轻量级 WEB 服务器 Nginx

By：Ivy cui
Time：2015.12.09
Mail:cuihuahua@uplooking.com

第 1 章　Nginx 简介

第 1.1 节　WEB 服务器简介

WEB 服务器也称为 WWW(WORLD WIDE WEB)服务器，主要功能是提供网上信息浏览服务。在 Internet 网络环境中，Web 服务无疑是最为流行的应用系统之一，有了 web 站点，于企业而言，可以充分展示自己的产品，宣传企业形象，提供与客户交流、电子商务交易的平台等丰富的网络资源，已经成为很多人在网上查找、浏览信息的主要手段。于用户而言，用户可以通过简单的图形界面就可以访问各个大学、组织、公司等的最新信息和各种服务。如果你有条件，你可以注册一个域名，申请一个 IP 地址，然后让你的 ISP 将这个 IP 地址解析到你的 LINUX 主机上。然后，在 LINUX 主机上架设一个 WEB 服务器。你就可以将主页存放在这个自己的 WEB 服务器上，通过它把自己的主页向外发布。

第 1.2 节　几个常用 WEB 服务器简介

Unix 和 Linux 平台下的常用 Web 服务器有 Apache、 Nginx、 Lighttpd、 Tomcat、 IBM WebSphere 等。目前应用最广泛的 Web 服务器是 Apache。 Windows 平台下最常用的服务器则是微软公司的 IIS(Internet Information Server)。

第 1.2.1 节　Apache 服务器

Apache 起初由 Illinois 大学 Urbana-Champaign 的国家高级计算程序中心开发。此后 Apache 被开放源代码团体的成员不断的发展和加强。1996 年 4 月以来，Apache 一直是 Internet 上最流行的 HTTP 服务器，1999 年 5 月它在 57% 的网页服务器上运行，到了 2005 年 7 月这个比例上升到 69%。Apache 是目前世界上用的最多的 Web 服务器。

优势（1）开放源代码

（2）支持跨平台的应用(可以运行在几乎所有的 Unix、Linux、Windows 系统平台之上)

（3）支持各种网络编程语言，如 php，python，甚至微软的 ASP 技术也能在 apache 服务器中使用。

（4）模块化设计，使之具有良好的扩展性

（5）运行稳定，安全性良好

Apache 的官方网站: http://www.apache.org

第 1.2.2 节　Lighttpd 服务器

Lighttpd 是一个德国人领导的开源软件，其根本的目的是提供一个专门针对高性能网站。Lighttpd 是众多 OpenSource 轻量级的 Web server 中较为优秀的一个。支持 FastCGI, CGI, Auth, 输出压缩(output compress), URL 重写, Alias 等重要功能，而 Apache 之所以流行，很大程度也是因为功能丰富，在 Lighttpd 上很多功能都有相应的实现了，这点对于 Apache 的用户是非常重要的，因为迁移到 Lighttpd 就必须面对这些问题。

优势：（1）内存消耗低

（2）安全性高

（3）兼容性好

（4）运行速度快

Lighttpd 的官方网站: http://www.lighttpd.net

第 1.2.3 节 Tomcat 服务器

Tomcat 是一个免费的开源的 Serlvet 容器，它是 Apache 基金会的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其它一些公司及个人共同开发而成。由于有了 Sun 的参与和支持，最新的 Servlet 和 Jsp 规范总能在 Tomcat 中得到体现。Tomcat 即是一个 Jsp 和 Servlet 的运行平台。同时 Tomcat 又不仅仅是一个 Servlet 容器，它也具有传统的 Web 服务器的功能：处理 Html 页面，但是与 Apache 相比，它的处理静态 Html 的能力就不如 Apache，我们可以将 Tomcat 和 Apache 集成到一块，让 Apache 处理静态 Html，而 Tomcat 处理 Jsp 和 Servlet。这种集成只需要修改一下 Apache 和 Tomcat 的配置文件即可。

Tomcat 的官方网站: http://tomcat.apache.org

第 1.2.4 节 IBM WebSphere

WebSphere Application Server 是一种功能完善、开放的 Web 应用程序服务器，是 IBM 电子商务计划的核心部分。基于 Java 和 Servlets 的 Web 应用程序运行环境，包含了为 Web 站点提供服务所需的一切，运行时可以协同并扩展 Apache、Netscape、 IIS 和 IBM 的 HTTPWeb 服务器，因此可以成为强大的 Web 应用服务器。

WebSphere 的官方网站: http://www.ibm.com

第 1.2.5 节 Windows IIS

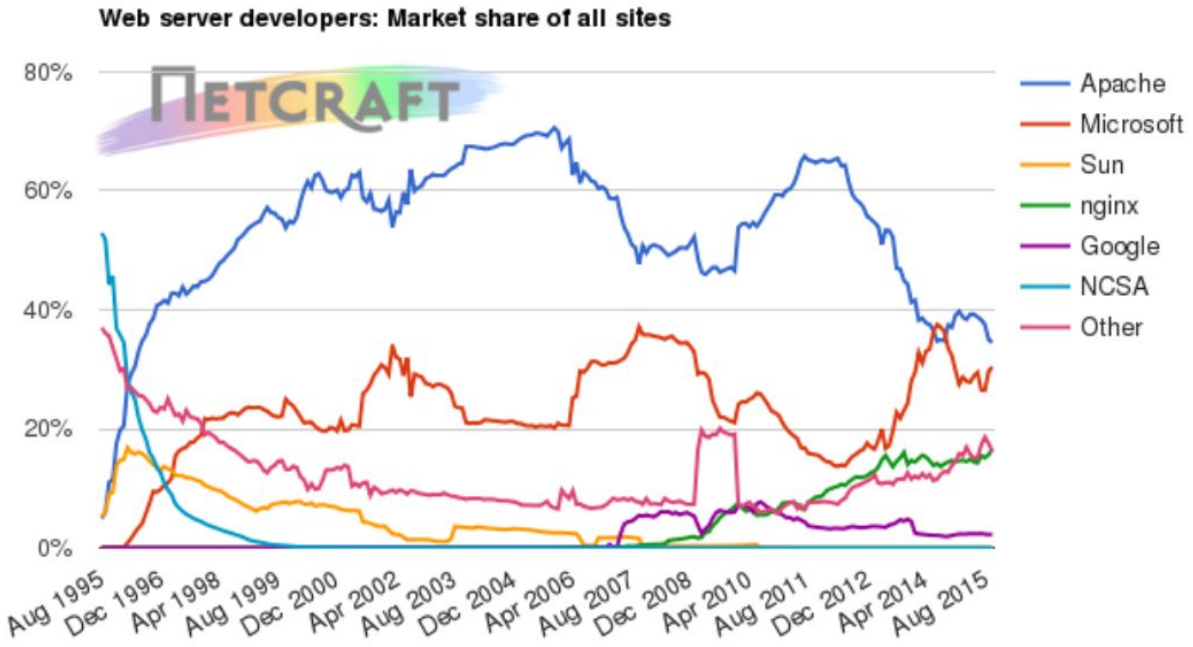
IIS 是 Internet Information Server 的缩写，它是微软公司主推的服务器。IIS 与 Window NT Server 完全集成在一起，因而用户能够利用 Windows NT Server 和 NTFS（NT File System，NT 的文件系统）内置的安全特性，建立强大，灵活而安全的 Internet 和 Intranet 站点。IIS 支持 HTTP (Hypertext Transfer Protocol ,超文本传输协议）,FTP(File Transfer Protocol，文件传输协议）以及 SMTP 协议，通过使用 CGI 和 ISAPI，IIS 可以得到高度的扩展。

IIS 的官方网站: http://www.iis.net

第 1.2.6 节 Nginx 的历史

Nginx 是俄罗斯人编写的十分轻量级的 HTTP 服务器,Nginx，它的发音为“engine X”， 是一个高性能的 HTTP 和反向代理服务器，同时也是一个 IMAP/POP3/SMTP 代理服务器。Nginx 是由俄罗斯人 Igor Sysoev 为俄罗斯访问量第二的 Rambler.ru 站点开发的，它已经在该站点运行超过两年半了。Igor Sysoev 在建立的项目时，使用基于 BSD 许可。自 Nginx 发布四年来，Nginx 已经因为它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。

在俄罗斯许多大网站都已经使用它，且一直表现不凡。截至 2007 年 4 月，俄罗斯大约有 20%左右的虚拟主机是由 nignx 服务或代理的。Google 在线安全博客中统计 Nginx 服务或代理了大约所有 Internet 虚拟主机的 4%。而 Netcraft 的统计显示，Nginx 服务的主机在过去的一年里以四倍的速度增长并且在这几年里，它的排名还在不断上升，下图为 Netcraft 截止至 2015 年 8 月的统计。



第 1.3 节　为什么要用 Nginx

（1）Nginx 性能强大。专为性能优化而开发，性能是其最重要的考量,实现上非常注重效率 。它支持内核 Poll 模型，能经受高负载的考验,有报告表明能支持高达 50,000 个并发连接数。

（2）Nginx 稳定性高。其它 HTTP 服务器当遇到访问的峰值，或者有人恶意发起慢速连接时，也很可能会导致服务器物理内存耗尽频繁交换，失去响应只能重启服务器。例如当前 Apache 一旦上到 200 个以上进程，web 响应速度就明显非常缓慢了。而 Nginx 采取了分阶段资源分配技术，使得它的 CPU 与内存占用率非常低。Nginx 官方表示保持 10,000 个没有活动的连接，它只占 2.5M 内存，所以类似 DOS 这样的攻击对 Nginx 来说基本上是毫无用处的。就稳定性而言，nginx 比 lighttpd 更胜一筹。

（3）Nginx 支持热部署。启动特别容易，并且几乎可以做到 7*24 不间断运行，即使运行数个月也不需要重新启动。你还能够在不间断服务的情况下，对软件版本进行进行升级。

（4）Nginx 采用 master-slave 模型。能够充分利用 SMP 的优势，且能够减少工作进程在磁盘 I/O 的阻塞延迟。

（5）Nginx 代码质量高。代码很规范，手法成熟， 模块扩展也很容易。

（6）Nginx 采用了一些 os 提供的最新特性如对 sendfile (Linux2.2+)，accept-filter(FreeBSD4.1+)，TCP_DEFER_ACCEPT (Linux 2.4+)的支持，从而大大提高了性能。

（7）nginx 处理请求是异步非阻塞的，而 apache 则是阻塞型的，在高并发下 nginx 能保持低资源低消耗高性能。

当然，nginx 还很年轻，多多少少存在一些问题，Nginx 的作者和社区都在一直努力解决，我们有理由相信 nginx 将继续以高速的增长率来分享轻量级 HTTP 服务器市场，会有一个更美好的未来。

第 1.4 节　获取 Nginx

你可以从 Nginx 的官方主页上下载到 Nginx 的各个版本 http://nginx.net

其中最新版本为 Nginx 的开发版本，之前的版本为当前稳定版本。例如最新版本为 1.9.X，则当前稳定版本为 1.8.X。你还可以在官方 http://nginx.org/en/docs 找到一些帮助文档，另外 http://wiki.nginx.org 也是一个不错的选择。

目前 redhat 官网提供的 Nginx 的版本为 1.6 版本，但是 Nginx 的最新稳定版本为 1.8 版本。相较而言，1.8 版本具有以下优势：

- 哈希负载均衡方法

  平衡算法设计的好坏直接决定了集群在负载均衡上的表现，设计不好的算法，会导致集群的负载失衡。一般的平衡算法主要任务是决定如何选择下一个集群节点，然后将新的服务请求转发给它。常见的算法有轮转法、散列法（也就是 hash 法）、最少连接法等。其中哈希算法通过单射不可逆的 HASH 函数，按照某种规则将网络请求发往集群节点。哈希法在其他几类平衡算法不是很有效时会显示出特别的威力。例如，在 UDP 会话的情况下，由于轮转法和其他几类基于连接信息的算法，无法识别出会话的起止标记，会引起应用混乱。而采取基于数据包源地址的哈希映射可以在一定程度上解决这个问题：将具有相同源地址的数据包发给同一服务器节点，这使得基于高层会话的事务可以以适当的方式运行。相对称的是，基于目的地址的哈希调度算法可以用在 Web Cache 集群中，指向同一个目标站点的访问请求都被负载平衡器发送到同一个 Cache 服务节点上，以避免页面缺失而带来的更新 Cache 问题。

