Linux 运维入门到高级全套系列

目　录

1. Linux 入门篇

1. 1 Linux 操作系统简介

Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。它能运行主要的 UNIX 工具软件、应用程序和网络协议。它支持 32 位和 64 位硬件。Linux 继承了 Unix 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

1991 年的 10 月 5 日，Linux 创始人林纳斯·托瓦兹（Linus Torvalds）

在 comp.os.minix 新闻组上发布消息，正式向外宣布 Linux 内核的诞生，1994 年 3 月，Linux 1.0 发布，代码量 17 万行，当时是按照完全自由免费的协议发布，随后正式采用 GPL（General Public License 的缩写，是一份 GNU 通用公共授权）协议。

Linux 具有如下优点：

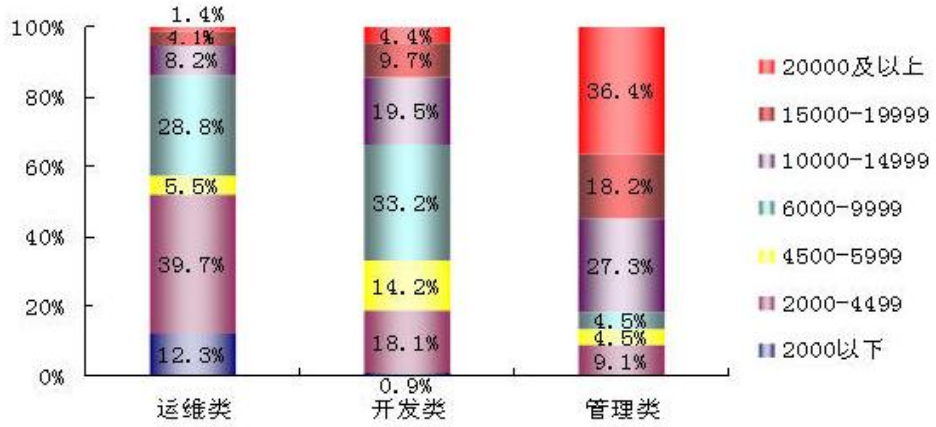
- 稳定、免费或者花费少
- 安全性高
- 多任务，多用户
- 耗资源少
- 由于内核小，所以它可以支持多种电子产品，如：Android 手机、PDA 等。

1.2 Linux 发展趋势

随着 IT 产业的不断发展，用户对网站体验要求也越来越高，而目前主流网站后端承载系统都是 Linux 系统，目前 Android 手机全部基于 Linux 内核研发。企业大数据、云存储、虚拟化等先进技术都是基于 Linux 系统。

2010 年据有关权威部门统计：将来几年内我国软件行业的从业机会十分庞大，中国每年对软件人才的需求将达到 50 万人左右。而对于 Linux 专业人才的就业前景，更是广阔；据悉在未来 5-10 年内 Linux 专业人才的需求将达到 120 万+！尤其是有经验的资深的 Linux 工程师目前非常的缺乏，薪资也是非常诱人，平均月薪都是

15-20K，能力强的薪资更高。



所以机会对每个人都是公平的，关键是我们每个人如何去行动，选择大于努力。

1.3 Linux 系统安装

在安装 Linux 系统之前，先来了解 windows 系统结构，windows 系统一般是安装在 C 盘系统盘，同样 Linux 也有类似的系统盘（/根分区），Linux 通常分区为（根分区/、swap 分区），Linux 系统以文件的存储方式，所有的文件都是存储在某个目录下的，类似于 windows 的文件夹。

对于文件系统的属性来说，windows 文件系统类型一般是 ntfs、fat32 等，而 Linux 文件系统类型则为 ext2、ext3、ext4 等（文件系统：是操作系统用于明确磁盘或分区上的文件的方法和数据结构，文件系统由三部分组成：与文件管理有关软件、被管理文件以及实施文件管理所需数据结构。）

安装 Linux 系统是每一个初学者的第一个门槛。在这个过程中间，最大的困惑莫过于给硬盘进行分区。虽然现在各种发行版本的 Linux 已经提供了友好的图形交互界面，但是很多人还是感觉无从下手。这其中的原因主要是不清楚 Linux 的分区规定。就好比如果我们了解了 windows 分区的规则，系统盘 C、数据盘 D 等，就很好分区了。

在 Linux 中规定，每一个硬盘设备最多只能有 4 个主分区（其中包含扩展分区）构成，任何一个扩展分区都要占用一个主分区号码，也就是在一个硬盘中，主分区和扩展分区一共最多是 4 个。

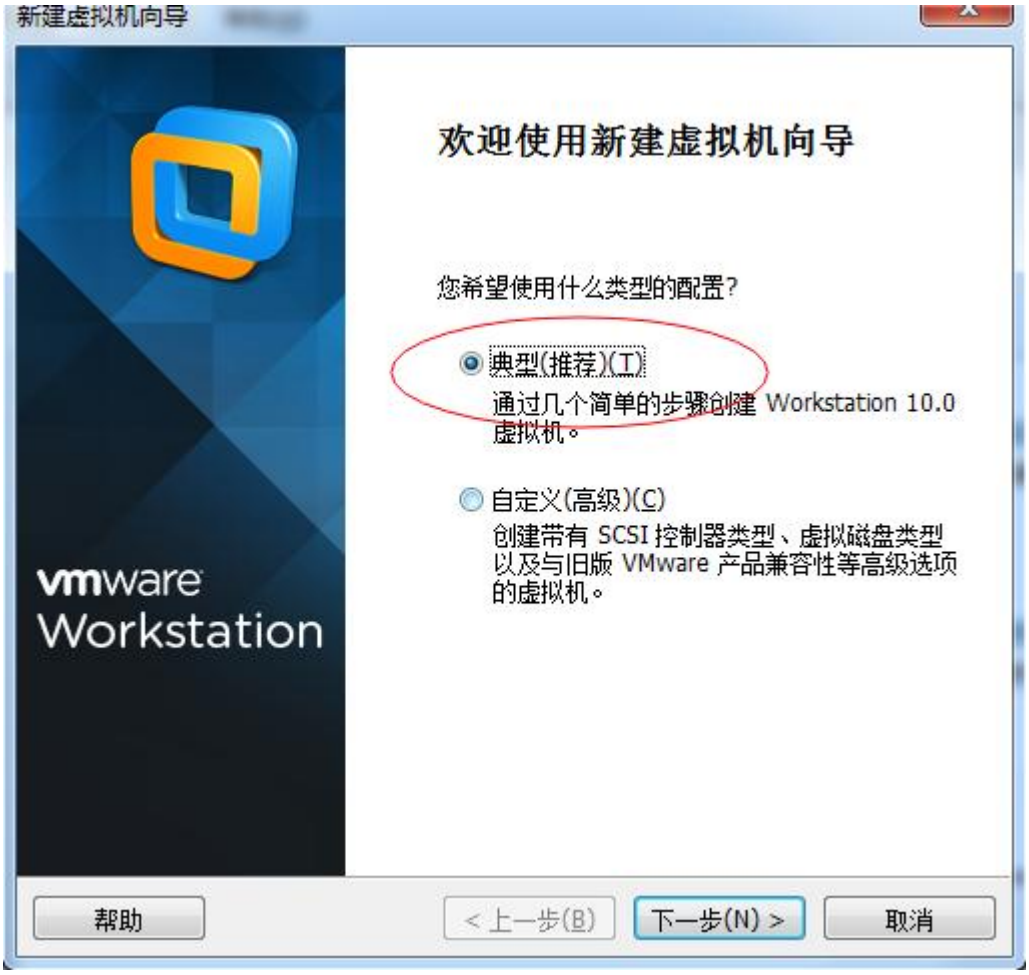
下面正式来安装 Linux 系统，安装系统前需要准备如下软件：

- ✓ VMware workstation 10.0
- ✓ CentOS 5.8 x86_i386.iso

安装图解如下：

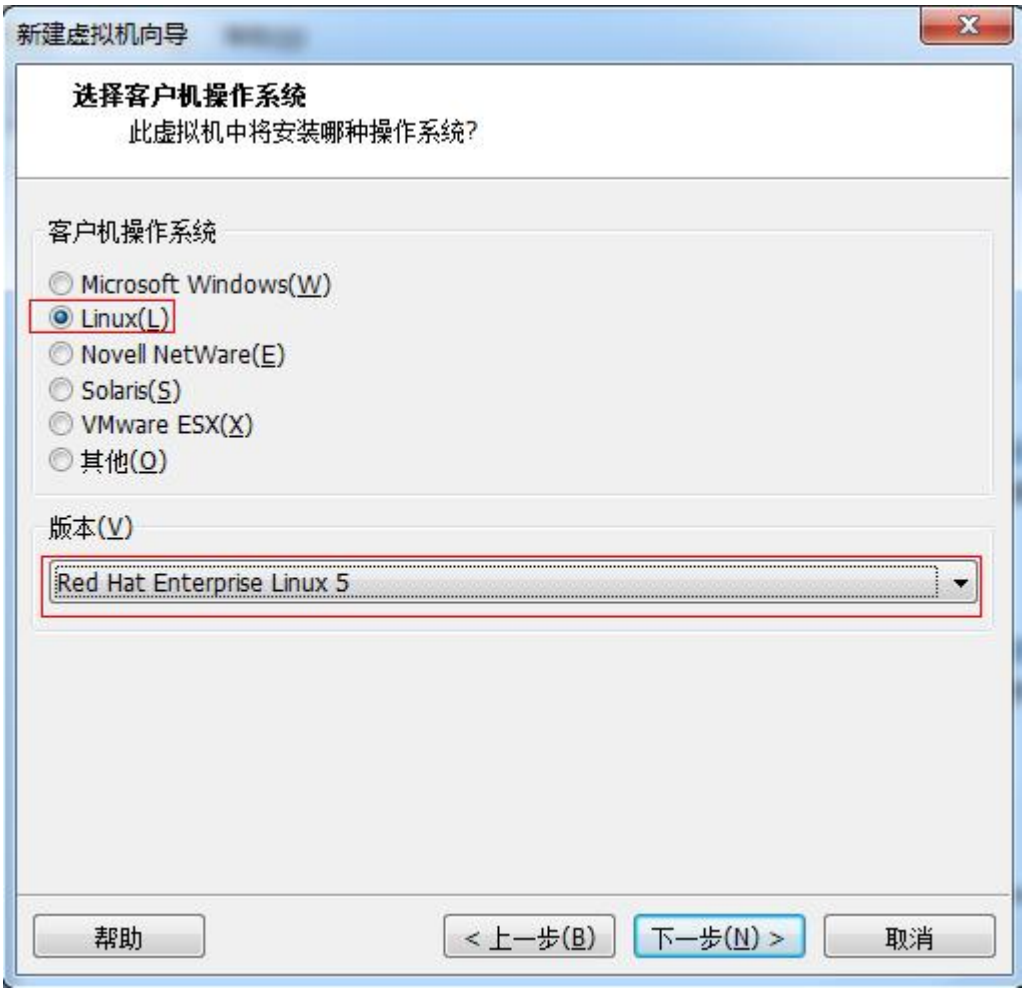
第一步，新建虚拟机如下图：

第二步，选择相关选项，如下图：



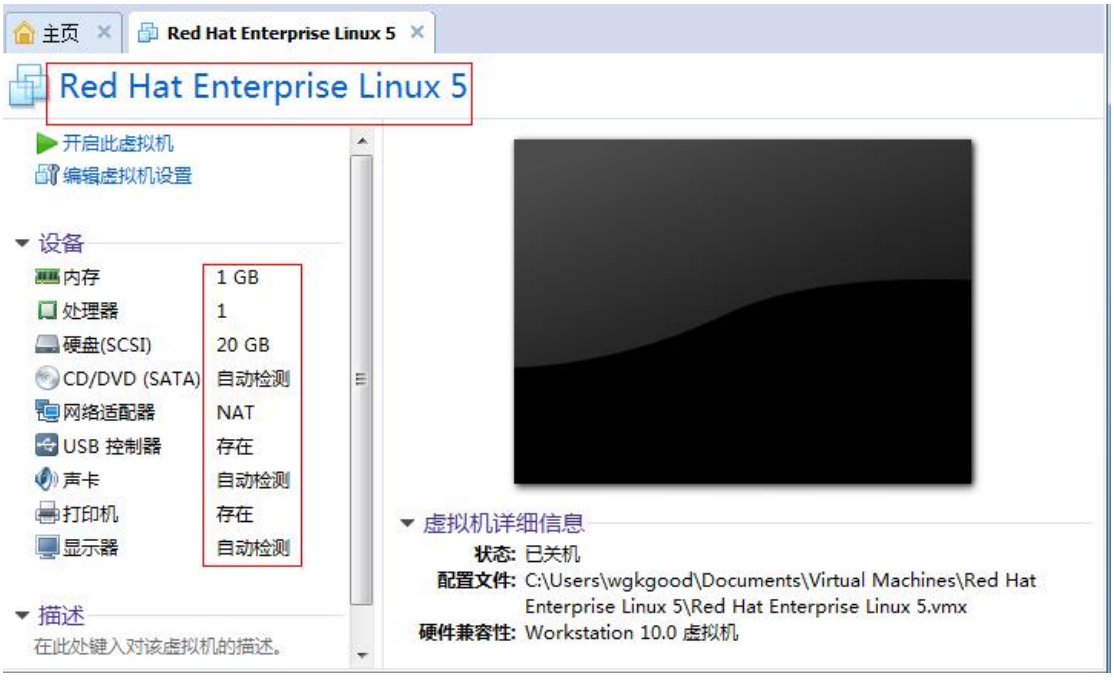
第三步选择“稍后安装操作系统”，如下图:

第四步，选择客户机操作系统类型如下图：

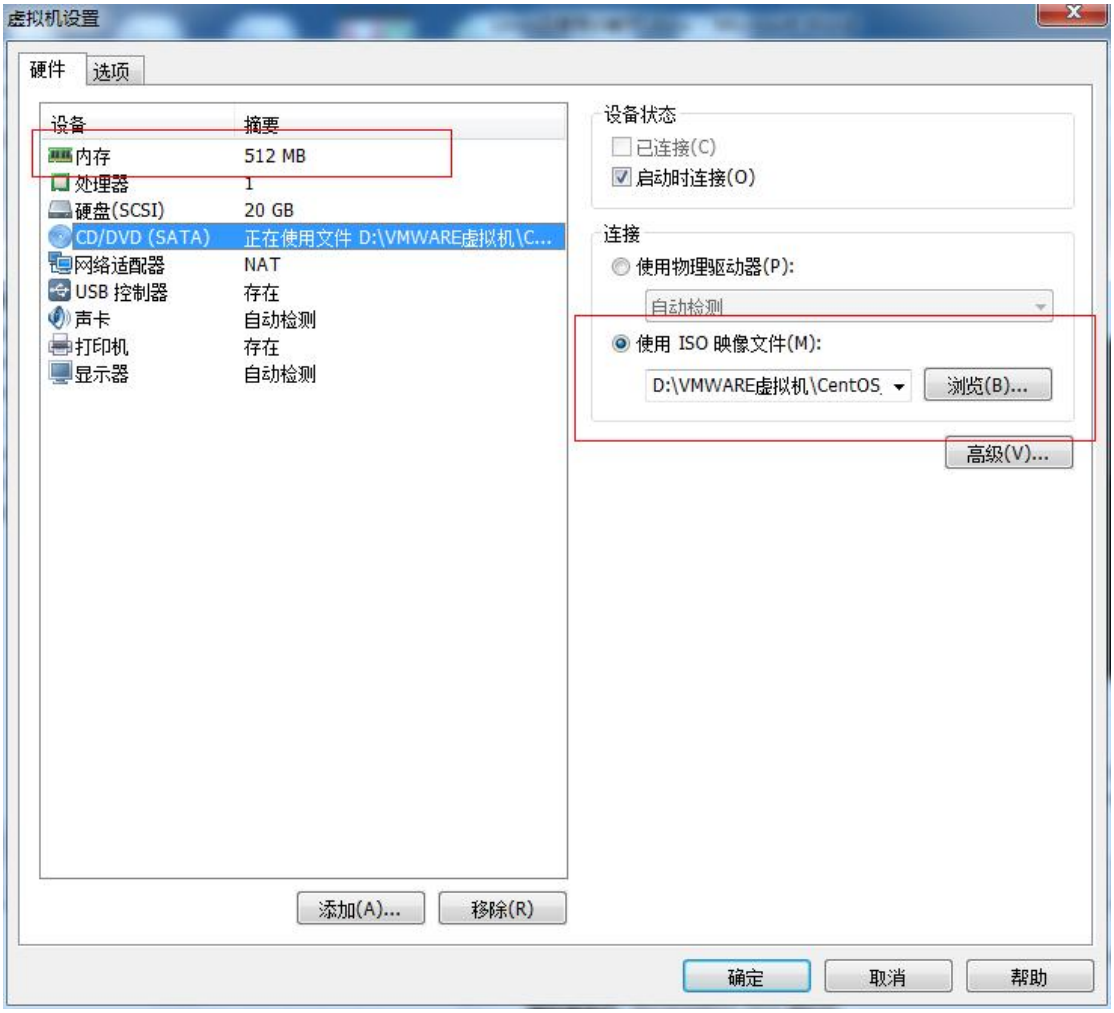


第五步，设置虚拟机硬盘大小为 20G，最低不能小于 5G，如下图：

第六步，虚拟机新建完成，如下图：

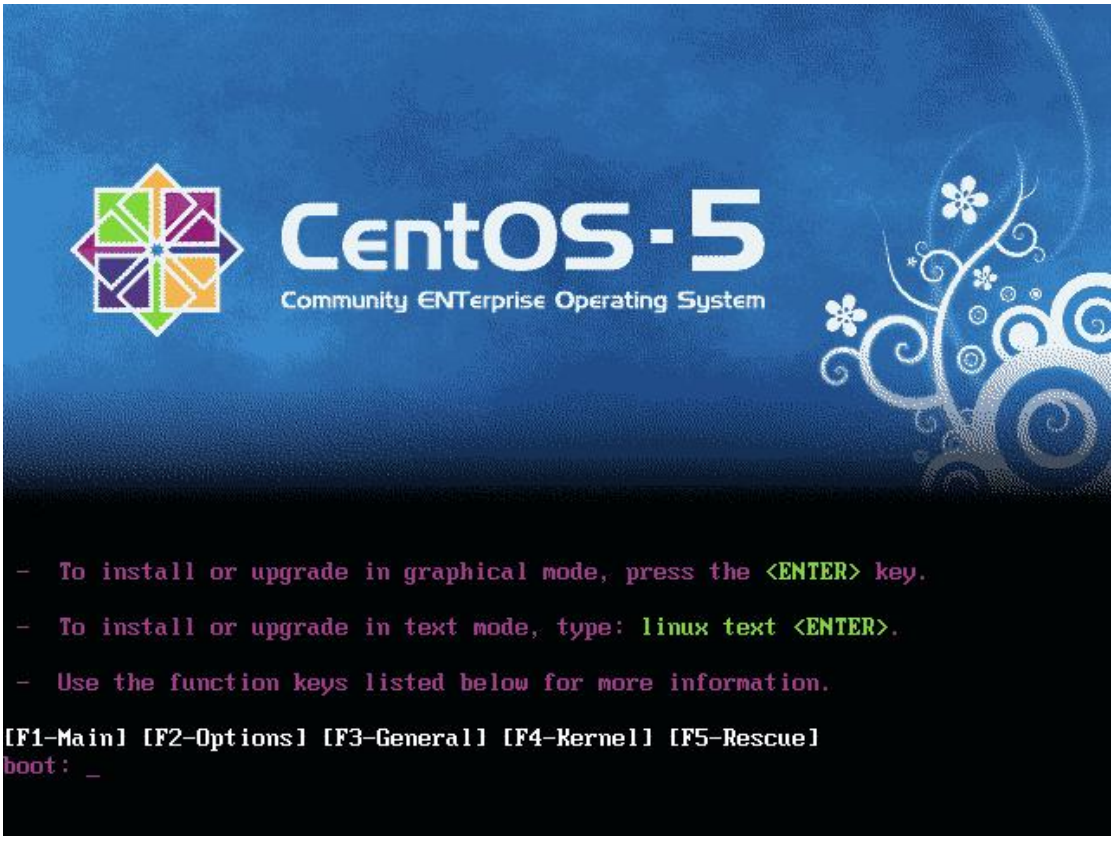


第七步，修改虚拟机内存为 512M，并添加 ISO 镜像，如下图：

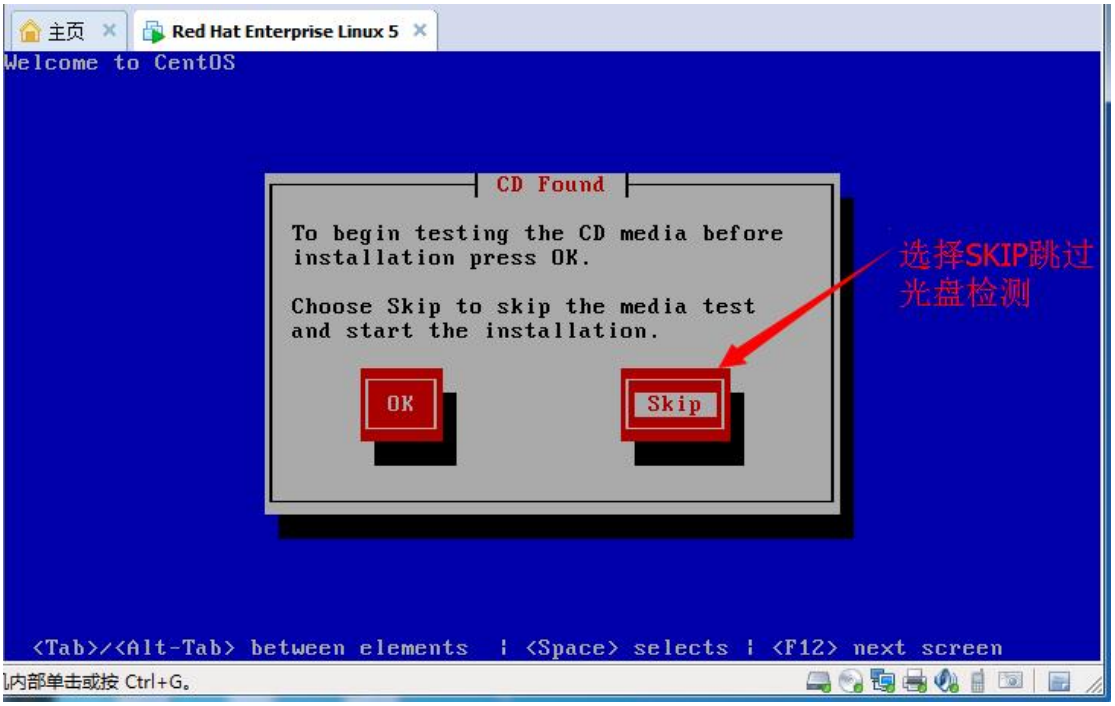


自此，虚拟机新建完成，接下来点击“启动此虚拟机”进行 Linux 系统安装，Linux 系统安装图解如下：

第一步，进入安装界面，直接按 Enter 回车键即可。



第二步，光盘检测，选择 SKIP 跳过。



第三步，选择安装过程中的语言，初学者可以选择“简体中文”。

第四步，选择初始化整个硬盘，清除所有数据。

第五步，选择分区方式为“自定义分区“。

第五步，点击“新建“-首先创建一个 swap 交换分区，大小为物理内存的 2 倍（1024M）。

第六步，继续创建分区，选择“新建“，然后创建根分区/，如下图选择，大小为剩余所有空间即可。

第七步，默认点击下一步，同时默认 DHCP 配置，时钟选择上海，去掉 UTC 勾，点击下一步。

第八步，设置 root 密码，至少六位，点击下一步。

第九步，系统安装包选择，这里选择“现在定制“。

第十步，系统安装包选择，左侧选择“开发“----右侧选择”开发工具“和“开发库”，语言选择“支持中文“，其他一概不选择。

安装完毕会提示“reboot“，直接回车即可。

1.4 Linux 学习技巧

初学者可以自己安装虚拟机，然后把 linux 常用命令例如 cd、ls、chmod、useradd、vi 等等多练习几十遍，把自己敲打命令的熟练程度提升上来。

然后根据文档搭建 Linux 下常见的各种服务（DHCP、SAMBA、DNS、Apache、Mysql 等），遇到问题后可以在 google 搜索，搜索的时候多看几篇文章，综合最好的文章来解决问题。

能够熟练的搭建服务后，理解每个服务的完整配置和优化，可以拓展思维。例如 LAMP，我们一般是把所有服务放在一台机器上，如果分开多台该如何部署呢？等等。

平时多积累 shell 编程，可以在网上查找前辈们写的非常好的 shell，自己下载下来多练习几遍，从中吸取，不断提高。

建立一个自己的学习博客，把平时工作学习中的知识都记录在里面，这样也可以供别人来参考同时也能提高自己的编写文档及方案的能力。

通过以上学习能够满足企业的一般应有，需要达到资深级别，还需要深入学习集群架构、负载均衡、自动化运维、运维开发等知识。最后还是一句话：多练习才是硬道理！实践出真知！

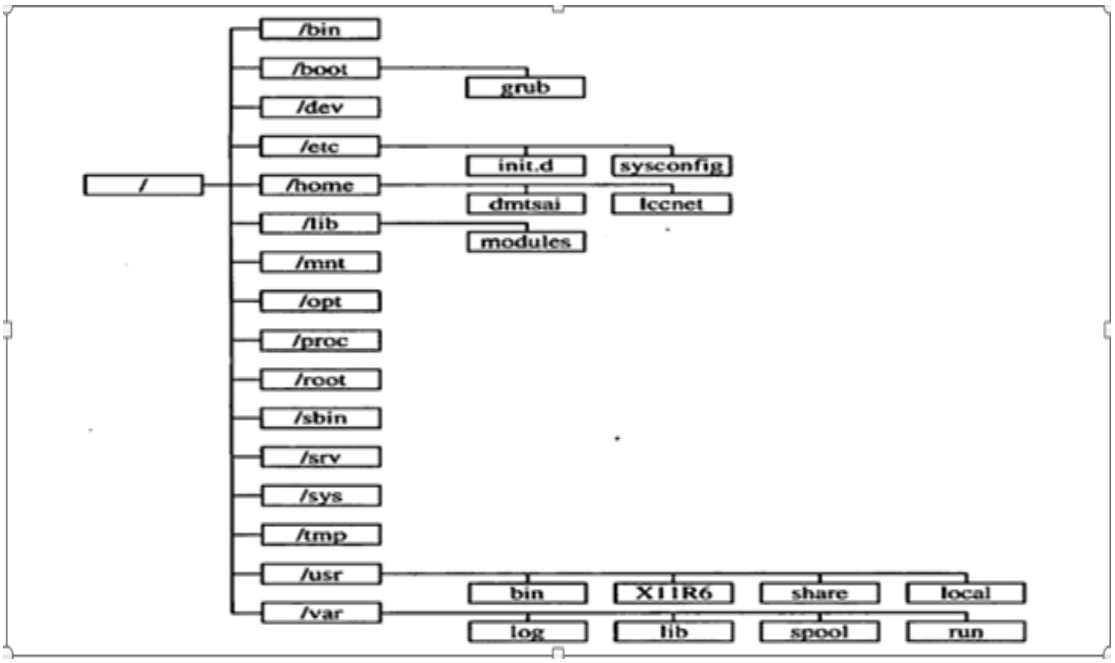
2. Linux 系统篇

2.1 Linux 系统管理

通过前两章的学习，我们已经能够独立安装 Linux 系统，已经掌握了 Linux 学习的技巧，那接下来，我们将系统的来了解 Linux 系统各目录、权限及常用命令的使用。

2.1. 1 Linux 目录初识

通过前面的学习,我们已经能够独立安装完一个 linux 系统，那接下来我们来熟悉一下 Linux 系统里面的各个目录文件夹的大致功能：主要的目录树的有/、/root、/home、/usr、/bin 等目录。下面是一个典型的 linux 目录结构如下：（附图表）



/ 根目录

/bin 存放必要的命令

/boot 存放内核以及启动所需的文件

/dev 存放设备文件

/etc 存放系统配置文件

/home 普通用户的宿主目录，用户数据存放在其主目录中

/lib 存放必要的运行库

/mnt 存放临时的映射文件系统，通常用来挂载使用。

/proc 存放存储进程和系统信息

/root 超级用户的主目录

/sbin 存放系统管理程序

/tmp 存放临时文件

/usr 存放应用程序，命令程序文件、程序库、手册和其它文档。

/var 系统默认日志存放目录

2.1. 2 Linux 常用命令

默认进入系统，我们会看到这样的字符: [root@localhost ~]#,其中#代表当前是 root 用户登录，如果是$表示当前为普通用户。

我们了解 linux 由很多目录文件构成，那我们来学习第一个 Linux 命令：

cd 命令， cd /home ；解析：进入/home 目录

cd /root 进入/root 目录 ； cd ../返回上一级目录;cd ./当前目录；（.和..可以理解为相对路径；例如 cd /hom/test ，cd 加完整的路径，可以理解为绝对路径）

