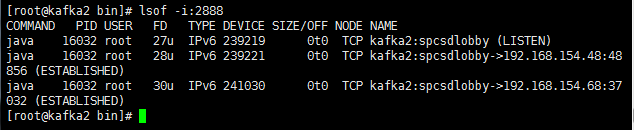
./zkServer.sh start

查看群集状态

./zkServer.sh status



查看ZK端口监听情况 losf -i:port



其他两台以同样的方式查看。

## 6.3 kafka集群安装部署

### 6.3.1安装kafka

先在三台机器上分别安装kafka，具体步骤可参考安装包使用说明：<http://192.168.150.95:3118/article/5f18fc9778059e2299bc5088>

以及S17\_V2.0.5平台安装部署手册中关于环境部分的安装：

<https://docs.qq.com/doc/DYndia1BXb0l2cWlQ>

安装包取包地址：

ftp://192.168.150.124:30021/Test/env/normal/v1.0.0\_mariadb/env\_dependency\_normal\_mariadb.tar.gz（账号信息：test/3edc4RFVtest@123）

### 6.3.2修改kafka配置（主机kafka1）

cd /myapp/kafka/config

cp –a server.properties server.properties\_bak #备份server配置

vim server.properties

num.partitions=3

broker.id=1 #每一台机器都不一样，唯一标识，第二台机器就是2，第三台是3

listeners= PLAINTEXT://192.168.154.48:19092 #修改为本机IP地址

zookeeper.connect=192.168.154.48:12181,192.168.154.58:12181,192.168.154.68:12181

zookeeper.connection.timeout.ms=600000 #考虑到分布式跨机连接的问题，修改zookeeper链接超时选项，以避免超时启动不起来kafka

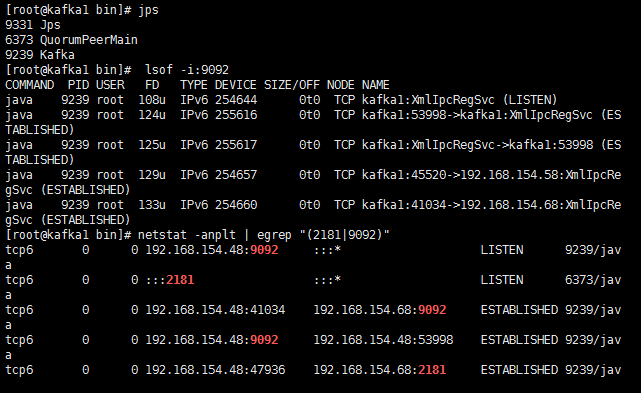
其他原生配置保持不变

启动Kafka 三台Kafka\_node全部启动，这里注意下先启动zookeeper，在启动kafka；先关闭kafka，再关闭zookeeper。

/myapp/kafka/bin/kafka-server-start.sh -daemon /myapp/kafka/config/server.properties

#查看进程连接情况

jps -m



至此kafka集群搭建完毕整个环境部署完毕

## 6.4 测验

### 6.4.1创建topic

cd /myapp/kafka/bin

./kafka-topics.sh /kafka-topics.sh --create --zookeeper 192.168.154.48:12181,192.168.154.58:12181,192.168.154.68:12181 --replication-factor 3 --partitions 3 --topic test

#–replication-factor 3 复制三份，高可用场景必须启用该参数

#–partitions 3 创建三个个分区

#–topic test 主题为test

### 6.4.2创建一个producter

./kafka-console-producer.sh --broker-list 192.168.154.48:19092,192.168.154.58:19092,192.168.154.68:19092 --topic test

### 6.4.3创建一个consumer

./kafka-console-consumer.sh --bootstrap-server 192.168.154.48:19092,192.168.154.58:19092,192.168.154.68:19092 --from-beginning --topic test

### 6.4.4查看topic

./kafka-topics.sh --bootstrap-server 192.168.154.48:19092,192.168.154.58:19092,192.168.154.68:19092 --describe --topic test

Topic: test PartitionCount: 3 ReplicationFactor: 3 Configs: segment.bytes=1073741824

Topic: test Partition: 0 Leader: 3 Replicas: 3,2,1 Isr: 3,2,1

Topic: test Partition: 1 Leader: 1 Replicas: 1,3,2 Isr: 3,2,1

Topic: test Partition: 2 Leader: 2 Replicas: 2,1,3 Isr: 2,3,1

### 6.4.5 选项解释：

--create：创建新的Topic

--bootstrap-server：指定要哪台Kafka服务器上创建Topic，主机加端口，指定的主机地址一定要和配置文件中的listeners一致

--zookeeper：指定要哪台zookeeper服务器上创建Topic，主机加端口，指定的主机地址一定要和配置文件中的listeners一致

--replication-factor：创建Topic中的每个分区(partition)中的复制因子数量，即为Topic的副本数量，建议和Broker节点数量一致，如果复制因子超出Broker节点将无法创建

--partitions：创建该Topic中的分区(partition)数量

--topic：指定Topic名称

--broker-list：指定使用哪台broker来生产消息

--topic：指定要往哪个Topic中生产消息

消费消息 我们在Kafka\_node1上的Topic生产消息，在Kafka\_node3中消费消息，可以得出，Broker存储过消息后会同步给集群内的其它Broker节点