shutdown now //关机

netstat -ntlp // 查看本地连接状态

vim /etc/sysconfig/network-scripts/ifcfg-ens33 // 命令配置

nmtui // 图形化配置界面

nmcli connection // 查看网卡信息

nmcli connection show ens33 // 查看具体的网络接口信息

nmcli device status // 显示所有设备状态

nmcli connection reload // 修改配置文件执行生效

nmcli connection show --active // 显示所有活动连接

nmcli connection delete eth0 // 删除一个网卡连接

nmcli connection add type ethernet con-name eth0 ifname eno33554992 //添加一个网卡连接

nmcli connection down eth0 //停用

nmcli connection up eth0 // 启用

hostnamectl set-hostname mylinux.local // 设置主机名

yum provides 工具名 //查询该工具是由哪个包提供的

yum install epel-release -y // 安装扩展源

yum install net-tools -y //安装网络工具

yum install keepalived -y //安装虚拟的路由协议

yum install -y net-tools // 安装网络工具

yum install zlib-devel -y // 支持 gzip 模块

yum install bzip2 -y // 支持 bzip2 模块

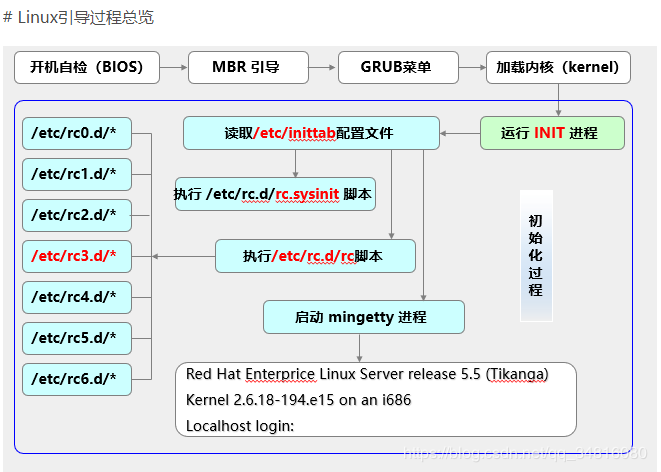
yum clean all //清除yum 缓存

yum makecache //把服务器的包信息下载到本地电脑缓存起来

yum update -y //更新软件包

rpm --rebuilddb //重建数据库

cat /etc/redhat-release //查看当前系统版本号



cat /etc/default/grub // 查看系统内核开机选单

echo "" > "" // 覆盖 echo"">>"" // 追加

wget -c http://nginx.org/download/nginx-1.16.0.tar.gz // 下载nginx

repo文件中enabled是否为1，1表示开启，0表示关闭 //查看源仓库是否开启，yum install走高版本的仓库

yum repolist enabled // 查看安装源仓库 是否开启

执行命令：yum install lrzsz -y 安装一个可以上传，下载的软件  
执行命令：rz 选中文件，然后上传  
执行命令：sz 文件名 下载文件

parted -s /dev/sdb mklabel gpt // 不进行交互来分区

mkfs.ext4 /dev/sdb

mount /dev/sdb /data/

mkpart jfedu1 0G 1000G // 分区名称 容量大小

mkpart jfedu2 1000G 2000G

mkpart jfedu3 part-type ext4 1000G 2000G //指定文件类型

p

df -h // 查看磁盘使用情况

fdisk -l // 查看本机磁盘具体情况

umount /dev/sdb // 卸载sdb磁盘

parted -s /dev/sdb mklabel gpt // 设置分区类型为gpt 格式

mkfs.ext4 /dev/sdb // 格式化成ext4的文件系统类型

mount /dev/sdb /data // 将/dev/sdb 磁盘挂载到 /data目录下

lsblk // 查看磁盘的空间大小

配置 /etc/rc.local/ //设备开机加载磁盘

chmod +x rc.local //给rc.local 加入可执行权限

ll

vim /etc/rc.local

mount /dev/sdb /data //写入最后一行

systemctl stop firewalld.service //临时关闭防火墙

setenforce 0 // 临时关闭selinux

GG切换到最后一行，gg切换到第一行，然后按下o 输入 // vi 编辑时快速切换操作

killall 杀所有进程 pkill 杀进程 kill -9 杀端口

杀死所有同名进程

killall nginx

killall -9 bash

向进程发送指定信号

killall -KILL nginx

列出所有信号名称

kill -l

apche 和nginx 是冲突的，都是使用80端口，不能同时运行

需要 先 pkill httpd

再运行 /usr/local/nginx/sbin/nginx

或者 pkill nginx

再运行 systemctl start httpd

vim /etc/httpd/conf/httpd.conf

命令模式下 输入 /listen 查找端口 自定义端口使用 1024以上的端口

修改默认端口 保存退出

rpm -ql httpd // 查询 yum方式安装的配置文件所在的目录

vim /etc/httpd/conf/httpd.conf

listen 8085

systemctl restart httpd

find / -name nginx.conf // 查找源码方式安装 的配置文件所在目录

yum install -y mlocate

**locate**(locate) 命令用来查找文件或目录

整个locate工作其实是由四部分组成的:

/usr/bin/updatedb   主要用来更新数据库，通过crontab自动完成的

/usr/bin/locate    ‘filename’    查询文件位置

/etc/updatedb.conf   updatedb的配置文件

/var/lib/mlocate/mlocate.db  存放文件信息的文件

ps -ef|grep vi

root 3268 2884 0 16:21 pts/1 00:00:00 vim install.log

root 3370 2822 0 16:21 pts/0 00:00:00 grep vi

kill -9 3268

源码安装Nginx:

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

wget http://nginx.org/download/nginx-1.16.1.tar.gz

wget -c // 支持断点续传

wget -P /data/http://nginx.org/download/nginx-1.16.1.tar.gz

ls /data

-P 指定下载文件得存放路径; -O 指定下载文件得存放路径，且可以自定义文件名

解压之后编译、安装：

先安装编译器以及依赖：

yum install gcc gc gcc-c++ pcre pcre-devel zlib-devel openssl openssl-devel -y

tar -xzf nginx-1.14.2.tar.gz

cd nginx-1.14.2

useradd www

./configure --prefix=/usr/local/nginx --user=www --group=www // 标准预编译

./configure --prefix=/usr/local/nginx --user=www --group=www --with-http\_stub\_status\_module --with-http\_ssl\_module --add-module=/root/echo-nginx-module-0.61

//开启nginx状态监控 开启ssl加密传输 添加echo模块

make && make install // 编译并安装

/usr/local/nginx/sbin/nginx -V

/usr/local/nginx/sbin/nginx -t

ss -napt // socket statistics 和netstat类似

ss -napti

ss -napti4

针对源码方式安装的nginx:

vim /etc/profile // 配置环境变量

export PATH=/usr/local/nginx/sbin/:$PATH //写到最后一行，设置环境变量

:wq

source /etc/profile //执行某个文件，让文件内容生效

完成后就可以直接使用nginx命令启动nginx服务

第一种方法：

echo ”export PATH=/usr/local/nginx/sbin:$PATH” > /etc/profile.d/nginx.sh //写入shell文件

. /etc/profile //重新读取

第二种方法：

alias nginx=’/usr/sbin/nginx’ // 定义别名

vim ~/.bashrc // 写入配置文件永久生效

第三种方法：

ln -s /usr/local/nginx/sbin/nginx /usr/local/sbin/ //定义软链接

nginx -v

如果不显示 就用 export PATH=$PATH:/usr/local/nginx/sbin 在环境变量里加上nginx变量

ps -ef|grep nginx //查看nginx服务是否开启

/usr/local/nginx/sbin/nginx //启动nginx服务

ps -ef|grep "nginx" //再次查看nginx服务是否开启

ip add //查看 当前 IP地址

systemctl status firewalld //查看防火墙状态

systemctl stop firewalld //停止防火墙服务systemctl start firewalld //开启防火墙

systemctl disable firewalld //禁止防火墙开机启动

setenforce 0 //关闭SElinux

sestatus //查看SElinux 的状态

firewall-cmd --list-all //查看防火墙状态

vim /etc/selinux/config

"SELINUX=disabled"

#开启防火墙  
systemctl start firewalld  
#查看状态：  
systemctl status firewalld   
#停止：  
systemctl disable firewalld  
在开机时启用一个服务：systemctl enable firewalld.service  
在开机时禁用一个服务：systemctl disable firewalld.service  
#配置打开firewall服务时firewall需要开放的端口或服务  
#添加端口  
firewall-cmd --zone=public(作用域) --add-port=80/tcp(端口和访问类型) --permanent(永久生效)

#添加服务  
firewall-cmd --zone=public --add-service=http --permanent  
#重新载入，更新防火墙规则  
firewall-cmd --reload

#查看：  
firewall-cmd  --list-all

使用 ens33的 IP地址 在浏览器上打开 welcome to nginx

rpm -qa|grep nginx // 查看 nginx 是否发布成功

cd /usr/local/nginx/html //进入页面文件所在目录

ls //查看所有文件

vim index.html //修改页面文件内容

/usr/local/nginx/sbin/nginx -s reload //源码安装方式 配置文件修改后重装载

vim /usr/share/nginx/html/index.html // yum安装方式 修改配置文件

nginx -s reload //配置文件修改后重新装载

nginx -c /usr/local/nginx/conf/nginx.conf //利用配置文件启动nginx

service nginx restart //重启服务

nginx -s stop // 快速停止或关闭nginx

nginx -s quit //正常停止或关闭nginx

ps -aux |grep nginx //查看 nginx 进程

/usr/local/nginx/sbin/nginx -s reload //重启nginx

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

单独安装nginx的echo 模块：

rz 上传echo 模块

tar xzf echo.tar.gz -C /root/nginx

./configure --add-module=echo-nginx-module-0.61

make

mv /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx.old

cp objs/nginx /usr/local/nginx/sbin/nginx

make upgrade

/usr/local/nginx/sbin/nginx -V

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

源码安装 和 yum安装 是两套系统 如果源码安装以后 发现rpm -qa|grep nginx 下有残留，就用 yum remove ‘filename’ -y 卸载掉

建议不要用yum remove 移除支持库文件  
移除yum装的支持库文件  可以先用yum history list 查看 序号  
运用yum history redo 序号  -->重装  
运用yum history undo 序号 -->回滚

例如yum history undo ID  
 yum remove 可以卸载一些单纯的服务

 yum history info  gd 或者 yum history info  ID # 查询安装信息

源码安装方式升级

./configure --prefix=/usr/local/nginx --user=www --group=www

make

find / -name nginx

cd /root/nginx-1.16.1/objs

ll

cp nginx /usr/local/nginx/sbin/

y

或者

mv /root/nginx-1.16.1/objs/nginx -b /usr/local/nginx/sbin //移动源文件覆盖目标文件并备份目标文件

make upgrade

卸载源码安装的程序

rm -rf /usr/local/nginx

#####################################

yum 方式安装nginx:

yum install nginx nginx-module-echo -y

yum安装方式升级

上传repo最新软件仓库

设置仓库

[nginx-1.16.1]

enabled = 1

gpgcheck=0

yum install nginx -y

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

启动防火墙 —— systemctl start firewalld.service

停止防火墙 —— systemctl stop firewalld.service

查看firewalld防火墙状态 —— firewall-cmd --state

禁止防火墙开机启动 —— systemctl disable firewalld.service

列出正在运行的服务状态 —— systemctl

启动一个服务 —— systemctl start postfix.service

关闭一个服务 —— systemctl stop postfix.service

重启一个服务 —— systemctl restart postfix.service

显示一个服务的状态 —— systemctl status postfix.service

在开机时启用一个服务 —— systemctl enable postfix.service

在开机时禁用一个服务 —— systemctl disable postfix.service

查看服务是否开机启动 —— systemctl is-enabled postfix.service

查看已启动的服务列表 —— systemctl list-unit-files|grep enabled

移动文件：mv 文件名 移动目的地文件名

重命名文件：mv 文件名 修改后的文件名

mv 命令参数：

-b : 若需覆盖文件，则覆盖前先行备份。

mv log1.txt -b log2.txt  
-f : force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖；  
-i : 若目标文件 (destination) 已经存在时，就会询问是否覆盖！  
-u : 若目标文件已经存在，且 source 比较新，才会更新(update)  
-t : --target-directory=DIRECTORY move all SOURCE arguments into DIRECTORY，即指定mv的目标目录，该选项适用于移动多个源文件到一个目录的情况，此时目标目录在前，源文件在后。

mv  a.dir  b.dir   c.dir  1.txt  2.txt  -t  des.dir

或者 mv -t des.dir a.dir b.dir c.dir 1.txt 2.txt

rpm -e make //单独删除make 程序包，不删除其依赖关系包

yum info php //查看版本信息

yum list nginx\* //查看nginx 支持的模块

rm -rf /\* //删除当前目录下所有文件

rm -rf !(1.txt) //排除1.txt后删除其它所有文件

rm -rf !(1.txt|2.txt) // 排除1.txt和2.txt后删除其它所有文件

#如果执行！号提示event not found,则需要开启通配符功能

shopt -s extglob

#查看通配符是否开启

shopt -s

touch .jfedu // . 代表隐藏文件,新建一个隐藏文件

touch -t 201908231240 jfedu //指定文件的时间

ls -lh

ls -lih // 查看包含Inode的 文件

ls -lah // 查看 隐藏文件

mv .jfedu jfedu // 更改隐藏文件为 普通文件

mkdir -p jfedu1/a/b/c/d // 递归创建目录

rm -v jfedu // 显示删除过程

cp 1.txt 2.txt // 拷贝旧文件到新文件

cp -r jfedu7 jfedu8 // 拷贝目录需要 加 -r 参数

cp -a jfedu7 jfedu10 //拷贝jfedu7后并改名为jfedu10 -a保留文件或文件夹原有权限

cp -dr jfedu7 // 保留原有链接，拷贝后不受影响

ln -s /etc/httpd/conf.d/welcome.conf welcome.conf //软链接

stat jfedu // 查看文件权限

cat -n // 查看文件内容的行号

cat -n /etc/passwd|head -20

cat -n /etc/passwd|tail -20

echo "test " > 1.txt

echo "test1" >> 1.txt

echo "test2" >> 1.txt

for i in 'seq 1 1000';do echo $i >1.txt;done // 循环增加从1到1000

for i in 'seq 1 1000';do echo $i >> 1.txt;done

tail -f // 动态查看文件后面追加的内容

icdiff // 比较两个文件,并列显示两个文件的不同

yum install git -y

$ git clone https://github.com/jeffkaufman/icdiff.git

$ cd icdiff

$ ./icdiff 1.css 2.css

完全删除用户

1.userdel -r "username"

2.userdel "username"

find / -name "\*username\*"

rm -rf

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

yum install -y wget tar make gzip gcc

yum install -y pcre pcre-devel zlib-devel

tar xzf nginx-1.16.0.tar.gz

cd nginx-1.16.0

useradd -s /sbin/nologin www -M # 添加一个不登录系统不创建家目录 的用户

./configure --prefix=/usr/local/nginx/ --user=www --group=www --with-http\_stub\_status\_module

make

make install

/usr/local/nginx/sbin/nginx

ps -ef|grep nginx

netstat -ntlp|grep -w 80 //grep -w字符串精确匹配 grep -e显示文件中符合条件的字符

setenforce 0

firewall-cmd --add-port=80/tcp --permanent

systemctl reload firewall.d.service

iptables -A INPUT -m tcp -p tcp --dport80 -j ACCEPT

service iptables restart

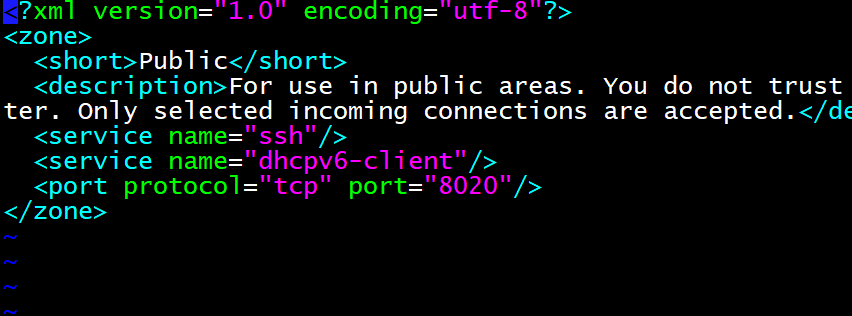
cat /etc/sysconfig/iptables

//本机防火墙开放8020端口

cat /etc/firewalld/zones/public.xml

vim /etc/firewalld/zones/public.xml 在<zone>最后一行添加下列代码</zone>

<port protocol="tcp" port="8020"/>



systemctl reload firewalld.service  
firewall-cmd --list-all

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

#定义变量

NGX\_SOFT=""

NGX\_DIR=""

NGX\_URL=""

NGX\_YUM=""

NGX\_SRC=$(echo $NGX\_SOFT|sed's/ .tar:\*//g')

NGX\_ARGS=""

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

echo "password"｜passwd --stdin root //修改当前用户登录密码

journalctl -xe // 提示错误信息

location其实就是用户请求nginx web的时候，作为匹配入口，判断用户是请求哪个文件或者URL

nginx负载均衡upstream的四种分配方式:

RR轮询：每个请求按照时间顺序轮流分配到不同的后端服务器，如果某个后端服务器down掉后，能自动剔除

upstream backend {

server 192.168.1.101:8888;

server 192.168.1.102:8888;

server 192.168.1.103:8888;

}

weight加权轮询: 可以指定轮询比率,weight和访问几率成正比,主要应用于后端服务器异质的场景下

upstream backend {

server 192.168.1.101 weight=1;

server 192.168.1.102 weight=2;

server 192.168.1.103 weight=3;

}

ip\_hash: 每个请求按照访问ip的hash结果分配，这样每个访客会固定访问一个后端服务器，可以解决session一致问题

注意：  
1、当负载调度算法为ip\_hash时，后端服务器在负载均衡调度中的状态不能是weight和backup。  
2、导致负载不均衡。

upstream backend {

ip\_hash;

server 192.168.1.101:7777;

server 192.168.1.102:8888;

server 192.168.1.103:9999;

}

least\_conn：lease\_conn用于为网络连接分配服务器组内的服务器，在功能上实现了最小连接数负载均衡算法，在选择组内的服务器时，考虑各服务器权重的同时，每次选择的都是当前网络连接最少的那台服务器，如果这样的服务器有多台，就采用加权轮询选择权重值大的服务器

**选取活跃连接数与权重weight的比值最小者为下一个处理请求的server**

upstream leastconn {

least\_conn;

server 192.168.1.101 weight=2;

server 192.168.1.102 weight=1;

server 192.168.1.103 weight=1;

}

如果上一个请求选择了第二台192.168.1.102, 下一个请求到来, 通过比较剩下可用的server的conns/weight值来决定选择哪一台

如果第一台连接数为100, 第三台连接数为80, 因为权重分别是2和1, 所以计算结果为100/2=50, 80/1=80。50<80所以选择第一台而不选第三台。尽管连接数第一台大于第三台。

netstat -nat|grep ESTABLISHED|wc -l # 查看服务器有效连接数

netstat -an|awk '/tcp/ {print $6}'|sort|uniq -c #查看不同状态的连接数

状态：描述

CLOSED：无连接是活动的或正在进行

LISTEN：服务器在等待进入呼叫

SYN\_RECV：一个连接请求已经到达，等待确认

SYN\_SENT：应用已经开始，打开一个连接

ESTABLISHED：正常数据传输状态

FIN\_WAIT1：应用说它已经完成

FIN\_WAIT2：另一边已同意释放

ITMED\_WAIT：等待所有分组死掉

CLOSING：两边同时尝试关闭

TIME\_WAIT：另一边已初始化一个释放

LAST\_ACK：等待所有分组死掉

————————————————

-p 递归创建目录

mkdir /jfedu

mkdir -p /jfedu/jf

mkdir -p /jfedu/jf{1,2} #创建两个目录

mkdir -p /jfedu/jf{1..100} #创建多个目录

chown root nginx //修改所属主

chgrp root nginx //修改所属组

chown -R root:root nginx-1.16.1 //同时修改所属主和所属组

chown -R nginx:root nginx-1.16.1

chmod g+r anaconda-ks.cfg // 组 具有 读的权限

chmod o+r anaconda-ks.cfg // 其他用户 具有读的权限

chmod 777 anaconda-ks.cfg // 最大权限

chmod g+x,o+x anaconda-ks.cfg // 同时给权限

chmod g-x anaconda-ks.cfg //撤销权限

$ exit //返回root用户

chmod o+t /data -R //给予sbit 权限

ll -d /data

tar -tf nginx-1.16.1.tar.gz //查看压缩包内的文件

tar -xzf nginx-1.16.1.tar.gz -C /opt/ //解压缩包 到 指定的目录下

tar -xjf nginx-1.16.1.bz2 //z 是gzip j是bz2 f是指定文件名称 x是解压 c是压缩

tar -cjf nginx.tar.bz2 nginx-1.16.1

tar -czf nginx.tar.gz nginx-1.16.1/ #把nginx-1.16.1目录下所有文件打包成 名为 nginx.tar.gz的一个压缩包文件

tar -czf docker-centos7.tar / # 把根目录下的文件打包成docker-centos7.tar

zip

unzip

rpm -ivh httpd // i 显示信息 v 显示过程 h 安装

rpm -ivh -nodeps //强制安装

rpm -qa|grep httpd //查看是否安装好

rpm -q nginx //查询某个软件是否已安装

rpm -qf /usr/’commond’ // 查询该命令是由哪个软件包提供的

rpm -ql nginx // 查询包的安装路径

rpm -qc nginx //查询包的配置文件

rpm -qa // 查询所有已安装的软件

rpm -qi nginx //查询包的版本信息

rpm -ivh nginx\_1.16.1-1.el7.x86\_64.rpm // 安装nginx二进制包

rpm -e nginx // 卸载安装包

mkdir – p /mnt/cdrom

mount /dev/cdrom /mnt/cdrom

cd /mnt/cdrom

cd packages/

#定义nginx运行的用户和用户组，可以新建www用户，设定一下配置

useradd -s /sbin/nologin www

user www www;

