第1天-Nginx进阶基础

一、Nginx 介绍

Nginx (engine x) 是一个高性能的 HTTP 和 反向代理 服务，也是一个IMAP/POP3/SMTP服务。Nginx是由伊戈尔·赛索耶夫为俄罗斯访问量第二的Rambler.ru站点（俄文：Рамблер）开发的，第一个公开版本0.1.0发布于2004年10月4日。其将源代码以类BSD许可证的形式发布，因它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。2011年6月1日，nginx 1.0.4发布。

Nginx是一款轻量级的Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，并在一个BSD-like 协议下发行。其特点是占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好，中国大陆使用nginx网站用户有：百度、京东、新浪、网易、腾讯、淘宝等。

在高连接并发的情况下，Nginx是Apache服务器不错的替代品。

创始人伊戈尔·赛索耶夫

二、为什么选择 Nginx

1、特性介绍

Nginx 是一个高性能的 Web 和反向代理服务器, 它具有有很多非常优越的特性:

作为 Web 服务器：可以作为HTTP 代理服务器和反向代理服务器，支持通过缓存加速访问，可以完成简单的负载均衡和容错，支持包过滤功能，支持SSL等，相比 Apache，Nginx 使用更少的资源，支持更多的并发连接，体现更高的效率，这点使 Nginx 尤其受到虚拟主机提供商的欢迎。能够支持高达 50,000 个并发连接数的响应，感谢 Nginx 为我们选择了 epoll and kqueue 作为开发模型.

作为负载均衡服务器：可以进行自定义配置，支持虚拟主机，支持URL重定向，支持网络监控，支持流媒体传输等。Nginx 既可以在内部直接支持 Rails 和 PHP，也可以支持作为 HTTP代理服务器 对外进行服务。Nginx 用 C 编写, 不论是系统资源开销还是 CPU 使用效率都比 Perlbal 要好的多。

作为邮件代理服务器: Nginx 同时也是一个非常优秀的邮件代理服务器（最早开发这个产品的目的之一也是作为邮件代理服务器），他支持IMAP/POP3 代理服务功能，支持内部SMTP代理服务功能。

Nginx 安装非常的简单，配置文件 非常简洁（还能够支持perl语法），Bugs非常少的服务器: Nginx 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进行软件版本的升级。

2、Nginx VS Apache

1、内核和功能上的比较

特性	Nginx	Apache
请求管理	事件驱动模型 使用异步套接字处理，较少了内存和CPU开销	同步套接字、进程和线程 每个请求都要使用一个单独的进程或线程，使用同步套接字
设计语言	C	C、C++
可移植性	多平台	多平台
诞生时间	2002	1994

2、一般功能比较

功能	Nginx	Apache
HTTPS支持	作为模块支持	作为模块支持
虚拟主机	原生支持	原生支持
CGI支持	仅支持FastCGI	支持CGI和FastCGI
系统模块	静态模块系统	动态模块系统

从以上功能上的对比，我们很难发现那些功能Apache无法实现。那我们为什么更喜欢用Nginx呢，Nginx 相对

Apache 有那些有点呢？

3、Nginx 相对 apache 的优点

- 轻量级，同样起web 服务比Apache 占用更少的内存及资源
- 静态处理，Nginx 静态处理性能比 Apache 高 3倍以上
- 抗并发，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。在Apache+PHP（prefork）模式下，如果PHP处理慢或者前端压力很大的情况下，很容易出Apache进程数飙升，从而拒绝服务的现象。
- 高度模块化的设计，编写模块相对简单
- 社区活跃，各种高性能模块出品迅速

