

全栈最后一公里 -- Linux从入门到实践



傻瓜都能写出计算机能理解的程序，而好的程序员能写出人能读懂的代码。

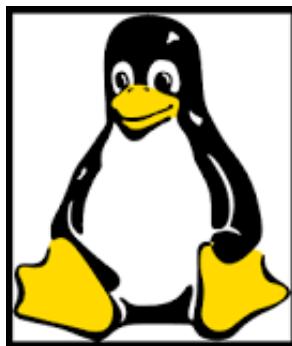
- [全栈最后一公里 —— Linux从入门到实践](#)
- [入门篇](#)
 - [二、前世今生](#)
 - [1、系统来源](#)
 - [2、系统版本](#)
 - [Linux内核版本：](#)
 - [Linux发行版本：](#)
 - [二、Linux系统安装](#)
 - [1、虚拟机安装](#)
 - [2、系统分区](#)
 - [3、Linux系统安装](#)
 - [3.1 创建虚拟机](#)
 - [3.2 安装Linux](#)
 - [三、Linux 序言](#)
 - [四、Linux 网络管理](#)
 - [1、网络模型与作用](#)
 - [1.1 OSI七层网络模型](#)
 - [1.2 TCP/IP四层网络模型](#)
 - [1.3 关于IP地址](#)
 - [1.4 关于子网掩码](#)
 - [1.5 关于端口](#)

- [1.6 关于DNS](#)
- [1.7 关于网关\(路由器\)](#)
- [2、Linux网卡与配置](#)
 - [2.1 网卡信息文件](#)
 - [2.2 主机名文件](#)
 - [2.3 DNS配置文件](#)
- [3、重启网络服务](#)
- [五、Linux 用户管理](#)
 - [1、用户与用户组](#)
 - [1.1 用户组信息文件](#)
 - [1.2 用户组的密码信息文件](#)
 - [1.3 用户信息文件](#)
 - [1.4 用户密码信息文件](#)
 - [2、基本命令](#)
- [六、Linux 权限管理](#)
 - [1、文件权限的修改](#)
 - [2、文件权限的作用](#)
 - [3、文件所属的修改](#)
 - [4、文件的默认权限](#)
- [七、Linux 软件安装管理](#)
 - [1、挂载](#)
 - [2、压缩和解压缩](#)
 - [3、软件包安装](#)
 - [3.1 RPM包手动安装](#)
 - [3.2 yum在线安装](#)
 - [yum源 配置文件](#)
 - [光盘搭建本地yum源](#)
 - [通过yum安装软件](#)
 - [3.3 源码包安装](#)
- [八、Linux 防火墙\(iptables\)](#)
 - [1、关于防火墙](#)
 - [2、关于iptables](#)
 - [3、iptables规则组成](#)
- [实践篇](#)
 - [一、FTP服务器\(vsftpd\)](#)
 - [1、关于vsftpd](#)
 - [2、安装并启动FTP服务](#)
 - [3、配置FTP权限](#)
 - [4、访问FTP服务](#)
 - [二、Web服务器\(Tomcat\)](#)

- [1、JDK下载安装](#)
- [2、Tomcat下载安装](#)
- [三、数据库 MySQL](#)
 - [1、MySQL安装与启动](#)
 - [2、MySQL 密码、用户、权限](#)
- [四、反向代理与负载均衡服务器 \(NGINX\)](#)
 - [1、NGINX 下载安装](#)
 - [2、NGINX的启动与常用命令](#)
 - [3、NGINX 反向代理](#)
 - [4、NGINX 负载均衡](#)
- [五、Java 用户系统项目实战](#)

入门篇

一、前世今生



1、系统来源

在Linux之前有一个叫 *Andrew S. Tanenbaum* 的教授为了给他的学生上课买了一个UNIX操作系统源码，然后参考这个系统但是没有任何抄袭成分的做了一个叫作Minix的操作系统，后来他把这个Minix操作系统的源码全部开放给学校和学生做学习研究使用，再后来经过一番优化之后他把这个系统就直接放到网络上去开源了，当时就一下受到很多人的关注，开源免费的操作系统非常受欢迎，很多人就开始使用，但是用着用着就有人发现这个系统有些bug，于是就向这个教授提了很多修复补丁程序希望教授可以更新修复，但是呢这个教授不这么想，他说：“我写这个代码就只是为了教学，你们能看得懂我的目的就达到了，我不需要任何外来的代码也没有想过把它商业发行。”

那么这个时候另一个家伙出现了，*Linus Torvalds*，当时是芬兰赫尔辛基大学大三的大学生，他以Minix为模板，自己开发了一些软件和并且集中了当时网上呼声较高的一些补丁程序重新写了一个操作系统——Linux操作系统。



Linus选择了芬兰的吉祥物企鹅作为该系统的标识，并在1991年的时候就将Linux操作系统发布了，短短20多年的时间，Linux系统就已经在服务器领域占据了举足轻重的地位。

2、系统版本

Linux经过20多年的发展已经拥有了非常众多的版本，但是我们需要知道的是这些版本中分为Linux内核版本和Linux发行版本，这两个版本不太一样。

Linux内核版本：

Linux内核版本就是核心版本，它由 [Linux内核官网](#) 发行，目前最新的内核版本是4.19版本。

The screenshot shows the homepage of The Linux Kernel Archives at kernel.org. The main title is "The Linux Kernel Archives". A yellow button in the center says "Latest Stable Kernel: 4.19" with a download icon. Below it is a table of kernel versions and their details. To the right is a "Social" sidebar with links to Atom feeds and Kernel Planet. At the bottom left is a "Other resources" sidebar with links to Cgit, Bugzilla, Mirrors, Documentation, Patchwork, Linux.com, Wikis, Kernel Mailing Lists, and Linux Foundation.

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel: 4.19

mainline:	4.19	2018-10-22	[tarball]	[pgp]	[patch]	[view diff]	[browse]		
stable:	4.18.16	2018-10-20	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.14.78	2018-10-20	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.9.135	2018-10-20	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.4.162	2018-10-20	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.18.124 [EOL]	2018-10-13	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.16.60	2018-10-21	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20181019	2018-10-19							[browse]

Other resources

Cgit
Bugzilla
Mirrors

Documentation
Patchwork
Linux.com

Wikis
Kernel Mailing Lists
Linux Foundation

Social

Site Atom feed
Releases Atom Feed
Kernel Planet

值得一提的是：对于服务器领域而言，一般不会采用最新版本，线上我们要本着“猥琐发育，不要浪”的原则！

Linux发行版本：

内核版本是由官网发布，任何人都可以去免费下载使用的，但是比如基于内核版本之上开发一些工具，开发一个桌面，放点好看的图标，或者做一些裁剪等，那么它就会变成各个厂商的发行版本。



在上面众多版本中，比较流行的主要还是Redhat和Ubuntu，其中Ubuntu的优势主要在于界面优化的非常漂亮，有的时候都认不出来这是一个Linux系统，但是这些花里胡哨的东西并不适合服务器使用，上面我们说过服务器一定要稳定优先，所以一般不会在服务器上去安装图像界面，所以下面本教程也会以Redhat系列来进行，只不过有一点需要说的是Redhat部分功能是收费的，所以我们会以Redhat的替身CentOS操作系统为例进行讲解。

二、Linux系统安装

在开始学习Linux系统之前我们迫切需要知道如何把Linux系统给安装上，但是呢作为一个初学者，如果直接在自己的电脑上就装一个Linux操作系统，那我估计你可能痛苦的坚持不了3天就要重装系统了，所以就引入了下面的虚拟机相关知识。

1、虚拟机安装

在Mac上（PS：下述所有内容如无特殊说明都是基于Mac进行）常用的虚拟机就两种
[VMware](#) 和 [Parallels Desktop](#)，相同点是这两个都收费，而且价格不菲，不同点是我这里有可以使用的“正版VMware”，下载链

接：https://pan.baidu.com/s/1dtnZ1V_evPFbJCrER2I9mg 密码:6o3p 关于如何在Mac上把VMware安装上就不作过多赘述了，与安装其他软件没有什么区别，一路Next然后完成之后使用天朝独特的Keygen去“优化（破解）”一下就可以使用了。



VMware安装完成之后打开应该是上面这样的，这里我们主要介绍三个功能，首先是左边一半重点便捷的“从光盘或映像安装”，这个功能没什么好说，给一个iso镜像文件然后就一路向西的安装完成了，其次是“导入现有虚拟机”，如果你已经有安装好的虚拟机文件(ps:使用虚拟机安装系统的时候会生成一个文件保存安装的系统数据信息)那么可以直接通过这个功能加载出来，最后是“创建自定义虚拟机”，这个功能就相当于你去电脑市场买了一个只有boot系统的电脑，里面并没有操作系统，需要你手动安装。我们下面安装Linux系统的时候也使用该功能。

2、系统分区

在进行Linux系统安装之前我们先要搞清楚一个问题，那就是系统分区。下面简单介绍一下系统分区的类型：

- 主分区：由硬盘硬件决定了主分区最多只能有4个
- 扩展分区：扩展分区最多只能有一个，并且主分区加扩展分区最多只能有4个，另外扩展分区不能写入数据，只能包含逻辑分区
- 逻辑分区

我们都知道在Windows中硬盘分区的时候需要指定盘符，例如：系统分区用字母C作为盘符，其他的用D、E、F以此类推，但是在Linux中我们把这个盘符称为“**挂载点**”，把一个硬盘分区让Linux系统读取到的过程称作“**挂载**”。关于Linux系统分区的要求如下：

- 必须分区：
 1. / (根分区)
 2. swap分区（交换分区，一般物理机内存4GB以下时建议为内存的2倍，物理机内存超过4GB时建议和物理机内存大小一致即可）
- 推荐分区：
 1. /boot (启动分区, 200MB)

那么是不是说一块硬盘分区之后就使用了呢？不是的！做好分区之后还需要做另外一件事情，格式化。

格式化：又称逻辑格式化，它是指根据用户选定的文件系统（如FAT32、NTFS、EXT4等），在磁盘的特定区域写入特定的数据，从而在分区中划出一块用于存放文件分配表、目录表等用于文件管理的磁盘空间的过程；

注意：这里有一个误区，很多人认为格式化是用来清空分区里的数据的，其实并不是这样，虽然格式化会清空分区里的数据，但是格式化的根本目的是为了写入文件系统。

3、Linux系统安装

安装好虚拟机并且了解了系统分区知识后我们才可以进行科学的Linux系统安装。

3.1 创建虚拟机

在前面我们已经安装好了VMware，启动它，选择“**创建自定义虚拟机**”，点击“**继续**”，之后看到如下画面：



为此虚拟机选择操作系统:

Microsoft Windows	▶ Asianux 4
Apple OS X	▶ Asianux 4 64 位
Linux	▶ Asianux Server 3
Novell NetWare	▶ Asianux Server 3 64 位
Solaris	▶ CentOS
VMware ESX	▶ CentOS 64 位
其他	▶ Debian 8.x Debian 8.x 64 位 Debian 7.x Debian 7.x 64 位 Debian 6 Debian 6 64 位 Debian 5 Debian 5 64 位



然后选择“Linux”->“CentOS 64位”点击“继续”，画面如下：



选择“新建虚拟磁盘”，点击“继续”，就可以完成虚拟机的创建了，出现如下画面，点击“完成”，选择好虚拟机文件存放路径，即可完成虚拟机的创建。



3.2 安装Linux

和Windows或者macOS操作系统一样，安装都需要一个系统镜像来完成安装，我们可以从[CentOS官网](#)下载各种版本的系统镜像，本教程使用的系统镜像是[CentOS 6.9 64位](#)

下载好系统镜像之后，打开刚安装好的虚拟机，点击“设置”按钮，打开设置面板：



选择“CD/DVD(IDE)”：



勾选“连接 CD/DVD 驱动器”，然后下拉选择“选择一个光盘或光盘映像...”，选择我们下载好

的系统镜像：

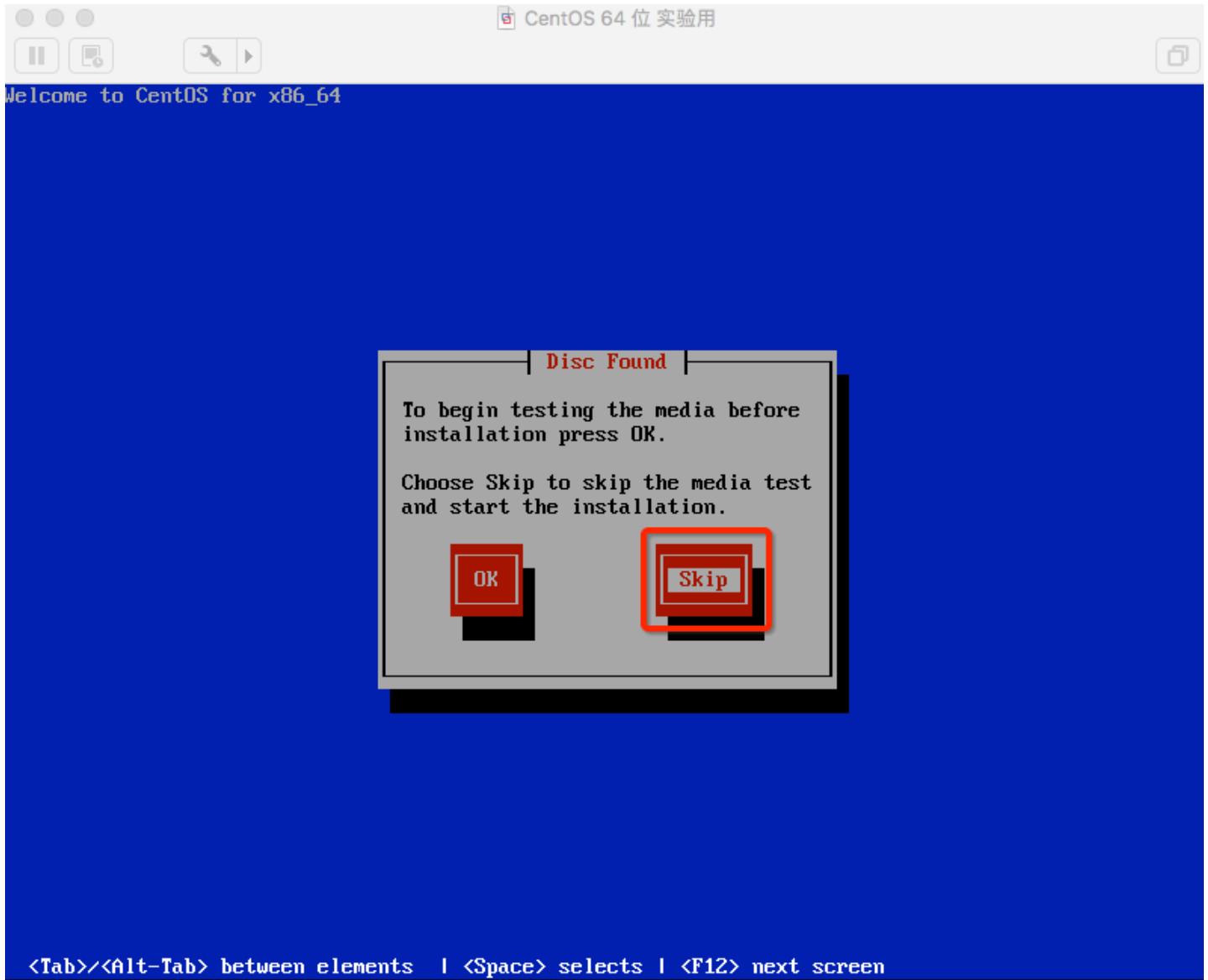


这就跟我们平时安装电脑在光驱里插入系统光盘一样，下面我们就就可以启动虚拟机进行Linux系统安装了。

启动虚拟机，虚拟机会自动读取光驱里的系统镜像，从而启动Linux系统的安装程序，如下图：



默认选择第一项即可，直接回车，会出现这样一个镜像数据完整性检测界面，简单的说就是用来检测你下载的这个系统镜像是不是可以使用，这里我们一般都认为是可用的不再进行检测，所以直接选择“Skip”：



