# Nginx教程

# 课程目标

## 了解反向代理和负载均衡的概念

## 掌握Nginx的安装和使用

## 利用Nginx实现负载均衡

# Nginx相关概念

## 反向代理

反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。



## 负载均衡

负载均衡，英文名称为Load Balance，是指建立在现有网络结构之上，并提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。其原理就是数据流量分摊到多个服务器上执行，减轻每台服务器的压力，多台服务器共同完成工作任务，从而提高了数据的吞吐量。



# Nginx的安装

## 下载nginx

官网：<http://nginx.org/>

## 上传并解压nginx

tar -zxvf nginx-1.8.1.tar.gz -C /usr/local/src

## 编译nginx

#进入到nginx源码目录

cd /usr/local/src/nginx-1.8.1

#检查安装环境,并指定将来要安装的路径

./configure --prefix=/usr/local/nginx

#缺包报错 ./configure: error: C compiler cc is not found

#使用YUM安装缺少的包

yum -y install gcc pcre-devel openssl openssl-devel

#编译安装

make && make install

安装完后测试是否正常：

启动

cd /usr/local/nginx/sbin/

./nginx

查看端口是否有ngnix进程监听

netstat -ntlp | grep 80

# 配置nginx

## 配置反向代理

1. 修改nginx配置文件 conf/nginx.conf

|  |
| --- |
| server {  listen 80;  server\_name nginx-01.itcast.cn; #nginx所在服务器的主机名  #反向代理的配置  location / { # /代表拦截所有请求  root html;  proxy\_pass <http://192.168.0.21:8080;> #这里是代理走向的目标服务器：tomcat  }  } |

1. 启动tomcat-01上的tomcat

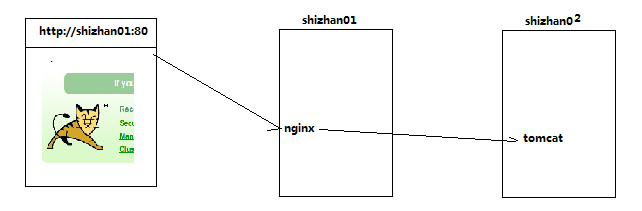
3.启动nginx-01上的nginx

./nginx -s reload

重启:

kill -HUP `cat /usr/local/nginx/logs/nginx.pid `

参考网址:http://www.cnblogs.com/jianxie/p/3990377.html



## 动静分离

将上面location替换成如下：

#动态资源 index.jsp

|  |
| --- |
| # ~ 代表大小写敏感； . 代表任何单个字符； \* 零或多个字符； \. 代表.  location ~ .\*\.(jsp|do|action)$ {  proxy\_pass http://tomcat-01.itcast.cn:8080;  } |

#静态资源

|  |
| --- |
| location ~ .\*\.(html|js|css|gif|jpg|jpeg|png)$ {  expires 3d;  } |

## 负载均衡

在http这个节下面配置一个叫upstream的，后面的名字可以随意取，但是要和location下的proxy\_pass http://后的保持一致。

|  |
| --- |
| http {  是在http里面的, 已有http, 不是在server里,在server外面  upstream tomcats {  server shizhan02:8080 weight=1;#weight表示表示权重  server shizhan03:8080 weight=1;  server shizhan04:8080 weight=1;  }  #卸载server里  location ~ .\*\.(jsp|do|action) {  proxy\_pass <http://tomcats;> #tomcats是后面的tomcat服务器组的逻辑组号  }  } |

# 利用keepalived实现高可靠（HA）

## 高可靠概念

HA(High Available), 高可用性集群，是保证业务连续性的有效解决方案，一般有两个或两个以上的节点，且分为活动节点及备用节点。



## 高可靠软件keepalived

keepalive是一款可以实现高可靠的软件，通常部署在2台服务器上，分为一主一备。Keepalived可以对本机上的进程进行检测，一旦Master检测出某个进程出现问题，将自己切换成Backup状态，然后通知另外一个节点切换成Master状态。

## keepalived安装

下载keepalived官网：<http://keepalived.org>

将keepalived解压到/usr/local/src目录下

tar -zxvf keepalived-1.2.19.tar.gz -C /usr/local/src

进入到/usr/local/src/keepalived-1.2.19目录

cd /usr/local/src/keepalived-1.2.19

注意：CentOS7需要从Github上下载最新的

git clone https://github.com/acassen/keepalived.git

或者下载了再上传

开始configure

./configure --prefix=/usr/local/keepalived

#编译并安装

make && make install

## 将keepalived添加到系统服务中

拷贝执行文件

cp /usr/local/keepalived/sbin/keepalived /usr/sbin/

将init.d文件拷贝到etc下,加入开机启动项

~~cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/keepalived~~

1.3.2版本：cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/init.d/keepalived

将keepalived文件拷贝到etc下

cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/

创建keepalived文件夹

mkdir -p /etc/keepalived

将keepalived配置文件拷贝到etc下

cp /usr/local/keepalived/etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf

添加可执行权限

chmod +x /etc/init.d/keepalived

##以上所有命令一次性执行：

|  |
| --- |
| cp /usr/local/keepalived/sbin/keepalived /usr/sbin/  cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/keepalived  cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/  mkdir -p /etc/keepalived  cp /usr/local/keepalived/etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf  chmod +x /etc/init.d/keepalived  chkconfig --add keepalived  chkconfig keepalived on |