三、Nginx IO多路复用

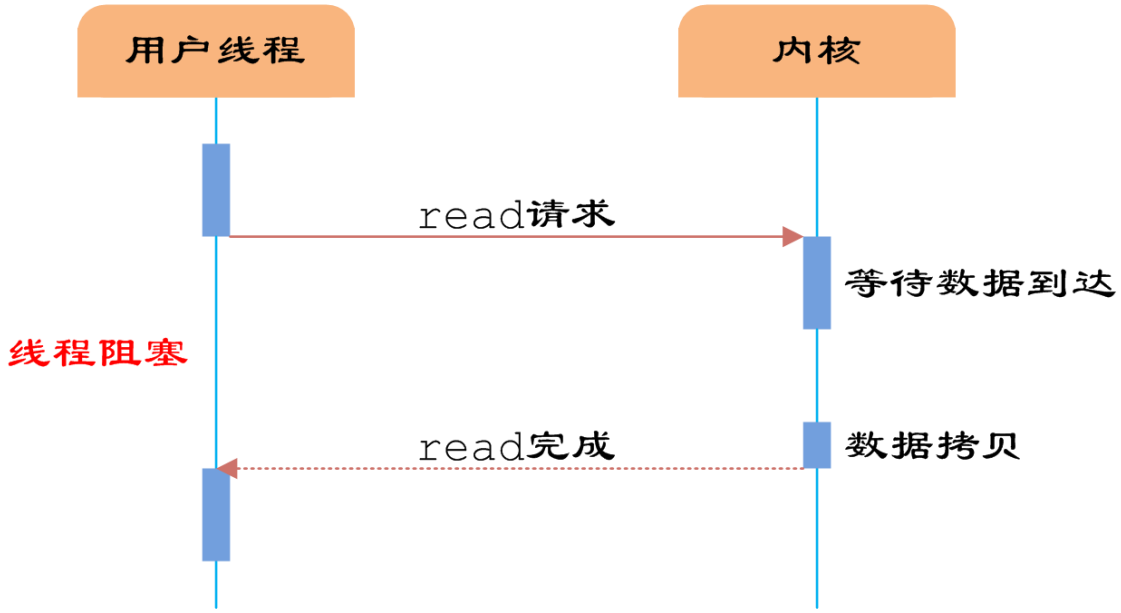
1、I/O模型

（1）同步阻塞IO（Blocking IO）：即传统的IO模型。

（2）同步非阻塞IO

（3）IO多路复用（IO Multiplexing）

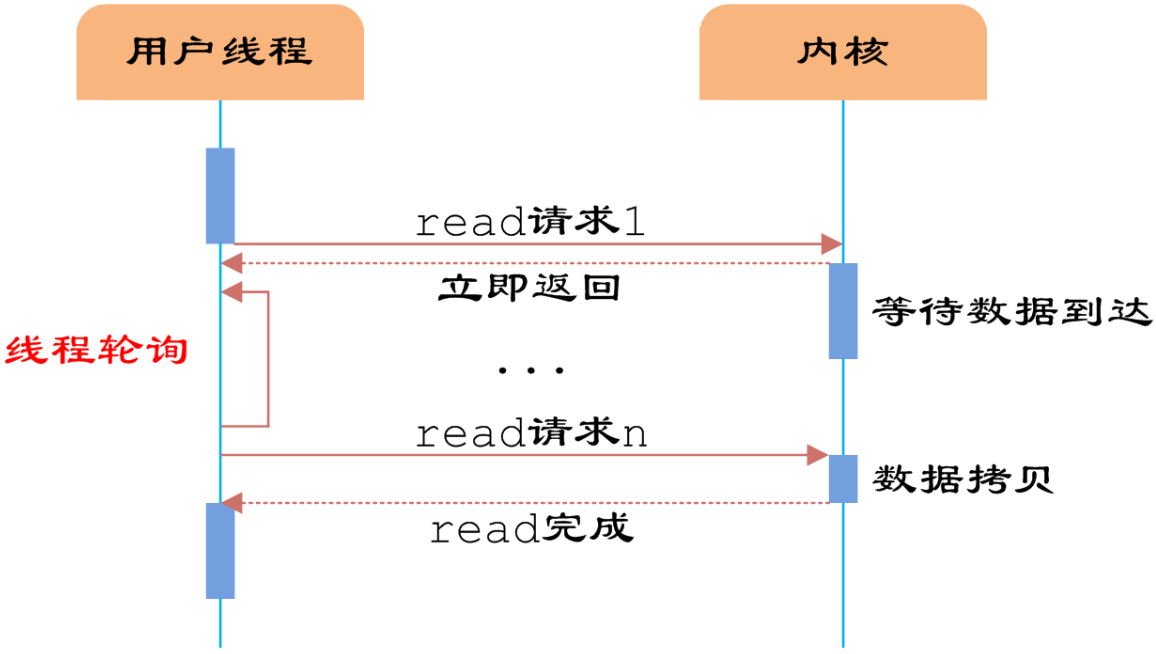
1）、同步阻塞



用户发起read IO读操作，由用户空间转到内核空间。内核等到数据包到达后，然后将接收的数据拷贝到用户空间，完成read操作。

用户需要等待read将socket中的数据读取到buffer后，才继续处理接收的数据。整个IO请求的过程中，用户线程是被阻塞的，这导致用户在发起IO请求时，不能做任何事情，对CPU的资源利用率不够。

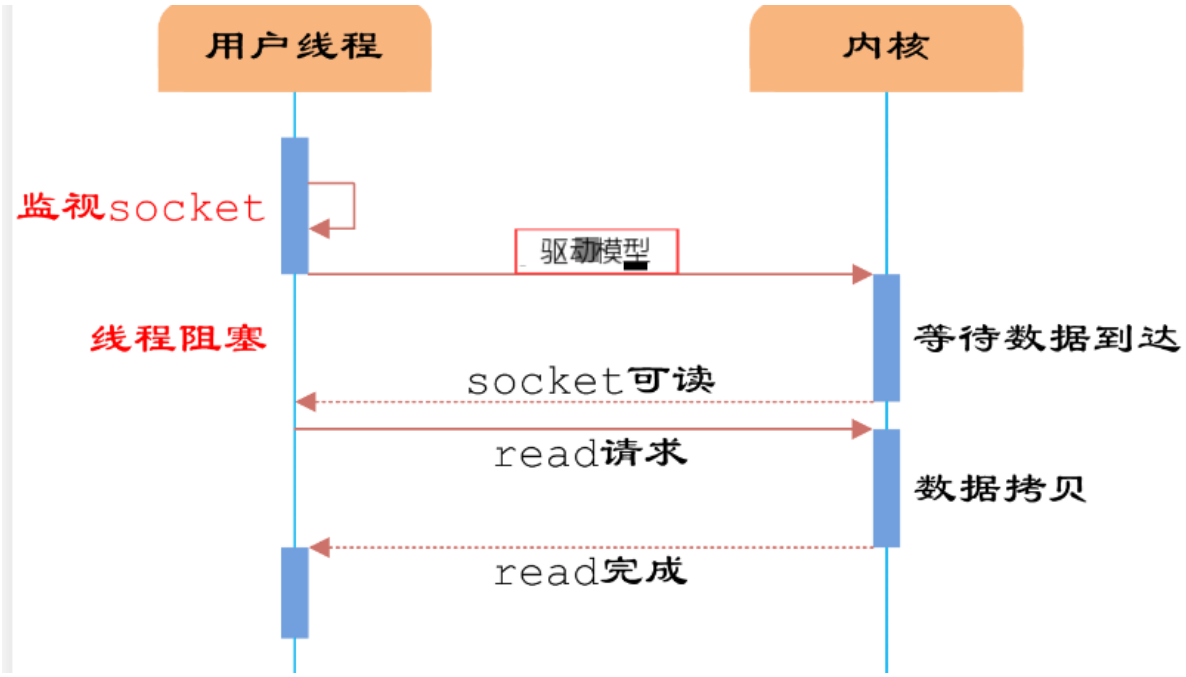
2）、同步非阻塞



用户线程发起IO请求时立即返回。但并未读取到任何数据，用户线程需要不断地发起IO请求，直到数据到达后，才真正读取到数据，继续执行。

用户需要不断地调用read，尝试读取socket中的数据，直到读取成功后，才继续处理接收的数据。整个IO请求的过程中，虽然用户线程每次发起IO请求后可以立即返回，但是为了等到数据，仍需要不断地轮询、重复请求，消耗了大量的CPU的资源。

3）、I/O多路复用



用户在发起请求建立起socket，首先将需要进行IO操作的socket添加到事件驱动模型中，然后等待驱动模型系统调用返回。当数据到达时，socket被激活，驱动模型函数返回。用户线程正式发起read请求，读取数据并继续执行。