- 后端 ssl 证书验证

- 体验式的线程池支持

  体验式的线程池支持线程池作用就是限制系统中执行线程的数量。根据系统的环境情况，可以自动或手动设置线程数量，达到运行的最佳效果；少了浪费了系统资源，多了造成系统拥挤效率不高。用线程池控制线程数量，其他线程排队等候。一个任务执行完毕，再从队列的中取最前面的任务开始执行。若队列中没有等待进程，线程池的这一资源处于等待。当一个新任务需要运行时，如果线程池中有等待的工作线程，就可以开始运行了；否则进入等待队列。使之减少了创建和销毁线程的次数，每个工作线程都可以被重复利用，可执行多个任务。并且可以根据系统的承受能力，调整线程池中工作线线程的数目，防止因为消耗过多的内存，而把服务器累趴下(每个线程需要大约 1MB 内存，线程开的越多，消耗的内存也就越大，最后死机)。

第 1.5 节　课后练习

1. 简述常见市场常见 WEB 服务器及他们的区别。
2. 简述 Nginx 的优点。

第 2 章　Nginx 的安装和启动

第 2.1 节　nginx 配置前期准备工作

如果 apache 有安装并且运行，先将 apache 停掉，因为 apache 和 nginx 都监听 80 端口，避免端口冲突。

```
[root@servera ~]# systemctl stop httpd.service
[root@servera ~]# systemctl disable httpd
[root@servera ~]# systemctl status httpd
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: inactive (dead)
```

第 2.2 节　nginx 的安装

将 serverb 机器配置为 nginx 服务器

（1）ssh 登录到 serverb 机器

```
[kiosk@foundation41 Desktop]$ ssh root@172.25.41.10 -p 22011
```

（2）下载安装 nginx.

Linux 下安装一个软件有三种方式，这边我们以 rpm 安装为例。

官方下载地址：http://nginx.org/en/download.html。

也可从教师机共享目录中下载。选择相应版本进行安装，此实验中 1.8 版本为例。

```
[root@serverb ~]# mount 172.25.254.250:/content/ /mnt/
[root@serverb ~]# cd /mnt/item/
[root@serverb item]# cd nginx/other/
[root@serverb other]# rpm -ivh nginx-1.8.0-1.el7.ngx.x86_64.rpm
```

第 2.3 节　启动 nginx

```
[root@serverb other]# systemctl enable nginx.service
[root@serverb other]# systemctl start nginx.service
[root@serverb other]# ps -ef | grep nginx
root     23847     1  0 13:53 ?        00:00:00 nginx: master process /usr/sbin/nginx
-c /etc/nginx/nginx.conf
nginx    23848 23847  0 13:53 ?        00:00:00 nginx: worker process
[root@serverb other]# netstat -ltunp | grep 80
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
23847/nginx: master
```

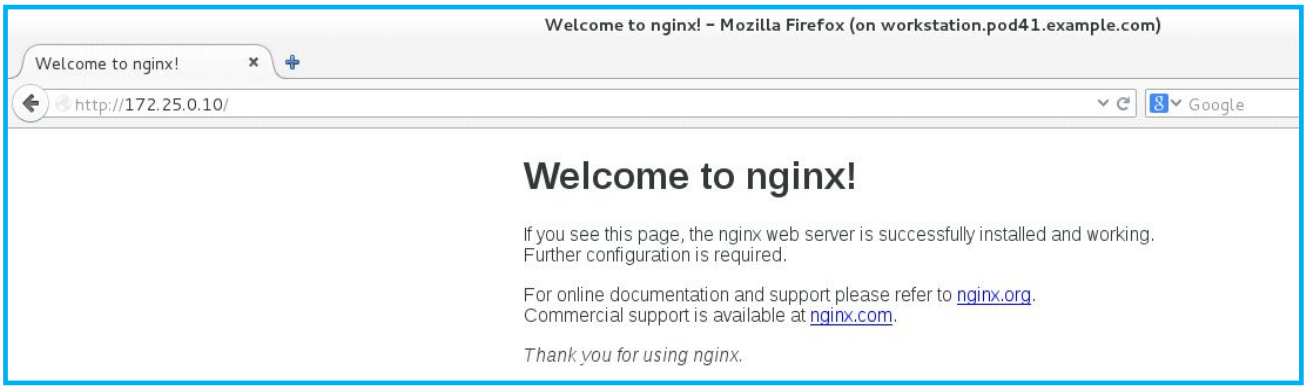
在 servera 上添加端口转发规则，访问 servera 机器外网网卡的 80 端口的请求转发给 nginx 服务内网网卡的 80 端口

```
[root@servera ~]# iptables -t nat -A PREROUTING -d 172.25.41.10 -i eth0 -p tcp --dport 80 -j DNAT
--to-destination 192.168.0.11:80
[root@servera ~]# iptables-save  > /etc/sysconfig/iptables
```

第 2.4 节　客户端测试

客户机 workstation 测试，访问到 nginx 的默认首页

```
[kiosk@foundation41 Desktop]$ ssh root@172.25.41.9 -X
[root@workstation ~]# firefox
```



第 3 章　nginx 基本配置

第 3.1 节　nginx 主配置概述

nginx 的主配置文件默认情况下位于/usr/local/nginx/conf/nginx.conf，以下为 Nginx 配置文件一些参数的注释。

```
#user  nobody;
#指定使用的用户

worker_processes  1;
＃开启的进程数，一般设置 1-5

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;
＃定义错误日志，以及记录的日志等级

#pid logs/nginx.pid;
＃定义 pid 文件位置

events {
#use [ kqueue | rtsig | epoll | /dev/poll | select | poll ];
#use epoll;  #使用 epoll（linux2.6 的高性能方式）
#Nginx 支持如下处理连接的方法（I/O 复用方法），这些方法可以通过 use 指令指定。
#select - 标准方法。 如果当前平台没有更有效的方法，它是编译时默认的方法。你可以使用配置参数 –with-select_module 和 –without-select_module 来启用或禁用这个模块。
#poll - 标准方法。 如果当前平台没有更有效的方法，它是编译时默认的方法。你可以使用配置参数 –with-poll_module 和 –without-poll_module 来启用或禁用这个模块。
#kqueue - 高效的方法，使用于 FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 和 MacOS X. 使用双处理器的 MacOS X 系统使用 kqueue 可能会造成内核崩溃。
#epoll - 高效的方法，使用于 Linux 内核 2.6 版本及以后的系统。在某些发行版本中，如 SuSE 8.2, 有让 2.4 版本的内核支持 epoll 的补丁。
#rtsig - 可执行的实时信号，使用于 Linux 内核版本 2.2.19 以后的系统。可是从 Linux 内核版本 2.6.6-mm2 开始，这个参数就不再使用了.
#/dev/poll - 高效的方法，使用于 Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ 和 Tru64 UNIX 5.1A+.
#eventport - 高效的方法，使用于 Solaris 10. 为了防止出现内核崩溃的问题， 有必要安装 这个 安全补丁。
worker_connections  1024;
＃worker_connections 51200; #每个进程最大连接数（最大连接=连接数 x 进程数）
}
http {
    include       mime.types;
#文件扩展名与文件类型映射表
    default_type  application/octet-stream;
#默认文件类型
  #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;
#开启高效文件传输模式
    #tcp_nopush    on;
#该选项用于防止网络阻塞
    #keepalive_timeout  0;
    keepalive_timeout  65;
##长链接超时时间
```

第 3.1.1 节　nginx 进程管理

进程是系统进行资源分配和调度的常见单位，在面向进程设计的程序中，进程是程序的基本执行实体，程序使用进程处理用户请求，进程是内存独享的，运行稳定，处理速度较快，但是资源占用比较大。

进程是线程的容器，也就是进程去开辟地址空间，线程去处理用户请求，在面向线程处理的程序中，线程是基本执行实体，线程是内存共享的，资源占用小，但是稳定性没有进程好。

Apache 在处理用户请求时，有 perfork 模式和 worker 模式两种，默认使用 perfork 模式，即使用进程处理用户请求，apache 程序开启后，第一个进程是由 root 用户拥有的，因为默认 1024 以下端口必须由 root 用户开启，这一个进程不处理用户请求，只接收用户请求，收到之后去调用子进程处理用户请求，这些子进程的拥有者是 apache，主进程会按照轮询算法选择一个子进程去处理用户请求。子进程数量允许动态增加，apache 中有相应的配置参数，可以定义可开启的最大进程数，进程数量和机器内存相关，假设一个静态页面需要消耗约几十 k 内存，一个动态页面需要消耗约 4M 内存，那么如果一个用户访问就会占用 4M 内存空间，1000 个用户同时访问 apache，那么需要消耗 4G 左右内存空间，如果 6000 个用户同时访问，那么需要消耗的就是 24G 左右内存空间，内存如果够用，服务继续运行，如果内存不够用，则内存溢出，导致机器宕机。

当然 apache 也可使用 worker 模式，需运行 httpd_worker 程序，worker 模式允许每个子进程有多个线程，通常来说，在一个高并发的 httpd 服务器上，worker 是个比较好的选择，因为其内存使用要低很多，但是 worker 模式也有不完善的地方，如果有一个线程崩溃，有可能会引起整个进程连同其所有线程一起死掉。

所以在运行一些动态页面（比如 php），我们更推荐使用 perfork 模式，以保证其稳定性。

Nginx 使用线程去处理用户请求，所以 nginx 在系统资源一致前提下支持更大并发。

设置 nginx 允许开启的进程数量和每个进程允许打开的线程数量

（1）修改主配置文件/etc/nginx/nginx.conf

```
user   nginx;
worker_processes   4;          #设置 nginx 允许开启的进程数量为 4

events {
     worker_connections   2048;      #设置每个进程允许打开 2048 个线程
}
```

（2）重启 nginx 服务

```
[root@serverb ~]# systemctl restart nginx.service
```

（3）查看 nginx 开启的进程信息

```
[root@serverb ~]# ps -ef | grep nginx
root       1321      1   0 15:36 ?          00:00:00 nginx: master process /usr/sbin/nginx -c /etc/nginx/nginx.conf
nginx      1322   1321   0 15:36 ?          00:00:00 nginx: worker process
nginx      1323   1321   0 15:36 ?          00:00:00 nginx: worker process
nginx      1324   1321   0 15:36 ?          00:00:00 nginx: worker process
nginx      1325   1321   0 15:36 ?          00:00:00 nginx: worker process
root       1331   1222   0 15:36 pts/0      00:00:00 grep --color=auto nginx
```

第 3.2 节　Nginx 下的虚拟主机配置

利用虚拟主机技术，可以把一台真正的主机分成许多"虚拟"的主机，每一台虚拟主机都具有独立的域名和 IP 地址，具有完整的

Internet 服务器（www, FTP,email）功能。 虚拟主机之间完全独立，在外界看来，每一台虚拟主机和一台独立的主机完全一样。效果一样但费用却大不一样了。由于多台虚拟主机共享一台真实主机的资源，每个虚拟主机用户承受的硬件费用、网络维护费用、通信线路的费用均大幅度降低，Internet 真正成为人人用得起的网络。

虚拟主机共分为三种：基于ＩＰ的虚拟主机，基于端口的虚拟主机和基于名称的虚拟主机。前两种由于受到成本和客户使用习惯的限制，相对使用的没有基于名称的虚拟主机多，下面我们简单介绍一下三种虚拟主机的配置。