接下来继续学习更多的命令：

ls ./ 查看当前目录所有的文件和目录。

ls -a 查看所有的文件，包括隐藏文件,以.开头的文件。

pwd 显示当前所在的目录。

mkdir 创建目录，用法 mkdir test ，命令后接目录的名称。

rmdir 删除空目录

rm 删除文件或者目录，用法 rm –rf test.txt (-r 表示递归，-f 表示强制)。

cp 拷贝文件，用法,cp old.txt /tmp/new.txt ，常用来备份；如果拷贝目录

需要加 –r 参数。

mv 重命名或者移动文件或者目录，用法, mv old.txt new.txt

touch 创建文件，用法，touch test.txt，如果文件存在，则表示修改当前文件时间。

Useradd 创建用户，用法 useradd wugk ，userdel 删除用户。

Groupadd 创建组，用法 groupadd wugk1 ，groupdel 删除组。

find 查找文件或目录，用法 find /home -name “test.txt”,命令格式为:

find 后接查找的目录，-name 指定需要查找的文件名称，名称可以使用*表示所有。

find /home -name “*.txt” ;查找/home 目录下，所有以.txt 结尾的文件或者目录。

vi 修改某个文件，vi 有三种模式：

命令行模式、文本输入模式、末行模式。

默认 vi 打开一个文件，首先是命令行模式，然后按 i 进入文本输入模式，可以在文件里写入字符等等信息。

写完后，按 esc 进入命令模式，然后输入:进入末行模式，例如输入:wq 表示保存退出。

如果想直接退出，不保存，可以执行:q!， q!叹号表示强制退出。

cat 查看文件内容，用法 cat test.txt 可以看到 test.txt 内容

more 查看文件内容，分页查看，cat 是全部查看，如果篇幅很多，只能看到最后的篇幅。可以使用 cat 和 more 同时使用,例如： cat

test.txt |more 分页显示 text 内容，|符号是管道符，用于把|前的输出作为后面命令的输入。

echo 回显，用法 echo ok，会显示 ok，输入什么就打印什么。

echo ok > test.txt ；把 ok 字符覆盖 test.txt 内容，>表示追加并覆盖的意思。

>>两个大于符号，表示追加，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。

初学者常见的命令就如上所示，当然还有很多深入的命令需要学习，后面的课程会讲解。

2.1. 3 Linux 用户权限

在 Linux 操作系统中，root 的权限是最高的，相当于 windows 的 administrator，拥有最高权限，能执行任何命令和操作。在系统中，通过 UID 来区分用户的权限级别，UID 等于 0，表示此用户具有最高权限，也就是管理员。其他的用户 UID 依次增加，通过/etc/passwd 用户密码文件可以查看到每个用户的独立的 UID。

每一个文件或者目录的权限，都包含一个用户权限、一个组的权限、其他人权限，例如下：

标红第一个 root 表示该文件所有者是 root 用户，第二个 root 代表该文件的所属的组为 root 组，其他用户这里默认不标出。

```
[root@node1 ~]# ls -l monitor_log.sh
-rw-r--r-- 1 root root 91 May  7 20:21 monitor_log.sh
```

[root@node1 ~]#

如果我们想改变某个文件的所有者或者所属的组，可以使用命令 chown

chown –R test:test monitor_log.sh 即可。

每个 Linux 文件具有四种访问权限：可读(r)、可写(w)、可执行(x)和无权限(-)。

利用 ls -l 命令可以看到某个文件或目录的权限，它以显示数据的第一个字段为

准。第一个字段由 10 个字符组成，如下：

[root@node1 ~]# ls -l monitor_log.sh

-rw-r--r-- 1 root root 91 May 7 20:21 monitor_log.sh

[root@node1 ~]#

第一位表示文件类型，-表示文件，d 表示目录；后面每三位为一组。

第一组：2-4 位表示文件所有者的权限，即用户 user 权限，简称 u

第二组：5-7 位表示文件所有者所属组成员的权限，group 权限，简称 g

第三组：8-10 位表示所有者所属组之外的用户的权限，other 权限，简称 o

从上面这个文件，我们可以看出，monito_log.sh 文件对应的权限为：

root 用户具有读和写的权限，root 组具有读的权限，其他人具有读的权限。

为了能更简单快捷的使用和熟悉权限，rwx 权限可以用数字来表示，分别表示为 r（4）、w（2）、x（1）。

Monitor_log.sh 权限可以表示为：644

如果给某个文件授权，命令为 chmod：chmod 777 monitor_log.sh

2.1. 4 Linux 网络配置

熟悉了常用的命令和 Linux 权限，那接下来如何让所在的 Linux 系统上网呢？管理 linux 服务器网络有哪些命令呢？

Linux 服务器默认网卡配置文件在/etc/sysconfig/network-scripts/下，命名的名称一般为:ifcfg-eth0 ifcfg-eth1 ，eth0 表示第一块网卡，eth1 表示第二块网卡，依次类推。一般 DELL R720 标配有 4 块千兆网卡。

修改网卡的 IP，可以使用命令：vi /etc/sysconfig/network-scripts/ifcfg-eth0 如果是 DHCP 获取的 IP，默认配置如下：

```
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:0c:29:52:c7:4e
ONBOOT=yes
TYPE=Ethernet
```

如果是静态配置的 IP，ifcfg-eth0 网卡配置内容如下：

```
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
```

DEVICE=eth0

BOOTPROTO=static

HWADDR=00:0c:29:52:c7:4e

ONBOOT=yes

TYPE=Ethernet

IPADDR=192.168.149.128

NETMASK=255.255.255.0

GATEWAY=192.168.149.1

网卡参数详解如下：

DEVICE=eth0　　#物理设备名

ONBOOT=yes　　# [yes|no]（重启网卡是否激活设备）

BOOTPROTO=static #[none|static|bootp|dhcp]（不使用协议|静态分配|BOOTP 协议|DHCP 协议）

TYPE=Ethernet　#网卡类型

IPADDR=192.168.149.128 #IP 地址

NETMASK=255.255.255.0 #子网掩码

GATEWAY=192.168.149.1 #网关地址

网卡配置完毕，重启网卡，命令: /etc/init.d/network restart 即可。

查看 ip 命令：ifconfig 查看当前服务器所有网卡的 IP，可以单独指定，ifconfig eth0 查看 eth0 的 IP 地址。

网卡配置完毕，如果来配置 DNS，首先要知道 DNS 配置在哪个目录文件下，vi /etc/resolv.conf 文件:

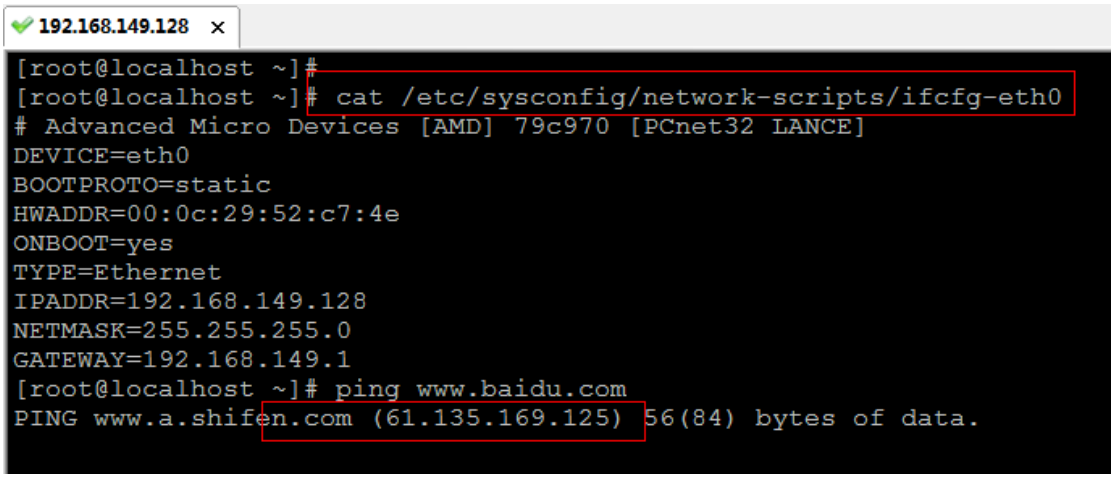
在该文件里面添加如下两条：

nameserver 202.106.0.20

nameserver 8.8.8.8

从上到下，分别表示主 DNS，备 DNS。配置完毕后，不需要重启网卡,DNS 立即生效。

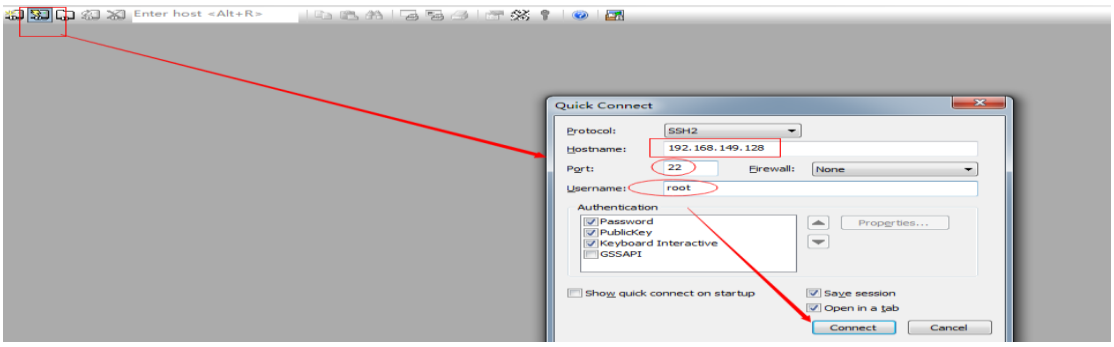
可以 ping www.baidu.com 看看效果:



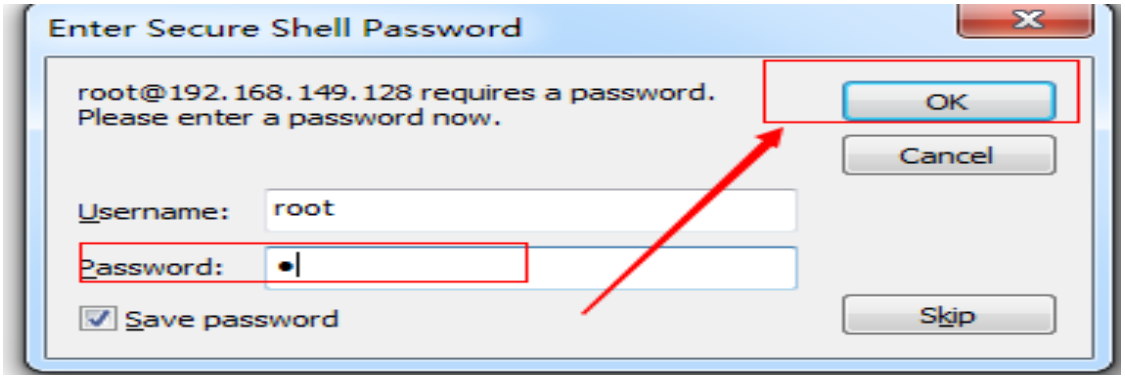
IP 配置完毕后，我们可以通过远程工具来连接 Linux 服务器，常见的 Linux 远程连接工具有:putty、secureCRT（主流）、xshell、xmanger 等工具。

下载安装 secureCRT，打开工具，然后如图配置：

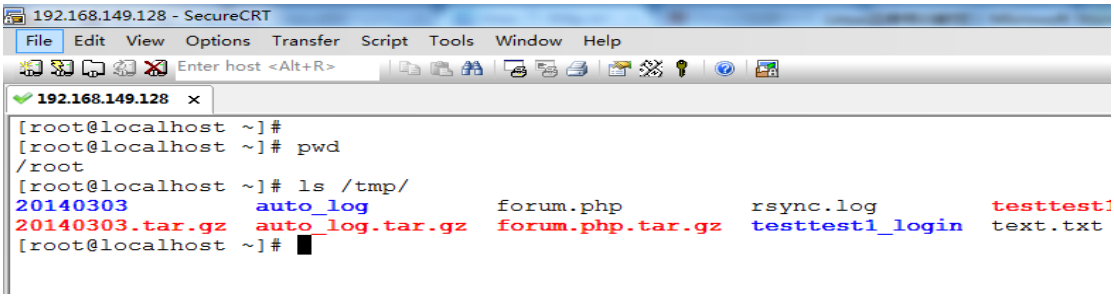
点击左上角 quick connect 快速连接，弹出界面，然后输入 IP，用户名，端口默认是 22，然后点击下方的 connect 连接，会提示输入密码，输入即可。



弹出输入密码框：



进入远程界面，与服务器真实登录一样，然后可以执行命令：



通过这几章的学习，我们已经熟练了 Linux 常用命令的操作，权限网络、网络配置、远程连接等知识，那接下来我们还能做什么呢？我们已经差不多入门了，接下来就是更进一步的服务配置，Linux 系统到底用来做什么呢？接下来的章节将跟大家一起来学习。

Linux 系统的应用，我们最开始介绍的时候简单介绍过，目前大中型企业都用它来承载 web 网站、数据库、虚拟化平台等，那接下来我们将在 Linux 系统安装各种服务和软件来实现 Linux 真正的价值。

3. Linux 服务篇

3.1 Linux 服务部署

3.1. 1 构建 NTP 时间服务器

NTP 服务器是用于局域网服务器时间同步使用的，可以保证局域网所有的服务器与时间服务器的时间保持一致，某些应用对时间实时性要求高的必须统一时间。

互联网的时间服务器也有很多，例如 ntpdate ntp.fudan.edu.cn 复旦大学的 NTP 免费提供互联网时间同步。

NTP 服务器监听端口为 UDP 的 123，那就需要在本地防火墙开启运行客户端访问 123 端口，vi /etc/sysconfig/iptables 添加如下规则：

-A INPUT -m state --state NEW -m udp -p udp --dport 123 -j ACCEPT

NTP 时间服务器配置：

yum install ntp ntpdate -y 即可！

修改 ntp.conf 配置文件

cp /etp/ntp.conf /etc/ntp.conf.bak

vi /etc/ntp.conf 只修改如下两行，把#号去掉即可！

server 127.127.1.0 # local clock

fudge 127.127.1.0 stratum 10

以守护进程启动 ntpd

/etc/init.d/ntpd start 即可

（注意*： ntpd 启动后，客户机要等几分钟再与其进行时间同步，否

则会提示“no server suitable for synchronization found”错误。）

配置时间同步客户机

crontab -e

增加一行，在每天的 6 点 10 分与时间同步服务器进行同步

10 06 * * * /usr/sbin/ntpdate ntp-server 的 ip >>/usr/local/logs/crontab/ntpdate.log

备注：如果客户机没有 ntpdate，可以 yum －y install ntp 即可！

以下是 ntp 服务器配置文件内容(局域网 NTP，如果需要跟外网同步，添加外网 server 即可)

driftfile /var/lib/ntp/drift

restrict default kod nomodify notrap nopeer noquery

restrict -6 default kod nomodify notrap nopeer noquery

restrict 127.0.0.1

restrict -6 ::1

server 127.127.1.0 # local clock

fudge 127.127.1.0 stratum 10

includefile /etc/ntp/crypto/pw

keys /etc/ntp/keys

下面是参数详解：

restrict default ignore	# 关闭所有的 NTP 要求封包
restrict 127.0.0.1	# 开启内部递归网络接口 lo
restrict 192.168.0.0 mask	#在内部子网里面的客户端可以进行

255.255.255.0 nomodify	网络校时，但不能修改 NTP 服务器的时间参数。
server 198.123.30.132	#198.123.30.132 作为上级时间服务器参考
restrict 198.123.30.132	#开放 server 访问我们 ntp 服务的权限
driftfile /var/lib/ntp/drift	在与上级时间服务器联系时所花费的时间，记录在 driftfile 参数后面的文件内
broadcastdelay 0.008	#广播延迟时间

自此 NTP 服务搭建完毕，然后在所有客户端 crontab 里面添加如下语句：

0 0 * * * /usr/sbin/ntpdate 10.0.0.155 >>/data/logs/ntp.log 2>&1

3.1. 2 构建 DHCP 服务器

DHCP(Dynamic Host Configuration Protocol，动态主机配置协议)是一个局域网的网络协议，使用 UDP 协议工作，主要用途：给内部网络或网络服务供应商自动分配 IP 地址，DHCP 有 3 个端口，其中 UDP67 和 UDP68 为正常的 DHCP 服务端口，分别作为 DHCP Server 和 DHCP Client 的服务端口。

DHCP 可以部署在服务器、交换机或者服务器，可以控制一段 IP 地址范围，客户机登录服务器时就可以自动获得 DHCP 服务器分配的 IP 地址和子网掩码。其中 DHCP 所在服务器的需要安装 TCP/IP 协议，需要设置静态 IP 地址、子网掩码、默认网关。

正式安装 DHCP 服务：

Yum install dhcp dhcp-devel –y 即可，然后修改 DHCP /etc/dhcpd.conf 配置文件内容如下：

```
ddns-update-style interim;
ignore client-updates;
next-server   192.168.0.79;
filename "pxelinux.0";
allow booting;
allow bootp;
subnet 192.168.0.0 netmask 255.255.255.0 {
# --- default gateway
option routers         192.168.0.1;
option subnet-mask     255.255.252.0;
#    option nis-domain      "domain.org";
#    option domain-name "192.168.0.10";
#    option domain-name-servers  192.168.0.11;
#    option ntp-servers     192.168.1.1;
#    option netbios-name-servers   192.168.1.1;
```

```
# --- Selects point-to-point node (default is hybrid). Don't change this unless
# -- you understand Netbios very well
#   option netbios-node-type 2;
range   dynamic-bootp   192.168.0.100 192.168.0.200;
host ns {
hardware ethernet  00:1a:a0:2b:38:81;
fixed-address 192.168.0.101;}
}
```

参数解析如下：

选　项	解　释
ddns-update-style interim|ad-hoc|none	参数用来设置 DHCP 服务器与 DNS 服务器的动态信息更新模式：interim 为 DNS 互动更新模式，ad-hoc 为特殊 DNS 更新模式，none 为不支持动态更新模式。
next-server ip	pxeclient 远程安装系统，指定 tftp server 地址
filename	开始启动文件的名称，应用于无盘安装，可以是 tftp 的相对或绝对路径
ignore client-updates	为忽略客户端更新

subnet-mask	为客户端设定子网掩码
option routers	为客户端指定网关地址
domain-name	为客户端指明 DNS 名字
domain-name-servers	为客户端指明 DNS 服务器的 IP 地址
host-name	为客户端指定主机名称
broadcast-address	为客户端设定广播地址
ntp-server	为客户端设定网络时间服务器的 IP 地址
time-offset	为客户端设定格林威治时间的偏移时间，单位是秒

注意如上配置，需要修改成对应服务器网段 IP，然后重启 DHCP 服务，/etc/init.d/dhcpd restart 即可。

客户端要从这个 DHCP 服务器获取 IP，需要做简单的设置，如果是 linux 需要把/etc/sysconfig/network-scritps/ifcfg-eth0 里 BOOTPROTO 相改成 dhcp 即可，windows 机器的话，需要修改本地连接，把它设置成自动获取 IP 即可。

BOOTPROTO=dhcp

3.1. 3 搭建 Samba 服务器

Samba 是在 Linux 和 UNIX 系统上实现 SMB 协议的一个免费软件，由服务器及客户端程序构成，

SMB（Server Messages Block，信息服务块）是一种在局域网上共享文件和打印机的一种通信协议，它为局域网内的不同计算机之间提供文件及打印机等资源的共享服务。

SMB 协议是客户机/服务器型协议，客户机通过该协议可以访问服务器上的共享文件系统、打印机及其他资源。通过设置“NetBIOS over TCP/IP”使得 Samba 不但能与局域网络主机分享资源，还能与全世界的电脑分享资源。

安装 SAMBA 服务器：

Yum install samba –y

安装完毕，然后做如下设置（过滤#号行、空行如下命令）

cp /etc/samba/smb.conf /etc/samba/smb.conf.bak ;egrep -v "#|^$" /etc/samba/smb.conf.bak |grep -v "^;" >/etc/samba/smb.conf

查看 smb.conf 配置文件如下：

```
[global]
        workgroup = MYGROUP
        server string = Samba Server Version %v
        security = share
        passdb backend = tdbsam
        load printers = yes
        cups options = raw

[temp]
```

comment=Temporary file space

path=/tmp

read only=no

public=yes

[data]

comment=Temporary file space

path=/data

read only=no

public=yes

根据需求修改之后重启服务：

[root@node1 ~]# /etc/init.d/smb restart

Shutting down SMB services:

[FAILED]

Shutting down NMB services:

[FAILED]

Starting SMB services:

[OK]

Starting NMB services:

[OK]

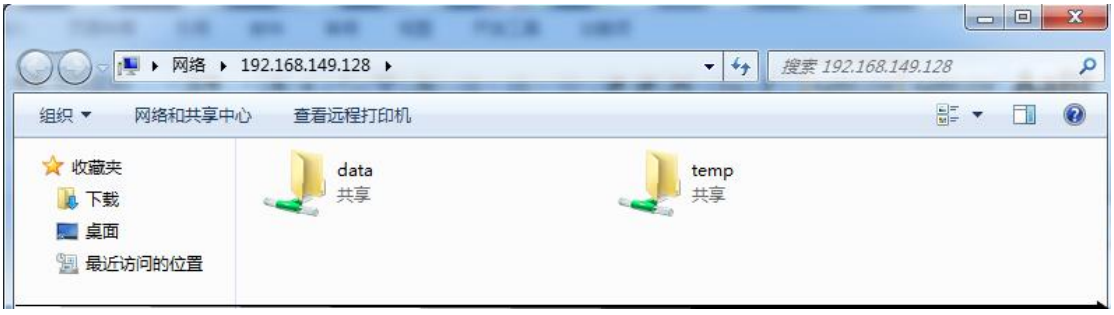
workgroup =	WORKGROUP 设 Samba Server 所要加入的工作

	组或者域。
server string = Samba Server Version %v	Samba Server 的注释，可以是任何字符串，也可以不填。宏%v 表示显示 Samba 的版本号。
security = user	1.share：用户访问 Samba Server 不需要提供用户名和口令，安全性能较低。 2. user：Samba Server 共享目录只能被授权的用户访问,由 Samba Server 负责检查账号和密码的正确性。账号和密码要在本 Samba Server 中建立。 3. server：依靠其他 Windows NT/2000 或 Samba Server 来验证用户的账号和密码,是一种代理验证。此种安全模式下,系统管理员可以把所有的 Windows 用户和口令集中到一个 NT 系统上,使用 Windows NT 进行 Samba 认证，远程服务器可以自动认证全部用户和口令,如果认证失败,Samba 将使用用户级安全模式作为替代的方式。 4. domain：域安全级别,使用主域控制器(PDC)来完成认证。
comment = test	是对该共享的描述，可以是任意字符串。
path = /home/test	共享目录路径

browseable= yes/no	用来指定该共享是否可以浏览。
writable = yes/no	writable 用来指定该共享路径是否可写。
available = yes/no	available 用来指定该共享资源是否可用
admin users = admin	该共享的管理者
valid users = test	允许访问该共享的用户
invalid users = test	禁止访问该共享的用户
write list = test	允许写入该共享的用户
public = yes/no	public 用来指定该共享是否允许 guest 账户访问。

在浏览器里面访问方式为：\\192.168.149.128 （SMB 文件共享服务端 IP），如何没有权限访问，需要注意防火墙和 selinux 设置，可以使用如下命令关闭：

/etc/init.d/iptables stop ;sed –i '/SELINUX/s/enforcing/disabled' /etc/sysconfig/selinux



3.1. 4 搭建 NFS 服务器

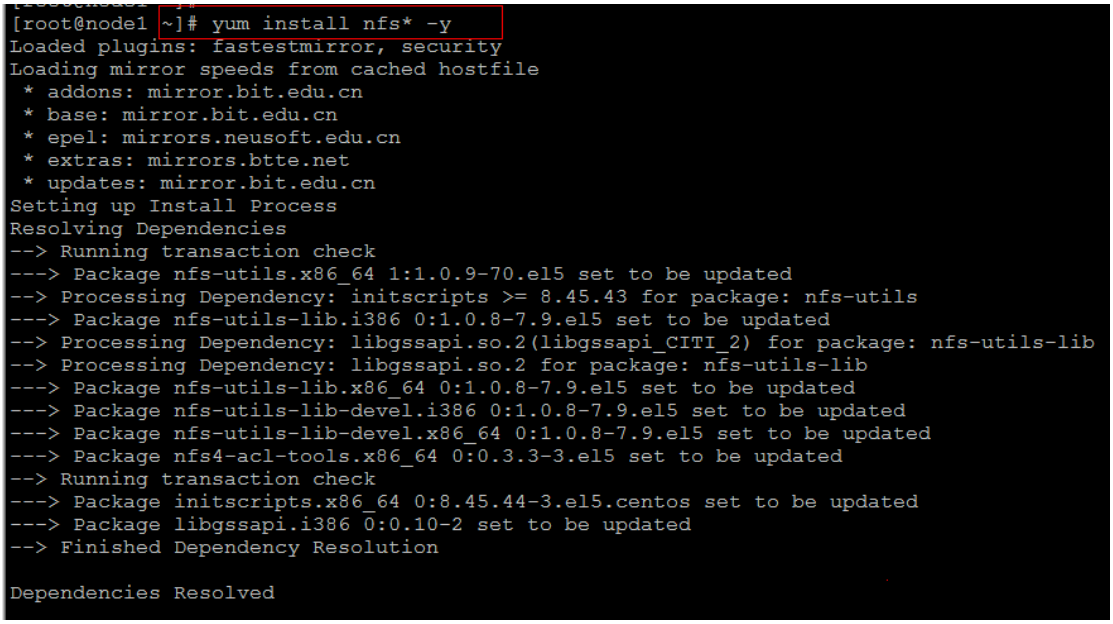
NFS 是 Network File System 的缩写，即网络文件系统。一种使用于分散式文件系统的协定，由 Sun 公司开发，于 1984 年向外公布。功能是通过网络让不同的机器、不同的操作系统能够彼此分享个别的数据，让应用程序在客户端通过网络访问位于服务器磁盘中的数据，是在类 Unix 系统间实现磁盘文件共享的一种方法。

NFS 在文件传送或信息传送过程中依赖于 RPC 协议。RPC，远程过程调用 (Remote Procedure Call) 是能使客户端执行其他系统中程序的一种机制。NFS 本身是没有提供信息传输的协议和功能的。

NFS 应用场景，常用于高可用文件共享，多台服务器共享同样的数据，可扩展性比较差，本身高可用方案不完善，取而代之的数据量比较大的可以采用 MFS、TFS、HDFS 等等分布式文件系统。

NFS 安装配置：

Yum install nfs* portmap -y 如下图，安装成功即可。



NFS 安装完毕，需要创建共享目录，共享目录在/etc/exports 文件里面配置，可配置参数如下：

/data/ 192.168.149.129(rw,sync,no_hide,no_all_squash)

在配置文件中添加如上一行，然后重启 Portmap，NFS 服务即可，/etc/init.d/portmap restart ;/etc/init.d/nfs restart

第一列/data/表示需要共享的目录。

IP 表示允许哪个客户端访问。

IP 后括号里的设置表示对该共享文件的权限。

ro 只读访问

rw 读写访问

sync 所有数据在请求时写入共享

hide 在 NFS 共享目录中不共享其子目录

no_hide 共享 NFS 目录的子目录

all_squash 共享文件的 UID 和 GID 映射匿名用户 anonymous，适合公用目录。

no_all_squash　　保留共享文件的 UID 和 GID（默认）

root_squash　　root 用户的所有请求映射成如 anonymous 用户一样的权限（默认）

no_root_squas　　root 用户具有根目录的完全管理访问权限

Linux 客户端，如何想使用这个 NFS 文件系统，需要在客户端挂载，挂载命令为：

Mount –t nfs 192.168.149.128:/data/ /mnt 即可。如果有报错根据错误信息排查。常见问题有 rpc 服务没有启动、防火墙没关闭、selinux 未关闭等问题。（拓展* 有兴趣的童鞋可以研究 MFS（分布式文件系统）。）