但是，使用函数以后最大的优势是可以在一个线程内同时处理多个socket的IO请求。用户可以注册多个socket，然后不断地调用驱动模型读取被激活的socket，即可达到在同一个线程内同时处理多个IO请求的目的。

在同一个线程里面， 通过拨开关的方式，来同时传输多个I/O流

2、nginx 事件驱动

一个请求到来了，nginx接收请求的过程是怎样的?

ngnix会有很多连接进来， epoll会把他们都监视起来，然后像拨开关一样，谁有数据就拨向谁，然后调用相

应的代码处理。

- **select, poll, epoll** 都是I/O多路复用的具体的实现，其实是他们出现是有先后顺序的。

I/O多路复用这个概念被提出来以后， 相继出现了多个方案

- **select**是第一个实现 (1983 左右在BSD里面实现的)。

select 被实现以后，很快就暴露出了很多问题。

• select 会修改传入的参数数组，这个对于一个需要调用很多次的函数，是非常不友好的。

• select 如果任何一个sock(I/O stream)出现了数据，select 仅仅会返回，但是并不会告诉你是那个sock上有数

据，于是你只能自己一个一个的找，10几个sock可能还好，要是几万的sock每次都找一遍...

• select 只能监视1024个链接。

• select 不是线程安全的，如果你把一个sock加入到select, 然后突然另外一个线程发现，这个sock不用，要收

回，这个select 不支持的，如果你丧心病狂的竟然关掉这个sock, select的标准行为是不可预测的

- 于是14年以后(1997年）一帮人又实现了**poll,** poll 修复了select的很多问题，比如

• poll 去掉了1024个链接的限制，于是要多少链接呢， 主人你开心就好。

• poll 从设计上来说，不再修改传入数组，不过这个要看你的平台了，所以行走江湖，还是小心为妙。

其实拖14年那么久也不是效率问题， 而是那个时代的硬件实在太弱，一台服务器处理1千多个链接简直就是神

一样的存在了，select很长段时间已经满足需求。

但是poll仍然不是线程安全的， 这就意味着，不管服务器有多强悍，你也只能在一个线程里面处理一组I/O流。

你当然可以那多进程来配合了，不过然后你就有了多进程的各种问题。

- 于是5年以后, 在2002, 大神 Davide Libenzi 实现了**epoll**.

epoll 可以说是I/O 多路复用最新的一个实现，epoll 修复了poll 和select绝大部分问题, 比如：

• epoll 现在是线程安全的。

• epoll 现在不仅告诉你sock组里面数据，还会告诉你具体哪个sock有数据，你不用自己去找了。

3、异步非阻塞

```
$ pstree |grep nginx
|-+= 81666 root nginx: master process nginx
| |--- 82500 nobody nginx: worker process
| --- 82501 nobody nginx: worker process
```

1个master进程，2个work进程

每进来一个request，会有一个worker进程去处理。但不是全程的处理，处理到什么程度呢？处理

到可能发生阻塞的地方，比如向上游（后端）服务器转发request，并等待请求返回。那么，这个处理

的worker不会这么一直等着，他会在发送完请求后，注册一个事件："如果upstream返回了，告诉我一声，

我再接着干"。于是他就休息去了。这就是异步。此时，如果再有request 进来，他就可以很快再按这种

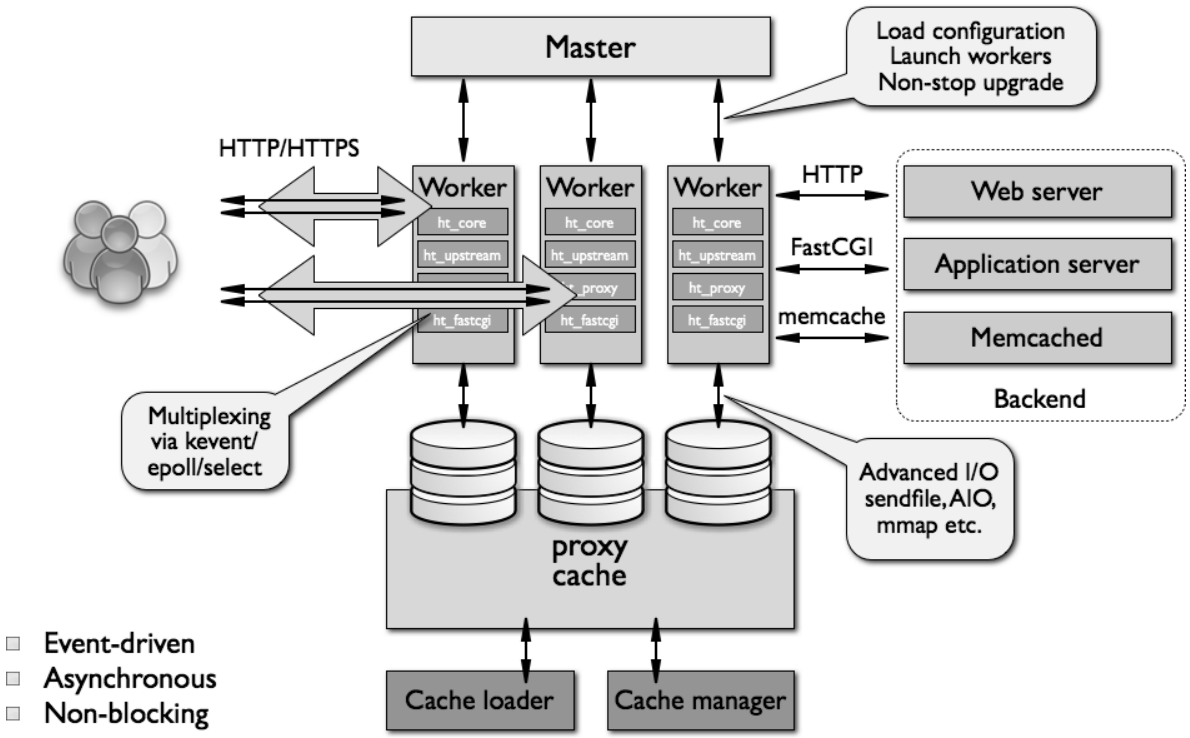
方式处理。这就是非阻塞和IO多路复用。而一旦上游服务器返回了，就会触发这个事件，worker才会来