第 3.2.1 节　Nginx 基于 ip 的虚拟主机配置

以下配置了两台虚拟主机，一台 IP 为 10.0.0.88，另一台为 10.0.0.87。它们都监听 80 端口。根据访问的 IP 地址不同，返回不同网站内容

```
server {
                listen 10.0.0.88:80;
                root      88.com;
                index     index.html;
        }
server {
                listen 10.0.0.87:80;
                root      87.com;
                index     index.html;
        }
[root@serverb ~]# ifconfig eth1:0 10.0.0.88
[root@serverb ~]# ifconfig eth1:1 10.0.0.87
[root@serverb ~]# mkdir /usr/local/nginx/88.com
[root@serverb ~]# echo 'this is 88.com' > /usr/local/nginx/88.com/index.html
[root@serverb ~]# mkdir /usr/local/nginx/87.com
[root@serverb ~]# echo 'this is 87.com' > /usr/local/nginx/87.com/index.html
```

第 3.2.2 节　Nginx 基于端口的虚拟主机配置

以上配置了两台虚拟主机，一台使用相同 IP。一台使用 80 端口，另一台使用 8080 端口。访问 8080 端口时需要在 URL 后加上 :8080 。

```
server {
            listen       80;
            root 80.com;
      }
server {
          listen  8080;
          root    8080.com;
      }
[root@serverb ~]# mkdir /usr/local/nginx/80.com
[root@serverb ~]# mkdir /usr/local/nginx/8080.com
[root@serverb ~]# echo 'this is 80.com' > /usr/local/nginx/80.com/index.html
[root@serverb ~]# echo 'this is 8080.com' > /usr/local/nginx/8080.com/index.html
```

第 3.2.3 节　基于名称的虚拟主机配置详解

下述例子中配置了 www.ivy.com 和 www.joy.com 两台虚拟主机

（1）进入/etc/nginx/conf.d/目录，该目录是 nginx 虚拟主机配置目录，该目录下有默认有两个文件，default.conf 文件是 nginx 的默认虚拟主机配置文件，也可作为之后自定义虚拟主机的模板，example_ssl.conf 文件是 https 配置的模板文件，后续讲 https 时会再对该文件做详细说明。

```
[root@serverb ~]# cd /etc/nginx/conf.d/
[root@serverb conf.d]# ls
default.conf    example_ssl.conf
```

（2）配置自定义虚拟主机 www.ivy.com（下述配置为最基本配置）

```
[root@serverb conf.d]# cp default.conf www.ivy.com.conf
[root@serverb conf.d]# vim www.ivy.com.conf
server {
    listen          80;
    #nginx 默认监听 80 端口
    server_name   www.ivy.com;
    #设置该虚拟主机名称
    root     /usr/share/nginx/ivy.com;
    #指定该虚拟主机的网页文件存放位置
    index    index.html index.htm;
    #指定该虚拟主机网站首页文件名称
}
```

（3）创建该虚拟主机网页文件存放目录及首页文件

```
[root@serverb conf.d]# mkdir /usr/share/nginx/ivy.com
[root@serverb conf.d]# echo ivy say hello to you >/usr/share/nginx/ivy.com/index.html
```

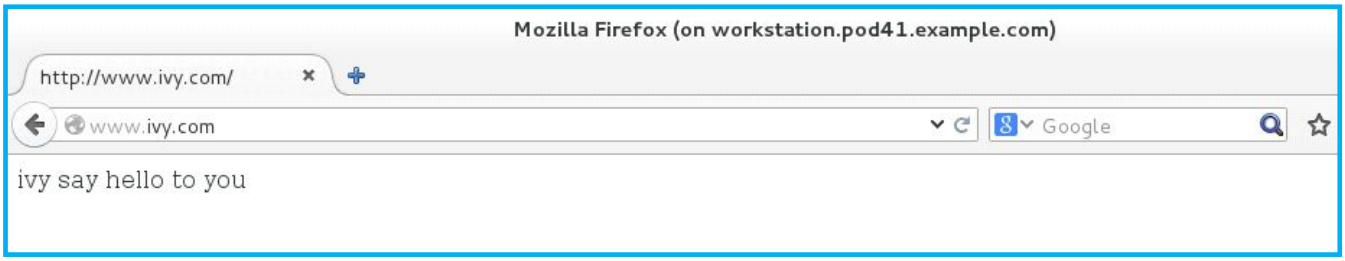
（4）重启 nginx 服务

```
[root@serverb conf.d]# systemctl restart nginx.service
```

（5）workstation 客户端测试。

注：需先添加 hosts 解析

```
[root@workstation ~]# echo 172.25.41.10 www.ivy.com >> /etc/hosts
```



（6）配置自定义虚拟主机 www.joy.com

```
[root@serverb conf.d]#cp default.conf www.joy.com.conf
[root@serverb conf.d]#vim www.joy.com.conf
server {
    listen          80;
```

```
    server_name    www.joy.com;

    #charset koi8-r;
    #浏览器默认字符集
    #access_log    /var/log/nginx/log/host.access.log    main;
    #访问日志存放位置
    location / {
        root     /usr/share/nginx/joy.com;
        index    index.html index.htm;
    }
    #location 用来匹配$path，即 url 地址中的路径部分，匹配成功后大括号中的内容生效，否则不生效
```

（7）创建该虚拟主机网页文件存放目录及首页文件

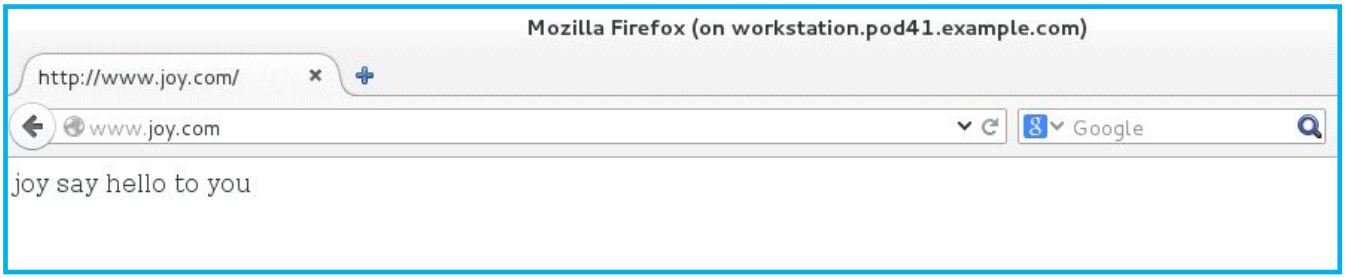
```
[root@serverb conf.d]# mkdir /usr/share/nginx/joy.com
[root@serverb conf.d]# echo joy say hello to you >/usr/share/nginx/joy.com/index.html
```

（8）重启 nginx 服务

```
[root@serverb conf.d]# systemctl restart nginx.service
```

（9）workstation 客户端测试。

```
[root@workstation ~]# echo 172.25.41.10 www.joy.com >> /etc/hosts
```



第 3.2.4 节　Nginx 基于名称的虚拟主机其他配置

（1）server_name

nginx 中 server_name（即虚拟主机名称）配置字段也支持通配符和正则表达式等

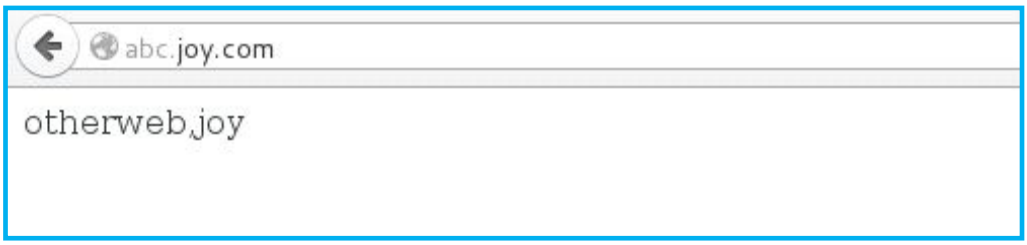
通配符实例：

```
[root@serverb conf.d]# cp default.conf any.joy.com.conf
[root@serverb conf.d]# vim any.joy.com.conf
server {
    listen          80;
    server_name    *.joy.com;
    #此处 server_name 定义采用的即为通配符，目的是不管客户端的 url 路径中的主机名字段是什么，只要以.joy.com 结尾，即可访问到该虚拟主机所对应的网页文件
    location / {
        root     /usr/share/nginx/otherjoy.com;
        index    index.html index.htm;
    }
```

```
#创建该虚拟主机的网页文件存放目录及首页文件
[root@serverb conf.d]# mkdir /usr/share/nginx/otherjoy.com
[root@serverb conf.d]# echo otherweb.joy > /usr/share/nginx/otherjoy.com/index.html
#重启 nginx 服务
[root@serverb conf.d]# systemctl restart nginx.service
```

workstation 客户端测试

```
[root@workstation ~]# echo 172.25.41.10 abc.joy.com >> /etc/hosts
```



正则表达式实例

正则表达式匹配符号：

~ 匹配

~* 不区分大小写匹配

!~ 不匹配

!~* 不区分大小写的不匹配

下述实例中，如果主机名匹配是 www.ivy.com，则去/usr/share/nginx/ivy.com/目录中找相应的网页文件，如果主机名匹配是 wwwX.ivy.com,则去/usr/share/nginx/wwwivy.com/目录中找相应的网页文件，如果主机名匹配是 X.ivy.com,则去到/usr/share/nginx/anyivy.com/目录中找相应的网页文件。

```
server {
    listen          80;
    server_name    www.ivy.com;
    root     /usr/share/nginx/ivy.com;
    index    index.html index.htm;
}
server {
    listen          80;
    server_name    ~^www.*\.ivy\.com;
    root     /usr/share/nginx/wwwivy.com;
    index    index.html index.htm;
}
server {
    listen          80;
    server_name    ~.*\.ivy\.com;
    root     /usr/share/nginx/anyivy.com;
    index    index.html index.htm;
}
```

创建相应的网页文件根目录和首页测试文件

```
[root@serverb conf.d]# mkdir /usr/share/nginx/wwwivy.com /usr/share/nginx/anyivy.com
[root@serverb conf.d]# echo www web > /usr/share/nginx/wwwivy.com/index.html
[root@serverb conf.d]# echo any ivy web > /usr/share/nginx/anyivy.com/index.html
[root@serverb conf.d]# systemctl restart nginx.service
```

重启 nginx 服务

```
[root@serverb conf.d]# systemctl restart nginx.service
```

workstation 客户端测试

```
#客户端 hosts 解析
172.25.41.10 www1.ivy.com
172.25.41.10 wwwdfrelkthewlkthwelkt.ivy.com
172.25.41.10 bbs.ivy.com
```

www web

www web

any ivy web

（2）location 字段匹配

location 中可使用正则表达式匹配，并且在匹配过程中有符号的匹配比没有符号的匹配优先级

下述例子中，将访问 www.joy.com 时如果访问的$path 匹配到/ivy.html 字符串，则去到/usr/share/nginx/ivy.com/目录下找请求的网页文件，$path 匹配了一下两个 location，但是有符号的 location 优先级高。

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

    location / {
        root     /usr/share/nginx/joy.com;
        index    index.html index.htm;
    }
    location ~ /ivy.html {
        root     /usr/share/nginx/ivy.com;
        index    index.html index.htm;
    }
```

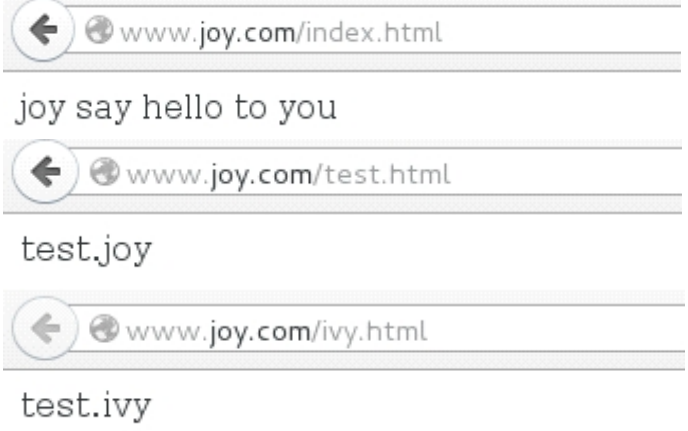
创建需要的测试文件，在 www.joy.com 虚拟主机的网页文件根目录下创建测试文件 test.html 文件，在 www.ivy.com 虚拟主机的网页文件根目录下创建 ivy.html 文件。