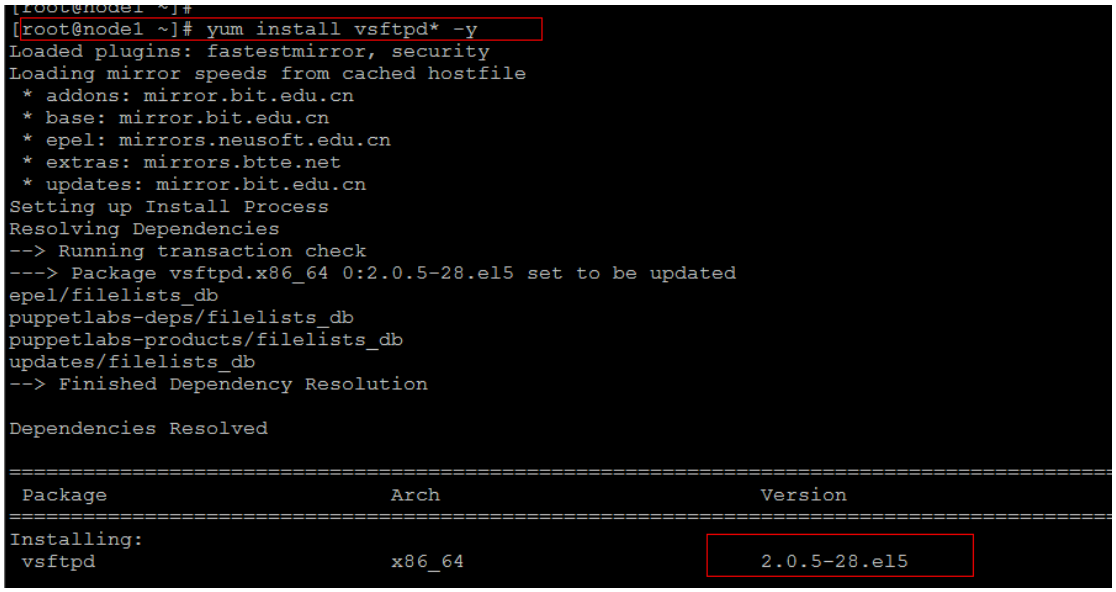
3.1. 5 搭建 FTP 服务器

FTP 是文件传输协议，正是由于这种协议使得主机间可以共享文件。 FTP 使用 TCP 生成一个虚拟连接用于控制信息，然后再生成一个单独的 TCP 连接用于数据传输。

vsftpd 是一款在 Linux 发行版中最主流的 FTP 服务器程序；特点是小巧轻快，安全易用；能让其自身特点得发发挥和掌握。

目前在开源操作系统中常用的 FTP 服务器程序主要有 vsftpd、ProFTPD、PureFTPd 和 wuftpd 等，这么多 FTP 服务器程序，关键在于自己熟练哪一个就使用哪一个。今天我们来研究一下 VSFTPD

简单安装及使用。安装命令: yum　install vsftpd*　-y



修改配置文件如下：

#vsftpd config 2014 by wugk

anonymous_enable=NO　　//禁止匿名用户访问

local_enable=YES　//允许本地用户登录 FTP

write_enable=YES　　//运行用户在 FTP 目录有写入的权限

local_umask=022　　//设置本地用户的文件生成掩码为 022，默认是 077

dirmessage_enable=YES　//激活目录信息,当远程用户更改目录时,将出现提示信息

xferlog_enable=YES　　//启用上传和下载日志功能

connect_from_port_20=YES　//启用 FTP 数据端口的连接请求

xferlog_std_format=YES　//是否使用标准的 ftpd xferlog 日志文件格式

listen=YES　//使 vsftpd 处于独立启动监听端口模式

pam_service_name=vsftpd //设置 PAM 认证服务配置文件名称，文件存放在/etc/pam.d/目录

userlist_enable=YES　　//用户列表中的用户是否允许登录 FTP 服务器,默认是不允许

tcp_wrappers=YES　　//使用 tcp_wrqppers 作为主机访问控制方式

1) 第一种方法就是使用系统用户登录 FTP，但是也是比较危险的，先测试系统用户登录 FTP，在 Linux 系统上创建 useradd test 用户，并为其设置名，然后在 xp 客户端打开我的电脑资源里面访问 ftp://192.168.149.128，输入用户名和密码即可访问，进行创建和删除操作。
2) 第二种方法比较安全，配置相对复杂一点，就是使用 vsftpd 虚拟用户登录 FTP 服务器进行常见的操作。
   - 首先安装 FTP 虚拟用户需要用到的软件及认证模块

   yum install pam* db4* --skip-broken –y

   创建并生成 vsftpd 数据库文件 vi /etc/vsftpd/ftpusers.txt，内容如下：

   第一行为 FTP 虚拟用户，登录用户名，第二行为密码，第三行为用户名，依次类推。

   wugk

   1

   wugk1

1

- 生成数据库文件命令：

db_load -T -t hash -f /etc/vsftpd/ftpusers.txt /etc/vsftpd/vsftpd_login.db

chmod 700 /etc/vsftpd/vsftpd_login.db

- 配置 PAM 验证文件：

在配置文件 vi /etc/pam.d/vsftpd 行首加入如下两行认证语句:（如果是 32 位，lib64 需改成 lib，如果 RedHat，加入的语句不一样，需注意）

auth sufficient /lib64/security/pam_userdb.so db=/etc/vsftpd/vsftpd_login

account sufficient /lib64/security/pam_userdb.so db=/etc/vsftpd/vsftpd_login

- 创建 vsftpd 映射本地用户:

所有的 FTP 虚拟用户需要使用一个系统用户，这个系统用户不需要密码，也不需要登录。主要用来做虚拟用户映射使用。

useradd –d /home/ftpuser –s /sbin/nologin ftpuser

- 修改完整版配置文件内容如下：

anonymous_enable=NO

local_enable=YES

write_enable=YES

local_umask=022

dirmessage_enable=YES

xferlog_enable=YES

connect_from_port_20=YES

xferlog_file=/var/log/vsftpd.log

xferlog_std_format=YES

ascii_upload_enable=YES

ascii_download_enable=YES

listen=YES

guest_enable=YES

guest_username=ftpuser

pam_service_name=vsftpd

user_config_dir=/etc/vsftpd/vsftpd_user_conf

virtual_use_local_privs=YES

保存重启,/etc/init.d/vsftpd restart 即可使用虚拟用户登录，这时候所有的虚拟用户共同使用/home/ftpuser 目录上传下载，如果想使用自己独立的目录，可以在/etc/vsftpd/vsftpd_user_conf 目录创建各自的配置文件，如给 wugk 创建独立的配置文件：

vi /etc/vsftpd/vsftpd_user_conf/wugk ，内容如下，建立自己的 FTP 目录。

local_root=/home/ftpsite/wugk

write_enable=YES

anon_world_readable_only=YES

anon_upload_enable=YES

anon_mkdir_write_enable=YES

anon_other_write_enable=YES

重启，使用客户端登录 FTP，测试即可。关于 FTP 讲解就到此，windows 还可以使用 Server-U 来搭建 FTP 服务器端，有兴趣的童鞋可以研究一下。

- FTP 主被动模式

FTP 主动模式：客户端从一个任意的非特权端口 N（N>1024）连接到 FTP 服务器的 port 21 命令端口。然后客户端开始监听端口 N+1，并发送 FTP 命令“port N+1”到 FTP 服务器。接着服务器会从它自己的数据端口（20）连接到客户端指定的数据端口（N+1）。

FTP 被动模式：客户端从一个任意的非特权端口 N（N>1024）连接到 FTP 服务器的 port 21 命令端口。然后客户端开始监听端口 N+1，同时客户端提交 PASV 命令。服务器会开启一个任意的非特权端口（P >1024），并发送 PORT P 命令给客户端。然后客户端发起从本地端口 N+1 到服务器的端口 P 的连接用来传送数据。

3.1. 6 构建 Apache WEB 服务器

Apache 是世界使用排名第一的 Web 服务器软件。它可以运行在几乎所有广泛使用的计算机平台上，由于其跨平台和安全性被广泛使用，

是最流行的 Web 服务器端软件之一。Apache 工作模式有多种,其中最常用的有两种：

Prefork 模式：Prefork MPM 使用多个子进程，每个子进程只有一个线程。每个进程在某个确定的时间只能维持一个连接。

在大多数平台上，Prefork MPM 在效率上要比 Worker MPM 要高，但是内存使用大得多。prefork 的无线程设计在某些情况下将比 worker 更有优势：它可以使用那些没有处理好线程安全的第三方模块，并且对于那些线程调试困难的平台而言，它也更容易调试一些。

Worker 模式：Worker MPM 使用多个子进程，每个子进程有多个线程。每个线程在某个确定的时间只能维持一个连接。通常来说，在一个高流量的 HTTP 服务器上，Worker MPM 是个比较好的选择，因为 Worker MPM 的内存使用比 Prefork MPM 要低得多。

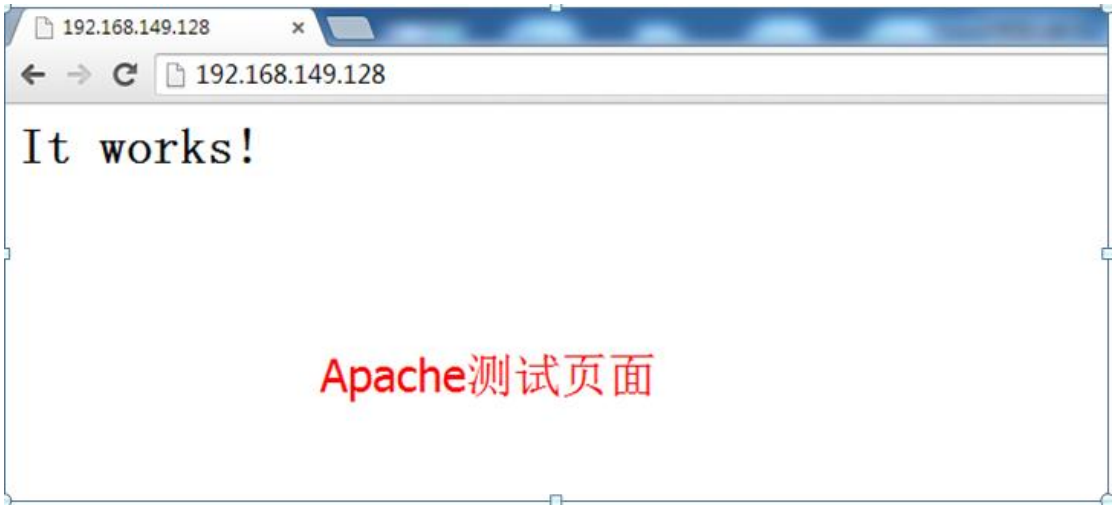
Worker MPM 也由不完善的地方，如果一个线程崩溃，整个进程就会连同其所有线程一起"死掉".由于线程共享内存空间，所以一个程序在运行时必须被系统识别为"每个线程都是安全的"。

- 源码安装 Apache

  官方下载目前稳定版本，http://mirror.esocc.com/apache/httpd/httpd-2.2.27.tar.gz ，解压安装如下，安装 apache 之前，需要先安装 apr apr-util。

```
[root@node1 ~]# cd soft/
[root@node1 soft]#
[root@node1 soft]# tar xzf httpd-2.2.27.tar.gz
[root@node1 soft]#
[root@node1 soft]# ls
httpd-2.2.27  httpd-2.2.27.tar.gz  zabbix
[root@node1 soft]# cd httpd-2.2.27
[root@node1 httpd-2.2.27]# ls
ABOUT_APACHE  buildconf      config.status  httpd.dep    InstallBin.dsp  Makefile      modules.o
acinclude.m4  buildmark.o    configure      httpd.dsp    LAYOUT          Makefile.in   NOTICE
Apache.dsw    CHANGES        configure.in   httpd.mak    libhttpd.dep    Makefile.win  NWGNUmakefile
build         config.layout  docs           httpd.spec   libhttpd.dsp    modules       os
BuildAll.dsp  config.log     emacs-style    include      libhttpd.mak    modules.c     README
BuildBin.dsp  config.nice    httpd          INSTALL      LICENSE         modules.lo    README.platforms
[root@node1 httpd-2.2.27]# yum install apr apr-util apr-devel apr-util-devel
[root@node1 httpd-2.2.27]#
[root@node1 httpd-2.2.27]#
[root@node1 httpd-2.2.27]# ./configure --prefix=/usr/local/apache2 --enable-rewrite --enable-so
[root@node1 httpd-2.2.27]#
[root@node1 httpd-2.2.27]#
[root@node1 httpd-2.2.27]# make
[root@node1 httpd-2.2.27]#
[root@node1 httpd-2.2.27]#
[root@node1 httpd-2.2.27]# make install
[root@node1 httpd-2.2.27]#
[root@node1 httpd-2.2.27]#
```

然后启动 apache 服务: /usr/local/apache2/bin/apachectl start



查看 apache 进程及端口:

```
[root@node1 ~]#
[root@node1 ~]# ps -ef |grep httpd
root      1585     1  0 15:32 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
daemon    1586  1585  0 15:32 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
daemon    1587  1585  0 15:32 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
daemon    1588  1585  0 15:32 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
daemon    1589  1585  0 15:32 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
daemon    1590  1585  0 15:32 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
daemon    1594  1585  0 15:32 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
root      1614  5801  0 15:35 pts/0    00:00:00 grep httpd
[root@node1 ~]#
[root@node1 ~]#
[root@node1 ~]# netstat -tnl |grep 80
tcp        0      0 :::80                       :::*                        LISTEN
[root@node1 ~]#
[root@node1 ~]# lsof -i :80
COMMAND   PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
httpd    1585   root    3u  IPv6  43197      0t0  TCP *:http (LISTEN)
httpd    1586 daemon    3u  IPv6  43197      0t0  TCP *:http (LISTEN)
httpd    1587 daemon    3u  IPv6  43197      0t0  TCP *:http (LISTEN)
httpd    1588 daemon    3u  IPv6  43197      0t0  TCP *:http (LISTEN)
httpd    1589 daemon    3u  IPv6  43197      0t0  TCP *:http (LISTEN)
httpd    1590 daemon    3u  IPv6  43197      0t0  TCP *:http (LISTEN)
httpd    1594 daemon    3u  IPv6  43197      0t0  TCP *:http (LISTEN)
[root@node1 ~]#
```

源码包安装 Apache 默认发布目录为:/usr/local/apache2/htdocs/下。

- Apache 基于域名虚拟主机配置

修改 vi /usr/local/apache2/conf/extra/httpd-vhosts.conf 虚拟主机配置文件内容如下：

```
NameVirtualHost *:80
<VirtualHost *:80>
    ServerAdmin wgkgood@163.com
    DocumentRoot "/data/webapps/www1"
    ServerName www.wugk1.com
  <Directory "/data/webapps/www1">
    AllowOverride All
    Options -Indexes FollowSymLinks
    Order allow,deny
    Allow from all
  </Directory>
    ErrorLog    logs/error_log
    CustomLog logs/access_log common
</VirtualHost>

<VirtualHost *:80>
    ServerAdmin wgkgood@163.com
    DocumentRoot "/data/webapps/www2"
```

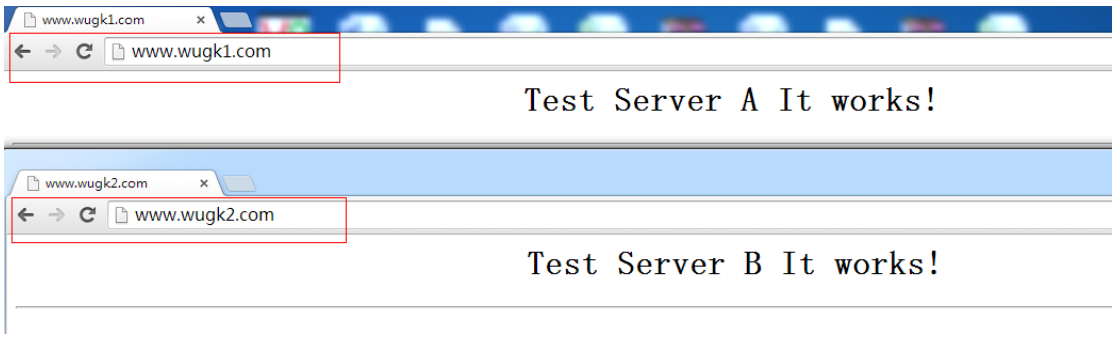
```
    ServerName www.wugk2.com
  <Directory "/data/webapps/www2">
    AllowOverride All
    Options -Indexes FollowSymLinks
    Order allow,deny
    Allow from all
  </Directory>
    ErrorLog    logs/error_log
    CustomLog logs/access_log common
</VirtualHost>
```

然后在/usr/local/apache2/conf/httpd.conf 最末尾加入如下配置：

Include conf/extra/httpd-vhosts.conf

重新加载 apache 即可，/usr/local/apache2/bin/apachectl graceful

效果演示如下：



Apache 基于 IP 虚拟主机同样跟域名一直，在服务器配置多个 IP，然后把域名改成 IP 即可。

- Apache Rewrite 规则讲解

Rewrite URL 重定向就是实现 URL 的跳转和隐藏真实地址，可以把

复杂的 URL 变成简洁直观的 URL，对 seo 优化有很大的帮助。如下几个简单的举例：

把所有配置的域名都跳转到一个域名：

RewriteEngine on　　//启用 rewrite 引擎

RewriteCond　%{HTTP_HOST}　^wugk1.com　[NC]　　// 匹 配 以 wugk1.com 开头的域名,NC 忽略大小写。

RewriteRule ^/(.*)$　http://www.wugk1.com/$1　[L]

//匹配上面条件，然后跳转到 http://www.wugk1.com

1) R 强制外部重定向。
2) F 禁用 URL,返回 403HTTP 状态码。
3) G 强制 URL 为 GONE，返回 410HTTP 状态码。
4) P 强制使用代理转发。
5) L 表明当前规则是最后一条规则，停止分析以后规则的重写。
6) N 重新从第一条规则开始运行重写过程。
7) C 与下一条规则关联。

3.1. 7 构建 MySQL 服务器

MySQL 是一个开放源码的小型关联式数据库管理系统，开发者为瑞典 MySQL AB 公司，目前属于 Oracle 公司,MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。

对应目前主流的 LAMP 架构来说,Mysql 更是得到各位 IT 运维、DBA 的青睐，目前 mysql 已被 orcacle 收购，不过好消息是原来 mysql 创始人已独立出来自己重新开发了一个 MariaDB，而且使用的人数越来越多。而且 MariaDB 兼容 mysql 所有的功能和相关参数。

Mysql 常用的两大引擎有 MyISAM 和 innoDB，那他们有什么明显的区别呢，什么场合使用什么引擎呢？

MyISAM 类型的表强调的是性能，其执行数 度比 InnoDB 类型更快，但不提供事务支持，如果执行大量的 SELECT 操作，MyISAM 是更好的选择，支持表锁。

InnoDB 提供事务支持事务，外部键等高级 数据库功能，执行大量的 INSERT 或 UPDATE，出于性能方面的考虑，应该使用 InnoDB 表，支持行锁。

MySQL 安装方式有两种，一种是 yum/rpm 安装，另外一种是 tar 源码安装。

Yum 安装方法很简单，执行命令如下即可：yum install –y mysql-server mysql-devel mysql

源码安装 MySQL 方式：

cd /usr/src ;

wget http://downloads.mysql.com/archives/mysql-5.1/mysql-5.1.63.tar.gz ;tar xzf mysql-5.1.63.tar.gz ;cd mysql-5.1.63 ;./configure --prefix=/usr/local/mysql --enable-assembler &&make &&make

install

```
make[3]: Nothing to be done for `install-data-am'.
make[3]: Leaving directory `/usr/src/mysql-5.1.63/server-tools'
make[2]: Leaving directory `/usr/src/mysql-5.1.63/server-tools'
Making install in instance-manager
make[2]: Entering directory `/usr/src/mysql-5.1.63/server-tools/instance-manager'
make[3]: Entering directory `/usr/src/mysql-5.1.63/server-tools/instance-manager'
test -z "/usr/local/mysql/libexec" || /bin/mkdir -p "/usr/local/mysql/libexec"
  /bin/sh ../../libtool --preserve-dup-deps   --mode=install /usr/bin/install -c 'mysqlmanager' '/usr/local/m
r'
libtool: install: /usr/bin/install -c mysqlmanager /usr/local/mysql/libexec/mysqlmanager
make[3]: Nothing to be done for `install-data-am'.
make[3]: Leaving directory `/usr/src/mysql-5.1.63/server-tools/instance-manager'
make[2]: Leaving directory `/usr/src/mysql-5.1.63/server-tools/instance-manager'
make[1]: Leaving directory `/usr/src/mysql-5.1.63/server-tools'
Making install in win
make[1]: Entering directory `/usr/src/mysql-5.1.63/win'
make[2]: Entering directory `/usr/src/mysql-5.1.63/win'
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/usr/src/mysql-5.1.63/win'
make[1]: Leaving directory `/usr/src/mysql-5.1.63/win'
[root@node2 mysql-5.1.63]#
[root@node2 mysql-5.1.63]# ./configure  --prefix=/usr/local/mysql --enable-assembler &&make &&make install
```

配置 Mysql 服务为系统服务：

cp /usr/local/mysql/share/mysql/my-medium.cnf /etc/my.cnf

cp /usr/local/mysql/share/mysql/mysql.server /etc/rc.d/init.d/mysqld

chkconfig --add mysqld

chkconfig --level 345 mysqld on

cd /usr/local/mysql

useradd mysql

chown -R mysql.mysql /usr/local/mysql

/usr/local/mysql/bin/mysql_install_db --user=mysql

chown -R mysql var

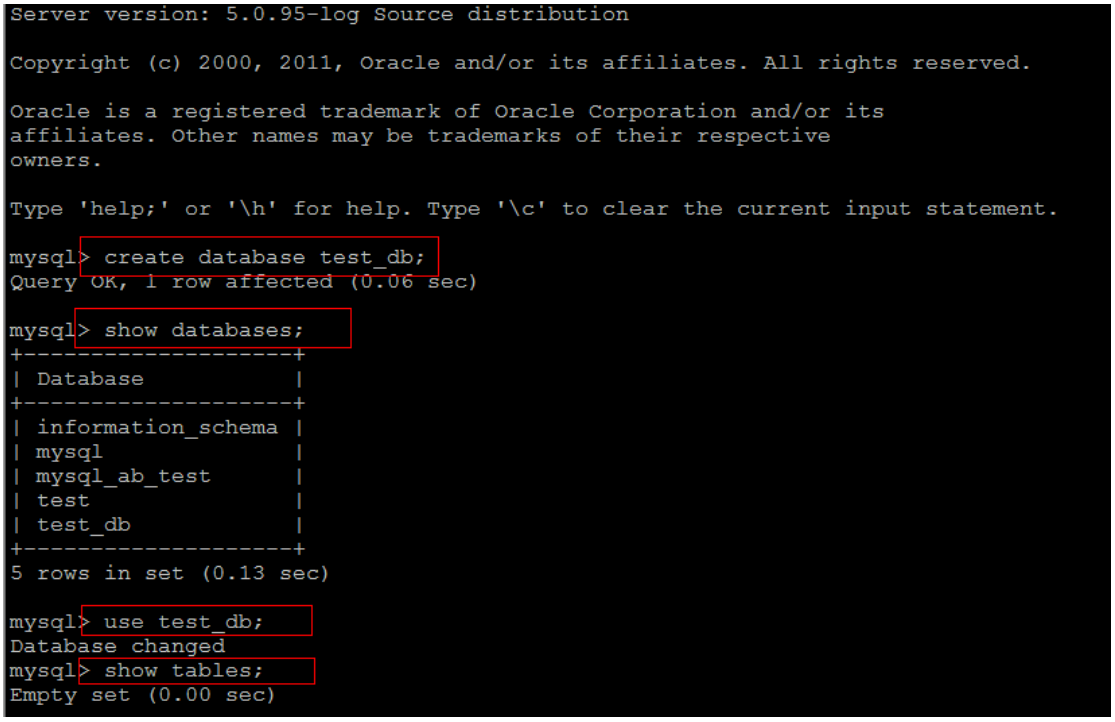
/usr/local/mysql/bin/mysqld_safe --user=mysql &

MySQL 日常操作命令：

create database test_db; 创建名为 test_db 数据库

use test_db; 进入 test_db 数据库

show tables; 查看数据库里有多少张表。



create table test01 (id varchar(20),name varchar(20));创建名为 test01 表，并创建两个字段，id、name、数据长度（用字符来定义长度单位。）

insert into test01 values ("001","wugk1"); 向表中插入数据。

select * from test01; 查看 test01 表数据内容。

```
mysql> create table test01 (id varchar(20),name varchar(20));
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+------------------+
| Tables_in_test_db |
+------------------+
| test01           |
+------------------+
1 row in set (0.00 sec)

mysql> insert into test01 values ("001","wugk1");
Query OK, 1 row affected (0.02 sec)

mysql> insert into test01 values ("002","wugk2");
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> select * from test01;
+------+-------+
| id   | name  |
+------+-------+
| 001  | wugk1 |
| 002  | wugk2 |
+------+-------+
2 rows in set (0.00 sec)
```

grant all privileges on test_db.* to test@localhost identified by '123456';

grant all on test_db.* to test@localhost identified by '123456';

grant select,insert,update,delete on *.* to test@"%" identified by '123456';

给 mysql 数据库授权。

flush privileges;刷新权限

mysqldump –uroot –p123456 test_db >/tmp/test.db.sql ;MySQL 备份或导出

mysql –uroot –p123456 test_db < /tmp/test.db.sql ;MySQL 导入

mysqladmin –uroot –p123456 password newpassword ;修改 MySQL root 密码

drop database test_db ; 删除数据库

drop table test01 ; 删除表

delete from test01 ; 清空表内容

show variables like '%char%'; 查看数据库字符集

修改 Mysql 字符集为 UTF-8 的方法：在/etc/my.cnf 对应如下配置段加入相应命令。

[client]字段里加入 default-character-set=utf8

[mysqld]字段里加入 character-set-server=utf8

[mysql]字段里加入 default-character-set=utf8

然后重启 MySQL 服务即可。

3.1. 8 LAMP 架构网站搭建

Linux+Apache+Mysql/MariaDB+Perl/PHP/Python 一组常用来搭建动态网站或者服务器的开源软件，本身都是各自独立的程序，但是因为常被放在一起使用，拥有了越来越高的兼容度，共同组成了一个强大的 Web 应用程序平台。

随着开源潮流的蓬勃发展，开放源代码的 LAMP 已经与 J2EE 和.Net 商业软件形成三足鼎立之势，并且该软件开发的项目在软件方面的投资成本较低，因此受到整个 IT 界的关注。

目前 LAMP 架构是大多数中小企业最青睐的 PHP 架构选择，也是

众多 Linux SA 喜欢选择的一套架构。那接下来我们就实战来操作一下，如果来搭建这样一套架构，当然可以使用 yum 方法，安装命令很简单，一条命令搞定所有。

yum install httpd httpd-devel mysql-server mysql-devel php php-devel php-mysql –y

这一条命令 LAMP 环境即可安装成功，只需要重启 apache、mysql 服务即可。

如果想要更多功能和自定义模块，需要使用源码包的方式来安装 LAMP 架构。如下我们使用源码包来实现 LAMP 架构安装与配置：

- 源码安装 LAMP 之 Apache

yum install apr-devel apr-util-devel –y；

cd /usr/src ; wget http://mirror.bit.edu.cn/apache/httpd/httpd-2.2.27.tar.gz ;tar xzf httpd-2.2.27.tar.gz ;cd httpd-2.2.27 ;./configure --prefix=/usr/local/apache --enable-so --enable-rewrite &&make &&make install

- 源码安装 LAMP 之 MySQL

cd /usr/src ;wget http://downloads.mysql.com/archives/mysql-5.1/mysql-5.1.63.tar.gz ;tar xzf mysql-5.1.63.tar.gz ;cd mysql-5.1.63 ;./configure --prefix=/usr/local/mysql --enable-assembler &&make &&make install

```
make[3]: Nothing to be done for `install-data-am'.
make[3]: Leaving directory `/usr/src/mysql-5.1.63/server-tools'
make[2]: Leaving directory `/usr/src/mysql-5.1.63/server-tools'
Making install in instance-manager
make[2]: Entering directory `/usr/src/mysql-5.1.63/server-tools/instance-manager'
make[3]: Entering directory `/usr/src/mysql-5.1.63/server-tools/instance-manager'
test -z "/usr/local/mysql/libexec" || /bin/mkdir -p "/usr/local/mysql/libexec"
  /bin/sh ../../libtool --preserve-dup-deps   --mode=install /usr/bin/install -c 'mysqlmanager' '/usr/local/m
r'
libtool: install: /usr/bin/install -c mysqlmanager /usr/local/mysql/libexec/mysqlmanager
make[3]: Nothing to be done for `install-data-am'.
make[3]: Leaving directory `/usr/src/mysql-5.1.63/server-tools/instance-manager'
make[2]: Leaving directory `/usr/src/mysql-5.1.63/server-tools/instance-manager'
make[1]: Leaving directory `/usr/src/mysql-5.1.63/server-tools'
Making install in win
make[1]: Entering directory `/usr/src/mysql-5.1.63/win'
make[2]: Entering directory `/usr/src/mysql-5.1.63/win'
make[2]: Nothing to be done for `install-exec-am'.
make[2]: Nothing to be done for `install-data-am'.
make[2]: Leaving directory `/usr/src/mysql-5.1.63/win'
make[1]: Leaving directory `/usr/src/mysql-5.1.63/win'
[root@node2 mysql-5.1.63]#
[root@node2 mysql-5.1.63]# ./configure  --prefix=/usr/local/mysql --enable-assembler &&make &&make install
```

配置 Mysql 服务为系统服务：

cp /usr/local/mysql/share/mysql/my-medium.cnf /etc/my.cnf

cp /usr/local/mysql/share/mysql/mysql.server /etc/rc.d/init.d/mysqld

chkconfig --add mysqld

chkconfig --level 345 mysqld on

cd /usr/local/mysql

useradd mysql

chown -R mysql.mysql /usr/local/mysql

/usr/local/mysql/bin/mysql_install_db --user=mysql

chown -R mysql var

/usr/local/mysql/bin/mysqld_safe --user=mysql &

- 源码安装 LAMP 之 PHP

cd /usr/src ;wget http://mirrors.sohu.com/php/php-5.3.28.tar.bz2 ;tar jxf

php-5.3.28.tar.bz2 ;cd php-5.3.28 ;./configure --prefix=/usr/local/php5

--with-config-file-path=/usr/local/php/etc

--with-apxs2=/usr/local/apache/bin/apxs --with-mysql=/usr/local/mysql/

```
Installing PHP CLI man page:      /usr/local/php5/man/man1/
Installing build environment:     /usr/local/php5/lib/php/build/
Installing header files:          /usr/local/php5/include/php/
Installing helper programs:       /usr/local/php5/bin/
  program: phpize
  program: php-config
Installing man pages:             /usr/local/php5/man/man1/
  page: phpize.1
  page: php-config.1
Installing PEAR environment:      /usr/local/php5/lib/php/
[PEAR] Archive_Tar    - installed: 1.3.11
[PEAR] Console_Getopt - installed: 1.3.1
warning: pear/PEAR requires package "pear/Structures_Graph" (recommended version 1.0.4)
warning: pear/PEAR requires package "pear/XML_Util" (recommended version 1.2.1)
[PEAR] PEAR           - installed: 1.9.4
Wrote PEAR system config file at: /usr/local/php5/etc/pear.conf
You may want to add: /usr/local/php5/lib/php to your php.ini include_path
[PEAR] Structures_Graph- installed: 1.0.4
[PEAR] XML_Util       - installed: 1.2.1
/usr/src/php-5.3.28/build/shtool install -c ext/phar/phar.phar /usr/local/php5/bin
ln -s -f /usr/local/php5/bin/phar.phar /usr/local/php5/bin/phar
Installing PDO headers:          /usr/local/php5/include/php/ext/pdo/
[root@node2 php-5.3.28]#
```

- 源码安装 Apache+PHP 整合

整合 apache+php 环境，修改 httpd.conf 配置文件，然后加入如下语句：

LoadModule php5_module modules/libphp5.so （默认已存在）

AddType application/x-httpd-php .php

DirectoryIndex index.php index.html (把 index.php 加入 index.html 之前)

然后在/usr/local/apache/htdocs 目录下创建 index.php 测试页面，执行如下命令：

```
cat >>/usr/local/apache/htdocs/index.php <<EOF
<?php
phpinfo();
?>
EOF
```

重新启动 apache 服务，通过 IP 访问界面如下图，即代表 LAMP 环境搭建成功。

System	Linux node2 2.6.18-308.el5 #1 SMP Tue Feb 21 20:06:06 EST 2012 x86_64
Build Date	May 26 2014 14:34:36
Configure Command	'./configure' '--prefix=/usr/local/php5' '--with-config-file-path=/usr/local/php/etc' '--with-apxs2=/usr/local/apache/bin/apxs' '--with-mysql=/usr/local/mysql/'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/php/etc
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626

- 源码安装 DISCUZ 论坛

下载 discuz 源码包文件，然后解压：

cd /usr/src ;wget

http://download.comsenz.com/DiscuzX/3.1/Discuz_X3.1_SC_UTF8.zip

解压 discuz 程序包：unzip Discuz_X3.1_SC_UTF8.zip -d /usr/local/apache/htdocs/

重命名程序文件：cd /usr/local/apache/htdocs/ ;mv upload/* .

