MySQL的分布式架构

分机器:分库

1.配置SSH

2.防火墙,iptables(阻止端口),selinux(篡改)

```
service iptables stop
```

3.mysql当中都需要开放一个远程用户

```
grant all privileges on *.* to 'hal'@'%'
identified by '123456' with grant option;
```

4.找到[mysqld]模块文件

mysql的主从复制传输的就是一个二进制的文件

做读写分离,分散数据库的压力,也可以实时的备份数据

主服务器把:自己的开放账号,IP地址,端口号,二进制文件的名称,口令给从服务器.

从服务器把主服务器的信息记录在配置项中,配置完成以后,从服务器会去访问主服务器,会把自己的IP,端口,给留在主服务器中.

当主服务器发生动作性的操作,主服务会通知从服务器,从服务从新连接主服务器,把binlog文件拷贝过来,执行binlog当中的语句.

当从服务器中发生动作性的操作以后,主服务器不会受到影响,其它的从服务器也不会受到影响

1.文件配置的过程

- 1.先找到mysql的配置文件

```
找到[mysqld]模块
查看到数据文件的存储位置  例如/data/mysql

进入到数据存储目录中查看log日志的文件前缀

配置log-bin=日志的文件前缀
配置server-id=ip地址不加上点 (备注:不能重复)
配置的是黑洞引擎log-slave-updates=1(备注:将二进制文件同步)(级联复制)

#master必须要配置的
binlog-do-db=test   #需要同步的数据库
binlog-ignore-db=mysql  #避免同步的数据库
#slave必须要配置的
replicate-do-db=test   #需要复制的数据库
replicate-ignore-db=mysql   #避免复制mysql库
```

- 2.重启服务器

```
service mysqld restart

systemctl stop mysqld.service
systemctl start mysqld.service
```

- 3.进入到终端

```
#主服务器
show master status\G   #显示当前的主服务器状态
*************************** 1. row
***************************
             File: mysql-bin.000059
 #从服务器拷贝当前的文件
         Position: 154
           #对接口令
     Binlog_Do_DB:
      #同步哪个数据中的数据,值为空复制所有的库
 Binlog_Ignore_DB:
  #避免同步那个数据中的数据
Executed_Gtid_Set:
```

```
#从服务器
stop slave;  #停止从服务器的运行
#配置项
change master to
master_host='10.11.52.111' ,
master_user='hal',
master_password='123456',
master_log_file='mysql-bin.000022',
```

```
master_log_pos=3064;
#开启从服务
start slave;
#查看从服务器的状态
show slave status\G;
######下个两个配置项必须是yes##########
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

```
#innodb引擎配置
################################服务器
#################
log-bin=mysql-bin
#将主服务器的log文件更新到mysql-bin中
log-slave-updates=1
server-id = 101152106
#master必须要配置的
binlog-do-db=test  #需要同步的数据库
binlog-ignore-db=mysql #避免同步的数据库
########################################
###########