```
[root@serverb conf.d]# cd /usr/share/nginx/joy.com/
[root@serverb joy.com]# echo test.joy > test.html
[root@serverb joy.com]# cd ../ivy.com/
[root@serverb ivy.com]# echo test.ivy > ivy.html
```

重启 nginx 服务

```
[root@serverb ivy.com]# systemctl restart nginx.service
```

workstation 客户端测试。如下测试中$path 是 index.html 和 test.html 是访问的都是/usr/share/nginx/joy.com 目录下的文件，但$path 如果匹配到 ivy.html 文件则访问的是/usr/share/nginx/ivy.com/目录下的文件。



（3）用于返回报错信息的虚拟主机配置

配置了基于名称的虚拟主机后，如果客户端的 url 地址中主机名字段并没有采用主机名，而是采用了 ip 地址，则会返回/usr/share/nginx/目录下排在第一位的子目录中的首页文件，如下所示。

一般生产环境中，如果没有匹配到用户指定的 url 地址中主机名字段，不是返回随机的页面，而是返回一个事先配置的默认页面，一般页面内容为报错。

（1）创建虚拟主机配置文件

```
[root@serverb conf.d]# cp default.conf err.com.conf
[root@serverb conf.d]# vim err.com.conf
server {
    listen          80 default;
     #在监听端口号后边添加 default 字段，即代表该虚拟主机为默认虚拟主机

    location / {
        root    /usr/share/nginx/err.com;
        index   index.html index.htm;
```

```
    }
```

（2）创建该虚拟主机网页文件存放目录和首页文件，一般首页文件中即为报错信息

```
[root@serverb conf.d]# mkdir /usr/share/nginx/err.com
[root@serverb conf.d]# echo cannot found this virtual host name > /usr/share/nginx/err.com/index.html
```

（3）重启 nginx 服务

```
[root@serverb conf.d]# systemctl restart nginx.service
```

（4）workstation 客户端测试

第 3.3 节　安全的连接 https

众所周知，我们在互联网上冲浪，一般都是使用的 http 协议（超文本传输协议），默认情况下数据是明文传送的，这些数据在传输过程中都可能会被捕获和窃听，因此是不安全的。https 可以说是 http 协议的安全版，就是为了满足对安全性要求比较高的用户而设计的。

如果您的邮件中有敏感数据，不希望被人窃听；如果您不希望被钓鱼网站盗用帐号信息，如果您希望您在使用邮箱的过程中更安全，那么我们推荐您使用 https 安全连接。

HTTPS 在 HTTP 的基础上加入了 SSL 协议，SSL 依靠证书来验证服务器的身份，并为浏览器和服务器之间的通信加密。SSL (Secure Socket Layer 安全套接层)为 Netscape 所研发，用以保障在 Internet 上数据传输之安全，利用数据加密技术，可确保数据在网络上之传输过程中不会被截取及窃听。数据密码方式常见的有三种

（1）对称加密：采用单钥密码系统的加密方法，同一个密钥可以同时用作信息的加密和解密，这种加密方法称为对称加密，也称为单密钥加密。服务端生成公钥和私钥，服务端将公钥传递给客户端，客户端通过公钥加密自己的数据后传递给服务器。

举例来说：甲和乙是一对生意搭档，他们住在不同的城市。由于生意上的需要，他们经常会相互之间邮寄重要的货物。为了保证货物的安全，他们商定制作一个保险盒，将物品放入其中。他们打造了两把相同的钥匙分别保管，以便在收到包裹时用这个钥匙打开保险盒，以及在邮寄货物前用这把钥匙锁上保险盒。只要甲乙小心保管好钥匙，那么就算有人得到保险盒，也无法打开。

对称加密算法的优点是算法公开、计算量小、加密速度快、加密效率高。

对称加密算法的缺点是在数据传送前，发送方和接收方必须商定好秘钥，然后使双方都能保存好秘钥。其次如果一方的秘钥被泄露，那么加密信息也就不安全了。另外，每对用户每次使用对称加密算法时，都需要使用其他人不知道的唯一秘钥，这会使得收、发双方所拥有的钥匙数量巨大，密钥管理成为双方的负担。

（2）非对称加密

非对称加密算法需要两个密钥来进行加密和解密，公钥和私钥。还是上述例子，乙方生成一对密钥并将公钥向甲方公开，得到该公钥的甲方使用该密钥对机密信息进行加密后再发送给乙方。乙方再用自己保存的私钥对加密后的信息进行解密。乙方只能用私钥解密由对应的公钥加密后的信息。同样，如果乙要回复加密信息给甲，那么需要甲先公布甲的公钥给乙用于加密，甲自己保存甲的私钥用于解密。在传输过程中，即使攻击者截获了传输的密文，并得到了乙的公钥，也无法破解密文，因为只有乙的私钥才能解密密文。

非对称加密与安全性更好，使用一对秘钥，一个用来加密，一个用来解密，而且公钥是公开的，秘钥是自己保存的。

非对称加密的缺点是加密和解密花费时间长、速度慢，只适合对少量数据进行加密。

（3）单向加密

Nginx 的 https 配置说明

（1）在 serverb 机器上创建私钥

```
[root@serverb ~]# openssl genrsa 2048 > serverb-web.key
```

（2）在 serverb 机器上创建证书的颁发请求

```
[root@serverb ~]# openssl req -new -key serverb-web.key -out serverb-web.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:shanghai
Locality Name (eg, city) [Default City]:shanghai
Organization Name (eg, company) [Default Company Ltd]:redhat.org ivy company
Organizational Unit Name (eg, section) []:www.redhat.com
Common Name (eg, your name or your server's hostname) []:www.redhat.com
Email Address []:root@test.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

（3）将证书颁发请求提交给 ca（实验环境以 servera 机器模拟 ca）

```
[root@serverb ~]# scp serverb-web.csr servera:/root
```

（4）将 servera 设置为 ca

```
[root@servera ~]# openssl genrsa -des3 -out ca.key 4096
[root@servera ~]# openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
Enter pass phrase for ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:shanghai
```

```
Locality Name (eg, city) [Default City]:shanghai
Organization Name (eg, company) [Default Company Ltd]:ca.shanghai
Organizational Unit Name (eg, section) []:ca shanghai
Common Name (eg, your name or your server's hostname) []:ca.shanghai.com
Email Address []:root@shanghai.com
```

（5）ca 经过资料审核后为 serverb 机器创建证书

```
[root@servera ~]# openssl x509 -req -days 365 -in serverb-web.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out
serverb-web.crt
Signature ok
subject=/C=CN/ST=shanghai/L=shanghai/O=redhat.org ivy
company/OU=www.redhat.com/CN=www.redhat.com/emailAddress=root@test.com
Getting CA Private Key
Enter pass phrase for ca.key:
```

（6）证书创建成功将证书颁发给 serverb

```
[root@servera ~]# scp serverb-web.crt serverb:/root
```

（7）serverb 取回合法证书后将证书拷贝至系统默认目录下

```
[root@serverb ~]# cp serverb-web.crt /etc/pki/tls/certs/
[root@serverb ~]# cp serverb-web.key /etc/pki/tls/private/
[root@serverb ~]# chmod 400 /etc/pki/tls/private/serverb-web.key
```

（8）配置 ssl 网站，nginx 默认有 ssl 的网站配置示例

```
[root@serverb ~]# cd /etc/nginx/conf.d/
[root@serverb conf.d]# cp example_ssl.conf www.redhat.com-ssl.conf
[root@serverb ~]# vim /etc/nginx/conf.d/www.redhat.com-ssl.conf
server {
    listen          443 ssl;
    server_name    www.redhat.com;

    ssl_certificate         /etc/pki/tls/certs/serverb-web.crt;
    ssl_certificate_key    /etc/pki/tls/private/serverb-web.key;
#以上两句在指定证书所存放的路径
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout    5m;

    ssl_ciphers    HIGH:!aNULL:!MD5;
#指出允许的密码,密码指定为 OpenSSL 支持的格式
    ssl_prefer_server_ciphers     on;
#依赖 SSLv3 和 TLSv1 协议的服务器密码将优先于客户端密码
    location / {
        root     /usr/share/nginx/redhatssl.com;
        index    index.html index.htm;
#ssl 网站的网页文件根目录存放位置
    }
```

```
}
```

（9）创建 ssl 网站的网页文件存放目录

```
[root@serverb ~]# mkdir /usr/share/nginx/redhatssl.com
[root@serverb ~]# echo redhatssl web > /usr/share/nginx/redhatssl.com/index.html
```

（10）重启 nginx 服务

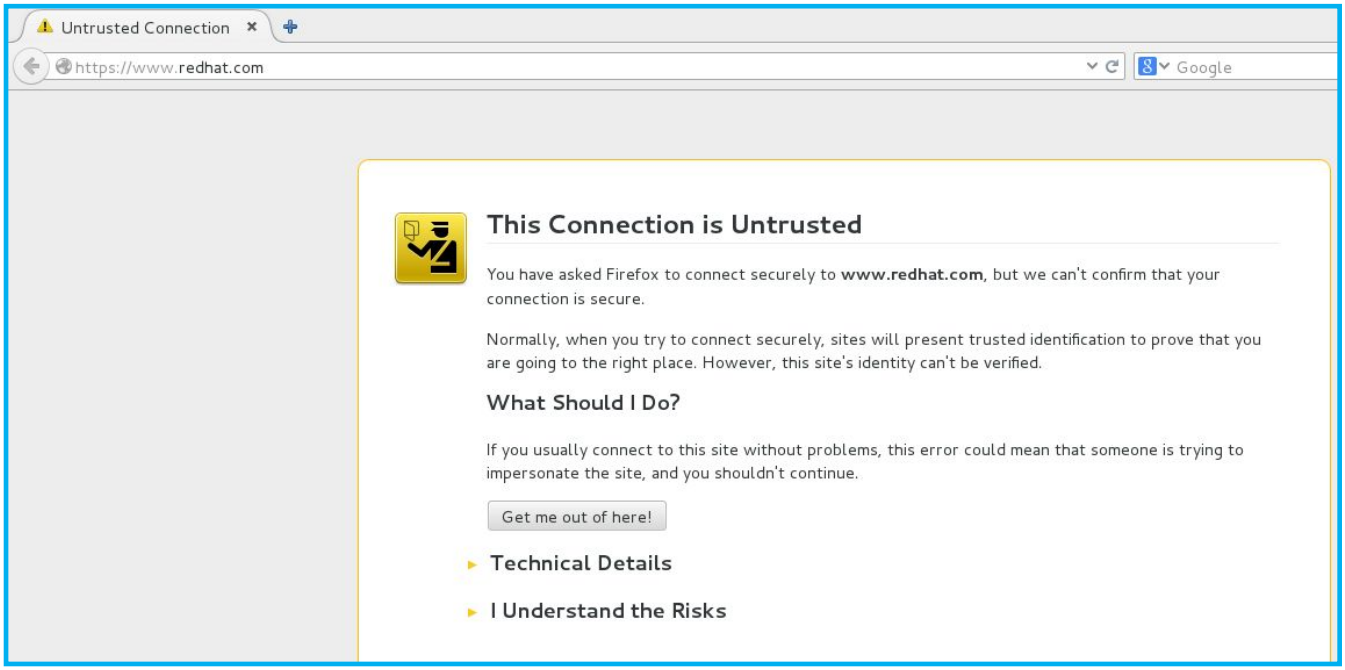
```
[root@serverb ~]# systemctl restart nginx.service
```