添加keepalived到开机启动

~~chkconfig --add keepalived~~

~~chkconfig keepalived on~~

CentOS7：systemctl enable keepalived

## 配置keepalived虚拟IP

修改配置文件： vi /etc/keepalived/keepalived.conf

#MASTER节点

|  |
| --- |
| global\_defs {  #注意这里为空！否则无法访问虚拟IP！  }  vrrp\_instance VI\_1 {  state MASTER #指定A节点为主节点 备用节点上设置为BACKUP即可  interface eth0 #绑定虚拟IP的网络接口 ip addr 查看自己的  virtual\_router\_id 51 #VRRP组名，两个节点的设置必须一样，以指明各个节点属于同一VRRP组  priority 100 #主节点的优先级（1-254之间），备用节点必须比主节点优先级低  advert\_int 1 #组播信息发送间隔，两个节点设置必须一样  authentication { #设置验证信息，两个节点必须一致  auth\_type PASS  auth\_pass 1111  }  virtual\_ipaddress { #指定虚拟IP, 两个节点设置必须一样  192.168.33.60/24 #如果两个nginx的ip分别是192.168.33.61,,...62，则此处的虚拟ip跟它俩同一个网段即可  }  } |

#BACKUP节点

|  |
| --- |
| global\_defs {  #注意这里为空！否则无法访问虚拟IP！  }  vrrp\_instance VI\_1 {  state BACKUP  interface eth0  virtual\_router\_id 51  priority 99  advert\_int 1  authentication {  auth\_type PASS  auth\_pass 1111  }  virtual\_ipaddress {  192.168.33.60/24  }  } |

#分别启动两台机器上的keepalived

service keepalived start

测试：

杀掉master上的keepalived进程，你会发现，在slave机器上的eth0网卡多了一个ip地址

查看ip地址的命令： ip addr

## 配置keepalived心跳检查

原理：

Keepalived并不跟nginx耦合，它俩完全不是一家人

但是keepalived提供一个机制：让用户自定义一个shell脚本去检测用户自己的程序，返回状态给keepalived就可以了

#MASTER节点

|  |
| --- |
| global\_defs {  }  vrrp\_script chk\_health {  # 脚本意义是检测nginx是否有两个进程，两个进程则正常  script "[[ `ps -ef | grep nginx | grep -v grep | wc -l` -ge 2 ]] && exit 0 || exit 1"  interval 1 #每隔1秒执行上述的脚本，去检查用户的程序ngnix  weight -2 #nginx进程有问题则减2个权重值  }  vrrp\_instance VI\_1 {  state MASTER  interface eth0  virtual\_router\_id 1  priority 100  advert\_int 2  authentication {  auth\_type PASS  auth\_pass 1111  }  track\_script {  chk\_health  }  virtual\_ipaddress {  10.0.0.10/24  }  #发生问题执行以下脚本，重启nginx  notify\_master "/usr/local/keepalived/sbin/notify.sh master" #启切换master，动nginx  notify\_backup "/usr/local/keepalived/sbin/notify.sh backup" #切换backup，先停止再启动nginx  notify\_fault "/usr/local/keepalived/sbin/notify.sh fault" #其它问题则停止ngnix  } |

#添加切换通知脚本

vi /usr/local/keepalived/sbin/notify.sh

|  |
| --- |
| #!/bin/bash  case "$1" in  master)  /usr/local/nginx/sbin/nginx  exit 0  ;;  backup)  /usr/local/nginx/sbin/nginx -s stop  /usr/local/nginx/sbin/nginx  exit 0  ;;  fault)  /usr/local/nginx/sbin/nginx -s stop  exit 0  ;;  \*)  echo 'Usage: notify.sh {master|backup|fault}'  exit 1  ;;  esac |

#添加执行权限

chmod +x /usr/local/keepalived/sbin/notify.sh

#BACKUP节点

|  |
| --- |
| global\_defs {  }  vrrp\_script chk\_health {  script "[[ `ps -ef | grep nginx | grep -v grep | wc -l` -ge 2 ]] && exit 0 || exit 1"  interval 1  weight -2  }  vrrp\_instance VI\_1 {  state BACKUP  interface eth0  virtual\_router\_id 1  priority 99  advert\_int 1  authentication {  auth\_type PASS  auth\_pass 1111  }  track\_script {  chk\_health  }  virtual\_ipaddress {  10.0.0.10/24  }    notify\_master "/usr/local/keepalived/sbin/notify.sh master"  notify\_backup "/usr/local/keepalived/sbin/notify.sh backup"  notify\_fault "/usr/local/keepalived/sbin/notify.sh fault"  } |

#在第二台机器上添加notify.sh脚本

#分别在两台机器上启动keepalived

systemctl start keepalived

systemctl enable keepalived

## 主副机配置主要区别

state MASTER / BACKUP

priority 100 / 99

## 测试

(1)关闭 192.168.10.103 中的 Nginx，Keepalived会将它重新启动

/usr/local/nginx/sbin/nginx -s stop

ps -ef | grep nginx

(2)关闭 192.168.10.103 中的 Keepalived，VIP 会切换到 192.168.10.104 中

systemctl stop keepalived

(3)重新启动 192.168.1.103 中的 Keepalived，VIP 又会切回到 192.168.10.103 中来

systemctl start keepalived