之后就很简单了，一路“Next”就行：



CentOS 64 位 实验用

要释放鼠标, 请按: Control-Alt



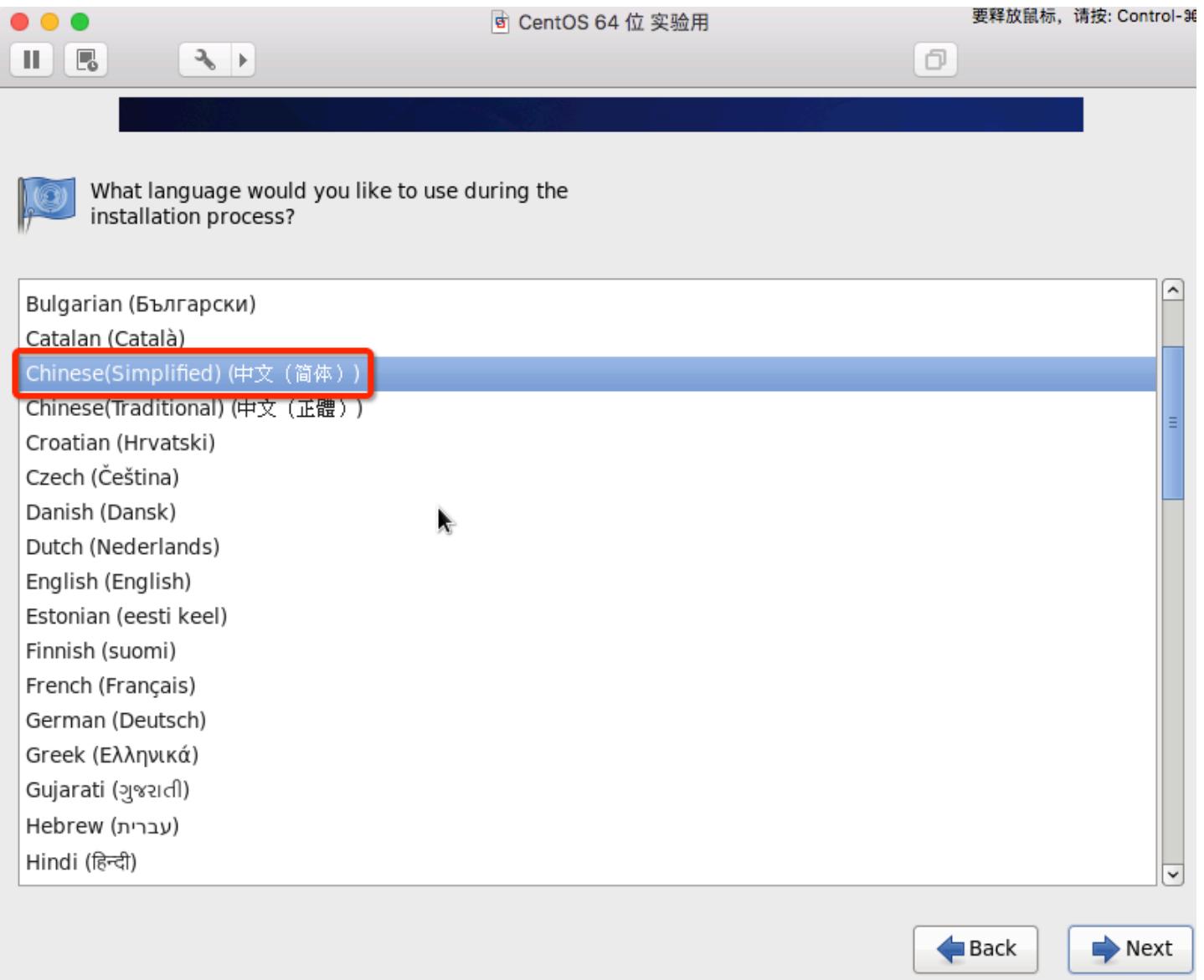
CentOS 6

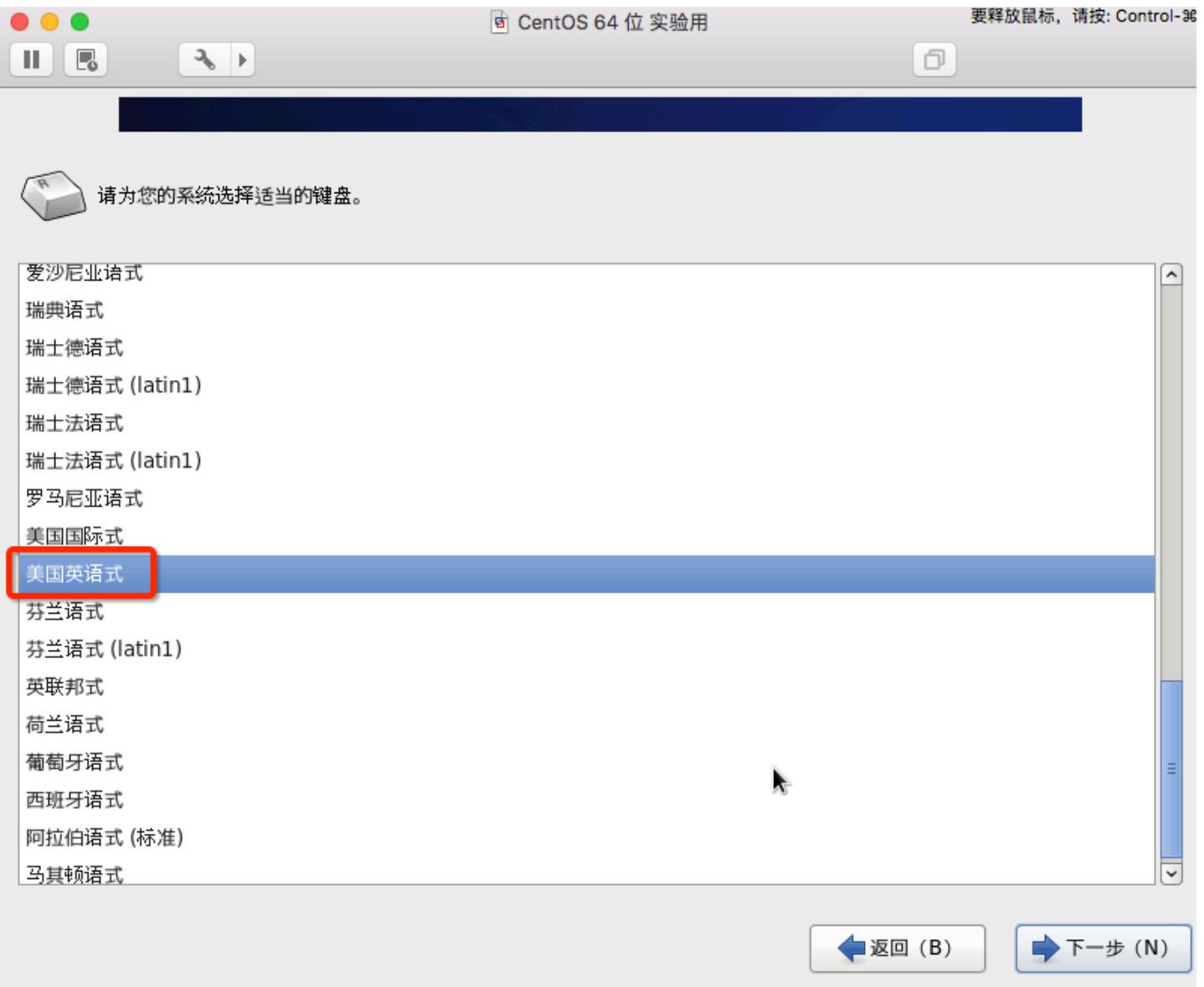
Community ENTerprise Operating System

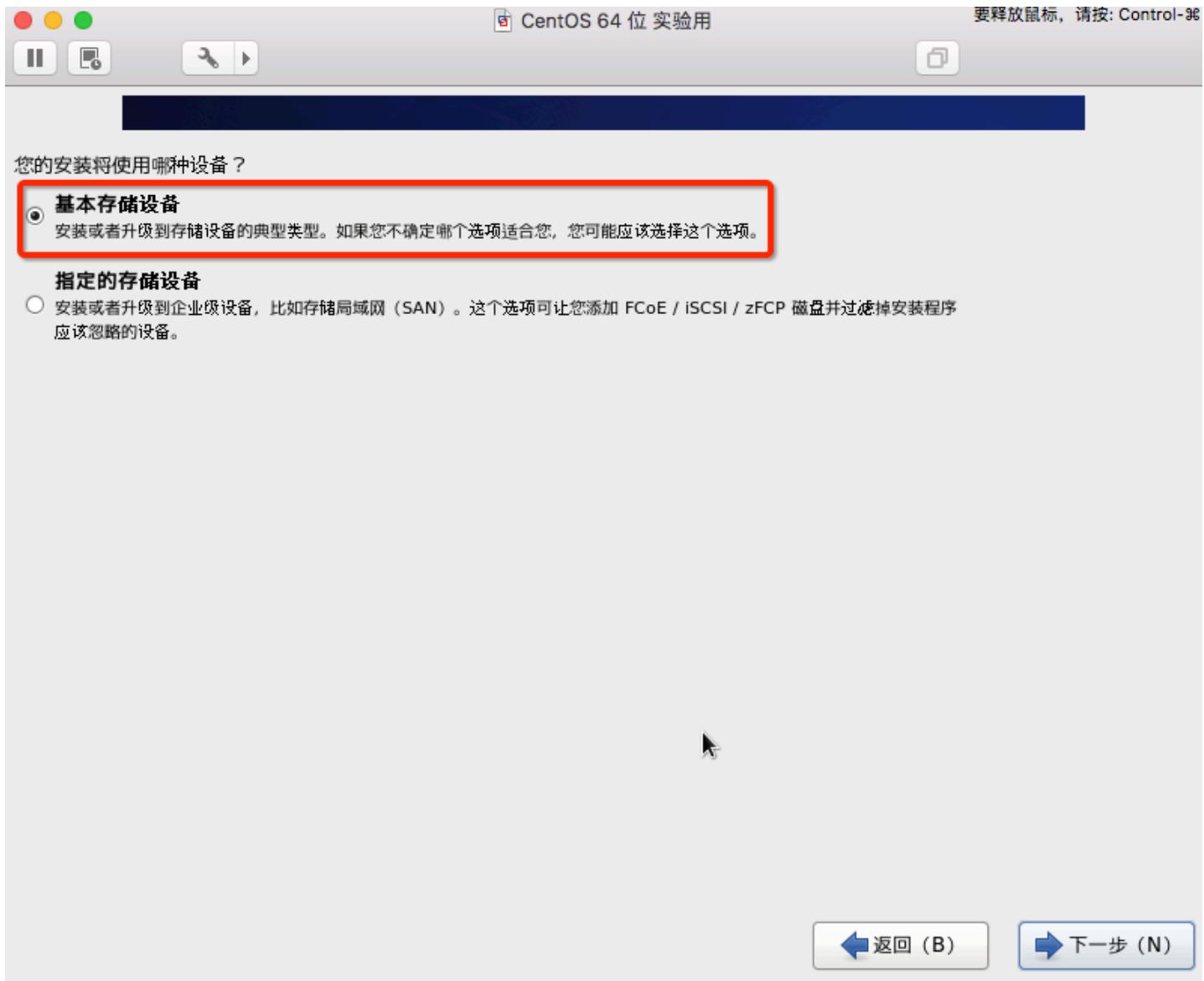


Back

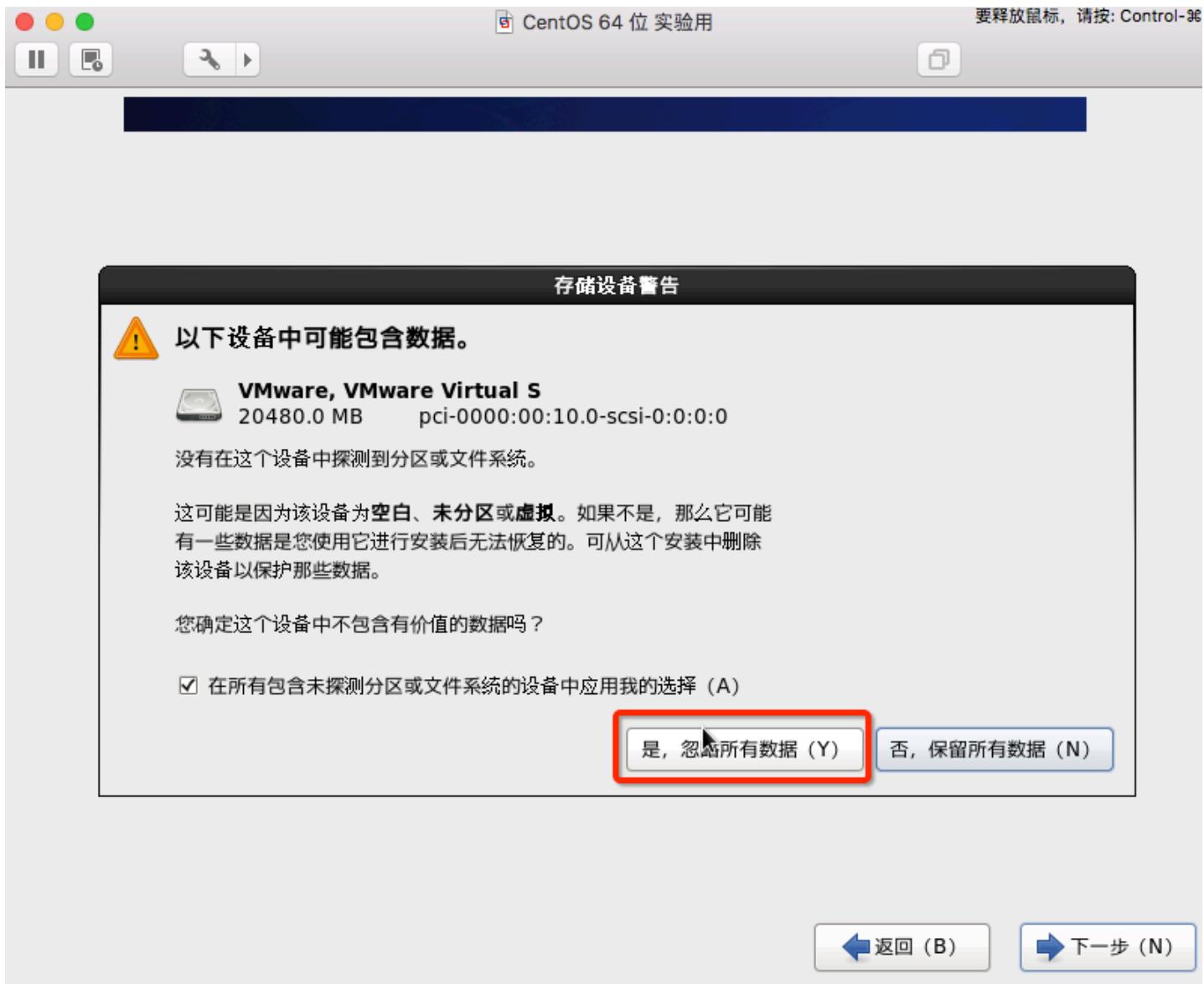
Next





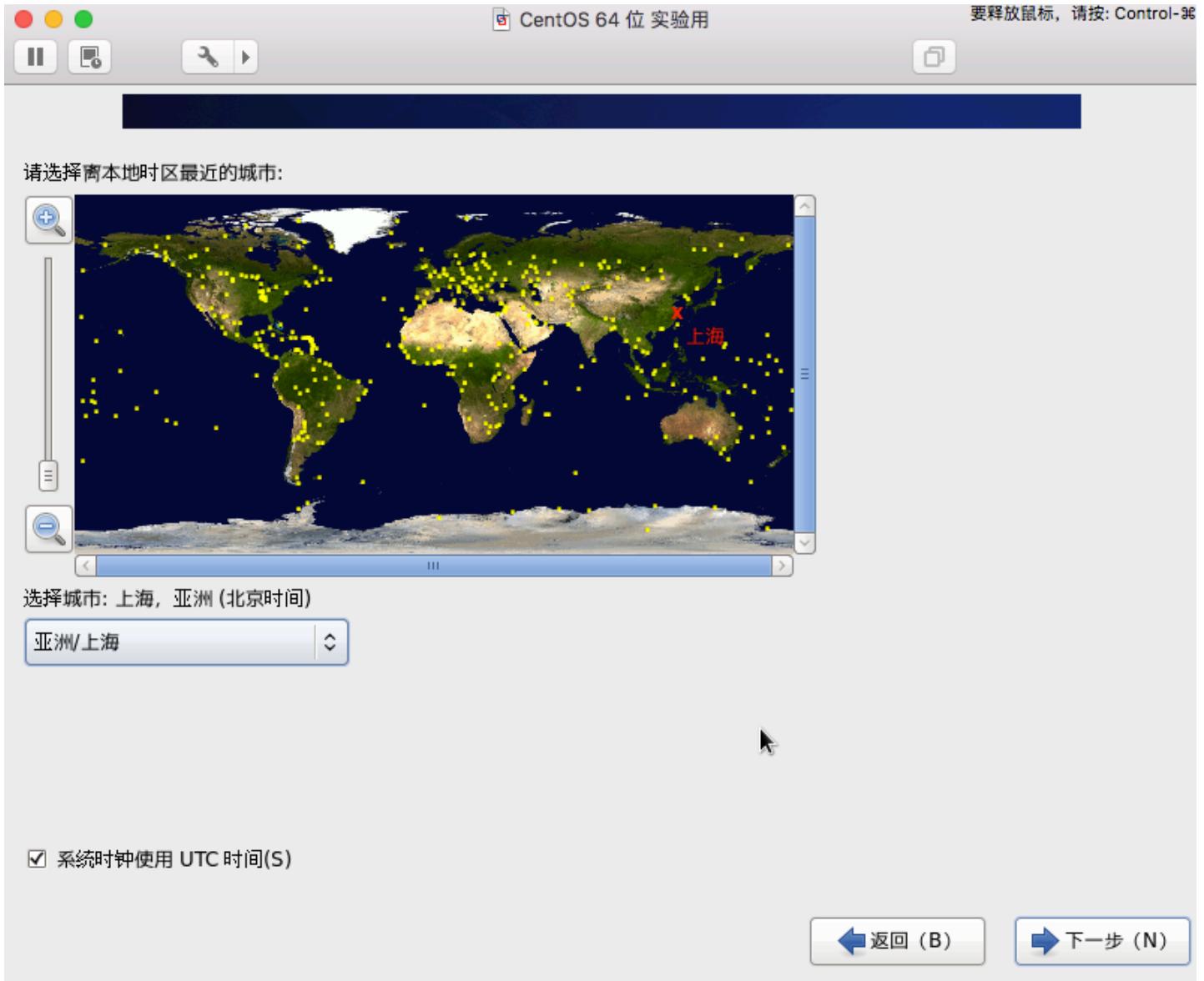


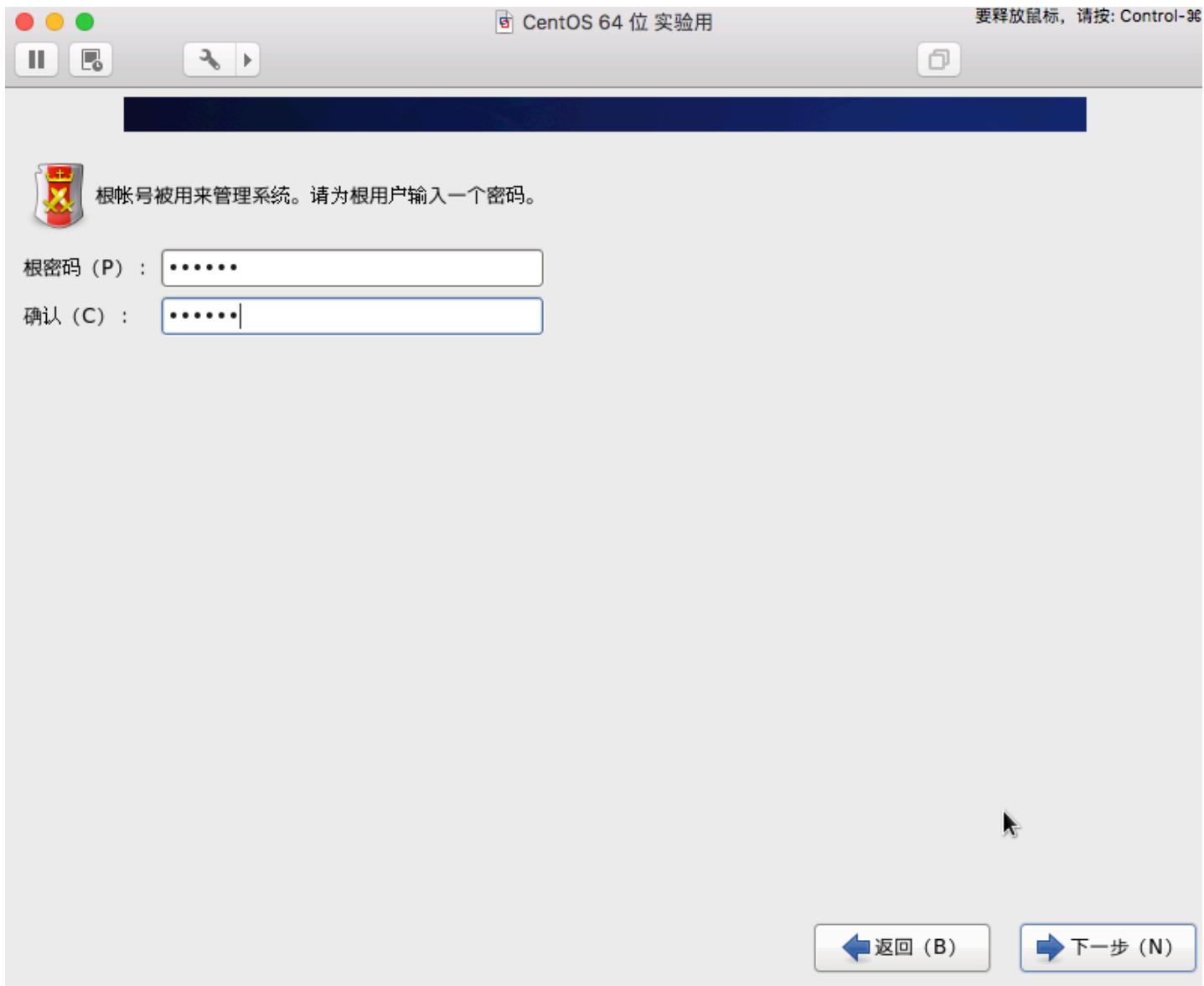
下面这里需要注意一下，如果曾经虚拟机安装过可能会有这个提示，直接忽略掉就可以了，选择“是，忽略所有数据”，然后点击“下一步”：



下面这里的主机名默认就好，因为对于Linux来说同一个局域网里并不是靠主机名来进行通信的，所以同一个局域网里可以有多个主机名相同的Linux计算机，这一点和Windows不一样。继续“下一步”：







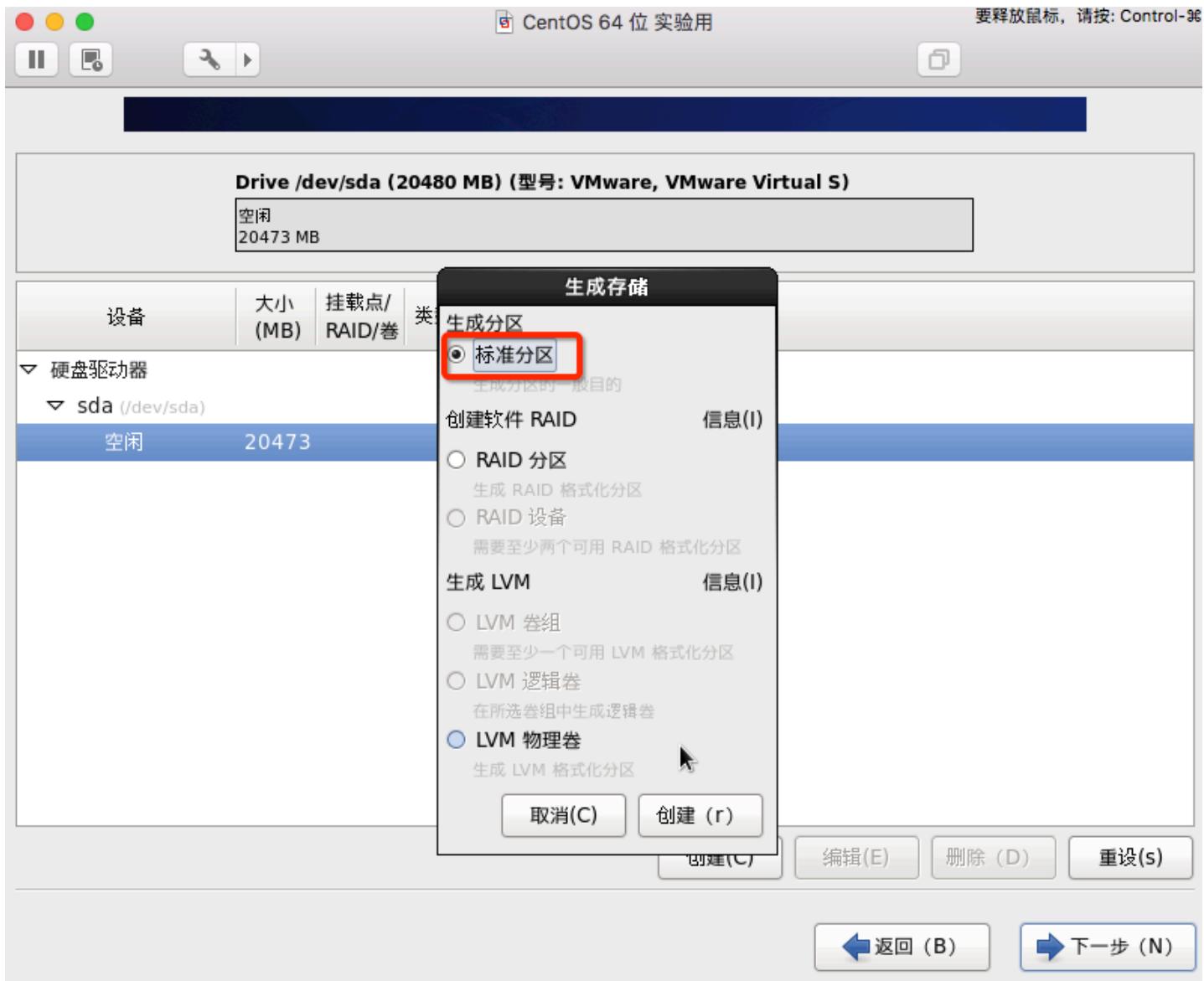
在上面设置好密码之后就到下面选择安装类型了，其实就是选择分区方式，这里前面我们讲了关于Linux系统分区的相关知识，所以我们选择“创建自定义布局”，然后点击“下一步”：



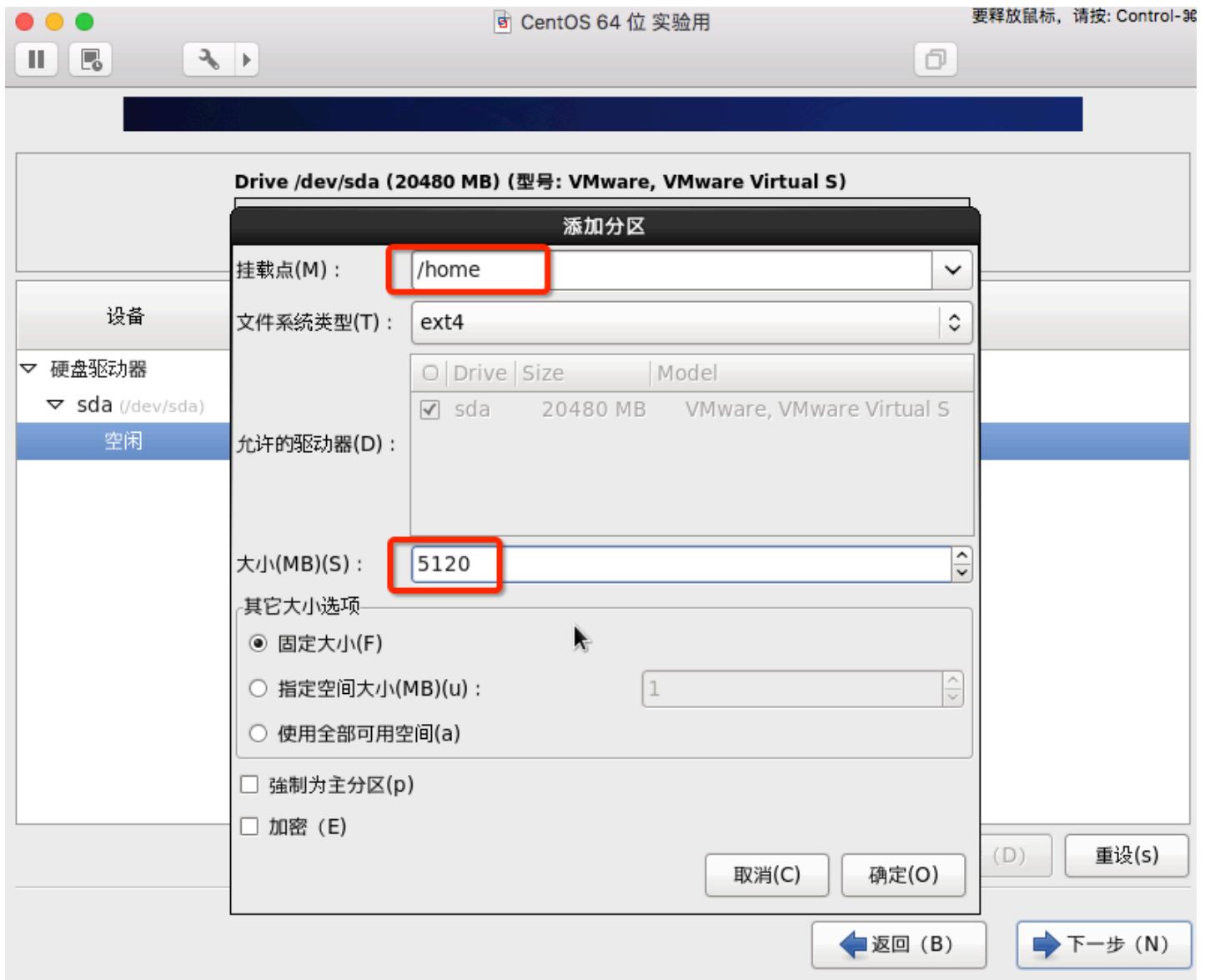
下一步之后我们看到了如下的图形化分区界面，选择“空闲”的硬盘驱动器，点击“创建”来创建我们想要的分区：