接手，这个request才会接着往下走。这就是异步回调。

四、Nginx 的内部技术架构

Nginx服务器，以其处理网络请求的高并发、高性能及高效率，获得了行业界的广泛认可，近年已稳居web服务器部署排名第二的位置，并被广泛用于反向代理和负载均衡。

Nginx是如何实现这些目标的呢？答案就是其独特的内部技术架构设计。看懂下面这张图，就明白了Nginx的内部技术架构。



简要说明几点：

1、nginx启动时，会生成两种类型的进程，一个是主进程（Master），一个（windows版本的目前只有一个）或多个工作进程（Worker）。主进程并不处理网络请求，主要负责调度工作进程，也就是图示的三项：加载配置、启动工作进程及非停升级。所以，nginx启动以后，查看操作系统的进程列表，我们就能看到至少有两个nginx进程。

2、服务器实际处理网络请求及响应的是工作进程（worker），在类unix系统上，nginx可以配置多个worker，而每个worker进程都可以同时处理数以千计的网络请求。

3、模块化设计。nginx的worker，包括核心和功能性模块，核心模块负责维持一个运行循环（run-loop），执行网络请求处理的不同阶段的模块功能，如网络读写、存储读写、内容传输、外出过滤，以及将请求发往上游服务器等。而其代码的模块化设计，也使得我们可以根据需要对功能模块进行适当的选择和修改，编译成具有特定功能的服务器。

nginx将各功能模块组织成一条链，当有请求到达的时候，请求依次经过这条链上的部分或者全部模块，进行处理。每个模块实现特定的功能。例如，实现对请求解压缩的模块，实现SSI的模块，实现与上游服务器进行通讯的模块，实现与FastCGI服务进行通讯的模块。

五、Nginx 安装部署和配置管理

1、Nginx 部署-Yum

> **注意：前面的课程已经学过Nginx安装，所以这里上课的时候只需要学生安装，不需要另行讲解**

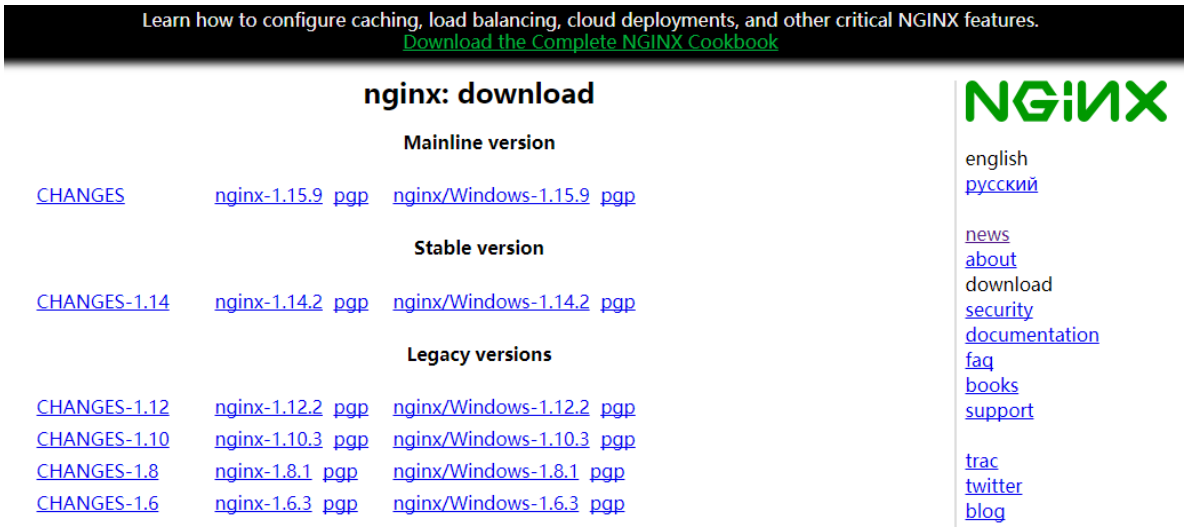
访问 nginx官方网站：http://www.nginx.org

Nginx版本类型

Mainline version： 主线版，即开发版

Stable version： 最新稳定版，生产环境上建议使用的版本

Legacy versions： 遗留的老版本的稳定版



1、官方 Yum 安装指导

Installation instructions

Before you install nginx for the first time on a new machine, you need to set up the nginx packages repository. Afterward, you can install and update nginx from the repository.

RHEL/CentOS

Install the prerequisites:

```
sudo yum install yum-utils
```

To set up the yum repository, create the file named `/etc/yum.repos.d/nginx.repo` with the following contents:

```
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key

[nginx-mainline]
name=nginx mainline repo
baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
gpgcheck=1
enabled=0
gpgkey=https://nginx.org/keys/nginx_signing.key
```

By default, the repository for stable nginx packages is used. If you would like to use mainline nginx packages, run the following command:

```
sudo yum-config-manager --enable nginx-mainline
```

To install nginx, run the following command:

```
sudo yum install nginx
```

When prompted to accept the GPG key, verify that the fingerprint matches `573B FD6B 3D8F BC64 1079 A6AB ABF5 BD82 7BD9 BF62`, and if so, accept it.

2、Yum 安装 nginx

```
[root@qfedu.com  ~]# yum -y install nginx
[root@qfedu.com  ~]# systemctl start nginx
[root@qfedu.com  ~]# systemctl enable nginx
```

注意 ： 查看防火墙状态， 需要关闭防火墙

```
[root@qfedu.com  ~]# getenforce
Disabled

[root@qfedu.com  ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor
pres
et: enabled)   Active: inactive (dead)
 Docs: man:firewalld(1)
```