default_storage_engine = InnoDB
```

```
#blackhole引擎配置
################################服务器
#################
log-bin=mysql-bin
#将主服务器的log文件更新到mysql-bin中
log-slave-updates=1
```

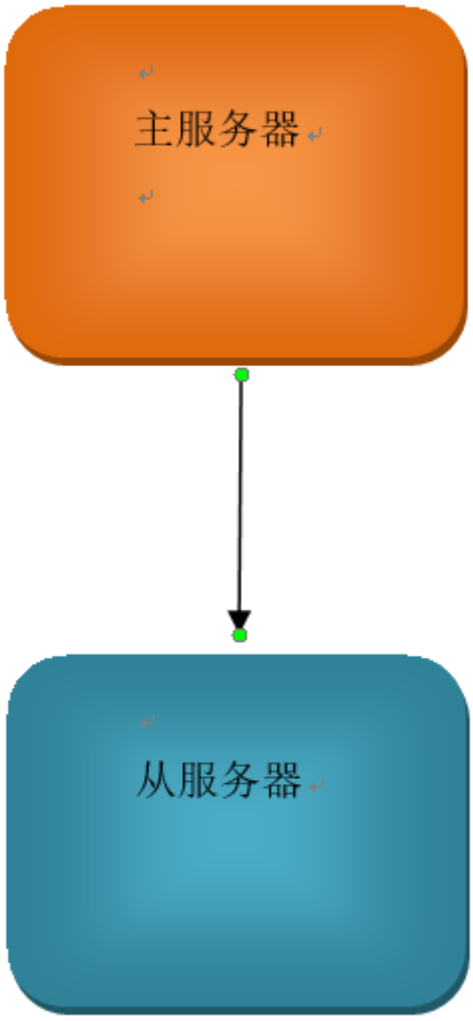
```
server-id = 101152105
#master必须要配置的
binlog-do-db=test  #需要同步的数据库
binlog-ignore-db=mysql #避免同步的数据库
#slave必须要配置的
replicate-do-db=test  #需要复制的数据库
replicate-ignore-db=mysql  #避免复制mysql库
#默认使用BLACKHOLE
default-storage-engine = BLACKHOLE
#不使用innodb
skip-innodb
########################################
###########