#启动工作进程，通常设置成与cpu的数量相等，如果cpu较多，可以设置成worker\_processes auto;

worker\_processes 8;

#为每个进程分配cpu,将8个进程分配到8个cpu中, 也可以写多个，或将一个进程分配到多个cpu

worker\_cpu\_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;

worker\_connections 10000;

最大连接数 = 连接数\*进程数

ulimit -n // 控制shell执行程序的资源

yum install -y httpd-tools

ab -c -n // apache自带的压力测试工具 -c 一次产生的请求个数，请求的用户量。-n 请求的总数量

ab -c 10 -n 100 -r http://www.qq.com/index.html //ab 命令测试高并发 如果写的比较大 还要加-r 参数（Don't exit on socket receive errors.） 否则直接退出

top // 查看cpu使用情况

lscpu // 查看cpu 状况

-c 并发数量 -n 请求数量

Server Software: Apache/1.4.18 //服务器软件版本

Server Hostname: www.06a.com // 请求的URL

Server Port: 80 //请求的端口号

Document Path: / //请求的服务器的路径

Document Length: 19590 bytes //页面长度 单位是字节

Concurrency Level: 200 // 并发数

Time taken for tests: 124.509 seconds // 一共使用了124秒

Complete requests: 1000 //请求的次数

Failed requests： 9 //失败的请求

（Connect:0,Receive:0,Length:9,Exceptions:0）

Total transferred: 19669661 bytes //总共传输的字节数 http头信息

HTML transferred: 19472463 bytes //实际页面传递的字节数

Requests per second: 8.03[#/sec](mean) // 每秒多少个请求

Time per request: 24901.805[ms](mean) //平均每个用户等待多长时间

Time per request: 124.509[ms](mean,across all concurrent requests) //服务器平均用时

Transfer rate: 154.28[kbytes/sec]received // 每秒获取多少数据

lsblk // 查看磁盘的空间大小

df -i // 查看已用的inode

ll -i ‘filename’ // 查看文件的inode

文件名和inode 是一一对应，多个文件可以使用同一个inode.

ln nginx.conf nginx1.conf // 硬链接 只针对文件，修改其中任一文件，另一个文件会自动同步；删除其中任一文件，不影响另外一个文件；不会占用额外的磁盘空间，不能跨分区使用, 主要用于备份

ll -i nginx.conf nginx1.conf // 查看两个文件的inode

cat nginx.conf -n // -n查看 行数

# 查看硬链接文件

ls -i // 查看文件的inode

find / -inum 267105 // 查看所有指向该inode的文件

# 查看软链接文件

ls -al

find -type l -exec ls -l {} \;

ln -s services services.bak // 软链接 可以跨分区使用，支持文件和目录，会消耗inode

ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

-s = soft，意为软链接,必须使用绝对路径  
-f = force，表示强制创建,替换掉任何已经存在的目的路径（使用前请备份之前存在文件）

-d 允许系统管理者硬链接自己的目录

-i 在删除与dist同档名的档案时先进行询问

-v 在链接前显示其档名

-b 将在链接时会被覆写或删除的档案进行备份

-S suffix 将备份的档案都加上suffix 的字尾

# service mysqld status 或者 # service mysql status 命令查看mysql 的启动状态  
如果是 mysqld is stopped 那就说明mysql服务是停止状态，  
如果是 mysqld is running 那就说明mysql服务是启动状态

systemctl restart mysqld //重启5.7 mysql服务

##################################################################

yum方式安装

# 下载mysql 5.7 源

wget https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm

# 导入mysql 5.7 源

yum -y localinstall mysql57-community-release-el7-11.noarch.rpm

# yum 直接安装

yum -y install mysql-community-server

systemctl start mysqld

grep 'temporary password' /var/log/mysqld.log

如果没有初始化密码显示，就要修改配置文件。

systemctl stop mysqld

vim /etc/my.cnf // 修改mysql配置文件

添加到[mysqld]模块中 skip-grant-tables // 跳过权限表

systemctl restart mysqld

mysql

update mysql.user set authentication\_string=password(‘123456’) where host=’localhost’ and user=’root’;

flush privileges;

exit;

systemctl stop mysqld

vim /etc/my.cnf

yum方式安装：删除 skip-grant-tables //跳过权限表

mysql -uroot

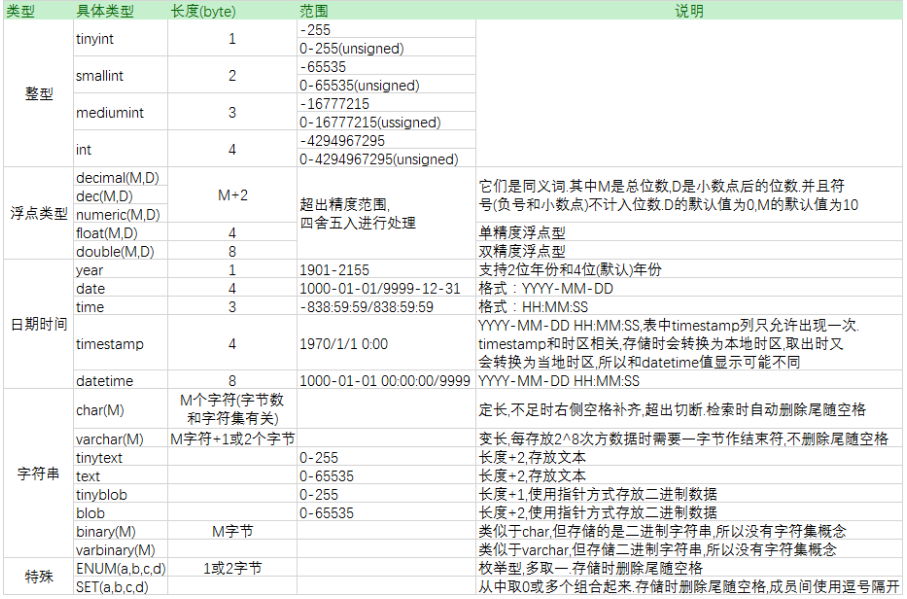
mysql> set password for 'root'@'localhost' = password('123456'); //设置登录密码

mysql> grant all privileges on \*.\* to root@'%' identified by '123456'; //设置远程连接及密码

delete from user where user=’’; // 删除初始化时创建的空用户

flush privileges; // 修改的配置立即生效

exit;



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

源码方式安装mysql：

yum install lrzsz -y

rz 上传 mysql\_5.7.21.tar.gz

rz 上传boost\_1.59.0.tar.gz

tar -xzf mysql\_5.7.21.tar.gz

tar -xzf boost\_1.59.0.tar.gz

mv boost\_1\_59\_0 /usr/local/boost

yum install -y gcc-c++ ncurses-devel cmake make perl gcc autoconf automake zlib libxml libgcrypt libtool bison

useradd mysql

id mysql # 查询mysql的id号

mkdir -p /data/mysql &&  chown  mysql:mysql -R   /data/mysql/

mkdir -p /usr/local/mysql5 && chown mysql:mysql -R /usr/local/mysql5/

# cd /usr/local/mysql5 &&   chown mysql:mysql -R .

# 修改完权限和用户之后，手动配置mysql5.7 版本的默认配置文件my.cnf

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

[root@localhost mysql5]# vim /etc/my.cnf

[mysqld]

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

basedir = /usr/local/mysql5

datadir = /data/mysql

port = 3306

# server\_id = .....

socket = /tmp/mysql.sock

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

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

#修改安全启动mysql的配置

[mysqld\_safe]

log-error=/data/mysql/mysql.log

pid-file=/data/mysql/mysql.pid

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

!includedir /etc/my.cnf.d

cd mysql\_5.7.21

cmake . -DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql5/ \

-DMYSQL\_UNIX\_ADDR=/tmp/mysql.sock \

-DMYSQL\_DATADIR=/data/mysql/ \

-DSYSCONFDIR=/etc \

-DMYSQL\_USER=mysql \

-DMYSQL\_TCP\_PORT=3306 \

-DWITH\_XTRADB\_STORAGE\_ENGINE=1 \

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \

-DWITH\_PARTITION\_STORAGE\_ENGINE=1 \

-DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1 \

-DWITH\_MYISAM\_STORAGE\_ENGINE=1 \

-DWITH\_READLINE=1 \

-DENABLED\_LOCAL\_INFILE=1 \

-DWITH\_EXTRA\_CHARSETS=1 \

-DDEFAULT\_CHARSET=utf8 \

-DDEFAULT\_COLLATION=utf8\_general\_ci \

-DEXTRA\_CHARSETS=all \

-DWITH\_BIG\_TABLES=1 \

-DWITH\_DEBUG=0 \

-DENABLE\_DTRACE=0 \

-DDOWNLOAD\_BOOST=1 \

-DWITH\_BOOST=/usr/local/boost

make && make install

# mysql 程序初始化的内容，mysql的默认程序表，系统表，用户表，都是在初始化的时进行创建的。

/usr/local/mysql5/bin/mysqld --initialize --user=mysql --datadir=/data/mysql  --basedir=/usr/local/mysql5

mysql5.7以下的版本初始化：

/usr/local/mysql5/scripts/mysql\_install\_db --user=mysql --datadir=/data/mysql/ --basedir=/usr/local/mysql5/

#制作软链接以便启动mysql

ln -s /usr/local/mysql5/bin/\*   /usr/bin/

#或者用系统环境变量设置启动路径

vim /etc/profile

export PATH=$PATH:/usr/local/mysql5/bin

source /etc/profile

# 制作mysql启动脚本

cd /usr/local/mysql5

\cp support-files/mysql.server /etc/init.d/mysqld   #强制复制

yes|cp support-files/mysql.server /etc/init.d/mysqld #把yes传到命令后面

chmod +x /etc/init.d/mysqld

chkconfig --add mysqld

chkconfig --level 35 mysqld on

service mysqld   restart 或者 systemctl restart mysqld

#如果restart 没有反应，就使用 /etc/init.d/mysqld 开启mysql服务

ps -ef|grep mysql

pkill mysql

service mysqld start

rm -rf /data/mysql/\*

/usr/local/mysql5/bin/mysqld --initialize --user=mysql --datadir=/data/mysql --basedir=/usr/local/mysql5

service mysqld restart

alter user 'root'@localhost identified by "123456"; #修改密码

flush privileges; #刷新内存 后生效

或者

use mysql;

update user set password=password(‘123456’) where user=’root’;

flush privileges;

/usr/local/mysql5/bin/mysqld\_safe  --user=mysql --skip-grant-tables &

systemctl start mysqld

mysql -uroot -p //要求输入mysql密码(123456) 登录 mysql

select version(); //查看当前版本号

show databases; //查看当前已存在的数据库名称

show tables from mysql; //查询数据库中的所有表

show engines; //查询引擎

alter table ‘table name’ engine=myisam; //修改引擎

show create table ‘table name’; //查询创建表的sql语句

show variables; # 查看所有的参数信息

mysql -e "show variables;" >>mysql.conf # 把数据库优化参数导出到文件中

desc mysql.user; // 查询表结构

select host,user,password from mysql.user; // 查询数据库表里的信息

yum 初始化 myql\_secure\_installation

yum remove mysql --nodeps // 删除mysql 而不删除其依赖关系

rm -rf /data/mysql/\*

mysql -h 10.10.10.112 -uroot -p123456 // 跨主机 连接数据库

rpm -ql nginx |grep conf

binlog日志是记录数据库，引起数据变化的sql语句

分析bin-log 与relay-log日志：

mysqlbinlog mysql-bin.000001 查看二进制日志文件

进入mysql以后执行

只查看第一个binlog文件的内容  
mysql> show binlog events;  
查看指定binlog文件的内容  
mysql> show binlog events in 'mysql-bin.000001';

获取binlog文件列表

mysql> show binary logs;

binlog三种模式：

row: 日志中会记录成每一行数据被修改的形式，然后在slave端再对相同的数据进行修改，只记录要修改的数据，只有value，不会有sql多表关联的情况。

statement: 每一条会修改数据的sql都会记录到master的binlog中，slave在复制的时候sql进程会解析成和原来master端执行的相同的sql再执行

mixed: 前两种模式的结合, 根据执行的每一条具体的 SQL 语句来区分对待记录的日志形式, 比如遇到表结构变更的时候就会以 statement 模式来记录，如果 SQL 语句确实就是 update 或者 delete 等修改数据的语句，那么还是会记录所有行的变更

Mysql BInlog日志格式可以通过mysql的my.cnf文件的属性binlog\_format指定。如以下：

binlog\_format           = MIXED                 //binlog日志格式

log\_bin                   =目录/mysql-bin.log    //binlog日志名

expire\_logs\_days     = 7                //binlog过期清理时间

max\_binlog\_size      100m                    //binlog每个日志文件大小

/usr/local/mysql5/bin/mysqld\_safe --user=mysql &

##源码编译的mysql通过这种方式启动，mysqld\_safe默认启动是在前台运行，前台运行的缺点是，当前shell终端关闭，服务就关闭了。所以要放入后台运行！

find / -name “\*mysql\*” -exec rm -rf {} \;

#删除所有与mysql相关的文件, -exec是一个参数一个参数传递删除

find / -iname “mysql” | xargs rm -rf

#xargs 是给命令传递参数的一个过滤器，也是组合多个命令的一个工具，可以将管道或标准输入数据转换成命令行参数，也能够从文件的输出中读取数据。它能捕获一个命令的输出，然后传递给另外一个命令。

# mysqldump备份工具

# 先把mysql导出，然后导入到另一台机器：

mysqldump -uroot -p discuz > discuz.sql

# 导出所有数据库

mysqldump -uroot -p123456 --all-databases --opt --single-transaction > 2019all.sql

--single-transaction参数的作用，设置事务的隔离级别为可重复读，即REPEATABLE READ，这样能保证在一个事务中所有相同的查询读取到同样的数据，也就大概保证了在dump期间，如果其他innodb引擎的线程修改了表的数据并提交，对该dump线程的数据并无影响，在这期间不会锁表。

scp -r discuz.sql root@192.168.11.200:/root/

相当于

rsync -aP --delete ./wordpress/ root@192.168.11.200:/usr/local/nginx/html/wordpress/

// rsync可实现增量拷贝 -P显示同步的进程 --delete删除目标目录与源文件不一样的内容

rsync -av file ip:/data/ // -a实现递归，保留文件属性 -v显示详细过程

rsync -av dir ip:/data/

mysql -uroot -p123456

create database discuz charset=utf8;

# 导入之前备份的数据库脚本

mysql -uroot -p discuz < discuz.sql

# 子查询

select name from (select name,min(score) from student group by name having min(score)>80) stu;

delete from student2 where id not in (select a.id from (select id from student2 group by no,name,coursename,courseno,score) a);

# xtrabackup备份工具

上传percona-xtrabackup-2.4.9

cd percona-xtrabackup-2.4.9/bin/

cp innobackupex /usr/bin/innobackupex

cp xtrabackup /usr/bin/xtrabackup

yum install -y libaio\*

mysql -uroot -p123456

grant all on \*.\* to backup@'192.168.0.161' identified by "123456";

exit;

innobackupex  --defaults-file=/etc/my.cnf  --host=192.168.0.161 --user=backup  --password="123456"  /data/backup/mysql/

# 数据库恢复,先检查数据一致性

innobackupex --defaults-file=/etc/my.cnf --user=backup --password="123456" --apply-log /data/backup/mysql/2019-12-25\_12-15-04

# 删除mysql数据目录的数据

cat /etc/my.cnf

datadir=/data/mysql

rm -rf /data /mysql/\*

innobackupex --defaults-file=/etc/my.cnf --user=backup --password="123456" --copy-back /data/backup/mysql/2019-12-25\_12-15-04

chown -R mysql:mysql /data/mysql/

df -h

文件系统 容量 已用 可用 已用% 挂载点

devtmpfs 1.9G 0 1.9G 0% /dev

tmpfs 1.9G 0 1.9G 0% /dev/shm

tmpfs 1.9G 12M 1.9G 1% /run

tmpfs 1.9G 0 1.9G 0% /sys/fs/cgroup

/dev/sda2 19G 5.8G 12G 34% /

/dev/sda1 453M 186M 240M 44% /boot

tmpfs 378M 0 378M 0% /run/user/0

# 进入本地磁盘后写文件测试读写速度

cd /data/

dd if=/dev/zero of=test.img bs=1M count=1000

# 进入内存后写文件测试读写速度

cd /dev/shm/

dd if=/dev/zero of=test.img bs=1M count=1000

for i in $(seq 1 1000);do mysql -e 'insert into test.t1 values('','warning');done'

mysql优化四个方面：系统内核参数，硬件配置，mysql配置文件，sql执行脚本

###############################################################

set 设置 shell

set -C 禁止使用输出重定向覆盖已存在的文件 ; 如果想强制覆盖输出，使用>|

set +C 关闭上诉功能

echo “hello”>>1.txt 可以追加不能覆盖

###所有以curl www.jfedu.com/index.php结尾的访问都拒绝

location ~\* .php$ {

deny all;

}

wget http://nchc.dl.sourceforge.net/project/boost/boost/1.59.0/boost\_1\_59\_0.tar.gz

vim 一共有12种模式。

常用的三种模式：编辑模式 i 命令模式 esc 末行模式 : wq! q!

分屏模式:

在末行模式下输入:vs 文件名/路径 垂直分屏 :sv 文件名/路径 水平分屏

切屏:ctrl+w+w

用鼠标实现分屏: 在命令模式下输入 :set mouse=a

可视化模式: 在命令模式下，按ctrl + v 进入。用方向键来选择需要操作的区域，然后按下大写的I (i),进入编辑模式，输入 #，再按esc退出编辑模式，即可实现多行注释。

取消多行注释则需要在命令模式下，按ctrl + v, 然后选择多行注释符#，再按d,即可取消。

查找命令 /string,?string #从光标所在处向前/向后查找字符串内容

拷贝复制命令 yy,p #拷贝一行到剪贴板/取出剪贴板中的内容

ZZ 保存并退出 GG 定位到最后一行 gg 定位到第一行

末行模式 :%s/old/new/g

vim 末行模式下 : set nu 显示当前文本行数 : set nonu 取消显示当前文本行数

 用 vi 多行注释   
如果要注释几百行，一个聪明的办法是：   
:.,+499 s/^/#/g

dd // vim 命令模式快速删除光标所在行

!.. //调用 上一个点点开头的命令

chown jfedu1:jfedu1 1.txt 和 chown jfedu. 1.txt 是一样的,更改1.txt的所属主和所属组

chmod u+s /usr/bin/netstat // 让文件有属主的权限，而不是当前执行用户的权限

su ‘user’

netstat -ntlp

如果是要特殊权限，可以在/etc/sudoers文件中添加www用户相关信息，或者将www加入wheel组。如果是想使用某个用户的权限，可以使用su - 用户 -c ‘命令

su -mysql -c ‘touch /tmp/mysql.txt’;

chmod g+s /data/ // g:group s: special 让用户拥有上级目录的所属组权限

chmod o+w /data/ // o :other w: write

chown -R mysql. /var/lib/mysql/ //-R 递归 授权mysql 用户来访问

chown -R mysql:mysql /var/run/mariadb/ // 授权mysql用户来访问

sestatus // 查看selinux 的状态

man setfacl // 查看相关命令

setfacl可以更精确的控制权限的分配。

比如: 让某一个用户对某一个文件具有某种权限; 让某一个文件, 不让单一的某个用户访问

用法: setfacl [-bkndRLP] { -m|-M|-x|-X ... } file ...