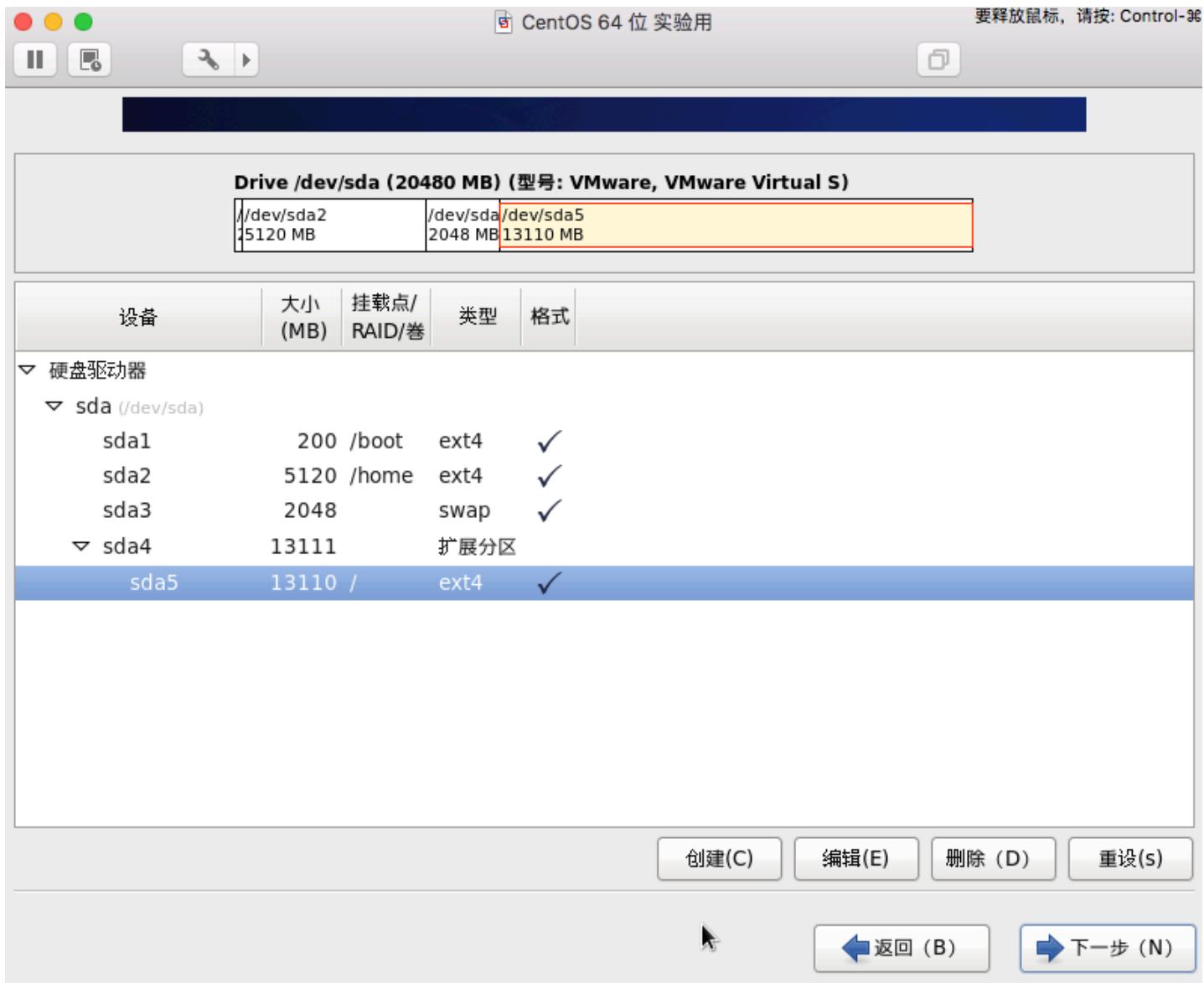
选择“标准分区”，点击“创建”：



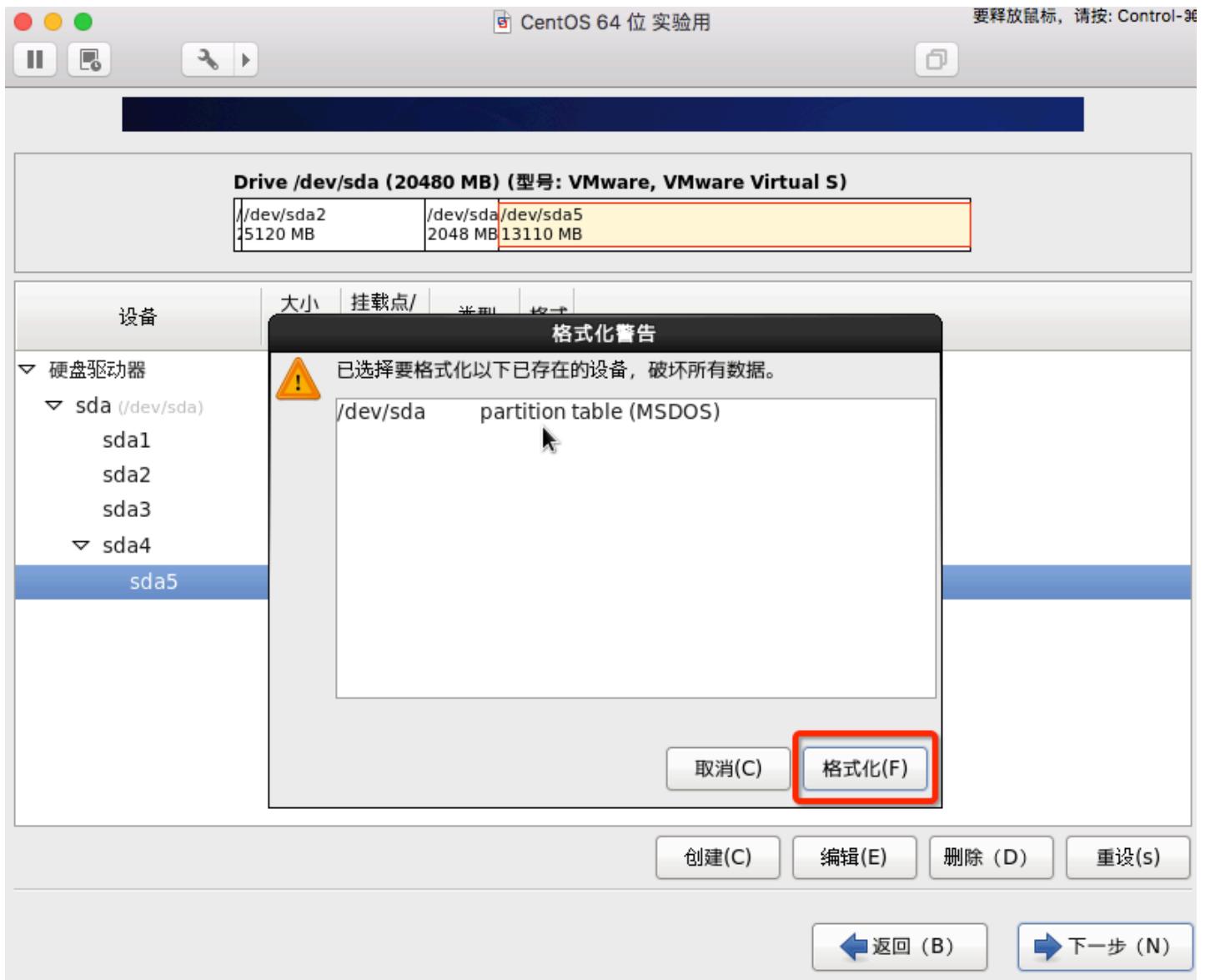
选择好“挂载点”和“大小”，点击“确定”：



重复上面的创建分区操作，根据之前说的系统分区要求做好分区后点击“下一步”，完整的分区创建完成后如下图：

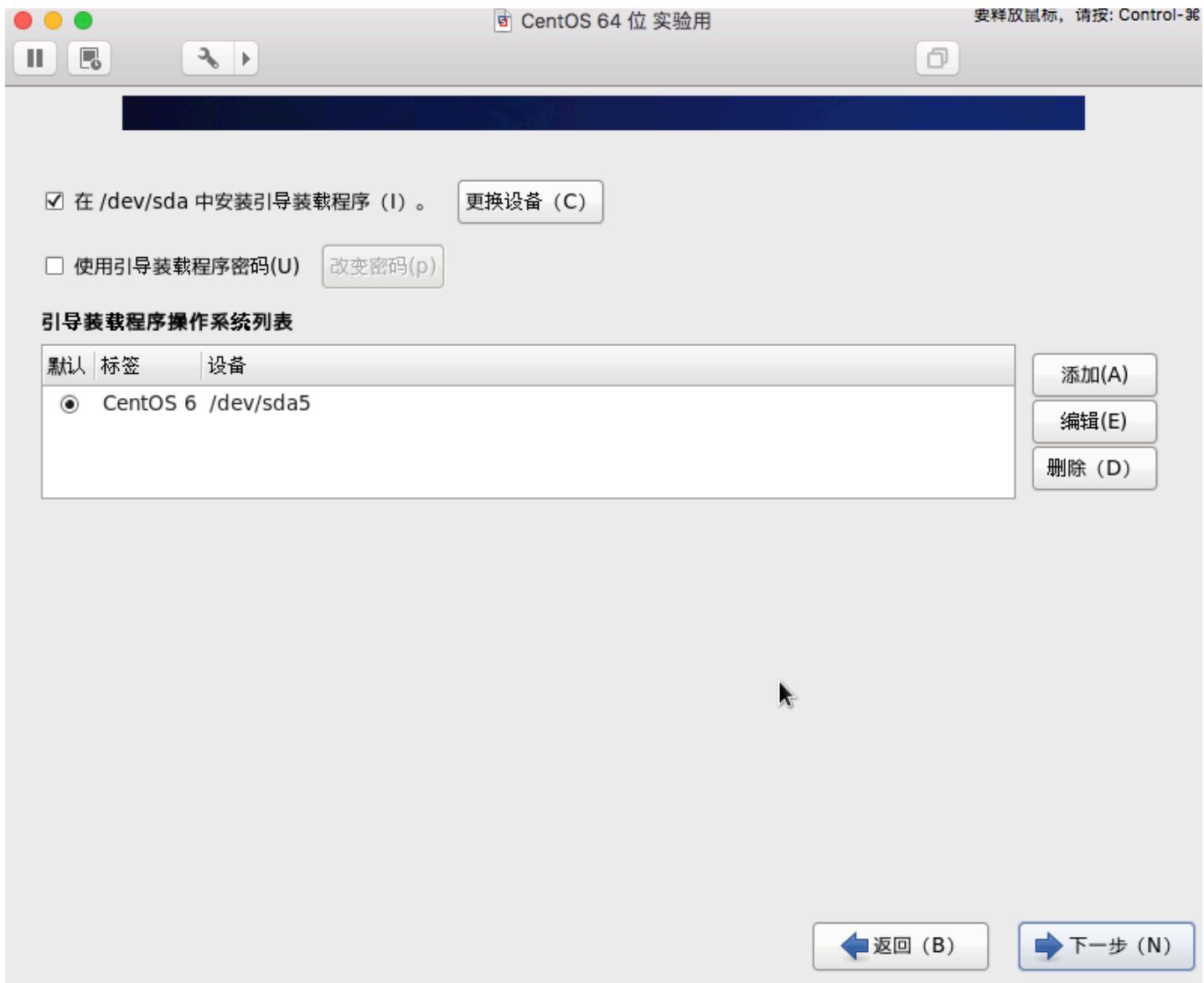


选择“格式化”，点击“下一步”：



选择“将修改写入磁盘”，点击“下一步”：





下面这里选择安装方式需要注意一下，这几种安装方式的说明如下：

- Desktop：基本的桌面系统，包括常用的桌面软件，如文档查看工具、浏览器等
- Minimal Desktop：基本的桌面系统，包含的软件较少
- Minimal：基本的系统，不含有任何可选的软件包
- Basic Server：安装的基本系统的平台支持，不包含桌面
- Database Server：基本系统平台，加上MySQL和PostgreSQL数据库，无桌面
- Web Server：基本系统平台，加上PHP，Web server，还有MySQL和PostgreSQL数据库的客户端，无桌面
- Virtual Host：基本系统加虚拟平台
- Software Development Workstation：包含软件包较多，基本系统，虚拟化平台，桌面环境，开发工具

这里需要说明的是服务器上安装的时候一般都是选择Minimal只进行最小安装，系统安装完成后需要什么软件装什么软件；但是由于我们初学者对Linux软件安装并不熟悉，所以我们为了后面的讲解方

便，我们这里就选择Basic Server，这样安装完成就包含了一些基本的软件包，使用起来更方便；

选择“Basic Server”，点击“下一步”：



下面就启动了系统安装程序，静静的等待就好...



CentOS 6

Community ENTerprise Operating System



已完成的软件包: 22 / 705

安装 glibc-common-2.12-1.209.el6.x86_64 (107 MB)

Common binaries and locale data for glibc

返回 (B)

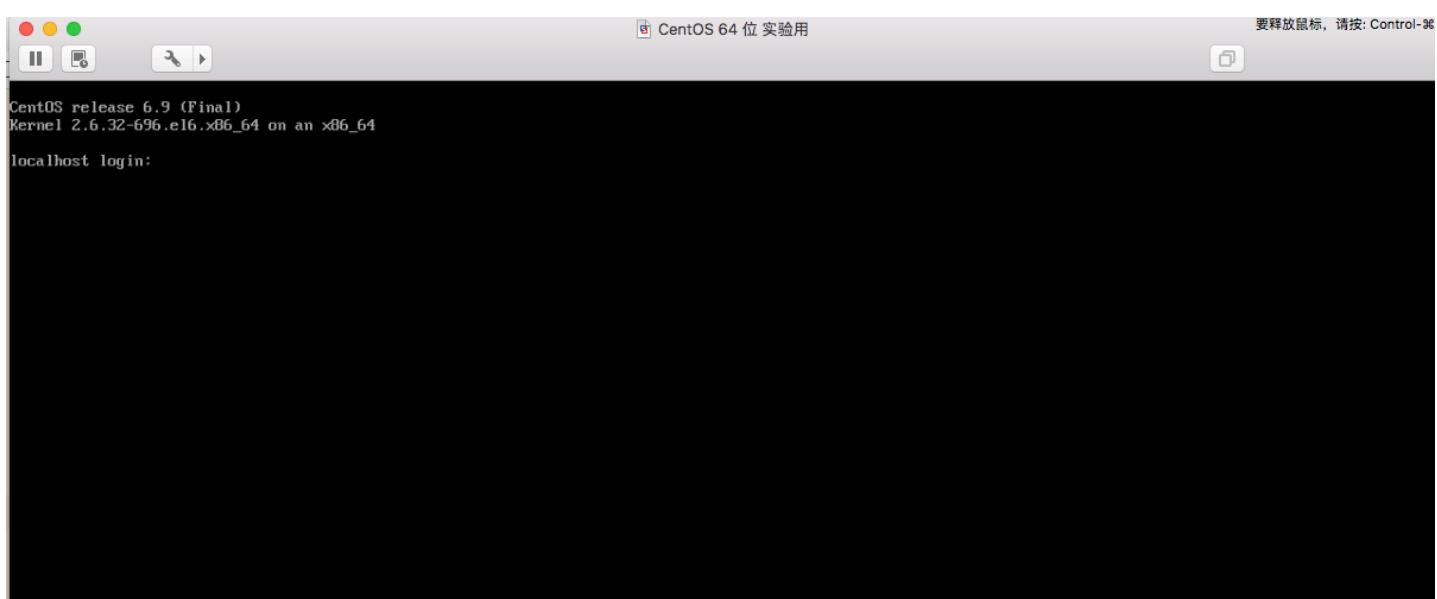
下一步 (N)



经过了上面漫长的等待，出现下面的画面，我们就完成了Linux系统的安装，点击“重新引导”：



重新引导完成后，CentOS系统启动，启动成功后界面如下，只有一个命令行字符界面，其他什么都没有：



我们输入用户名：root和密码即可登录，登录成功后如下图：

```
CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.x86_64 on an x86_64

localhost login: root
Password:
[root@localhost ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog
[root@localhost ~]#
```

安装完成之后在 `/root` 目录下有三个安装日志文件，说明如下：

- `/root/install.log`：存储了安装在系统中的软件包及其版本信息
- `/root/install.log.syslog`：存储了安装过程中留下的事件记录
- `/root/anaconda-ks.cfg`：以Kickstart配置文件的格式记录安装过程中设置的选项信息（用作网络批量安装）

OK，到此为止，我们学习Linux的旅程已经踏出了第一步，Linux系统安装完成。

三、Linux 序言

在经历了一顿猛如虎的操作之后，我们把操作系统安装好了并且启动了，那么接下来我们即将正式进入Linux基础学习了。

对于Linux学习，其实就一句话——“一切皆文件！”。

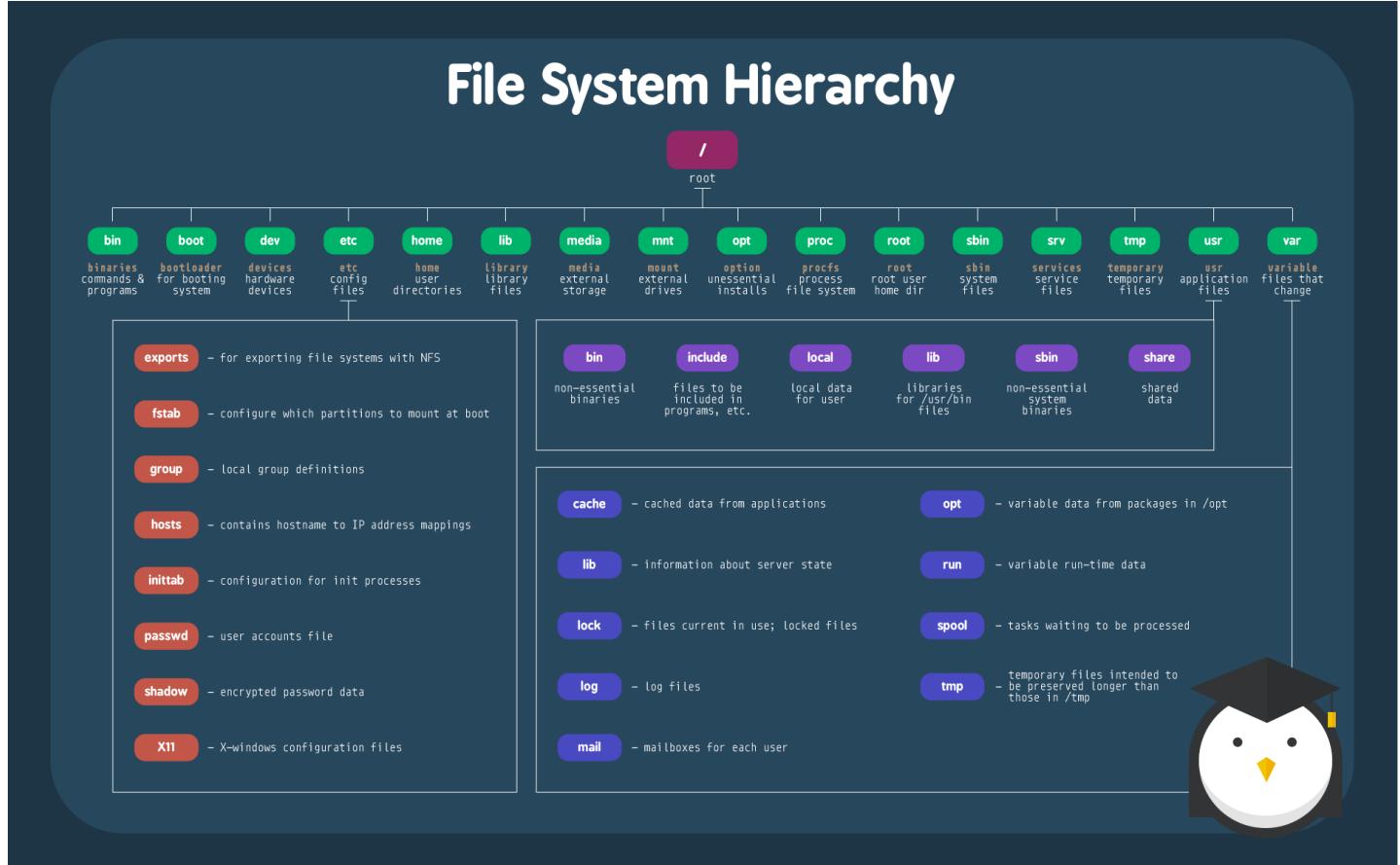
OK，我的教程到此结束（开始）了。。。

好吧，言归正传，在学习Linux之前我们了解了她的前世今生，可是到目前为止我们还不知道她的庐山真面目呢，下面我们就来介绍一下Linux的文件结构：

```
CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.x86_64 on an x86_64

localhost login: root
Password:
Last login: Wed Oct 24 22:23:35 on tty1
[root@localhost ~]# ls /
bin  cgroup  etc  lib  lost+found  misc  net  proc  sbin  srv  tmp  var
boot  dev  home  lib64  media  mnt  opt  root  selinux  sys  usr
[root@localhost ~]#
```

File System Hierarchy



目录	描述
/	主层次的根，也是整个文件系统层次结构的根目录
/bin	存放在单用户模式可用的必要命令二进制文件，所有用户都可用，如 cat、ls、cp等等
/boot	存放引导加载程序文件，例如kernels、initrd等
/dev	存放必要的硬件设备文件，例如/dev/sda1表示第一块硬盘第一个分区
/etc	存放系统管理和应用管理需要的所有配置文件
/home	用户的主目录，包括保存的文件，个人配置，等等
/media	可移动的多媒体(如CD-ROMs)的挂载点，例如：光盘
/mnt	临时挂载的文件系统，例如：优盘，移动硬盘等
/opt	存放可选的应用程序软件包，例如：数据库等软件安装包
/root	root用户的主目录
/sbin	存在超级管理员root用户可用的必要命令二进制文件，仅限root权限执行，例如：reboot、poweroff等等
/tmp	存放临时文件,通常在系统重启后删除

/var	各式各样的文件，一些随着系统常规操作而持续改变的文件就放在这里，比如日志文件，脱机文件，还有临时的电子邮件文件
/usr	这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似于Windows下的Program Files目录

四、Linux 网络管理

在这个网络高速发展的时代，拿到电脑的第一件事儿我们要干嘛呢？对，没错，插上网线（或连接无线网）进行上网，那么问题来了，插上网线就能上网了吗？NoNoNo，这中间还隔着好几个太平洋呢，不过好在我们有一个用来跨越这些太平洋很重要的硬件，叫做“网卡”。下面我们就来讲一下如何让Linux上网。

1、网络模型与作用

要想上网首先我们就要了解网络模型，目前公认的网络模型有两种，一种是经典OSI七层网络模型，另外一种是TCP/IP四层网络模型。

1.1 OSI七层网络模型

下图详细描述了OSI七层网络模型架构：

TCP/IP

第7层 应用层

各种应用程序协议，如HTTP、FTP、SMTP、POP3。



7
6
5

第6层 表示层

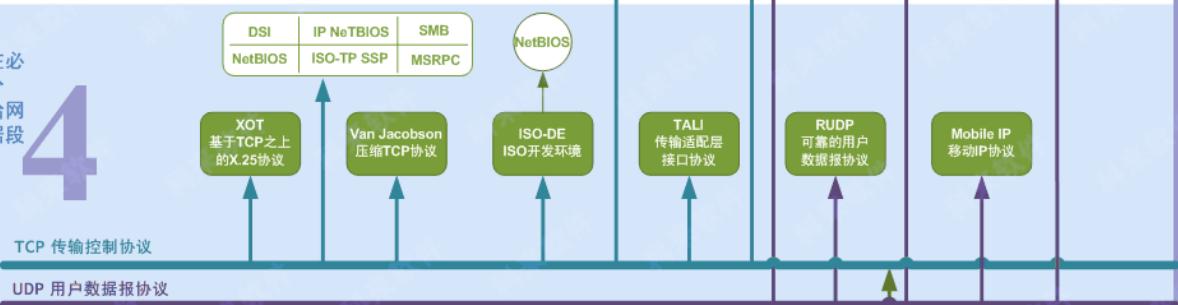
信息的语法语义以及它们的关联，如加密解密、转换翻译、压缩解压缩。

第5层 会话层

不同机器上的用户之间建立及管理会话。

第4层 传输层

接受上一层的数据，在必要的时候把数据进行分割，并将这些数据交给网络层，且保证这些数据段有效到达对端。



第3层 网络层

控制子网的运行，如逻辑编址、分组传输、路由选择。



第2层 数据链路层

物理寻址，同时将原始比特流转变为逻辑传输线路。

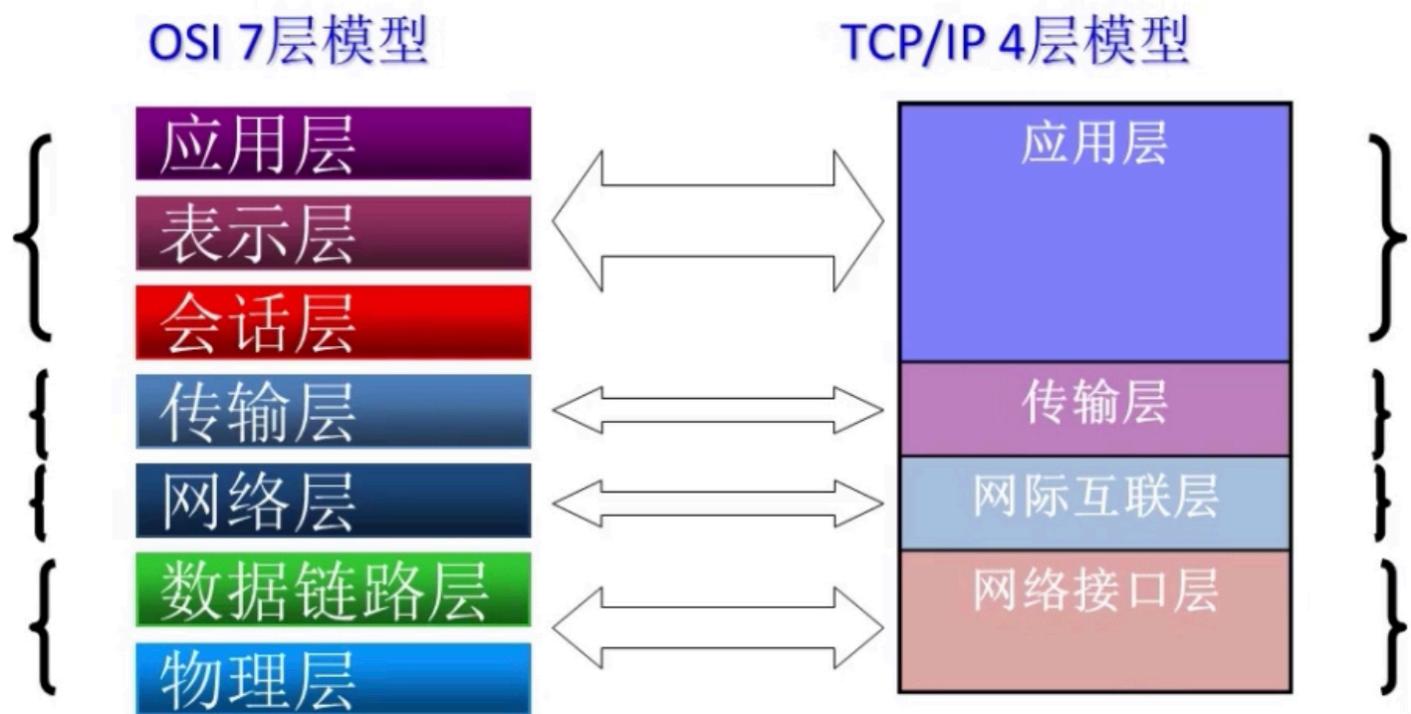


第1层 物理层

机械、电子、定时接口通信信道上的原始比特流传输。

1.2 TCP/IP四层网络模型

我们说上面的七层网络模型很经典，但是在实际应用中很多时候网络架构并不是这样的，而是采用TCP/IP四层网络模型，那么它们之间到底有怎样的联系呢？我们来看下图：



1.3 关于IP地址

从上面的网络模型中我们能够知道，IP其实是工作在网络层的互联网协议，它的作用这里不作过多阐述，主要想给大家说的是IP的分类，IP地址分为A~E五大类，但是其中D类IP、E类IP并不开放民用，所以下面是A、B、C三类IP的划分情况：