#default_storage_engine = InnoDB
#default-storage-engine = MyISAM
```

```
#myisam服务器
#从服务器引擎
log-bin=mysql-bin
#将主服务器的log文件更新到mysql-bin中
log-slave-updates=1  #级联复制的时候使用的
server-id = 101152103
#slave必须要配置的
replicate-do-db=test  #需要复制的数据库
replicate-ignore-db=mysql  #避免复制mysql库
#默认使用
default-storage-engine = myisam
#不使用innodb
skip-innodb
```

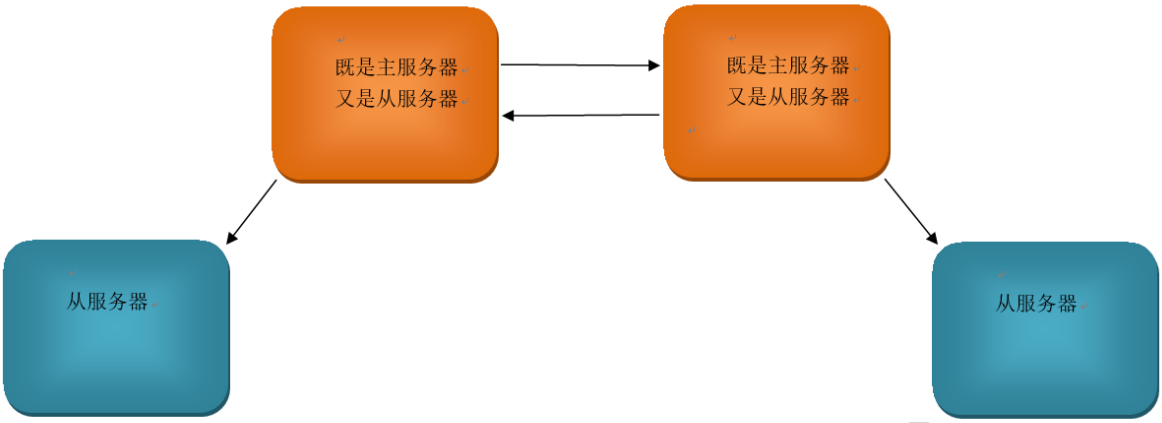
```
##############################################
#######