（11）在 servera 添加相应的端口转发规则，https 默认监听 443 端口

```
[root@servera ~]# iptables -t nat -A PREROUTING -d 172.25.41.10 -i eth0 -p tcp --dport 443 -j DNAT
--to-destination 192.168.0.11:443
[root@servera ~]# iptables-save > /etc/sysconfig/iptables
```

（12）客户端 workstation 机器测试。弹出警告页面，因为我们不是权威 ca 颁发的证书，是自己为自己颁发的伪造证书（做实验嘛），所以浏览器会给出警告。客户端手动添加信任即可。

```
[root@workstation ~]# echo 172.25.41.10 www.redhat.com >> /etc/hosts
```



为了让大家配置方便，服务器共享目录中放置了证书创建的脚本，运行脚本就可以创建 serverb 机器所需证书。也就是使用脚本取去替代上述步骤中 1 到 6 步。

```
[root@serverb ~]#mount 172.25.254.250:/content /mnt/
[root@serverb ~]# cd /mnt/item/nginx
```

```
[root@serverb nginx]# cp mkcert.sh   ~
[root@serverb nginx]# cd   ~
[root@serverb ~]# ./mkcert.sh --create-ca-keys
[root@serverb ~]# ./mkcert.sh  –create-web-keys
[root@serverb ~]# scp /etc/pki/CA/my-ca.crt 172.25.41.9:/tmp/
[root@serverb ~]# cp /etc/pki/CA/web_server.crt /etc/pki/tls/certs/
[root@serverb ~]# cp /etc/pki/CA/web_server.key /etc/pki/tls/private/
[root@serverb ~]# chmod 400 /etc/pki/tls/private/web_server.key
[root@serverb private]# vim /etc/nginx/conf.d/www.redhat.com-ssl.conf
server {
    listen          443 ssl;
    server_name   www.redhat.com;

    ssl_certificate         /etc/pki/tls/certs/web_server.crt;
    ssl_certificate_key   /etc/pki/tls/private/web_server.key;

    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout   5m;

    ssl_ciphers   HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers    on;

    location / {
        root    /usr/share/nginx/redhatssl.com;
        index   index.html index.htm;
    }
}
[root@serverb ~]# systemctl restart nginx.service
```

第 3.4 节　Nginx 日志管理

做为一个优秀的管理员，应该能够对你所维护的服务器了如指掌，那么日志的阅读就是必不可少的，我们可以通过日志，了解到你的服务器是否正常运行，有谁在访问你的服务器，每天的访问量是多少，有没有非法的不受欢迎的访问等等。

在 Nginx 主配置中有关 Nginx 日志的相关参数主要有两条。

```
log_format     指定日志格式
access_log     指定日志存放路径
```

对于一个日均 PV 数十万以上的网站来说，日志的增长将会很迅速，一天的日志几个 G 属于正常情况。所以为了我们单个日志文件不至于太大，我们需要定期的去截断我们的日志。不可很可惜，Nginx 默认并不支持我们所熟悉的 cronolog 方式来对每天的日志进行截断，那么我们可以通过以下方法来解决。

```
[root@example ~]# mv /usr/local/nginx/logs/access.log /usr/local/nginx/logs/access.log1
```

```
[root@example ~]# touch /usr/local/nginx/logs/access.log
[root@example ~]# kill -HUP `cat /usr/local/nginx/logs/nginx.pid`
```

然后以 crond 的方式或者 logrotate 的方式运行。

对于习惯使用 cronolog 来做日志分割的管理员来说，无法使用 cronolog 总会成为他心里挥之不去的痛。然而有时候事情总会很奇妙，只要你想去解决，就一定有办法解决。我在这里介绍一种通过管道符文件另僻溪径的方法使用 cronolog。

首先下载并编译安装我们 cronolog

```
[root@example ~]# tar xf cronolog-1.6.2.tar.gz
[root@example ~]# ./configure
[root@example ~]# make && make install
[root@example ~]# service nginx stop
[root@example ~]# mv /usr/local/nginx/logs/access.log /usr/local/nginx/logs/access.log.bak
[root@example ~]# mkfifo    /usr/local/nginx/logs/access.log
[root@example ~]# cronolog /usr/local/nginx/logs/access.log.%Y%m%d < /usr/local/nginx/logs/access.log &
[root@example ~]# service nginx start
```

访问一下自己的主机，然后查看/usr/local/nginx/logs/目录下是否多出个 access.log.%Y%m%d 文件，查看一下这个文件里的内容是否是我们的访问日志。

这里需要注意的是必需在 Nginx 启动之前执行 cronolog 命令，否则 Nginx 将无法正常启动。

对于一些你不需要记录的日志内容，我们可以使用以下方式关闭对这些文件的日志记录。

```
location ~ .*\.(js|jpg|JPG|jpeg|JPEG|css|bmp|gif|GIF)$
      {
            access_log off;
```

第 3.5 节　访问控制

（1）有时我们会有这么一种需求，就是你的网站并不想提供一个公共的访问或者某些页面不希望公开，我们希望的是某些特定的客户端可以访问。那么我们可以在访问时要求进行身份认证，就如给你自己的家门加一把锁，以拒绝那些不速之客。我们在服务课程中学习过 apache 的访问控制，对于 Nginx 来说同样可以实现，并且整个过程和 Apache 非常的相似。

```
location / {
                root     html;
                index    index.html index.htm;
                auth_basic    "xxxx";
                auth_basic_user_file /usr/share/nginx/passwd.db;
          }
```

设置密码

```
[root@example nginx]# htpasswd -cm /usr/share/nginx/passwd.db mark
New password: （输入密码）
Re-type new password: （again）
```

```
Adding password for user mark
```

（2）想阻止别人访某些目录下的特定文件，比如我网站主目录下有个 test 目录，我不想让别人访问我 test 目录下的”.txt”和”.doc”的文件，那么我们可以通过 deny 的方式来做拒绝。

```
location ~* \.(txt|doc)$ {
    deny all;
        }
```

此处 ~* 代表不区分大小写方式匹配。

此处 all 代表是客户端的 IP，你也可以使用 ip/mask 的方式来定义。

第 3.6 节　课后练习

1.配置两个基于名称的虚拟主机 www.test.com 和 www.example.com （注意，做此实验之前你应该已经有正确的 DNS 配置）。

修改 Nginx 主配置文件，增加两个 server 字段。

创建相关网站主目录并重启 Nginx 服务。

2.配置一个 https。以使你的服务器可以提供一个安全的连接。

生成安全密钥。

修改 Nginx 主配置文件，加入 ssl 相关配置

重启 Nginx 服务。

3.配置你的虚拟主机（www.test.com），当别人试图访问时，要求进行身份认证，允许 mark 用户可以通过密码 uplooking 进行访问。

修改 Nginx 主配置文件，加入 auth 相关配置。

创建认证使用密码文件

重启 Nginx 服务。

第 4 章　Rewrite 规则

第 4.1 节　什么是 Rewrite

Rewrite 对称 URL Rewrite，即 URL 重写，就是把传入 Web 的请求重定向到其他 URL 的过程。URL Rewrite 最常见的应用是 URL 伪静态化 ,是将动态页面显示为静态页面方式的一种技术。比如 http://www.123.com/news/index.asp?id=123 使用 URLRewrite 转换后可以显示为 http://www.123.com/news/123.html

对于追求完美主义的网站设计师，就算是网页的地址也希望看起来尽量简洁明快。形如 http://www.123.com/news/index.asp?id=123 的网页地址，自然是毫无美感可言，而用 UrlRewrite 技术，你可以轻松把它显示为 http://www.123.com/news/123.html。

理论上，搜索引擎更喜欢静态页面形式的网页，搜索引擎对静态页面的评分一般要高于动态页面。所以，UrlRewrite 可以让我们网站的网页更容易被搜索引擎所收录。

从安全角度上讲，如果在 url 中暴露太多的参数，无疑会造成一定量的信息泄漏，可能会被一些黑客利用，对你的系统造成一定的破坏，所以静态化的 url 地址可以给我们带来更高的安全性。

第 4.2 节　Rewrite 语法

Nginx Rewrite 为地址改写，在改写之前需先匹配。

匹配：

1.location：只匹配$path,不匹配主机名等其他信息

2.if：可以匹配除路径以外的其他信息，如主机名、客户端 ip 等。

if 的语法应用于 server 和 location 环境内

```
if (condition) { ... }  #匹配表达式，匹配成功则执行大括号中的地址重写
```

if 可以支持如下条件判断匹配符号

```
~ 为区分大小写匹配
~* 为不区分大小写匹配
!~和!~*分别为区分大小写不匹配及不区分大小写不匹配
-f 和!-f 用来判断是否存在文件
-d 和!-d 用来判断是否存在目录
-e 和!-e 用来判断是否存在文件或目录
-x 和!-x 用来判断文件是否可执行
```

在匹配过程中可以引用一些 Nginx 的全局变量，参考 http://wiki.nginx.org/NginxHttpCoreModule 的 Variables

```
$args, 请求中的参数;
$document_root, 针对当前请求的根路径设置值;
$http_host, 请求信息中的"Host"，如果请求中没有 Host 行，则等于设置的服务器名;
$limit_rate, 对连接速率的限制;
$request_method, 请求的方法，比如"GET"、"POST"等;
$remote_addr, 客户端地址;
```

```
$remote_port, 客户端端口号;
$remote_user, 客户端用户名，认证用;
$request_filename, 当前请求的文件路径名
$query_string, 与$args 相同;
$scheme, 所用的协议，比如 http 或者是 https
$server_protocol, 请求的协议版本，"HTTP/1.0"或"HTTP/1.1";
$server_addr, 服务器地址，如果没有用 listen 指明服务器地址，使用这个变量将发起一次系统调用以取得地址(造成资源浪费);
$server_name, 请求到达的服务器名;
$document_uri 与$uri 一样，URI 地址;
$server_port, 请求到达的服务器端口号;
```

例 4-1 匹配访问的 url 地址是否是个目录

```
if (-d $request_filename) {
        ...;
    }
```

例 4-2 匹配访问的地址是否以 www 开头

```
if ($hosts ~* ^www) {
        ...;
    }
```

改写：

{rewrite |旧地址 新地址 标记位}

每行 rewrite 指令最后应该根一个 flag 标记，支持的 flag 标记有，其中最为常用的有 last、break、permanent

```
last        相当于 Apache 里的[L]标记，表示完成 rewrite
break       本条规则匹配完成后，终止匹配，不再匹配后面的规则
redirect    返回 302 临时重定向，浏览器地址会显示跳转后的 URL 地址
permanent   返回 301 永久重定向，浏览器地址会显示跳转后 URL 地址
```

last 和 break 标记的区别在于，last 标记在本条 rewrite 规则执行完后，会对其所在的 server { ... } 标签重新发起请求，而 break 标记则在本条规则匹配完成后，停止匹配，不再做后续的匹配。另有些时候必须使用 last，比如在使用 alias 指令时，而使用 proxy_pass 指令（见后面章节）时则必须使用 break。

redirect 和 permanent 区别则是返回的不同方式的重定向 ,对于客户端来说一般状态下是没有区别的。而对于搜索引擎，相对来说 301 的重定向更加友好，如果我们把一个地址采用 301 跳转方式跳转的话，搜索引擎会把老地址的相关信息带到新地址，同时在搜索引擎索引库中彻底废弃掉原先的老地址。 使用 302 重定向时，搜索引擎(特别是 google)有时会查看跳转前后哪个网址更直观，然后决定显示哪个，如果它觉的跳转前的 URL 更好的话，也许地址栏不会更改，那么很有可能出现 URL 劫持的现像。