类别	最大网络数	IP地址范围	最大主机数	私有IP地址范围
A	$126 (2^7 - 2)$	1.0.0.0~126.255.255.255	$2^{24} - 2$	10.0.0.0~255.255.255.255
B	$16384 (2^{14})$	128.0.0.0~191.255.255.255	$2^{16} - 2$	172.16.0.0~172.31.255.255
C	$2097152 (2^{21})$	192.0.0.0~223.255.255.255	$2^8 - 2$	192.168.0.0~192.168.255.255

1.4 关于子网掩码

子网掩码是一种用来指明一个IP地址的哪些位标识的是主机所在的子网，以及哪些位标识的是主机的位掩码。好吧，说人话就是子网掩码是用来确定两个IP是否属于同一个网段的。子网掩码不能单独存在，它必须结合IP地址一起使用。既然这样我们就结合上面三类IP地址来看一下子网掩码，一般情况下，不同类别的IP地址与子网掩码具有如下对应关系：

IP Address	10.	1.1.200
Subnet Mask	255.	0.0.0
Network ID	10.	0.0.0

A类IP地址与子网掩码的对应关系

IP Address	172.16.	1.200
Subnet Mask	255.255.	0.0
Network ID	172.16.	0.0

B类IP地址与子网掩码的对应关系

IP地址	192. 168. 1	200
Subnet Mask	255. 255. 255.	0
Network ID	192. 168. 1.	0

C类IP地址与子网掩码的对应关系

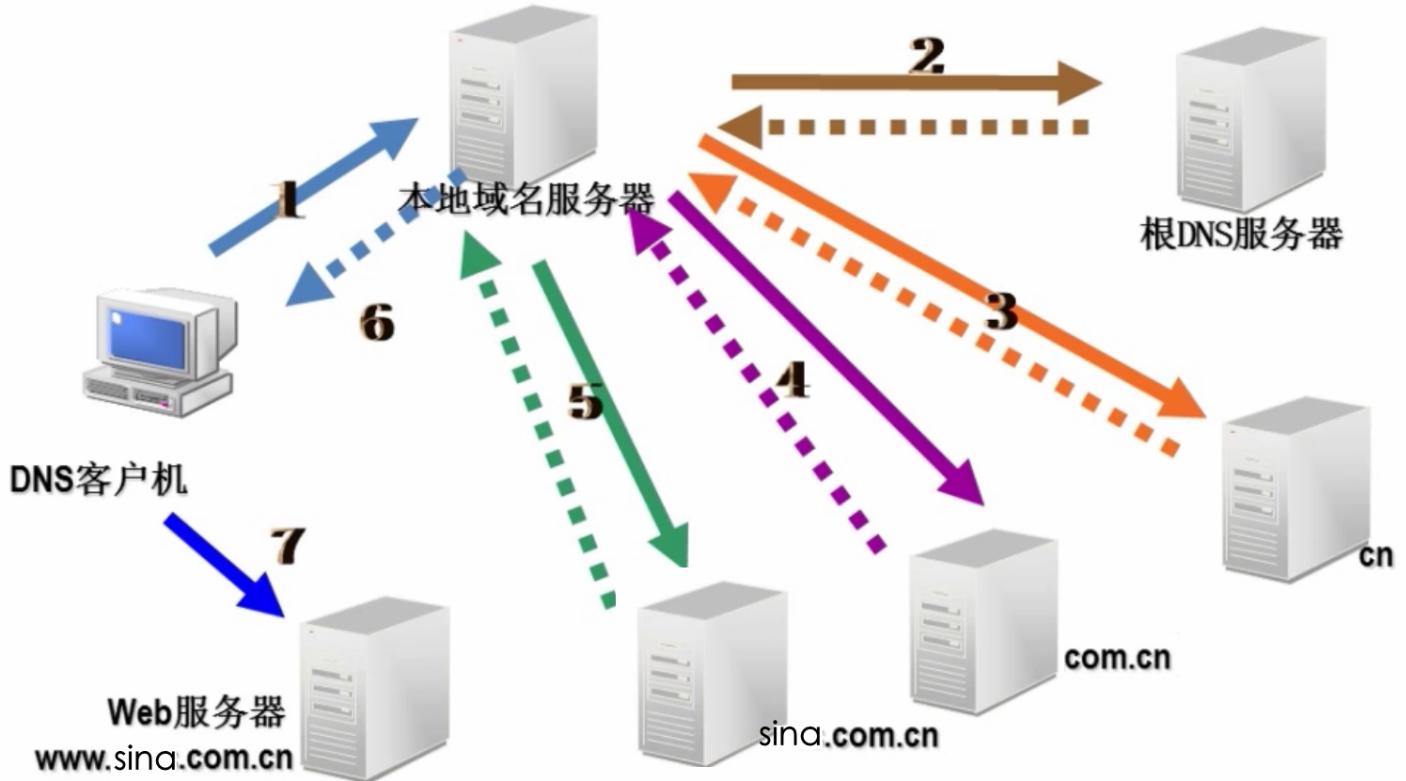
1.5 关于端口

端口是计算机对外服务的大门，每开启一个服务都必然有一个对应的端口号来让外部访问，常见端口号如下：

端口号	对应服务
20、21	FTP 文件传输服务
22	SSH 安全shell协议
23	Telnet 远程登录协议（现在默认关闭）
53	DNS 域名解析服务
80	HTTP 超文本传输协议
25、110	SMTP 邮件传输协议、POP3 邮局协议3代

1.6 关于DNS

DNS，域名解析服务，主要负责把域名解析成IP地址，细节不作过多描述，这里需要说明的是DNS的查询过程，详见下图：



1.7 关于网关（路由器）

网关是访问互联网的必须配置项之一，主要作用：

- 网关在所有内网计算机访问的不是本网段的数据报时用于数据转发；
- 网关负责进行内网IP与公网IP之间的相互转换；

一般情况下，网关是路由器，也可以是服务器做路由功能。

Tips：如果仅仅是内网通信其实是可以不配置DNS和网关的，但是如果需要访问互联网就必须配置网关。

2、Linux网卡与配置

Linux配置IP地址的方法有四种：

- ifconfig命令临时配置IP地址
- setup工具永久配置IP地址
- 修改网络配置文件
- 图形界面配置IP地址

我们这里只讲通过网络配置文件的配置方式，原因很简单，因为Linux一切皆文件！

2.1 网卡信息文件

网卡信息文件在Linux中的路径为：`/etc/sysconfig/network-scripts/ifcfg-eth0`，关于该配置文件的基本说明如下：

字段名=对应值	描述
<u>DEVICE=eth0</u>	网卡设备名
<u>HWADDR=00:0C:29:71:27:EE</u>	网卡设备MAC地址
<u>TYPE=Ethernet</u>	网卡类型为以太网
<u>UUID=6c3e1dff-3e00-4933-81aa-317095967170</u>	唯一识别码
<u>ONBOOT=yes</u>	是否随网络服务启动，eth0生效
<u>_NM_CONTROLLED=no</u>	是否可以由Network Manager图形管理工具托管
<u>BOOTPROTO=dhcp</u>	是否自动获取IP (none、static、dhcp)，如果该项配置为dhcp，则该文件就只需要配置该表中带有下划线加粗的字段，但前提是确保网络中有dhcp服务器
<u>USERCTL=no</u>	不允许非root用户控制此网卡
IPADDR=192.168.1.110	IP地址
NETMASK=255.255.255.0	子网掩码
GATEWAY=192.168.1.1	网关
DNS1=192.168.0.1	DNS (可以配置多个，DNS2、DNS3...)
IPV6INIT=no	不启用IPv6

2.2 主机名文件

主机名文件路径为：`/etc/sysconfig/network`，关于该配置文件的基本说明如下：

字段名=对应值	描述
NETWORKING=yes	网络服务是否工作
HOSTNAME=localhost.localdomain	主机名

该配置必须重启后才能生效
可以通过`hostname`命令临时修改

2.3 DNS配置文件

DNS配置文件路径为：`/etc/resolve.conf`，该配置文件说明如下：

字段名 对应值	描述
nameserver 192.168.0.1	域名服务器

域名服务器可以有多个，换行写或者后面使用空格分割

当网卡配置文件中的 `BOOTPROTO=dhcp` 时，该文件可以为空，不用配置

3、重启网络服务

完成上面的配置后，我们的网卡配置就完成了，下面我们需要重启一下网络服务来让网卡工作，执行命令：`service network restart`，如下图：

```
[root@localhost ~]# service network restart
Shutting down loopback interface: [ OK ]
Bringing up loopback interface: [ OK ]
Bringing up interface eth0:
Determining IP information for eth0... done. [ OK ]
[root@localhost ~]#
```

网络服务重启完成之后我们就可以执行命令：`ifconfig` 来查看Linux系统的IP地址，也可以访问互联网了，如下图：

```
[root@localhost ~]# service network restart
Shutting down loopback interface:                                [ OK ]
Bringing up loopback interface:                                 [ OK ]
Bringing up interface eth0:                                    [ OK ]
Determining IP information for eth0... done.                  [ OK ]

[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0C:29:71:27:EE
          inet addr:192.168.84.132 Bcast:192.168.84.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe71:27ee/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:10 errors:0 dropped:0 overruns:0 frame:0
            TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:1168 (1.1 KiB) TX bytes:1802 (1.7 KiB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:65536 Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)

[root@localhost ~]# _
```

OK，到这里为止，Linux网络管理我们就全部讲完了！！

五、Linux 用户管理

在完成了网络配置这一小小的难关之后，我们讲是不是就可以开始使用我们的Linux系统了呢？其实理论上是可以了，但生活嘛，总是理想很丰满，现实很骨干啊！不知道大家有没有留意到一个细节，就是我们在安装Linux系统的时候从来没有要求过我们输入用户名，只是让我们输入了一个密码而已，咦，这好像跟我们安装界面话的操作系统，比如：Windows或者macOS 似乎都不太一样哎，这是为什么呢？OK，下面我们就来学习一下Linux的用户管理。

1、用户与用户组

现在解释一下为什么之前安装Linux的时候没有让我们输入用户名，这是因为Linux采用无界面化安装的时候她并不会主动为我们分配用户，她会为自己创建一个统一的 `root` 用户，她认为剩下的事情都应该由 `root` 用户来操作，同时也给这个 `root` 用户赋予了极大的权限，后面我们讲到Linux 权限管理的时候大家就会体会到这个 `root` 用户是有多无敌，多寂寞了。也正是因为这一点我们需要管理好我们Linux的用户。

简单的说，用户就是使用该操作系统的人，而用户组是指具有相同权限的一组用户。我们一开始就说“Linux一切皆文件！”，那么用户和用户组对应的文件又在哪里呢？在开始讲这些文

件之前，我需要大家先了解一个Linux文件编辑器——**VIM**，这里不会赘述怎么使用这个编辑器，请大家自行参考：[简明 VIM 练级攻略](#)

1.1 用户组信息文件

当前系统中所有用户组信息都保存在 `/etc/group` 这个文件中，执行 `vim /etc/group` 可以看到如下内容：

```
root:x:0:
bin:x:1:bin,daemon
daemon:x:2:bin,daemon
sys:x:3:bin,adm
adm:x:4:adm,daemon
tty:x:5:
disk:x:6:
lp:x:7:daemon
mem:x:8:
kmem:x:9:
wheel:x:10:
mail:x:12:mail,postfix
uucp:x:14:
man:x:15:
games:x:20:
gopher:x:30:
video:x:39:
dip:x:40:
ftp:x:50:
lock:x:54:
audio:x:63:
nobody:x:99:
users:x:100:
dbus:x:81:
rpc:x:32:
utmp:x:22:
utempter:x:35:
floppy:x:19:
vcsa:x:69:
abrt:x:173:
cdrom:x:11:
tape:x:33:
dialout:x:18:
wbpriv:x:88:
rpcuser:x:29:
nfsnobody:x:65534:
"/etc/group" 49L, 657C
```

每一行的内容以冒号分隔，释义如下：

group	x	123	abc,def,xyz
组名称	组密码占位符	组编号	组中用户列表

组密码占位符：一定是x

组编号：root用户组的编号一定是0，1~499是系统预留组编号，一般是预留给安装的软件或服务的。

组中用户列表：为空并不代表该组中就一定没有用户，因为如果该组中有且仅有一个用户，并且用户名和组名一致，这时候是可以省略的

1.2 用户组的密码信息文件

当前系统中所有用户组的密码信息都保存在 /etc/gshadow 这个文件中，执行 `vim /etc/gshadow` 可以看到如下内容：

```

root:::
bin:::bin,daemon
daemon:::bin,daemon
sys:::bin,adm
adm:::adm,daemon
tty:::
disk:::
lp:::daemon
mem:::
kmem:::
wheel:::
mail:::mail,postfix
uucp:::
man:::
games:::
gopher:::
video:::
dip:::
ftp:::
lock:::
audio:::
nobody:::
users:::
dbus::!:
rpc::!:
utmp::!:
utempter::!:
floppy::!:
vcsa::!:
abrt::!:
cdrom::!:
tape::!:
dialout::!:
wbpriv::!:
rpcuser::!:
nfsnobody::!:
"/etc/gshadow" [readonly] 49L, 537C

```

与用户组信息一样，每一行的内容以冒号分隔，释义如下：

group	空/*	空	abc,def,xyz
组名称	组密码	组管理者	组中用户列表

组密码：为空或为*号的时候都表示当前组没有密码

组管理者：一般都为空，表示组内任何用户都可以管理该组

1.3 用户信息文件

当前系统中所有用户信息都保存在 `/etc/passwd` 这个文件中，执行 `vim /etc/passwd` 可以看到如下内容：

同样，每一行的内容以冒号分隔，释义如下：

user	x	123	456	xxx	/home/user	/bin/bash
用户名	密码占位符	用户编号	用户组编号	用户注释信息	用户家目录	shell类型

用户家目录：创建用户时系统会自动在/home目录下创建一个与用户名同名的目录作为用户的家目录，用户登录系统时自动到该目录下

shell类型: 用来指定用户登录时执行命令的方式, 一般是**bash**

1.4 用户密码信息文件

当前系统中所有用户信息都保存在 `/etc/shadow` 这个文件中，执行 `vim /etc/shadow` 可以看到如下内容：

这个文件就没什么好说的了，其中第二个字段就表示用户密码，采用的是hash加盐的加密方式。

OK，和用户和用户组相关的文件都介绍完了，这几个文件我们只要了解就行，一般不会去修改它，因为我们操作用户和用户组都有其他方式。

2、基本命令

与操作用户组相关的基本命令如下：

- `groups [username]` : 查看用户所属组, 不写参数默认查看当前登录用户所属组
 - `groupadd groupname` : 添加用户组
 - `groupdel groupname` : 删除用户组
 - `groupmod -n [new_groupname] [old_groupname]` : 修改用户组名称
 - `groupmod -g groupid groupname` : 修改用户组id

与操作用户相关的基本命令如下：

- `useradd username` : 添加用户， 默认添加用户时会创建一个同名用户组， 添加的用户属于该用户组， 同时在/home目录下会创建一个同名目录为该用户的家目录
- `useradd -g groupname username` : 添加用户到指定用户组
- `useradd -d /home/xxx username` : 添加用户时指定家目录
- `userdel username` : 删除用户
- `userdel -r username` : 删除用户同时删除用户的家目录
- `usermod -l new_username old_username` : 修改用户名
- `usermod -d /home/xxx username` : 修改用户家目录
- `usermod -g groupname username` : 修改用户所属组

`passwd username` : 设置/修改用户密码

注意：创建用户之后必须设置密码才可以登录系统！！

六、Linux 权限管理

这里我们所说的权限管理实际上就是指Linux的文件权限管理，因为Linux一切皆文件！下面我们就来看一下Linux文件权限是怎么管理的。执行 `ls -l /etc` 可以看到如下内容：

```

drwxr-xr-x.  2 root root  4096  8月 21 2010 popt.d
drwxr-xr-x.  2 root root  4096 10月 24 08:28 portreserve
drwxr-xr-x.  2 root root  4096 10月 24 08:28 postfix
drwxr-xr-x.  3 root root  4096 10月 24 08:28 ppp
-rw-r--r--.  1 root root 188384 10月 24 09:08 prelink.cache
-rw-r--r--.  1 root root    789 7月 19 2013 prelink.conf
drwxr-xr-x.  2 root root  4096 8月 19 2013 prelink.conf.d
-rw-r--r--.  1 root root    233 1月 12 2010 printcap
-rw-r--r--.  1 root root  1841 3月 22 2017 profile
drwxr-xr-x.  2 root root  4096 10月 24 08:30 profile.d
-rw-r--r--.  1 root root   6455 1月 12 2010 protocols
drwxr-xr-x.  2 root root  4096 10月 24 08:27 pulse
-rw-r--r--.  1 root root   220 10月 14 2008 quotagrpadmins
-rw-r--r--.  1 root root   259 7月 24 2015 quotatab
lrwxrwxrwx.  1 root root      7 10月 24 08:28 rc -> rc.d/rc
lrwxrwxrwx.  1 root root     10 10月 24 08:28 rc0.d -> rc.d/rc0.d
lrwxrwxrwx.  1 root root     10 10月 24 08:28 rc1.d -> rc.d/rc1.d
lrwxrwxrwx.  1 root root     10 10月 24 08:28 rc2.d -> rc.d/rc2.d
lrwxrwxrwx.  1 root root     10 10月 24 08:28 rc3.d -> rc.d/rc3.d
lrwxrwxrwx.  1 root root     10 10月 24 08:28 rc4.d -> rc.d/rc4.d

```

图中红色框部分就表示对应文件的权限，详细定义如下：



关于文件类型：除了上面图中展示的三种类型的文件之外还有其他类型的文件，但是，友情提示，如果你不是特别熟悉Linux的文件系统，那么当你看到除以上三种类型文件的时候请你绕着走！

关于特殊权限：我就只说一点，如无特殊需求，请无视它，谢谢！

1、文件权限的修改

修改文件权限的命令格式为：`chmod [选项] 模式 文件名`

- 选项：选项有很多，但常用的就一个，`-R` 表示递归修改
- 模式：分为两种，一种是字母模式 `[ugo]a [+-=] [rwx]`，意思为“为哪一组权限添加、删除或赋予什么样的权限，其中a表示所有组（包括所有者、所属组和其他人）”，例如：`chmod u+x test.sh` 表示为 `test.sh` 这个文件的所有者添加可执行权限；另外一种是数字模式 `[mode=421]`，采用rwx对应的数字来表示权限，每一个权限之间是与的关系，所以采用加法运算，每一组权限只用一个数字表示，例如：
`chmod 744 test.sh` 表示将 `test.sh` 这个文件的权限修改为其所有者具有读写执行权限，而其所属组和其他人只有读权限。

关于模式：可能一眼看上去字母模式更清晰，但说了你可能不信，这种模式使用度几乎为零，原因有二，其一字母模式你需要知道原文件是什么权限，其二字母并不如数字简单好记。

2、文件权限的作用

权限是Linux文件系统的核心，本质目的还是为了安全，其实前面介绍完文件权限的时候大家对权限的作用就基本上心里有数了，这里我需要敲黑板的只有一个问题：“对文件有读写权限就能删除文件吗？？”

```
[sands@localhost test]$ ll  
总用量 4  
-rwxr--r--. 1 sands sands 33 10月 26 01:34 test.sh  
[sands@localhost test]$ rm test.sh  
rm: 无法删除"test.sh": 权限不够  
[sands@localhost test]$
```

从上图我们就可以看到，不是这样的，就算你对文件有读写执行的一切权利，也不代表你就能删除文件，主要原因跟Linux的文件系统和存储结构有关，简单的解释为你对这个文件的权限只是表示了你对这个文件内的数据所拥有的权限，而对于一切皆文件的Linux来说，这个文件本身实际上是存储在 `/home/sands/test` 文件夹下的，所以它其实是这个文件夹文件内的数据，要操作它那就得拥有这个文件夹的权限。

```
[sands@localhost test]$ ll  
总用量 4  
-rwxr--r--. 1 sands sands 33 10月 26 01:34 test.sh  
[sands@localhost test]$ rm test.sh  
rm: 无法删除"test.sh": 权限不够  
[sands@localhost test]$ pwd  
/home/sands/test  
[sands@localhost test]$ cd ..  
[sands@localhost ~]$ ll  
总用量 4  
dr-xr-xr-x. 2 sands sands 4096 10月 26 01:34 test  
[sands@localhost ~]$
```

可以看到我们当前用户对 `test` 文件夹并不具备写权限，也就解释了为什么我们无法将这个文件夹里面的 `test.sh` 文件删掉了。

注意：我们这里所说的权限 `root` 用户不适用！！！

另外，这里补充一点关于权限对目录的作用：

- r: 可以查询目录下的文件 (`ls`)
- w: 具有修改目录结构的权限，例如：新建、删除、复制、剪切 (`touch, rm, cp, mv`)
- x: 可以进入目录 (`cd`)

总结：对于文件来说，最高权限是执行 (`x`)，但对于目录来说，最高权限是写

(w)。所以对于目录有效权限只有0（没有任何权限）、5（有读r和执行x权限，可以查看内容）、7（有任何权限），**禁止**对文件夹赋予像1、4、6等这些没有意义的权限。

3、文件所属的修改

修改文件的所属者和所属组的命令格式为：`chown [选项] owner:[groupname] filename`，其中 [选项] 有很多，但常用的只有一个 `-R` 表示递归修改，`groupname` 是可选的，例如：

- `chown sands test.sh` 表示只将 `test.sh` 这个文件的所属者改为 `sands` 用户
- `chown sands:usergroup test.sh` 表示将 `test.sh` 这个文件的所属者改为 `sands` 用户，所属组改为 `usergroup` 用户组

4、文件的默认权限

说了这么多文件权限的东西，似乎我们还忽略了一点，就是如果我们直接创建一个文件，那么它的权限是什么呢？为什么是这样呢？

```
[sands@localhost ~]$ touch test.sh
[sands@localhost ~]$ ll
总用量 0  664
-rw-rw-r--. 1 sands sands 0 10月 26 12:09 test.sh
[sands@localhost ~]$
```

从上图可以看到，在非root用户下直接创建一个文件默认权限为664，下面我们就来解释一下这是为什么：

在开始解释之前，我需要大家明白一个原则性问题，之前我们也提到过，对于Linux来说文件的最高权限是x执行，但是Linux认为这个最高权限太危险了，所以文件默认是不能新建为可执行文件的，必须手动赋予执行权限，这样一来文件默认的最大权限就只能是666了。那是不是说新建的文件默认权限就是666了呢？显然不是，这还得受一个叫作 `umask` 值的家伙的管制：

```
[sands@localhost ~]$ touch test.sh  
[sands@localhost ~]$ ll  
总用量 0  
-rw-rw-r--. 1 sands sands 0 10月 26 12:09 test.sh  
[sands@localhost ~]$ umask  
0002  
[sands@localhost ~]$
```

第一位0：表示文件特殊权限

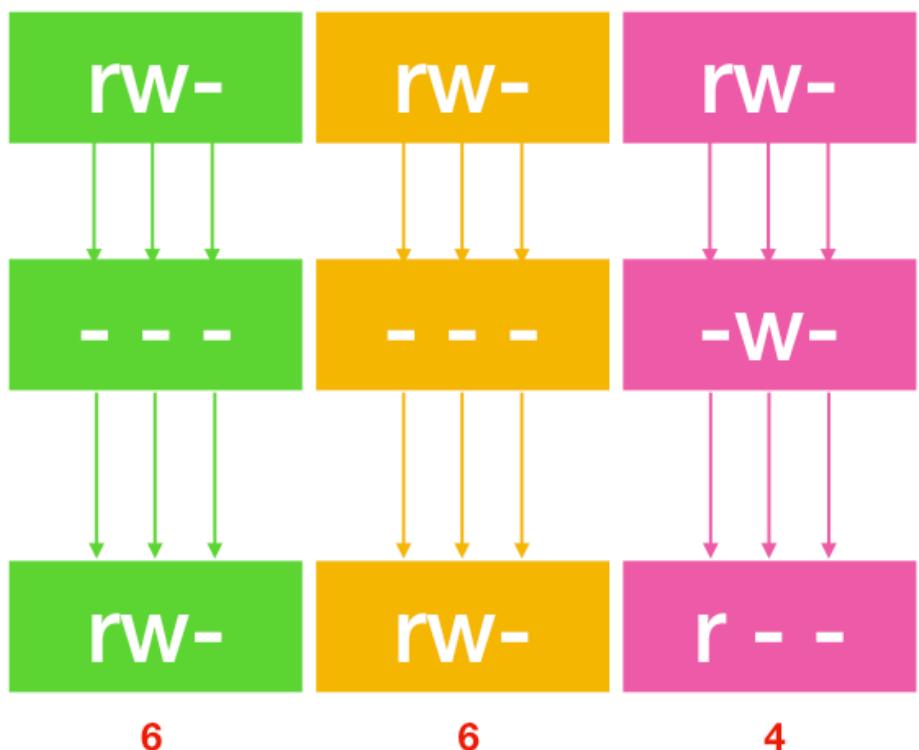
后面三位002：表示文件默认权限

所以默认文件权限的具体计算方式如下图：

文件最大默认权限666对应字母表示为：

umask值的后三位002对应字母表示为：

将文件最大默认权限对应的字母与umask值的后三位对应字母做“减法”运算：



- 这里所说的减法并不是真正意义上的减法，而是按位异或运算，说成字母减法只是为了方便大家理解。

现在我们知道了Linux文件的默认权限是这么来的，那其实文件夹默认权限也是同样的方式计算来的，但不同的是我们前面说过文件夹的最高权限是w写，所以Linux就认为文件夹的写并不具备什么危险，所以文件夹默认的最大权限就是777，所以计算默认权限的时候也应该用777来进行计算。