#default_storage_engine = InnoDB
```

2.不同的主从复制

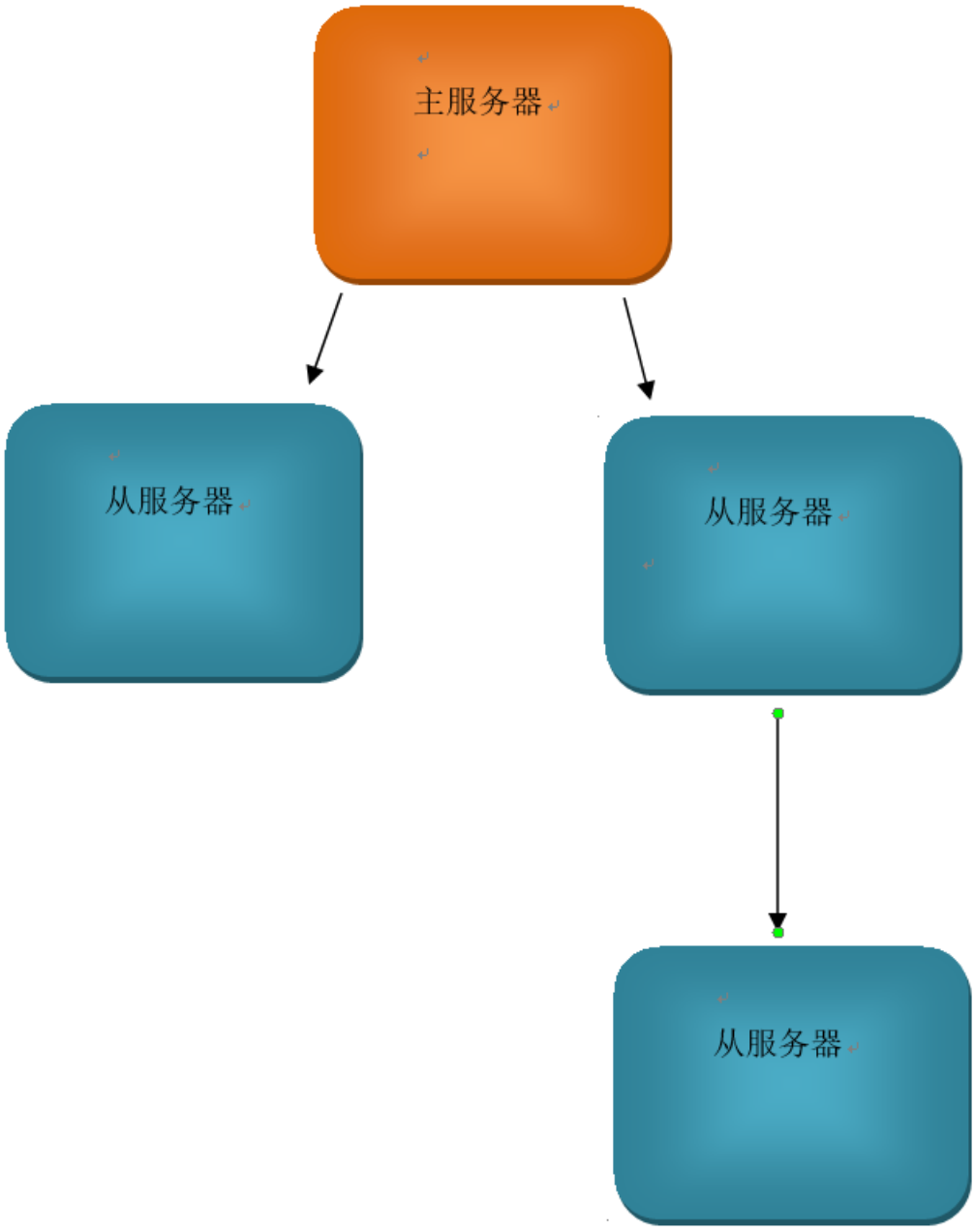
- 一主一从



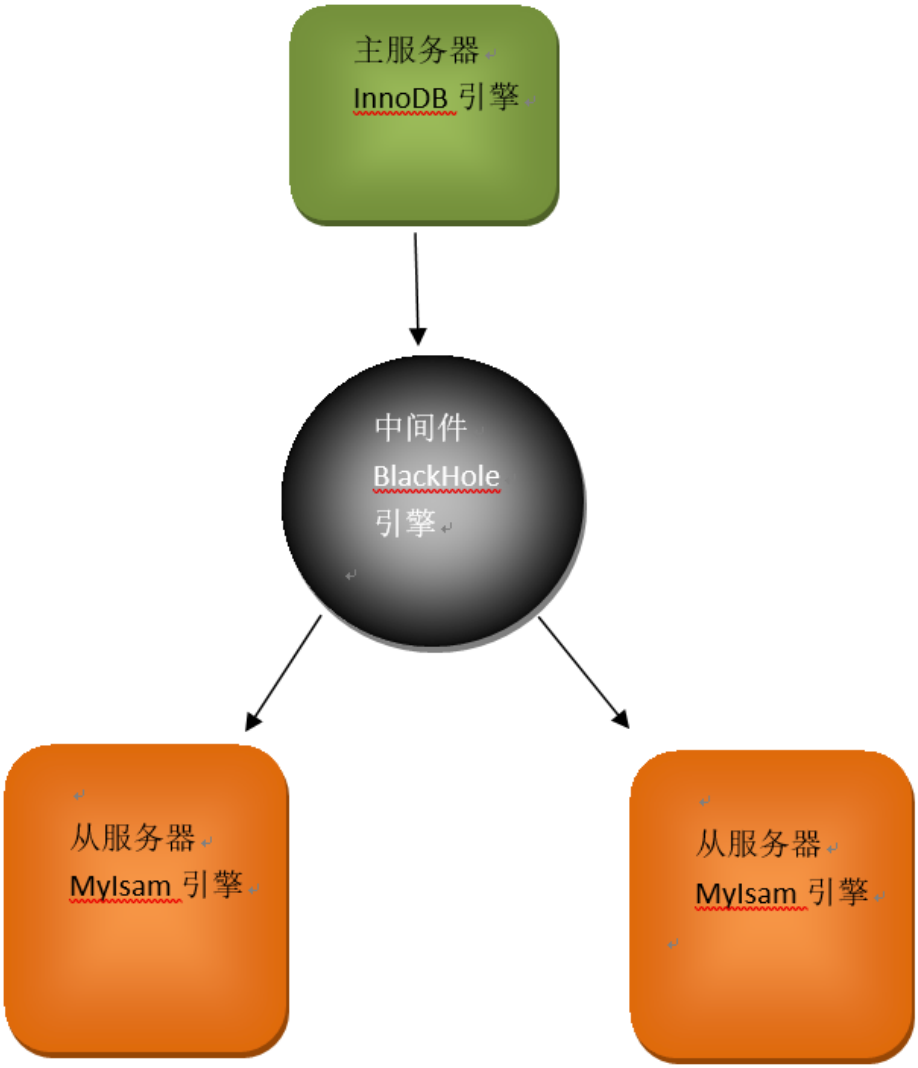
- 互为主从



- 级联复制



- 黑洞优化复制



主从面试题

不出故障:

- 配置黑洞引擎,让从服务器链接黑洞,从而给主服务器缓解压力
- 级联复制,一个主服务器有一个从服务器,从服务器下面还有一个从服务器,从服务器下面可以添加3-5个服务器.

3.读写分离

为什么要用读写分离?

- 面对绝大的访问压力,单台服务器的性能有瓶颈,需要分担负载

读写的方式:

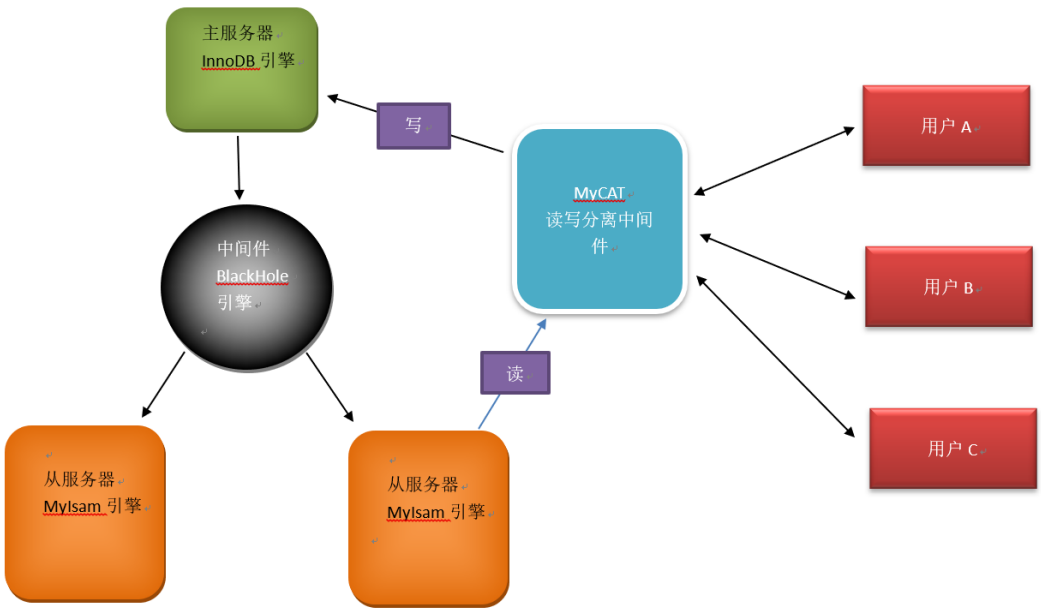
- 1.在代码当中去配置:
  - 应用程序内部使用代码的逻辑判断进行分割,代码模块只要被搭建了就能使用
  - 减少了一定的部署困难