-m, --modify-acl 更改文件的访问控制列表

-M, --modify-file=file 从文件读取访问控制列表条目更改

-x, --remove=acl 根据文件中访问控制列表移除条目

-X, --remove-file=file 从文件读取访问控制列表条目并删除

-b, --remove-all 删除所有扩展访问控制列表条目

-k, --remove-default 移除默认访问控制列表

--set=acl 设定替换当前的文件访问控制列表

--set-file=file 从文件中读取访问控制列表条目设定

--mask 重新计算有效权限掩码

-n, --no-mask 不重新计算有效权限掩码

-d, --default 应用到默认访问控制列表的操作

-R, --recursive 递归操作子目录

-L, --logical 依照系统逻辑，跟随符号链接

-P, --physical 依照自然逻辑，不跟随符号链接

--restore=file 恢复访问控制列表，和“getfacl -R”作用相反

--test 测试模式，并不真正修改访问控制列表属性

-v, --version 显示版本并退出

-h, --help 显示本帮助信息

————————————————

type setfacl

setfacl –help

setfacl -m u:lutixia2:r,u:lutixia3:rw jfedu.txt

getfacl jfedu.txt

setfacl -m u:wangwei:rw- test.txt # 修改文件的acl权限, 添加一个用户权限

getfacl test.txt

setfacl -m g:wangwei:rw- test.txt # 添加一个组权限

getfacl test.txt

setfacl -m u:wangwei:rx test.txt # 给用户向test文件增加读和执行权限

getfacl test.txt

vim acl.txt # 写入设置权限的规则

setfacl -M acl.txt jfedu.txt # 从文件中读取acl规则,对文件进行设置

setfacl -x u:lutixia jfedu.txt # 清除lutixia用户对jfedu.txt的所有权限

setfacl -b jfedu.txt # 清除所有acl规则

getfacl jfedu.txt |setfacl --set-file=- jfedu3.txt //将jfedu.txt的权限复制给jfedu3.txt

getfacl jfedu3.txt

chattr // 主要针对日志文件

lsattr //查看文件权限

> /var/log/nginx/access.log //清空日志

chattr +a /var/log/nginx/access.log //让文件只能追加内容,不能修改,删除,强制删除

chattr -a /var/log/nginx/access.log

tail /var/log/nginx/nginx/access.log //查看访问记录

chattr +i acl.txt // 防止文件被删除

chattr -i acl.txt

lsattr acl.txt

rm -rf acl.txt // 提示无法删除

getenforce //查看防火墙状态

shell 文本中 这些字符需要转义\ $  `  "  \

ls > nor.txt // 重定向

0 标准输入

1 标准输出

2 标准错误

& 标准正确和标准错误输出

2>/dev/null 把错误输出扔进黑洞，不显示在屏幕上

ls jfedu6.txt asdfg > nor.txt 2 > error.txt

tail error.txt

ls jfedu6.txt asdfg &> all.txt

tail all.txt

ls > /dev/pts/3

ls > /dev/null

cp -S “.bak” error.txt /root/

`date +%F`

日志切割：

在/usr/local/nginx/logs 下操作

mv host.access.log{,.`date +%F`} //将原文件备份生成一个以日期结尾的新文件

ls

nginx -s reopen

cat host.access.log

mv /tmp/tomcat.png . //移动文件到当前目录下

watch 以周期性的方式执行给定的指令，指令输出以全屏方式显示。Watch可以监测一个命令的运行结果，省去一遍遍手动运行的麻烦。

watch -n 1 -d netstat -ant #每隔1秒高亮显示网络链接数的变化

watch -n 10 ‘cat /proc/loadavg’ #每隔10秒输出系统的平均负载

scp -r 10.0.0.7:/usr/local/nginx . #把10.0.0.7的nginx拷贝到当前虚拟机下

find / -name "\*mysql\*" -exec rm -rf {} \; # 卸载mysql 所有相关文件

sudo mkdir -p ./mydir/dir\_test/ #允许创建目录和子目录

mkdir -m 777 mydir #创建指定权限的目录

[**linux清空日志文件内容 比如log日志**](https://www.cnblogs.com/renmengkai/p/9452857.html)

权限要求： 至少执行用户对该文件有写的权限

--w------- 1 QA\_Deploy QA\_Deploy       1 Jun 27 17:35 test.out

$ : > filename   
$ > filename   
$ echo "" > filename   
$ echo > filename   
个人推荐最简单的方式，直接 > filename.

执行操作后，细心操作可以发现文件大小是不一样的。

1和2 两种方式，是0kb

3和4 两种方式，是1kb

还有一种方式为：cat /dev/null > file.log

这条命令的意思是：清空messages文件。

可以把/dev/null看作"黑洞". 它非常等价于一个只写文件. 所有写入它的内容都会永远丢失. 而尝试从它那儿读取内容则什么也读不到.

因此， /dev/null 对命令行和脚本都非常的有用

yum 方式安装 php:

配置/etc/yum.repos.d/remi-php71.repo && remi-safe.repo

yum install php php-devel php-fpm php-mysql -y

systemctl start php-fpm.service #启动php-fpm模块

netstat -ntlp|grep 9000 #查看端口是否开启

nohup /usr/sbin/php-fpm & # 后台启动php-fpm

或者 使用 yum install -y nc

nc -zv 192.168.0.161 9000 #查看远程主机是否开启了9000端口

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

源码方式安装 PHP

yum install -y gd gd-devel curl curl-devel libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel mysql-devel libmcrypt-devel

#安装编译所需程序包与依赖包

wget http://museum.php.net/php5/php-5.6.9.tar.gz

tar xzf php-5.6.9.tar.gz

cd php-5.6.9

make clean

./configure --prefix=/usr/local/php5 \  
--enable-fpm \

--enable-sockets \

--enable-bcmath \  
--enable-debug \

--enable-xml \

--enable-mbstring \

--with-zlib \

--with-pear \

--with-kerberos \

--with-pdo-mysql \

--with-pdo-sqlite \

--with-openssl \

--with-pcre-regex \

--with-gd \

--with-gettext \  
--with-jpeg-dir \  
--with-png-dir \  
--with-freetype-dir \  
--with-libxml-dir \

--with-xmlrpc \  
--with-curl \

--with-mysql=/usr/local/mysql5/ \  
--with-mysqli=/usr/local/mysql5/bin/mysql\_config \

--disable-fileinfo \

--with-config-file-path=/usr/local/php5/etc \

--disable-fileinfo \

--with-mysql=mysqlnd \

--with-mysqli=mysqlnd \

--with-pdo-mysql=mysqlnd \

--with-config-file-path=/usr/local/php5/etc \

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

mysql 与 php分离不在同一台机的时候,编译方案

./configure --prefix=/usr/local/php5 \

--enable-fpm \

--enable-debug \

--with-gd \

--with-jpeg-dir \

--with-png-dir \

--with-freetype-dir \

--enable-mbstring \

--with-curl \

--with-mysql=mysqlnd \ 是用于底层通信

--with-mysqli=mysqlnd \ 是用于数据传输，保证数据稳定传输

--with-pdo-mysql=mysqlnd \ 是用于底层驱动

--with-config-file-path=/usr/local/php5/etc 指定配置文件路径

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

make && make install

#编辑PHP文件权限与配置文件

\cp php.ini-development /usr/local/php5/lib/php.ini

\cp /usr/local/php5/etc/php-fpm.conf.default /usr/local/php5/etc/php-fpm.conf

\cp /root/php-5.6.9/sapi/fpm/init.d.php-fpm /etc/init.d/php-fpm

#设置系统服务

chmod 755 /etc/init.d/php-fpm

chkconfig --add php-fpm

chkconfig php-fpm on

vim /etc/profile

export PATH=$PATH:/usr/local/php5/bin #给环境变量加上PHP的环境变量

:wq

source /etc/profile

如果是跨主机访问，需要修改/etc/php-fpm.d/www.conf

listen = 192.168.0.161:9000

……

listen.allowed\_clients=192.168.0.200 # 默认允许所有主机访问

vim /usr/local/nginx/conf/nginx.conf

http {

include mime.types;

default\_type application/octet-stream;

server {

listen 80;

server\_name localhost;

#location ~ .\*\.(html|htm|js|png|jpeg|txt|css|gif)$ {

#root html;

#}

location / {

root /usr/local/nginx/html;

index index.html index.php;

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000; #指定了 fastcgi 进程侦听的端口,nginx 就是通过这里与php交互的

fastcgi\_index index.php; #脚本请求的前缀名

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

#脚本文件请求的路径，这个配置的意思是 在浏览器中访问的.php 文件，实际读取的

是 $document\_root( 网 站 根 目 录 ) 下 的 .php 文 件 -- 也 就 是 说 当 访 问 127.0.0.1/index.php 的时候，需要读取网站根目录下面的 index.php 文件，如果没有配 置这一配置项时，nginx 不会去网站根目录下访问.php 文件，所以返回空白

##$fastcgi\_script\_name 变量的值中设置一个文件名(是.php 结尾的文件)， 该文件名将附加在以斜杠结尾的 URI 之后

include fastcgi\_params;

}

}

}

systemctl start php-fpm.service #启动php服务

service php-fpm restart #重启 php 服务

nginx -s reload #重载nginx

ps -ef|grep php #查看php是否运行

/usr/local/nginx/sbin/nginx –t 检测 Nginx 服务参数

/usr/local/nginx/sbin/nginx 启动 nginx 服务参数

cat >/usr/local/nginx/html/index.php<<EOF

<?php

phpinfo();

?>

EOF

本机hosts文件路径 C:\windows\system32\drivers\etc\

把 linuxfile1 的档案内容加上行号后输入 linuxfile2 这个档案里

cat -n linuxfile1 > linuxfile2

把 linuxfile1 和 linuxfile2 的档案内容加上行号(空白行不加)之后将内容附加到 linuxfile3 里。

cat -b linuxfile1 linuxfile2 >> linuxfile3

cat /dev/null > /etc/test.txt 此为清空/etc/test.txt档案内容

浏览器输入本机Ip/index.php 测试 是否能显示 PHP页面

如果要实现 主机A(nginx) 🡪 主机B（PHP）🡪主机B（mysql）🡪主机A/B（discuz）

需要在 主机B上 配置

vim /usr/local/php5/etc/php.fpm.conf

listen = 192.168.11.161:9000 # 此为本机IP地址和监听端口

……

listen.allowed\_clients = 192.168.11.200 #此为发送调用PHP请求的主机IP

systemctl restart php-fpm

或者 vim /etc/php-fpm.d/www.conf

listen = 192.168.11.161:9000

……

listen.allowed\_clients = any #允许所有客户端访问

listen.allowed\_clients = 192.168.11.161,192.168.11.162 #允许多台客户端访问

主机A 和 主机B 上面 都需要在 /usr/local/nginx/html 下面放置 网站的程序

如果要删掉现有的源码安装的php重新安装,就需要删掉以前的编译文件

make clean

再重新编译

enable是软件已经有的功能 可以开启可以不开启，--with相当于第三方扩展  比如安装php 参数--with-curl 你不指定curl路径肯定安装过程肯定报错，会提示configure: error: Please reinstall the libcurl distribution -easy.h should be in <curl-dir>/include/curl/ 这个错误， 一般提前yum install curl curl-devel -y  再 --with-curl就不用指定路径了

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

chmod adb filename

a,b,c 分别表示 user,group,other 的权限

r=4, w=2, x=1

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=5.

普通文件的默认权限是644，目录是755.

killall -0 nginx #检测程序是否存在 echo $? 1为不存在0为存在

etcdctl member list  # 查看节点信息

yum install gpm-libs -y # 安装 VIM 的依赖包

yum list installed |grep php # 查询已经安装的 PHP的各种包

yum localinstall -y filename # 安装本地的包

sed 处理文件文件，并输出到标准输出

-e 可以指定多上命令

-f 指定命令文件

-n 取消默认控制台输出，与p一起使用可打印指定内容

-i 输出到原文件，静默执行(修改原文件)

sed遵循简单的工作流：

读取（从输入中读取某一行）

执行（在某一行上执行sed命令）

显示（把结果显示在输出中）

默认是显示修改后内容，不会修改原文件，除非使用-i 参数

sed  -e ‘/^$/d’ -e’/^ \*#/d’ nginx.conf #删除空行和注释行

sed -e ‘s/#.\*//g’ -e ‘/^$/d’ nginx.conf #更改内核版本号

a\ 在当前行下面插入文本

i\ 在当前行上面插入文本

c\ 把选定的行替换成新的文本

d 删除选择的行

D 删除模板块的第一行

s 替换指定的字符

h 拷贝模板块的内容到内存中的缓冲区

H追加模板块的内容到内存中的缓冲区

g 获得内存缓冲区的内容，并替代当前模板块中的文本

G获得内存缓冲区的内容，并追加到当前模板块文本的后面

l 列表不能打印字符的清单

n 读取下一个输入行，用下一个命令处理新的行而不是用第一个命令

N追加下一个输入行到模板块后面并在二者间嵌入一个新行，改变当前行号码

p 打印模板块的行

P 打印模板块的第一行

q 退出sed

！表示后面的命令对所有没有被选定的行发生作用

= 打印当前行号码

# 把注释扩展到下一个换行符以前

^锚定行的开始如：/^sed/匹配所有以sed开头的行。

$锚定行的结束 如：/sed$/匹配所有以sed结尾的行。

.匹配一个非换行符的字符 如：/s.d/匹配s后接一个任意字符，然后是d。

\*匹配零或多个字符 如：/\*sed/匹配所有模板是一个或多个空格后紧跟sed的行。

[]匹配一个指定范围内的字符，如/[Ss]ed/匹配sed和Sed。

[^]匹配一个不在指定范围内的字符，如：/[^A-RT-Z]ed/匹配不包含A-R和T-Z的一个字母开头，紧跟ed的行。

\(..\）保存匹配的字符，如s/\(love\)able/\1rs，loveable被替换成lovers。

&；保存搜索字符用来替换其他字符，如s/love/\*\*&\*\*/，love换成\*\*love\*\*。

\<；锚定单词的开始，如：/\<love/匹配包含以love开头的单词的行。

\>；锚定单词的结束，如/love\>/匹配包含以love结尾的单词的行。

x\{m\}重复字符x，m次，如：/0\{5\}/匹配包含5个o的行。

x\{m,\}重复字符x，至少m次，如：/o\{5,\}/匹配至少有5个o的行。

x\{m,n\}重复字符x，至少m次，不多于n次，如：/o\{5,10\}/匹配5--10个有o的行

sed ':a;N;s/\n/ /;ba;' list :a 冒号a的意思是定义了一个标签a 也可以写成:f 都可以 ,自己定义 。 然后执行 N;s/\n/ / 意思是合并并将换行替换为空格 N是追加下一行的内容到当前行,也可以 N;s/\n/ \t/ 然后b相当于c语言的goto语句，可以跳到标签除在执行内部语句，实际就是循环执行

mv /usr/local/nginx/html/{mysql.php,.bak} #制作一个名为.bak的文件,逗号分开两个文件

rpm -ql nginx|grep conf # 查询nginx 中的 conf 文件

  location ~ .\*\.(html|htm|js|png|jpeg|txt|css|gif)$ #匹配静态文件  
        {  
             root  html;  
        }

:%s/wp.jf.com/wp/g # vim命令行 将wp.jf.com 全部替换成wp

/dev/sdc1 /mnt xfs

vim /etc/vimrc 在 最后一行 加入 syntax on 即可高亮显示语法

在vim命令行模式 按下 shift+% 可以定位相应字符

在vim命令行模式 按下 小写u 即可 撤消；按下ctrl+r 即可恢复撤销

检测index用户认证：

location /img {

auth\_basic “auth”;

auth\_basic\_user\_file /usr/local/nginx/conf/passwd;

autoindex on;

}

mkdir – p /usr/local/nginx/html/img

cp /usr/local/nginx/html/night.jpg /usr/local/nginx/html/img

mv /usr/local/nginx/html/img{night.jpg,jfedu.jpg}

htpasswd -c /usr/local/nginx/conf/passwd admin

new password: admin

re-type new password: admin

cat /usr/local/nginx/conf/passwd

nginx -s reload

ll -d /usr/local/nginx/html/img/

ll /usr/local/nginx/html/img/

浏览器访问 localhost/img 验证

mysql 对客户端的授权：

1. 操作命令，普通运维一般只有select 权限。
2. 可以操作某个数据库或者单独数据库的某张表。
3. 给来访的客户端授权并创建访问的账号和密码

grant all on jfedu.\* to test@localhost identified by ‘password’;

grant select,insert,update,delete on \*.\* to test@”%” identified by ‘password’;

#创建数据库并授权哪台客户端来访问

mysql -uroot -p123456

create database wordpress charset=utf8;

grant all on wordpress.\* to wordpress@’192.168.11.162’ identified by “123456”;

flush privileges;

配置虚拟主机：

cd /usr/local/nginx/

ls

mkdir vhosts

cd vhosts

vim dz.jf.com

#写入nginx.conf中dz.jf.com的配置代码

cp dz.jf.com wp.jf.com

#写入nginx.conf中wp.jf.com的配置代码

grep -vE “#|^$” nginx.conf

vim /usr/local/nginx/conf/nginx.conf

include vhosts/\*;

mysql 主从同步操作：

主库修改my.cnf

[mysqld]

server-id=1

log-bin=mysql-bin

……

从库修改my.cnf

[mysqld]

server-id=2

log-bin=mysql-bin

……

主库：

grant replication slave on \*.\* to ‘sync’@192.168.0.162 identified by “123456”;

show master status;

从库：

stop slave;

change master

to master\_host=’192.168.0.161’,master\_user=’sync’,master\_password=’123456’,master\_log\_file=’mysql-bin.000001’,master\_log\_pos=617;

start slave;

show slave status\G #查询从库的链接信息

在主库上创建数据库来测试主从同步

creat database jfedu01 charset=utf8;

在从库上查询 jfedu01 的数据库

show databases;

drop database jfedu01;

锁表：

flush tables with read lock; #终端不能退出，退出就失效了

解锁表：

unlock tables;

解除主从关系：

stop slave;

reset slave all;

show slave status\G

reset master;

show master status;

查询被锁的库

show status like ‘innodb\_row\_lock%’;

create table t1(id int auto\_increment primary key,name varchar(20));

#设置主键

Got fatal error 1236 from master when reading data from binary log: ‘Client requested master to start replication from impossible position; the last event was read from’./mysql-bin.000006’ at 9985939, the last byte read was read from ‘./mysql-bin.000006’

由于master要删除一条记录，而slave上找不到故报错，这种情况主上都将其删除了，那么从机可以直接跳过。可用命令：

stop slave;  
set global sql\_slave\_skip\_counter=1;  
start slave;

df -h

free -m

cd /data/ # 硬盘设备

dd if=/dev/zero of=test.img bs=1M count=1000

#dd 测试读写性能 if 输入文件 of 输出文件 bs 写入文件的大小 count 写入的次数

cd /dev/shm # 内存设备

dd if=/dev/zero of=test.img bs=1M count=1000

for i in `seq 1 100000`;do mysql -e ‘insert into test.t1 values(‘’, ‘wanger’ );’;done

#向t1表中插入十万条数据

delete from user where host=”192.168.10.73”; #删除用户表中主机名为192的整行数据

yum install -y nfs nfs-utils #安装nfs网络文件系统

vim /etc/exports

/usr/local/nginx/html/ \*(rw,no\_root\_squash,no\_all\_squash,sync)

systemctl restart nfs

systemctl restart portmap

mount -t nfs 192.168.0.200:/usr/local/nginx/html /mnt/

umount -f /mnt/ #强制卸载

服务端和客户端都需要安装 yum install -y nfs-utils

cat nginx.conf|egrep -v "#|^$" //过滤掉#和空行来查看

tail -f /usr/local/nginx/logs/error.log //动态查看文件

tail -fn 10 /usr/local/nginx/logs/access.log //动态查看访问日志后10行

yum provides ifconfig //查看 ifconfig 工具是由哪个包提供的

源码安装redis：

rz redis-4.0.11.tar.gz

tar xzf redis-4.0.11.tar.gz

mkdir -p /usr/local/redis-4.0.11.tar.gz

cd /root/redis-4.0.11

make

make install PREFIX=/usr/local/redis-4.0.11/

make test

yum install -y tcl

make test

cp /root/redis-4.0.11/redis.conf /usr/local/redis-4.0.11/bin/

cd /usr/local/redis-4.0.11/bin/

vim redis.conf

/bind

bind 0.0.0.0

:wq

./redis-server redis.conf #启动redis 服务

./redis-cli

redis-cli monitor

#修改redis.conf的参数配置

vim redis.conf

#查找daemonize no改为 yes以守护进程方式运行 即以后台运行方式去启动

daemonize yes

#修改dir ./为绝对路径, 默认的话redis-server启动时会在当前目录生成或读取dump.rdb 所以如果在根目录下执行redis-server /etc/redis.conf的话  
#, 读取的是根目录下的dump.rdb,为了使redis-server可在任意目录下执行 所以此处将dir改为绝对路径

dir /usr/local/redis-4.0.11/bin

#修改appendonly为yes

#指定是否在每次更新操作后进行日志记录， Redis在默认情况下是异步的把数据写入磁盘，

#如果不开启，可能会在断电时导致一段时间内的数据丢失。 因为 redis本身同步数据文件是按上面save条件来同步的，   
#所以有的数据会在一段时间内只存在于内存中。默认为no

appendonly yes   
#redis 日志生成位置  
logfile "/usr/local/redis-4.0.11/log/redis.log"