赋予 discuz 目录完全访问权限：cd /usr/local/apache/htdocs/ ;chmod 777 -R data/ uc_server/ config/ uc_client/

然后访问 IP 安装 discuz 论坛，如下图，选择“我同意”

进入如下界面，数据库安装，如果不存在则需要新建数据库并授权。

数据库创建及授权命令如下：

```
create database discuz charset=utf8;

grant all on discuz.* to root@'localhost' identified by "123456";
```

```
[root@node2 ~]# /usr/local/mysql/bin/mysql -uroot -p123456
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.1.63-log Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database discuz charset=utf8;
Query OK, 1 row affected (0.00 sec)

mysql> grant all on discuz.* to root@'localhost' identified by "123456";
Query OK, 0 rows affected (0.00 sec)

mysql>
```

点击下一步，直至安装完成，进入等待已久的论坛画面：

自此 LAMP 环境整合并搭建成功，通过 IP 直接访问即可。

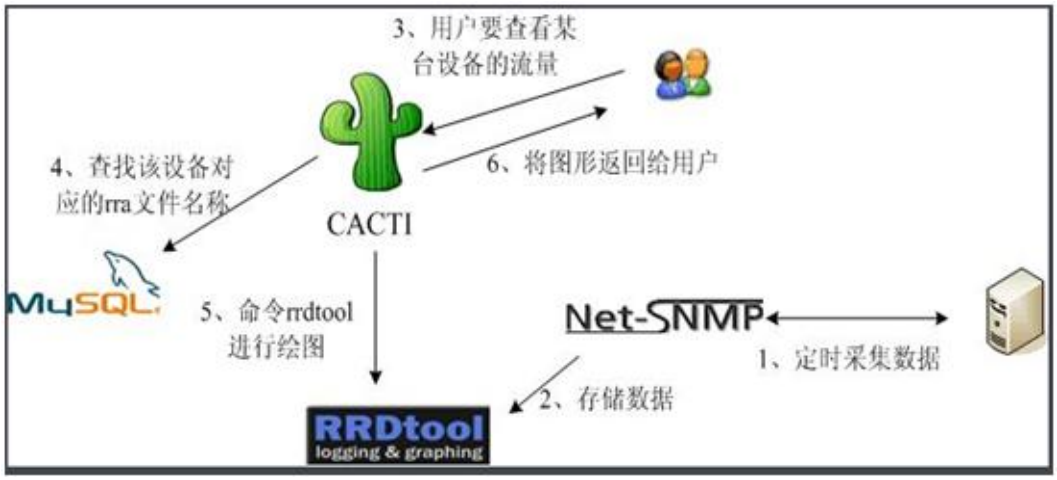
3.1. 9 Cacti 监控平台搭建

作为一名 Linux SA，日常最重要的就是保证网站正常稳定的运行，我们需要实时监控网站、服务器的运行状态，这时需要借助开源软件（cacti、nagios、zabbix 等）监控来实现。

Cacti 是用 php 语言实现的一个软件，它的主要功能是用 snmp 服务获取数据，然后用 rrdtool 储存和更新数据，当用户需要查看数据

的时候用 rrdtool 生成图表呈现给用户。因此，snmp 和 rrdtool 是 cacti 的关键。

Snmp 关系着数据的收集，rrdtool 关系着数据存储和图表的生成。snmp 抓到数据不是存储在 mysql 中，而是存在 rrdtool 生成的 rrd 文件中（在 cacti 根目录的 rra 文件夹下，一般以 rra 为后缀名称）。简单原理图如下：



1) Cacti 服务器端安装

官网下载 cacti 相关软件，一共需要三个软件，下载地址分别如下：

http://www.cacti.net/downloads/cacti-0.8.8a.tar.gz

http://oss.oetiker.ch/rrdtool/pub/rrdtool-1.4.5.tar.gz

http://www.cacti.net/downloads/spine/cacti-spine-0.8.8a.tar.gz

2) 安装 LAMP 系统环境

这里采用 yum 安装方式，安装命令：

yum install httpd mysql mysql-server php php-mysql php-json php-pdo -y

3) 安装 rrdtool 采集工具

安装 rrdtool 之前需要安装相应的 lib 库，如下安装：

yum install cairo-devel libxml2-devel pango pango-devel –y

tar xzf rrdtool-1.4.5.tar.gz ;cd rrdtool-1.4.5 ; ./configure --prefix=/usr/local/rrdtool/

make &&make install ;ln -s /usr/local/rrdtool/bin/* /usr/local/bin/

4) 安装 SNMP 服务

yum install net-snmp net-snmp-utils －y

5) 安装 cacti 主程序

tar xzf cacti-0.8.8a.tar.gz && mv cacti-0.8.8a /var/www/html/cacti/

6) 创建 cacti 数据库

mysql -uroot －p 输入你的密码进入数据库，然后创建数据库：

create database cacti;创建数据库

grant all on cacti.* to cacti@'localhost' identified by "123456";创建用户并授权

flush privileges; 刷新权限

mysql 配置完毕后，把 cacti 数据导入 cacti 数据库

mysql -ucacti -p123456 cacti </var/www/html/cacti/cacti.sql

配置完 mysql 后，我们需要设置 cacti rra、log 目录的权限，这里设置为 777：

chmod -R 777 /var/www/html/cacti/rra/

chmod -R 777 /var/www/html/cacti/log/

7) 修改 cacti 全局配置文件

vi /var/www/html/cacti/include/config.php 为如下配置：

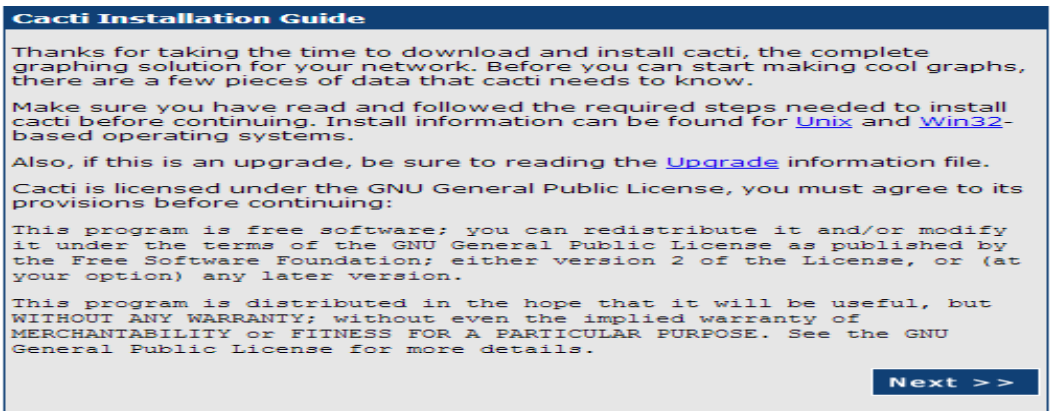
```
 | http://www.cacti.net/                                                    |
 +--------------------------------------------------------------------------+
*/

/* make sure these values refect your actual database/host/user/password */
$database_type = "mysql";
$database_default = "cacti";
$database_hostname = "localhost";
$database_username = "cacti";
$database_password = "123456";
$database_port = "3306";
$database_ssl = false;

/*
   Edit this to point to the default URL of your Cacti install
   ex: if your cacti install as at http://serverip/cacti/ this
   would be set to /cacti/
```

8) 添加 Rrdtool 抓图任务计划

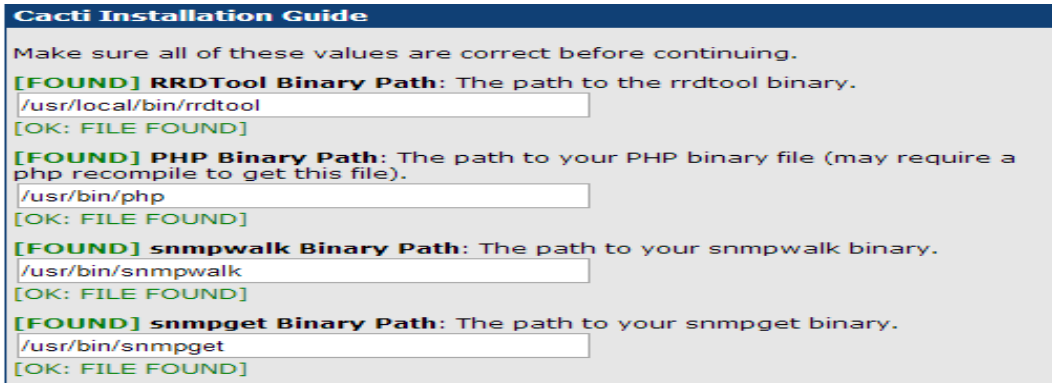
*/5 * * * * /usr/bin/php /var/www/html/cacti/poller.php >>/tmp/cacti_rrdtool.log 2 >&1

9) Cacti 安装完毕，测试访问

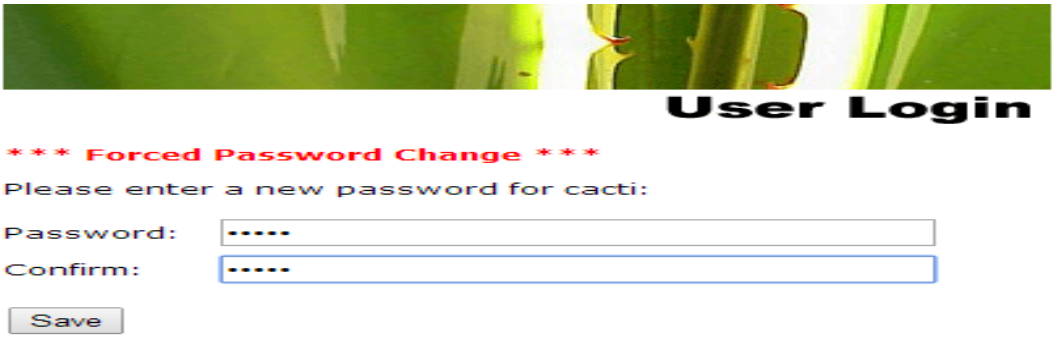
通过 http://ip/cacti/ 访问出现如下界面，点击 NEXT 下一步。



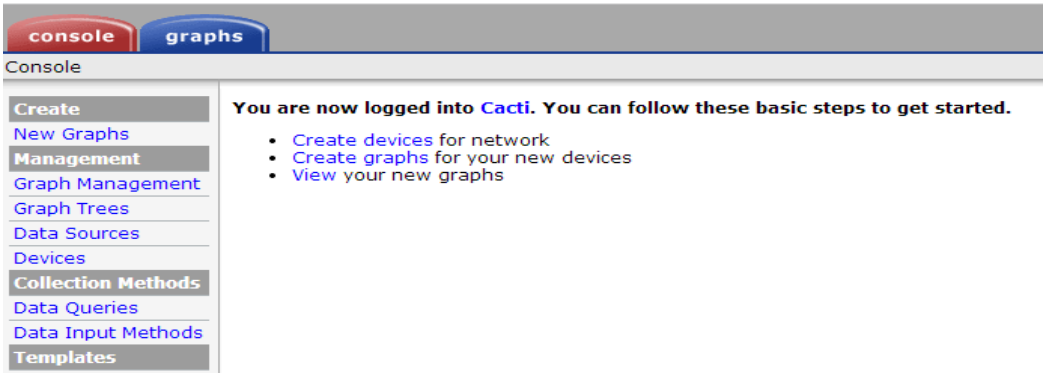
默认一直点击下一步：



进入登录界面，第一次需要修改密码：

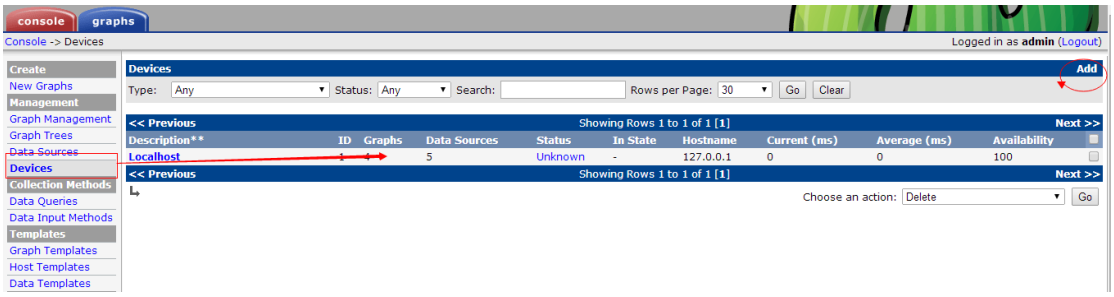


进入 Cacti 配置管理界面

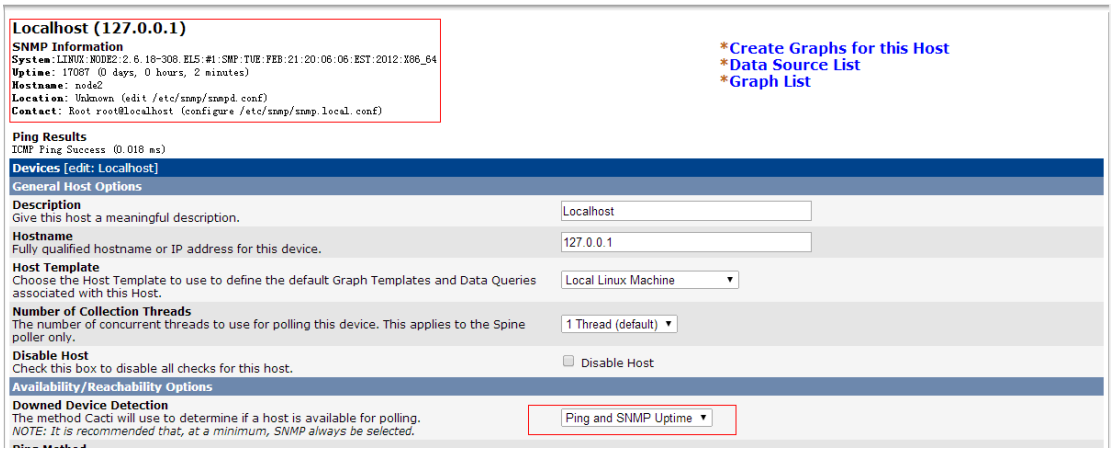


点击 device，可以添加设备，默认可以看到 127.0.0.1 这台服务器。

右上角 Add 可以增加设备。



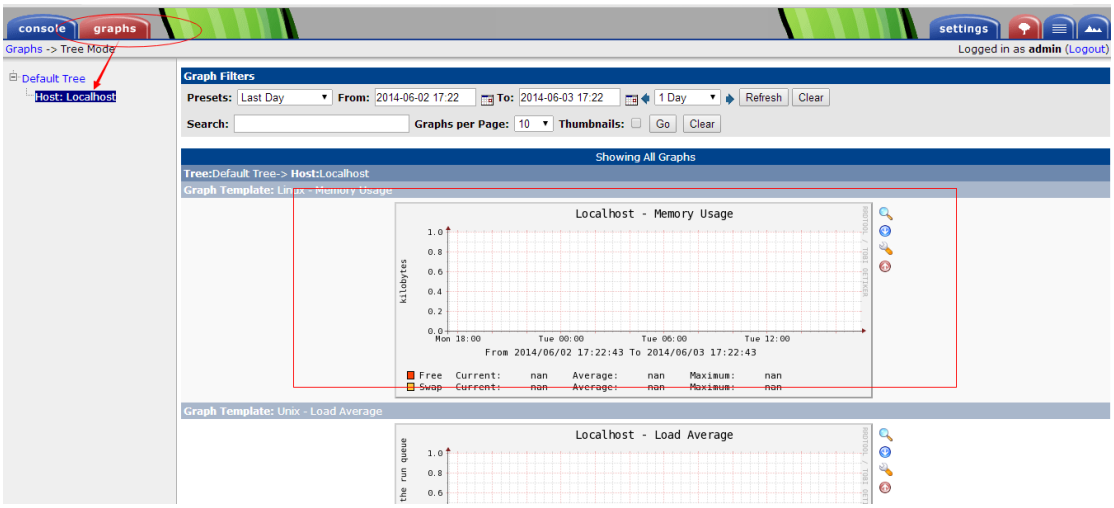
点击 localhost 可以看到具体的设置，包括采用的 snmp 协议版本，监控的名称等：



如果出现 snmp error，检查 snmp 服务是否已启动，或者是否有权限。

出现如上信息则表示正常。

点击左上角第二个按钮，graphs 查看 cacti 图像—选择 localhost 主机—右边会显示 cacti 每 5 分钟的监控图像。



更多 cacti 深入知识，根据需求深入讲解。

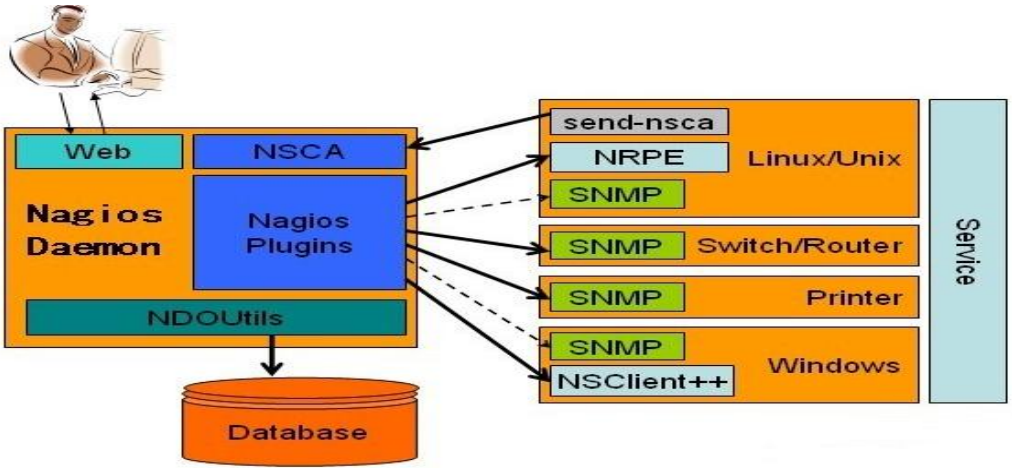
3.1. 10 Nagios 监控平台搭建

Nagios 是一款开源的免费网络监视工具，能有效监控 Windows、Linux 和 Unix 的主机状态，交换机路由器等网络设置，打印机等。在系统或服务状态异常时发出邮件或短信报警第一时间通知网站运维

人员，在状态恢复后发出正常的邮件或短信通知。

Nagios 和 cacti 有什么区别呢？简单的来说 cacti 主要监控流量，服务器状态页面展示；nagios 主要监控服务，邮件及短信报警灯，当然也有简单的流量监控界面，二者综合使用效果更好。（附 Nagios 工作简单逻辑图）

Nagios 监控客户端需要借助插件及 NRPE 软件来实现，NRPE 作为中间的代理程序，接收 Nagios 服务器端发来的请求，另一端在远程主机上指定的相关的监控信息。



1) Nagios 服务端安装

同样安装 nagios 服务需要安装 LAMP 环境，这里省略，可以参考之前的 cacti PHP 环境安装方法：官网下载 nagios 相应版本和插件：

wget

http://sourceforge.net/projects/nagios/files/nagios-3.x/nagios-3.2.1/nagios-3.2.1.tar.gz/download

http://down1.chinaunix.net/distfiles/nagios-plugins-1.4.14.tar.gz

/usr/sbin/useradd nagios

tar zxvf nagios-3.2.1.tar.gz

cd nagios-3.2.1

./configure --prefix=/usr/local/nagios --with-command-group=nagios

make all

make install //来安装主程序,CGI 和 HTML 文件

make install-init //在/etc/rc.d/init.d 安装启动脚本

make install-config // 来安装示例配置文件,安装的路径是 /usr/local/nagios/etc

make install-commandmode //来配置目录权限

make install-webconf // 配置 nagios 跟 apache 整合

2) 安装 Nagios-plugins

tar zxvf nagios-plugins-1.4.14.tar.gz

cd nagios-plugins-1.4.14

./configure --prefix=/usr/local/nagios --with-nagios-user=nagios --with-nagios-group=nagios

make && make install

3) nagios 访问控制设置

htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

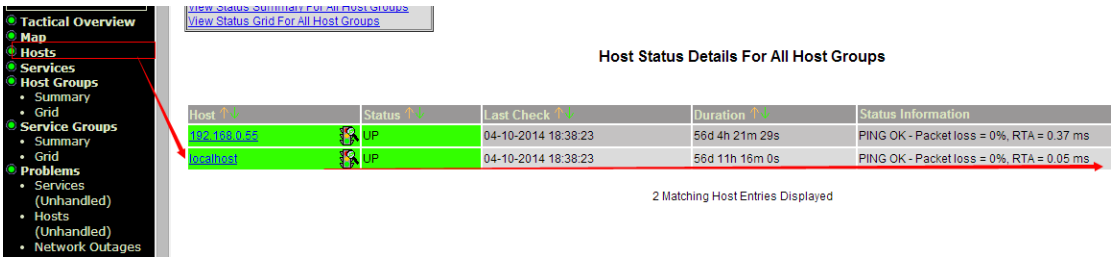
输入两次密码即可,登录页面的时候会用到这个密码.

4) Nagios 测试访问

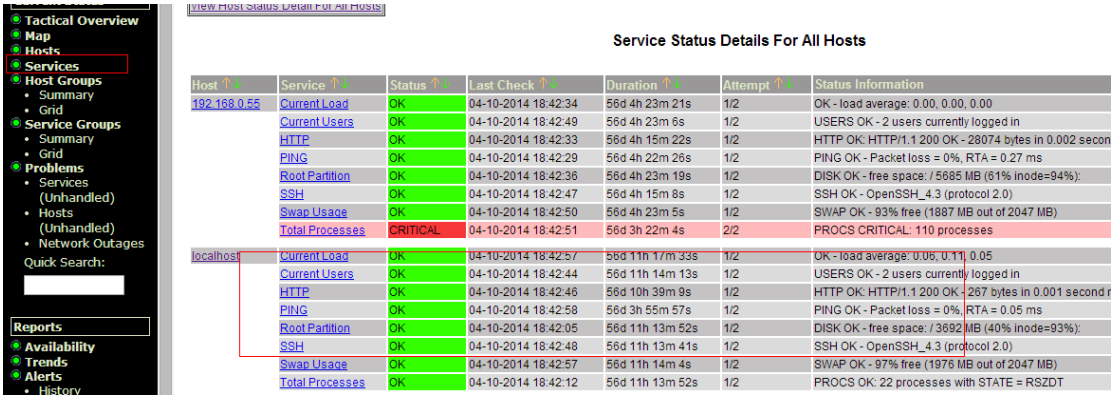
重启 nagios ,/etc/init.d/nagios restart ;/etc/init.d/httpd restart ;

http://localhost/nagios/ 如下图：

点击左侧的 Hosts 可以看到右侧默认 localhost 主机的监控，UP 表示主机目前运行正常：



点击左侧的 Services 可以看到右侧默认 localhost 监控的各个服务的状态，绿色 OK 表示正常：



5) Nagios 案例配置

默认安装完 nagios，配置文件主目录在/usr/local/nagios/下，目录各种的功能如下：

bin	Nagios 可执行程序所在目录
etc	Nagios 配置文件所在目录

sbin	Nagios CGI 文件所在目录，也就是执行外部命令所需文件所在的目录
share	Nagios 网页文件所在的目录
libexec	Nagios 外部插件所在目录
var	Nagios 日志文件、lock 等文件所在的目录
var/archives	Nagios 日志自动归档目录
var/rw	用来存放外部命令文件的目录

这里先来了解 etc/objects 目录主要包括监控主机的配置、模板、监控时间段等配置文件。

简单来添加一个客户端监控的步骤：

cp localhost.cfg 192.168.149.128.cfg

把默认配置文件里面的 locahost、127.0.0.1、check_local 替换成最新

sed -i 's#localhost#192.168.149.128#g;s#127.0.0.1#192.168.149.128#g;s#check_local#check#g ; s#linux-servers#192.168.149.128#g ' 192.168.149.128.cfg

在 nagios.cfg 36 行 后 加 入

cfg_file=/usr/local/nagios/etc/objects/192.168.149.128.cfg

sed -i '36a cfg_file=/usr/local/nagios/etc/objects/192.168.149.128.cfg' /usr/local/nagios/etc/nagios.cfg

