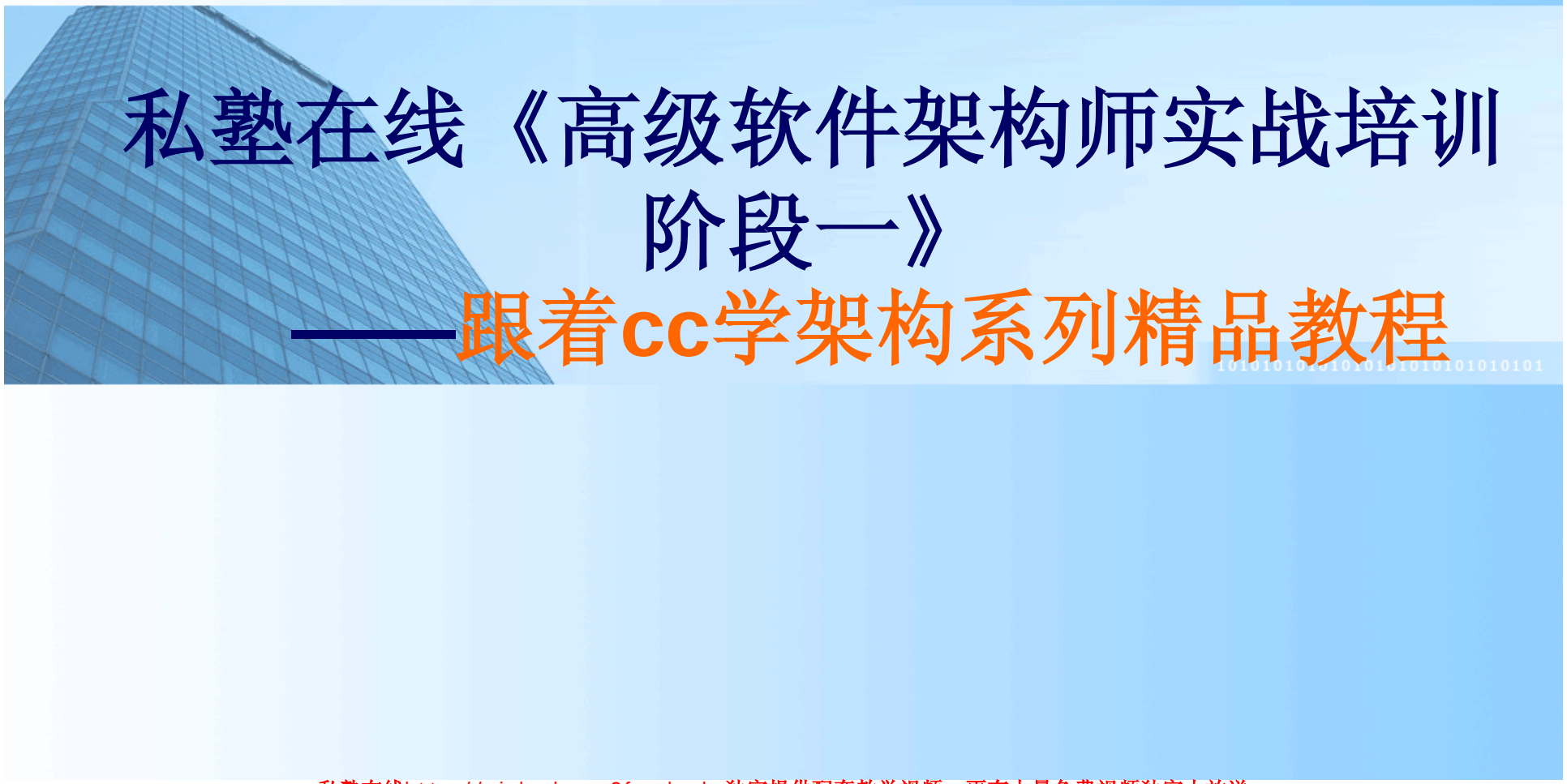

私塾在线《高级软件架构师实战培训阶段一》

——跟着cc学架构系列精品教程

本部分课程概览

- 根据实际的应用需要，学习要用到的Nginx的知识，以快速上手、理解并掌握Nginx
- 一：Nginx简介
  包括：Nginx是什么、能干什么、特点
- 二：Nginx安装和基本使用
  包括：源码安装、安装配置选项、基本使用
- 三：Nginx基本配置
  包括：结合示例的配置文件，熟悉Nginx的基本配置
- 四：学习核心模块、日志模块和事件模块的常用指令
- 五：学习Http模块的常用配置和指令
- 六：学习反向代理、动静分离、负载均衡、Geo和GeoIP模块
- 七：学习Rewrite模块和更多其它模块的功能
- 八：学习更多常见功能的片断配置
- 九：Nginx的配置优化建议