./redis-server redis.conf

./redis-cli

#使用redis启动脚本设置开机自启动

cp /root/redis-4.0.11/utils/redis\_init\_script /etc/init.d/redisd-6379

vim /etc/init.d/redisd-6379

#修改启动文件的路径

chkconfig redisd-6379 on

service redisd-6379 start

service redisd-6379 stop

#设置系统环境变量以便 使用redis-cli 直接启动客户端

vim /etc/profile

export PATH=$PATH:/usr/local/redis-4.0.11/bin/

redis-cli

info

set web1 www.baidu.com

get web1

values

FLASHALL # 清除所有key

keys \* # 查询所有key

访问一下网站，就可以重新获取网站缓存到redis的key

keys \*

#在网站代码的config\_global.php配置文件中配置redis相关信息

----CONFIG MEMORY----

$\_config['memory']['redis']['server'] = '192.168.0.200';

$\_config['memory']['redis']['port'] = 6379;

yum install -y php-redis # 安装redis连接驱动

systemclt restart php-fpm

查看info.php页面信息中 是否包含redis模块

<?php

phpinfo();

?>

# redis主从配置:

# 在redis从库上修改配置文件vim /etc/redis.conf

slaveof 192.168.0.161 6379 # 配置主库的IP和端口

slave-read-only yes # 设置从库为只读

systemctl restart redis

# 在redis主库上修改配置文件vim /etc/redis.conf

bind 0.0.0.0

systemctl restart redis

在主库上新建key测试主从同步

set jfedu666 www.jfedu.com

get jfedu666

在从库上操作

redis-cli monitor # 监控从库的操作

redis-cli

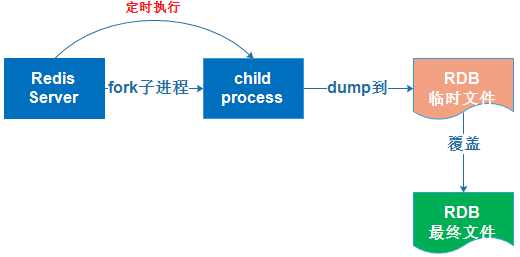
get jfedu666

ls /var/lib/redis/ # redis硬盘存储数据目录

备份 dump.rdb 就是redis 的数据库

redis 只有当内存数据有变化时 才会同步到硬盘上, 也可以 手动触发备份

RDB持久化是指在指定的时间间隔内将内存中的数据集快照写入磁盘，实际操作过程是fork一个子进程，先将数据集写入临时文件，写入成功后，再替换之前的文件，用二进制压缩存储。



AOF持久化以日志的形式记录服务器所处理的每一个写、删除操作，查询操作不会记录，以文本的方式记录，可以打开文件看到详细的操作记录。

redis是单线程，rdb是半持久化，时间不连续；aof全持久化，每次修改都做记录。

redis-cli

bgsave 叉出一个子进程, 子进程负责调用rdbsave, 并在保存完成之后向主进程发送信号, 通知保存已完成. 由于rdbsave在子进程被调用, 所以redis服务器在bgsave执行期间仍然可以继续处理客户端的请求.( 一般使用这个命令手动保存数据 )

或者

save 直接调用rdbsave, 阻塞redis主进程, 直到保存完成为止, 主进程阻塞期间, 服务器不能处理客户端的任何请求, 如果数据量比较大时, 会导致卡死.

vim /etc/redis.conf

save 900 1 # 900秒内变更了1个key就同步数据到硬盘

save 300 10 # 300秒内变更了10个key就同步数据到硬盘

save 60 10000 # 60秒内变更了10000个key就同步数据到硬盘

redis默认有16个库:

redis-cli

select 0

select 15

redis启用多实例

/usr/bin/redis-server /etc/redis\_7000.conf

/usr/bin/redis-server /etc/redis\_7001.conf

/usr/bin/redis-server /etc/redis\_7002.conf

firewall-cmd --permanent --add-port/tcp=80 #centos7 开启防火墙80端口

grep -C5 "wp.jf.net" nginx.conf #从nginx.conf中查询wp.jf.net 字符串所在的位置

常用linux监控工具 Zabbix Nagios Prometheus

wget -C https://jaist.dl.sourceforge.net/project/jforum2/jforum-2.5.0.war

/usr/local/jdk1.8.0\_131/bin/jar -xf jforum-2.5.0.war #手动解压

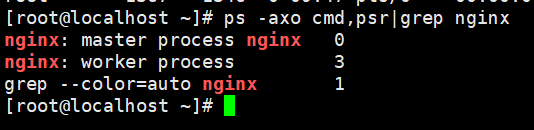
cp ifcfg-ens33{,.bak} #复制一份并改名后缀为.bak的文件

\cp ./webapps/\* /data/webapps/ -a # -a保持源目录的属性不变

<Context path="" docBase="/data/tomcat\_root/v1-1.jf.com" debug="0" reloadable="true" />

 在server.xml 的host段里面加这么一个配置就可以更改默认发布路径

查看进程对应的cpu 是哪一核在处理，0是第一核



ajp://192.168.11.161 #apache 使用的

yum install httpd-tools -y

ab -c 1000 -n 10000 http://localhost:8080/

chown -R apache.apache conf

#设置虚拟机 DNS

vm /etc/resolv.conf  
# Generated by NetworkManager  
nameserver 114.114.114.114

#还原文件和文件夹初始权限

find . -type f -exex chmod 644 {} -R \;

find . -type d -exec chmod 755 {} -R \;

系统变量 都是系统赋值

sh nginx.sh www.jf.net www.jd.com

$0 是文件名

$1 是文件名后第一个值

$2 是文件名后第二个值

依次类推$0,$1,……$9

$# 统计 变量的值 的个数

$\* 统计 所有的变量的值

$? 显示上一次执行命令是否成功 0 表示 成功

$$ 显示程序的PID号

Declare #声明变量属性

ABC=123456

echo $ABC

set #查询用户定义的变量

unset ABC #取消用户定义的变量

env #查看当前系统的环境变量

shell编程运算符：

-f     判断文件是否存在 eg: if [ -f filename ]  
-d     判断目录是否存在 eg: if [ -d dir     ]  
-eq    等于 应用于：整型比较  
-ne    不等于 应用于：整型比较  
-lt    小于  应用于：整型比较  
-gt    大于  应用于：整型比较  
-le    小于或等于 应用于：整型比较  
-ge    大于或等于 应用于：整型比较

-ne 不等于 应用于：整形比较  
-a    双方都成立（and） 逻辑表达式 –a 逻辑表达式  
-o    单方成立（or） 逻辑表达式 –o 逻辑表达式  
-z    空字符串

-n 字符串不为空

== 前后需要加空格 功能 和 -eq 相同

test 就是 if fi 中的 [ ]

(( )) 就是数学 运算符

( ) 就是命令组

sh -x auto\_install\_nignx\_v5.sh #打印执行过程

grep -v “#” nginx.conf|grep -v “^$” #过滤配置文件中以#号开头的行 -v 表示反向查找 ^表示以什么开头 $表示以什么结尾 ^abc$ 表示以abc开头，以abc结尾

ssh-keygen

ssh-copy-id -i /root/.ssh/id\_rsa.pub 118.31.55.30🡪(远程主机IP地址)

ssh -l root 118.31.55.30🡪(登录到远程主机中)

ssh -l root 118.31.55.30“df -h” 🡪(在远程主机中执行命令)

ssh -l root 118.31.55.30 “ls /tmp/;df -h;ifconfig” 🡪(在远程主机中同时执行多个命令)

ps -ef|grep sshd

kill -9 (pts/0 的进程ID) 🡪(断开远程主机的连接)

# pkill 接进程名 kill -9 接进程ID

for ip in `seq 140 143`;do echo -e "\033[32m10.10.10.$ip | CHANGED | rc=0 >>\033[0m";ssh -l root 10.10.10.$ip "df -h" ;done

expr `echo {1..100}|sed ‘s/ / + /g’` #expr 表示计算 后面的 公式

#备份数据库的授权

grant select,lock tables on \*.\* to backup@‘localhost’ identified by "backup123";

## echo命令详解

echo命令主要用于打印字符或者回显，例如输入echo ok，会显示ok， echo ok > test.txt 则会把ok字符覆盖test.txt内容。>表示覆盖，原内容被覆盖，>>表示追加，原内容不变。

例如echo ok >> test.txt,表示向test.txt文件追加OK字符，不覆盖原文件里的内容，常用参数详解如下:

|  |
| --- |
| 使用-e扩展参数选项时，与如下参数一起使用，有不同含义，例如：  \a 发出警告声  \b 删除前一个字符  \c 最后不加上换行符号；  \f 换行但光标仍旧停留在原来的位置；  \n 换行且光标移至行首；  \r 光标移至行首，但不换行；  \t 插入tab； \v 与\f相同；  \\ 插入\字符；  \033[30m 黑色字 \033[0m  \033[31m 红色字 \033[0m  \033[32m 绿色字 \033[0m  \033[33m 黄色字 \033[0m  \033[34m 蓝色字 \033[0m  \033[35m 紫色字 \033[0m  \033[36m 天蓝字 \033[0m  \033[37m 白色字 \033[0m  \033[40;37m 黑底白字 \033[0m  \033[41;37m 红底白字 \033[0m  \033[42;37m 绿底白字 \033[0m  \033[43;37m 黄底白字 \033[0m  \033[44;37m 蓝底白字 \033[0m  \033[45;37m 紫底白字 \033[0m  \033[46;37m 天蓝底白字 \033[0m  \033[47;30m 白底黑字 \033[0m |

echo `hostname` 和 echo $(hostname) 是一样的结果

sed -i ‘s/www/http:\/\/www/g’ list.txt sed -i ‘s#www#http://www#g’ list.txt 是一样的结果 -i表示编辑写入

sed -n /qq/p lisht.txt 筛选包含有qq字符的文件或目录 /p 表示 print

du -sh `find . -type f -size +10M` 查找当前目录下文件大小>10M 的所有文件

for i in `find . -type d ! -name "." -name "\*unix"`;do  rm -rf $i ;done

for i in `find . -type d ! -name "." ! -name "\*unix"`;do  rm -rf $i ;done

du -ah --max-depth 1 统计当前目录的 所有文件的大小

find . -maxdepth 1 -type f

find . -maxdepth 1 -type d

find / -name \*.rpm -name lib\* -type f -size +1k -size -200k -exec cp {} /tmp/ \;

-exec \; 和 |xargs -i 是一样的

find / -name \*.rpm -name libxll\* -type f -size +1k -size -200k |xargs -i rm -rf {}

echo www.jd.com|sed ‘s/com/net/g’

NGX\_SOFT=nginx-1.16.1.tar.gz

NGX\_SRC=`echo $NGX\_SOFT|sed ‘s/.tar.gz//g’` 把.tar.gz 换成 空

sed -n ‘2,5p’ list.txt 打印2行到5行的内容

sed -n ‘2p;5p’ list.txt 打印2行和5行的内容

sed -i '9,10d' list.txt 删除第9行到第10行

sed -i '2d;4d;6d' list.txt 删除指定的行

sed '/#/d' nginx.conf 删除带#号的行

sed ‘/^$/d’ nginx.conf 删除空行 ^表示以什么开头 $ 表示以什么结尾 ^#$ 表示以#开头并且结尾的行

sed ‘/2.100/s/^/&1)/g’ list.txt s替换 ^以什么开头 &添加 g全局

sed ‘/2.100/s/$/&./g’ list.txt $ 结尾

sed -e ‘/2.100/s/^/&1)/g -e /2.100/s/$/&./g’ list.txt

sed ‘s/^/&1)/g’ list.txt

sed ‘s/$/&./g’ list.txt

for i in $(seq 1 `cat list.txt|wc -l`);do sed -e “s/^/&$i/g” -e “s/$/&./g” list.txt ;done

sed ‘www.baidu.com/a’ 在后面添加

sed ‘www.baidu.com/i’ 在前面添加

sed -i -e ‘www.baidu.com/a\www.jf.net’ -e ‘www.tb.com/a\www.zj.net’ list.txt

sed '/^IPADDR.\*/s/IPADDR=10.10.10.1/disabled/g' ifcfg-eth0  .\* 匹配任意一个以上的字符

sed ‘/baidu/!s/com/net/g’ list.txt

sed ‘1,8s/com/net/g’ list.txt

sed -e ‘/baidu/!s/com/net/g’ -e ‘s/qq.com/qq.net/g’ list.txt -e添加多个执行命令

sed ‘s/^/&)/g’ list.txt|cat -n # &表示添加 cat -n 显示行号 grep -n 显示行号

sed = list.txt|sed ‘N;s/\n/)/g’ # sed = 显示行号

echo -e "www.aqy.com\nwww.yk.com\nwww.aaa.com">>list.txt \n 表示 换行符

seq 1 8|sed ‘N;N;N;s/\n//g’

sed -n ‘/port/p’ `find /usr/local -name server.xml`|sed ‘s/ /\n/g’|grep “port=”|sed ‘s/port=//g;s/”//g;’|sort -n|uniq|grep -v 8443

tail -n 50 /var/log/messages |grep -C 10 -i "failed" 不区别大小写,过滤failed的前后10行

grep -A -B -C after before center

grep -v -E "#|^$” nginx.conf ^$ 表示 空行  
grep -vE “#|^$” nginx.conf

netstat -ntlp|grep -w 80 –color word 包含80 单独这个字符 的 这一行 打印出来

for i in `grep -rl “localhost” \*|grep -v xml`;do sed -i ‘s/localhost/10.10.10.100/g’; echo $i ;done

cat >while\_v1.sh<<EOF

i=0

while ((\$i<=10))

do

echo -ne “\033[32m-\033[0m”

sleep 1

((i++))

done

echo

EOF

expr `echo {1..100}|sed ‘s/ / + /g’` #计算0到100的和

for i in $(seq 1 `cat list.txt|wc -l`);do sed -i “${i}s/^/&${i})/g” list.txt ;done

ifconfig|grep inet|sed ‘s/ /\n/g’|grep -vE “^$|255|net”

grep dd\* 正则表达式 一般是字符  忽略\*号 前面的最近的一个字符

ls dd\* 通配符 一般是词组

grep sed awk  vim location 都可以使用正则表达

.\* 表示 匹配任意字符

[^0-9] 表示 不匹配 0-9 只匹配 其它 字符

^[0-9] 表示 以 0-9 开头

grep “\<ww” list.txt 锚定以这个组合字符串开头的

grep “\<80” list.txt 锚定以80字符开头 包含80字符的都会匹配到

grep “\<80\>” list.txt 锚定以80字符开头并结尾 只匹配 80 字符

grep -E “[0-9]{3}” list.txt 匹配三次 连续的0-9

grep -E “[0-9]{4,}” list.txt 匹配四次以上 连续的0-9

grep -E “[0-9]{1,3}\>” list.txt 锚定最多只匹配三次连续的 0-9 {1,3} 分别 进行1次 2次 3次 匹配

grep -E “[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}” list.txt

grep -E “([0-9]{1,3}\.){3}[0-9]{1,3}” list.txt

echo 10.10.10.10|grep -E “^([0-9]{1,3}\.){3}[0-9]{1,3}$”

ifconfig|grep network|grep -E “([0-9]{1,3}\.)[0-9]{1,3}”

ifconfig|grep network|grep -oE “([0-9]{1,3}\.)[0-9]{1,3}”

cut是以每一行为一个处理对象的，这种机制和sed一样。  
cut接受三个定位方法：  
1）byte： -b  字节  
2）characters： -c  字符  
3）fields： -d  
  
cut -b 3-5 9    提取3,4,5和9字节 会自动从小到大排序   -3 是1-3   &&     3- 是3到末尾  
  
cut -c 1,2,3     如果有多字节字符 要加n   cut -nc 1,2,3 不会把多字节字符拆开   实测不懂-n区别   一个汉字算一个字符要用-c!  
  
cut -d “\.” -f 1    以.点为分隔划分区域 -f来选择区域 192.168.1.10 -f 1 表示取值192  
cut -d ‘’ -f 2  空格作为间隔 应该用‘’表示

cat /etc/passwd|cut -d":" -f1 #cut 分割 按冒号分割并取分割后第一段字符显示

awk -F: '{print $1}' /etc/passwd #以冒号为分割 打印 文件 中 第一列的值

df -h|awk 'NR>1&&NR<4 {print $1,$(NF-1)}' #打印大于1小于4的行中 第一列和倒数第二列的值

awk '{print"1)" $1,$3}' list01.txt #一对双引号表示 添加其中的内容 $1 $3表示第一列和第三列的值

awk 'NR==3{print $1};NR==5{print $1}' list01.txt>>123.txt #把第三行和第五行的第一列的值追加到123.txt

awk ‘gsub(/root/,”admin”)’ list01.txt 把找到的root 字符 换成 admin

awk ‘NR>1 {print $1}’ list01.txt

WEB=www.jd.com

awk ‘NR>1 {print $1”’$WEB’”}’ list01.txt #调用自定义变量 里面用单引号外面用双引号 单引号表示把内容变成字符 双引号表示添加该内容

awk -v WEB=www.jf.com ‘NR>1 {print $1,WEB}’ list01.txt

awk -F. ‘{num+=$1}END{print num}’ list01.txt #以.号来分割 从第一行开始计算第一列的值的总和 并打印总和

awk -F. ‘NR>1 {num=0,num=num+$1}END{print num}’ list01.txt

awk -F. ‘BEGIN{num=0}{num+=$1}END{print num}’ list01.txt #BEGIN表示开始赋值 END表示结束 不赋值就是0

awk ‘length($0)<10 {print}’ list01.txt # $0 表示匹配到的全部内容

awk -F: ‘{print (NR-1),$1}’ /etc/passwd #NR默认是1，以冒号分割,从第0行开始打印第一列的值

awk -F: ‘BEGIN{NR=-1}{print NR,$1}’ /etc/passwd # $NR 行数 $NF 最后一列

awk ‘NR%2==0 {next} {print NR,$0}’ /etc/passwd #行数除以2整除 就打印下一行，打印行号和内容

awk ‘NR%2==0 {print NR,$0}’ /etc/passwd

echo 100 200 300|awk ‘{ if((($1>$2))&&(($2>$3))) {print $1} else {print $3} }’

awk ‘{if($9~/404/)} print $1,$9’ /var/log/nginx/access.log|more # more 分页查看

awk ‘{if(($9==404)) print $1}’ /var/log/nginx/access.log|sort -n|uniq -c|sort -nr # sort -n 正向排序 sort -nr 反向排序 uniq -c去除重复项并统计去重的次数

awk ‘/Failed/’ /var/log/secure|grep -oE "([0-9]{1,3}\.){3}[0-9]{1,3}"|sort -n|uniq -c|sort -nr|awk ‘{if($1>=10) print $0}’

awk ‘/Failed/’ /var/log/secure|grep -oE "([0-9]{1,3}\.){3}[0-9]{1,3}"|sort -n|uniq -c|sort -nr|awk ‘{if($1>=8) print "sshd:"$2":deny"}’>>/etc/hosts.deny  拒绝以上IP通过远程访问本机

awk ‘/Failed/’ /var/log/secure|grep -oE "([0-9]{1,3}\.){3}[0-9]{1,3}"|sort -n|uniq -c|sort -nr|awk ‘{if($1>=8) print $2}’|sed -e ‘s/^/&sshd:/g’ -e "s/$/&:deny/g"

awk ‘/Failed/’ /var/log/secure|grep -oE “([0-9]{1,3}\.){3}[0-9]{1,3}”|sort -n|uniq -c|sort -nr|awk ‘{if($1>=8) print $2}’|sed ‘s/^/&sshd:/g;s/$/&:deny/g’

awk ‘BEGIN{info=”how are you hello world.”;split(info,A,” ”);for(i in A){print i,A[i]}}’