最后执行:/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg 没有报错即可。

默认有报错，因为没有在客户端安装 nagios 插件及 NRPE，需删掉配置文件里 disk、swap、process、user、cpu 等监控配置段：

```
define service{
        use                             local-service
        host_name                       192.168.149.128
        service_description             Swap Usage
        check_command                   check_swap!20!10
        }
```

```
Checking host escalations...
        Checked 0 host escalations.
Checking host dependencies...
        Checked 0 host dependencies.
Checking commands...
        Checked 24 commands.
Checking time periods...
        Checked 5 time periods.
Checking for circular paths between hosts...
Checking for circular host and service dependencies...
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@node2 objects]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

这里注意* 如果没有配置 check_nrpe 监控，默认不能监控客户端例如 swap、disk、CPU、process 等状态，需要在配置文件里删除或者注释掉。

如下是刚刚添加的默认的客户端监控图：

Host	Service	Status	Last Check	Duration	Attempt	Status Information
192.168.149.128	HTTP	CRITICAL	06-05-2014 19:18:59	0d 0h 1m 31s	2/4	Connection refused
	PING	OK	06-05-2014 19:18:54	0d 0h 0m 36s	1/4	PING OK - Packet loss = 0%, RTA = 0.28 ms
	SSH	PENDING	N/A	0d 0h 2m 15s+	1/4	Service check scheduled for Thu Jun 5 19:19:48 CS
localhost	Current Load	OK	04-10-2014 18:19:57	56d 11h 41m 8s	1/4	OK - load average: 0.00, 0.03, 0.04
	Current Users	OK	04-10-2014 18:19:44	56d 11h 37m 48s	1/4	USERS OK - 2 users currently logged in
	HTTP	OK	04-10-2014 18:19:46	56d 11h 2m 44s	1/4	HTTP OK: HTTP/1.1 200 OK - 267 bytes in 0.001 se
	PING	OK	06-05-2014 19:18:26	56d 4h 19m 32s	1/4	PING OK - Packet loss = 0%, RTA = 0.04 ms
	Root Partition	OK	06-05-2014 19:19:21	56d 11h 37m 27s	1/4	DISK OK - free space: / 3734 MB (40% inode=92%):
	SSH	OK	04-10-2014 18:19:48	56d 11h 37m 16s	1/4	SSH OK - OpenSSH_4.3 (protocol 2.0)
	Swap Usage	OK	04-10-2014 18:19:57	56d 11h 37m 39s	1/4	SWAP OK - 96% free (1950 MB out of 2047 MB)
	Total Processes	OK	04-10-2014 18:20:12	56d 11h 37m 27s	1/4	PROCS OK: 24 processes with STATE = RSZDT

6) Nagios 客户端插件安装

Nagios 客户端安装需要安装两个软件，nagios-plugins-1.4.15.tar.gz 和 nrpe-2.13.tar.gz，安装方法如下：

useradd nagios ;tar -xzf nagios-plugins-1.4.15.tar.gz &&cd nagios-plugins-1.4.15 &&./configure –prefix=/usr/local/nagios &&make &&make install

tar -xzf nrpe-2.13.tar.gz && cd nrpe-2.13 &&./configure --enable-ssl --with-ssl-lib &&make all && make install-plugin && make install-daemon && make install-daemon-config

chown -R nagios:nagios /usr/local/nagios/ ； cd .. ;cp nrpe.cfg /usr/local/nagios/etc/nrpe.cfg

启动 nrpe 客户端命令：/usr/local/nagios/bin/nrpe -c /usr/local/nagios/etc/nrpe.cfg –d

7) Nrpe 客户端配置

修改 vi /usr/local/nagios/etc/nrpe.cfg 修改默认配置段的内容如下，去掉#号，做相应修改。

command[check_users]=/usr/local/nagios/libexec/check_users -w 5 -c 10

command[check_load]=/usr/local/nagios/libexec/check_load -w 15,10,5 -c 30,25,20

```
command[check_disk]=/usr/local/nagios/libexec/check_disk -w 20 -c
10 -p /dev/sda2
command[check_procs]=/usr/local/nagios/libexec/check_procs -w
50 -c 100
command[check_swap]=/usr/local/nagios/libexec/check_swap -w
20 -c 10
```

8) Nagios 服务器 Nrpe 配置

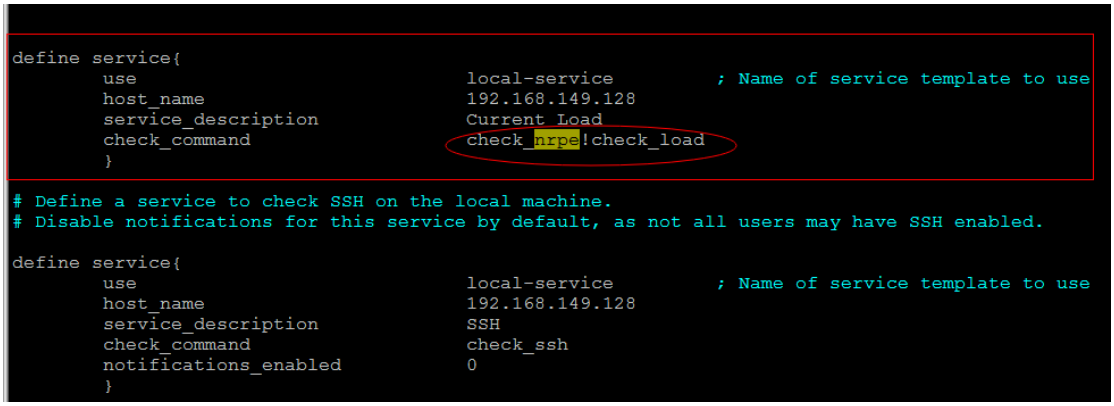
Nagios 服务器端也需要安装 nrpe，同时需要定义 Nrpe 监控命令，写 command.cfg 末尾即可:

```
define command{
        command_name check_nrpe
        command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -c
$ARG1$
        }
```

9) Nagios 监控端客户机配置

在 192.168.149.128.cfg 加入如下配置段，引用客户端 nrpe.cfg 里面配置的 check_load 命令，命令一般格式为：check_nrpe!command

```
define service{
        use                             local-service
        host_name                       192.168.149.128
        service_description             Current Load
        check_command
check_nrpe!check_load
        }
```



其他同理，添加的方法一样。只要在客户端 nrpe.cfg 里面添加的监控命令，都可以在服务端引用。

10) Nagios 监控端 HTTP 关键词

在真实的线上环境中，如果要监控 HTTP、web、tomcat 某个 URL 关键词，监控网站关键词是否被篡改，如果来实现呢？

这里可以使用默认监控命令 check_http 命令+相关的参数来实现，如下：

在 command.cfg 添加如下关键词监控命令：check_http_word，参数解析：-I 指定 IP 或者主机名，-u 指定 URL，-p 指定端口，-s 指定关键词。

```
define command{
        command_name    check_http_word
        command_line    $USER1$/check_http -I $HOSTADDRESS$ -u $$ARG1$ -p $$ARG2$ -s $ARG3$
        }
```

然后在服务器端监控主机的配置文件里面引用即可，引用的方法如下：

```
define service{
        use                             local-service         ; Name of service template to use
        host_name                       192.168.149.128
        service_description             SSH
        check_command                   check_ssh
        notifications_enabled           0
        }


# Define a service to check HTTP on the local machine.
# Disable notifications for this service by default, as not all users may have HTTP enabled.

define service{
        use                             local-service         ; Name of service template to use
        host_name                       192.168.149.128
        service_description             HTTP-ATM-Monitor
        check_command                   check_http_word!192.168.149.128!/index.html!82!ATM
        notifications_enabled           1
        }
```

也可以在服务器端命令行执行如下命令来做测试，例如监控页面不存在 ATM 关键词，但 82 端口 web 服务可以访问，依然会发送报警。

/usr/local/nagios/libexec/check_http -I 192.168.149.129 -u /index.html -p 82 -s "ATM"

```
[root@node2 objects]#
[root@node2 objects]# /usr/local/nagios/libexec/check_http -I 192.168.149.129 -u /index.html -p 82 -s "ATM"
HTTP CRITICAL: HTTP/1.1 200 OK - string 'ATM' not found on 'http://192.168.149.129:82/index.html' - 267 bytes
e time |time=0.000824s;;;0.000000 size=267B;;;0
[root@node2 objects]#
```

如上截图表示，关键词 ATM 不存在，则 nagios 在监控页面上会显示 CRITICAL 紧急。

11) Nagios 邮件及短信报警

使用 nagios 报警，以前可以用飞信发送报警，但是自从飞信更改接口后，就不方便了，那我们要发短信报警怎么办呢，我们可以 139 邮箱，机制是 nagios 给 139 邮箱发送信息，然后信息会自动发到我们绑定的手机。提前在 139 上绑定好手机即可。除此之外还可以使用短信猫（收费）设备来发送报警。

默认 command.cfg 里面已经配置好了邮件报警设置，可以使用默认的配置，使用系统默认的 mail 发送邮件；还可以自己定义发送的内容格式及发送的邮件 smtp 服务器端软件。

这里使用默认的配置文件，要能收到短信报警，除了在 139.com 界面绑定 139 邮箱之外，还需要在 nagios 服务器端配置文件修改邮件收件人如下：

修改配置文件：vi /usr/local/nagios/etc/objects/contacts.cfg 内容如下：

```
# This contact definition inherits a lot of default values from the 'generic-contact'
# template which is defined elsewhere.

define contact{
        contact_name                    nagiosadmin             ; Short name of user
        use                             generic-contact         ; Inherit default values from g
)
        alias                           Nagios Admin            ; Full name of user

        email                           wgkgood@139.com ; <<***** CHANGE THIS TO YOUR EMAIL ADD
        }


################################################################################
################################################################################
#
```

同样也可以使用 sed 命令修改：cd /usr/local/nagios/etc/objects/ ;

sed -i 's#nagios@localhost#wgkgood@139.com#g' contacts.cfg

如上配置完毕后，重启 nagios 服务，可以测试关闭某个服务，过一会就会收到 nagios 发来的报警邮件。

自此，Nagios 相关的配置就到此为止，当然有兴趣的童鞋还可以进一步研究，例如 nagios 跟 cacti 如何整合，nagios 如何优化等等。

3.1. 11 Kickstart 自动化安装平台

随着公司业务不断增加，经常需要采购新服务器，并要求安装 Linux 系统，并且要求 Linux 版本要一致，方便以后的维护和管理，每次人工安装 linux 系统会浪费掉更多时间，如果我们有办法能节省一次一次的时间岂不更好呢？

大中型互联网公司一次采购服务器上百台,如果采用人工手动一台一台的安装,一个人得搞坏 N 张光盘,得多少个加班加点才能完成这项”艰巨”的任务呢,我们可以看到全人工来完成这样的工作太浪费人力了,有没有自动化安装平台呢,通过一台已存在的系统然后克隆或者复制到新的服务器呢。Kickstart 可以毫不费力的完成这项工作。

PXE(preboot execute environment，预启动执行环境)是由 Intel 公司开发的最新技术，工作于 Client/Server 的网络模式，支持工作站通过网络从远端服务器下载映像，并由此支持通过网络启动操作系统，在启动过程中，终端要求服务器分配 IP 地址，再用 TFTP（trivial file transfer protocol）协议下载一个启动软件包到本机内存中执行。

要使用 kickstart 安装平台，包括的完整架构为：Kickstart+DHCP+NFS+TFTP+PXE，从架构可以看出，大致需要安装的服务，例如 dhcp、tftp、nfs、kickstart/pxe 等。

1) DHCP、TFTP 安装

```
yum install -y dhcp* tftp*
```

首先配置 tftp 服务：

```
vi /etc/xinetd.d/tftp
service tftp
{
disable = no
socket_type = dgram
protocol = udp
```

wait = yes

user = root

server = /usr/sbin/in.tftpd

server_args = -u nobody -s /tftpboot

per_source = 11

cps = 100 2

flags = IPv4

}

只需要把 disable = yes 改成 disable = no 即可。

2) TFTP+PXE 配置

要实现远程安装系统，首先需要在 TFTPBOOT 目录指定相关 PXE 内核模块及相关参数。配置步骤如下：

mount /dev/cdrom /mnt 挂载本地光盘

#如果系统是 5.x，默认 tftpboot 目录已经自动创建到/根目录下

#如果系统是 6.x，默认 tftpboot 目录在/var/lib/下，所以 centos6.x 需要做软链接到/根目录下。

cp /usr/lib/syslinux/pxelinux.0 ./

cp /mnt/images/pxeboot/{vmlinuz,initrd.img} ./ 拷贝内核至 tftpboot 目录下

mkdir -p pxelinux.cfg &&cp /mnt/isolinux/isolinux.cfg pxelinux.cfg/default

拷贝 isolinux.cfg 配置文件重命名，系统安装的时候会根据这个文

件的配置启动相应的选项。

修改 pexlinux.cfg/default 内容如下：

```
default linux
prompt 1
timeout 10
display boot.msg
F1 boot.msg
F2 options.msg
F3 general.msg
F4 param.msg
F5 rescue.msg
label centos5.8
kernel vmlinuz
append  ks=nfs:192.168.0.79:/centosinstall/ks.cfg  ksdevice=eth0
initrd=initrd.img
label text
kernel vmlinuz
append initrd=initrd.img text
label ks
kernel vmlinuz
append ks initrd=initrd.img
label local
```

localboot 1

label memtest86

kernel memtest

append –

解析：192.168.0.79 是 kickstart 服务器，/centosinstall 是 nfs 共享 linux 镜像的目录，也是 linux 存放安装文件的路径，ks.cfg 是 kickstart 主配置文件；设置 timeout 10 /*超时时间为 10S */；ksdevice=etho 代表当我们有多块网卡的时候，要实现自动化需要设置从 eth0 安装。TFTP 配置完毕，由于是 TFTP 是非独立服务，需要依赖 xinetd 服务来启动，启动命令为：

chkconfig tftp --level 35 on && service xinetd restart

3) NFS+KICKSTART 配置

远程系统安装，客户端需要下载系统所需的软件包，所以需要使用 NFS 或者 httpd 把镜像文件共享出来。

mkdir -p /centosinstall

nohup cp -rf /mnt/* /centosinstall &

echo “/centosinstall *(rw,sync)” >>/etc/exports

在 NFS 配置文件/etc/exports 中加入如上语句：/centosinstall *(rw,sync)，表示允许任何主机访问/centosinstall 目录，有读写权限。

配置 kickstart，可以使用 system-kickstart 系统软件包来配置，也可以直接拷贝/root/目录下 anaconda-ks.cfg 重命名为 ks.cfg，并把 ks.cfg 拷贝至刚共享的/centosinstall 目录下,赋权限为 chmod 777 ks.cfg

如下我这里采用配置文件内容如下：vi ks.cfg

```
# Kickstart file automatically generated by anaconda.
install
text
nfs --server=192.168.0.79 --dir=/centosinstall
key --skip
lang zh_CN.UTF-8
keyboard us
network --device eth0 --bootproto=dhcp --noipv6
rootpw 123456
firewall --disabled
authconfig --enableshadow --enablemd5
selinux --disabled
timezone Asia/Shanghai
bootloader --location=mbr --driveorder=sda --append="rhgb
quiet"
clearpart --all --initlabel
part /boot --fstype ext3 --size=200
part swap --size=4000
```

```
part / --fstype ext3 --size=80000
part /data --fstype ext3 --size=1 --grow
%packages
@admin-tools
@base
@core
@development-libs
@development-tools
@editors
@system-tools
@base-x
@chinese-support
keyutils
kexec-tools
trousers
fipscheck
device-mapper-multipath
imake
audit
xorg-x11-server-Xnest
xorg-x11-server-Xvfb
```

第一步 install，以 text 字符界面，指定 nfs 共享 IP 和目录，设置安装后的服务器字符集、网络分配方式、密码；

然后设置防火墙状态，磁盘采用 MBR 方式引导，然后客户机分区的情况；%packages

后写入客户机系统需要安装的软件包，可以自己定制。更多详细的参数在此就不做过多的说明了，可以进一步学习。

在真实环境中，通常我们会发现一台服务器好几块硬盘，做完 raid，整个硬盘有等 10T，如果来使用 kickstart 自动安装并分区呢；一般服务器硬盘超过 2T，如何来使用 kickstart 安装配置呢？这里就不能使用 MBR 方式来分区，需要采用 GPT 格式来引导并分区。

需要在 ks.cfg 末尾添加如下命令来实现需求：

```
%pre
parted -s /dev/sdb  mklabel gpt
%end
```

为了实现 kickstart 安装完系统后，自动初始化系统等等工作，我们可以在系统安装完后，自动执行定制的脚本，需要在 ks.cfg 末尾加入如下配置：

```
%post
mount  -t  nfs 192.168.0.79:/centos/init /mnt
cd  /mnt/ ;/bin/sh  auto_init.sh
%end
```

4) DHCP 配置及测试

Pxe+kickstart 自动安装系统，需要用到 DHCP 分配的客户端的 IP 地址，这里直接上 dhcpd.conf 配置文件：

```
ddns-update-style interim;
ignore client-updates;
next-server 192.168.0.79;
filename "pxelinux.0";
allow booting;
allow bootp;
subnet 192.168.0.0 netmask 255.255.255.0 {
# --- default gateway
option routers                  192.168.0.1;
option subnet-mask           255.255.252.0;
range dynamic-bootp 192.168.0.100 192.168.0.200;
host ns {
hardware ethernet    00:1a:a0:2b:38:81;
fixed-address 192.168.0.101;}
}
```

最后重启所有服务，并关闭 iptables 和 selinux，然后找一台新服务器，接入网线与 kickstart 服务器在一个交换机或通过中继能获取到 IP 的网络即可。

```
service   xinetd   restart
service nfs restart
```

service dhcpd restart

注*KICKSTART 所有配置就此告一段落，真实环境需要注意，新服务器跟 kickstart 最后独立在一个网络，不要跟办公环境或者服务器机房网络混在一起，如果别的机器以网卡就会把它的系统重装成 Linux 系统。

4. Linux 编程篇

4.1 Linux Shell 编程

4.1. 1 Shell 编程简介

shell 是操作系统的最外层。shell 合并编程语言以控制进程和文件，以及启动和控制其它程序。shell 通过提示您输入，向操作系统解释该输入，然后处理来自操作系统的任何结果输出来管理您与操作系统之间的交互。

Shell 是用户与 Linux 操作系统之间沟通的桥梁。用户可以输入命令执行，又可以利用 Shell 脚本编程去运行。随着 Linux 企业应用越来越多，维护 Linux 日常工作频繁，所以如果单靠手工去敲打命令是非常困难的，所以学会熟练使用 SHELL 编程是每个 Linux SA 必备的功课。

Linux Shell 种类非常多，常见的有：Bourne Shell（/usr/bin/sh 或 /bin/sh）、Bourne Again Shell（/bin/bash）、C Shell（/usr/bin/csh）、K Shell（/usr/bin/ksh）、Shell for Root（/sbin/sh）等。不同的 Shell 语言的语法有所不同，所以不能交换使用。

最常用的 shell 是 Bash，也就是 Bourne Again Shell，由于易用和免费，Bash 在日常工作中被广泛使用，也是大多数 Linux 系统默认的 Shell。接下来我们来写一个简单的 shell 脚本。(shell 脚本一般文件名以.sh 结尾，同时文件第一行定义该脚本为 shell 脚本)

vi first_shell.sh

#!/bin/bash

#This is my First shell

echo "Hello World !"

这就是我们的第一个脚本，是不是很简单呢，注解如下：

#!/bin/bash //表示定义该脚本是一个 shell 脚本（固定格式）。

#This is my First shell //这里的#号属于注解，没有任何的意义，SHELL 不会解析它。

echo "Hello World !" //shell 脚本主命令，我们执行这个脚本讲看到: Hello World ! 信息。

脚本编写完毕，如何来执行呢，首先执行 shell 脚本需要执行权限，赋予执行权限：

chmod o+x first_shell.sh 然后./first_shell.sh 执行即可;也可以直接使用命令执行: /bin/sh first_shell.sh，显示效果一样。

4.1. 2 Shell 变量设置

Shell 编程语言是非类型的解释型语言,不像 C++/JAVA 语言编程时需要事先声明变量，SHELL 给一个变量赋值,实际上就是定义了变量,

在 Linux 支持的所有 shell 中,都可以用赋值符号(=)为变量赋值。

SHELL 变量可分为两类：局部变量和环境变量。局部变量只在创建它们的 shell 脚本中使用。而环境变量则可以在创建它们的 shell 及其派生出来的任意子进程中使用。有些变量是用户创建的，其他的则是专用 shell 变量。

例如在脚本里面定义 A=123 ,定义这样一个变量，前面变量名，后面是变量的值。

引用变量可以使用$A，把变量放在脚本里面会出现什么样的效果呢？如下：

```
#!/bin/bash
#Author wugk 2014-06-10
A=123
echo  “Printf variables equal is $A”
```

执行脚本：sh test.sh，结果将会显示：

```
Printf variables equal is 123
```

简单的理解变量，相当于定义一个别名-名称，引用的时候加上$符号就可以了。

例如定义变量 name=wuguangke

执行 echo $name 将会显示 wuguangke

SHELL 常见的系统变量解析:

$0 当前程序的名称

$n 当前程序的第 n 个参数,n=1,2,…9

$* 当前程序的所有参数(不包括程序本身)

$# 当前程序的参数个数(不包括程序本身)

$? 命令或程序执行完后的状态，一般返回 0 表示执行成功。

$UID 当前用户的 ID

$PWD 当前所在的目录

4.1. 3 Shell 流程控制语句

在 Linux Shell 编程中，if、for、while、case 等条件流程控制语句用的非常多，把这些学好，对提升脚本的功力有非常大的帮助。下面将逐个来讲解具体的用法：

- If 条件判断语句

If (表达式) #if (Variable in Array)

语句 1

else

语句 2

fi

案例一，测试数字大小

```
#!/bin/sh
NUM=100
if (( $NUM > 4 )) ;then
    echo “this num is $NUM greater  4 !”
fi
```

案例二，测试目录是否存在，不存在则新建（注意，中括号之间必须要空格）

```
#!/bin/sh
#judge dir exist
if    [    !    -d    /data/20140515 ];then
      mkdir    -p /data/20140515
else
    echo    “This DIR is exist,Please exit …..”
fi
```

逻辑运算符解析：

-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

-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

案例三，多个条件测试判断

```
#!/bin/sh
scores=80;
if    [[ $scores -gt 85 ]]; then
    echo "very good!";
elif [[ $scores -gt 75 ]]; then
    echo "good!";
elif [[ $scores -gt 60 ]]; then
    echo "pass!";
else
    echo "no pass!";
fi;
```

- 循环语句 for

  For 变量 in 字符串

  do

  语句 1

  done

案例一，打印 seq 多个数

```
#!/bin/sh
for     i     in    `seq 15`
do
    echo    “NUM is $i”
done
```

案例二，找到相关 log，然后批量打包

```
#!/bin/sh
for     i     in    `find /var/log    -name “*.log”`
do
    tar –czf    2014log.tgz $i
done
```

- 循环语句 while

  while 条件语句

  do

  语句 1

  done

案例一，while 条件判断数字

```
#!/bin/sh
i=1;
while [[ $i -lt 10 ]];do
        echo $i;
        ((i++));
done;
```

案例二，while 逐行读取某个文件

```
#!/bin/sh
while read line
do
        echo    $line;
done    < /etc/hosts
```

- Until 循环语句

```
until  条件
do
     action
done
```

直到满足条件，才退出。否则执行 action。

案例一，条件判断数字

```
#!/bin/sh
a=10;
until [[ $a -lt 0 ]];do
echo $a;
((a--));
done;
```

- Case 选择语句

```
case $arg in
    pattern1)
    语句 1
    ;;
    pattern2)
    语句 2
    ;;
    *)
    语句 3
    ;;
esac
```

案例一，创建选择参数脚本

```
#!/bin/sh
case $1 in

        monitor_log)
        monitor_log
        ;;
        archive_log)
        archive_log
        ;;

        *               )
        echo "Usage:{$0 monitor_log | archive_log |help }"
        ;;
esac
```

- select 选择语句

```
#!/bin/sh
PS3="What you like most of the open source system?"
select i in CentOS RedHat Ubuntu
do
    echo "Your Select System: "$i
done
```

4.1. 4 Shell 脚本案例

a) 自动删除 test.txt 文件脚本

脚本的功能实现从/root/目录 cp 拷贝 test.txt 到/tmp 目录，并且在/tmp 目录创建一个目录 abc,并且删除原/root/下 test.txt。

首先命名脚本名称为 auto_cp.sh（名称可以自己定义），内容如下：

```
#!/bin/bash
#This is First shell for auto cp Files
#定义文件和目录变量
FILES=/root/test.txt
DIR=/tmp
cp      $FILES    $DIR
cd      $DIR ; mkdir –p abc
rm          -rf    $FILES
echo    “The Shell Scripts exec successfully !”
```

“自动 CP 并删除文件”的脚本编写完毕，保存退出即可。

b) 自动备份 Mysql 数据库脚本

```
#!/bin/sh
#auto backup mysql
#wugk    2012-12-12
#Define PATH 定义变量
```

```
BAKDIR=/data/backup/mysql/`date +%Y-%m-%d`
MYSQLDB=webapp
MYSQLPW=backup
MYSQLUSR=backup
#must use root user run scripts  必须使用 root 用户运行，$UID 为系统变量
if
    [ $UID -ne 0 ];then
    echo This script must use the root user ! ! !
    sleep 2
    exit 0
fi
#Define DIR and mkdir DIR  判断目录是否存在，不存在则新建
if
    [ ! -d $BAKDIR ];then
    mkdir -p $BAKDIR
else
    echo This is $BAKDIR exists....
fi
#Use mysqldump backup mysql  使用 mysqldump 备份数据库
/usr/bin/mysqldump -u$MYSQLUSR -p$MYSQLPW -d $MYSQLDB >$BAKDIR/webapp_db.sql
```

```
echo "The mysql backup successfully "
```

c) 自动打包 tar 目录下 log 文件脚本

```
#!/bin/sh
#auto tar czf shell to Files
#Author wugk 2014-05-15
SRC_DIR=/opt/
DES_DIR=/opt/backup/`date +%Y%m%d`

If
    [ ! –d    $DES_DIR ];then
    Mkdir –p $DES_DIR
fi
for    i    in    `find    $SRC_DIR    -name “*.log”`
do
    tar    czf    $i.tgz    $i
done
echo “The scripts exec end, Files tar successfully !”
```

d) 自动拒绝恶意 IP 脚本

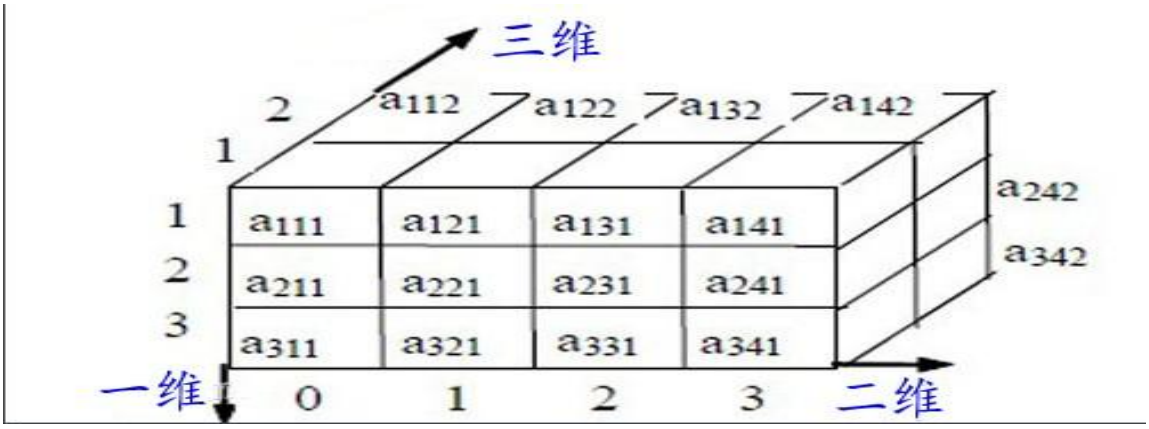
```
 #!/bin/sh
#auto drop ssh failed IP address
```

```
#wugk 2013-1-2
#定义变量
SEC_FILE=/var/log/secure
#如下为截取 secure 文件恶意 ip 远程登录 22 端口，大于等于 4 次就写入防火墙，禁止以后再登录服务器的 22 端口
IP_ADDR=`tail -n 1000 /var/log/secure |grep "Failed password"| egrep -o "([0-9]{1,3}\.){3}[0-9]{1,3}" | sort -nr | uniq -c |awk ' $1>=4 {print $2}'`
IPTABLE_CONF=/etc/sysconfig/iptables
echo
cat <<EOF
++++++++++++++welcome to use ssh login drop failed ip++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
++++++++++++++++----------------------------------+++++++++++++++++
+
EOF
echo -n "请等待 5 秒后开始执行 "
for ((j=0;j<=4;j++)) ;do echo -n "----------";sleep 1 ;done
echo
for i in `echo $IP_ADDR`
```

```
do
   #查看 iptables 配置文件是否含有提取的 IP 信息
   cat $IPTABLE_CONF |grep $i >/dev/null
if
   [ $? -ne 0 ];then
   #判断 iptables 配置文件里面是否存在已拒绝的 ip，如何不存在就不再添加相应条目
   sed -i "/lo/a -A INPUT -s $i -m state --state NEW -m tcp -p tcp --dport 22 -j DROP" $IPTABLE_CONF
else
   #如何存在的话，就打印提示信息即可
   echo  "This is $i is exist in iptables,please exit ....."
fi
done
#最后重启 iptables 生效
/etc/init.d/iptables restart
```

4.1. 5　Shell 数组编程

数组，就是相同数据类型的元素按一定顺序排列的集合，就是把有限个类型相同的变量用一个名字命名，然后用编号区分他们的变量的集合，这个名字成为数组名，编号成为下标。



今天这里我们来探讨一维数组的定义、统计、引用和删除等操作。首先来定义一个一维数组：

A=(test1　test2　test3) ，定义数组一般以括号的方式来定义，数组的值可以随机定义。如何来引用呢？

echo　${A[0]}，代表引用第一个数组变量，结果会显示 test1,数组引用从 0 开始，代表第一个数组，依次类推。

echo　${A[1]}，代表引用第二个数组变量，结果会显示 test2,数组引用也是从 0 开始计算的。

如何显示该数组所有参数呢？echo　${A[@]} 将显示所有参数 test1 test2 test3 。

如何显示该数组参数个数呢？echo　${#A[@]} 将显示该数组的参数个数 3。

如果替换某个数组呢？例如替换第二个 test2 数组为 test5：echo ${A[@]/test2/test5}

如何删除一个数组呢？例如删除 test3 数组命令为：unset A[2] ;echo ${A[@]}查看效果。

那输入如何在编程来使用呢？请看下面例子：