例 4-3 用户在访问 www.joy.com 网站的 news 目录时，我这个目录还在建设中，那么想要实现的就是用户访问该目录下任何一个文件，返回的都是首页文件，给用户以提示。

```
#进入 www.joy.com 的网页文件根目录，该目录下创建 news 测试目录，在 news 下创建一些测试文件，测试文件中包含 index.html
[root@serverb /]# cd /usr/share/nginx/joy.com/
[root@serverb joy.com]# mkdir news
```

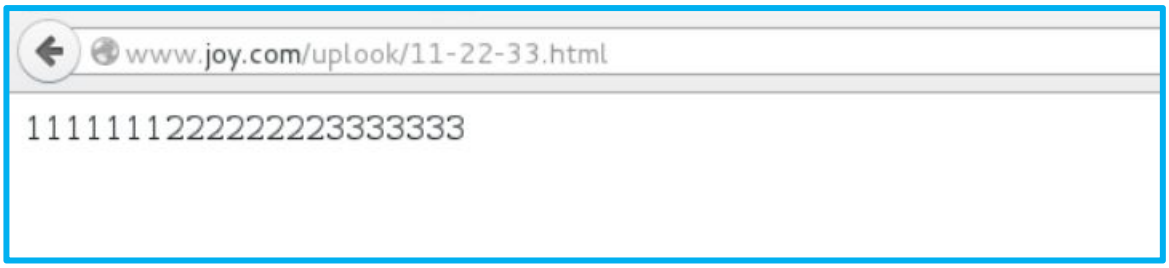
```
[root@serverb joy.com]# cd news/
[root@serverb news]# echo building > index.html     ^C
[root@serverb news]# touch new1.html new2.html
#编写 rewrite，因为是匹配路径，所以使用 location 即可，如下所示，如果$path 中匹配到/news/，则将从/news/开始后边匹配任意字符的地址重写为/news/index.html，标记位使用 break，停止匹配，如果使用 last 则出现死循环，因为改写完成的地址中仍然包含/news，使用 last 会进行二次改写。
[root@serverb news]# vim /etc/nginx/conf.d/www.joy.com.conf
server {
    listen          80;
    server_name    www.joy.com;

    #charset koi8-r;
    #access_log    /var/log/nginx/log/host.access.log    main;
    root     /usr/share/nginx/joy.com;
    index    index.html index.htm;
    location ~ /ivy.html {
        root     /usr/share/nginx/ivy.com;
        index    index.html index.htm;
    }
    location ~* /news {
    rewrite ^/news/.* /news/index.html break;
    }
#重启 nginx 服务
[root@serverb news]# systemctl restart nginx.service
```

客户端 workstation 机器测试

例 4-4 用户在访问 www.joy.com 网站时访问地址如果为/uplook/11-22-33.html 重写后真实地址为/uplook/11/22/33.html

注：做 URI 重写时，有时会发现 URI 中含有相关参数，如果需要将这些参数保存下来，并且在重写过程中重新引用，我们可以用到( )进行保存和$N 进行引用的方式

```
#创建测试目录和文件
[root@serverb ~]# cd /usr/share/nginx/joy.com/
[root@serverb joy.com]# mkdir uplook
[root@serverb joy.com]# mkdir -p uplook/11/22
[root@serverb joy.com]# cd uplook/11/22
[root@serverb 22]# echo 111111122222223333333 > 33.html
#编写 rewrite，location 匹配$path,改写中使用到正则表达式，旧地址中^表示以某个字符串开始，[ ]表示匹配字符组的任一字符，+表示匹配前一字符 1 次及 1 次以上，\表示转义，（ ）用于字符串保存，$表示以某个字符串结尾，新地址中$用于引用保存的内存，$1 表示第一个被保存的字符串，依次类推。
[root@serverb 22]# vim /etc/nginx/conf.d/www.joy.com.conf
    location ~* /uplook {
        rewrite ^/uplook/([0-9]+)-([0-9]+)-([0-9]+)\.html$ /uplook/$1/$2/$3.html last;
    }
#重启 nginx 服务
[root@serverb 22]# systemctl restart nginx.service
```



例 4-5 用户在访问 tom.joy.com 网站时访问的实际位置为/usr/share/nginx/joy.com/tom 目录下的文件

```
[root@serverb ~]# vim /etc/nginx/conf.d/www.joy.com.conf
server {
    listen          80;
    server_name   ~.*.joy.com;
    root    /usr/share/nginx/joy.com;
    index   index.html index.htm;
    if ( $http_host ~* ^www\.joy\.com$ )  {
```

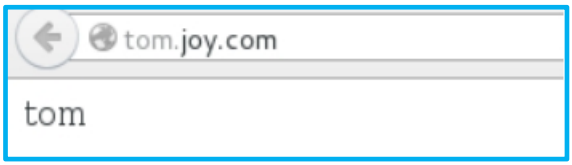
```
        break;
    }
    #如果用户访问的是 www.joy.com，则不做 rewrite
    if ( $http_host ~* ^(.*)\.joy\.com$ )  {
        set $domain $1;
        rewrite /.* /$domain/index.html break;
    }
#将用户输入的.joy.com 之前的字符串保存并将保存的内存赋值给 domain 变量
```

创建测试文件

```
[root@serverb ~]# cd /usr/share/nginx/joy.com/
[root@serverb joy.com]# mkdir tom
[root@serverb joy.com]# echo tom > tom/index.html
[root@serverb ~]# systemctl restart nginx.service
```

客户端测试

```
[root@workstation ~]# echo 172.25.41.10 tom.joy.com >> /etc/hosts
```



第 5 章　Nginx 反向代理

第 5.1 节　反向代理及其特点

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

反向代理又称为 Web 服务器加速，是针对 Web 服务器提供加速功能的。它作为代理 Cache，但并不针对浏览器用户，而针对一台或多台特定 Web 服务器（这也是反向代理名称的由来）。代理服务器可以缓存一些 web 的页面，降低了 web 服务器的访问量，所以可以降低 web 服务器的负载。web 服务器同时处理的请求数少了，响应时间自然就快了。同时代理服务器也存了一些页面，可以直接返回给客户端，加速客户端浏览。实施反向代理，只要将反向代理设备放置在一台或多台 Web 服务器前端即可。当互联网用户访问某个 WEB 服务器时，通过 DNS 服务器解析后的 IP 地址是代理服务器的 IP 地址,而非原始 Web 服务器的 IP 地址,这时代理服务器设备充当 Web 服务器，浏览器可以与它连接，无需再直接与 Web 服务器相连。因此，大量 Web 服务工作量被转载到反向代理服务上。不但能够很大程度上减轻 web 服务器的负担，提高访问速度，而且能够防止外部网主机直接和 web 服务器直接通信带来的安全隐患。

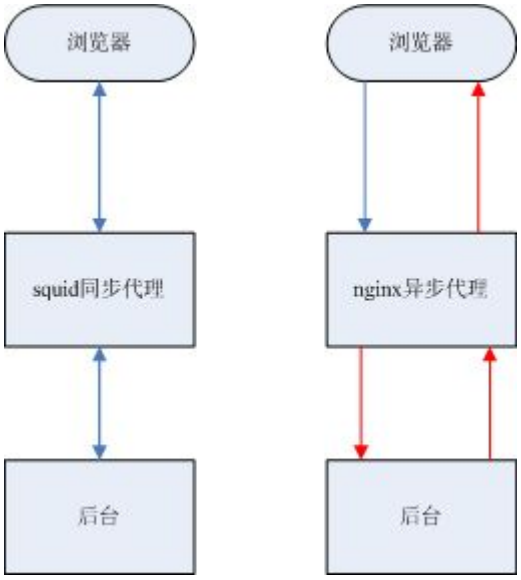
第 5.2 节　Nginx 反向代理

Nginx proxy 是 Nginx 的王牌功能，利用 proxy 基本可以实现一个完整的 7 层负载均衡，它有这 些特色：

1. 功能强大，性能卓越，运行稳定。
2. 配置简单灵活。
3. 能够自动剔除工作不正常的后端服务器。
4. 上传文件使用异步模式。
5. 支持多种分配策略，可以分配权重，分配方式灵活。

在 nginx 的反向代理优点介绍中，提到了异步传输模式 下面就来讲解下传统的代理（apache/squid）的同步传输和 nginx 的异步传输的差异。

看图：



squid 同步传输：浏览器发起请求，而后请求会立刻被转到后台，于是在浏览器和后台之间就建立了一个通道。在请求发起直到请求完成，这条通道都是一直存 在的。

nginx 异步传输：浏览器发起请求，请求不会立刻转到后台，而是将请求数据（header）先收到 nginx 上，然后 nginx 再把这个请求发到后端，后端处理完之后把数据返回到 nginx 上 ,nginx 将数据流发到浏览器 ,这点和 lighttpd 有点不同 ,lighttpd 是将后端数据完全接收后才发 送到浏览器。

那么这到底有什么好处呢？

1) 假设用户执行一个上传文件操作，因为用户网速又比较慢，因此需要花半个小时才能把文件传到服务器。squid 的同步代理在用户开始上传后就和后台建立了连 接，半小时后文件上传结束，由此可见，后台服务器连接保持了半个小时；而 nginx 异步代理就是先将此文件收到 nginx 上，因此仅仅是 nginx 和用户 保持了半小时连接，后台服务器在这半小时内没有为这个请求开启连接，半小时后用户上传结束，nginx 才将上传内容发到后台，nginx 和后台之间的带宽 是很充裕的，所以只花了一秒钟就将请求发送到了后台，由此可见，后台服务器连接保持了一秒。同步传输花了后台服务器半个小时，异步传输只花一秒，可见优化 程度很大。

2) 在上面这个例子中，假如后台服务器因为种种原因重启了，上传文件就自然中断了，这对用户来说是非常恼火的一件事情，想必各位也有上传文件传到一半被中断的 经历。用 nginx 代理之后，后台服务器的重启对用户上传的影响减少到了极点，而 nginx 是非常稳定的并不需要常去重启它，即使需要重启，利用 kill -HUP 就可以做到不间断重启 nginx。

3) 异步传输可以令负载均衡器更有保障，为什么这么说呢？在其它的均衡器（lvs/haproxy/apache 等）里，每个请求都是只有一次机会的，假如用 户发起一个请求，结果该请求分到的后台服务器刚好挂掉了，那么这个请求就失败了；而 nginx 因为是异步的，所以这个请求可以重新发往下一个后台，下一个 后台返回了正常的数据，于是这个请求就能成功了。还是用用户上传文件这个例子，假如不但用了 nginx 代理，而且用了负载均衡，nginx 把上传文件发往 其中一台后台，但这台服务器突然重启了，nginx 收到错误后，会将这个上传文件发到另一台后台，于是用户就不用再花半小时上传一遍。

4) 假如用户上传一个 10GB 大小的文件，而后台服务器没有考虑到这个情况，那么后台服务器岂不要崩溃了。用 nginx 就可以把这些东西都拦在 nginx 上， 通过 nginx 的上传文件大小限制功能来限制，另外 nginx 性能非常有保障，就放心的让互联网上那些另类的用户和 nginx 对抗去吧。