#split 分割 按变量 数组名 空格来分割

netstat -an|awk ‘/^tcp/ {print $NF}’|sort -n|uniq -c|sort -nr

netstat -an|awk ‘/^tcp/ {print $4}’|awk -F: ‘{print $2}’|sort -n|grep -v ”^$”|more

netstat -an|awk ‘/^tcp/ {print &4}’|awk -F: ‘{print $2}’|sort -n|grep -v “^$”|uniq|wc -l

netstat -an|awk ‘/tcp/ {print $NF}’|sort|uniq -c

netstat -an|awk ‘/tcp/ {s[$NF]++}END{for(a in s) {print a,s[a]}}’ #s是一个数组

Sort 针对文本文件的内容，以行为单位来排序

  -b    忽略每行前面开始出的空格字符。  
   -c    检查文件是否已经按照顺序排序。  
   -d    排序时，处理英文字母、数字及空格字符外，忽略其他的字符。  
   -f    排序时，将小写字母视为大写字母。  
   -i    排序时，除了040至176之间的ASCII字符外，忽略其他的字符。  
   -m    将几个排序好的文件进行合并。  
   -M    将前面3个字母依照月份的缩写进行排序。  
   -n    依照数值的大小排序。  
   -o<输出文件>    将排序后的结果存入指定的文件。  
   -r    以相反的顺序来排序。  
   -t<分隔字符>    指定排序时所用的栏位分隔字符。  
   +<起始栏位>-<结束栏位>    以指定的栏位来排序，范围由起始栏位到结束栏位的前一栏位

Uniq 输出或者忽略文件中的重复行

-u: 去掉了连续的重复行,保留文件中的唯一

-c：去掉重复，并计算每行出现的次数

-d: 只显示重复的行

AWK练习题

   wang     4  
  cui      3  
  zhao     4  
  liu      3  
  liu      3  
  chang    5  
  li       2

   1 通过第一个域找出字符长度为4的  
   2 当第二列值大于3时，创建空白文件，文件名为当前行第一个域$1 (touch $1)  
   3 将文档中 liu 字符串替换为 hong  
   4 求第二列的和  
   5 求第二列的平均值  
   6 求第二列中的最大值  
   7 将第一列过滤重复后，列出每一项，每一项的出现次数，每一项的大小总和

   1、字符串长度  
    awk 'length($1)=="4"{print $1}'  
   2、执行系统命令  
    awk '{if($2>3){system ("touch "$1)}}'  
   3、gsub(/r/,"s",域) 在指定域(默认$0)中用s替代r  (sed 's///g')  
    awk '{gsub(/liu/,"hong",$1);print $0}' a.txt  
   4、列求和  
    df -h | awk '{a+=$2}END{print a}'  
   5、列求平均值  
    df -h | awk '{a+=$2}END{print a/NR}'  
    df -h | awk '{a+=$2;b++}END{print a,a/b}'   
   6、列求最大值  
    df -h | awk 'BEGIN{a=0}{if($2>a) a=$2 }END{print a}'  
   7、将第一列过滤重复列出每一项，每一项的出现次数，每一项的大小总和  
    awk '{a[$1]++;b[$1]+=$2}END{for(i in a){print i,a[i],b[i]}}'

定义数组

WEB=(www.tb.net www.jf.net www.yk.net)

echo ${WEB[0]}

www.tb.net

echo ${WEB[1]}

www.jf.net

echo ${WEB[2]}

www.yk.net

JFEDU=([0]=A [1]=B [3]=D) 自定义下标与变量值

echo ${JFEDU[@]} 打印所有变量值

A B D

echo ${JFEDU[@]:0:2} 打印下标为0到2的值 默认数组下标从0开始

A B

echo ${JFEDU[@]:1} 打印下标为1的变量值

B

JFEDU[2]=C 向数组中添加带下标的变量值

echo ${JFEDU[@]}

A B C D

unset JFEDU[0]

echo ${JFEDU[@]}

B C D

ctrl+a: 光标跳到行首。  
ctrl+d: 删除光标后一个字符或exit、logout。  
ctrl+e: 光标移到行尾。  
ctrl+f：向后移一个字符。  
ctrl+h: 删除光标前一个字符，同backspace键相同。  
ctrl+k: 剪切光标后至行尾的内容。  
ctrl+l: 清屏，相当于clear。  
Ctrl+p：重复上一次命令。  
ctrl+r: 搜索之前的命令历史。多次ctrl+r 会一直向上搜索。  
ctrl+u: 剪切光标前至行首间的所有内容。  
ctrl+w: 剪切前面的字符至上一个空格处。  
ctrl+t: 交换光标位置前的两个字符。  
ctrl+y: 粘贴或者恢复上次的删除。  
ctrl+z: 把当前进程转到后台运行，使用fg命令恢复。  
Ctrl+x: 跳回之前移动的原位置。  
ctrl+m: 等同于回车键  
ctrl+o: 等同于回车键  
ctrl+s: 暂时冻结当前shell的输入  
ctrl+q: 解冻  
  
补充：  
Esc+B:  移动到当前单词的开头  
Esc+F:  移动到当前单词的结尾

keepalived 高可用构建：

yum 安装 keepalived #需在部署在多台机器上,才可实现高可用

yum install -y keepalived

vim /etc/profile

export PATH=$PATH:/usr/sbin/

source /etc/profile

systemctl start keepalived

pkill keepalived

systemctl start keepalived

vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

wangwei@codemao.cn

}

notification\_email\_from 65636962@qq.com

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

#如果接收到的报文和上一个报文来自同一个服务器，就不检查

# vrrp\_skip\_check\_adv\_addr

#执行VRRP协议规范，不支持节点单播，开启无法PING通VIP

# vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33

virtual\_router\_id 51

priority 100 # 优先级的权限最大

advert\_int 1 # 间隔1秒发送组播包, 检测master 还在不在

# nopreempt # 主机重启之后是否抢占回服务

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.11.110 # 虚拟IP

}

}

ip addr list #查看 VIP

配置 多IP 实现一个网卡多IP,在一台服务器配置多IP或者 配置多网卡，每个网卡一个IP

cd /etc/sysconfig/network-scripts/

cp ifcfg-ens33 ifcfg-ens33:1

vim ifcfg-ens33:1

DEVICE=ens33:1

ONBOOT=yes

IPADDR=192.168.0.110

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

DNS1=202.103.24.68

DNS2=114.114.114.114

ifup ifcfg-ens33:1 //启用网卡

ifdown ifcfg-ens33:1 //关闭网卡

tail -n 30 /var/log/messages // 查看系统日志

advert\_int 5 // 间隔5秒发送组播包 发组播包是为了检测master 还在不在

nopreempt // 不抢占master

# 每隔5秒检测一次nginx

while sleep 5;do sh /data/shell/2019-12-11/auto\_check\_nginx\_vip.sh;done

#keepalived 监控Apache 的脚本

cat /data/sh/check\_httpd.sh

#!/bin/bash

NUM=$(ps -ef|grep httpd|grep -vcE “grep|check”)

if [ $NUM -eq 0 ];then

pkill keepalived

fi

yum install -y tcpdump

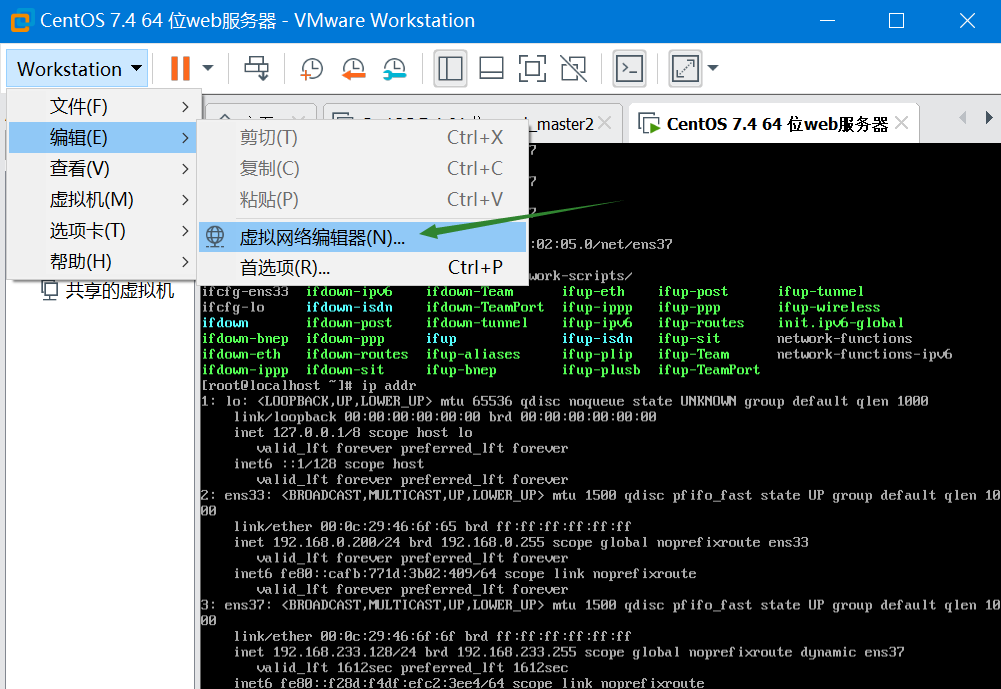
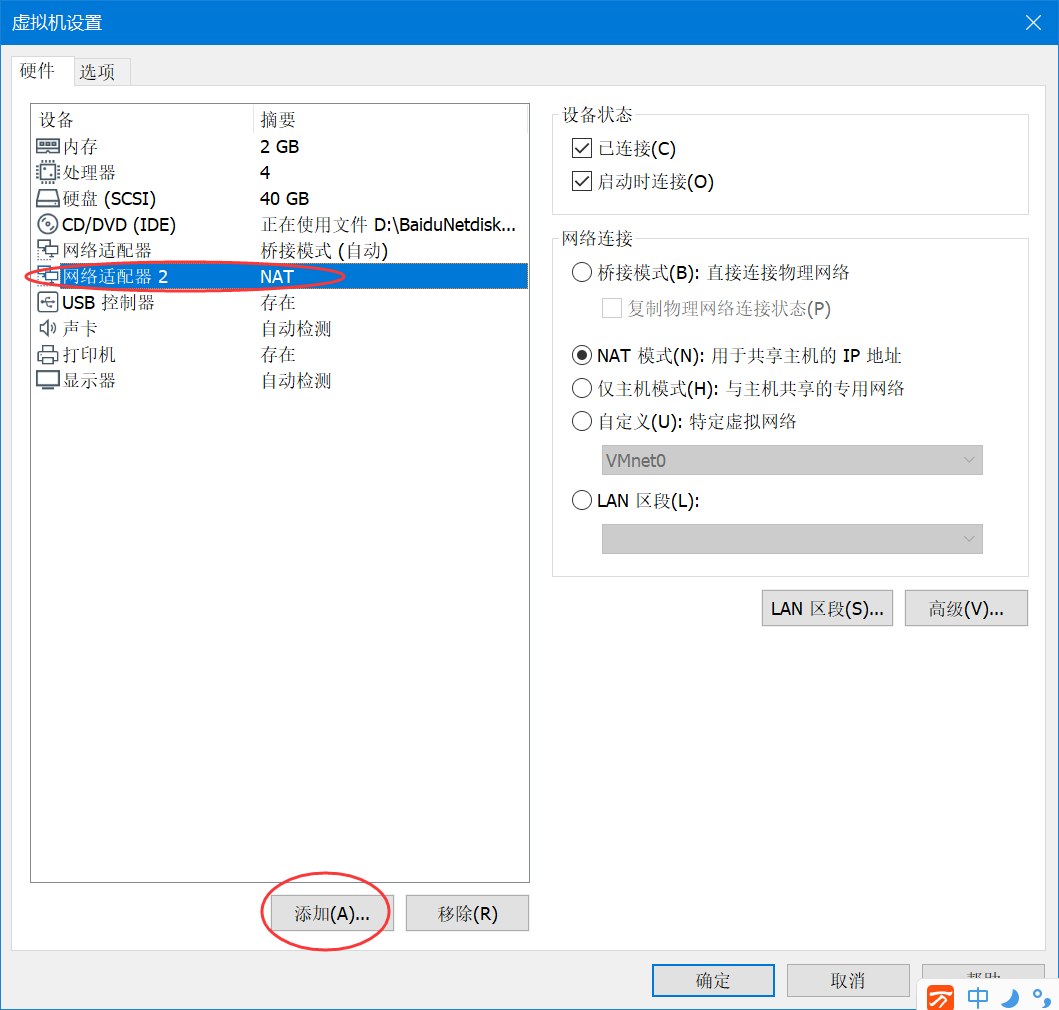
tcpdump -i ens33 -nn host (虚拟主机IP) // 抓包测试

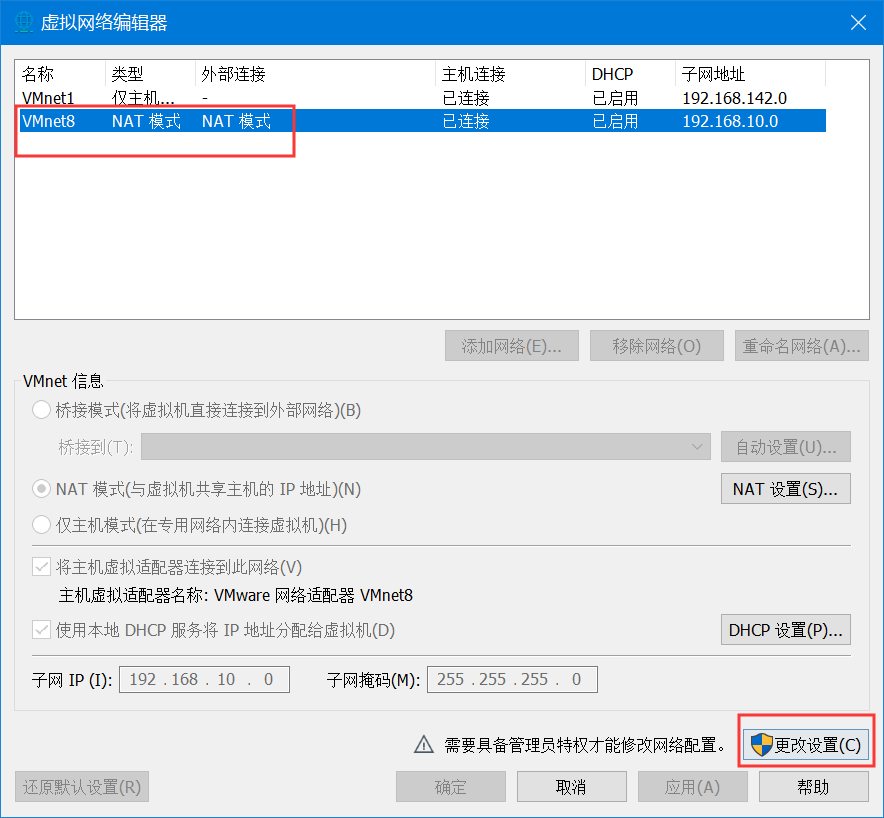
tcpdump -i ens33 -nn port 80 and host 10.10.10.188 // 抓端口

tcpdump -i lo:0 -nn port 80 //抓本地网卡

ssh -l root 192.168.11.162

**LVS NAT模式构建需要配置两块网卡，一块内网，一块外网**

在VMware的LVS服务器中添加一块NAT模式的网卡



cd /etc/sysconfig/network-scripts/

cp ifcfg-ens33 ifcfg-ens37

vim ifcfg-ens37

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static

DEFROUTE=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_FAILURE\_FATAL=no

IPV6\_ADDR\_GEN\_MODE=stable-privacy

NAME=ens37

#UUID=0c9eaa89-b14f-4bb2-9b81-24bfd5dfdca8

DEVICE=ens37

ONBOOT=yes

IPADDR=192.168.10.128

NETMASK=255.255.255.0

GATEWAY=192.168.10.2

DNS1=202.103.24.68

DNS2=114.114.114.114

cp ifcfg-ens37 ifcfg-ens37:1

vim ifcfg-ens37:1

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static

DEFROUTE=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_FAILURE\_FATAL=no

IPV6\_ADDR\_GEN\_MODE=stable-privacy

NAME=ens37:1

#UUID=0c9eaa89-b14f-4bb2-9b81-24bfd5dfdca8

DEVICE=ens37:1

ONBOOT=yes

IPADDR=192.168.10.188

NETMASK=255.255.255.0

GATEWAY=192.168.10.2

DNS1=202.103.24.68

DNS2=114.114.114.114

systemctl restart network

ifconfig

modprobe ip\_vs // 加载 ip\_vs 内核模块

lsmod |grep ip\_vs // 查看模块是否加载成功

yum install -y ipvsadm

rpm -qa|grep ipvsadm

ipvsadm -v // 检测安装版本

在LVS服务器操作：

ipvsadm -A -t 192.168.10.128:80 -s rr //-A 添加虚拟服务器 -t TCP -s 调度算法 rr 轮询

ipvsadm -Ln // 查询是否添加成功

ipvsadm -a -t 192.168.10.128:80 -r 192.168.0.161 -m -w 100 //NAT模式 -m DR 模式 -g -a 在虚拟服务器中增加后端均衡服务器 -r 指定后端真实server

ipvsadm -a -t 192.168.10.128:80 -r 192.168.0.162 -m -w 100

ipvsadm -a -t 192.168.10.128:80 -r 192.168.0.163 -m -w 100

ipvsadm -L -n // 查看请求到哪台主机上了

echo 1 > /proc/sys/net/ipv4/ip\_forward //开启路由转发 0 是关闭 1 是开启

cat /proc/sys/net/ipv4/ip\_forward

分别在所有后端主机操作：

后端主机需要修改IP，后端主机的网关要修改成LVS主机的ens33的真实IP

sed -i ‘s/0.1$/0.200/g’ /etc/sysconfig/network-scripts/ifcfg-ens33

route -n // 查看路由

arp -a // 查看MAC地址

vim /etc/resolv.conf // 修改本机DNS

ipvsadm -L -n

ipvsadm -D -t 192.168.10.128:80 //删除VIP和端口

ipvsadm -d -t 192.168.10.128:80 -r 192.168.0.161 //删除真实网卡IP

ipvsadm -d -t 192.168.10.128:80 -r 192.168.0.162

ipvsadm-save -n >lvs.txt //避免重新丢失,导出配置到文档

ipvsadm-restore -n <lvs.txt //导入配置文档

ipvsadm -L -n

ipvsadm --help

**LVS DR模式构建需要只需要一块网卡**

lvs 服务器上操作：

ipvsadm -A -t 192.168.0.111:80 -s rr

ipvsadm -a -t 192.168.0.111:80 -r 192.168.0.161 -g -w 100

ipvsadm -a -t 192.168.0.111:80 -r 192.168.0.162 -g -w 100

ipvsadm -Ln

cd /etc/sysconfig/network-scripts/

cp ifcfg-ens33 ifcfg-ens33:1

vim ifcfg-ens33:1

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static

DEFROUTE=yes

NAME=ens33:1

DEVICE=ens33:1

ONBOOT=yes

IPADDR=192.168.0.111

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

DNS1=202.103.24.68

systemctl restart network

ip addr list

在每台后端服务器执行Realserver.sh这个脚本：

#!/bin/bash

#LVS Client Server

VIP=192.168.0.111 #和虚拟机在同一网段

case $1 in start) ifconfig lo:0 $VIP netmask 255.255.255.255 broadcast $VIP

/sbin/route add -host $VIP dev lo:0

echo "1" >/proc/sys/net/ipv4/conf/lo/arp\_ignore

echo "2" >/proc/sys/net/ipv4/conf/lo/arp\_announce

echo "1" >/proc/sys/net/ipv4/conf/all/arp\_ignore

echo "2" >/proc/sys/net/ipv4/conf/all/arp\_announce

sysctl -p >/dev/null 2>&1

echo "RealServer Start OK"

exit 0

;; stop)

ifconfig lo:0 down

route del $VIP >/dev/null 2>&1

echo "0" >/proc/sys/net/ipv4/conf/lo/arp\_ignore

echo "0" >/proc/sys/net/ipv4/conf/lo/arp\_announce

echo "0" >/proc/sys/net/ipv4/conf/all/arp\_ignore

echo "0" >/proc/sys/net/ipv4/conf/all/arp\_announce

echo "RealServer Stoped OK"

exit 1

;;

\*)

echo "Usage: $0 {start|stop}"