关于umask值的修改：

- 临时修改：直接执行 `umask 0022`
- 永久修改：修改 `/etc/profile` 这个环境变量文件

```
# By default, we want umask to get set. This sets it for login shell
#
# Current threshold for system reserved uid/gids is 200
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`"
]; then
    umask 002
else
    umask 022
fi

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        if [ "${{-#*i}}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null 2>&1
```

七、Linux 软件安装管理

软件安装对于任何操作系统来说都是非常重要的部分，原因很简单，我们安装操作系统就是要做事儿的，而一个空的操作系统能帮我们做的事情微乎其微，所以我们才需要通过安装各种软件来为我们服务。

1、挂载

在讲软件安装之前先给大家讲一个概念叫做**挂载**，我们都知道在macOS和Windows中，当我们把光盘放入光驱或者是优盘插入USB接口的时候在我们的电脑里就能查看到对应的里面的内容了，但是很不幸的告诉你，在Linux中并不是这样的。我们在讲Linux系统分区的时候就说过，硬盘每个分区的盘符称为“挂载点”，读取分区的过程称为“挂载”，其实对于其他设备也一样，我们也要通过挂载的方式来读取设备内容。

一般我们选择将光盘挂载到系统的 `/media` 文件夹下，而将优盘等挂载到系统的 `/mnt/xxx` 文件夹下，`xxx`的意思是指在 `/mnt` 文件夹下再创建一个文件夹来挂载对应设备，挂载命令如下：

- 挂载光盘：`mount /dev/sr0 /media` 表示将光盘挂载到 `/media` 目录下，如下图：

```
[root@localhost /]# ls /media/
[root@localhost /]# mount /dev/sr0 /media/
mount: block device /dev/sr0 is write-protected, mounting read-only
[root@localhost /]# ls /media/
CentOS_BuildTag      RELEASE-NOTES-en-US.html
EFI                  repodata
EULA                RPM-GPG-KEY-CentOS-6
GPL                 RPM-GPG-KEY-CentOS-Debug-6
images              RPM-GPG-KEY-CentOS-Security-6
isolinux            RPM-GPG-KEY-CentOS-Testing-6
Packages            TRANS.TBL
[root@localhost /]#
```

- 挂载优盘：先执行 `fdisk -l` 来查看优盘设备文件名，再执行 `mount /dev/sdb1 /mnt/usb` 来将文件名为sdb1的优盘挂载到mnt文件夹下的usb文件夹里

卸载已经挂载好的设备也很简单：`umount /dev/sr0` 即可卸载已经挂载的光盘。

2、压缩和解压缩

快要吐血了啊，要讲一个东西涉及的实在是太多了，为了安装软件我们应当知道软件包是如何进行压缩和解压缩的？

- `.zip` 格式压缩命令为：`zip [选项] 压缩包名 源文件名`，例如：
`zip test.zip test.sh` 表示将 `test.sh` 这个文件压缩为 `test.zip`，常用选项为
`-r` 压缩文件夹，例如：`zip -r test.zip test` 表示将 `test` 这个文件夹压缩为 `test.zip`
- `.zip` 格式解压缩命令为：`unzip 压缩包名` 解压缩到当前文件夹
- `.gz` 格式压缩命令为：`gzip 源文件名`，例如：`zip cangls` 表示将 `cangls` 这个文件压缩为 `cangls.gz`。需要注意的是该命令压缩后源文件会消失，并且不适合压缩文件夹。
- `.gz` 格式解压缩命令为：`gzip -d 压缩包名`

以上这两种格式都可以压缩文件，但是！！！在Linux中并不常用，原因是因为他们的姿势实在是不够风骚，用着很不舒服。

为了解决上面的问题，Linux为我们准备了一个打包命令 `tar`，命令格式为：

`tar -cvf 打包后的包名 源文件/文件夹`，同样也对应一个解打包命令：
`tar -xvf 打包后的包名`

- `-c`: 打包
- `-x`: 解包

- `-v`: 显示过程
- `-f`: 指定打包后的文件名

有了这个命令之后我们就可以将一切文件或文件夹先打包再压缩，这样就不会出现文件夹不能压缩的问题了，但是如果真的这样做，总觉得不得劲，难道就不能一步到位吗？肯定是可以的。

压缩和解压缩的终极解决方案：其实 `tar` 命令是支持直接压缩为 `.tar.gz` 格式的，命令为 `tar -zcvf 压缩包名.tar.gz 源文件/文件夹` 与上面相比选项中多了个 `z`，即表示将打包后的文件压缩为 `.gz` 格式的压缩包。同理，`.tar.gz` 格式的压缩包对应的解压命令为 `tar -zxvf 压缩包名.tar.gz`

Tips: `.tar.gz` 也是Linux软件源码包最常见的压缩格式。

3、软件包安装

在Linux中，软件包分为源码包和二进制包（RPM包）两种，其实本来就只有源码包一种，但是由于源码包安装需要经过配置、编译、安装这些繁琐的步骤，安装速度极其慢，而且很容易报错，所以才出现了二进制包，它其实是由源码包配置编译来的。Linux的安装镜像里就已经包含了所有编译好的RPM包供我们直接安装使用，我们只有将系统镜像挂载进来就可以了。

3.1 RPM包手动安装

RPM包安装命令为：`rpm -ivh 包全名`，选项释义如下：

- `-i`: install 安装
- `-v`: verbose 显示详细信息
- `-h`: hash 显示进度

命令很简单，但是过程一点都不简单，下面我们用一个小示例来看一下RPM包的安装过程：

```
[root@localhost /]# cd /media/
[root@localhost media]# ls
CentOS_BuildTag isolinux RPM-GPG-KEY-CentOS-Debug-6
EFI Packages RPM-GPG-KEY-CentOS-Security-6
EULA RELEASE-NOTES-en-US.html RPM-GPG-KEY-CentOS-Testing-6
GPL repodata TRANS.TBL
images RPM-GPG-KEY-CentOS-6
[root@localhost media]# cd Packages/
[root@localhost Packages]# ls |grep http
httpd-2.2.15-59.el6.centos.x86_64.rpm
httpd-devel-2.2.15-59.el6.centos.x86_64.rpm
httpd-manual-2.2.15-59.el6.centos.noarch.rpm
httpd-tools-2.2.15-59.el6.centos.x86_64.rpm
jakarta-commons-httpclient-3.1-0.9.el6_5.x86_64.rpm
[root@localhost Packages]# rpm -ivh httpd-2.2.15-59.el6.centos.x86_64.rpm
warning: httpd-2.2.15-59.el6.centos.x86_64.rpm: Header V3 RSA/SHA1 Signature, key
ID c105b9de: NOKEY
error: Failed dependencies:
        apr-util-ldap is needed by httpd-2.2.15-59.el6.centos.x86_64
        httpd-tools = 2.2.15-59.el6.centos is needed by httpd-2.2.15-59.el6.centos.x86_64
        libapr-1.so.0()(64bit) is needed by httpd-2.2.15-59.el6.centos.x86_64
        libaprutil-1.so.0()(64bit) is needed by httpd-2.2.15-59.el6.centos.x86_64
[root@localhost Packages]#
```

```
[root@localhost Packages]# rpm -ivh apr-util-ldap-1.3.9-3.el6_0.1.x86_64.rpm
warning: apr-util-ldap-1.3.9-3.el6_0.1.x86_64.rpm: Header V3 RSA/SHA256 Signature
, key ID c105b9de: NOKEY
error: Failed dependencies:
        apr-util = 1.3.9-3.el6_0.1 is needed by apr-util-ldap-1.3.9-3.el6_0.1.x86_64
[root@localhost Packages]# rpm -ivh apr-util-1.3.9-3.el6_0.1.x86_64.rpm
warning: apr-util-1.3.9-3.el6_0.1.x86_64.rpm: Header V3 RSA/SHA256 Signature, key
ID c105b9de: NOKEY
error: Failed dependencies:
        libapr-1.so.0()(64bit) is needed by apr-util-1.3.9-3.el6_0.1.x86_64
[root@localhost Packages]# rpm -ivh apr-1.3.9-5.el6_2.x86_64.rpm
warning: apr-1.3.9-5.el6_2.x86_64.rpm: Header V3 RSA/SHA1 Signature, key ID c105b9de: NOKEY
Preparing... #####
1:apr #####
[root@localhost Packages]# rpm -ivh apr-util-1.3.9-3.el6_0.1.x86_64.rpm
warning: apr-util-1.3.9-3.el6_0.1.x86_64.rpm: Header V3 RSA/SHA256 Signature, key
ID c105b9de: NOKEY
Preparing... #####
1:apr-util #####

```

```
[root@localhost Packages]# rpm -ivh apr-util-ldap-1.3.9-3.el6_0.1.x86_64.rpm
warning: apr-util-ldap-1.3.9-3.el6_0.1.x86_64.rpm: Header V3 RSA/SHA256 Signature
, key ID c105b9de: NOKEY
Preparing... ################################################ [100%]
 1:apr-util-ldap ################################################ [100%]
[root@localhost Packages]# rpm -ivh httpd-2.2.15-59.el6.centos.x86_64.rpm
warning: httpd-2.2.15-59.el6.centos.x86_64.rpm: Header V3 RSA/SHA1 Signature, key
ID c105b9de: NOKEY
error: Failed dependencies:
        httpd-tools = 2.2.15-59.el6.centos is needed by httpd-2.2.15-59.el6.centos.x86_64
[root@localhost Packages]# rpm -ivh httpd-tools-2.2.15-59.el6.centos.x86_64.rpm
warning: httpd-tools-2.2.15-59.el6.centos.x86_64.rpm: Header V3 RSA/SHA1 Signature, key
ID c105b9de: NOKEY
Preparing... ################################################ [100%]
 1:httpd-tools ################################################ [100%]
[root@localhost Packages]# rpm -ivh httpd-2.2.15-59.el6.centos.x86_64.rpm
warning: httpd-2.2.15-59.el6.centos.x86_64.rpm: Header V3 RSA/SHA1 Signature, key
ID c105b9de: NOKEY
Preparing... ################################################ [100%]
 1:httpd ################################################ [100%]
[root@localhost Packages]# 
```

从上面的安装过程中我们可以看到，RPM包之间还有依赖关系，而且错综复杂，难以捉摸，所以如果按照这种方式去装软件，那估计软件安装好黄花菜都凉了~

3.2 yum在线安装

为了解决上面RPM包安装过程繁琐的问题，Linux将所有软件包放到官方服务器上，并且开发了一个yum安装工具，当进行yum在线安装的时候，它可以自动解决依赖问题。类似于iOS的CocoaPods，Java的Maven，前端的npm一样。

yum源 配置文件

yum是Redhat系列的Linux自带的软件安装工具（Debian系列是apt-get），yum源就是指存放所有软件包的服务器地址，它的默认配置文件为：

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

```

# CentOS-Base.repo
#
# The mirror system uses the connecting IP address of the client and the
# update status of each mirror to pick mirrors that are updated to and
# geographically close to the client. You should use this for CentOS updates
# unless you are manually picking other mirrors.
#
# If the mirrorlist= does not work for you, as a fall back you can try the
# remarked out baseurl= line instead.
#
#
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6

#released updates
[updates]
name=CentOS-$releasever - Updates
@
"CentOS-Base.repo" 52L, 1991C

```

1,1

顶端

配置文件中的每一段内容都相似，具体字段说明如下表：

字段名称	描述
[base]	容器名称，一定要放在[]中
name	容器说明，内容自己随意
mirrorlist	镜像站点，与baseurl二选一，即软件包的服务器地址
baseurl	yum源服务器地址，默认是CentOS官方yum源，国内可能速度较慢，建议换成国内的 网易镜像源 或者 阿里镜像源
enabled	设置此容器是否生效，省略不写或写成enabled=1都是生效，写成enabled=0则不生效
gpgcheck	RPM的数字证书是否生效，1表示生效，0表示不生效
gpgkey	数字证书的公钥文件保存位置，不要修改。

光盘搭建本地yum源

很多时候在网络的情况下我们使用yum在线源来安装软件就可以了，但是偏偏他就有这么一种情况，没有网络！！那么这个时候我们又想起了我们的系统镜像光盘里其实已经有所有我们需要的软件包了，那么它不就相当于是一个yum源了吗？没错，确实是这样的，所以我们可以使用光盘来做本地yum源，步骤如下：

- 挂载光盘: `mount /dev/sr0 /media`
- 使网络yum源失效: 在上面配置文件介绍的时候我们知道, 只需要将配置文件里容器的 enabled值设置为0就可以了, 但是默认配置文件中有好几个容器, 每个都设置一遍比较麻烦, 所以既然yum源的默认配置文件为 `/etc/yum.repos.d/CentOS-Base.repo`, 那么如果我将这个文件删掉它就不起作用了吧? 这样是可以, 但是我们说对于系统默认配置文件我们要本着只备份不删除的原则, 所以我们只需要将这个文件改个名备份一下即可 `mv CentOS-Base.repo CentOS-Base.repo.bak`

- 使光盘yum源生效: 修改光盘的yum配置文件

`/etc/yum.repos.d/CentOS-Media.repo`, 如下图:

```
# CentOS-Media.repo
#
# This repo can be used with mounted DVD media, verify the mount point for
# CentOS-6. You can use this repo and yum to install items directly off the
# DVD ISO that we release.
#
# To use this repo, put in your DVD and use it with the other repos too:
# yum --enablerepo=c6-media [command]
#
# or for ONLY the media repo, do this:
#
# yum --disablerepo=\* --enablerepo=c6-media [command]

[c6-media]
name=CentOS-$releasever - Media
baseurl=file:///media/ 真实光盘挂载点
#       file:///media/cdrom/
#       file:///media/cdrecorder/ 不需要的注释掉
gpgcheck=1 设置enabled=1
enabled=1 使其生效
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
~
~
~

"CentOS-Media.repo" 21L, 625C 已写入 20,9 全部
```

完成上述步骤后执行 `yum list` 即可验证本地yum源是否生效。

通过yum安装软件

首先, 我们可以通过 `yum search 关键字` 来查询yum源里是否有需要的软件包。

yum安装命令为: `yum -y install 包名`, 其中选项 `-y` 表示自动回答yes, 例如:

`yum -y install gcc`

上面我们使用rpm命令手动艰难的安装了httpd (也就是Apache服务器软件) 的主包和 `httpd-tools`, 还有 `httpd-manual` 文档包和 `httpd-devel` 开发包没有安装, 我们使用 yum安装 `yum -y install httpd-manual httpd-devel`

Apache是个急性子, 安装好就自动启动了。

RPM包安装的Apache手动启动的方式有两种: 一种是安装好的可执行命令启

启动: `/usr/sbin/httpd -k start` 或者 `/usr/sbin/apachectl start`, 另外一种是通过Linux服务启动: `service httpd restart`

3.3 源码包安装

源码包安装时一定要安装在指定位置，因为通过源码包安装的软件没有卸载命令，所以如果你想卸载的时候只要把指定位置下的文件全部删掉就可以了，安装位置一般是：

/usr/local/软件名/

使用源码包安装之前必须保证系统已安装 `gcc` :

```
[root@localhost /]# rpm -qa |grep gcc  
libgcc-4.4.7-18.el6.x86_64  
gcc-4.4.7-18.el6.x86_64  
[root@localhost /]#
```

下载源码包，一般Linux的软件包都有对应的官方网站下载地址，我们下面以Apache软件来讲解，Apache 2.4下载地址为：<http://mirror.bit.edu.cn/apache/httpd/httpd-2.4.37.tar.gz>，执行

```
 wget http://mirror.bit.edu.cn/apache/httpd/httpd-2.4.37.tar.gz 即可将软件源码包下载到当前路径下。
```

```
95% [=====>] 8,723,616 408K/s eta(英国中部时  
95% [=====>] 8,761,600 408K/s eta(英国中部时  
96% [=====>] 8,873,376 412K/s eta(英国中部时  
97% [=====>] 8,974,720 408K/s eta(英国中部时  
99% [=====>] 9,096,576 405K/s eta(英国中部时  
100%[=====>] 9,177,278 404K/s in 18s  
  
2018-10-27 09:50:11 (489 KB/s) - 已保存 “httpd-2.4.37.tar.gz” [9177278/9177278]  
  
[root@localhost opt]# ls  
httpd-2.4.37.tar.gz  rh
```

1. 由于Apache 2.4依赖Apache的运行环境下的两个包 `apr` 和 `apr-util`
2. 所以采用同样的wget方式将其下载下来
3. 另外，Apache编译时还依赖了 `pcre-config`，它属于 `pcre-devel`，所以我们要使用 `yum -y install pcre-devel` 命令安装好这个依赖包

全部下载完成后将其一一解压缩到下载的目录下：

```
[root@localhost src]# ls  
httpd-2.4.37.tar.gz  
[root@localhost src]# tar -zxvf httpd-2.4.37.tar.gz  
httpd-2.4.37/  
httpd-2.4.37/config.layout  
httpd-2.4.37/configure.in  
httpd-2.4.37/Makefile.win  
httpd-2.4.37/configure  
httpd-2.4.37/test/  
httpd-2.4.37/test/test_parser.c  
httpd-2.4.37/test/check_chunked  
httpd-2.4.37/test/make_sni.sh  
httpd-2.4.37/test/README
```

解压完成之后，首先我们要安装的是 `apr`，所以需要进入到解压后的 `apr-1.6.5` 目录下，然后依次执行：

- `./configure --prefix=/usr/local/apr`：软件配置与检查，主要是将软件安装的功能选项和系统环境信息写入Makefile，为后续的make做准备，**这里 `--prefix=PATH` 一定不能省略**；
- `make`：编译，这个不用多说。清空编译缓存执行 `make clean`；
- `make install`：安装；

安装完成之后，在 `/usr/local/apr` 下就生成了相应的文件。

然后我们需要安装的是 `apr-util`，所以需要进入到解压后的 `apr-util-1.6.1` 目录下，然后依次执行：

- `./configure --prefix=/usr/local/apr-util --with-apr=/usr/local/apr` :
apr-util 编译时要依赖 apr , 所以这里的 `--with-apr=PATH` 是用来指定编译 apr-util 时依赖的 apr 使用哪一个;
- `make` : 编译;
- `make install` : 安装;

安装完成之后，在 `/usr/local/apr-util` 下就生成了相应的文件。

最后，我们进入解压好的 `httpd-2.4.37` 目录下：

```
httpd-2.4.37/docs/manual/urlmapping.html
httpd-2.4.37/buildconf
httpd-2.4.37/Makefile.in
httpd-2.4.37/srclib/
httpd-2.4.37/srclib/Makefile.in
[root@localhost src]# ls
httpd-2.4.37 httpd-2.4.37.tar.gz
[root@localhost src]# cd httpd-2.4.37
[root@localhost httpd-2.4.37]# ls
ABOUT_APACHE buildconf httpd.dsp libhttpd.mak README.cmake
acinclude.m4 CHANGES httpd.mak LICENSE README.platforms
Apache-apr2.dsw CMakeLists.txt httpd.spec Makefile.in ROADMAP
Apache.dsw config.layout include Makefile.win server
apache_probes.d configure INSTALL modules srclib
ap.d configure.in InstallBin.dsp NOTICE support
build docs LAYOUT NWGNUMakefile test
BuildAll.dsp emacs-style libhttpd.dep libhttpd.dsp os VERSIONING
BuildBin.dsp httpd.dep
[root@localhost httpd-2.4.37]#
```

依次执行：

- `./configure --prefix=/usr/local/apache2 --with-apr=/usr/local/apr --with-apr-util=/usr/local/apr-util`
：编译时要依赖 apr 和 apr-util ;
- `make` : 编译;
- `make install` : 安装;

安装完成之后可以看到如下效果：

```
[root@localhost src]# ls
apr-1.6.5      apr-util-1.6.1      httpd-2.4.37
apr-1.6.5.tar.gz  apr-util-1.6.1.tar.gz  httpd-2.4.37.tar.gz
[root@localhost src]# cd ../apache2/
[root@localhost apache2]# ls
bin  cgi-bin  error  icons  logs  manual
build  conf  htdocs  include  man  modules
[root@localhost apache2]# ll bin/
总用量 2532
-rwxr-xr-x. 1 root root  82409 10月 27 11:26 ab
-rwxr-Xr-x. 1 root dip   3441 10月 27 11:24 apachectl
-rwxr-xr-x. 1 root dip   23516 10月 27 11:24 apxs
-rwxr-xr-x. 1 root root  12245 10月 27 11:26 checkgid
-rwxr-xr-x. 1 root dip   8925 10月 27 11:24 dbmmanage
-rw-r--r--. 1 root dip   1075 10月 27 11:24 envvars
-rw-r--r--. 1 root dip   1075 10月 27 11:24 envvars-std
-rwxr-xr-x. 1 root root  20344 10月 27 11:26 fcgistarter
-rwxr-xr-x. 1 root root  66594 10月 27 11:26 htcacheclen
-rwxr-xr-x. 1 root root  40642 10月 27 11:26 htdbm
-rwxr-xr-x. 1 root root  22447 10月 27 11:26 htdigest
-rwxr-xr-x. 1 root root  41746 10月 27 11:26 htpasswd
-rwxr-xr-x. 1 root root 2157074 10月 27 11:26 httpd
-rwxr-xr-x. 1 root root  19456 10月 27 11:26 httxt2dbm
-rwxr-xr-x. 1 root root  22068 10月 27 11:26 logresolve
-rwxr-xr-x. 1 root root  37480 10月 27 11:26 rotatelogs
```

执行 `/usr/local/apache2/bin/apachectl start` 即可开启Apache服务器，通过浏览器访问效果如下（关闭防火墙）：



八、Linux 防火墙 (iptables)

1、关于防火墙

首先，什么是防火墙？从逻辑上讲，防火墙可以大体分为主机防火墙和网络防火墙。主机防火墙针对于单个主机进行防护，而网络防火墙往往是处于网络入口或边缘，针对网络入口进行防护。网络防火墙和主机防火墙并不冲突，可以理解为网络防火墙主外（集体），主机防火墙主内（个人）。从物理上讲，防火墙又可分为硬件防火墙和软件防火墙。硬件防火墙在硬件级别就实现了部分防火墙功能，另一部分靠软件实现，性能高，成本也高。软件防火墙就是通过一些软件处理逻辑运行于通用硬件平台上的防火墙，性能低，成本也低。



具备 FirePOWER 服务的 ASA 5500-X

- 适用于中小企业和分支机构
- 防火墙吞吐量：256 Mbps 至 1750 Mbps
- 威胁检测吞吐量：125 Mbps 到 1250 Mbps
- 状态防火墙、应用可视性与可控性、NGIPS、高级恶意软件保护、URL 过滤



Firepower 4100 系列

- 互联网边缘、高性能环境
- 防火墙吞吐量：12 Gbps 到 30 Gbps
- 威胁检测吞吐量：10 Gbps 到 24 Gbps
- 状态防火墙、AVC、NGIPS、AMP、URL



Firepower 2100 系列

- 适用于互联网边缘到数据中心环境
- 防火墙吞吐量：2.0 Gbps 至 8.5 Gbps
- 威胁检测吞吐量：2.0 到 8.5 Gbps
- 状态防火墙、应用可视性与可控性、NGIPS、高级恶意软件保护、URL 过滤



Firepower 9000 系列

- 适用于运营商和数据中心环境
- 防火墙吞吐量：最高 225 Gbps
- 威胁检测吞吐量：最高 90 Gbps
- 状态防火墙、应用可视性与可控性、NGIPS、高级恶意软件保护、URL 过滤、DDoS

2、关于iptables



iptables 其实并不是真正的防火墙，我们可以把它理解成客户端代理，用户通过这个代理将安全设定执行到对应的 **安全框架** 中，这个 **安全框架** 才是真正的防火墙——**netfilter**。由于 **netfilter** 是位于内核空间的，而 **iptables** 是位于用户空间的一个命令，所以我们用它来操作 **netfilter**。

在CentOS中对应的命令和服务就是 **iptables**，可以使用 `service iptables start` 来启动 **iptables** 服务，那么对应的就有 `service iptables stop` 停止和

```
service iptables restart 重启 iptables 服务。
```

3、iptables规则组成

可以使用 `iptables -L` 命令查看系统当前的iptables规则，如下图：

```
[root@localhost /]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere           state NEW tcp dpt:ssh
REJECT     all  --  anywhere             anywhere           reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
REJECT     all  --  anywhere             anywhere           reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@localhost /]#
```

其实这段规则不太看得懂，因为它是来自于一个文件配置好系统识别的，iptables配置文件的位置为：`/etc/sysconfig/iptables`

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.