```
import pymysql

class MyCat():
    def __init__(self,sql):
        self.sql = sql.strip()

    def query(self):
        #select * from table_name;
        head = self.sql.split(' ')[0]
        if head == 'select':
            return self.read()
        else:
            return self.write()

    def write(self):
```

```
        conn =
pymysql.connect(host='10.11.52.106',
user='hal', passwd="123456",
                            db='test',
port=3306)

 cursor=conn.cursor(cursor=pymysql.cursor
s.DictCursor)
        try:
            cursor.execute(self.sql)
            conn.commit()
            return [{'res':'ok'}]
        except Exception as e:
            conn.rollback()
            return  '语句执行失
败:\n'+str(e)
        finally:
            cursor.close()
            conn.close()

    def read(self):
        conn =
pymysql.connect(host='10.11.52.103',
user='hal', passwd="123456",
                            db='test',
port=3306)

 cursor=conn.cursor(cursor=pymysql.cursor
s.DictCursor)
        try:
```

```
            cursor.execute(self.sql)
            conn.commit()
            result = cursor.fetchall()
            return result
        except Exception as e:
            conn.rollback()
            print('查询失败:\n'+str(e))
        finally:
            cursor.close()
            conn.close()


sql = "insert into `qwer1` set id=2"
action = MyCat(sql)
print(action.query())
```

- 2.中间件层实现



  - 管理起来比较方便(mycat,java编写的)
  - 搭建非常的繁琐

```
准备jdk
准备mycat
#下载
scp root@10.11.52.106:/root/mycat.tar.gz
.
scp root@10.11.52.106:/root/jdk8.tar.gz .