;;

esac

ipvsadm -Ln // 查看请求到哪台主机上了

windows 🡪 运行🡪 cmd 🡪 arp -a // 查看mac 地址

arp -d //清空缓存

**LVS + keepalived 构建:**

LVS 服务器操作：

yum install –y ipvsadm

yum install –y keepalived

vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

wangwei\_201011@163.com

}

notification\_email\_from wangwei\_201011@163.com

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

# vrrp\_skip\_check\_adv\_addr

# vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33

virtual\_router\_id 51

priority 100

advert\_int 1

nopreempt

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.0.110

}

}

virtual\_server 192.168.0.110 80 {

delay\_loop 6

lb\_algo rr

lb\_kind DR

persistence\_timeout 60

protocol TCP

real\_server 192.168.0.161 80 {

weight 100

TCP\_CHECK {

connect\_timeout 10

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

real\_server 192.168.0.162 80 {

weight 100

TCP\_CHECK {

connect\_timeout 10

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

}

#161和162 都需要安装keepalived, 并备份原keepalived.conf

scp -r keepalived.conf root@192.168.0.161:/etc/keepalived/

scp –r keepalived.conf root@192.168.0.162:/etc/keepalived/

systemctl restart keepalived

ipvsadm –Ln

systemctl restart keepalived

ipvsadm –Ln

#看到所有物理主机添加好了即可

ip addr list

#查看VIP

在161 和 162 执行

systemctl restart keepalived

sh Realserver.sh start

后端两台机器的keepalived优先级要设置不一样

单主的模式 state 设置为 BACKUP

如果是双主模式 VI\_1 和 VI\_2 的virtual\_router\_id 必须不一样

#digest校验码的获取方法:

genhash -s 192.168.0.161 -p 80 -u http://192.168.0.161/index.html

real\_server 10.10.10.142 80 {  
        weight 100  
        HTTP\_GET {  
          url {  
          path /index.html  
          digest a76b2b824459a563428efee4e4e10dfa  
          }  
          connect\_timeout 3  
          nb\_get\_retry 3  
          delay\_before\_retry 3  
        }  
    }

ansible -k -i /etc/ansible/hosts 192.168.0.161 –m command –a “df -h”

#-k 要求输入登录密码 -i 指定操作对象 -m 指定所使用的命令模块名称 –a 指定模块参数

免密钥登录操作:

ssh-keygen

ll /root/.ssh/id\_rsa\*

ssh-copy-id -i /root/.ssh/id\_rsa.pub 192.168.0.161

输入对方主机的登录密码

ssh-copy-id -i /root/.ssh/id\_rsa.pub 192.168.0.162

输入对方主机的登录密码

或者 关掉主机公钥检测

vim /etc/ansible/ansible.cfg

host\_key\_checking = False

pipelining = True

免密钥登录以后恢复使用密钥登录

vim /root/.ssh/authorized\_keys

清空文件中的内容

/usr/sbin/sshd -D

如果提示could not load host key ,就输入以下命令

ssh-keygen -t rsa -f /etc/ssh/ssh\_host\_rsa\_key

ssh-keygen -t ecdsa -f /etc/ssh/ssh\_host\_ecdsa\_key

ssh-keygen -t ed25519 -f /etc/ssh/ssh\_host\_ed25519\_key

-t type加密的类型 -f file文件的位置和名字

time ansible-playbook add\_user.yaml # 可以查看脚本执行完成所需的时间

#并发执行

ansible 192.168.0.161,192.168.0.162 -m command -a “df -h”

ansible all -m command –a “df -h” # all 表示所有hosts中配置的主机

vim /etc/ansible/hosts

[web]

192.168.0.161

[db]

192.168.0.162

#分组管理IP

ansible db -m command -a “df -h” #db 表示IP组名

ansible web -m command -a “df -h” # web 表示组名

#串行执行

for ip in `seq 140 143`;do echo -e "\033[32m10.10.10.$ip | CHANGED | rc=0 >>\033[0m";ssh -l root 10.10.10.$ip "df -h" ;done

ansible db -m shell -a “yum install -y screen” #使用shell命令模块 安装工具

ansible db -m shell -a “sed -i ‘s/www.jfedu.net/www.jd.com/g’ /tmp/jfedu.txt”

ansible db -m shell -a “cat /tmp/jfedu.txt”

ansible all -m copy -a "src=/root/zabbix-3.2.6.tar.gz dest=/tmp/ mode=644 owner=root"

#src源路径 dest目标路径mode权限owner所属用户

ansible all -m shell -a "ls /tmp/" # all表示/etc/ansible/hosts中所有主机

ansible all -m shell -a "echo '0 0 \* \* \* /usr/sbin/ntpdate pool.ntp.org >>/dev/null' >>/var/spool/cron/root;crontab -l"

ansible  all  -m  cron  -a  "minute=0  hour=0  day=\*  month=\*  weekday=\* name=’Ntpdate server for sync time’  job=’/usr/sbin/ntpdate pool.ntp.org >/dev/null’"

ansible all -m shell -a "sed -i '/ntpdate/d' /var/spool/cron/root"

ansible all -m shell -a "echo -e '#Jfedu: Ntpdate server for sync time\n0 0 \* \* \* /usr/sbin/ntpdate pool.ntp.org >>/dev/null' >>/var/spool/cron/root"

ansible all -m user -a “name=jfedu666,jfedu777 state=absent force=yes”

ansible all -m shell -a “userdel -r jfedu666;userdel -r jfedu777”

ansible all -m service -a “name=nginx state=stopped”

ansible all -m shell -a “systemctl stop nginx”

ansible -k -i /etc/ansible/hosts 10.10.10.188 -m command -a “df -h”

ansible all -m synchronize -a “src=/var/www/html dest=/tmp/”

ansible all -m copy -a “src=auto\_install\_nginx.sh dest=/tmp/ mode=755 owner=root” -f1 //从第一台主机开始安装

ansible all -m copy -a “src=auto\_install\_nginx.sh dest=/tmp/ mode=755 owner=root” -f 100

//100台主机随机安装

ansible all -m shell -a “rpm -qa|grep nginx”

**#编写playbook剧本**

vim nginx\_install.yaml

- hosts: all

remote\_user: root

gather\_facts: yes

task:

- name: Install nginx lib pcre-devel and zlib

yum: name=gcc,pcre-devel,pcre,zlib-devel state=installed

- name: copy nginx-1.16.1.tar.gz to client /tmp/

copy: src=/root/nginx-1.16.1.tar.gz dest=/tmp/

- name: nginx web server install process

shell: cd /tmp;tar -xzf nginx-1.16.1.tar.gz;cd nginx-1.16.1;./configure --prefix=/usr/local/nginx;make;make install

- name: start nginx web service

shell: pkill nginx;sleep 3;/usr/local/nginx/sbin/nginx

ansible-playbook nginx\_install.yaml #执行yaml剧本

#判断nginx安装目录存不存在，存在就不创建，不存在就创建

- hosts: all

remote\_user: root

tasks:

- name: nginx sever install

file: path=/usr/local/nginx/ state=directory #如果目标目录不存在就创建

notify: #通知hanlders上面的命令执行成功后就执行任务

- nginx install

- nginx start

handlers: # 这是触发器

- name: nginx install

shell: cd /tmp;tar -xzf nginx-1.16.1.tar.gz;cd nginx-1.16.1;./configure --prefix=/usr/local/nginx;make;make install

- name: nginx start

shell: /usr/local/nginx/sbin/nginx

ansible all -m file -a "path=/usr/local/nginx/ state=directory"

#**shell脚本 和ansible-playbook整合**

- hosts: all

remote\_user: root

task:

- name: nginx server install

copy: src=/root/auto\_install\_nginx.sh dest=/tmp/ #本机文件和对端文件的md5sum一样的话就不会触发handlers执行

- name: exec auto\_install\_nginx.sh

shell: cd /tmp/;/bin/sh auto\_install\_nginx.sh

notify:

- nginx start

handlers:

- name: nginx start

shell: /usr/local/nginx/sbin/nginx

vim auto\_install\_nginx.sh

#!/bin/bash

if [ ! -d /usr/local/nginx ];then

yum install -y gcc pcre pcre-devel zlib-devel

mkdir -p /usr/local/nginx

cd /tmp/

tar -xzf nginx-1.16.1.tar.gz

cd nginx-1.16.1

./configure --prefix=/usr/local/nginx --user=www --group=www --with-http\_stub\_status\_module --with-http\_ssl\_module

make

make install

if [ $? -eq 0 ];then

/usr/local/nginx/sbin/nginx

fi

fi

#**变量批量创建用户**

- hosts: all

remote\_user: root

task:

- name: linux add user

user: name={{ item }} state=present # item为变量

with\_items: #items是ansible内置参数,不能修改

- jfedu1

- jfedu2

- jfedu3

- jfedu4

- jfedu5

ansible all -m shell -a "tail -n 5 /etc/passwd"

#**变量批量删除用户**

- hosts: all

remote\_user: root

task:

- name: linux delete user

user: name={{ item }} state=absent # item为变量

with\_items: #items是ansible内置参数,不能修改

- jfedu1

- jfedu2

- jfedu3

- jfedu4

- jfedu5

ansible all -m shell -a "tail -n 5 /etc/passwd"

#**shell批量创建用户**

- hosts: all

remote\_user: root

task:

- name: Linux add user list

shell: for user in `seq 1 100`;do useradd jfedu$user ;done

ansible all -m shell -a "tail -n 10 /etc/passwd"

vim /etc/ansible/hosts

[web]

192.168.0.161 nginx\_port=80 ansible\_user=root ansible\_password=123456

192.168.0.162 nginx\_port=81 ansible\_user=root ansible\_password=123456

[db]

192.168.0.163 nginx\_port=82

192.168.0.164 nginx\_port=83

#设置了ansible\_user ansible\_password就不用做ssh-keygen免密钥了

#拷贝nginx.conf并重命名, 修改listen 80为listen {{httpd\_port}}

cp nginx.conf nginx.conf.j2

listen {{httpd\_port}};

#创建nginx\_intall.yaml

- hosts: all

remote\_user: root

tasks:

- name: Nginx server Install

file: path=/usr/local/nginx/ state=directory

notify:

- nginx install

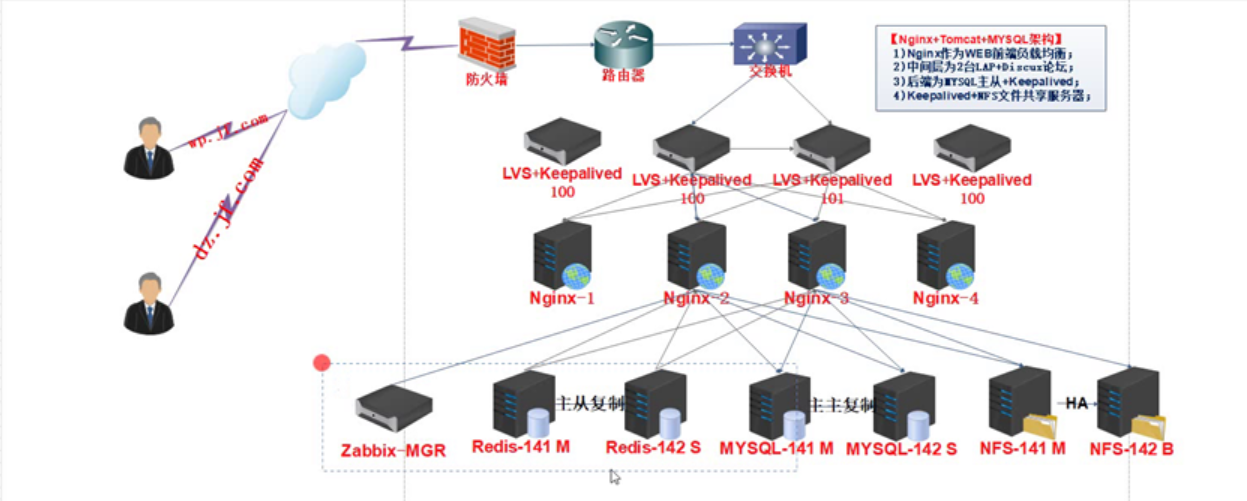
- nginx config

handlers:

- name: nginx install

shell: cd /tmp;rm -rf nginx-1.12.0.tar.gz;wget http://nginx.org/download/nginx-1.12.0.tar.gz;tar xzf nginx-1.12.0.tar.gz;cd nginx-1.12.0;./configure --prefix=/usr/local/nginx;make;make install

- name: nginx config

template: src=/data/sh/nginx.conf.j2 dest=/usr/local/nginx/conf/nginx.conf 

nc -w 10 10.0.0.188 80 && echo ok||echo no //-w 超时时间 10秒内连接 IP的80端口 成功就 显示 ok 否则 显示 no

echo -e “\n”|telnet www.baidu.com 80|grep Conneted|wc -l

nmap www.baidu.com -p 80

ifdown ens33:1 //关闭子接口

ifup ens33:1

yum install -y yum-utils

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

yum install lxc cgroup\* device-mapp\* -y // 安装docker依赖包

yum makecache

yum install -y docker-ce

yum list docker

systemctl start docker //启动docker服务端

systemctl enable docker

systemctl stop docker

docker info // 查看系统信息

docker --version //查看docker版本号

docker version

docker stats CONTAINER ID --no-stream # 查看容器状态

docker stats CONTAINER ID --no-stream|awk '{print $2}'

docker stats CONTAINER ID --no-stream|awk 'NR>1 {print "CPU Usage:"$2}'

docker exec CONTAINER ID df -h # 查看容器硬盘空间

docker search nginx1.16.1 // 在容器中的查找镜像

docker pull docker.io/nginx // 下载nginx镜像

docker images // 查看镜像列表判断镜像是否下载成功

ll /var/lib/docker/images // 从目录查看镜像文件

docker run -itd -p 80:80 docker.io/nginx // 基于nginx镜像启动nginx应用容器

// run创建一台新容器(虚拟机) -i interactive交互模式 -t tty打开终端 -d daemon后台启动 -p开启DNAT端口映射, 将宿主机的80端口(第一个)映射至容器的80端口

docker ps // 查看已启动的容器

docker inspect CONTAINER ID // 查看容器所有信息

docker inspect CONTAINER ID | grep -i ipaddr // 查看IP

docker inspect CONTAINER ID | grep -i ipaddr | tail -1 | awk -F\" '{print $4}' //只显示IP -F分割

docker inspect CONTAINER ID | grep -i ipaddr | tail -1 | grep -oE "([0-9]{1,3}\.){3}[0-9]{1,3}"

docker exec -it CONTAINER ID /bin/bash // 进入容器 -i交互模式 -t打开终端

find / -name nginx.conf

cat /etc/nginx/nginx.conf

cat /etc/nginx/conf.d/default.conf

cd /usr/share/nginx/html/

echo www.baidu.com >index.html

docker cp index.html CONTAINER ID:/usr/share/nginx/html/ // 把宿主机的文件拷贝到容器的目录里面

docker exec CONTAINER ID nginx -s reload //重启nginx容器中web服务

docker history docker.io/nginx (镜像名) // 查看镜像制作过程中的内容

docker exec CONTAINER ID nginx -V // 查看容器中的nginx版本信息

docker run -itd 镜像名 /bin/bash //创建并进入容器

exit //退出容器

docker exec -it container ID /bin/bash // 进入容器

docker attach container ID // 进入容器（不推荐使用）

docker stop container ID // 停止容器

docker start container ID // 开启容器

docker restart container ID // 重启容器

docker images // 显示docker 内的镜像

docker rmi 镜像名 // 删除docker 内的镜像

docker run -itd 镜像名 /bin/bash //后台生成 并运行 容器 并登录 容器

docker ps // 查看正在运行的容器

docker ps -a // 查看 当前运行的 与 已经关闭的 容器

docker rm container ID //删除容器

docker rename oldname newname // 修改容器名称

docker commit CONTAINER ID name:tag // 把容器制作成镜像并命名

ps aux // 显示当前终端下所有程序

vim /etc/docker/daemon.json #配置docker下载镜像源，只需要写一条即可

{ "registry-mirrors":["http://hub-mirror.c.163.com"] } #163的源

{ "registry-mirrors":["https://registry.docker-cn.com"]" } #docker官方源

docker search centos7

docker images

docker images|grep -i centos

tar -tf centos7.tar # 查看镜像内容

cat centos7.tar|docker import - centos7 # 把上传的镜像导入到docker中

docker images # 查看镜像是否导入成功

echo 1 > /proc/sys/net/ipv4/ip\_forward #开启路由转发功能，把Linux服务器当成路由器，开启转发功能，这样才能将请求转发到其他的机器 如果不使用-p映射端口就无需开启

docker run -itd --privileged --name=centos7-v1 -p 6022:22 centos7

# 基于centOS镜像启动一个centOS容器 并给予root用户超级特权 并给容器命名 并把宿主机的6022端口映射到容器的22端口

docker run -itd --privileged --net=host --name=centos7-v1 -p 6022:22 centos7

**--net选项指定容器的网络模式，Docker有以下4种网络模式**：

**· host模式，使用--net=host指定**  
**· container模式，使用--net=container:NAME\_or\_ID指定**  
**· none模式，使用--net=none指定**  
**· bridge模式，使用--net=bridge指定，默认设置**  
**· 还有一种：用户自定义模式**

docker inspect CONTAINER ID | grep -i ipaddr # 查看容器的IP

vim /etc/ssh/sshd\_config

PermiRootLogin yes

PasswordAuthentication yes

docker ps # 查看已启动的容器

docker exec -it CONTAINER ID /bin/bash # 进入容器

yum install -y net-tools vim

passwd # 修改容器密码

exit

docker stop Container ID # 停止运行容器

docker start Container ID # 启动容器

/usr/sbin/sshd -D # 启动ssh服务 前台持续运行

ssh-copy-id -i /roo/.ssh/id\_rsa.pub 172.17.0.2 # 拷贝公钥到容器中以便免密登录

ssh -l root CONTAINER IP # 从宿主机远程登录到这个容器中

docker exec -it CONTAINER ID /bin/bash # 进入容器

hostname docker-v1 # 修改主机名

创建Dockerfile来启动容器

vim Dockerfile

# 设置基本的镜像，后续命令都以这个镜像为基础

FROM centos7

# 作者信息

MAINTAINER wangwei

RUN echo 123456|passwd --stdin root

RUN yum install net-tools -y

RUN yum install wget tar make gcc -y

RUN sed -i '/PermitRootLogin/s/no/yes/g' /etc/ssh/sshd\_config

RUN sed -i '/PasswordAuthntication/s/#//g' /etc/ssh/sshd\_config

CMD /usr/sbin/sshd -D;sleep 9999d

#CMD 是容器运行以后要执行的命令

docker build -t centos7-v1 . # 在Dockerfile所在目录执行这条命令 -t表示指定镜像的名称 centos7-v1是镜像名 build会创建一个新的镜像,用docker images查看

Dokerfile # 基于镜像 安装软件,创建或修改文件,启动服务, 最后生成一个新的镜像

docker images

ls /var/lib/docker/image/

docker build -t centos7-v1 - < Dockerfile # - 表示占位符 制作镜像文件, 如果不在dockfile的同级目录 就使用这条命令执行

docker history IMAGE ID # 查看镜像相关信息

docker run -itd --privileged -p 80:80 -p 6022:22 centos-v6 #宿主机映射80、6022到容器

docker exec -it CONTAINER ID /bin/bash

netstat -ntl

yum install -y openssh-server openssh-devel openssh

docker commit CONTAINER ID # 把容器生成镜像文件

docker images # 查看生成的镜像

docker tag IMAGE ID centos-ssh:v1 # 通过镜像ID给镜像重命名 :v1定义TAG

docker tag centos-v1 centos:v1 # 通过镜像名 给镜像重命名+tag

docker run -itd centos:v1 # 启动同名不同tag的容器

docker ps -aq # 查看容器ID

docker ps -aq|xargs docker rm -f # 删除所有容器

docker ps -aq|xargs docker stop # 停止所有容器

docker rm -f Container ID # 强制删除指定容器

docker rename原名称 新名称 # 容器重命名

docker ps

docker ps -aq|xargs docker start # 启动所有容器

docker rmi $(docker images -q)

docker rmi `docker images -q` # 删除所有镜像

docker rmi -f IMAGE ID # 强制删除指定镜像

docker run -itd -p 6023:22 --privileged centos-ssh:v1 # 加上tag来启动镜像

# 一个镜像 可以 生成 多个容器

docker save IMAGE ID >centos-ssh.tar #导出镜像

docker load -i centos-ssh.tar #导入镜像

docker tag IMAGEID centos-ssh:v1 #通过镜像ID给镜像重命名

tar -tf centos-ssh.tar #查看镜像内容

docker export CONTAINER ID >centos-ssh.tar #把容器导出成镜像,会丢失内存中正在运行的数据

cat centos-ssh.tar | docker import - centos-ssh # 把镜像导入到Docker Images

docker run -itd docker.io/nginx sleep 9999d # 后台启动 持续运行9999天后退出

# sleep 9999d 相当于CMD命令 也叫入口命令 保持容器一直运行, CMD命令执行完成了,容器就退出了 run创建容器

docker run -itd docker.io/jdeathe/centos-ssh sleep 9999d #持续运行容器9999天后退出

docker run -itd docker.io/jdeathe/centos-ssh /bin/bash # 保持容器永久不退出

docker run -itd docker.io/nginx /usr/sbin/nginx -g daemon off # 表示前台启动

docker run -itd --privileged -p 6022:22 -v /data/:/usr/share/nginx/html/ docker.io/nginx

# 把宿主机的/data/ 目录映射到容器中, 宿主机同一目录可以映射到多个容器中 -v volume 指定数据挂载点, 多个容器 可以 映射宿主机的同一目录

如果要取消映射关系, 就得把该容器删除

docker inspect CONTAINER ID|grep -C 10 -i "binds" # 查看容器的映射端口及目录

docker history IMAGE ID # 查看镜像制作过程及CMD命令

docker history docker.io/nginx # 用镜像名查看也可以

docker cp sshd\_config CONTAINER ID:/etc/ssh/ # 把宿主机的文件拷贝到容器中

docker restart CONTAINER ID

docker exec -it CONTAINER ID /bin/bash # 进入容器

/usr/sbin/sshd -D

ls /etc/ssh

ssh\_host\_ecdsa\_key ssh\_host\_ed25519\_key ssh\_host\_rsa\_key #必须要有这三个文件

chmod 600 ssh\_host\_ecdsa\_key ssh\_host ed25519\_key ssh\_host\_rsa\_key # 给予文件-rw权限

docker search busybox

docker pull docker.io/busybox

docker run -itd --privileged --name=busybox IMAGE ID /bin/bash

docker exec -it CONTAINER ID /bin/sh

按键盘TAB键 连续按两次 即可出现所有命令

mkdir lib lib64

把宿主机的这两个文件夹的内容 拷贝到busybox中

在busybox中安装一些相关工具

创建一个Dockerfile

vim Dockerfile

# 设置基本的镜像，后续命令都以这个镜像为基础

FROM docker.io/busybox

# 作者信息

MAINTAINER wangwei

RUN echo 123456|passwd --stdin root;mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.bak

COPY CentOS-Base.repo /etc/yum.repos.d/

VOLUME /var/data/2019/

WORKDIR /root/

EXPOSE 22

CMD /usr/sbin/init;/usr/sbin/sshd;/usr/local/nginx/sbin/nginx;/bin/bash

# 分号；表示前面的命令无论执行成功与否都会执行后面的命令

# 与号&& 表示 前面的命令如果执行不成功就不会执行后面的命令

docker build -t Centos7-v2 .