*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

要想看得懂这个文件，首先我们就要来看一下iptables的组成规则，iptables由 **四张表 + 五条链 (hook point) + 规则** 组成，释义如下：

table	command	chain	Parameter & Xmatch	target
-t filter nat	-A -D -L -F ...	INPUT FORWARD OUTPUT PREROUTING POSTROUTING	-p tcp -s -d --sport --dport --dports -m tcp state	-j ACCEPT DROP REJECT DNAT SNAT

看着这些规则可能还是不知道这个东西到底怎么配置，下面我们来讲一个场景：

我们之前启动了Apache服务器之后需要把iptables防火墙关闭才能正常访问，但是这种操作跟“删库跑路”都一个性质的，肯定不能被允许，那么怎么办呢？这个时候我们就要来修改我们的iptables规则，实际上外面访问我们的Apache服务器时是访问我们的80端口，但是从默认的规则里我们看到并没有允许80端口的访问，所以我们可以采用命令

`iptables -I INPUT -p tcp --dport 80 -j ACCEPT` 来插入一条iptables规则，让80端口能够被访问。当然，我们也可以直接修改iptables对应的配置文件：

```

# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT

```

其中 `-I` 表示在最前面插入，`-A` 表示在最后面追加，`iptables`读取是按配置顺序的。

`iptables` 规则可以说是非常强大，这里我们只做简单的端口拦截与放行使用介绍，后面用到

的时候再做相关补充。

实践篇

一、FTP服务器（vsftpd）

1、关于vsftpd

vsftpd是用于UNIX系统（包括Linux）的GPL许可的FTP服务器。它安全、稳定且速度极快。vsftpd是一个成熟且值得信赖的文件传输解决方案。更多请查看：[vsftpd官网](#)

2、安装并启动FTP服务

使用 `yum` 安装 `vsftpd`，执行 `yum install -y vsftpd`：

```
[root@localhost ~]# yum install -y vsftpd
已加载插件: fastestmirror, refresh-packagekit, security
设置安装进程
Loading mirror speeds from cached hostfile
解决依赖关系
--> 执行事务检查
--> Package vsftpd.x86_64 0:2.2.2-24.el6 will be 安装
--> 完成依赖关系计算

依赖关系解决

=====
软件包           架构       版本          仓库      大小
正在安装：
vsftpd           x86_64     2.2.2-24.el6   base      156 k

事务概要
=====
Install 1 Package(s)

总下载量: 156 k
Installed size: 340 k
下载软件包:
vsftpd-2.2.2-24.el6.x86_64.rpm
运行 rpm_check_debug
执行 事务 测试
事务 测试 成功
执行 事务
正在安装  : vsftpd-2.2.2-24.el6.x86_64
Verifying  : vsftpd-2.2.2-24.el6.x86_64
1/1
1/1

已安装:
vsftpd.x86_64 0:2.2.2-24.el6

完毕!
[root@localhost ~]#
```

安装完成后，启动FTP服务，执行 `service vsftpd start`：

```
[root@localhost ~]# service vsftpd start
Starting vsftpd for vsftpd: [ OK ]
[root@localhost ~]#
```

成果启动之后，可以看到系统已经监听了21端口 `netstat -ntlp | grep 21`：

```
[root@localhost ~]# netstat -ntlp | grep 21
tcp        0      0 0.0.0.0:21          0.0.0.0:*              LISTEN      3504/vsftpd
[root@localhost ~]#
```

此时，访问 `ftp://192.168.231.128` 就可以浏览机器上的 `/var/ftp` 目录了。



3、配置FTP权限

vsftpd的配置目录为 `/etc/vsftpd`，包含下列配置文件：

- `vsftpd.conf` 为主要配置文件
- `ftpusers` 配置 禁止 访问FTP服务的用户列表
- `user_list` 配置用户访问控制

首先，我们需要阻止匿名访问和切换根目录。匿名访问和切换根目录都会给服务器带来安全风险，我们把这两个功能关闭，编辑 `/etc/vsftpd/vsftpd.conf`，找到下面两处并修改：

```
# 禁用匿名用户
anonymous_enable=NO
# 禁止切换根目录
chroot_local_user=YES
```

```
# Example config file /etc/vsftpd/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
# Allow anonymous FTP? (Beware - allowed by default if you comment this out)
#anonymous_enable=YES
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
```

```
# You may specify an explicit list of local users to chroot() to their home
# directory. If chroot_local_user is YES, then this list becomes a list of
# users to NOT chroot().
chroot_local_user=YES
#chroot_list_enable=YES
# (default follows)
#chroot_list_file=/etc/vsftpd/chroot_list
"
```

除了以上几个简单配置之外，补充说明一点：

如果线上防火墙想要设置的很严格的话，那么我们的FTP还需要配置传输模式与端口：

```
# 设置为被动传输模式
pasv_enable=YES
# 设置数据传输最小端口
pasv_min_port=61001
# 设置数据传输最大端口
pasv_max_port=62000
```

关于FTP的传输模式：

- 主动模式 (*PORT*, *pasv_enable=NO*) 的连接过程是：客户端向服务器的FTP端口（默认21）发送连接请求，服务器接受连接，建立一条命令链路。当需要传送数据时，客户端在命令链路上用*PORT*命令告诉服务器：“我打开了xx端口，你过来连接我”，于是服务器从20端口向客户端的xx端口发送连接请求，建立一条数据链路来传送数据。

- 被动模式 (*PASV*, *pasv_enable=YES*) 的连接过程是：客户端向服务器的 FTP 端口（默认是 21）发送连接请求，服务器接受连接，建立一条命令链路。当需要传送数据时，服务器在命令链路上用 *PASV* 命令告诉客户端：“我打开了 xx 端口，你过来连接我”。于是客户端向服务器的 xx 端口发送连接请求，建立一条数据链路来传送数据。
从上面可以看出，两种方式的命令链路连接方法是一样的，而数据链路的建立方法就完全不同。而 FTP 的复杂性就在于此。

然后，我们来创建一个专门用于访问 FTP 服务的用户，用户名叫 `ftpuser`：

```
useradd ftpuser
```

为用户 `ftpuser` 设置密码：

```
echo "ftpuser" | passwd ftpuser --stdin
```



```
[root@localhost /]# useradd ftpuser
正在创建信箱文件：文件已存在
[root@localhost /]# echo "ftpuser" | passwd ftpuser --stdin
更改用户 ftpuser 的密码。
passwd: 所有的身份验证令牌已经成功更新。
[root@localhost /]#
```

限制该用户仅能通过 FTP 访问服务器，而不能直接登陆服务器：

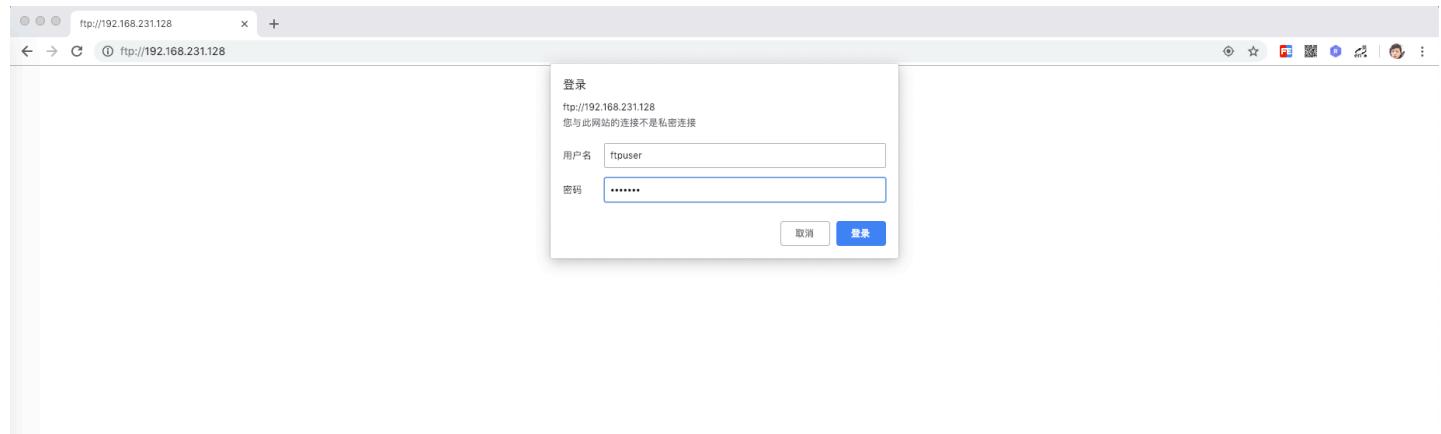
```
usermod -s /sbin/nologin ftpuser
```

为用户分配主目录为 `/var/ftp`，该目录不可上传文件，文件只能上传到 `/var/ftp/pub` 目录下：

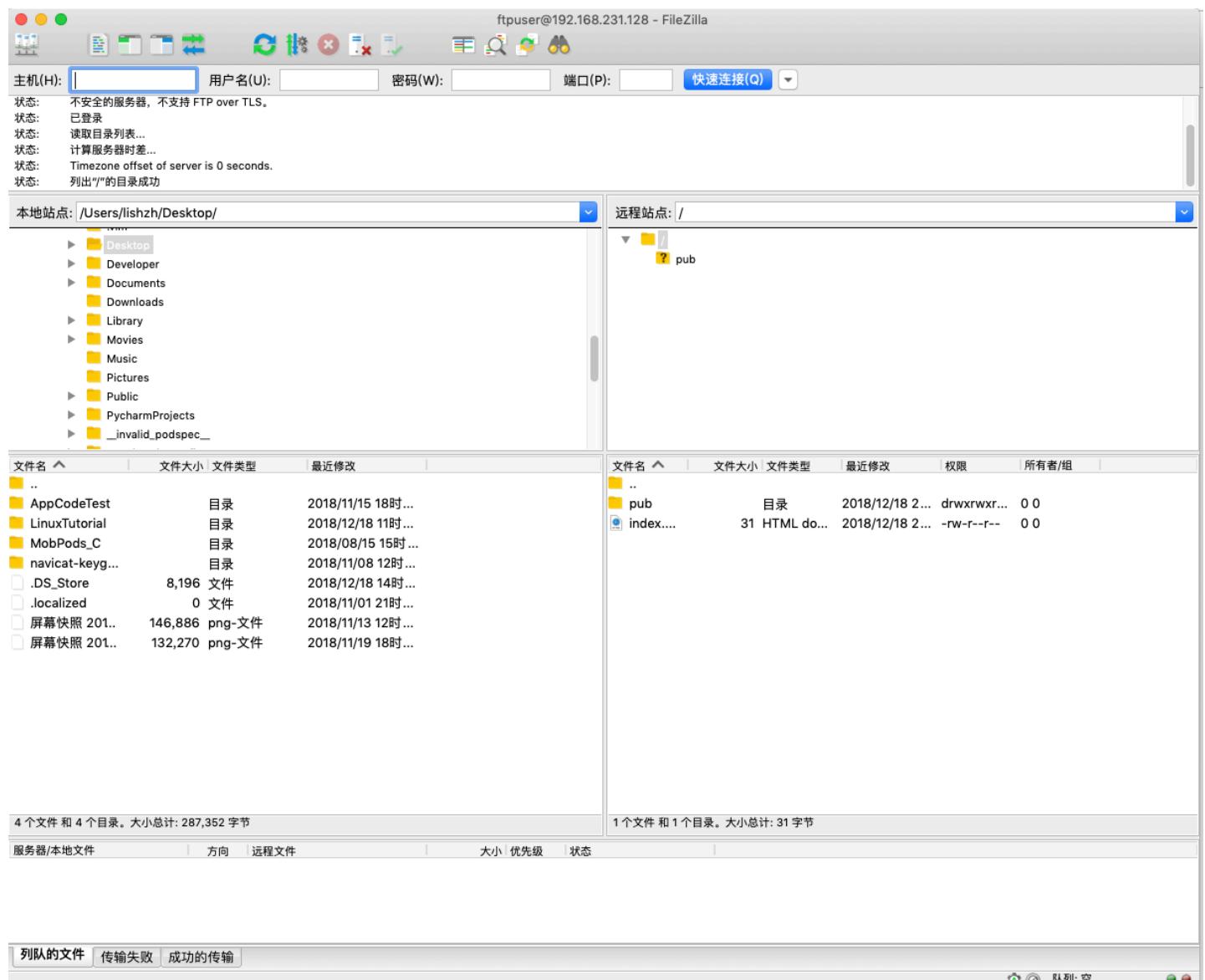
```
# 设置用户的主目录
usermod -d /var/ftp ftpuser
# 设置访问权限
chmod a-w /var/ftp && chmod -R 777 /var/ftp/pub
# 创建登陆欢迎文件
echo "Welcome to use my FTP service." > /var/ftp/index.html
# 重启FTP服务
service vsftpd restart
```

4、访问 FTP 服务

FTP服务搭建完成之后我们可以使用浏览器来直接访问FTP服务：



或者，Mac上我推荐使用 [FileZilla](#) 这个FTP工具来连接FTP服务：

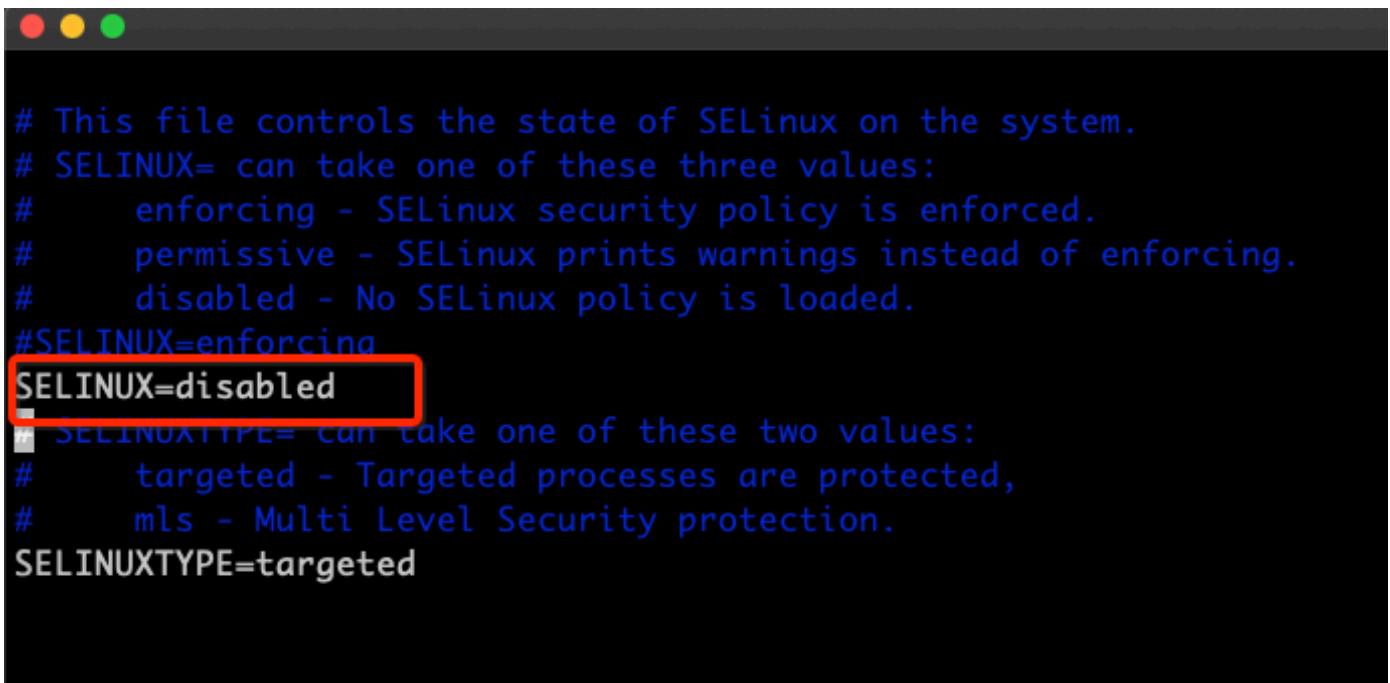


如果出现无法上传文件的错误, 请关闭 [SELinux](#) 安全:

1、临时关闭(不用重启)

```
# 查看SELinux状态  
sestatus -v  
# 关闭  
setenforce 0
```

2、修改配置文件 [/etc/sysconfig/selinux](#) (需要重启) :



```
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#       enforcing - SELinux security policy is enforced.  
#       permissive - SELinux prints warnings instead of enforcing.  
#       disabled - No SELinux policy is loaded.  
#SELINUX=enforcing  
SELINUX=disabled  
# SELINUXTYPE= can take one of these two values:  
#       targeted - Targeted processes are protected,  
#       mls - Multi Level Security protection.  
SELINUXTYPE=targeted
```

所以常规做法是两个一起操作!

二、Web服务器 (Tomcat)

说到Tomcat就不得不提一下强大的Java开发语言, 以示尊重。所以在正式开始介绍Tomcat之前, 我们先来安装一下Java开发环境JDK (Java Development Kit)。

1、JDK下载安装

打开 [Oracle官网](#) 找到 [Java SE下载页](#), 当前最新版本是 JDK 11, 但是我们依然不选择这个版本, 还是那句线上我们要本着“猥琐发育别浪的原则”, 所以我们选择下面的 JDK 8。

Java SE
Java EE
Java ME
Java SE Subscription
Java Embedded
Java Card
Java TV
Community
Java Magazine

Overview Downloads Documentation Community Technologies Training

Java SE Downloads



Java Platform (JDK) 11

Java Platform, Standard Edition

Java SE 11.0.1(LTS)
Java SE 11.0.1 is the latest release for the Java SE 11 Platforms
[Learn more](#)

- Installation Instructions
- Release Notes
- Oracle JDK License
- Java SE Licensing Information User Manual
 - Includes Third Party Licenses
- Certified System Configurations
- Readme

Oracle JDK DOWNLOAD

Looking for Oracle OpenJDK builds?

- **Oracle Customers and ISVs targeting Oracle LTS releases:** Oracle JDK is Oracle's supported Java SE version for customers and for developing, testing, prototyping or demonstrating your Java applications.
- **End users and developers looking for free JDK versions:** [Oracle OpenJDK](#) offers the same features and performance as Oracle JDK under the [GPL license](#).

To Learn more about these options visit [Oracle JDK Releases for Java 11 and Later](#)

Java SE 8u191 / Java SE 8u192
Java SE 8u191 / Java SE 8u192 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.
[Learn more](#)

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Licensing Information User Manual
 - Includes Third Party Licenses
- Certified System Configurations
- Readme Files

JDK DOWNLOAD

Server JRE DOWNLOAD

点击“DOWNLOAD”之后就看到了如下界面：

Java SE
Java EE
Java ME
Java SE Subscription
Java Embedded
Java Card
Java TV
Community
Java Magazine

Overview

Downloads

Documentation

Community

Technologies

Training

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day hands-on workshops \(free\)](#) and other events
- [Java Magazine](#)

JDK 8u191 [checksum](#)

JDK 8u192 [checksum](#)

Java SDKs and Tools

- ❖ [Java SE](#)
- ❖ [Java EE and Glassfish](#)
- ❖ [Java ME](#)
- ❖ [Java Card](#)
- ❖ [NetBeans IDE](#)
- ❖ [Java Mission Control](#)

Java Resources

- ❖ [Java APIs](#)
- ❖ [Technical Articles](#)
- ❖ [Demos and Videos](#)
- ❖ [Forums](#)
- ❖ [Java Magazine](#)
- ❖ [Developer Training](#)
- ❖ [Tutorials](#)
- ❖ [Java.com](#)

Java SE Development Kit 8u191

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.89 MB	jdk-8u191-linux-i586.rpm
Linux x86	185.69 MB	jdk-8u191-linux-i586.tar.gz
Linux x64	167.99 MB	jdk-8u191-linux-x64.rpm
Linux x64	182.87 MB	jdk-8u191-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u191-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	jdk-8u191-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.28 MB	jdk-8u191-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.04 MB	jdk-8u191-solaris-x64.tar.Z
Solaris x64	92.13 MB	jdk-8u191-solaris-x64.tar.gz
Windows x86	197.34 MB	jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	jdk-8u191-windows-x64.exe

Java SE Development Kit 8u191 Demos and Samples

Downloads

You must accept the [Oracle BSD License](#) to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	9.05 MB	jdk-8u191-linux-arm32-vfp-hflt-demos.tar.gz
Linux ARM 64 Hard Float ABI	9.05 MB	jdk-8u191-linux-arm64-vfp-hflt-demos.tar.gz
Linux x86	55.92 MB	jdk-8u191-linux-i586-demos.rpm
Linux x86	55.78 MB	jdk-8u191-linux-i586-demos.tar.gz
Linux x64	56.02 MB	jdk-8u191-linux-x64-demos.rpm
Linux x64	55.88 MB	jdk-8u191-linux-x64-demos.tar.gz
Mac OS X	56.04 MB	jdk-8u191-macosx-x86_64-demos.zip
Solaris SPARC 64-bit	12.27 MB	jdk-8u191-solaris-sparcv9-demos.tar.Z

这里首先我们要选择接受证书许可（官方，你懂的，不接受也不行），然后选择我们需要的“Linux x64”版本对应的rpm包，下载完成后可以使用我们上面说到的FTP工具传输到我们的Linux服务器上。但是！！！难道就不能直接在Linux系统中去下载吗？？？

于是乎我们就选择了常规操作，将我们要下载的rpm包对应的链接地址复制下来：

Java SE
Java EE
Java ME
Java SE Subscription
Java Embedded
Java Card
Java TV
Community
Java Magazine

[Overview](#) [Downloads](#) [Documentation](#) [Community](#) [Technologies](#) [Training](#)

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day hands-on workshops \(free\)](#) and other events
- [Java Magazine](#)

JDK 8u191 [checksum](#)

JDK 8u192 [checksum](#)

Java SE Development Kit 8u191

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.89 MB	jdk-8u191-linux-i586.rpm
Linux x86	185.69 MB	jdk-8u191-linux-i586.tar.gz
Linux x64	167.99 MB	jdk-8u191-linux-x64.rpm
Linux x64	182.87 MB	jdk-8u191-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u191-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	jdk-8u191-solaris-sparc-svr4-jdk-package.tar.gz
Solaris SPARC 64-bit	94.28 MB	jdk-8u191-solaris-sparc-svr4-jdk.tar.gz
Solaris x64 (SVR4 package)	134.04 MB	jdk-8u191-solaris-sparc-svr4-jdk-package.tar.gz
Solaris x64	92.13 MB	jdk-8u191-solaris-sparc-svr4-jdk.tar.gz
Windows x86	197.34 MB	jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	jdk-8u191-windows-x64.exe

Java SE Development Kit 8u191 Demos Downloads

You must accept the [Oracle BSD License](#) to download

Accept License Agreement Download

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	9.05 MB	jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	9.05 MB	jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	55.92 MB	jdk-8u191-linux-i586.rpm
Linux x86	55.78 MB	jdk-8u191-linux-i586.tar.gz
Linux x64	56.02 MB	jdk-8u191-linux-x64.rpm

在新标签页中打开链接

在新窗口中打开链接

在隐身窗口中打开链接

链接存储为

复制链接地址

复制

使用Google搜索“jdk-8u191-linux-x64.rpm”

打印...