第 5.3 节　　Nginx 反向代理配置

第 5.3.1 节　一对一

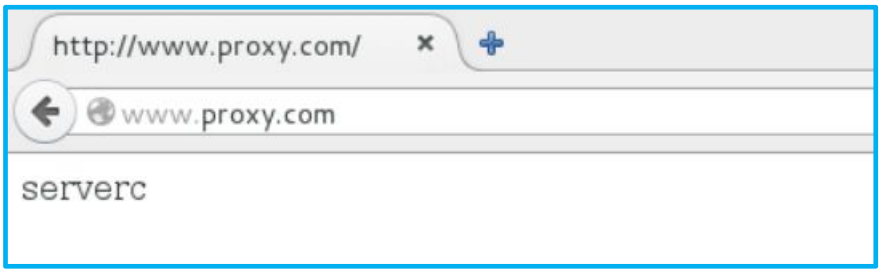
在 serverc 搭建 apache，serverb 作为代理服务器，访问 serverc 的 apache 服务的用户请求，发给 serverb。

```
[root@serverc ~]# yum install httpd
[root@serverc ~]# cd /var/www/html/
[root@serverc html]# echo serverc > index.html
[root@serverc html]# systemctl enable    httpd.service
[root@serverc html]# systemctl start httpd.service
```

```
[root@serverb ~]# cd /etc/nginx/conf.d/
[root@serverb conf.d]# vim www.proxy.com.conf
server {
      listen          80;
      server_name    www.proxy.com;
      location / {
            proxy_pass http://192.168.0.12;
       #确定需要代理的 URL，端口或 socket。
      }
```

```
[root@serverb conf.d]# systemctl restart nginx.service
```

```
[root@workstation ~]#echo 172.25.41.10 www.proxy.com >> /etc/hosts
```



第 5.3.2 节　一对多

后台 server 是两台或两台以上，如下实验中，serverc 和 serverd 都是由 serverb 机器代理的服务器

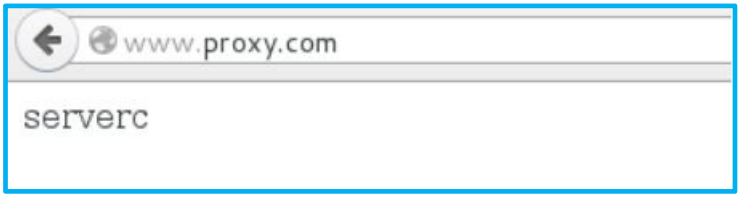
```
[root@serverb ~]# vim /etc/nginx/nginx.conf
      upstream apache-servers {
```

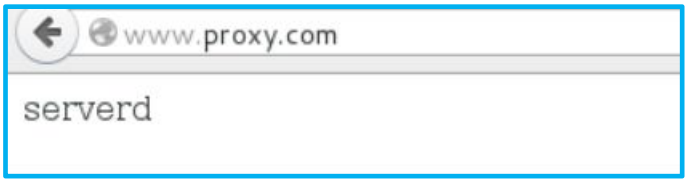
```
        server 192.168.0.12:80;
        server 192.168.0.13:80;
}
#HTTP 负载均衡模块。upstream 这个字段设置一群服务器，可以将这个字段放在 proxy_pass 和 fastcgi_pass 指令中作为一个单独的实体，它们可以是监听不同端口的服务器，并且也可以是同时监听 TCP 和 Unix socket 的服务器。 服务器可以指定不同的权重，默认为 1。
```

```
[root@serverb ~]# vim /etc/nginx/conf.d/www.proxy.com.conf
    location / {
        proxy_pass http://apache-servers;
     #确定需要代理的 URL，端口或 socket。
        proxy_set_header Host $host;
     #这个指令允许将发送到后端服务器的请求头重新定义或者增加一些字段。 这个值可以是一个文本，变量或者它们的组合。
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_404;
        proxy_set_header X-Real-IP $remote_addr;
     #确定在何种情况下请求将转发到下一个服务器：
     #error - 在连接到一个服务器，发送一个请求，或者读取应答时发生错误。
     #timeout - 在连接到服务器，转发请求或者读取应答时发生超时。
     #invalid_header - 服务器返回空的或者错误的应答。
     #http_500 - 服务器返回 500 代码。
     #http_502 - 服务器返回 502 代码。
     #http_503 - 服务器返回 503 代码。
     #http_504 - 服务器返回 504 代码。
     #http_404 - 服务器返回 404 代码。
     #off - 禁止转发请求到下一台服务器。
        proxy_redirect off;
     #如果需要修改从后端服务器传来的应答头中的"Location"和"Refresh"字段，可以用这个指令设置。
    }
```

重启 nginx 服务

```
[root@serverb ~]# systemctl restart nginx.service
```





第 5.3.3 节　upstream

作用：HTTP 负载均衡模块。upstream 这个字段设置一群服务器，可以将这个字段放在 proxy_pass 和 fastcgi_pass 指令中作为一个单独的实体，它们可以是监听不同端口的服务器，并且也可以是同时监听 TCP 和 Unix socket 的服务器。

分配方式：

- 轮询（默认）

  每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器 down 掉，能自动剔除。

- weight

  指定轮询几率，weight 和访问比率成正比，用于后端服务器性能不均的情况。

  例如：

```
upstream bakend {
        server 192.168.0.12:80 weight=4;
        server 192.168.0.13:80 weight=1;
}
```

- ip_hash

  每个请求按访问 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。

  例如：

```
upstream bakend {
        ip_hash;
        server 192.168.0.12:80;
        server 192.168.0.13:80;
}
```

- 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。

  例如：

```
 upstream bakend {
      server 192.168.0.12:80;
      server 192.168.0.13:80;
      fair;
  }
```

- url_hash（第三方）

  按访问 url 的 hash 结果来分配请求，使每个 url 定向到同一个后端服务器，后端服务器为缓存时比较有效。

  例如：

```
 upstream bakend {
      server 192.168.0.12:80;
      server 192.168.0.13:80;
      fair;
  }
```

设备状态值：

- down 表示单前的 server 暂时不参与负载
- weight 默认为 1.weight 越大，负载的权重就越大。
- max_fails ：允许请求失败的次数默认为 1.当超过最大次数时，返回 proxy_next_upstream 模块定义的错误
- fail_timeout:max_fails 次失败后，暂停的时间。
- backup： 其它所有的非 backup 机器 down 或者忙的时候，请求 backup 机器。所以这台机器压力会最轻。

Nginx 支持同时设置多组的负载均衡，用来给不用的 server 来使用。

第 5.3.4 节 nginx 缓存

修改全局配置选项，指定缓存文件放置位置

[root@serverb ~]# vim /etc/nginx/nginx.conf

```
proxy_temp_path     /usr/share/nginx/proxy_temp_dir 1 2;
proxy_cache_path /usr/share/nginx/proxy_cache_dir levels=1:2 keys_zone=cache_web:50m inactive=1d
```

```
max_size=30g;
        upstream apache-servers {
                server 192.168.0.12:80 weight=4;
                server 192.168.0.13:80 weight=1;
}
```

建立全局配置项中的缓存文件放置位置，并将目录的拥有者设置为 nginx

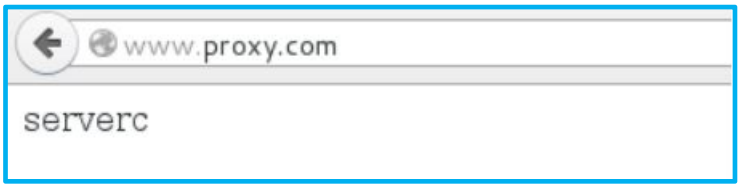
```
[root@serverb ~]# mkdir -p /usr/share/nginx/proxy_temp_dir /usr/share/nginx/proxy_cache_dir
[root@serverb ~]# chown nginx /usr/share/nginx/proxy_temp_dir/ /usr/share/nginx/proxy_cache_dir/
[root@serverb ~]# vim /etc/nginx/conf.d/www.proxy.com.conf
        location / {
                proxy_pass http://apache-servers;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_404;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_redirect off;

                client_max_body_size            10m;
                client_body_buffer_size       128k;
                proxy_connect_timeout          90;
                proxy_send_timeout              90;
                proxy_read_timeout              90;
                proxy_cache cache_web;
                proxy_cache_valid 200 302 12h;
                proxy_cache_valid 301 1d;
                proxy_cache_valid any 1h;
                proxy_buffer_size                4k;
                proxy_buffers                     4 32k;
                proxy_busy_buffers_size       64k;
                proxy_temp_file_write_size 64k;
        #对缓存做相关设置，如缓存文件大小等
        }
```

```
[root@serverb ~]# systemctl restart nginx.service
```

没访问之前查看缓存目录，目录为空

```
[root@serverb ~]# ll /usr/share/nginx/proxy_cache_dir/
total 0
```

Workstation 客户端访问



访问完成后再次查看缓存目录，有相应缓存文件，也可将后台服务停掉，对于缓存文件，客户端仍然可以访问到。

```
[root@serverb ~]#ll /usr/share/nginx/proxy_cache_dir/
drwx------ 3 nginx nginx 15 Dec  9 15:00 4
[root@serverc html]# systemctl stop httpd.service
```

第 6 章　lnmp 项目搭建

第 6.1 节　搭建过程

一：安装软件包

1.安装 CGI 管理器

用户在访问过程中如果访问动态页面，apache 和 nginx 都解释不了 php 页面，需要调用相应的 php 进程，apache 在调用过程中使用的 libphp5.so 的模块，因为 apache 的配置文件中加载模块相应配置，nginx 没有，默认不能加载模块，所以需要安装额外程序，php 进程管理器。Php 进程管理器比较常见的有 spawn-fcgi 和 php-fpm。

Spawn-cfgi 和 php-fpm 相比，后者性能更高，但后者必须和 php 程序版本完全一致，如果升级了 php，那么 php-fpm 程序也要做相应的版本升级。

本例中使用的 spawn-fcgi 程序。

Php 进程管理器在 lnmp 环境中的作用：

（1）监听端口，nginx 把请求交给 php 管理器，php 管理器监听 9000 端口

（2）调用和管理 php 进程，管理程序去看你本地有没有 php 命令，有的话调用起来，php 命令运行之后再去处理刚才收到的页面请求，处理 php 请求

```
[root@serverb ~]# mount 172.25.254.250:/content /mnt    ^C
[root@serverb ~]# cd /mnt/item/nginx/epel/
[root@serverb epel]# rpm -ivh spawn-fcgi-1.6.3-5.el7.x86_64.rpm
```

2.安装 php 程序、数据库程序和 php 连接数据库的驱动

```
[root@serverb epel]# yum install php php-mysql mariadb-server -y
```

二：配置虚拟主机

```
[root@serverb epel]# cd /etc/nginx/conf.d/
[root@serverb conf.d]# cp default.conf www.bbs.com.conf
[root@serverb conf.d]# vim www.bbs.com.conf
server {
    listen          80;
    server_name    www.bbs.com;
    root     /usr/share/nginx/bbs.com;
    index    index.php index.html index.htm;

    location ~ \.php$ {
        fastcgi_pass     127.0.0.1:9000;
```

```
        fastcgi_index   index.php;
        fastcgi_param   SCRIPT_FILENAME   /usr/share/nginx/bbs.com$fastcgi_script_name;
        include         fastcgi_params;
    }
[root@serverb conf.d]# mkdir /usr/share/nginx/bbs.com
[root@serverb conf.d]# systemctl restart nginx.service
```

三．配置和启动 spawn-fcgi 程序

```
[root@serverb conf.d]# vim /etc/sysconfig/spawn-fcgi
OPTIONS="-u nginx -g nginx -p 9000 -C 32 -F 1 -P /var/run/spawn-fcgi.pid -- /usr/bin/php-cgi"
[root@serverb conf.d]# systemctl start spawn-fcgi
[root@serverb conf.d]# systemctl enable spawn-fcgi
```

四．配置和启动数据库程序

```
[root@serverb conf.d]# systemctl enable mariadb.service
[root@serverb conf.d]# systemctl start mariadb.service
[root@serverb conf.d]# mysqladmin -u root password "uplooking"
```

五．创建数据文件

```
[root@serverb ~]# cd /mnt/item/php/
[root@serverb php]# cp Discuz_X3.2_SC_UTF8.zip /tmp/
[root@serverb php]# cd /tmp/
[root@serverb tmp]# unzip Discuz_X3.2_SC_UTF8.zip
[root@serverb tmp]# cp -r upload/* /usr/share/nginx/bbs.com/
[root@serverb tmp]# chown nginx /usr/share/nginx/bbs.com/ -R
```