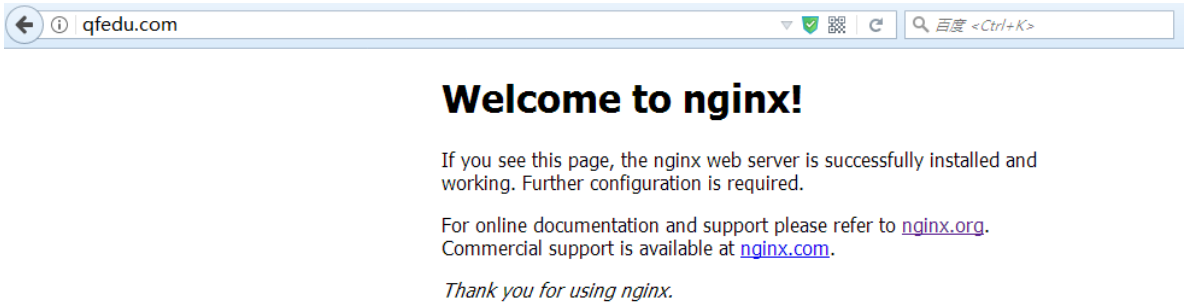
3、查看 nginx 安装版本

```
[root@qfedu.com  ~]# nginx -v
nginx version: nginx/1.14.2

[root@qfedu.com  ~]# nginx -V
nginx version: nginx/1.14.2
built by gcc 4.8.5 20150623 (Red Hat 4.8.5-11) (GCC)
built with OpenSSL 1.0.1e-fips 11 Feb 2013
TLS SNI support enabled
configure arguments:
 --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --modules-path
=/usr/lib64/nginx/modules --conf-path=/etc/nginx/nginx.conf --error-log-
path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --pid-
path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --http-client-body-temp-
path=/var/cache/nginx/client_temp --http-proxy-temp-
path=/var/cache/nginx/proxy_temp --http-fastcgi-temp-
path=/var/cache/nginx/fastcgi_temp --http-uwsgi-temp-
path=/var/cache/nginx/uwsgi_temp --http-scgi-temp-
path=/var/cache/nginx/scgi_temp --user=nginx --group=nginx --with-compat --with-
file-aio --with-threads --with-http_addition_module --with-
http_auth_request_module --with-http_dav_module --with-http_flv_module --with-
http_gunzip_module --with-http_gzip_static_module --with-http_mp4_module --with-
http_random_index_module --with-http_realip_module --with-
http_secure_link_module --with-http_slice_module --with-http_ssl_module --with-
http_stub_status_module --with-http_sub_module --with-http_v2_module --with-mail
--with-mail_ssl_module --with-stream --with-stream_realip_module --with-
stream_ssl_module --with-stream_ssl_preread_module --with-cc-opt='-O2 -g -pipe -
Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong --param=ssp-
buffer-size=4 -grecord-gcc-switches -m64 -mtune=generic -fPIC' --with-ld-opt='-
Wl,-z,relro -Wl,-z,now -pie'
```

4、浏览器访问验证

浏览器访问 www.qfedu.com (在访问机器上添加 hosts 解析)



2、Nginx 编译安装与配置使用

> **注意：前面的课程已经学过，所以这里上课的时候只需要学生安装，不需要另行讲解**

1、安装编译环境

```
[root@qfedu.com ~]# yum -y install gcc gcc-c++
```

2、安装pcre软件包（使nginx支持http rewrite模块）

```
[root@qfedu.com ~]# yum install -y pcre pcre-devel
```

3、安装 openssl-devel（使nginx支持ssl）

```
[root@qfedu.com ~]# yum install -y openssl openssl-devel
```

4、安装zlib

```
[root@qfedu.com ~]# yum install -y zlib zlib-devel
```

5、创建用户 nginx

```
[root@qfedu.com ~]# useradd nginx
[root@qfedu.com ~]# passwd nginx
```

6、编译安装 nginx

1、下载安装包

```
[root@qfedu.com ~]#wget http://120.52.51.15/nginx.org/download/nginx-
1.14.2.tar.gz
```

2、解压编译安装

```
[root@qfedu.com ~]#tar -vzxf nginx-1.14.2.tar.gz -C /usr/local
[root@qfedu.com ~]#cd nginx-1.14.2/
[root@qfedu.com nginx-1.14.2]# ./configure \
--group=nginx \
--user=nginx \
--prefix=/usr/local/nginx \
--sbin-path=/usr/sbin/nginx \
--conf-path=/etc/nginx/nginx.conf \
--error-log-path=/var/log/nginx/error.log \
--http-log-path=/var/log/nginx/access.log \
--http-client-body-temp-path=/tmp/nginx/client_body \
--http-proxy-temp-path=/tmp/nginx/proxy \
--http-fastcgi-temp-path=/tmp/nginx/fastcgi \
--pid-path=/var/run/nginx.pid \
--lock-path=/var/lock/nginx \
--with-http_stub_status_module \
--with-http_ssl_module \
--with-http_gzip_static_module \
--with-pcre
[root@qfedu.com nginx-1.11.3]# make &&make install
```

7、Nginx 编译参数

```
# 查看 nginx 安装的模块
[root@qfedu.com ~]# nginx -V

# 模块参数具体功能
--with-cc-opt='-g -O2 -fPIE -fstack-protector'    # 设置额外的参数将被添加到CFLAGS变
量。（FreeBSD或者ubuntu使用）
--param=ssp-buffer-size=4 -Wformat -Werror=format-security -D_FORTIFY_SOURCE=2'
--with-ld-opt='-Wl,-Bsymbolic-functions -fPIE -pie -Wl,-z,relro -Wl,-z,now'
```

```
--prefix=/usr/share/nginx                          # 指向安装目录
--conf-path=/etc/nginx/nginx.conf                  # 指定配置文件
--http-log-path=/var/log/nginx/access.log          # 指定访问日志
--error-log-path=/var/log/nginx/error.log          # 指定错误日志
--lock-path=/var/lock/nginx.lock                   # 指定lock文件
--pid-path=/run/nginx.pid                          # 指定pid文件

--http-client-body-temp-path=/var/lib/nginx/body     # 设定http客户端请求临时文件路径
--http-fastcgi-temp-path=/var/lib/nginx/fastcgi      # 设定http fastcgi临时文件路径
--http-proxy-temp-path=/var/lib/nginx/proxy          # 设定http代理临时文件路径
--http-scgi-temp-path=/var/lib/nginx/scgi            # 设定http scgi临时文件路径
--http-uwsgi-temp-path=/var/lib/nginx/uwsgi          # 设定http uwsgi临时文件路径

--with-debug                                         # 启用debug日志
--with-pcre-jit                                      # 编译PCRE包含“just-in-time
compilation”
--with-ipv6                                          # 启用ipv6支持
--with-http_ssl_module                               # 启用ssl支持
--with-http_stub_status_module                       # 获取nginx自上次启动以来的状态
--with-http_realip_module                  # 允许从请求标头更改客户端的IP地址值，默认为
关
--with-http_auth_request_module            # 实现基于一个子请求的结果的客户端授权。如果
该子请求返回的2xx响应代码，所述接入是允许的。如果它返回401或403中，访问被拒绝与相应的错误代
码。由子请求返回的任何其他响应代码被认为是一个错误。
--with-http_addition_module                # 作为一个输出过滤器，支持不完全缓冲，分部分
响应请求
--with-http_dav_module                     # 增加PUT,DELETE,MKCOL：创建集合,COPY和
MOVE方法 默认关闭，需编译开启
--with-http_geoip_module                   # 使用预编译的MaxMind数据库解析客户端IP地
址，得到变量值
--with-http_gunzip_module                  # 它为不支持“gzip”编码方法的客户端解压具
有“Content-Encoding: gzip”头的响应。
--with-http_gzip_static_module             # 在线实时压缩输出数据流
--with-http_image_filter_module            # 传输JPEG/GIF/PNG 图片的一个过滤器）（默
认为不启用。gd库要用到）
--with-http_spdy_module                    # SPDY可以缩短网页的加载时间
--with-http_sub_module                     # 允许用一些其他文本替换nginx响应中的一些文
本
--with-http_xslt_module                    # 过滤转换XML请求
--with-mail                                # 启用POP3/IMAP4/SMTP代理模块支持
--with-mail_ssl_module                     # 启用ngx_mail_ssl_module支持启用外部模块
支持
```