当前连接生成二维码

翻译“jdk-8u191-linux-x64.rpm”

朗读所选文本

FeHelper工具

检查

语音

服务

大概是这样的

<https://download.oracle.com/otn-pub/java/jdk/8u191-b12/2787e4a523244c269598db4e85c51e0c/jdk-8u191-linux-x64.rpm>

，然后使用 `wget` 命令来下载：

```
 wget https://download.oracle.com/otn-pub/java/jdk/8u191-b12/2787e4a523244c269598db4e85c51e0c/jdk-8u191-linux-x64.rpm
```

于是命令执行也正常了，结果也出来了：

```
[root@localhost tmp]# wget https://download.oracle.com/otn-pub/java/jdk/8u191-b12/2787e4a523244c269598db4e85c51e0c/jdk-8u191-linux-x64.rpm
--2018-12-19 03:18:09-- https://download.oracle.com/otn-pub/java/jdk/8u191-b12/2787e4a523244c269598db4e85c51e0c/jdk-8u191-linux-x64.rpm
正在解析主机 download.oracle.com... 23.33.248.114
正在连接 download.oracle.com|23.33.248.114|:443... 已连接。
已发出 HTTP 请求, 正在等待回应... 302 Moved Temporarily
位置: https://edelivery.oracle.com/otn-pub/java/jdk/8u191-b12/2787e4a523244c269598db4e85c51e0c/jdk-8u191-linux-x64.rpm [跟随至新的 URL]
--2018-12-19 03:18:11-- https://edelivery.oracle.com/otn-pub/java/jdk/8u191-b12/2787e4a523244c269598db4e85c51e0c/jdk-8u191-linux-x64.rpm
正在解析主机 edelivery.oracle.com... 184.26.250.202, 2600:1417:76:181::2d3e, 2600:1417:76:19a::2d3e
正在连接 edelivery.oracle.com|184.26.250.202|:443... 已连接。
已发出 HTTP 请求, 正在等待回应... 302 Moved Temporarily
位置: http://download.oracle.com/errors/download-fail-1505220.html [跟随至新的 URL]
--2018-12-19 03:18:11-- http://download.oracle.com/errors/download-fail-1505220.html
正在连接 download.oracle.com|23.33.248.114|:80... 已连接。
已发出 HTTP 请求, 正在等待回应... 301 Moved Permanently
位置: https://download.oracle.com/errors/download-fail-1505220.html [跟随至新的 URL]
--2018-12-19 03:18:12-- https://download.oracle.com/errors/download-fail-1505220.html
正在连接 download.oracle.com|23.33.248.114|:443... 已连接。
已发出 HTTP 请求, 正在等待回应... 200 OK
长度: 5307 (5.2K) [text/html]
正在保存至: "jdk-8u191-linux-x64.rpm"

100%[=====] 5,307          --.-K/s   in 0s

2018-12-19 03:18:13 (25.6 MB/s) - 已保存 “jdk-8u191-linux-x64.rpm” [5307/5307]

[root@localhost tmp]# ll
总用量 12
-rw-r--r--. 1 root    root    5307 3月 20 2012 jdk-8u191-linux-x64.rpm
drwx-----. 3 lishzh lishzh 4096 11月 20 03:03 yumlishzh-JhsMf_
[root@localhost tmp]# ll -h
总用量 12K
-rw-r--r--. 1 root    root    5.2K 3月 20 2012 jdk-8u191-linux-x64.rpm
drwx-----. 3 lishzh lishzh 4.0K 11月 20 03:03 yumlishzh-JhsMf_
[root@localhost tmp]#
```

可是我们看到怎么就5.2k的大小呢？这很明显不对啊，所以我们来查看一下这个文件的内容：

```
root@localhost:/var/tmp
<html>
<head>
<title>Unauthorized Request</title>
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
<link rel="stylesheet" type="text/css" href="/errors/us/assets/hp-styles.css" />
<link rel="stylesheet" type="text/css" href="/errors/us/assets/master-master.css" />

<body style="margin: 0px" bgcolor="#ffffff">
<div id="banner">
  <table width="100%" border="0" cellspacing="0" cellpadding="0">
    <tr>
      <td rowspan="2" valign="middle" nowrap><a href="http://www.oracle.com"></a></td>
      <td align="right" valign="top" width="70%" nowrap class="padMid"><div id="bannerMid
"> </div></td>

      <td width="30%" align="left" valign="bottom" nowrap></td>
    </tr>
    <tr>
      <td align="right" valign="bottom" nowrap class="padMid"></td>
      <td width="30%" align="left" valign="bottom" nowrap></td>
    </tr>
  </table>
</div>
<!-- Header END //-->
<table BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH="100%">

  <tr>
```

搜噶，看到这里就清楚了，其实这并没有下载我们需要的JDK，而只是下载了对应的网页源代码，那么问题来了，怎么下载我们需要的JDK安装包呢？？

请注意下面我要进行一波骚操作了，这个时候我们选择让浏览器去下载，但是当浏览器开始下载的时候我们将其 暂停 下来，然后选择 显示全部：



Java SE
Java EE
Java ME
Java SE Subscription
Java Embedded
Java Card
Java TV
Community
Java Magazine

Overview

Downloads

Documentation

Community

Technologies

Training

Java SDKs and Tools

- [Java SE](#)
- [Java EE and Glassfish](#)
- [Java ME](#)
- [Java Card](#)
- [NetBeans IDE](#)
- [Java Mission Control](#)

Java Resources

- [Java APIs](#)
- [Technical Articles](#)
- [Demos and Videos](#)
- [Forums](#)
- [Java Magazine](#)
- [Developer Training](#)
- [Tutorials](#)
- [Java.com](#)

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day hands-on workshops \(free\)](#) and other events
- [Java Magazine](#)

[JDK 8u191 checksum](#)[JDK 8u192 checksum](#)**Java SE Development Kit 8u191**

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.89 MB	jdk-8u191-linux-i586.rpm
Linux x86	185.69 MB	jdk-8u191-linux-i586.tar.gz
Linux x64	167.99 MB	jdk-8u191-linux-x64.rpm
Linux x64	182.87 MB	jdk-8u191-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u191-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	jdk-8u191-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.28 MB	jdk-8u191-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.04 MB	jdk-8u191-solaris-x64.tar.Z
Solaris x64	92.13 MB	jdk-8u191-solaris-x64.tar.gz
Windows x86	197.34 MB	jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	jdk-8u191-windows-x64.exe

Java SE Development Kit 8u191 Demos and Samples Downloads

You must accept the [Oracle BSD License](#) to download this software.



Accept License Agreement



Decline License Agreement

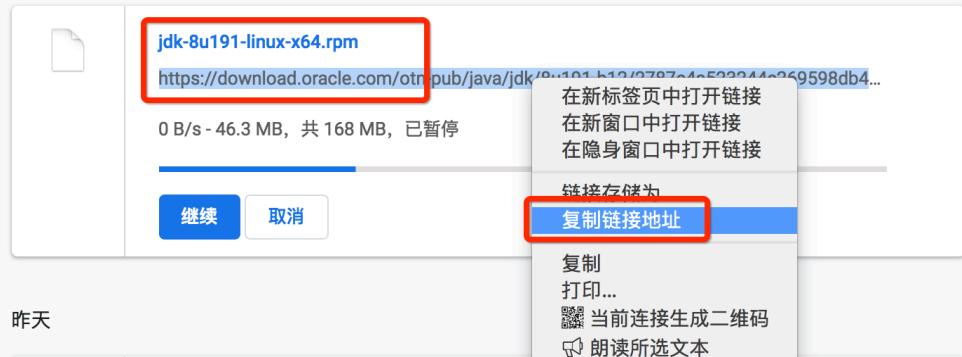
Product / File Description	File Size	Download
完成后打开	9.05 MB	jdk-8u191-linux-arm32-vfp-hflt-demos.tar.gz
总是打开此类文件	9.05 MB	jdk-8u191-linux-arm64-vfp-hflt-demos.tar.gz
继续	55.92 MB	jdk-8u191-linux-i586-demos.rpm
在 Finder 中显示	55.78 MB	jdk-8u191-linux-i586-demos.tar.gz
取消	56.02 MB	jdk-8u191-linux-x64-demos.rpm

显示全部



就可以查看到当前正在下载的任务列表，同时这里也会有一个文件对应的链接，我们选择复制这个链接地址：

今天



这个时候会发现链接地址其实变成了这样的：

```
https://download.oracle.com/otn-pub/java/jdk/8u191-
b12/2787e4a523244c269598db4e85c51e0c/jdk-8u191-linux-x64.rpm?
AuthParam=1545193703_3bbf2633c3d9e7961bd8433738fe79ca
```

就是在原来的链接后面加了个动态的 `AuthParam`，使用这个链接去下载就可以了。下载完成后执行：`rpm -ivh jdk-8u191-linux-x64.rpm` 即可进行安装，默认会安装在 `/usr/java/jdk1.8.0_191-amd64` 这个目录下，所以我们还需要添加环境变量，以便使用：

```
unset i
unset -f pathmunge

# jdk
export JAVA_HOME=/usr/java/jdk1.8.0_191-amd64
export CLASS_PATH=.:${JAVA_HOME}/jre/lib/rt.jar:${JAVA_HOME}/lib/dt.jar:${JAVA_HOME}/lib/tools.jar

export PATH=${JAVA_HOME}/bin:$PATH
```

添加完成之后执行一下：`source /etc/profile` 使环境变量生效

最后，执行：`java -version`，看到如下结果则说明JDK环境安装成功：

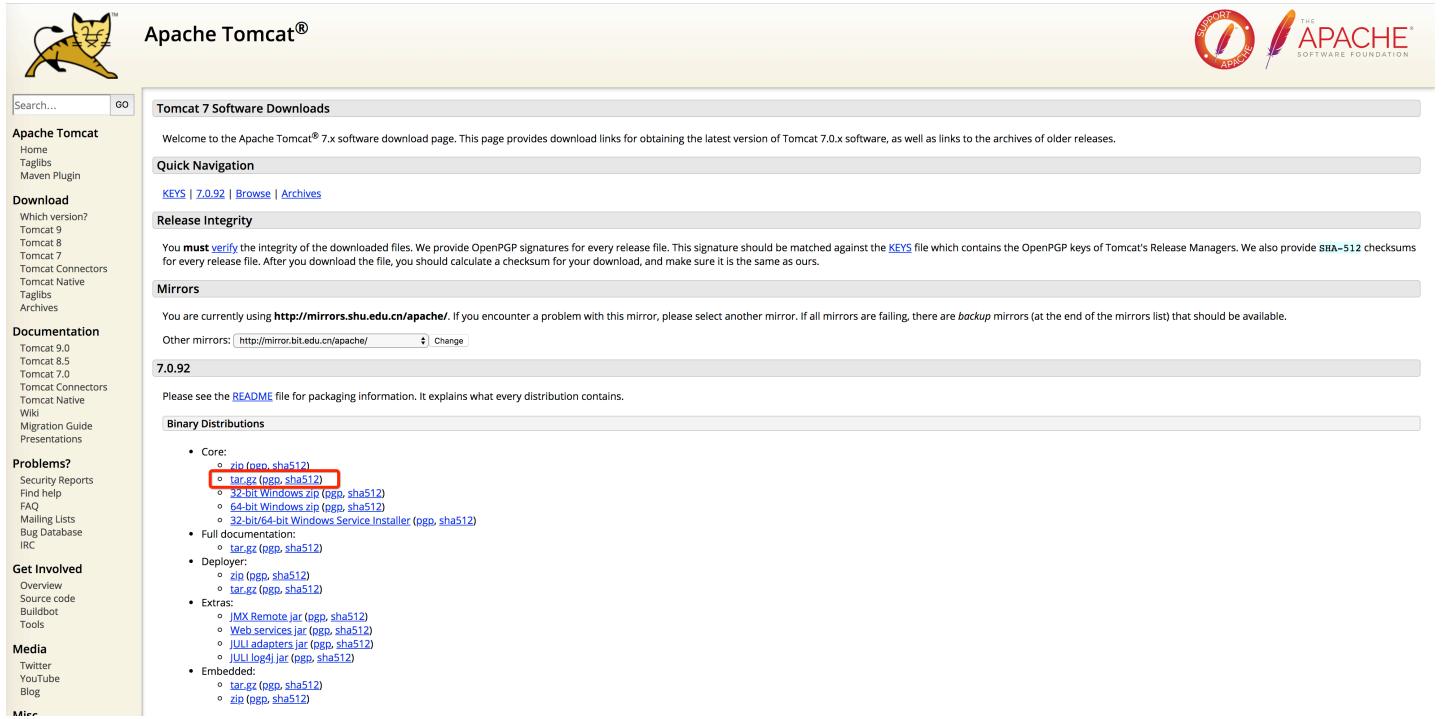
```
[root@localhost src]# java -version
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
[root@localhost src]#
```

2、Tomcat下载安装

[Apache Tomcat](#) 是Apache软件基金会一款开源的 Java Servlet, JavaServer Pages, Java

Expression Language 和 Java WebSocket 的技术实现。它同时支持HTTP协议，所以后端使用Java开发的网页一般都是选择使用Tomcat来部署应用。

Tomcat的启动依赖于JDK，所以在安装Tomcat之前得先完成我们上面的JDK环境安装。安装Tomcat也比较简单，直接到官网下载对应的安装包，然后解压缩到相应目录下即可。



The screenshot shows the Apache Tomcat 7 Software Downloads page. At the top, there's a search bar and a logo. The main content area is titled "Tomcat 7 Software Downloads". It includes sections for "Quick Navigation", "Release Integrity" (with a note about OpenPGP signatures), and "Mirrors" (listing "http://mirrors.shu.edu.cn/apache/"). Below these, a "7.0.92" section provides a link to the "README" file. The "Binary Distributions" section lists several categories: Core, Full documentation, Deployer, Extras, and Embedded. Under "Core", there are links for "zip (pgp, sha512)" and "tar.gz (pgp, sha512)". The "tar.gz (pgp, sha512)" link is highlighted with a red box.

然后需要说明的是我们需要修改一下Tomcat的配置，使其默认支持UTF-8字符集编码，直接修改解压缩之后的conf目录下的server.xml，添加如下配置：

```
root@localhost:/usr/local/apache-tomcat-7.0.92/conf
    Documentation at /docs/config/service.html
-->
<Service name="Catalina">

    <!--The connectors can use a shared executor, you can define one or more named thread
pools-->
    <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
    -->

    <!-- A "Connector" represents an endpoint by which requests are received
        and responses are returned. Documentation at :
            Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
            Java AJP Connector: /docs/config/ajp.html
            APR (HTTP/AJP) Connector: /docs/apr.html
            Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort="8443" URIEncoding="UTF-8" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
        port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000"
        redirectPort="8443" />
```

之后我们就可以启动Tomcat了，执行bin目录下的 `startup.sh`，看到如下结果即为启动成功：

```
[root@localhost bin]# ls
bootstrap.jar          configtest.bat    setclasspath.sh  tomcat-native.tar.gz
catalina.bat           configtest.sh     shutdown.bat    tool-wrapper.bat
catalina.sh            daemon.sh        shutdown.sh    tool-wrapper.sh
catalina-tasks.xml    digest.bat      startup.bat    version.bat
commons-daemon.jar    digest.sh       startup.sh    version.sh
commons-daemon-native.tar.gz  setclasspath.bat  tomcat-juli.jar
[root@localhost bin]# ./startup.sh
Using CATALINA_BASE:   /usr/local/apache-tomcat-7.0.92
Using CATALINA_HOME:   /usr/local/apache-tomcat-7.0.92
Using CATALINA_TMPDIR: /usr/local/apache-tomcat-7.0.92/temp
Using JRE_HOME:        /usr/java/jdk1.8.0_191-amd64
Using CLASSPATH:       /usr/local/apache-tomcat-7.0.92/bin/bootstrap.jar:/usr/local/apach
e-tomcat-7.0.92/bin/tomcat-juli.jar
Tomcat started.
```

此时在浏览器中访问 `http://192.168.231.128:8080` 即可看到如下界面：

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Developer Quick Start

Documentation

Getting Help

Copyright ©1999-2018 Apache Software Foundation. All Rights Reserved

三、数据库 MySQL

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB公司开发，目前属于Oracle公司旗下产品。MySQL使用的SQL语言是用于访问数据库的最常用标准化语言，由于其体积小、速度快、成本低并且开源等特点，一般中小型网站的开发都选择MySQL作为网站数据库。

The world's most popular open source database

Contact MySQL | Login | Register

MySQL.COM DOWNLOADS DOCUMENTATION DEVELOPER ZONE

Products Cloud Services Partners Customers Why MySQL? News & Events How to Buy

New! MySQL Enterprise Masking and De-identification

Meet Regulatory Requirements
Protect Confidential Information
Reduce the Risk of a Data Breach

LEARN MORE



MySQL Enterprise Edition

The most comprehensive set of advanced features, management tools and technical support to achieve the highest levels of MySQL scalability, security, reliability, and uptime.
[Learn More »](#)



Oracle MySQL Cloud Service

Built on MySQL Enterprise Edition and powered by the Oracle Cloud, Oracle MySQL Cloud Service provides a simple, automated, integrated and enterprise ready MySQL cloud service, enabling organizations to increase business agility and reduce costs.
[Learn More »](#)



MySQL Cluster CGE

MySQL Cluster enables users to meet the database challenges of next generation web, cloud, and communications services with uncompromising scalability, uptime and agility.
[Learn More »](#)



MySQL for OEM/ISV

Over 2000 ISVs, OEMs, and VARs rely on MySQL as their products' embedded database to make their applications, hardware and appliances more competitive, bring them to market faster, and lower their cost of goods sold.
[Learn More »](#)

1、MySQL安装与启动

执行：`yum install -y mysql-server` 即可完成安装：

```
[root@localhost ~]# yum install -y mysql-server
已加载插件: fastestmirror, refresh-packagekit, security
设置安装进程
Loading mirror speeds from cached hostfile
base                                         | 3.7 kB     00:00
extras                                        | 3.4 kB     00:00
updates                                       | 3.4 kB     00:00
解决依赖关系
--> 执行事务检查
--> Package mysql-server.x86_64 0:5.1.73-8.el6_8 will be 安装
--> 处理依赖关系 mysql = 5.1.73-8.el6_8, 它被软件包 mysql-server-5.1.73-8.el6_8.x86_64 需要
--> 处理依赖关系 perl-DBI, 它被软件包 mysql-server-5.1.73-8.el6_8.x86_64 需要
--> 处理依赖关系 perl-DBD-MySQL, 它被软件包 mysql-server-5.1.73-8.el6_8.x86_64 需要
--> 处理依赖关系 perl(DBI), 它被软件包 mysql-server-5.1.73-8.el6_8.x86_64 需要
--> 执行事务检查
--> Package mysql.x86_64 0:5.1.73-8.el6_8 will be 安装
--> Package perl-DBD-MySQL.x86_64 0:4.013-3.el6 will be 安装
--> Package perl-DBI.x86_64 0:1.609-4.el6 will be 安装
--> 完成依赖关系计算

依赖关系解决
```

```
总计                                         6.8 MB/s | 10 MB   00:01
运行 rpm_check_debug
执行事务测试
事务测试成功
执行事务
正在安装 : perl-DBI-1.609-4.el6.x86_64          1/4
正在安装 : perl-DBD-MySQL-4.013-3.el6.x86_64    2/4
正在安装 : mysql-5.1.73-8.el6_8.x86_64          3/4
正在安装 : mysql-server-5.1.73-8.el6_8.x86_64   4/4
Verifying : perl-DBD-MySQL-4.013-3.el6.x86_64    1/4
Verifying : mysql-5.1.73-8.el6_8.x86_64          2/4
Verifying : mysql-5.1.73-8.el6_8.x86_64          3/4
Verifying : perl-DBI-1.609-4.el6.x86_64          4/4

已安装 :
mysql-server.x86_64 0:5.1.73-8.el6_8

作为依赖被安装:
mysql.x86_64 0:5.1.73-8.el6_8                  perl-DBD-MySQL.x86_64 0:4.013-3.el6           perl-DBI.x86_64 0:1.609-4.el6

完毕!
```

安装完成后建议将其设置为开机自启动：`chkconfig mysqld on`，使用`chkconfig --list mysqld`命令查看到2345位为ON即可。

```
[root@localhost ~]# chkconfig mysqld on
[root@localhost ~]# chkconfig --list mysqld
mysqld           0:关闭  1:关闭  2:启用  3:启用  4:启用  5:启用  6:关闭
[root@localhost ~]#
```

为了解决数据库中文乱码的问题，我们修改数据库配置文件：`/etc/my.cnf`，插入`default-character-set=utf8`，如下图：

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
default-character-set=utf8
[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
~
```

最后，启动MySQL服务： `service mysqld start` :

```
[root@localhost ~]# service mysqld start
Initializing MySQL database:  Installing MySQL system tables...
OK
Filling help tables...
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:

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

Alternatively you can run:
/usr/bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default. This is
strongly recommended for production servers.

See the manual for more instructions.

You can start the MySQL daemon with:
cd /usr ; /usr/bin/mysqld_safe &

You can test the MySQL daemon with mysql-test-run.pl
cd /usr/mysql-test ; perl mysql-test-run.pl

Please report any problems with the /usr/bin/mysqlbug script!

Starting mysqld: [ OK ]
```

2、MySQL 密码、用户、权限

上面我们启动MySQL服务后直接使用 `mysql -u root` 即可登MySQL数据库，不需要任何密码，显然这是不合理的。

所以第一步，我们就是要清理一下用户并且为 root 用户设置密码：

```
# 清理匿名用户
delete from mysql.user where User="";
# 设置 root 用户密码
set password for root@localhost=password("root");
set password for root@127.0.0.1=password("root");
# 刷新权限
flush privileges;
```

第二步，我们要插入一个新的用户，与Linux系统一样，我们尽量不要总是以 root 用户：

```
# 插入MySQL用户
insert into mysql.user(Host,User,Password) values("localhost", "sands", password("sands"));
insert into mysql.user(Host,User,Password) values("%", "sands", password("sands"));
```

第三步，我们创建一个数据库，并将该数据库所有权限赋予上面插入的新用户：

```
# 创建 database
create database `sands_db` default character set utf8 collate utf8_general_ci;
# 刷新权限
flush privileges;
# 将创建的 database 下所有表的权限赋予上述用户
grant all privileges on sands_db.* to sands@'localhost' identified by 'sands' with grant option;
grant all privileges on sands_db.* to sands@'%' identified by 'sands' with grant option;
# 刷新权限
flush privileges;
# 创建数据表
CREATE TABLE IF NOT EXISTS `user_tbl`(
  `id` INT UNSIGNED AUTO_INCREMENT,
  `name` VARCHAR(50) NOT NULL,
  `gender` TINYINT NOT NULL,
  `age` TINYINT NOT NULL,
```

```
`birthday` DATE,  
 `create_at` DATE,  
 `update_at` DATE,  
 PRIMARY KEY (`id`)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

删除数据库：`drop database db_name`

四、反向代理与负载均衡服务器（NGINX）

[NGINX](#) 是一款开源的内存占用小，稳定性高，性能强大的HTTP和反向代理服务器，邮件代理服务器以及通用的TCP/UDP代理服务器。

1、NGINX 下载安装

直接从NGINX官网即可 [下载](#) 到稳定版本（Stable version）的源码包，然后按照我们之前的软件安装章节中说到的源码包安装步骤即可完成安装：

```
# 下载源码包  
wget https://nginx.org/download/nginx-1.14.2.tar.gz  
# 解压缩  
tar -zxvf nginx-1.14.2.tar.gz  
# 进入解压后的目录  
cd nginx-1.14.2  
# 执行configure  
.configure --prefix=/usr/local/nginx  
# 编译  
make  
# 安装  
make install
```

注意：NGINX需要依赖gcc、pcre-devel、zlib-devel，请先使用yum将依赖安装好。

完成之后nginx就安装在了 `/usr/local/nginx` 目录下。

当然了，你也可以选择使用 `yum` 来快速安装，但是遗憾的是各种yum源默认都是没有nginx的，所以你需要自己添加nginx的yum源：

```
# 创建、编辑nginx.repo  
vim /etc/yum.repos.d/nginx.repo
```

```
# 写入如下内容
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/OS/OSRELEASE/$basearch/
gpgcheck=0
enabled=1
```

根据具体使用的系统情况，替换其中的“OS”为“rhel”或者“centos”，并且对于6.x或者7.x版本将“OSRELEASE”替换为“6”或者“7”。

添加完成yum源之后就可以直接使用yum安装了，这个比较简单，就不过多赘述了。

2、NGINX的启动与常用命令

NGINX的启动非常简单，执行安装路径下的 `sbin/nginx` 可执行文件即可启动：

```
[root@localhost nginx]# ./sbin/nginx
[root@localhost nginx]# netstat -nltp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp        0      0 0.0.0.0:3306           0.0.0.0:*            LISTEN     16639/mysqld
tcp        0      0 0.0.0.0:80            0.0.0.0:*            LISTEN     42101/nginx
tcp        0      0 0.0.0.0:21           0.0.0.0:*            LISTEN     6993/vsftpd
tcp        0      0 0.0.0.0:22           0.0.0.0:*            LISTEN     2286/sshd
tcp        0      0 127.0.0.1:631          0.0.0.0:*            LISTEN     2080/cupsd
tcp        0      0 127.0.0.1:25          0.0.0.0:*            LISTEN     2461/master
tcp        0      0 ::ffff:127.0.0.1:8005   ::*:                 LISTEN     16041/java
tcp        0      0 :::8008              ::*:                 LISTEN     3088/httpd
tcp        0      0 :::8009              ::*:                 LISTEN     16041/java
tcp        0      0 :::8080              ::*:                 LISTEN     16041/java
tcp        0      0 :::22               ::*:                 LISTEN     2286/sshd
tcp        0      0 :::1:631             ::*:                 LISTEN     2080/cupsd
tcp        0      0 :::1:25              ::*:                 LISTEN     2461/master
[root@localhost nginx]#
```