```
#!/bin/sh
#Auto Make KVM Virtualization
#Auto config bond scripts
eth_bond()
{
NETWORK=(
   HWADDR=`ifconfig  eth0  |egrep  "HWaddr|Bcast"  |tr  "\n"  "  "|awk
'{print $5,$7,$NF}'|sed -e 's/addr://g' -e 's/Mask://g'|awk '{print $1}'`
   IPADDR=`ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk '{print
$5,$7,$NF}'|sed -e 's/addr://g' -e 's/Mask://g'|awk '{print $2}'`
   NETMASK=`ifconfig  eth0  |egrep  "HWaddr|Bcast"  |tr  "\n"  "  "|awk
'{print $5,$7,$NF}'|sed -e 's/addr://g' -e 's/Mask://g'|awk '{print $3}'`
   GATEWAY=`route -n|grep "UG"|awk '{print $2}'`
)
cat >ifcfg-bond0<<EOF
DEVICE=bond0
BOOTPROTO=static
${NETWORK[1]}
${NETWORK[2]}
${NETWORK[3]}
ONBOOT=yes
```

TYPE=Ethernet

NM_CONTROLLED=no

EOF

如上脚本为定义三个数组变量，然后分别来引用，这样让脚本可读性更强，更整洁。关于数组就简单的介绍到这里。

5. Linux 深入篇

5.1 构建 Nginx WEB 服务器

nginx [engine x]是 Igor Sysoev 编写的一个 HTTP 和反向代理服务器，另外它也可以作为邮件代理服务器。 它已经在众多流量很大的俄罗斯网站上使用了很长时间，这些网站包括 Yandex、Mail.Ru、VKontakte，以及 Rambler。

据 Netcraft 统计，在 2012 年 8 月份，世界上最繁忙的网站中有 11.48%使用 Nginx 作为其服务器或者代理服务器。目前互联网主流公司 360、百度、新浪、腾讯、阿里等都在使用 nginx 作为自己的 web 服务器。

Nginx 由内核和模块组成，其中，内核的设计非常微小和简洁，完成的工作也非常简单，仅仅通过查找配置文件将客户端请求映射到一个 location block（location 是 Nginx 配置中的一个指令，用于 URL 匹配），而在这个 location 中所配置的每个指令将会启动不同的模块去完成相应的工作。

Nginx 相对于 Apache 优点：

1) 高并发响应性能非常好，官方 Nginx 处理静态文件并发 5w/s

2) 反向代理性能非常好。（可用于负载均衡）

3) 内存和 cpu 占用率低。（为 Apache 的 1/5-1/10）

4) 功能较 Apache 少（常用功能均有）

5) 对 php 可使用 cgi 方式和 fastcgi 方式。

5.1. 1 Nginx WEB 安装

首先需要安装 pcre 库，然后再安装 Nginx：

#安装 pcre 支持 rewrite 库,也可以安装源码，注*安装源码时，指定 pcre 路径为解压

源码的路径，而不是编译后的路径，否则会报错

（make[1]: *** [/usr/local/pcre/Makefile] Error 127 错误）

yum install pcre-devel pcre -y

#下载 Nginx 源码包

cd /usr/src ;wget -c http://nginx.org/download/nginx-1.4.2.tar.gz

#解压 Nginx 源码包

tar -xzf nginx-1.4.2.tar.gz

#进入解压目录，然后 sed 修改 Nginx 版本信息为 WS

cd nginx-1.4.2 ; sed -i -e 's/1.4.2//g' -e 's/nginx\//WS/g' -e 's/"NGINX"/"WS"/g' src/core/nginx.h

#预编译 Nginx

useradd www ;./configure --user=www --group=www --prefix=/usr/local/nginx --with-http_stub_status_module --with-http_ssl_module

#.configure 预编译成功后，执行 make 命令进行编译

make

#make 执行成功后，执行 make install 正式安装

make install

#自此 Nginx 安装完毕

/usr/local/nginx/sbin/nginx -t 检查 nginx 配置文件是否正确，返回 OK 即正确。

[root@localhost ~]# /usr/local/nginx/sbin/nginx -t

nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

[root@localhost ~]#

然后启动 nginx，/usr/local/nginx/sbin/nginx 回车即可。查看进程是否已启动：

[root@localhost ~]# ps -ef |grep nginx

nobody 5381 30285 0 May16 ? 00:04:31 nginx: worker process

root 30285 1 0 2014 ? 00:00:00 nginx: master process /usr/local/nginx/sbin/nginx

root 32260 32220 0 12:34 pts/0 00:00:00 grep nginx

[root@localhost ~]#

5.1. 2 Nginx 虚拟主机配置

在真实的服务器环境，为了充分利用服务器资源，一台 nginx web 服务器同时会配置 N 个虚拟域名主机，即多个域名对于同样一个 80 端口。然后服务器 IP 数量很多，也可以配置基于多个 IP 对应同一个端口。

vi 修改 nginx.conf server 段配置内容如下：

```
#virtual hosts config 2014/5/18
server {
        listen          80;
        server_name    www.a.com;

        #access_log    logs/host.access.log    main;

        location / {
            root      html/a;
            index    index.html index.htm;
        }
```

```
server {
        listen          80;
        server_name    www.b.com;

        #access_log    logs/host.access.log    main;

        location / {
            root      html/b;
            index    index.html index.htm;
        }
```

创建两个不同的目录 mkdir –p /usr/local/nginx/html/{a,b}，然后分别在两个目录创建两个不同的 index.html 网站页面即可。通过客户端配置 hosts 指向两个域名，然后在 IE 浏览器访问测试效果。

5.1. 3 Nginx 性能优化

随着访问量的不断增加，需要对 Nginx 和内核做相应的优化来满足高并发用户的访问，那下面在单台 Nginx 服务器来优化相关参数。

1) Nginx.conf 配置优化：

worker_processes 8;

nginx 进程数，建议按照 cpu 数目来指定，一般为它的倍数。

worker_cpu_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;

为每个进程分配 cpu，上例中将 8 个进程分配到 8 个 cpu，当然可以写多个，或者将一
个进程分配到多个 cpu。

worker_rlimit_nofile 102400;

这个指令是指当一个 nginx 进程打开的最多文件描述符数目，理论值应该是最多打
开文件数（ulimit -n）与 nginx 进程数相除，但是 nginx 分配请求并不是那么均匀
，所以最好与 ulimit -n 的值保持一致。

use epoll;

使用 epoll 的 I/O 模型。epoll 是 Linux 内核为处理大批量文件描述符而作了改进的
poll，它能显著提高程序在大量并发连接中只有少量活跃的情况下的系统 CPU 利用
率。

worker_connections 102400;

每个进程允许的最多连接数，理论上每台 nginx 服务器的最大连接数为
worker_processes*worker_connections。

keepalive_timeout 60;

keepalive 超时时间，客户端到服务器端的连接持续有效时间,当出现对服务器的后

继请求时,keepalive-timeout 功能可避免建立或重新建立连接。

client_header_buffer_size 4k;

客户端请求头部的缓冲区大小，这个可以根据你的系统分页大小来设置，一般一个

请求的头部大小不会超过 1k，不过由于一般系统分页都要大于 1k，所以这里设置为

分页大小。分页大小可以用命令 getconf PAGESIZE 取得。

open_file_cache max=102400 inactive=20s;

这个将为打开文件指定缓存，默认是没有启用的，max 指定缓存数量，建议和打开

文件数一致，inactive 是指经过多长时间文件没被请求后删除缓存。

open_file_cache_valid 30s;

这个是指多长时间检查一次缓存的有效信息。

open_file_cache_min_uses 1;

open_file_cache 指令中的 inactive 参数时间内文件的最少使用次数，如果超过这

个数字，文件描述符一直是在缓存中打开的，如上例，如果有一个文件在 inactive

时间内一次没被使用，它将被移除。

2) Linux 内核参数优化：

net.ipv4.tcp_max_tw_buckets = 10000

timewait 的数量，默认是 180000。

net.ipv4.ip_local_port_range = 1024 65000

允许系统打开的端口范围。

net.ipv4.tcp_tw_recycle = 1

启用 timewait 快速回收。

net.ipv4.tcp_tw_reuse = 1

开启重用。允许将 TIME-WAIT sockets 重新用于新的 TCP 连接。

net.ipv4.tcp_syncookies = 1

开启 SYN Cookies，当出现 SYN 等待队列溢出时，启用 cookies 来处理。

5.1. 4 Nginx 参数深入理解

Nginx 常用配置参数有 upstream，主要用于均衡后端多个实例：

Nginx 的 upstream 目前支持 5 种算法分配方式：

1) 轮询（默认 rr）

每个请求按时间顺序逐一分配到后端不同的服务器，如果后端某台服务器 down 掉，自动剔除，待恢复自动添加上。

2) Weight 权重

指定轮询权重，权重越高，处理的请求就越多，weight 和访问比率成正比，用于后端服务器性能不均的情况。

3) ip_hash

每个请求根据访问的 IP 的 hash 结果分配，这样每个访客固定访

问一个后端服务器，可以解决 session 的问题，一般用于登录会话。

4) fair（第三方）

按后端服务器的响应时间来分配请求，响应时间短的优先分配。

5) url_hash（第三方）

upstream 的 fail_timeout 和 max_fails 参数是用来判断负载均衡 upstream 中的某个 server 是否失效。

在 fail_timeout 的时间内，nignx 与 upstream 中某个 server 的连接尝试失败了 max_fails 次，则 nginx 会认为该 server 已经失效。在接下来的 fail_timeout 时间内，nginx 不再将请求分发给失效的 server。

例如在 nginx.conf 里面配置如下的 tdt_app 均衡：

```
upstream tdt_app {
    server 10.10.1.11:8080 weight=1 max_fails=2 fail_timeout=30s;
    server 10.10.1.12:8080 weight=1 max_fails=2 fail_timeout=30s;
}
```

Tdt_app 均衡两台后端 JAVA 服务，在 30 秒内 nginx 会与后端的某个 server 通信检测，如果检测连接失败 2 次，则 Nginx 会认为该 server 已经失效，然后踢出转发列表，然后在接下来的 30s 内，nginx 不再讲请求转发给失效的 server。

另外，fail_timeout 设置的时间对响应时间没影响，这个响应时间是用 proxy_connect_timeout 和 proxy_read_timeout 来控制的。

proxy_connect_timeout : Nginx 与后端服务器连接的超时时间，发起握手等候响应超时时间。

proxy_read_timeout：连接成功后_等候后端服务器响应时间，其实已经进入后端的排队之中等候处理（也可以说是后端服务器处理请求的时间）。

proxy_send_timeout :后端服务器数据回传时间，在规定时间之内后端服务器必须传完所有的数据。

keepalive_timout：一个 http 产生的 tcp 连接在传送完最后一个响应后，还需要等待多少秒后，才关闭这个连接。

5.1. 5 Nginx Rewrite 规则

Rewrite 规则含义就是某个 URL 重写成特定的 URL，从某种意义上说为了美观或者对搜索引擎友好，提高收录量及排名等。

Rewrite 规则的最后一项参数为 flag 标记，支持的 flag 标记主要有以下几种：

1) last ：相当于 Apache 里德(L)标记，表示完成 rewrite；
2) break；本条规则匹配完成后，终止匹配，不再匹配后面的规则
3) redirect：返回 302 临时重定向，浏览器地址会显示跳转后的 URL 地址
4) permanent：返回 301 永久重定向，浏览器地址栏会显示跳转后的 URL 地址

5) last 和 break 用来实现 URL 重写，浏览器地址栏 URL 地址不变。

a) 例如用户访问 www.test.com，想直接跳转到网站下面的某个页面，www.test.com/new.index.html 如何来实现呢？

我们可以使用 Nginx Rewrite 来实现这个需求，具体如下：

在 server 中加入如下语句即可：

```
rewrite    ^/$    http://www.wugk.com/index.html permanent;
```

*代表前面 0 或更多个字符

+代表前面 1 或更多个字符

？代表前面 0 或 1 个字符

^代表字符串的开始位置

$代表字符串结束的位置

。为通配符，代表任何字符

b) 例如多个域名跳转到同一个域名，nginx rewrite 规则写法如下：

```
server
{
        listen 80;
        server_name www.wugk.com wugk.com;
        if ($host != 'www.wugk.com' ) {
        rewrite ^/(.*)$    http://www.wugk.com/$1    permanent;
}
```

更多深入的 rewrite 可以继续学习。

5.2 构建 Rsync 同步服务器

Rsync 是 Unix/Linux 下的一款应用软件，利用它可以使多台服务器数据保持同步一致性，第一次同步时 rsync 会复制全部内容，但在下一次只传输修改过的文件。

Rsync 在传输数据的过程中可以实行压缩及解压缩操作，因此可以使用更少的带宽。可以很容易做到保持原来文件的权限、时间、软硬链接等。

5.2.1 Rsync 服务端配置

下面正式来配置 Rsync 服务器，模拟真实环境服务器数据同步。

A 是源服务器，B、C 为客户端服务器，因需求，B、C 服务器需要从 A 某个同步某个目录到本地。

正式安装，官网下载 rsync 稳定版本，然后进行安装编译。

cd /usr/src ;wget http://rsync.samba.org/ftp/rsync/src/rsync-3.0.7.tar.gz

tar xzf rsync-3.0.7.tar.gz && cd rsync-3.0.7 && ./configure --prefix=/usr/local/rsync &&make &&make install

安装完毕，配置 rsync 配置文件，默认/etc/不存在 rsyncd.conf 配置文件，需要手动创建，配置内容为如下：cat rsyncd.conf

#########[global] 全局配置

uid = nobody

gid = nobody

```
use chroot = no

max connections = 30

pid file = /var/run/rsyncd.pid

lock file = /var/run/rsyncd.lock

log file = /var/log/rsyncd.log

transfer logging = yes

log format = %t %a %m %f %b

syslog facility = local3

timeout = 300


[www]

read only = yes

path = /usr/local/webapps

comment = www

auth users =test

secrets file = /etc/rsync.pas

hosts allow = 192.168.0.11,192.168.0.12


[web]

read only = yes

path = /data/www/web

comment = web
```

auth users =test

secrets file = /etc/rsync.pas

hosts allow = 192.168.1.11,192.168.0.0/24

Rsync 配置参数说明：

[www]　#要同步的模块名

path = /usr/local/webapps　#要同步的目录

comment = www　　#这个名名称无所谓，最后模块名一直)

read only = no　　# no 客户端可上传文件,yes 只读

write only = no　# no 客户端可下载文件,yes 不能下载

list = yes　　#是否提供资源列表

auth users =test　#登陆系统使用的用户名，没有默认为匿名。

hosts allow = 192.168.0.10,192.168.0.20　#本模块允许通过的 IP 地址

hosts deny = 192.168.1.4　　#禁止主机 IP

secrets file=/etc/rsync.pas　#密码文件存放的位置

启动服务器端 RSYNC 主进程，/usr/local/rsync/bin/rsync --daemon ，监听端口 TCP 873

设置 rsync 服务器端同步密钥：

vi　/etc/rsync.pas

username:userpasswd (表示用户名：密码)

test:test999

保存完毕，chmod 600 /etc/rsync.pas 设置权限为宿主用户读写。

最后在客户端配置同步密钥和命令，如下设置即可同步。

vi　/etc/rsync.pas　输入服务器端配置的密码：

test999

保存即可开始同步：执行如下语句

Rsync -aP --delete test@192.168.0.100::www /usr/local/webapps --password-file=/etc/rsync.pas

Rsync -aP --delete test@192.168.0.100::web /data/www/web --password-file=/etc/rsync.pas

注*/usr/local/webapps 为客户端的目录，@前 test 是认证的用户名；IP 后面 www 为 rsync 服务器端的模块名称。

Rsync 常用参数解析：

-a, —archive	归档模式，表示以递归方式传输文件，并保持所有文件属性。
—exclude=PATTERN	指定排除一个不需要传输的文件匹配模式
—exclude-from=FILE	从 FILE 中读取排除规则
—include=PATTERN	指定需要传输的文件匹配模式
—delete	删除那些接收端还有而发送端已经不存在的文件

-P	等价于 —partial —progress
-v, —verbose	详细输出模式
-q, —quiet	精简输出模式
—rsyncpath=PROGRAM	指定远程服务器上的 rsync 命令所在路径
—password-file=FILE	从 FILE 中读取口令，以避免在终端上输入口令， 通常在 cron 中连接 rsync 服务器时使用

5.2. 2 Rsync 基于 SSH 同步

除了可以使用 rsync 密钥进行同步之外，还有一个比较简单的同步方法就是基于 linux ssh 来同步。具体方法如下：

rsync -aP --delete root@192.168.0.10:/data/www/webapps /data/www/webapps ，如果想每次同步不输入密码，需要做 Linux 主机之间免密码登录。

5.2. 3 Rsync 实时同步配置

在企业日常 web 应用中，某些特殊的数据需要要求保持跟服务器端实时同步，那我们该如何来配置呢？如何来实现呢？这里可以采用 rsync+inotify 来实现需求。

Inotify 是一个 Linux 特性，它监控文件系统操作，比如读取、写

入和创建。Inotify 反应灵敏，用法非常简单，并且比 cron 任务的繁忙轮询高效得多。

Rsync 安装完毕后，需要安装 inotify 文件检查软件。同时为了同步的时候不需要输入密码，这样可以使用 ssh 免密钥方式进行同步。

安装 inotify-tools-3.14.tar.gz 软件，tar –xzf inotify-tools-3.14.tar.gz ;./configure ;make

;make install 即可。配置 auto_inotify.sh 同步脚本，内容如下：

```
#!/bin/sh
src=/data/webapps/www
des=/home/webapps/
ip=192.168.0.11
inotifywait -mrq --timefmt '%d/%m/%y-%H:%M' --format '%T %w%f' -e modify,delete,create,attrib ${src} | while read file
do
 for i in $ip
        do
        /usr/local/rsync/bin/rsync -aP --delete $src root@$ip:$des
        done
done
```

在服务器端后台启动该脚本，nohup sh auto_inotify.sh & ，

在服务器端目录新建或者删除，客户端都会实时进行相关操作。

5.3 Tomcat/Resin JAVA 服务器

Tomcat 是由 Apache 软件基金会下属的 Jakarta 项目开发的一个 Servlet 容器，按照 Sun Microsystems 提供的技术规范，实现了对 Servlet 和 JavaServer Page（JSP）的支持，Tomcat 本身也是一个 HTTP 服务器，可以单独使用，apache 是一个以 C 语言编写的 HTTP 服务器。Tomcat 主要用来解析 JSP 语言。目前最新版本为 8.0。

5.3.1 Tomcat 安装配置

安装 tomcat 之前需要安装 jdk (Java Development Kit) 是 Java 语言的软件开发工具包(SDK)，这里选择 jdk-6u18-linux-x64-rpm.bin，bin 文件安装跟 sh 文件方法一样，sh ./ jdk-6u18-linux-x64-rpm.bin，回车即可，默认安装到/usr/java/jdk1.6.0_18 目录下。

配置 java 环境变量，vi /etc/profile 添加如下语句：

```
export JAVA_HOME=/usr/java/jdk1.6.0_18
export CLASSPATH=$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib
export PATH=$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$PATH:$HOMR/bin
source /etc/profile //使环境变量立刻生效。
java -version //查看 java 版本，显示版本为 1.6.0_18，
```

证明安装成功。

在官网下载 tomcat 相应版本，这里下载的版本为 apache-tomcat-6.0.30.tar.gz，下载完后解压：

tar -xzf apache-tomcat-6.0.30.tar.gz ;mv apache-tomcat-6.0.30 /usr/local/tomcat 即可。

启动 tomcat，命令为：/usr/local/tomcat_test/bin/startup.sh

查看 ps -ef |grep tomcat 进程及端口是否存在，通过页面访问可以看到 tomcat 默认测试页面：

这个画面是默认网站，怎么来创建一个自己的网站页面呢，定义自己的发布目录，方法如下：在 server.xml 配置文件末尾加入如下行：（附截图）

<Context path="/" docBase="/data/webapps/www" reloadable="true"/>

```
        that are performed against this UserDatabase are immediately
        available for use by the Realm.  -->
    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
           resourceName="UserDatabase"/>

    <!-- Define the default virtual host
         Note: XML Schema validation will not work with Xerces 2.2.
     -->
    <Host name="localhost"  appBase="webapps"
          unpackWARs="true" autoDeploy="true"
          xmlValidation="false" xmlNamespaceAware="false">

      <!-- SingleSignOn valve, share authentication between web applications
           Documentation at: /docs/config/valve.html -->
      <!--
      <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
      -->

      <!-- Access log processes all example.
           Documentation at: /docs/config/valve.html -->
      <!--
      <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
             prefix="localhost_access_log." suffix=".txt" pattern="common" resolveHosts="false"/>
      -->
      <Context path="/" docBase="/data/webapps/www"  reloadable="true"/>

    </Host>
  </Engine>
```

在/data/webapps/www 目录下，创建自己的 jsp 代码，重启 tomcat 即可访问。

5.3. 2 Tomcat 性能优化

线上环境使用默认 tomcat 配置文件，性能很一般，为了满足大量用户的访问，需要对 tomcat 进行参数性能优化，具体优化的地方如下：

- Linux 内核的优化
- 服务器资源配置的优化
- Tomcat 参数优化
- 配置负载集群优化

这里着重讲解 tomcat 参数的优化：server.xml 文件,关闭 DNS 查询、配置最大并发等参数。

maxThreads：tomcat 起动的最大线程数，即同时处理的任务个数，默认值为 200

acceptCount：当 tomcat 起动的线程数达到最大时，接受排队的请求

个数，默认值为 100

当然这些值都不是越大越好，需要根据实际情况来设定。可以基于测试的基础上来不断的调优分析。

```
<Connector port="8080"
               protocol="org.apache.coyote.http11.Http11NioProtocol"
               connectionTimeout="20000"
               redirectPort="8443"
               maxThreads="5000"
               minSpareThreads="20"
               acceptCount="1000"
               disableUploadTimeout="true"
               enableLookups="false"
               URIEncoding="UTF-8" />
```

Catalina.sh JVM 参数优化，添加如下内容：

```
CATALINA_OPTS="$CATALINA_OPTS –Xms4000M –Xmx4000M –Xmn1000M -XX:SurvivorRatio=4 -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=82 -DLOCALE=UTF-16LE -DRAMDISK=/ -DUSE_RAM_DISK=ture -DRAM_DISK=true"
```

配置多个 tomcat 实例，方法也很简单，只需要在服务器上 cp 多个 tomcat，然后修改三个端口和发布目录即可，然后分别启动即可。

为了提升整个网站的性能，还需要在 tomcat 前面架设 nginx web 反向代理服务器，用以提高用户高速访问。

5.3. 3 Resin 安装配置

Resin 是 CAUCHO 公司的产品，是一个非常流行的 application server，对 servlet 和 JSP 提供了良好的支持，性能也比较优良，resin 自身采用 JAVA 语言开发。

resin 普通版本和 pro 版本主要区别是 pro 支持缓存和负载均衡。pro 因为有强大的 cache 功能，独立作为 web 服务器处理静态页面性能都可以和 apache 有一比。但普通版本独立作为 web 服务器性能就要差一些。当然可以使用 apache+resin 的方案借助 apache 的缓存功能提高性能。

一般个人使用都使用开源免费版，如果想更高的性能，可以购买使用企业版 resin，售后服务有保障。

```
wget http://www.caucho.com/download/resin-4.0.33.tar.gz
tar –xzvf  resin-4.0.33.tar.gz
cd resin-4.0.33 &&./configure  --prefix=/usr/local/resin
\--with-resin-log=/data/logs/resin/
--with-java-home=/usr/java/jdk1.6.0_18/
make &&make install
```

安装完毕后，修改/usr/local/resin/conf/resin.xml 配置文件发布目录，如图：

```
  <!-- define the servers in the cluster -->
  <server id="" address="127.0.0.1" port="6800">
  </server>

  <!-- the default host, matching any host name -->
  <host id="" root-directory=".">
    <!--
       - configures an explicit root web-app matching the
       - webapp's ROOT
      -->
    <web-app id="/" root-directory="/data/webapps/www"/>

    <!--
       - Administration application /resin-admin
      -->
    <web-app id="/resin-admin" root-directory="${resin.root}/doc/admin">
      <prologue>
        <resin:set var="resin_admin_external" value="false"/>
        <resin:set var="resin_admin_insecure" value="true"/>
      </prologue>
    </web-app>

    <!--
       - Resin documentation - remove for a live site
      -->
    <web-app id="/resin-doc" root-directory="${resin.root}/doc/resin-doc"/>

    <!--
       - <resin:LoadBalance regexp="^/load" cluster="backend-tier"/>
-- INSERT --
```

然后启动 resin，/usr/local/resin/bin/resin.sh start 测试成功访问如下图（发布目录未修改之前 resin 默认测试页面）：

5.3. 4 Resin 性能优化

Resin 同 tomcat 一样，都需要优化 JVM 参数，resin 的 JVM 参数配置在 resin.xml 里面，配置最大最小内存，会话保持时间及并发数等如下所示：

```
<http address="*" port="8080"/>

        <!-- SSL port configuration: -->

        <http address="*" port="8443">
```

```
    <jsse-ssl self-signed-certificate-name="resin@localhost"/>
  </http>
    <jvm-arg>-Xms4000m</jvm-arg>
    <jvm-arg>-Xmx4000m</jvm-arg>
    <jvm-arg>-Xmn1000m</jvm-arg>
    <jvm-arg>-XX:PermSize=128m</jvm-arg>
    <jvm-arg>-XX:MaxPermSize=256m</jvm-arg>
    <thread-max>10000</thread-max>
    <socket-timeout>30s</socket-timeout>
    <keepalive-max>5000</keepalive-max>
    <keepalive-timeout>60s</keepalive-timeout>
    <jvm-arg>-agentlib:resin</jvm-arg>
```

Resin 参数优化同样包括最大内存、最小内存，年轻带，最大并发，会话超时时间等。根据实际的应用来调节不同的参数。

5.3. 5 Resin 多实例配置

为了资源最大利用，单台服务器可以配置多个 resin 实例，配置 resin 多实例的方式跟 tomcat 大部分一致，但还有一些区别：

cd /usr/local/resin/conf 下，然后 cp resin.xml resin1.xml ；cp resin.xml resin2.xml

修改两个配置文件如下图所示：

HTTP 端口为 8080 Resin1.xml 配置如下：

```
<!-- defaults for each server, i.e. JVM -->
<server-default>
  <!-- The http port -->
  <http address="*" port="8080"/>

  <!-- SSL port configuration: -->
  <http address="*" port="8443">
    <jsse-ssl self-signed-certificate-name="resin@localhost"/>
  </http>

  <!--
     - <jvm-arg>-Xmx512m</jvm-arg>
     - <jvm-arg>-agentlib:resin</jvm-arg>
    -->

</server-default>

<!-- define the servers in the cluster -->
<server id="1" address="127.0.0.1" port="6800">
</server>

<!-- the default host, matching any host name -->
<host id="" root-directory=".">
  <!--
     - configures an explicit root web-app matching the
     - webapp's ROOT
    -->
  <web-app id="/" root-directory="/data/webapps/www1"/>
```

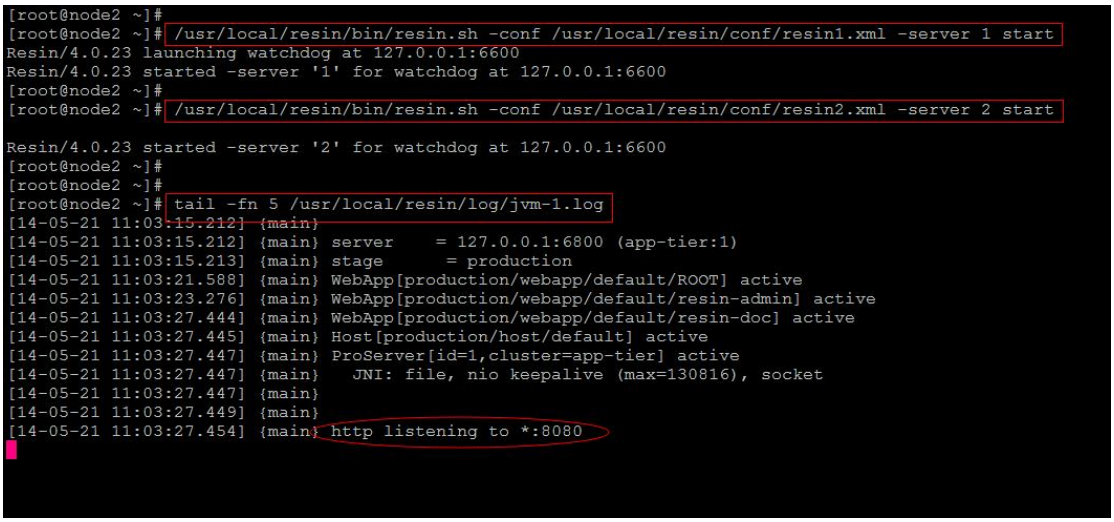
HTTP 端口为 8081 Resin2.xml 配置如下：