8、Nginx 配置文件

```
[root@qfedu.com ~]# vim /etc/nginx/nginx.conf
# 全局参数设置
worker_processes  1;          # 设置nginx启动进程的数量，一般设置成与逻辑cpu数量相同
error_log  logs/error.log;    # 指定错误日志
worker_rlimit_nofile 102400;  # 设置一个nginx进程能打开的最大文件数
pid        /var/run/nginx.pid;
events {                      # 事件配置
    worker_connections  10240; # 设置一个进程的最大并发连接数
    use epoll;                # 事件驱动类型
}
# http 服务相关设置
http {
```

```
    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  /var/log/nginx/access.log  main;     #设置访问日志的位置和格式
    sendfile            on;       # 用于开启文件高效传输模式，一般设置为on，若nginx是用来进行磁盘IO负载应用时，可以设置为off，降低系统负载
    tcp_nopush          on;       # 减少网络报文段数量，当有数据时，先别着急发送，确保数据包已经装满数据，避免了网络拥塞
    tcp_nodelay         on;       # 提高I/O性能，确保数据尽快发送，提高可数据传输效率

    gzip                on;       # 是否开启 gzip 压缩
    keepalive_timeout   65;       # 设置长连接的超时时间，请求完成之后还要保持连接多久，不是请求时间多久，目的是保持长连接，减少创建连接过程给系统带来的性能损
                耗，类似于线程池，数据库连接池
    types_hash_max_size 2048;  # 影响散列表的冲突率。types_hash_max_size 越大，就会消耗更多的内存，但散列key的冲突率会降低，检索速度就更快。
              types_hash_max_size越小，消耗的内存就越小，但散列key的冲突率可能上升
    include             /etc/nginx/mime.types;  # 关联mime类型，关联资源的媒体类型(不同的媒体类型的打开方式)
    default_type        application/octet-stream;  # 根据文件的后缀来匹配相应的MIME类型，并写入Response header，导致浏览器播放文件而不是下载
# 虚拟服务器的相关设置
    server {
        listen      80;         # 设置监听的端口
        server_name  localhost;         # 设置绑定的主机名、域名或ip地址
        charset koi8-r;         # 设置编码字符
        location / {
            root  /var/www/nginx;            # 设置服务器默认网站的根目录位置
            index  index.html index.htm;     # 设置默认打开的文档
            }
        error_page  500 502 503 504  /50x.html; # 设置错误信息返回页面
            location = /50x.html {
            root  html;         # 这里的绝对位置是/var/www/nginx/html
        }
    }
 }
```

9、检测 nginx 配置文件是否正确

```
[root@qfedu.com ~]# usr/local/nginx/sbin/nginx -t
```

10、启动 nginx 服务

```
[root@qfedu.com ~]# /usr/local/nginx/sbin/nginx
```

11、Nginx 命令控制

```
nginx -c /path/to/nginx.conf      # 以特定目录下的配置文件启动nginx:
nginx -s reload                       # 修改配置后重新加载生效
nginx -s reopen                                 # 重新打开日志文件
nginx -s stop                                       # 快速停止nginx
nginx -s quit                                   # 完整有序的停止nginx
nginx -t                                                    # 测试当前配置文件是否正确
nginx -t -c /path/to/nginx.conf   # 测试特定的nginx配置文件是否正确
```