#上传
scp mycat.tar.gz
root@101.123.186.200:/root/.
```

jdk配置安装

```
#解压
tar fx jdk8.tar.gz -C /usr/java
#进入目录
cd /usr/java
#修改名称
mv jdk1.8.0_161/  jdk1.8.0
#配置环境变量
vim /etc/profile +

##############################javaenv########
#####################
JAVA_HOME=/usr/java/jdk1.8.0
CLASSPATH=$JAVA_HOME/lib/
PATH=$PATH:$JAVA_HOME/bin/
export JAVA_HOME CLASSPATH  PATH

#刷新
```

```
source /etc/profile
```

mycat的配置和安装

```
#解压
tar fx mycat.tar.gz -C /usr/local/
#切换目录
cd /usr/local/mycat
#主要目录介绍
conf-------->配置项
logs-------->查看mycat的运行状态

#配置环境变量
vim /etc/profile +

##########################mycatenv#########
#####################
MYCAT_HOME=/usr/local/mycat
PATH=$PATH:$MYCAT_HOME/bin/
export MYCAT_HOME PATH

#刷新
source /etc/profile

#条件
vim /etc/hosts  #本地DNS
#10.11.52.105 Centos7
```

配置登录的用户信息

```
#配置mycat连接服务器
```

```
vim /usr/local/mycat/conf/server.xml

</system>
#配置用户名(仅是mycat用户)
        <user name="root">
        #密码
                <property
name="password">123456</property>
        #可以访问的库
                <property
name="schemas">test</property>
        </user>

        <user name="hal">
              <property
name="password">123456</property>
                <property
name="schemas">test</property>
       #读写权限
                <property
name="readOnly">false</property>
        </user>
```

配置可操作数据库信息

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM
"schema.dtd">
<mycat:schema
xmlns:mycat="http://io.mycat/">
        <!--可以被显示的数据库名称-->
```

```
        <!--dataNode是指定使用mysql中的哪个
数据库-->
        <schema name="test"
checkSQLschema="false" sqlMaxLimit="100"
dataNode="node1">
        </schema>
        <!--指定使用mysql中的test库-->
        <dataNode name="node1"
dataHost="host1" database="test" />
        <!--对host1的操作项-->
        <!--balance="1"是开启读写-->
        <dataHost name="host1"
maxCon="1000" minCon="10" balance="1"
                          writeType="0"
dbType="mysql" dbDriver="native"
switchType="1"  slaveThreshold="100">
                <heartbeat>select user()
</heartbeat>
                <!--写库进行配置-->
                <writeHost host="Centos7"
url="10.11.52.111:3306" user="hal"

password="123456">
                        <!--读库的配置-->
                        <readHost
host="centos68" url="10.11.52.112:3306"
user="hal" password="123456" />
                </writeHost>
        </dataHost>
</mycat:schema>
```