六．客户端测试

```
[root@workstation ~]# echo 172.25.41.10 www.bbs.com >> /etc/hosts
```


www.bbs.com/install/
Google
Discuz! 安装向导
DiscuzX3.2 简体中文 UTF8 版 20150609
中文版授权协议 适用于中文用户
版权所有 (c) 2001-2013，北京康盛新创科技有限责任公司保留所有权利。
感谢您选择康盛产品。希望我们的努力能为您提供一个高效快速、强大的站点解决方案，和强大的社区论坛解决方案。康盛公司网址为 http://www.comsenz.com，产品官方讨论区网址为 http://www.discuz.net。
用户须知：本协议是您与康盛公司之间关于您使用康盛公司提供的各种软件产品及服务的法律协议。无论您是个人或组织、盈利与否、用途如何（包括以学习和研究为目的），均需仔细阅读本协议，包括免除或者限制康盛责任的免责条款及对您的权利限制。请您审阅并接受或不接受本服务条款。如您不同意本服务条款及/或康盛随时对其的修改，您应不使用或主动取消康盛公司提供的康盛产品。否则，您的任何对康盛产品中的相关服务的注册、登陆、下载、查看等使用行为将被视为您对本服务条款全部的完全接受，包括接受康盛对服务条款随时所做的任何修改。
本服务条款一旦发生变更，康盛将在网页上公布修改内容。修改后的服务条款一旦在网站管理后台上公布即有效代替原来的服务条款。您可随时登陆康盛官方论坛查阅最新版服务条款。如果您选择接受本条款，即表示您同意接受协议各项条件的约束。如果您不同意本服务条款，则不能获得使用本服务的权利。您若有违反本条款规定，康盛公司有权随时中止或终止您对康盛产品的使用资格并保留追究相关法律责任的权利。
我同意
我不同意
©2001 - 2013 Comsenz Inc.


www.bbs.com/forum.php
设为首页 收藏本站
社区动力
DISCUZ!
论坛
请输入搜索内容
帖子
热搜: 活动 交友 discuz
论坛
今日: 0 | 昨日: 0 | 帖子: 0 | 会员: 1 | 欢迎新会员: admin
Discuz!
默认版块
在线会员 - 1 人在线 - 0 会员(0 隐身), 1 位游客 - 最高记录是 1 于 2015-12-9.
管理员
超级版主
版主
会员
当前只有游客或隐身会员在线

第 7 章　程序迁移/复制

第 7.1 节　迁移/复制步骤

程序迁移步骤

程序迁移-->配置文件迁移-->数据文件迁移-->地址修改-->权限相关

第 7.2 节　数据库程序步骤

（1）迁移 mariadb-server 程序

```
[root@serveri ~]# yum -y install mariadb-server
```

（2）启动数据库服务

```
[root@serveri ~]# systemctl start mariadb
```

（3）将 serverb（旧的数据库服务器）上的数据库导出备份到一个文件中

```
[root@serverb ~]# mysqldump --all-databases -uroot -puplooking > /tmp/mariadb.all.sql
```

（4）将导出的文件拷贝至新的数据库服务器 serveri

```
[root@serverb ~]# scp /tmp/mariadb.all.sql 172.25.0.18:/tmp/
```

（5）在 serveri 机器上将导出的数据库导入

```
[root@serveri ~]# mysql < /tmp/mariadb.all.sql
```

（6）修改 php 代码，将 dbhost 改为新的数据库服务器

```
[root@serverb ~]# for i in $(find /usr/share/nginx/bbs.com/ -name *.php);do grep -q "uplooking" $i && echo $i;done
[root@serverb ~]# vim /usr/share/nginx/bbs.com/config/config_global.php
[root@serverb ~]# vim /usr/share/nginx/bbs.com/config/config_ucenter.php
[root@serverb ~]# vim /usr/share/nginx/bbs.com/uc_server/data/config.inc.php
```

（7）授权。允许 php 程序所在机器读取数据库中的内容。

```
[root@serveri ~]# mysqladmin -u root password "uplooking"
[root@serveri ~]# echo "grant all on *.* to root@'172.25.0.11' identified    by 'uplooking';" | mysql -uroot -puplooking
[root@serveri ~]# echo "grant all on *.* to root@'172.25.0.1' identified    by 'uplooking';" | mysql -uroot -puplooking
[root@serveri ~]# echo "grant all on *.* to root@'serverb.pod0.example.com' identified    by 'uplooking';" | mysql -uroot -puplooking
[root@serveri ~]# mysqladmin -uroot -puplooking flush-privileges
```

第 7.3 节　PHP 程序迁移

（1）安装 php php-mysql spawn-fcgi 程序

```
[root@serverc ~]# yum -y install php php-mysql
[root@serverc ~]# mount 172.25.254.250:/content /mnt/
[root@serverc ~]# cd /mnt/items/nginx/nginx-rpms/
[root@serverc nginx-rpms]# rpm -ivh spawn-fcgi-1.6.3-5.el7.x86_64.rpm
```

（2）迁移配置文件。

```
[root@serverb ~]# scp /etc/sysconfig/spawn-fcgi 172.25.0.12:/etc/sysconfig/
```

（3）启动 spawn-fcgi 程序

```
[root@serverc bbs.com]# systemctl start spawn-fcgi.service
```

（4）迁移数据文件

```
[root@serverb ~]# tar cf /tmp/datafile.tar /usr/share/nginx/bbs.com/
[root@serverb ~]# scp /tmp/datafile.tar 172.25.0.12:/tmp/
```

（5）修改虚拟主机配置文件，访问 php 的请求交给新的 php 进程管理器所在机器做处理。

```
[root@serverb ~]# vim /etc/nginx/conf.d/www.bbs.com.conf
        location ~ \.php$ {
                fastcgi_pass      172.25.0.12:9000;
                fastcgi_index     index.php;
                fastcgi_param     SCRIPT_FILENAME     /usr/share/nginx/bbs.com$fastcgi_script_name;
                include             fastcgi_params;
}
```

（6）重启 nginx 服务

```
[root@serverb ~]# systemctl restart nginx.service
```

（7）在 serverc 机器上添加 nginx 用户。使 php 页面 nginx 拥有读写权限

```
[root@serverc bbs.com]# groupadd -g 994 nginx
[root@serverc bbs.com]# useradd -u 996 -g nginx nginx
```

（8）数据库授权

```
[root@serveri ~]# echo "grant all on *.* to root@'172.25.0.12' identified    by 'uplooking';" | mysql -uroot -puplooking
[root@serveri ~]# echo "grant all on *.* to root@'serverc.pod0.example.com' identified    by 'uplooking';" | mysql -uroot -puplooking
[root@serveri ~]# mysqladmin -uroot -puplooking flush-privileges
```

第 7.4 节　PHP 程序复制

通过程序拆分操作，每台服务器上已经只运行一个程序，但是可能还是不能够处理大量的用户请求，我们就可以使用程序复制，也就是多台机器使用跑同一个程序，负载均衡。以 php 程序复制为例。

第 7.4.1 节　程序复制

（1）进入服务器公共目录，安装 php 进程管理器 spawn-fcfgi

```
[root@servere ~]# mount 172.25.254.250:/content /mnt/
[root@servere ~]# cd /mnt/items/nginx/nginx-rpms/
[root@servere ~]# rpm -ivh spawn-fcgi-1.6.3-5.el7.x86_64.rpm
```

（2）安装 php 程序和 php 连接 mariadb 的驱动

```
[root@servere ~]# yum -y install php php-mysql
```

（3）将第一台 cgi 服务器上的配置文件复制到第二台 cgi 服务器上

```
[root@serverc ~]# scp /etc/sysconfig/spawn-fcgi 172.25.0.14:/etc/sysconfig/
```

（4）将第一台 cgi 服务器上的 php 页面文件复制到第二台 cgi 服务器上

```
[root@serverc ~]# tar cf /tmp/data.tar /usr/share/nginx/bbs.com/
[root@serverc ~]# scp /tmp/data.tar 172.25.0.14:/tmp/
[root@servere ~]# tar xf /tmp/data.tar -C /
```

（5）定义 upsteam 字段，地址池中包含后台两台 cgi 服务器，以便 fastcgi_cgi 字段引用

```
[root@serverb ~]# vim /etc/nginx/nginx.conf
        upstream php_pools {
                server 172.25.0.12:9000;
                server 172.25.0.14:9000;
}
```

（6）php 文件的访问请求交给后台 cgi 服务器

```
[root@serverb ~]# vim /etc/nginx/conf.d/www.bbs.com.conf
            location ~ \.php$ {
                fastcgi_pass     php_pools;
                fastcgi_index    index.php;
                fastcgi_param    SCRIPT_FILENAME    /usr/share/nginx/bbs.com$fastcgi_script_name;
                include          fastcgi_params;
}
```

（7）修改新的 cgi 服务器上 UGO 权限，保证 nginx 用户对所有的 php 文件有读写权限

```
[root@servere ~]#groupadd -g 994 nginx
[root@servere ~]#useradd -u 996 -g nginx nginx
[root@servere ~]#systemctl start spawn-fcgi.service
```

（8）数据库授权

```
[root@serveri ~]# echo "grant all on *.* to root@'172.25.0.14' identified by 'uplooking';" | mysql -uroot -puplooking
[root@serveri ~]# echo "grant all on *.* to root@'servere.pod0.example.com' identified by 'uplooking';" | mysql -uroot -puplooking
[root@serveri ~]# mysqladmin -uroot -puplooking flush-privileges
```

（9）客户端测试，两台 cgi 服务器是否都能够正常工作。

第 7.4.2 节　数据一致性问题

如果后台是两台 cgi 服务器，那么会存在数据一致性问题。比如，A 用户上传图片到论坛的请求经过轮询被提交到 serverc 机器，那么这张图片就会被保存到 serverc 机器，以后如果 B 用户的请求被提交到 serverc 机器，那么 B 用户可以访问下载该图片，但是如果用户 C 请求经过轮询之后被提交到 servere 机器，那么用户 C 是不能浏览下载这张图片的，会出现报错（如下图所示）。原因就是因为图片被上传到 serverc 机器，servere 机器上没有这张图片。

可通过共享存储来解决数据不一致的问题。

（1）serverj 作为共享存储服务器，安装上 nfs_utils 和 rpcbind 两个软件包（默认已安装）

```
[root@serverj ~]# rpm -q nfs-utils
nfs-utils-1.3.0-0.8.el7.x86_64
[root@serverj ~]# rpm -q rpcbind
rpcbind-0.2.0-26.el7.x86_64
```

（2）将其中一台 cgi 服务器上的 php 页面文件拷贝至共享存储服务器。

```
[root@serverc ~]# tar cf /tmp/data1.tar /usr/share/nginx/bbs.com/
```

```
[root@serverc ~]# tar cf /tmp/data1.tar /usr/share/nginx/bbs.com/
[root@serverj ~]# tar -xf /tmp/data1.tar -C /
```

（3）添加 nginx 用户和组。

```
[root@serverj  ~]# groupadd -g 994 nginx
[root@serverj  ~]# useradd -u 996 -g nginx nginx
```

（4）配置 nfs。

```
[root@serverj  ~]# vim /etc/exports
/usr/share/nginx/bbs.com 172.25.0.0/255.255.255.0(rw)
```

（5）启动 rpc 服务和 nfs 服务。

```
[root@serverj  ~]# systemctl start rpcbind
[root@serverj bbs.com]# systemctl restart nfs-server
```

（6）serverc 和 servere 作为 nfs 客户端去挂载共享存储服务器上共享出来的目录。（永久挂载写到/etc/fstab 文件）

```
[root@serverc ~]# mount 172.25.0.19:/usr/share/nginx/bbs.com /usr/share/nginx/bbs.com
[root@servere ~]# mount 172.25.0.19:/usr/share/nginx/bbs.com /usr/share/nginx/bbs.com
```

（7）客户端测试。再上传图片。（图略）