```
<!-- defaults for each server, i.e. JVM -->
<server-default>
  <!-- The http port -->
  <http address="*" port="8081"/>

  <!-- SSL port configuration: -->
  <http address="*" port="8444">
    <jsse-ssl self-signed-certificate-name="resin@localhost"/>
  </http>

  <!--
     - <jvm-arg>-Xmx512m</jvm-arg>
     - <jvm-arg>-agentlib:resin</jvm-arg>
    -->

</server-default>

<!-- define the servers in the cluster -->
<server id="2" address="127.0.0.1" port="6801">
</server>

<!-- the default host, matching any host name -->
<host id="" root-directory=".">
  <!--
     - configures an explicit root web-app matching the
     - webapp's ROOT
    -->
  <web-app id="/" root-directory="/data/webapps/www2"/>
```

创建两个发布目录 mkdir -p /data/webapps/{www1,www2}写入测试 jsp 文件即可。最后如下方法启动两个 resin 实例：

/usr/local/resin/bin/resin.sh -conf /usr/local/resin/conf/resin1.xml -server 1 start

/usr/local/resin/bin/resin.sh -conf /usr/local/resin/conf/resin2.xml -server 2 start



真实环境，需要调整 jvm 参数，需要在 resin.xml 里面配置，同时需要开启启动 resin，只需要把上述脚本加入/etc/rc.local 即可。

5.4 Nginx Tomcat 动静分离

Nginx 动静分离简单来说就是把动态跟静态请求分开，不能理解成只是单纯的把动态页面和静态页面物理分离。严格意义上说应该是动态请求跟静态请求分开，可以理解成使用 Nginx 处理静态页面，Tomcat、Resin 出来动态页面。

动静分离从目前实现角度来讲大致分为两种，一种是纯粹的把静态文件独立成单独的域名，放在独立的服务器上，也是目前主流推崇的方案；另外一种方法就是动态跟静态文件混合在一起发布，通过 nginx 来分开。这样也是本次课程要讲解的，具体怎么来实现呢，如

下图，通过 location 指定不同的后缀名实现不同的请求转发。

通过 expires 参数设置，可以使浏览器缓存过期时间，减少与服务器之前的请求和流量。具体 Expires 定义：是给一个资源设定一个过期时间，也就是说无需去服务端验证，直接通过浏览器自身确认是否过期即可，所以不会产生额外的流量。

此种方法非常适合不经常变动的资源。（如果经常更新的文件，不建议使用 Expires 来缓存），我这里设置 3d，表示在这 3 天之内访问这个 URL，发送一个请求，比对服务器该文件最后更新时间没有变化，则不会从服务器抓取，返回状态码 304，如果有修改，则直接从服务器重新下载，返回状态码 200。

```
####www.wuguangke.cn
server
  {
    listen       80;
    server_name  www.wuguangke.cn;
    index index.html index.htm;
#配置发布目录为/data/www/wugk
    root  /data/www/wugk;
    location /
    {
         proxy_next_upstream http_502 http_504 error timeout invalid_header;
         proxy_set_header Host  $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_pass http://tdt_wugk;
         expires      3d;
    }
#动态页面交给http://tdt_wugk，也即我们之前在nginx.conf定义的upstream tdt_wugk 均衡
    location ~ .*\.(php|jsp|cgi)?$
    {
         proxy_set_header Host  $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
         proxy_pass http://tdt_wugk;
    }
#配置Nginx动静分离，定义的静态页面直接从Nginx发布目录读取。
    location ~ .*\.(html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css)$
    {
    root /data/www/wugk;
    #expires定义用户浏览器缓存的时间为3天，如果静态页面不常更新，可以设置更长，这样可以节省带宽和缓解服务器的压
    expires      3d;
    }
#定义Nginx输出日志的路径
    access_log  /data/logs/nginx_wugk/access.log main;
    error_log   /data/logs/nginx_wugk/error.log  crit;
}
```

如下为 nginx.conf 里面 server 配置段，直接添加在 nginx.conf 里即可。

####www.wuguangke.cn

```
server
  {
    listen       80;
    server_name  www.wuguangke.cn;
    index index.html index.htm;
#配置发布目录为/data/www/wugk
    root  /data/www/wugk;
    location /
    {
         proxy_next_upstream http_502 http_504 error
timeout invalid_header;
         proxy_set_header Host  $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
         proxy_pass http://tdt_wugk;
         expires      3d;
    }
#动态页面交给 http://tdt_wugk，也即我们之前在 nginx.conf 定
义的 upstream tdt_wugk 均衡
    location ~ .*\.(php|jsp|cgi)?$
    {
```

```
        proxy_set_header Host  $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_pass http://tdt_wugk;
    }
#配置Nginx动静分离，定义的静态页面直接从Nginx发布目录读取。
    location
~ .*\.(html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css)$
    {
    root /data/www/wugk;
    #expires 定义用户浏览器缓存的时间为3天，如果静态页面不
常更新，可以设置更长，这样可以节省带宽和缓解服务器的压力
    expires      3d;
    }
#定义 Nginx 输出日志的路径
    access_log  /data/logs/nginx_wugk/access.log main;
    error_log   /data/logs/nginx_wugk/error.log  crit;
}
```

真实环境网站程序包只有一个，需要把这个程序包在 nginx 前端放一份，同时需要在 Tomcat、Resin 后端也放置一份，如果服务器涉及数量很多，那每台服务器都需要更新，可以使用批量更新方法。

5.5 LNAMP 高性能架构配置

LNAMP(Linux+Nginx+Apache+Mysql+PHP)架构受到很多 IT 企业的青睐,取代了原来认为很好的 LNMP(Linux+Nginx+Mysql+PHP)架构,那我们说 LNAMP 到底有什么优点呢,还得从 Nginx 和 apache 的优缺点说起。

Nginx 处理静态文件能力很强,Apache 处理动态文件很强而且很稳定,把二者综合在一块,性能提升很多倍。可能很多 Linux SA 在从事 LNMP 运维中，会发现 PHP（FastCGI）模式会出现一些 502 错误的现象，这是因为 Nginx+PHP（FastCGI）组合不稳定的原因造成的。

- 源码安装 LNAMP 之 Nginx

yum install prce-devel –y ;cd /usr/src ;wget http://nginx.org/download/nginx-1.6.0.tar.gz ;cd nginx-1.6.0 ;./configure –prefix=/usr/local/nginx && make &&make install

- 源码安装 LNAMP 之 Apache

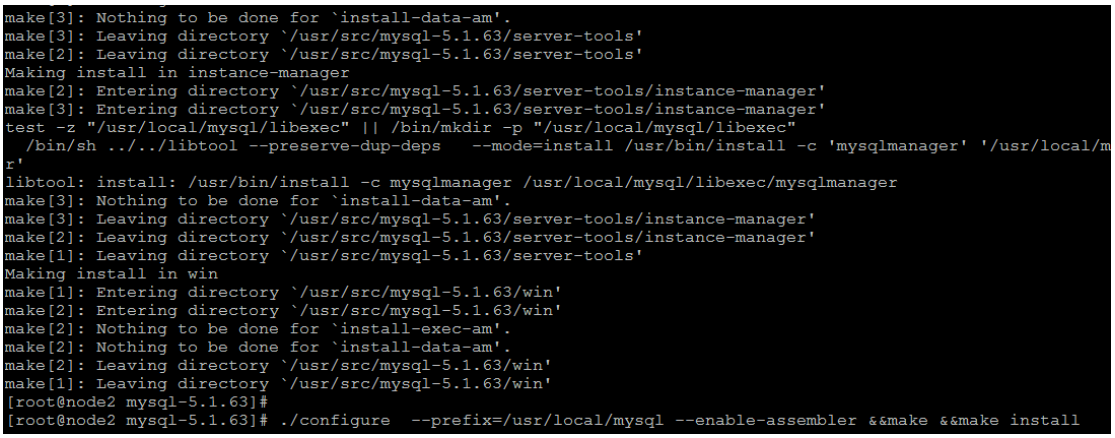
yum install apr-devel apr-util-devel –y；

cd /usr/src ; wget http://mirror.bit.edu.cn/apache/httpd/httpd-2.2.27.tar.gz ;tar xzf httpd-2.2.27.tar.gz ;cd httpd-2.2.27 ;./configure --prefix=/usr/local/apache --enable-so --enable-rewrite &&make &&make install

- 源码安装 LNAMP 之 MySQL

cd /usr/src ;wget

http://downloads.mysql.com/archives/mysql-5.1/mysql-5.1.63.tar.gz ;tar xzf mysql-5.1.63.tar.gz ;cd mysql-5.1.63 ;./configure --prefix=/usr/local/mysql --enable-assembler &&make &&make install



配置 Mysql 服务为系统服务：

cp /usr/local/mysql/share/mysql/my-medium.cnf /etc/my.cnf

cp /usr/local/mysql/share/mysql/mysql.server /etc/rc.d/init.d/mysqld

chkconfig --add mysqld

chkconfig --level 345 mysqld on

cd /usr/local/mysql

useradd mysql

chown -R mysql.mysql /usr/local/mysql

/usr/local/mysql/bin/mysql_install_db --user=mysql

chown -R mysql var

/usr/local/mysql/bin/mysqld_safe --user=mysql &

- 源码安装 LNAMP 之 PHP

cd /usr/src ;wget http://mirrors.sohu.com/php/php-5.3.28.tar.bz2 ;tar jxf

php-5.3.28.tar.bz2 ;cd php-5.3.28 ;./configure --prefix=/usr/local/php5 --with-config-file-path=/usr/local/php/etc --with-apxs2=/usr/local/apache/bin/apxs --with-mysql=/usr/local/mysql/

```
Installing PHP CLI man page:      /usr/local/php5/man/man1/
Installing build environment:     /usr/local/php5/lib/php/build/
Installing header files:          /usr/local/php5/include/php/
Installing helper programs:       /usr/local/php5/bin/
  program: phpize
  program: php-config
Installing man pages:             /usr/local/php5/man/man1/
  page: phpize.1
  page: php-config.1
Installing PEAR environment:      /usr/local/php5/lib/php/
[PEAR] Archive_Tar    - installed: 1.3.11
[PEAR] Console_Getopt - installed: 1.3.1
warning: pear/PEAR requires package "pear/Structures_Graph" (recommended version 1.0.4)
warning: pear/PEAR requires package "pear/XML_Util" (recommended version 1.2.1)
[PEAR] PEAR           - installed: 1.9.4
Wrote PEAR system config file at: /usr/local/php5/etc/pear.conf
You may want to add: /usr/local/php5/lib/php to your php.ini include_path
[PEAR] Structures_Graph- installed: 1.0.4
[PEAR] XML_Util       - installed: 1.2.1
/usr/src/php-5.3.28/build/shtool install -c ext/phar/phar.phar /usr/local/php5/bin
ln -s -f /usr/local/php5/bin/phar.phar /usr/local/php5/bin/phar
Installing PDO headers:          /usr/local/php5/include/php/ext/pdo/
[root@node2 php-5.3.28]#
```

- 源码安装 Apache+PHP 整合

整合 apache+php 环境，修改 httpd.conf 配置文件，然后加入如下语句：

LoadModule php5_module modules/libphp5.so （默认已存在）

AddType application/x-httpd-php .php

DirectoryIndex index.php index.html (把 index.php 加入 index.html 之前)

然后在/usr/local/apache/htdocs 目录下创建 index.php 测试页面，执行如下命令：

```
cat >>/usr/local/apache/htdocs/index.php <<EOF
<?php
phpinfo();
?>
```

EOF

重新启动 apache 服务，通过 IP 访问界面如下图，即代表 LAMP 环境搭建成功。

System	Linux node2 2.6.18-308.el5 #1 SMP Tue Feb 21 20:06:06 EST 2012 x86_64
Build Date	May 26 2014 14:34:36
Configure Command	'./configure' '--prefix=/usr/local/php5' '--with-config-file-path=/usr/local/php/etc' '--with-apxs2=/usr/local/apache/bin/apxs' '--with-mysql=/usr/local/mysql/'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/php/etc
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626

- 源码安装 DISCUZ 论坛

下载 discuz 源码包文件，然后解压：

cd /usr/src ;wget

http://download.comsenz.com/DiscuzX/3.1/Discuz_X3.1_SC_UTF8.zip

解压 discuz 程序包：unzip Discuz_X3.1_SC_UTF8.zip -d /usr/local/apache/htdocs/

重命名程序文件：cd /usr/local/apache/htdocs/ ;mv upload/* .

赋予 discuz 目录完全访问权限：cd /usr/local/apache/htdocs/ ;chmod 777 -R data/ uc_server/ config/ uc_client/

然后访问 IP 安装 discuz 论坛，如下图，选择“我同意”

进入如下界面，数据库安装，如果不存在则需要新建数据库并授权。

数据库创建及授权命令如下：

```
create database discuz charset=utf8;
```

grant all on discuz.* to root@'localhost' identified by "123456";

```
[root@node2 ~]# /usr/local/mysql/bin/mysql -uroot -p123456
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.1.63-log Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database discuz charset=utf8;
Query OK, 1 row affected (0.00 sec)

mysql> grant all on discuz.* to root@'localhost' identified by "123456";
Query OK, 0 rows affected (0.00 sec)

mysql>
```

点击下一步，直至安装完成，进入等待已久的论坛画面：

自此 LAMP 环境整合并搭建成功，那如何使用 Nginx 来整合 LAMP 呢？

- 源码安装 Nginx+LAMP 整合

先修改 apache 访问端口为 8080，Nginx 端口为 80。

然后修改 nginx 配置文件：vi /usr/local/nginx/conf/nginx.conf，server 配置段内容如下：

(定义 upstream 均衡模块，配置动静分离，动态转发至 apache，静态文件直接本地响应)

```
upstream app_lamp {
        server      127.0.0.1:8080 weight=1 max_fails=2 fail_timeout=30s;
        }
        server {
                listen              80;
                server_name    localhost;

                location / {
                        root      /usr/local/apache/htdocs;
                        index    index.php index.html index.htm;
                }
                location ~ .*\.(php|jsp|cgi)?$
        {
                  proxy_set_header Host    $host;
                  proxy_set_header X-Real-IP $remote_addr;
                  proxy_set_header                                X-Forwarded-For
$proxy_add_x_forwarded_for;
                  proxy_pass http://app_lamp;
        }
                location
~ .*\.(html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css)$
              {
```

```
        root    /usr/local/apache/htdocs;
        expires         3d;
    }
}
```

测试，访问 nginx ip+port 如下图所示：

查看系统启动的端口及进程如下图：

```
[root@node2 ~]# netstat -tnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:3306                0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:80                  0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:23                  0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:767                 0.0.0.0:*                   LISTEN
tcp        0      0 :::8080                     :::*                        LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
[root@node2 ~]# ps -ef |grep nginx
root     10966     1  0 15:34 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/n
nobody   11055 10966  0 15:41 ?        00:00:00 nginx: worker process
root     11187 17865  0 15:51 pts/1    00:00:00 grep nginx
[root@node2 ~]# ps -ef |grep mysql
root     10521     1  0 14:48 pts/1    00:00:00 /bin/sh /usr/local/mysql/bin/mysqld_safe --da
sr/local/mysql/var/node2.pid
mysql    10624 10521  0 14:48 pts/1    00:00:01 /usr/local/mysql/libexec/mysqld --basedir=/us
/var --user=mysql --log-error=/usr/local/mysql/var/node2.err --pid-file=/usr/local/mysql/var/
t=3306
root     11189 17865  0 15:51 pts/1    00:00:00 grep mysql
[root@node2 ~]# ps -ef |grep httpd |head -3
root     10719     1  0 14:58 ?        00:00:00 /usr/local/apache/bin/httpd -k restart
daemon   10773 10719  0 15:09 ?        00:00:00 /usr/local/apache/bin/httpd -k restart
daemon   10774 10719  0 15:09 ?        00:00:00 /usr/local/apache/bin/httpd -k restart
```

自此，LNAMP 全部整合完毕，接下来就是对系统内核、各个服务、

架构进行优化，同样优化是一项长期的任务。

5.6 LVS+Keepalived 负载均衡

- LVS 简介及工作原理

LVS 是 Linux Virtual Server 的简写，意即 Linux 虚拟服务器，是一个虚拟的服务器集群系统。本项目在 1998 年 5 月由章文嵩博士成立，是中国国内最早出现的自由软件项目之一。

LVS 简单工作原理：用户请求 LVS VIP，LVS 根据转发方式和算法，将请求转发给后端服务器，后端服务器接受到请求，返回给用户。对于用户来说，看不到 WEB 后端具体的应用。

LVS 转发方式有三种，分别是 NAT、DR、TUN 模式，常用算法：RR、LC、WRR、WLC 模式等（RR 为轮询模式，LC 为最少连接模式）

LVS NAT 原理：用户请求 LVS 到达 director,director 将请求的报文的目标地址改成后端的 realserver 地址，同时将报文的目标端口也改成后端选定的 realserver 相应端口，最后将报文发送到 realserver，realserver 将数据返给 director，director 再把数据发送给用户。（两次请求都经过 director，所以访问大的话，director 会成为瓶颈）

LVS DR 原理：用户请求 LVS 到达 director,director 将请求的报文的目标 MAC 地址改成后端的 realserver MAC 地址，目标 IP 为 VIP（不变），源 IP 为用户 IP 地址（保持不变），然后 Director 将报文发送到 realserver，realserver 检测到目标为自己本地 IP，如果在同一个网段，然后将请求直接返给用户。如果用户跟 realserver 不在一个网段，则通过网关

返回用户。（此种转发效率最高）

LVS TUN 原理：跟 LVS DR 类似，也是改变封装 MAC 地址，多了一层隧道加密。实施环境复杂，比 LVS DR 模式效率略低。

- LVS 环境安装配置

下载 LVS 所需软件 ipvsadm-1.2.4.tar.gz 软件，编译安装：

```
wget -c http://www.linuxvirtualserver.org/software/kernel-2.6/ipvsadm-1.24.tar.gz
ln -s /usr/src/kernels/2.6.* /usr/src/linux  //IPVS 模块编译进内核里，需要做软连接
tar xzvf ipvsadm-1.24.tar.gz &&cd ipvsadm-1.24 && make && make install
```

LVS 安装完毕之后，需要进行配置，配置的步骤有两步，第一步为定义端口服务，第二步为添加 realserver 后端服务。

```
ipvsadm -A -t 192.168.149.129:80 -s rr
ipvsadm -a -t 192.168.149.129:80 -r 192.168.149.130 -m -w 2
ipvsadm -a -t 192.168.149.129:80 -r 192.168.149.131 -m -w 2
```

参数说明：

-A 增加一台虚拟服务器地址。

-t 虚拟服务器提供的是 tcp 服务。

-s 使用的调度算法。

-a 在虚拟服务器中增加一台后端真实服务器。

-r　指定真实服务器地址。

-m 设置当前转发方式为 NAT 模式；-g 为直接路由模式；-i　模式为隧道模式。

-w 后端真实服务器的权重。

查看 LVS 转发列表命令为：ipvsadm –Ln

```
[root@node2 ~]#
[root@node2 ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.149.129:80 rr
  -> 192.168.149.131:80           Masq    2      0          0
  -> 192.168.149.130:80           Masq    2      0          0
[root@node2 ~]#
```

我们会发现，如果这台 LVS 发生突发情况，down 机了，那后端所有的应用程序都访问不了。如何避免这种问题呢，这里需要用到故障切换，也就是如果有一台备用的 LVS 就好了，主 down 了，自动切换到从，怎么实现这个需求，接下来讲解的 keepalived 软件就是专门用来做故障检测及切换的。

Keepalived 基于三层检测（IP 层，TCP 层，及应用层），主要用于检测 web 服务器的状态，如果有一台 web 服务器死机，或工作出现故障，Keepalived 检测到并将有故障的 web 服务器从系统中剔除；

当 web 服务器工作正常后 Keepalived 自动将 web 服务器加入到服务器群中，这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的 web 服务器。

需要注意一点，如果使用了 keepalived.conf 配置，就不需要再执行 ipvs –A 命令去添加均衡的 realserver 命令了，所有的配置都会在 keepalived.conf 里面，一个配置文件搞定所有，即只需要安装 ipvs 模

块。

- Keepalived 安装配置

官方下载 keepalived 相应稳定版本：

cd /usr/src ;wget -c http://www.keepalived.org/software/keepalived-1.1.15.tar.gz

tar -xzvf keepalived-1.1.15.tar.gz &&cd keepalived-1.1.15 && ./configure && make && make install

安装完毕，配置 keepalived 服务为系统服务。

DIR=/usr/local/

cp $DIR/etc/rc.d/init.d/keepalived /etc/rc.d/init.d/ && cp $DIR/etc/sysconfig/keepalived /etc/sysconfig/ && mkdir -p /etc/keepalived && cp $DIR/sbin/keepalived /usr/sbin/

在 MASTER 上/etc/keepalived/目录创建 keepalived.conf 配置文件，并写入如下内容：

```
! Configuration Filefor keepalived
global_defs {
 notification_email {
  wgkgood@163.com
 }
 notification_email_from wgkgood@163.com
 smtp_server 127.0.0.1
 smtp_connect_timeout 30
```

```
  router_id LVS_DEVEL
}
# VIP1
vrrp_instance VI_1 {
  state MASTER
  interface eth0
  lvs_sync_daemon_inteface eth0
  virtual_router_id 51
  priority 100
  advert_int 5
  authentication {
    auth_type PASS
    auth_pass 1111
  }
  virtual_ipaddress {
    192.168.149.129
  }
}
#REAL_SERVER_1
virtual_server 192.168.149.129 80 {
  delay_loop 6
  lb_algo wlc
```

```
  lb_kind DR
  persistence_timeout 60
  protocol TCP
  real_server 192.168.149.130 80 {
    weight 100
    TCP_CHECK {
    connect_timeout 10
    nb_get_retry 3
    delay_before_retry 3
    connect_port 80
    }
}
#REAL_SERVER_2
  real_server 192.168.149.131 80 {
    weight 100
    TCP_CHECK {
    connect_timeout 10
    nb_get_retry 3
    delay_before_retry 3
    connect_port 80
    }
  }
```

}

如上配置文件，红色标记的地方需要注意，state 状态主服务器设置 MASTER，从设置为 BACKUP，优先级备机设置比 MASTER 小，例如设置 90，使用 TCP 端口检测。

在 LVS BACKUP 服务器写入如下配置，需要注意的是客户端的配置要修改优先级及状态：

```
! Configuration Filefor keepalived
global_defs {
  notification_email {
   wgkgood@163.com
  }
  notification_email_from wgkgood@163.com
  smtp_server 127.0.0.1
  smtp_connect_timeout 30
  router_id LVS_DEVEL
}
# VIP1
vrrp_instance VI_1 {
  state BACKUP
  interface eth0
  lvs_sync_daemon_inteface eth0
  virtual_router_id 51
```

```
  priority 90
  advert_int 5
  authentication {
    auth_type PASS
    auth_pass 1111
  }
  virtual_ipaddress {
    192.168.149.129
  }
}
#REAL_SERVER_1
virtual_server 192.168.149.129 80 {
  delay_loop 6
  lb_algo wlc
  lb_kind DR
  persistence_timeout 60
  protocol TCP
  real_server 192.168.149.130 80 {
    weight 100
    TCP_CHECK {
    connect_timeout 10
    nb_get_retry 3
```

```
    delay_before_retry 3
    connect_port 80
    }
}
#REAL_SERVER_2
 real_server 192.168.149.131 80 {
    weight 100
    TCP_CHECK {
    connect_timeout 10
    nb_get_retry 3
    delay_before_retry 3
    connect_port 80
    }
 }
}
```

如上设置，LVS 主备配置完毕，接下来需要在 realserver 配置 LVS VIP，为什么要在 realserver 绑定 VIP 呢？

客户端访问 director 的 VIP，director 接收请求，将通过相应的算法将请求转发给相应的 realserver。在转发的过程中，会修改请求包的目的 mac 地址，目的 ip 地址不变。

Realserver 接收请求，并直接响应客户端。这时便出现一个问题，director 此时与 realserver 位于同一个网络中，当 director 直接将请求

转发给 realserver 时，realserver 检测到该请求包的目的 ip 是 vip 而并非自己，便会丢弃，而不会响应。为了解决这个问题，所以需要在所有 Realserver 上都配上 VIP。

为什么一定要配置在 lo 接口上呢？

在 realserver 上的 lo 口配置 VIP,这样限制了 VIP 不会在物理交换机上产生 MAC 地址表，从而避免 IP 冲突。

客户端启动 Realserver.sh 脚本内容：

```
#!/bin/sh
#LVS Client Server
VIP=192.168.149.118
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
```

LVS 网站故障排查经验：

如果发现主网站无法访问，首先第一步 ping 网站域名是否能 ping 通，如果域名无法访问，试着使用 IP 能不能访问，如果 IP 能访问，首先排查到域名解析问题。

如果 IP 也无法访问，登录 LVS 服务器，使用命令 ipvsadm –Ln 查看当前连接状态和查看/var/log/messages 日志信息，可以在 LVS 上访问 realserver ip，进行排查。

如果 LVS 服务正常，后端 realserver 服务异常，然后查看 nginx 日

志信息，是否有大量恶意访问，临时重启看是否能访问。

如果有恶意 ip 访问，找出恶意 ip，经确认可以关闭后，使用 iptables 防火墙临时关闭即可。

5.7 Squid 缓存服务器配置

随着网站访问人数越来越多，对体验的要求也越来越高，网站承受的并发和压力也越来越大，所以需要对网站和架构进行优化，优化的策略有很大，系统内核、程序、配置均衡、加入缓存等，那今天我们来讨论使用 Squid 对架构进行缓存优化。

Squid cache（简称为 Squid）是一个流行的自由软件，它符合 GNU 通用公共许可证。Squid 作为网页服务器的前置 cache 服务器，可以代理用户向 web 服务器请求数据并进行缓存，也可以用在局域网中，使局域网用户通过代理上网。Squid 主要设计用于在 Linux 一类系统运行。

简单的来说就是：用户请求 www 网站，经过 squid，squid 检查本地硬盘目录有没有这个文件的缓存；如果没有，squid 则去后端真实 web 服务器获取该页面，返回给用户，同时在自己本地缓存一份，如果另外一个用户再访问同样请求页面时，squid 直接从本地返回。

squid 有 ufs, aufs, coss, diskd, null 五种存储机制，其中 ufs, aufs, diskd 都是在文件系统上面保存很多小文件， coss 是 squid 自己实现了一个简单的文件系统,可以使用一个大文件或者一个磁盘设备来存储。null 则是给不想要磁盘缓存的情况准备的，coss 看起来好像

很不错，但是以前试验并不足够稳定,因此并不推荐使用。

对于一些老系统,使用 aufs 或者 diskd 是比较好的选择,如果系统的线程库比较好(如 Linux,Solaris),那么使用 aufs。

- 正式安装 squid

安装 squid 也非常简单，可以用源码安装，也可以使用 rpm、yum 安装，这里使用 yum 安装，根据实际经验使用，squid 2.6 系列的 squid 比较稳定，可以考虑采用。

安装命令：yum install -y squid

创建 squid.conf 配置文件，内容如下：

```
http_port 80 accel vhost vport
cache_peer 192.168.149.130 parent 80 0 originserver name=wugk1
cache_peer 192.168.149.131 parent 80 0 originserver name=wugk2
cache_peer_domain wugk1 www.wugk1.com
cache_peer_domain wugk2 www.wugk2.com
visible_hostname localhost
forwarded_for off
via off
cache_vary on
#acl config
acl manager proto cache_object
acl localhost src 127.0.0.1/32
acl to_localhost dst 127.0.0.0/8 0.0.0.0/32
```

```
acl localnet src 10.0.0.0/8     # RFC1918 possible internal network
acl localnet src 172.16.0.0/12  # RFC1918 possible internal network
acl localnet src 192.168.0.0/16 # RFC1918 possible internal network
acl SSL_ports port 443
acl Safe_ports port 80 8080        # http
acl Safe_ports port 21         # ftp
acl Safe_ports port 443        # https
acl all src 0.0.0.0/0
acl CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localnet
http_access allow localhost
http_access allow all
acl PURGE method PURGE
http_access allow PURGE localhost
http_access deny PURGE
#squid config 2014-03-25
cache_dir aufs /data/cache1 10240 16 256
cache_mem 4000 MB
```

maximum_object_size 8 MB

maximum_object_size_in_memory 256 KB

hierarchy_stoplist cgi-bin ?

coredump_dir /var/spool/squid

refresh_pattern ^ftp: 1440 20% 10080

refresh_pattern ^gopher: 1440 0% 1440

refresh_pattern -i (/cgi-bin/|\?) 0 0% 0

refresh_pattern \.(jpg|png|gif|mp3|xml|html|htm|css|js)

1440 50% 2880 ignore-reload

refresh_pattern . 0 20% 4320

- Squid 参数详解

#vhost 和 vport 表示支持虚拟主机和虚拟端口，如果再加上 transparent 表示支持透明代理

http_port 80 accel vhost vport

#cache_peer 表示如果本机缓存中找不到客户端请求的数据，则与后端主机联系，以 parent 类型进行联系；

使用 HTTP 协议进行联系，联系端口是 80，orginserver 表示此服务器是源服务器，name 表示别名。

cache_peer 192.168.149.128 parent 80 0 originserver name=wugk1

cache_peer 192.168.149.129 parent 80 0 originserver name=wugk2

#设置别名所对应的域名,如果 cache_peer 中使用域名而不是 IP 的话；那么 cache_peer_domain 中一定要用相同的域名，否则无法访问。