启动后打开浏览器访问：<http://192.168.231.128:80> 看到如下页面表示nginx服务正常：



NGINX其他常用命令：

```
# 测试配置文件是否合法  
.sbin/nginx -t  
# 重启NGINX  
.sbin/nginx -s reload  
# 停止NGINX  
.sbin/nginx -s stop/quit
```

补充说明：

查看进程命令：`ps -ef | grep nginx`

平滑重启：`kill -HUP [NGINX主进程号(即PID)]`

3、NGINX 反向代理

说到 NGINX 反向代理可能大家心里都有一个疑问：“代理就代理呗，怎么还反向的呢？难道还有正向的啊？”对，你还真别说！这里给大家科普一个关于代理的小知识点。

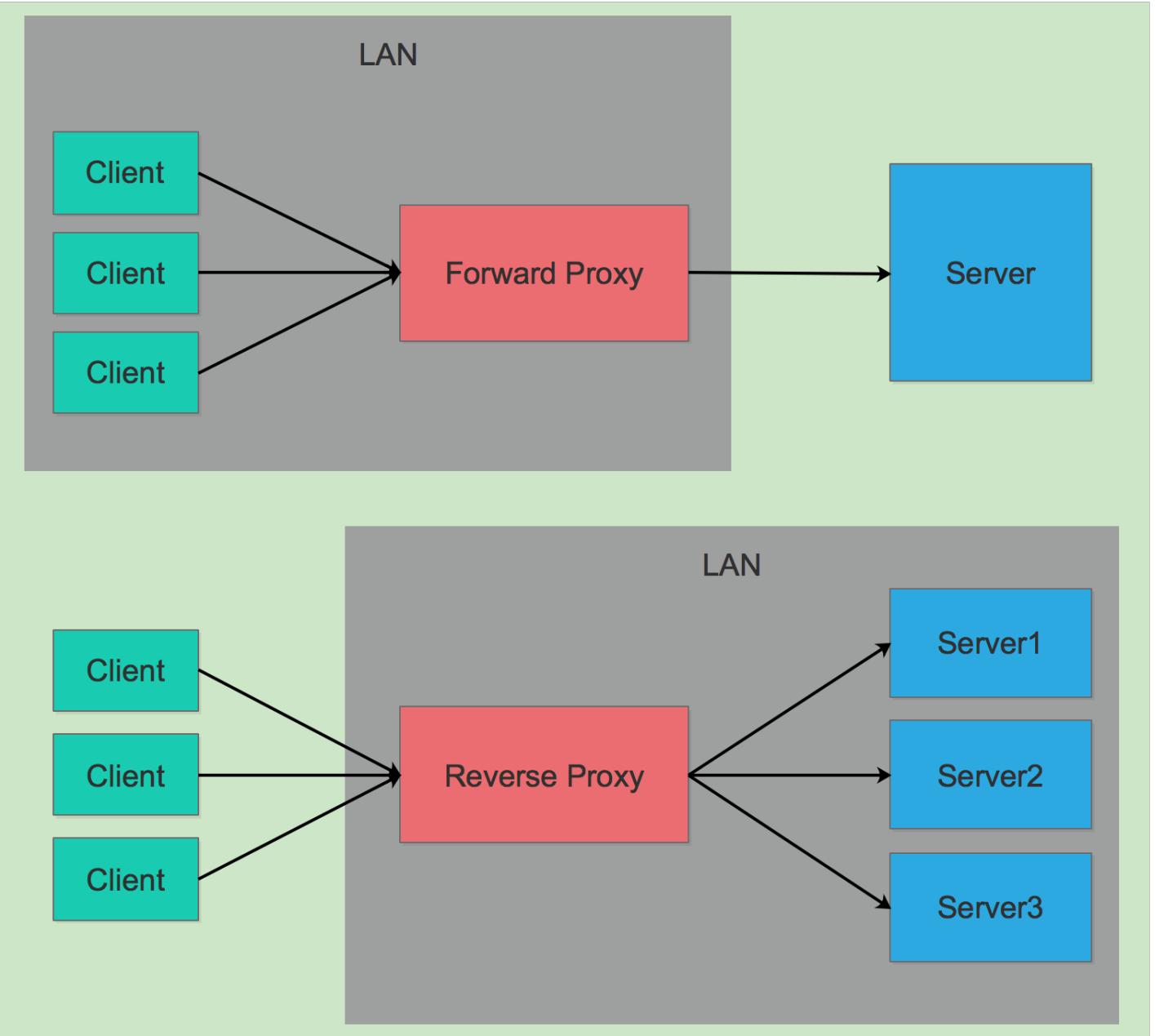
代理分为 **正向代理** 和 **反向代理**，我们通常所说的代理（Charles, VPN）其实都是指正向代理。

- 正向代理是一个位于客户端和目标服务器之间的代理服务器。由于某种原因，客户端无法直接请求目标服务器，那么为了从目标服务器上获取到内容，客户端要请求代理服务器，并且指定目标服务器，之后代理服务器向目标服务器转交请求并将响应内容返回给客户端。正向代理时客户端需要进行一些设置才能使用。
- 反向代理正好相反。对于客户端而言，反向代理服务器就好像目标服务器，并且客户端不需要进行任何设置。客户端请求反向代理服务器，接着反向代理服务器分发请求并将响应转交给客户端，这个过程看起来就好像反向代理服务器自己完成的一样，因此客户端并不会感知到反向代理背后的服务，也因此客户端不需要做任何设置，只需要把反向代理服务器当成 **真正的 目标服务器** 就好了。

正向代理和反向代理之间的区别也很明显：

1. 正向代理需要你主动设置代理服务器的IP或者域名进行访问，由设置的服务器IP或者域名去获取访问内容并返回；而反向代理不需要你做任何设置，直接访问服务器真实IP或者域名，但是反向代理服务器内部会自动根据请求进行跳转及返回，你不知道它最终访问的是哪些机器。
2. 正向代理是代理客户端，为客户端收发请求，使真实客户端对服务器不可见；而反向代理是代理服务器端，为服务器收发请求，使真实服务器对客户端不可见。

下面用一张图来描述两者之间的差异：



LAN: *Local Area Network*, 局域网

WAN: *Wide Area Network*, 广域网

WLAN: *Wireless LAN*, 无线局域网 (*Wi-Fi*只是实现*WLAN*的一种协议)

VLAN: *Virtual LAN*, 虚拟局域网

VPN: *Virtual Private Network*, 虚拟专用网络

在正向代理中，Proxy和Client同属一个LAN，对Server透明；而反向代理中，Proxy和Server同属一个LAN，对Client透明。正向代理主要是为了给在防火墙内的局域网提供Internet访问途径；反向代理则主要是将防火墙后面的服务器提供出来让Internet用户访问，另外还可以完成诸如负载均衡等功能。

NGINX的配置文件路径为其安装目录下的 `nginx/conf/nginx.conf` :

```
#user nobody;
worker_processes 1;

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;

#pid      logs/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include      mime.types;
    default_type application/octet-stream;

    #log_format main '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    sendfile      on;
    #tcp_nopush   on;

    #keepalive_timeout 0;
    keepalive_timeout 65;

    #gzip  on;

    server {
        listen       80;
        server_name  localhost;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;
```

```

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

#error_page 404                  /404.html;

# redirect server error pages to the static page /50x.html
#
error_page   500 502 503 504  /50x.html;
location = /50x.html {
    root    html;
}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80
#
#location ~ \.php$ {
#    proxy_pass    http://127.0.0.1;
#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
#
#location ~ \.php$ {
#    root            html;
#    fastcgi_pass    127.0.0.1:9000;
#    fastcgi_index   index.php;
#    fastcgi_param   SCRIPT_FILENAME  /scripts$fastcgi_script_name;
#    include         fastcgi_params;
#}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
#
#location ~ /\.ht {
#    deny  all;
#}
}

#####
include vhost/*.conf;
#####

```

以上是部分配置截图，这里我们主要看 `server` 节点里的配置，我们把每一个 `server` 叫一个虚拟主机，为了方便理解，我们把它看成是一个server对应一个域名的请求代理。例如图上配置的大致意思是：监听80端口，如果遇到localhost(127.0.0.1)的请求将其定位到html目录下，并且如果有index.html或者index.htm则返回index页面。这个配置文件里的server可以写很多，根据需要加就可以了，但是我们一般不这么干，这里要注意一下图上的最后，我们添加了一行：

```
#####
include vhost/*.conf;
#####
```

vhost (virtual host的简写) 是我们接下来要在nginx.conf同级目录里创建的一个文件夹，里面会放我们需要添加的虚拟主机对应的server节点配置文件，例如：我们希望访问 `www.mobpods.com` 的时候能够访问到我们的 `Tomcat` 上，那么我们就会创建一个 `www.mobpods.com.conf` 的文件（以server_name命名的.conf文件），然后写入如下内容：

```
server {
    listen 80;
    autoindex on;
    server_name www.mobpods.com;
    access_log /usr/local/nginx/logs/access.log combined;
    index index.html index.htm index.jsp index.php;

    if ( $query_string ~* ".*[\;'\<\>].*" ) {
        return 404;
    }
    location / {
        proxy_pass http://127.0.0.1:8080;
        add_header Access-Control-Allow-Origin *;
    }
}
```

监听80端口，当遇到`www.mobpods.com`的请求时将它转发到本地的8080端口（其实就是Tomcat）上

这就是NGINX反向代理配置的第一种，服务转发，也是最常用的一种。这里再给大家介绍另外一种也比较常用的反向代理配置：

```
server {
    listen 80;
    autoindex on;
    server_name image.mobpods.com;
    access_log /usr/local/nginx/logs/access.log combined;
    index index.html index.htm index.jsp index.php;

    if ( $query_string ~* ".*[\;'\<\>].*" ) {
        return 404;
    }
}
```

```
location / {  
    root /ftppfile/images;  
    add_header Access-Control-Allow-Origin *;  
}  
}
```

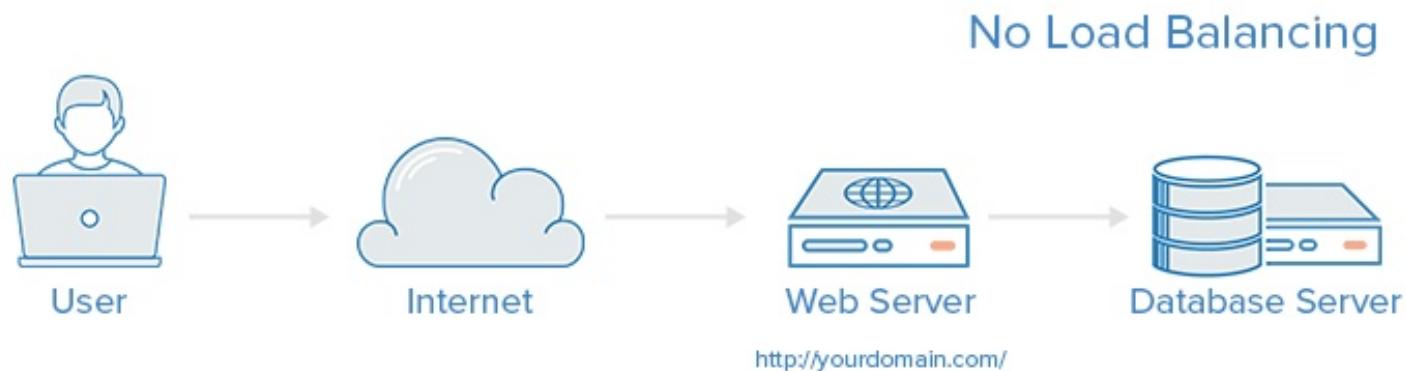
这种配置是直接转发到具体目录的，主要用于访问静态资源，比如一些商品图片我们后台管理时通过FTP上传到服务器，那么前台访问的时候就可以直接通过一个指定的域名访问过来了。

4、NGINX 负载均衡

负载平衡 (Load balancing) 是一种计算机技术，用来在多个计算机（计算机集群）、网络连接、CPU、磁碟驱动器或其他资源中分配负载，以达到最佳化资源使用、最大化吞吐率、最小化响应时间、同时避免过载的目的。

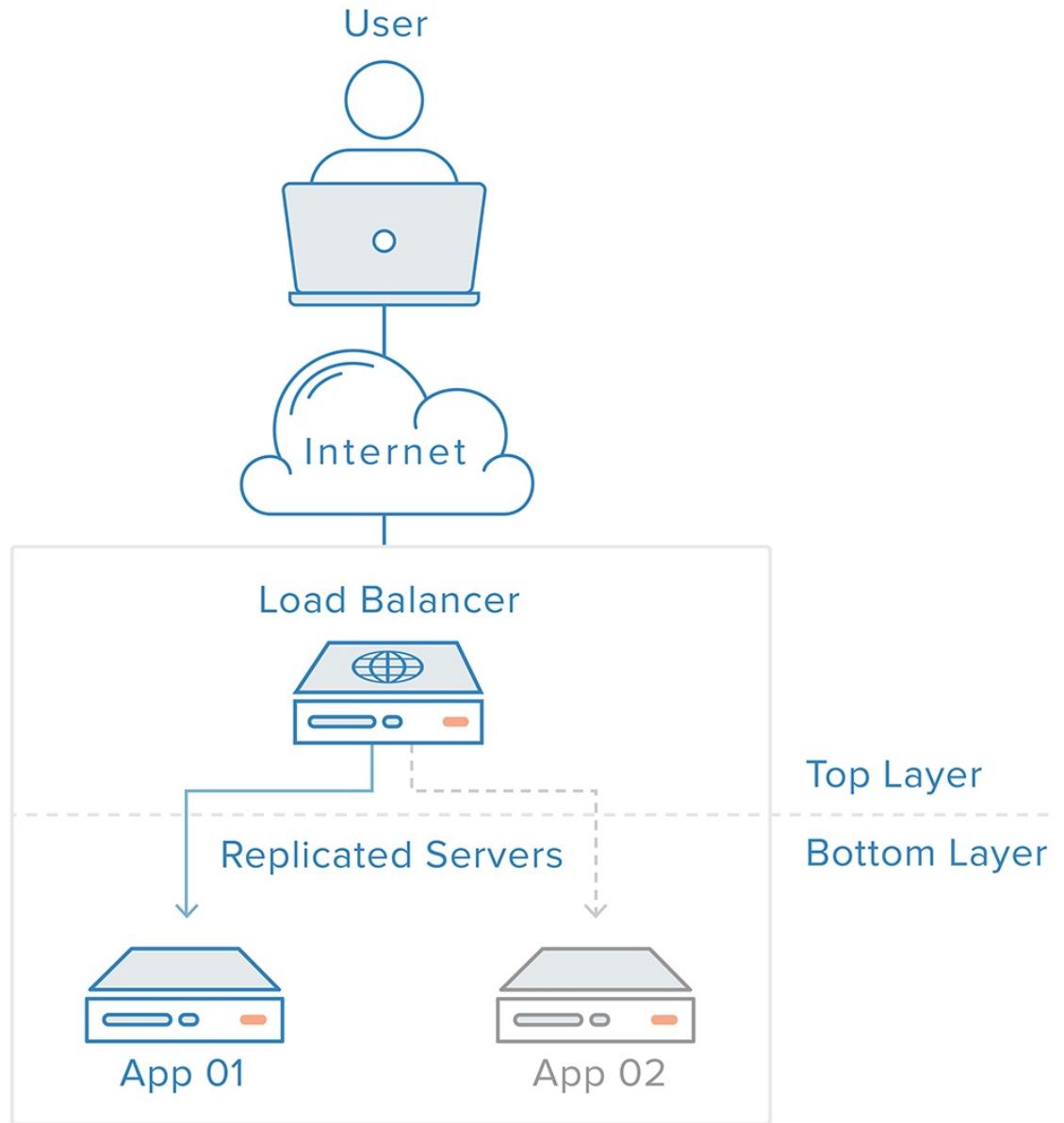
负载均衡是高可用网络基础架构的关键组件，通常用于将工作负载分布到多个服务器来提高网站、应用、数据库或其他服务的性能和可靠性。

一个没有负载均衡的Web服务架构类似下面这样：



用户直连到Web服务器，如果这个服务器宕机了，那么用户自然也就没办法访问了。另外，如果同时有很多用户试图访问服务器，超过了其能处理的极限，就会出现加载速度缓慢或根本无法连接的情况。

而通过在后端引入一个负载均衡服务器和至少一个额外的Web服务器，可以缓解这个故障。通常情况下，所有的后端Web服务器会保证提供相同的内容，以便用户无论哪个服务器响应，都能收到一致的内容。



从图里可以看到，用户访问负载均衡服务器，再由负载均衡服务器将请求转发给后端服务器。在这种情况下，单点故障现在转移到负载均衡服务器上了。我们暂且先不讨论如何解决这个问题，我们先来看一下常用的几种负载均衡策略：

- **Round Robin (轮询)**：为第一个请求选择列表中的第一个服务器，然后按顺序向下移动列表直到结尾，然后循环。
- **Least Connections (最小连接)**：优先选择连接数最少的服务器，在普遍会话较长的情况下推荐使用。
- **Source (来源)**：根据请求源的IP的散列 (hash) 来选择要转发的服务器。这种方式可以一定程度上保证特定用户能连接到相同的服务器。

以服务转发为例的 NGINX 负载均衡配置如下：

在http节点下配置好负载均衡池和相应策略，下图没有指定策略则默认为轮询策略：

```
#user    nobody;
worker_processes  1;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid      logs/nginx.pid;

events {
    worker_connections  1024;
}

http {
    upstream tomcats {
        server 127.0.0.1:8080;
        server 127.0.0.1:8081;
        server 127.0.0.1:8082;
    }

    include      mime.types;
    default_type application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                      '$status $body_bytes_sent "'.$http_referer" '
    #                      '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    sendfile      on;
    #tcp_nopush    on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    server {
        listen      80;
        server_name localhost;
    }
}
```

负载均衡池名

可以配置多个负载
均衡池

然后在server节点中使用负载均衡池：

```
server {
    listen 80;
    server_name www.sands.com;
    access_log /usr/local/nginx/logs/access.log combined;
    index index.html index.htm index.jsp index.php;

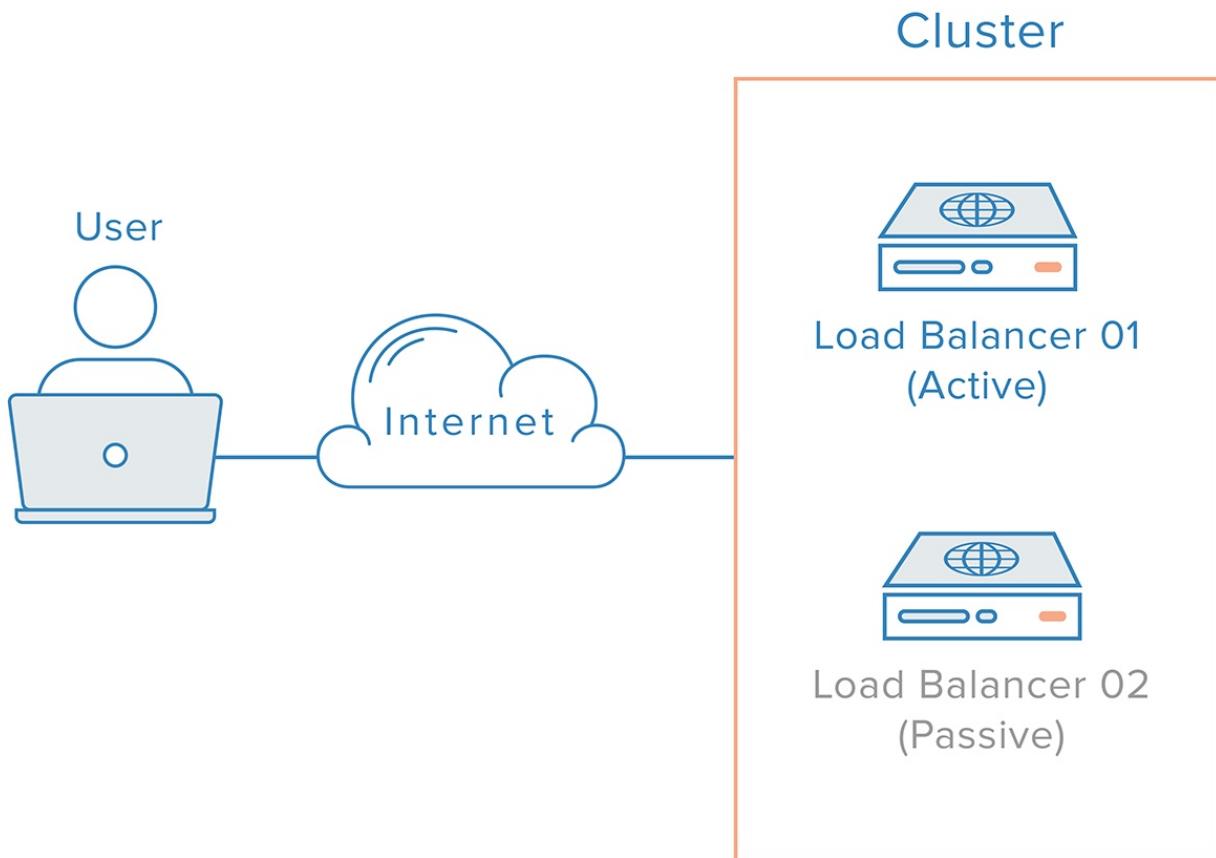
    if ( $query_string ~* ".*[\\;'\<\>].*" ) {
        return 404;
    }

    location / {
        proxy_pass http://tomcats;
        add_header Access_Control_Allow_Origin *;
    }
}
```

以上配置其实是将 `www.sands.com` 的请求转发到了本机起的三个 `Tomcat` 上，多次刷新就会看到轮询访问到三个不同版本的Tomcat。

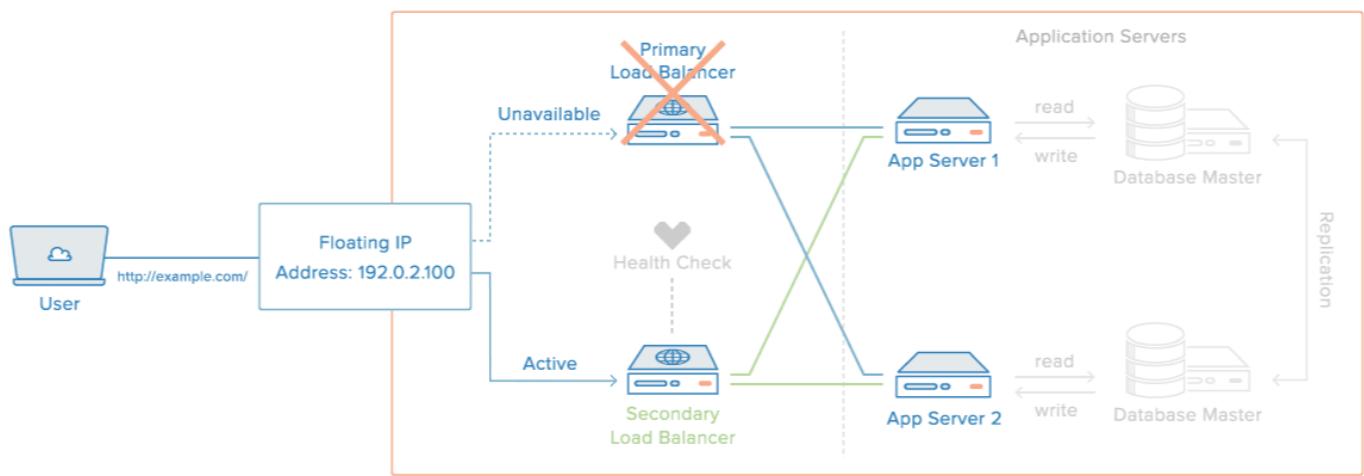
到这里为止，NGINX的负载均衡也就说完了，至于最小连接数策略和IP–hash策略就留给各位自行实践了。

别忘了上面我们有一个问题还没有解决，就是负责均衡服务器单点故障的问题，要想解决这个问题，我们可以将第二个负载均衡服务器连接到第一个上，从而形成一个负载均衡服务器集群。



当主负载均衡服务器发生了故障，就将用户请求转到第二个负载均衡服务器上，但是这里有另外一个问题，DNS更改通常需要较长的时间才能生效，因此我们需要一个能灵活解决IP地址重映射的方法，比如：浮动IP（Floating IP），这样域名可以保持解析到相同的IP，而IP自身能在服务器之间移动。

一个使用浮动IP的负载均衡架构示意图：



- ① Active/Passive Cluster is healthy
- ② Primary node fails
- ③ Floating IP is assigned to Secondary node

通过浮动IP加负载均衡服务器集群就可以解决绝大部分单点故障的问题了。

五、Java 用户系统项目实战

未完待续.....