# 在Dockerfile所在目录执行这条命令 -t表示指定镜像的名称 centos7-v2是镜像名 build会创建一个新的镜像,用docker images查看

创建本地私有docker仓库

docker search registry

docker pull docker.io/registry

mkdir -p /data/registry/ #创建宿主机仓库目录

docker run -itd --privileged --name=registry-v1 -p 5000:5000 -v /data/registry:/var/lib/registry docker.io/registry #创建私有仓库容器

docker tag docker.io/jdeathe/centos7-ssh:v1 192.168.0.200:5000/docker.io/centos7-ssh:v1 # 把原镜像名改成 宿主机IP:PORT/镜像名 的格式

docker rmi -f docker.io/jdeathe/centos7-ssh:v1

修改docker配置文件来解决无法上传的故障

vim /etc/systconfig/docker

OPTIONS='…… --insecure-registry 192.168.0.200:5000' 在引号内容的最后添加 允许宿主机IP连接

systemctl restart docker

docker start docker.io/registry

docker push 192.168.0.200:5000/centos7-ssh:v1 # 把镜像上传到本地仓库

docker images

docker ps

docker exec -it docker.io/registry /bin/sh # 进入容器查看镜像内容

ls -l /var/lib/registry

exit

du -sh /data/registry # 进入宿主机查看镜像仓库下 目录的大小

打开虚拟机192.168.0.161，从192.168.0.200的docker仓库下载镜像

curl -XGET http://192.168.0.200:5000/v2/\_catalog # 检测是否存在镜像文件

curl -XGET http://192.168.0.200:5000/v2/centos7-ssh/tags/list

修改客户端的docker配置文件

vim /etc/sysconfig/docker

OPTIONS='…… --insecure-registry 192.168.0.200:5000' 在引号内容的最后添加 允许宿主机IP连接

ADD\_REGISTRY='--add-registry 192.168.0.200:5000' # 添加到OPTIONS下面

systemctl restart docker

docker pull 192.168.0.200:5000/centos7-ssh:v1

docker tag 192.168.0.200:5000/centos7-ssh:v1 centos7-ssh:v1

docker images

docker rmi -f 192.168.0.200:5000/centos7-ssh

docker run-itd --privileged --name=centos7-ssh -p 80:80 -p 6022:22 centos7-ssh:v1 #创建容器

docker ps

rm -rf /var/lib/docker/\* # 删除docker目录才能看到容器的实际大小

vim /etc/sysconfig/docker-storage # 更改容器大小

DOCKER\_STORAGE\_OPTIONS="--storage-driver overlay2 --storage-opt dm.basesize=40G"

systemctl restart docker

docker exec CONTAINER ID df -h # 查看容器的大小

yum install lsof -y # 查看当前系统打开的所有文件的工具

lsof | grep nginx | more # 根据端口或进程查看打开的文件

rpm -qf /usr/sbin/nginx # 如果能看到源码包就证明是二进制安装的

# 创建网桥，让docker和宿主机在同一个局域网内，无需再做映射

yum install -y bridge-utils

brctl show # 查看docker的桥接网卡信息

vim /etc/sysconfig/network-scripts/ifcfg-ens33 # 修改物理网卡的配置

TYPE="Ethernet"

PROXY\_METHOD="none"

BROWSER\_ONLY="no"

BOOTPROTO="static"

DEFROUTE="yes"

NAME="ens33"

UUID="b729aa5b-67a4-4fcf-8790-5104028d3d18"

DEVICE="ens33"

ONBOOT="yes"

BRIDGE="br0"

vim /etc/sysconfig/network-scripts/ifcfg-br0 # 创建虚拟网卡

TYPE="Bridge"

PROXY\_METHOD="none"

BROWSER\_ONLY="no"

BOOTPROTO="static"

DEFROUTE="yes"

IPV4\_FAILURE\_FATAL="no"

IPV6INIT="yes"

IPV6\_AUTOCONF="yes"

IPV6\_DEFROUTE="yes"

IPV6\_FAILURE\_FATAL="no"

IPV6\_ADDR\_GEN\_MODE="stable-privacy"

NAME="br0"

DEVICE="br0"

ONBOOT="yes"

IPADDR=192.168.0.200

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

DNS1=202.103.24.68

DNS2=114.114.114.114

systemctl restart network

# 重启后br0就和ens33绑定了

# 启动 容器时 使用这个br0的网桥 有两种方法：

1. 修改配置docker的配置文件

vim /etc/sysconfig/docker-network

DOCKER\_NETWORK\_OPTIONS="-b=br0" # 默认使用的是docker0

systemctl restart network

docker images

docker run -itd --privileged --name=centos7-ssh --net=none docker.io/centos7-ssh:v1

docker ps

docker network ls # 查看当前docker网络类型

docker network create --subnet=172.18.0.0/16 mynetwork # 创建自定义网络

# 安装pipework

yum install -y git

git clone https://github.com/jpetazzo/pipework

cp ~/pipework/pipework /usr/local/bin/

# 启动容器并设置网络

pipework br0 CONTAINER ID 192.168.0.201/24@192.168.0.1 # 给容器分配IP

docker exec CONTAINER ID ifconfig # 查看容器的IP

docker exec CONTAINER ID route -n # 查看容器的GATEWAY

docker exec -it CONTAINER ID /bin/bash # 进入容器

ping www.baidu.com

# 创建10台docker容器

for i in $(seq 80 89);do docker run -itd --privileged --name=jfedu\_0\_$i --net=none centos7-ssh:v1; done

# 给刚才创建的10台容器分配IP地址

for i in $(seq 80 89);do pipework br0 jfedu\_0\_$i 192.168.0.$i/24@192.168.0.1 ;done

docker ps -aq #查看容器ID

docker ps -a | awk 'NR>1 {print $1,$NF}'

docker exec CONTAINER ID ifconfig|grep "netmask"|grep -v 127.0.0.1|awk '{print $2}'

docker exec CONTAINER ID ifconfig|grep -oE "([0-9]{1,3}\.){3}[0-9]{1,3}"|grep -vE "^127|255" #去掉127和255开头的地址

docker exec CONTAINER ID ifconfig|awk 'NR==2'

docker exec CONTAINER ID ifconfig|awk 'NR==2 {print $2}' # 实现和grep相同的结果

#把容器IP信息导出成excel表格

docker ps -a|awk 'NR>1 {print $1,$NF}'|sed 's/ /,/g' >docker.csv

sz docker.csv



# 也可以把这个数据放到数据库中

# docker部署的jenkins发布网站代码, 不同容器之间通讯, 需要做免密钥

docker run -itd --privileged --name=centos7-ssh -p 80:80 docker.io/centos7-ssh:v1

docker run -itd --privileged -p 8080:8080 docker.io/jenkins

docker exec -it docker.io/jenkins /bin/bash

cd /var/jenkins\_home

ssh-keygen

ls -l

ip addr

exit

docker exec -it docker.io/centos7-ssh /bin/bash

passwd

123456

exit

ssh -l root 172.17.0.3 # 宿主机远程登录测试是否连接

docker exec -it docker.io/jenkins /bin/bash

cd /var/jenkins\_home

ssh-copy-id -i .shh/id\_rsa.pub root@172.17.0.3

ssh -l root 172.168.0.3 # 免密登录对方主机

rsync -aP --delete /tmp/www/ /usr/share/nginx/html/ # 同步与目标目录差异的文件

#jenkins源码管理 配置git地址, jenkins就可以自动到git获取代码

# 安装jenkins工具

yum install -y https://mirrors.cloud.tencent.com/jenkins/redhat/jenkins-2.207-1.1.noarch.rpm

ln -s /usr/local/jdk1.8.0\_131/bin/java /usr/bin/java

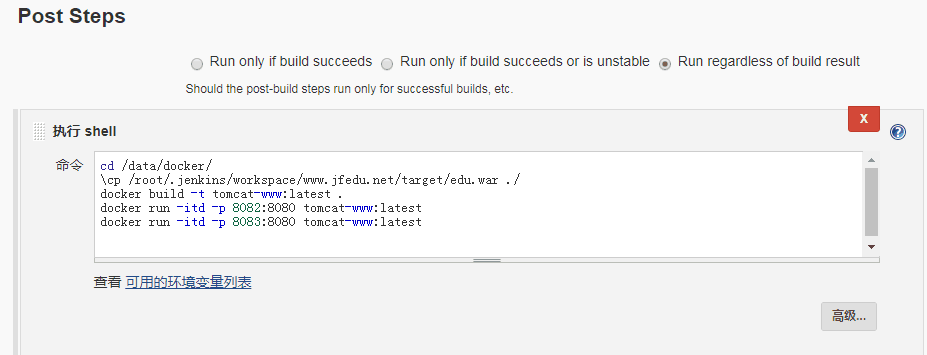
systemctl start Jenkins

#修改jenkins默认的8080端口

vim /etc/sysconfig/Jenkins

JENKINS\_PORT="8089"

# 浏览器访问虚拟机IP:8089



vim nginx-1.16.1/objs/Makefile

# makefile决定了源文件之间的依赖关系,定义了编译器、源代码、依赖、主键、需要产生的二进制文件

# jenkins构建 一个maven的项目

源码管理-> SVN仓库地址-> 账户和密码

系统配置-> 全局工具配置-> Maven ->新增Maven->name: MAVEN-> MAVEN\_HOME:/usr/local/maven/

JDK->新增JDK->别名: JAVA\_HOME-> JAVA\_HOME->: /usr/local/java/jdk1.8.0\_131/

配置-> Build-> Root POM: pom.xml->Goals and options clean install-Dmaven.test.skip=true

# 不对网站源代码做功能测试

rz maven-3.6.3-bin.tar.gz

mv maven-3.6.3-bin.tar.gz /usr/local/

cd /usr/local/

tar -xzf maven-3.6.3-bin.tar.gz

# jenkin构建上面配置的maven的项目，构建完成后

cd /root/.jenkins/workspace/www.jfedu.net/target/

mkdir -p /tmp/`date +%F`/

cp edu.war /tmp/20200108/

jar -xvf edu.war # 解压网站代码

点击网站项目->Build-> Post Steps-> Run regardless of build result

执行shell

cd /data/docker/

\cp /root/.jenkins/workspace/www.jfedu.net/target/edu.war ./

docker build -t tomcat-www:latest .

docker run -itd -p 8082:8080 tomcat-www:latest

docker run -itd -p 8083:8080 tomcat-www:latest

# 服务器新建一个Dockerfile

cd /data/docker/

vim Dockerfile

FROM docker.io/tomcat

MAINTAINER JFEDU.NET

RUN rm -rf /usr/local/tomcat/webapps/\*; mkdir -p /usr/local/tomcat/webapps/ROOT/

COPY /root/.jenkins/workspace/www.jfedu.net/target/edu.war /usr/local/tomcat/webapps/ROOT/

RUN cd /usr/local/tomcat/webapps/ROOT/; jar -xvf edu.war

jenkins-> 立即构建www.jfedu.net项目 -> 构建历史

浏览器访问一下网站地址：服务器IP+端口

maven pom.xml # 编译源代码文件生成二进制文件(适用于java开发的网页)

# 给容器做磁盘配额

df - Th # 查看磁盘分区类型

磁盘必须为xfs格式

# 查看磁盘配额详情

xfs\_quota -x -c 'report' /data/

# 在VMware添加一块新的硬盘

fdisk -l

mkfs.xfs -f /dev/sdb # 格式化硬盘为xfs格式

mount -o uquota,prjquota /dev/sdb /data/ # 挂载data目录，并开启磁盘配额功能

df -h # 查看挂载是否成功

xfs\_quota -x -c 'report' /data/ # 查看配额使用情况

#限制jfedu用户对/data/目录只有10M的操作权限

xfs\_quota -x -c 'limit bsoft=10M bhard=10M jfedu' /data/

xfs\_quota -x -c 'report' /data/

chmod o+w /data/ -R

cd /data/

dd if=/dev/zero of=test.img bs=1M count=20 #写入测试，创建1M的文件，写入20次# 会提示超过10M大小的报错信息

rm -rf test.img

mkdir docker

cd /var/lib/

rm -rf docker/

ln -s /data/docker/ .

systemctl restart docker

vim /etc/sysconfig/docker

可以指定默认容器的大小（在启动容器的时候指定），可以在docker配置文件：/etc/sysconfig/docker中，OPTIONS参数后面添加如下代码,指定Docker容器rootfs容量大小为40G：(只适用于新容器生成，并且修改后需要重启docker，无法做到动态给正在运行容器指定大小)

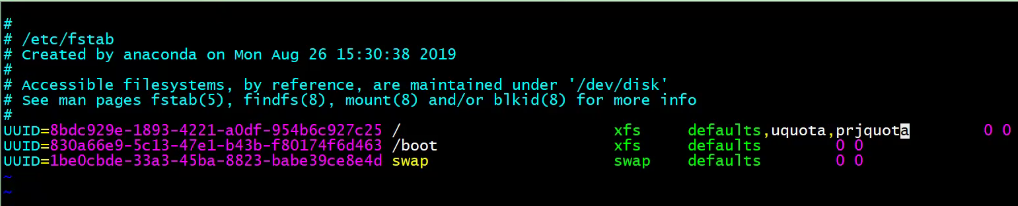
OPTIONS='--storage-opt overlay2.size=40G'

vim /etc/sysconfig/docker-storage

DOCKER\_STORAGE\_OPTIONS="--storage-driver overlay2 "

# 开启根目录的配额功能

vim /etc/fstab



#给容器指定逻辑CPU个数 和 内存 大小

docker run -itd --privileged --name=jfedu\_net\_170 --net=none --cpuset-cpus=0-0 -m 2048m centos:v3 /bin/bash

top # 查看CPU使用情况，按数字1可以查看所有逻辑CPU

docker inspect CONTAINER ID | grep -i "cpu" # 查看该容器使用哪个CPU

cat /proc/cpuinfo # 查看系统有几个CPU

# 给容器动态分配内存

docker update -m 512m CONTAINER ID

docker stats CONTAINER ID --no-stream # 静态查看容器运行状态

# 给容器动态分配CPU

docker update --cpuset-cpus 0-0 CONTAINER ID

docker update --cpuset-cpus 0-1 CONTAINER ID

docker stats CONTAINER ID --no-stream # 静态查看容器运行状态

docker stats CONTAINER ID --no-stream|awk 'NR>1 {print "CPU Usage:"$2}'

SVN服务器仓库地址：  
svn://139.224.227.121:8801/edu/vv1

用户名：shareku1  
密码：www.jfedu.netQAZ

grant all on ultrax.\* to root@"172.17.0.\*" identified by "123456";

flush privileges;

tar -czf discuz.tar /var/www/html/ # 打包网站代码

svn checkout svn://139.224.227.121:8801/edu/discuz/ vv6.jfedu.net

# svn客户端把svn仓库的目录内容下载到本机的目录中

cd discuz/

cp -a ../\* .

svn add \* # 把本机文件上传到SVN

svn commit -m bbs # 提交文件 到svn并 发送消息

du -sh .

vim Dockerfile

FROM centos:v3

MAINTAINER wangwei

copy /root/.jenkins/workspace/bbs.jfedu.net/ /var/www/html/

CMD systemctl start httpd;s;/bin/bash

jenkins执行shell

cd /data/docker/

\cp -a /root/.jenkins/workspace/discuz.jfedu.net discuz.jfedu.net

docker build -t centos7 .

for i in `seq 81 90`;do docker run -itdd --privileged -p 60$i:22 -p $i:80 centos7:v6 ;done

容器要连接mysql所在宿主机的docker0的 ip

进入容器修改PHP数据库连接配置

vim config\_global.php

-------CONFIG DB--------

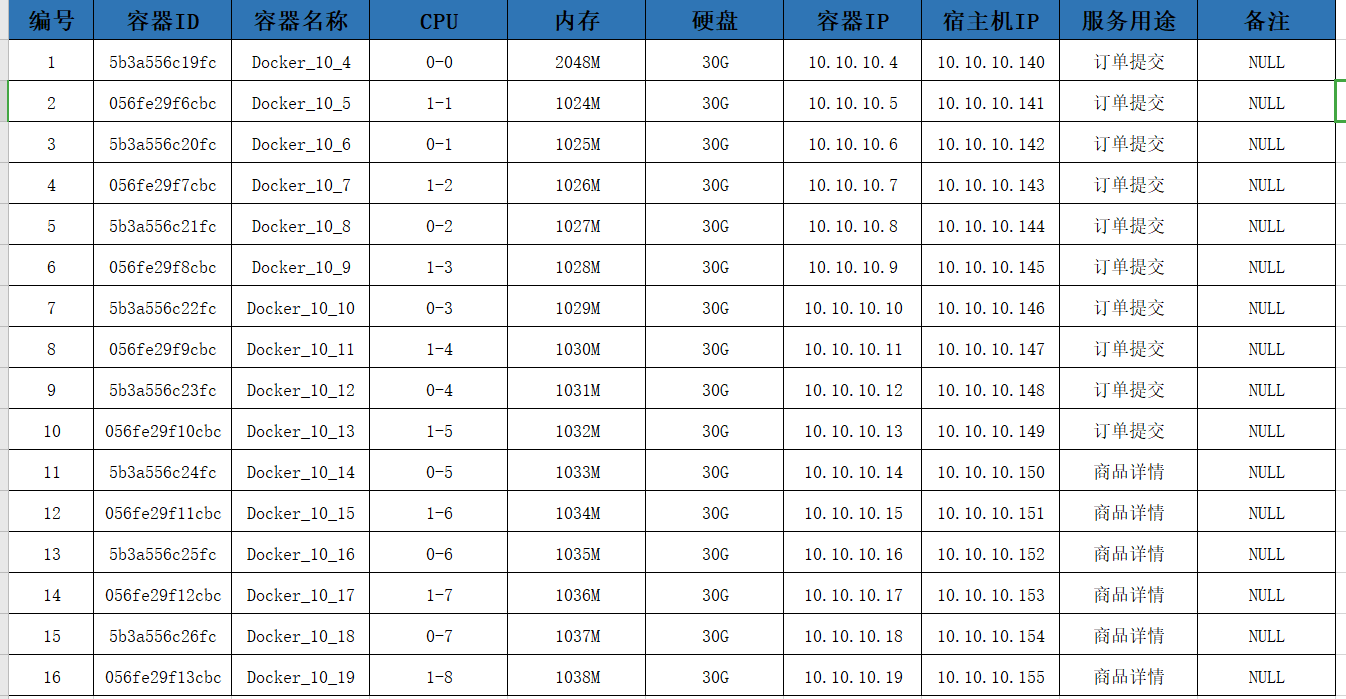
$\_config['db']['1']['dbhost'] = '172.17.0.1'; #修改成宿主机的IP（docker0）

systemctl stop iptables

cd /usr/local/tomcat/webapps/

chmod o+w ROOT/ -R

浏览器访问网站IP+端口



部署ELK环境：

elasticsearch logstash kibana必须是同一版本，且分别部署在不同的主机上，elasticsearch和logstash需要jdk才能运行，必须先安装jdk