```
cache_peer_domain wugk1 www.wugk1.com

cache_peer_domain wugk2 www.wugk2.com

#设置缓存服务器名称

visible_hostname localhost

forwarded_for off

via off

cache_vary on

#acl config

acl manager proto cache_object

acl localhost src 127.0.0.1/32

acl to_localhost dst 127.0.0.0/8 0.0.0.0/32

acl localnet src 10.0.0.0/8         # RFC1918 possible internal network

acl localnet src 172.16.0.0/12    # RFC1918 possible internal network

acl localnet src 192.168.0.0/16 # RFC1918 possible internal network

acl SSL_ports port 443

acl Safe_ports port 80 8080              # http

acl Safe_ports port 21                 # ftp

acl Safe_ports port 443                # https

acl all src 0.0.0.0/0

acl CONNECT method CONNECT

http_access allow manager localhost

http_access deny manager
```

```
http_access deny !Safe_ports

http_access deny CONNECT !SSL_ports

http_access allow localnet

http_access allow localhost

##设置访问控制,允许所有客户端访问上面设置的两个网站

http_access allow all

#支持 purge 方式清除缓存

acl PURGE method PURGE

http_access allow PURGE localhost

http_access deny PURGE

#squid config 2014-03-25

#设置缓存文件夹的路径和参数，缓存机制为 aufs，10240 表示 10G，

目录下面分为 16 级，每级有 256 个目录

cache_dir aufs /data/cache1 10240 16 256

#设置缓存内存大小，最大内存为 4g

cache_mem 4000 MB

#设置硬盘中可缓存的最大文件大小

maximum_object_size 8 MB

#设置内存中可缓存的最大文件大小

maximum_object_size_in_memory 256 KB

hierarchy_stoplist cgi-bin ?
```

#当 squid 突然挂掉的时候，或者突然出现什么故障的时候，将 squid 在内存中的资料写到硬盘中

coredump_dir /var/spool/squid

#<refresh_pattern> <regex> <最小时间> <百分比> <最大时间>

#refresh_pattern 用于确定缓存的类型,缓存过期时间,及百分比。

#如果希望内容缓存 cache 后不删除，直到被主动用 purge 清除，可以加 ignore-reload 选项

refresh_pattern ^ftp: 1440 20% 10080

refresh_pattern ^gopher: 1440 0% 1440

refresh_pattern -i (/cgi-bin/|\?) 0 0% 0

refresh_pattern \.(jpg|png|gif|mp3|xml|html|htm|css|js) 1440 50% 2880 ignore-reload

refresh_pattern . 0 20% 4320

5.8 MySQL 高可用架构

MySQL 是一个开放源码的小型关联式数据库管理系统，开发者为瑞典 MySQL AB 公司，目前属于 Oracle 公司,MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。

对应目前主流的 LAMP 架构来说,Mysql 更是得到各位 IT 运维、DBA 的青睐，目前 mysql 已被 orcacle 收购，不过好消息是原来 mysql 创始

人已独立出来自己重新开发了一个 MariaDB，而且使用的人数越来越多。而且 MariaDB 兼容 mysql 所有的功能和相关参数。

Mysql 常用的两大引擎有 MyISAM 和 innoDB，那他们有什么明显的区别呢，什么场合使用什么引擎呢？

MyISAM 类型的表强调的是性能，其执行数 度比 InnoDB 类型更快，但不提供事务支持，如果执行大量的 SELECT 操作，MyISAM 是更好的选择，支持表锁。

InnoDB 提供事务支持事务，外部键等高级 数据库功能，执行大量的 INSERT 或 UPDATE，出于性能方面的考虑，应该使用 InnoDB 表，支持行锁。

随着访问量的不断增加，Mysql 数据库压力不断增加，需要对 mysql 进行优化和架构改造，可以使用高可用、主从复制、读写分离来、拆分库、拆分表进行优化。下面我们来学习 MySQL 主从复制高可用如何来实现。

- MySQL 数据库主从复制原理

  Mysql 主从同步其实是一个异步复制的过程，要实现复制首先需要在 master 上开启 bin-log 日志功能，整个过程需要开启 3 个线程，分别是 Master 开启 IO 线程，slave 开启 IO 线程和 SQL 线程。

a) 在从服务器执行 slave start，从服务器上 IO 线程会通过授权的用户连接上 master，并请求 master 从指定的文件和位置之后发送 bin-log 日志内容。

b) Master 服务器接收到来自 slave 服务器的 IO 线程的请求后，master

服务器上的 IO 线程根据 slave 服务器发送的指定 bin-log 日志之后的内容，然后返回给 slave 端的 IO 线程。（返回的信息中除了 bin-log 日志内容外，还有本次返回日志内容后在 master 服务器端的新的 binlog 文件名以及在 binlog 中的下一个指定更新位置。）

c) Slave 的 IO 进程接收到信息后，将接收到的日志内容依次添加到 Slave 端的 relay-log 文件的最末端，并将读取到的 Master 端的 bin-log 的文件名和位置记录到 master-info 文件中，以便在下一次读取的时候能够清楚的告诉 Master“我需要从某个 bin-log 的哪 个位置开始往后的日志内容，请发给我”；

d) Slave 的 Sql 进程检测到 relay-log 中新增加了内容后，会马上解析 relay-log 的内容成为在 Master 端真实执行时候的那些可执行的内容，并在自身执行。

- MySQL 数据库主从配置

环境准备：192.168.149.128 为 master 主服务器，192.168.149.129 为 slave 从服务器。

在主和从服务器都安装 mysql 相关软件，命令如下：

yum install -y mysql mysql-devel mysql-server mysql-libs

安装完毕后，在 Master 修改 vi /etc/my.cnf 内容为如下：

```
[mysqld]
datadir=/data/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin
```

server-id = 1
auto_increment_offset=1
auto_increment_increment=2
[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
replicate-do-db =all

创建/data/mysql 数据目录， mkdir –p /data/mysql ;chown –R mysqld:mysqld /data/mysql

启动 mysql 即可，/etc/init.d/mysqld restart

然后修改 slave Mysql 数据库 my.cnf 配置文件内容如下：

[mysqld]
datadir=/data/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-bin=mysql-bin
server-id = 2
auto_increment_offset=2
auto_increment_increment=2
[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
master-host =192.168.149.128
master-user=tongbu
master-pass=123456
master-port =3306
master-connect-retry=60
replicate-do-db =all

在 Master 数据库服务器上设置权限，执行如下命令：

grant replication slave on *.* to 'tongbu'@'%' identified by '123456';

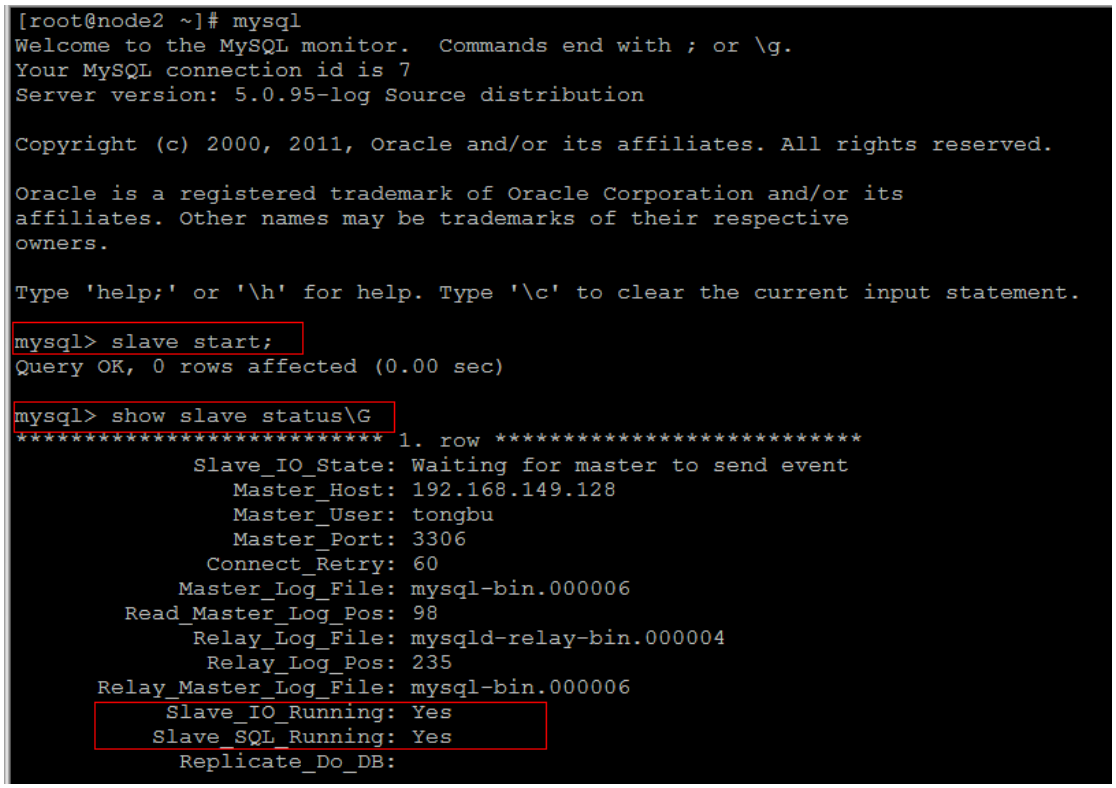
在 Master 数据库执行如下命令：

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000006 |       98 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

然后在 slave 服务器指定 master IP 和同步的 pos 点：

```
change master to
master_host='192.168.149.128',master_user='tongbu',master_passw
ord='123456',master_log_file='mysql-bin.000006',master_log_pos=9
8;
```

在 slave 启动 slave start，并执行 show slave status\G 查看 Mysql 主从状态：

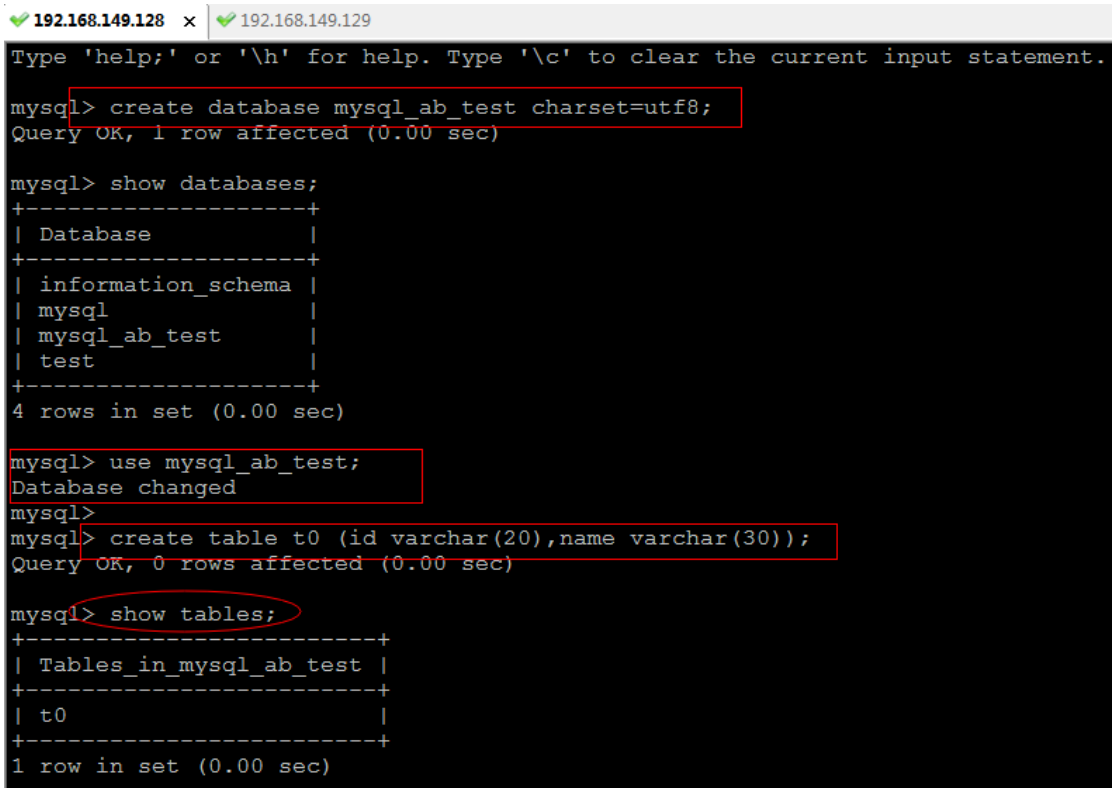


Slave_IO_Running: Yes

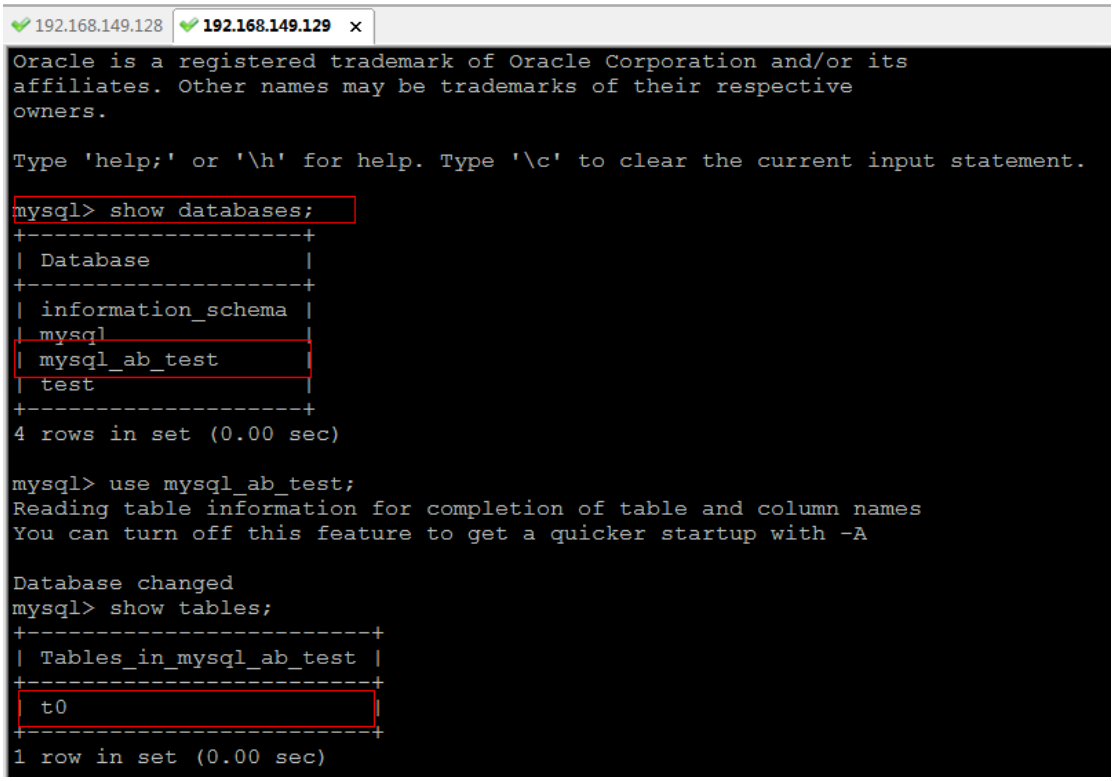
Slave_SQL_Running: Yes 两个状态为 YES，代表 slave 已经启动两个

线程，一个为 IO 线程，一个为 SQL 线程。

然后在 Master 服务器创建一个数据库和表，命令如下：

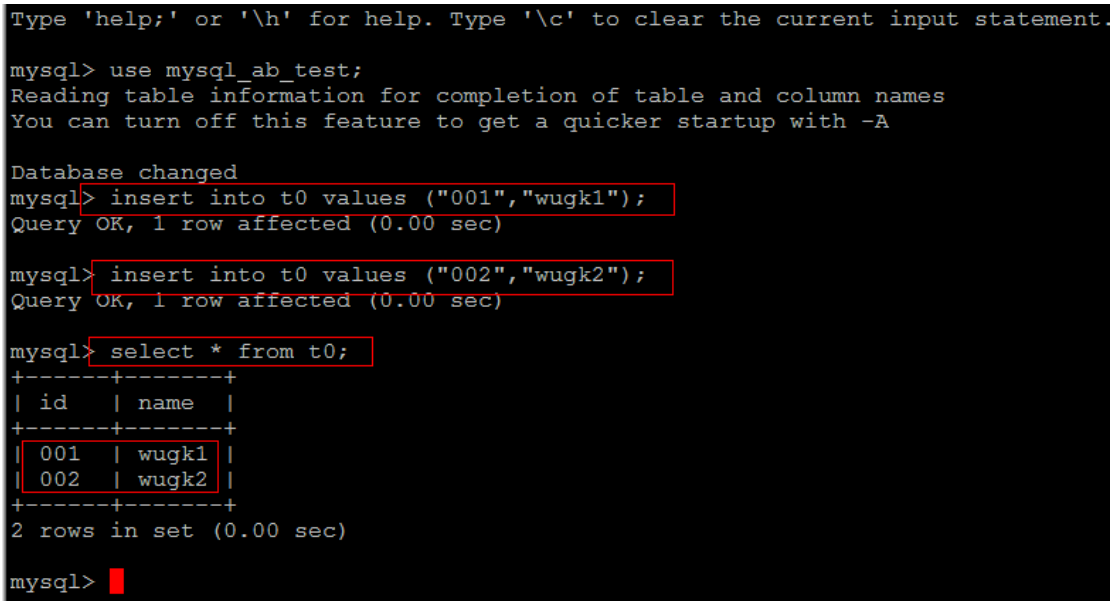


然后去 slave 服务器查看是否有 mysql_ab_test 数据库和相应 t0 的表，如果存在则代表 Mysql 主从同步搭建成功：

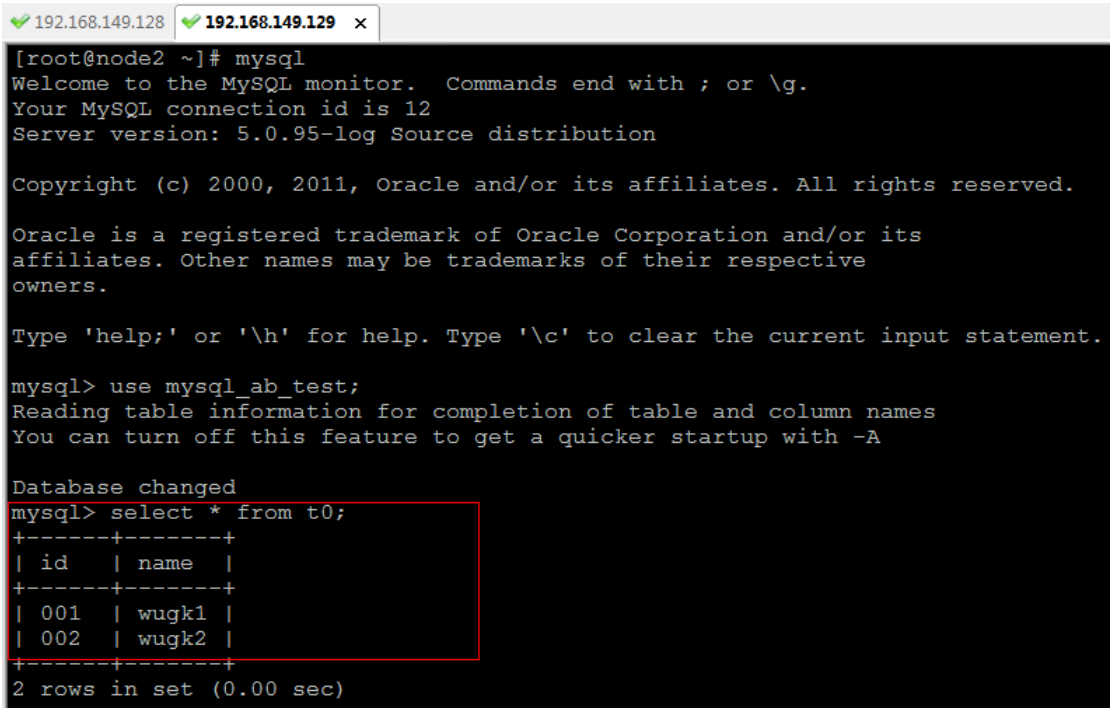


同样还可以测试在 master 服务器插入两条数据，在 slave 查看 insert 数据是否已同步：

128 master 上执行如下图：



129 slave 上执行如下图，在 master 插入的数据已经同步到 slave 上：



自此 Mysql 主从搭建完毕，现在有一个问题，如果 master 服务器 down 机了，如何快速恢复服务呢？

可以通过两种方法：

第一种方法，如果程序连接的是 master 的 IP，直接在 slave 服务器上添加 master 的 IP 即可。这个手动去操作，而且需要花费时间比较长，可能还会出现误操作的情况，不推荐。

第二种方法，可以使用 keepalived、heartbeat 作为 HA 检测软件，检查 MySQL 服务是否正常，不正常则自动切换到 slave 上，推荐使用。

- Mysql+keepalived 高可用配置

继上一章节 MySQL 主从配置完毕后，接着配置 keepalived 服务，主要用于 Mysql 故障自动切换。

Keepalived 安装配置：

tar zxf keepalived-1.2.1.tar.gz

cd keepalived-1.2.1 &&./configure

--with-kernel-dir=/usr/src/kernels/2.6.18-164.el5-i686 &&make && make install

DIR=/usr/local/ ;cp $DIR/etc/rc.d/init.d/keepalived /etc/rc.d/init.d/

cp $DIR/etc/sysconfig/keepalived /etc/sysconfig/ && mkdir -p /etc/keepalived

cp $DIR/sbin/keepalived /usr/sbin/

修改 Master 服务器上 keepalived.conf 配置如下， vi /etc/keepalived/keepalived.conf

```
! Configuration File for keepalived
global_defs {
    notification_email {
        wgkgood@139.com
    }
    notification_email_from wgkgood@139.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}
# VIP1
vrrp_instance VI_1 {
     state BACKUP
     interface eth0
```

```
        lvs_sync_daemon_inteface eth0
        virtual_router_id 151
        priority 100
        advert_int 5
        nopreempt
        authentication {
            auth_type PASS
            auth_pass 2222
        }
        virtual_ipaddress {
            192.168.149.100
        }
}
virtual_server 192.168.149.100 3306 {
        delay_loop 6
        lb_algo wrr
        lb_kind DR
        persistence_timeout 60
        protocol TCP
        real_server 192.168.149.128 3306 {
            weight 100
            notify_down /data/sh/mysql.sh
```

```
        TCP_CHECK {
        connect_timeout 10
        nb_get_retry 3
        delay_before_retry 3
        connect_port 3306
        }
    }
}
```

Mysql 从服务器配置 keepalived.conf 跟 master 一样，只需要把 Realserver IP 修改成 real_server 192.168.149.129 ；优先级从 100 改成 90 即可。

在 master、slave 数据库上创建/data/sh/mysql.sh 脚本，内容为：

```
pkill    keepalived
```

然后分别重启两台数据库上 keepalived 服务即可。最后测试停止 master Mysql 服务，是否会自动切换到 Backup 上。

关于 Mysql 集群高可用就在此告一段落，当然除了 keepalived 高可用之外，Mysql 优化还可以进行读写分离、Mysql+DRBD、拆分表等等优化，有兴趣的童鞋可以继续深入研究。

6. Linux 运维职业规划

对于从事 Linux 岗位的童鞋们,最关注的问题莫过于这个行业到底怎么样,能不能挣钱?我以后能做什么?

对于第一个问题:

随着互联网飞速的发展,用户对网站体验各方面都要求很高,所以作为网站底层承载的 linux 系统来说,得到大批量的应用,可以说大中型互联网公司 Linux 在服务器领域已经占到 7-80%,而且 Android 手机也是基于 Linux 来研发定制的。未来 Linux 会在各行各业得到普遍的应用。

这里讨论 Linux 运维，如果是 Linux 开发的话，薪资更高，所以只要你技术熟练、精通，薪资根本不是问题，初级薪资一般都在 4-5K 以上，中间 6-8K，高级 Linux 运维薪资一般都在 10K+。对应 Linux 岗位管理方面薪资则 20K+左右。所有不要再问能给有多少钱，关键是你有多熟练，你的能力在哪里？

从 Linux 运维领域来说，可以努力学习的方向有：

1) 熟练 Linux 系统的性能优化、网络日常管理。
2) 高性能集群架构部署及优化等。
3) 大并发网站运维及管理。
4) Mysql、Oracle 数据库集群管理。
5) 自动化运维平台开发与管理
6) 网站架构 GSLB、CDN 缓存等。

一个行业要想熟练、甚至精通至少要花上 5-10 年的时间，做一件事重在专一,即使现在不会,只要每天进步一点点,每天实践一点点，改变一点点，相信未来更美好。只有专注才能成功。

7. Linux 运维面试总结

8.1. 1 面试技巧总结

通过全面具体的学习，我们已经正式遨游进入了 Linux 运维世界，接下来我们就需要正式的找一份 Linux 岗位的工作，很多人谈到找工作就害怕，为什么呢，害怕面试不上、面试紧张、知识准备不充分等等。

通过这样一个完整的 Linux 高级运维的学习，我们了解了目前企业里面使用的技术和架构信息，那接下来我们来总结一下企业一般问什么问题？以及面试的过程中要注意哪些细节？

总结日常面试的技巧（以正式讲课为准）：

1) 首先穿着要得体，最好标准的职业装面试，不能随意穿着；简单一点就是要让人一看你，就感觉清爽、能干、有活力。
2) 要准备充分，尽量提前 15 分钟到面试公司，提前翻阅资料了解公司的简单背景及相关文化。
3) 保持微笑，不要太古板，要随和，保持心态放松，不要抢话抢答；要懂礼貌，有时候细节决定成败。
4) 在回答问题上要简单明了，不要阐述一个问题绕来绕去，把自己都绕迷糊了；要说到恰到好处。该回答的回答，不该说尽量别说，做到有的放矢。
5) 要保持谦逊，遇到不会的题目，不知道就是不知道，不要非懂装懂；

6) 面试通常开始会让你做自我介绍，自我介绍说些什么？多长合适？

   一般自我介绍，就介绍自己叫什么名字，毕业时间学校，已经之前工作经验，自己比较熟练的技能和自己的性格和优点等等；注介绍完毕，最后说声介绍完毕，谢谢。

7) 面试要有自信，不要低着头，面试是双向的，你选择公司，公司也在选择你。机会是非常的多的，关键是看你自己是否能把握住，是否之前已经准备好。

8) 面试的心态一定要保持平静，不要因为一次面试不上，就觉得自己到处都是缺点，要总结自己上次面试的不足，然后下一次改变掉，相信坚持不懈一定能找到满意的工作。

9) 最后总结一点，做什么事情自信很重要，相信自己可以做到，然后勇敢的去做，结果一定让你倍感惊喜。

8.1.2 面试题目总结

通过不断的面试，我们会总结到更多的知识和技巧，这里总结一下日常面试到的问题及简单回答方法：

1) 你平时在公司主要做什么？

2) 你们原来公司的网站架构是怎样的？

3) 你对哪一块比较熟练或者精通？

4) Squid、varnish 等缓存服务器维护过吗？squid 缓存代理的原理是什么？缓存命中率怎么查看及清空缓存？

5) LVS 的工作原理是什么？有哪些算法？

6) Nginx 日常的优化的参数都有哪些？Nginx 动静分离做过吗？描述简单的步骤。

7) Linux 内核优化，你都优化哪些参数？

8) 你在维护网站的过程中，曾经遇到过什么重大的问题？怎么解决的？

9) Shell 编程熟练吗？编写一个自动化备份 Mysql 数据库的脚本？

10) Mysql 主从架构的原理是什么？如果主从不同步，报错了，怎么恢复？

11) 如果备份大数据 Mysql 数据文件？Mysql 优化有哪些步骤？

12) FTP 主被动模式的区别是什么？

13) Apache 两种工作模式的区别及优化？

14) Nagios、cacti 维护过吗？平时都监控些什么？

15) 你们公司的网络出口带宽是多少？每天网站的 PV、UV 是多少？

16) 你觉得 Linux 运维工程师的职责是什么？

17) 你为什么离职，离职的原因是什么？

18) 你未来 5-10 年的职业规划是什么样的？