12、实现nginx开机自启

1、添加启动脚本 vim /etc/init.d/nginx

```
#!/bin/sh
#
# nginx - this script starts and stops the nginx daemon
#
# chkconfig:  - 85 15
# description:  Nginx is an HTTP(S) server, HTTP(S) reverse \
#               proxy and IMAP/POP3 proxy server
# processname: nginx
# config:      /etc/nginx/nginx.conf
# config:      /etc/sysconfig/nginx
# pidfile:    /var/run/nginx.pid

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ "$NETWORKING" = "no" ] && exit 0

nginx="/usr/sbin/nginx"
prog=$(basename $nginx)

NGINX_CONF_FILE="/etc/nginx/nginx.conf"

[ -f /etc/sysconfig/nginx ] && . /etc/sysconfig/nginx

lockfile=/var/lock/subsys/nginx

make_dirs() {
  # make required directories
  user=`nginx -V 2>&1 | grep "configure arguments:" | sed 's/[^*]*--user=\([^
]*\).*/\1/g' -`
  options=`$nginx -V 2>&1 | grep 'configure arguments:'`
  for opt in $options; do
      if [ `echo $opt | grep '.*-temp-path'` ]; then
          value=`echo $opt | cut -d "=" -f 2`
          if [ ! -d "$value" ]; then
              # echo "creating" $value
              mkdir -p $value && chown -R $user $value
          fi
      fi
  done
}

start() {
    [ -x $nginx ] || exit 5
    [ -f $NGINX_CONF_FILE ] || exit 6
    make_dirs
    echo -n $"Starting $prog: "
    daemon $nginx -c $NGINX_CONF_FILE
    retval=$?
```

```
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    echo -n $"Stopping $prog: "
    killproc $prog -QUIT
    retval=$?
    echo
    [ $retval -eq 0 ] && rm -f $lockfile
    return $retval
}

restart() {
    configtest || return $?
    stop
    sleep 1
    start
}

reload() {
    configtest || return $?
    echo -n $"Reloading $prog: "
    killproc $nginx -HUP
    RETVAL=$?
    echo
}

force_reload() {
    restart
}

configtest() {
  $nginx -t -c $NGINX_CONF_FILE
}

rh_status() {
    status $prog
}

rh_status_q() {
    rh_status >/dev/null 2>&1
}

case "$1" in
    start)
        rh_status_q && exit 0
        $1
        ;;
    stop)
        rh_status_q || exit 0
        $1
        ;;
    restart|configtest)
        $1
        ;;
    reload)
```

```
        rh_status_q || exit 7
        $1
        ;;
    force-reload)
        force_reload
        ;;
    status)
        rh_status
        ;;
    condrestart|try-restart)
        rh_status_q || exit 0
            ;;
    *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-
restart|reload|force-reload|configtest}"
        exit 2
esac
```

2、添加权限

```
[root@qfedu.com ~]# chmod +x /etc/init.d/nginx
```

3、重载系统启动文件

```
[root@qfedu.com ~]# systemctl daemon-reload
```

4、设置开机自启

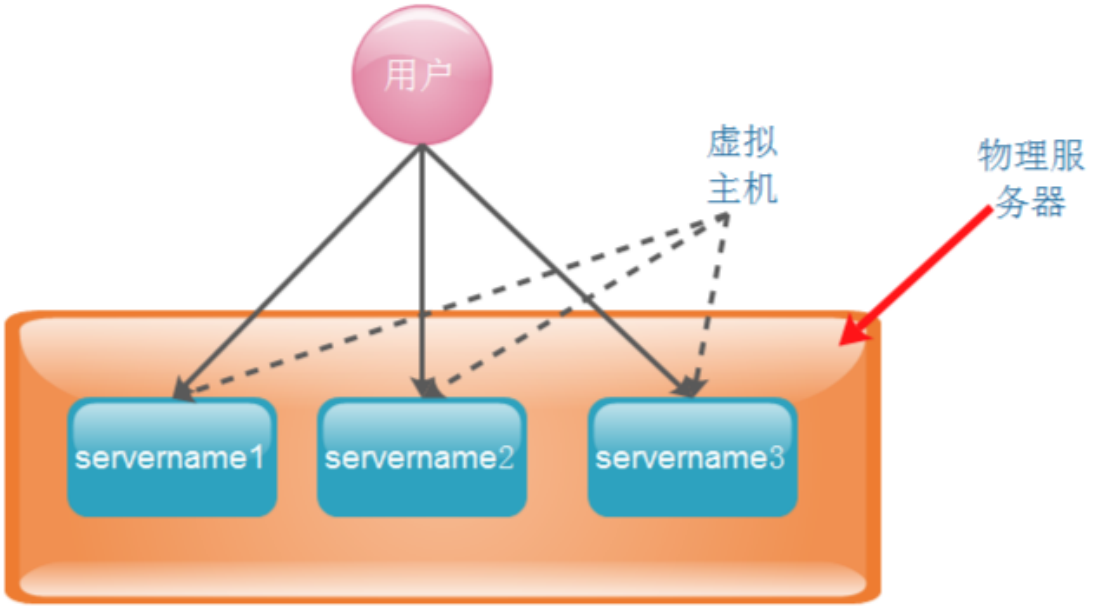
```
[root@qfedu.com ~]# systemct start nginx
```

六、Nginx 虚拟机主机

1、什么是虚拟主机？

虚拟主机是一种特殊的软硬件技术，它可以将网络上的每一台计算机分成多个虚拟主机，每个虚拟主机可以独立对外提供www服务，这样就可以实现一台主机对外提供多个web服务，每个虚拟主机之间是独立的，互不影响。

虚拟主机指的是在单一机器上运行多个网站（例如 company1.qfedu.com 和 company2.qfedu.com ）。虚拟主机可以是"基于 IP"的，即每个IP一个站点;或者是"基于域名"的，即每个域名一个站点。这些站点运行在同一物理服务器上，对用户不会有任何感知。Nginx也可以配置多种类型的虚拟主机: 基于IP的虚拟主机、基于端口的虚拟主机、基于域名的虚拟主机。下面将分别介绍这些虚拟主机的配置，及优缺点。



1、配置位置

此章节所有内容，既可以在七层代理层配置，又可以在业务逻辑层配置。但七层代理层配置的虚拟主机常配合nginx的ngx_http_proxy_module模块使用。 而业务逻辑层配置的虚拟主机常配合nginx的ngx_http_fastcgi_module模块使用。

2、nginx支持三种类型的虚拟主机配置

1、基于域名的虚拟主机（server_name来区分虚拟主机——应用：外部网站）
2、基于ip的虚拟主机，（一块主机绑定多个ip地址）
3、基于端口的虚拟主机（端口来区分虚拟主机——应用：公司内部网站，外部网站的管理后台）

2、 基于域名的虚拟主机

1、配置通过域名区分的虚拟机

```
# 标准 location
server {
        listen 80;
        server_name www.qfedu01.com;
        root         /usr/share/nginx/web1;
        access_log   /var/log/www.qfedu01.com.log main;
        error_log    /var/log/www.qfedu01.com.error.log;
        location /
                index index.html;
        }
}
# 多个 location
server {
        listen 80;
        server_name www.qfedu02.com;
        root         /usr/share/nginx/html;
        access_log   /var/log/www.qfedu01.com.log main;
        error_log    /var/log/www.qfedu01.com.error.log;
        location / {
                index index.html;
        }
        location /test {
```

```
                index index.html;
        }
}



# 模板配置
    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  _;
        root         /usr/share/nginx/html;
        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;
        location / {
        }
        error_page 404 /404.html;
            location = /40x.html {
        }
        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
        }
    }
```