上传jdk并解压到 /usr/local/jdk/

vim /etc/profile

在末行加入下列代码

export JAVA\_HOME=/usr/local/jdk1.8.0\_131/

export CLASSPATH=$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib

source /etc/profile

上传elasticsearch和logstash, 并解压到/usr/local/ 下

cd /usr/local/elasticsearch-6.8.5

vim config/jvm.options # 修改内存堆大小

-Xms1g

-Xmx1g

# elasticsearch需要 占用 物理内存的 四分之三

vim config/elasticsearch.yml

network.host: 0.0.0.0 # 监听所有的IPv4 host地址

usradd elasticsearch

chown -R elasticsearch:elasticsearch /usr/local/elasticsearch/

su - root

vim /etc/security/limits.conf

\* soft nofile 65536

\* hard nofile 65536

vim /etc/security/limits.d/20-nproc.conf

soft nproc 2048

vim /etc/sysctl.conf

vm.max\_map\_count=655360

sysctl -p # 使配置生效

su - elasticsearch

/usr/local/elasticsearch-6.8.5/bin/elasticsearch -d # 启动elasticsearch服务

ps -ef | grep elasticsearch

ps -ef | grep java

tail -fn 30 /usr/local/elasticsearch-6.8.5/logs/elasticsearch.log

netstat -ntlp|grep -E "9200|9300"

# 9200 数据存储通讯端口 9300 节点通讯端口

到另一台虚拟机 上传kibana并解压到/usr/local/

cd /usr/local/kibana-6.8.5/config/

vim kibana.yml

server.host: "0.0.0.0"

elasticsearch.hosts: ["http://192.168.0.200:9200"] # 修改为elasticsearch所在的主机

i18n.locale: "zh-CN" # 汉化kibana

cd /usr/local/kibana-6.8.5/bin/

nohup ./kibana & # 后台启动kibana

ps -ef|grep node

netstat -ntlp|grep 5601

# 重启 kibana

ps -ef|grep node

kill -9 node的进程号

nohup ./kibana &

浏览器访问192.168.0.161:5601

# 到logstash所在的主机创建键盘输入配置文件, 以便收集日志

cd /usr/local/logstash-6.8.5/bin/

vim keyboard\_input.yml

input {

stdin { }

}

output {

stdout {

codec => rubydebug {}

}

elasticsearch {

hosts => "192.168.0.200:9200" }

}

cd /usr/local/logstash-6.8.5/bin/

./logstash -f keyboard\_input.yml # 前台启动logstash服务来收集键盘输入的信息日志

www.jfedu.net Test logs for ES # 手动输入测试日志数据

# 所有日志文件存放在/usr/local/elasticsearch-6.8.5/data/nodes/ 这个目录下

nohup ./logstash -f ../config/nginx.yml & #后台启动logstash来收集nginx的访问日志

# 浏览器 -> kibana -> management -> 创建索引模式

man iptables-extensions

iptables -t nat -L -n --line-numbers

service iptables restart

alias l=ls # 设置 别名(重启bash就失效)

unalias l # 取消别名

alias -p # 查看当前所有别名

vim ~/.bashrc # 持久化保存alias

alias a=ls

# 安装并配置Iptables

yum install iptables iptables-utils iptables-devel iptabales-services -y

或者 yum install iptables\* -y

systemctl start iptables.service # 启动iptables服务

lsmod | grep filter # lsmod检测系统当前加载了哪些模块

systemctl status iptables # 查看iptables是否开启

vim /etc/sysconfig/iptables

\*filter

:INPUT DROP [0:0] # 前面0表示进入这个链的请求数, 后面0表示请求包的字节大小

:FORWARD ACCEPT [0:0]

:OUTPUT ACCEPT [0:0]

-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT # 添加远程连接端口

-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT

# -A顺序添加一条新规则 -p指定协议 -m state调用state扩展模块 --state指定状态NEW当SSH客户端第一个数据包到达服务器时, EATABLISHED建立连接后数据包的状态字段, RELATED数据包正在建立一个新的连接，这个连接是和一个已建立的连接相关的 -m tcp 调用tcp模块功能 -dport目标端口 -j执行目标

# 默认规则优先级最低, 配置规则按顺序执行

-A INPUT -p tcp -m limit --limit 1/minute -m tcp --dport 80 -j ACCEPT

# 限制每分钟 只能访问 一次本机80端口

-A INPUT -p tcp -m tcp -m time --timestart 23:00 --timestop 00:00 --dport 80 -j DROP

# 23点到凌晨0点 不能 访问本机80端口

-A INPUT 4 -m mac --mac-source 00:E0:4C:36:65:78 -p tcp -m tcp --dport 80 -j DROP

# 禁止 该MAC地址访问本机80端口

# 开放mysql端口

-A INPUT -p tcp -m state --state NEW -m tcp --dport 3306 -j ACCEPT

# 允许接收远程主机的HTTP请求   
iptables -A INPUT -i eth0 -p tcp –dport 80 -m state –state NEW,ESTABLISHED -j ACCEPT

# 允许发送本地主机的HTTP响应  
iptables -A OUTPUT -o eth0 -p tcp –sport 80 -m state –state ESTABLISHED -j ACCEPT

# 禁止没有通过请求回应的数据包从本机发出

iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

-A FORWARD -s 192168.0.0/24 -p tcp -m tcp -j ACCEPT

# 添加允许192.168.0.0网段从此主机转发到其它主机的规则

-A OUTPUT -o ens33 -p tcp -m tcp --dport 80 -j ACCEPT

# 指定数据包出去的网卡

-A INPUT -j REJECT --reject-with icmp-host-prohibited

# 拒绝ping本机并返回消息

systemctl restart iptables

iptables -L -n

# 查看防火墙规则，防火墙规则 从前往后匹配，匹配到了就不再执行后面的规则

iptables -L -n -t nat

# 查看nat表上面的规则

# 在linux中执行命令 也可以 实现添加规则

iptables --help

iptabless -t filter -I INPUT 4 -p tcp -m tcp --dport 80 -j ACCEPT

# -I INPUT 4 插入到INPUT链中第四行 不能添加到最后一行, 顺序执行后会不生效

iptables -t filter -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT

# linux下直接添加规则到链中

iptables -t filter -A INPUT -p icmp -s 192.168.0.161 -j ACCEPT

# 允许192.168.0.161 ping本机

iptables -A INPUT -p tcp -s 192.168.0.200 -j DROP

# 禁止192.168.0.200访问本机

iptables -L -n --line-numbers

# 查看带编号的防火墙规则

iptables-save >/etc/sysconfig/iptables

# 配置导入文件, 永久生效

service iptables save

# 把当前内存中的配置保存到配置文件中

iptables -D INPUT 4

# 删除INPUT链的第四条规则

iptables -L -n --line-numbers

# 查看带编号的防火墙规则

iptables -I INPUT 4 -p tcp --dport 80 -j DROP

# 禁止所有人访问80端口

iptables -I INPUT 4 -s 192.168.0.161/32 -p tcp -m tcp --dport 80 -j DROP

# 禁止这个IP访问 本机80端口

iptables -I INPUT 4 -s 192.168.0.0/16 -p tcp -m tcp --dport 80 -j DROP

# 禁止192.168.0.0这个网段访问 本机80端口

iptables -I INPUT 4 -m mac --mac-source 00:E0:4C:36:65:78 -p tcp -m tcp --dport 80 -j DROP

# 禁止该MAC地址主机访问本机80端口

iptables -t nat -A PREROUTING -d 192.168.0.161/32 -p tcp -m tcp --dport 8080 -j DNAT --to-destination 192.168.0.162:80

iptables -t nat -A POSTROUTING -d 192.168.0.162/32 -p tcp -m tcp --dport 80 -j SNAT --to-source 192.168.0.161

iptables -t nat -L -n --line-numbers # 查看规则是否添加成功

echo 1 > /proc/sys/net/ipv4/ip\_forward # 开启iptables forward路由转发功能

# 将本机的8080端口转发至其他主机,源主机IP:192.168.0.161,目标主机IP和端口192.168.0.162:80

systemctl restart iptables # 重启iptables使新增规则生效

# firewall的使用方法

iptables和firewall同一台主机只能启动一个服务

systemctl stop iptables

systemctl start firewalld

firewall-cmd --get-zones # 查看所有firewall所有区域

firewall-cmd --get-default-zone # 查看firewall默认使用区域

firewall-cmd -h # 查看所有相关参数

firewall-cmd --list-all # 查看当前firewall规则

# 配置firewall规则

vim /etc/firewalld/zones/public.xml

<service name="http"/>

<port tcp="80"/>

# 主配文件中<service name="http"> 对应 模板文件中配置的端口

vim /usr/lib/firewalld/services/ssh.xml

<port protocol="tcp" port="80"/>

# 主配文件中<service name="ssh"> 对应 模板文件中配置的端口

vim /usr/lib/firewalld/services/ssh.xml

<port protocol="tcp" port="22"/>

# 如果没有该服务模板文件就创建一个模板

cd /usr/lib/firewalld/services/

cp http.xml tomcat.xml

vim tomcat.xml

<port protocol="tcp" port="8080">

systemctl reload firewalld # 重启firewall使规则生效

# 第二种 使用方法

firewall-cmd --add-port=8888/tcp --permanent # 添加端口

firewall-cmd --remove-port=8888/tcp --permanent # 移除端口

# 批量添加不连续的端口

vim port.txt

22

8080

3306

5601

for i in `cat port.txt`;do firewall-cmd --add-port=$i/tcp --permanent ;done

systemctl reload firewalld # 使firewall规则生效

firewall-cmd --reload

firewall-cmd --list-all # 查看当前规则

firewall-cmd --zone=work --list-all # 查看work区域规则

firewall-cmd --zone=public --list-all # 查看public区域规则

# 允许192.168.0.142来访问本机6379端口

firewall-cmd --permanent --add-rich-rule="rule family="ipv4" source address="192.168.0.142" port protocol="tcp" port="6379" accept"

# 限制192.168.0.142来访问本机6379端口

firewall-cmd --permanent --add-rich-rule="rule family="ipv4" source address="192.168.0.142" port protocol="tcp" port="6379" drop"

#将内网主机局域网代理上网,启用区域伪装功能,私有网络地址被隐藏并映射到一个公有IP

firewall-cmd --permanent --zone=<ZONE> --add-masquerade

firewall-cmd --permanent --zone=public --add-rich-rule="rule family=ipv4 source address=192.168.0.0/24 masquerade"

#访问本地端口映射至远程主机端口

firewall-cmd --add-masquerade --permanent

firewall-cmd --permanent --zone=public --add-forward-port=port=8080:proto=tcp:toaddr=192.168.0.142:toport=80

# 将访问192.168.0.141的8888端口的请求转发至22端口( 必须先禁止外部访问22端口)

firewall-cmd --permanent --zone=public --add-forward-port=port=8888:proto=tcp:toport=22:toaddr=192.168.0.141

# kubernetes搭建与使用

setenforce 0

systemctl stop firewalld

systemctl disable firewalld

yum install -y ntp

ntpdate pool.ntp.org

systemctl start ntpd

systemctl enable ntpd

# master主机操作

yum install -y kubernetes-master flannel etcd

vim /etc/etcd/etcd.conf

ETCD\_DATA\_DIR="/data/etcd/"

ETCD\_LISTEN\_CLIENT\_URLS="http://localhost:2379,http://192.168.0.200:2379"

ETCD\_NAME="default"

ETCD\_ADVERTISE\_CLIENT\_URLS=http://localhost:2379,http://192.168.0.200:2379

mkdir -p /data/etcd/

chown -R etcd:etcd /data/etcd/

systemctl restart etcd

vim /etc/kubernetes/apiserver

KUBE\_API\_ADDRESS="--insecure-bind-address=0.0.0.0"

KUBE\_API\_PORT="--port=8080" # 此处使用的端口不能被其他程序占用

KUBE\_ETCD\_SERVERS="--etcd-servers=http://19.168.0.200:2379"

# K8S Master主配置文件, 指定Master Apiserver的IP+端口

vim /etc/kubernetes/config

KUBE\_ALLOW\_PRIV="--allow-privileged=true"

KUBE\_MASTER="--master=http://192.168.0.200:8080"

vim /etc/sysconfig/flanneld

FLANNEL\_ETCD\_ENDPOINTS="http://192.168.0.200:2379"

# 在etcd所在的服务器执行

etcdctl mk /atomic.io/network/config '{"Network":"172.17.0.0/16"}'

etcdctl ls / # 查看集群中的key是否创建成功

etcdctl get /atomic.io/network/config # 查看key的值

etcdctl ls /atomic.io/network/subnets # 查看有哪些key

etcdctl get /atomic.io/network/subnets/172.17.0.54.0-24 # 查看此IP段所属的宿主机

systemctl restart etcd

# 为集群中的所有节点重新规划 IP 地址的使用规则，从而使得不同节点上的容器能够获得同属一个内网且不重复的 IP 地址，并让属于不同节点上的容器能够直接通过内网 IP 通信

systemctl restart flanneld

# 用户和整个K8S集群交互操作的统一入口,所有的操作,配置,资源申请都需经过Apiserver接口

systemctl restart kube-apiserver

# K8S副本控制配置文件，用于控制后期容器资源副本数，时刻保障副本的正常运行

systemctl restart kube-controller-manager

# K8S节点资源调度配置文件,用于调度和计算节点的可用资源,将资源申请需求进行调度和分配

systemctl restart kube-scheduler

# minion主机操作

yum install -y kubernetes-node flannel docker \*rhsm\*

# rhsm是系统需要使用的redhat认证证书

yum install bash-completion -y

bash # 开启kubectl自动补全功能

vim /etc/kubernetes/config

KUBE\_ALLOW\_PRIV="--allow-privileged=true"

KUBE\_MASTER="--master=http://192.168.0.200:8080"

vim /etc/kubernetes/kubelet

KUBELET\_ADDRESS="--address=0.0.0.0"

KUBELET\_HOSTNAME="--hostname-override=192.168.0.161" # minion的IP

KUBELET\_API\_SERVER="--api-servers=http://192.168.0.200:8080" # master的IP和端口

vim /etc/sysconfig/flanneld

FLANNEL\_ETCD\_ENDPOINTS="http://192.168.0.200:2379"

systemctl restart flannel

systemctl restart docker

# 重启docker后docker0的网段和flannel在同一网段

# 实现K8S代理转发容器端口,可以实现Forward或者NAT地址转换,后期Service服务访问,感知

systemctl restart kube-proxy

# 跟本地的Docker引擎服务通信,负责连接并控制Docker,给Docker引擎发送任务,管理Docker容器（启动、停止、删除、重启等）

systemctl restart kubelet

systemctl restart flanneld docker

# 在master查看node信息

kubectl get nodes

kubectl delete nodes/192.168.0.161 #删除节点,重启节点服务即可恢复,一般不可轻易删除

# master主机 修改apiserver配置文件

vim /etc/kubernetes/apiserver

KUBE\_ADMISSION\_CONTROL="--admission-control=NamespaceLifecycle,NamespaceExists,LimitRanger,SecurityContexDeny,ServiceAccount,ResourceQuota"

# 删掉ServiceAccount认证服务账号

# Kubernetes-Dashboard安装 在minion端

上传两个镜像并导入到docker

docker pull registry.cn-hangzhou.aliyuncs.com/google\_containers/kubernetes-dashboard-amd64:v1.10.1

kubernetes-dashboard-amd64 pod-infrastructure

#在master端创建这两个文件

dashboard-controller.yaml dashboard-service.yaml

vim dashboard-controller.yaml

spec:

image: kubernetes-dashboard-adm64:v1.10.1

args:

- --apiserver-host=http://192.168.0.200:8080

kubectl create -f dashboard-controller.yaml

kubectl create -f dashboard-service.yaml

# 如果创建出错, 就要删掉原来的,再重新创建

kubectl delete -f dashboard-controller.yaml

kubectl delete -f dashboard-service.yaml

iptables -P FORWARD ACCEPT # 每台服务器都要执行,每次开机都要重新运行

kubectl get pods --namespace kube-system # 查看 创建的dashboard是否成功

kubectl describe pods/kubernetes-dashboard-3572929714-3sk2q --namespace kube-system # 查看容器日志信息

#浏览器访问192.168.0.200:8080/ui/

overview -> 创建 ->

应用名称: nginx

容器镜像:docker.io/nginx

容器组数量:1

服务: 外部 （通过node的IP+端口访问（通过容器对应的服务来查看端口））

端口: 源端口 -> 目标端口

ps -ef|grep kube

kubectl get nodes

yum install -y traceroute

traceroute 172.17.94.2 # 抓取路由查看网络是如何到达docker

**配置ServiceAccount**

1、首先生成密钥：   
openssl genrsa -out /etc/kubernetes/serviceaccount.key 2048

2、编辑/etc/kubenetes/apiserver   
添加以下内容：   
KUBE\_API\_ARGS="--service\_account\_key\_file=/etc/kubernetes/serviceaccount.key"

3、再编辑/etc/kubernetes/controller-manager   
添加以下内容：   
KUBE\_CONTROLLER\_MANAGER\_ARGS="--service\_account\_private\_key\_file=/etc/kubernetes/serviceaccount.key"

无论是哪种解决方式都需要再重启kubernetes服务：   
systemctl restart etcd kube-apiserver kube-controller-manager kube-scheduler

yum install -y kubernetes

kubectl describe pods/v1-jfedu-net-2422990518-dtfk9 -n default

kubectl get deployments -n default

kubectl get pods -n default

kubectl describe pods/部署名称 -n default # 查看 镜像 名称

kubectl get nodes

kubectl get pods

# 抓取192.168.0.200:80端口的数据

yum install -y tcpdump

tcpdump -i ens33 -nn host 192.168.0.200 and port 80

curl 192.168.0.200

ack=1就代表已经连接上了

ulimit -n # 查看用户最大可打开的文件数

ulimit -n 65535

vim /etc/security/limits.conf

\* soft noproc 65535 # number of process

\* hard noproc 65535

\* soft nofile 65535 # number of open file

\* hard nofile 65535

cat /proc/sys/fs/file-max # 查看当前系统最大可打开的文件数

echo 3865161233 > /proc/sys/fs/file-max # 设置 系统最大可打开的文件数

ulimit -SHn 102400 # 设置最大可打开的文件数

vim /etc/sysctl.conf # 存储内核参数的配置文件

sysctl -p # 使配置参数生效

top回车

按数字1显示 逻辑CPU

top -b 静态显示

负载 -> 所有逻辑CPU的个数加起来 乘以2就是最大负载

free -m

free + shared + buffers + cached

yum provides vmstat # 查询是由哪个工具包提供的该命令

whereis vmstat # 查询是由本机哪个安装包提供的

vmstat 2 # 每隔2秒输出一次CPU性能

iostat -x 5 # 每隔5秒 查看一次读写性能的详细信息

dd if=/dev/zero of=test.img bs=1M count=2000 # 写入数据测试硬盘io

while true;do ab -c 100 -n 1000 -r http://www.qq.com/index.html; done

sar -u 2 10 # 查看CPU使用率 -u显示详细信息 每隔2秒打印一次总共打印10次

iotop # 看哪个程序的IO最高

iftop -i ens33(或者eth0) # 查看网络流量情况rates: 第一个值

cat /proc/cpuinfo # 查看CPU参数

uptime

netstat -n|grep 80 | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}';

vim /etc/sysctl.conf

kernel优化参数：

net.ipv4.ip\_forward = 0

net.ipv4.conf.default.rp\_filter = 1

net.ipv4.conf.default.accept\_source\_route = 0

kernel.sysrq = 0

kernel.core\_uses\_pid = 1

net.ipv4.tcp\_syncookies = 1

kernel.msgmnb = 65536

kernel.msgmax = 65536

kernel.shmmax = 68719476736

kernel.shmall = 4294967296

net.ipv4.tcp\_max\_tw\_buckets = 10000

net.ipv4.tcp\_sack = 1

net.ipv4.tcp\_window\_scaling = 1

net.ipv4.tcp\_rmem = 4096 87380 4194304

net.ipv4.tcp\_wmem = 4096 16384 4194304

net.core.wmem\_default = 8388608

net.core.rmem\_default = 8388608

net.core.rmem\_max = 16777216

net.core.wmem\_max = 16777216

net.core.netdev\_max\_backlog = 262144

net.core.somaxconn = 262144

net.ipv4.tcp\_max\_orphans = 3276800

net.ipv4.tcp\_max\_syn\_backlog = 262144 / 8192

net.ipv4.tcp\_timestamps = 0

net.ipv4.tcp\_synack\_retries = 1

net.ipv4.tcp\_syn\_retries = 1

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_mem = 94500000 915000000 927000000

net.ipv4.tcp\_fin\_timeout = 1

net.ipv4.tcp\_keepalive\_time = 30

net.ipv4.ip\_local\_port\_range = 1024 65535

sysctl -p 使参数生效