mycat的启动和关闭

```
mycat start|restart|stop
```

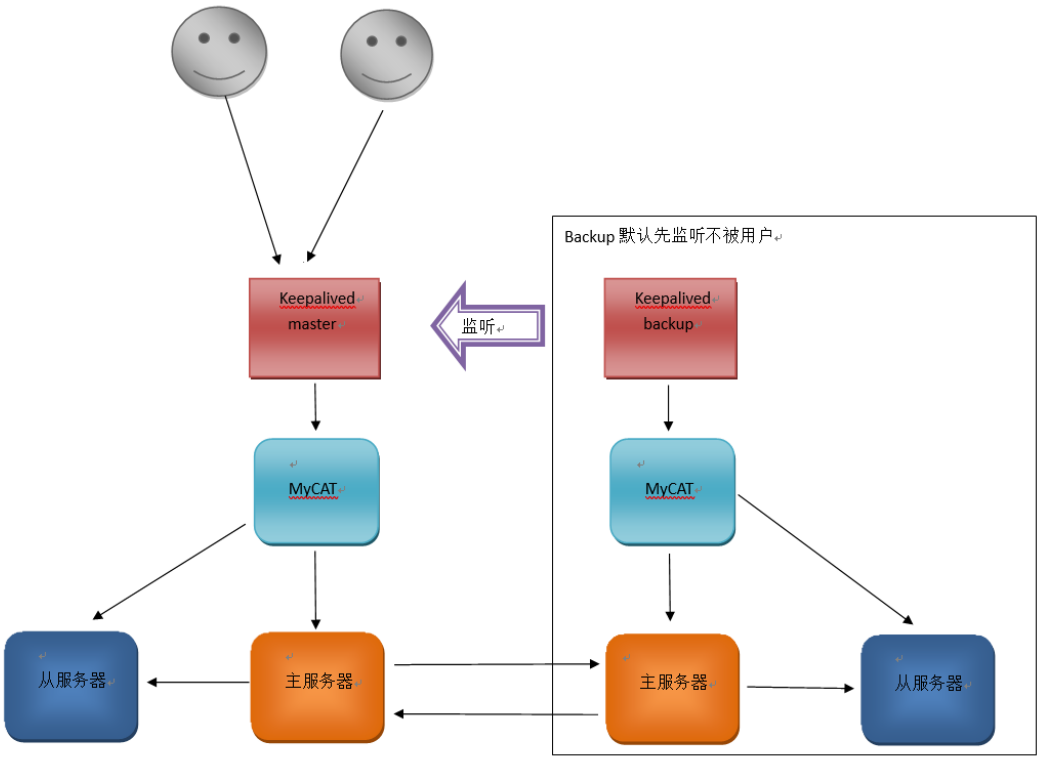
如果验证mycat已经启动

```
mycat status
```

日志查看

```
cat /usr/local/mycat/logs/wrapper.log
```

4.高可用的主从(故障切换)



虚拟冗余路由配置

keepalived的安装

```
yum install keepalived -y
```

```
#配置文件路径
vim /etc/keepalived/keepalived.conf
```

配置项

```
#主机器配置
global_defs {
     #出故障了把信息发给谁,邮件
    notification_email {
         123456@qq.com
    }
    #用谁的邮件发送
    notification_email_from
pwd.654321@qq.com
    #smtp邮件服务器
    smtp_server smtp.qq.com
    #邮件如果30秒内没有发出去
    smtp_connect_timeout 30
 #昵称
    router_id lvs_1
}
#router_id
vrrp_instance lvs_1 {
     #确定主机身份
     state MASTER
     #绑定使用的网卡,可以使用ifconfig来查看
     interface ens33
      #路由id,要求主从相同(备注:如果不相同,会裂脑)
```

```
    virtual_router_id 1
    #权重大的为主
    priority 100
    #故障切换需要消耗的时间 ,单位是秒
    advert_int 1
    #主机的口令,两边必须一致
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    #虚拟的路由地址,在公网中的话,需要多使用一张网卡,可以多接入一个空闲的ip
    #局域网中空闲的ip很多
    virtual_ipaddress {
        10.11.52.200
        10.11.52.201
        10.11.52.202
    }
}
```

```
#从机器
global_defs {
    #出故障了把信息发给谁,邮件
   notification_email {
        123456@qq.com
   }
   #用谁的邮件发送
   notification_email_from pwd.654321@qq.com
   #smtp邮件服务器
```

```
   smtp_server smtp.qq.com
   #邮件如果30秒内没有发出去
   smtp_connect_timeout 30
 #昵称
   router_id lvs_2
}
#router_id
vrrp_instance lvs_2 {
    #确定主机身份
    state BACKUP
    #绑定使用的网卡,可以使用ifconfig来查看
    interface eth0
     #路由id,要求主从相同(备注:如果不相同,会裂脑)
    virtual_router_id 1
    #权重大的为主
    priority 99
    #故障切换需要消耗的时间 ,单位是秒
    advert_int 1
    #主机的口令,两边必须一致
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    #虚拟的路由地址,在公网中的话,需要多使用一张网卡,可以多接入一个空闲的ip
    #局域网中空闲的ip很多
    virtual_ipaddress {
        10.11.52.200
        10.11.52.201
        10.11.52.202
```

```
    }
}
```

启动keepalived

```
service keepalived start
```

如何检验配置是否成功

```
ip addr #获取当前的ip    也可以获取代理ip
```