2、 为 域名为 www.qfedu02.com 的虚拟机，创建 index 文件

```
[root@server html]#pwd
[root@server html]# /usr/share/nginx/html
[root@server html]# vim index.html
[root@server html]# cat index.html
<html>
<p>
this is my 1000phone02
</p>
</html>
[root@server html]#mkdir test
[root@server html]#cd test
[root@server html]#cat index.html
This is test.index.html
```

3、重新加载配置文件

```
# 如果编译安装的执行
[root@qfedu.com]# /usr/local/nginx/bin/nginx -s reload
# 如果 yum 安装的执行
[root@qfedu.com]# nginx -s reload
```

4、客户端配置路由映射

在 centos ： /etc/hosts　　windows:C:\Windows\System32\drivers\etc\hosts 文件中添加两行

```
[root@web1 ~]# curl 192.168.28.129 www.qfedu01.com
www.qfedu01.com
[root@web1 ~]# curl 192.168.28.129 www.qfedu02.com
<html>
<p>
this is my 1000phone02
</p>
</html>
```

5、 测试访问

浏览器输入：http://www.qfedu01.com/

浏览器输入：http://www.qfedu02.com/

6、补充：如果配置不能正常访问

问题描述： 配置完 nginx 两个虚拟机后，客户端能够访问原始的server ,新增加的 server 虚拟机不能够访问，报错如下页面

解决过程：

1、查看报错日志（找到错误日志）

```
  [root@qfedu.com]# cat logs/error.log
  2017/06/15 04:00:57 [error] 6702#0: *14 "/root/html/index.html" is forbidden
(13: Permission denied), client: 10.219.24.1, server: www.qfedu02.com, request:
"GET / HTTP/1.1", host: "www.qfedu02.com"
  [root@logs]# date
  Tue Mar 12 10:05:53 CST 2019
```

2、检查权限

```
  [root@qfedu.com ~]# ll
  drwxr-xr-x. 2 root root 4096 Jun 15 03:59 html
  [root@qfedu.com html]# ll
  total 8
  -rw-r--r--. 1 root root 537 Jun 15 03:59 50x.html
  -rw-r--r--. 1 root root 616 Jun 15 03:51 index.html
  说明：发现目录权限没有问题
```

3、检查nginx启动进程

```
 [root@qfedu.com]# ps anx|grep nginx
6546 ? Ss 0:00 nginx: master process ./sbin/nginx
6702 ? S 0:00 nginx: worker process
6726 pts/1 S+ 0:00 grep nginx
说明：发现nginx的work process是 nobody 的
```

4、修改 nginx.conf 文件

```
打开nginx.conf文件所在的目录，查看文件的属性 （root root）
[root@qfedu.com]# ll
drwxr-xr-x. 2 root root 4096 Jun 15 04:08 conf
在nginx.conf文件的第一行加上 user root root;
[root@qfedu.com]# cat conf/nginx.conf
user root root;
```

5、重新 reload nginx 进程

```
[root@qfedu.com]#nginx -s reload
注意：
nginx -s reload 命令加载修改后的配置文件,命令下达后发生如下事件
1. Nginx 的 master进程检查配置文件的正确性，若是错误则返回错误信息，nginx继续采用原配置文件进行工作（因为worker未受到影响）
2. Nginx 启动新的 worker 进程，采用新的配置文件
3. Nginx 将新的请求分配新的 worker 进程
4. Nginx 等待以前的 worker 进程的全部请求已经都返回后，关闭相关 worker 进程
5. 重复上面过程，直到全部旧的worker进程都被关闭掉
```

6、再次访问，成功！

2、 基于 ip 的虚拟主机

1、一块网卡绑定多个ip

```
[root@qfedu.com]# ifconfig eth0:1 192.168.95.200
[root@qfedu.com]# ifconfig
eth0 Link encap:Ethernet HWaddr 00:0C:29:79:F4:02
inet addr:192.168.95.134 Bcast:10.255.255.255 Mask:255.0.0.0
...
eth0:1 Link encap:Ethernet HWaddr 00:0C:29:79:F4:02
inet addr:192.168.95.200 Bcast:10.255.255.255 Mask:255.0.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

2、配置通过ip区分的虚拟机

```
server {
    listen 192.168.152.192:80;
    server_name www.qfedu01.com;
    root         /usr/share/nginx/html;
    access_log   /var/logs/www.qfedu01.com.log main;
    error_log    /var/logs/www.qfedu01.com.error.log;
    location / {
        root html;
        index index.html index.htm;
        }
}
```

```
server {
    listen 192.168.152.100:80;
    server_name www.qfedu02.com;
    root         /usr/share/nginx/lys;
    access_log   /var/logs/www.qfedu02.com.log main;
    error_log    /var/logs/www.qfedu02.com.error.log;
    location / {
        root html;
        index index.html index.htm;
        }
}
```

3、重新 reload nginx 进程

```
[root@qfedu.com]# nginx -s reload
```

4、 测试访问

浏览器输入：http://192.168.28.129

浏览器输入：http://192.168.28.133

5、补充：

```
-- 删除绑定的vip
[root@qfedu.com]# ifconfig eth0:1 192.168.95.100 down
```

3、 基于端口的虚拟主机

1、修改配置文件

```
server {
    listen 8000;
    server_name www.qfedu01.com;
    root         /usr/share/nginx/html;
    access_log   /var/logs/www.qfedu01.com.log main;
    error_log    /var/logs/www.qfedu01.com.error.log;
    location / {
        root html;
        index index.html index.htm;
        }
}

server {
    listen 8080;
    server_name www.qfedu01.com;
    root         /usr/share/nginx/lys;
    access_log   /var/logs/www.qfedu02.com.log main;
    error_log    /var/logs/www.qfedu02.com.error.log;
    location / {
        root html;
        index index.html index.htm;
        }
}
```

2、重新 reload nginx进程

```
[root@qfedu.com]# nginx -s reload
```

3、 测试访问

浏览器输入：http://www.qfedu01.com/

浏览器输入：http://www.qfedu02.com:8080