Nginx简介

- Nginx是什么

  Nginx是一款轻量级的Web服务器，也是一款轻量级的反向代理服务器。

- Nginx能干什么

  Nginx能干的事情很多，这里简要罗列一些：

  1：直接支持Rails和PHP的程序

  2：作为HTTP反向代理服务器

  3：作为负载均衡服务器

  4：作为邮件代理服务器

  5：帮助实现前端动静分离

  ……

- Nginx特点

  高稳定、高性能、资源占用少、功能丰富、模块化结构、支持热部署

Nginx安装

- 源码安装

演示环境：CentOS6.5

1：需要gcc，系统自带了，没有的话，需要先安装

2：需要pcre，安装的命令示例如下： yum install pcre*

3：需要zlib，安装的命令示例如下：yum install zlib zlib-devel

4：如果需要支持ssl的话，安装OpenSSL，安装的命令示例如下：

yum install openssl openssl-devel

5：上http://nginx.org/去下载源码包，然后进行解压安装，示例如下：

（1）先解压源码包，然后进入到这个包里面

（2）安装命令示例如下：

第一步：./configure --prefix=/usr/common/nginx --with-http_stub_status_module --with-http_ssl_module

如果提示确少啥，就加上相应的选项，比如缺少pcre的话，就加上 --with-pcre=/usr/common/temp/pcre-8.34 （当然我们这里是不缺东西的）

第二步：配置后就依次 make ， make install

常见的Nginx安装配置选项-1

- 编译参数可能会根据版本的不同进行变化，./configure --help查看编译参数列表，常见的选项如下：
- --prefix=<path> - 安装路径，如果没有指定，默认为/usr/local/nginx。
- --sbin-path=<path> - nginx可执行命令的文件，如果没有指定，默认为<prefix>/sbin/nginx。
- --conf-path=<path> - 在没有使用-c参数指定的情况下nginx.conf的默认位置，如果没有指定，默认为<prefix>/conf/nginx.conf。
- --pid-path=<path> - nginx.pid的路径，如果没有在nginx.conf中通过“pid”指令指定，默认为<prefix>/logs/nginx.pid。
- --lock-path=<path> - nginx.lock文件路径，如果没有指定，默认为<prefix>/logs/nginx.lock。
- --error-log-path=<path> - 当没有在nginx.conf中使用“error_log”指令指定时的错误日志位置，如果没有指定，默认为<prefix>/logs/error.log。
- --http-log-path=<path> - 当没有在nginx.conf中使用“access_log”指令指定时的访问日志位置，如果没有指定，默认为<prefix>/logs/access.log。
- --user=<user> - 当没有在nginx.conf中使用“user”指令指定时nginx运行的用户，如果没有指定，默认为“nobody”。
- --group=<group> - 当没有在nginx.conf中使用“user”指令指定时nginx运行的组，如果没有指定，默认为“nobody”。
- --builddir=DIR - 设置构建目录。
- --with-rtsig_module - 启用rtsig模块。

常见的Nginx安装配置选项-2

- --with-select_module –without-select_module - 如果在configure的时候没有发现kqueue, epoll, rtsig或/dev/poll其中之一，select模块始终为启用状态。
- --with-poll_module –without-poll_module - 如果在configure的时候没有发现kqueue, epoll, rtsig或/dev/poll其中之一，poll模块始终为启用状态。
- --with-http_ssl_module - 启用ngx_http_ssl_module，启用SSL支持并且能够处理HTTPS请求。需要OpenSSL，在Debian系统中，对应的包为libssl-dev。
- --with-http_realip_module - 启用ngx_http_realip_module
- --with-http_addition_module - 启用ngx_http_addition_module
- --with-http_sub_module - 启用ngx_http_sub_module
- --with-http_dav_module - 启用ngx_http_dav_module
- --with-http_flv_module - 启用ngx_http_flv_module
- --with-http_stub_status_module - 启用”server status”（服务状态）页
- --without-http_charset_module - 禁用ngx_http_charset_module
- --without-http_gzip_module - 禁用ngx_http_gzip_module，如果启用，需要zlib包。
- --without-http_ssi_module - 禁用ngx_http_ssi_module
- --without-http_userid_module - 禁用ngx_http_userid_module
- --without-http_access_module - 禁用ngx_http_access_module
- --without-http_auth_basic_module - 禁用ngx_http_auth_basic_module

常见的Nginx安装配置选项-3

- --without-http_autoindex_module - 禁用ngx_http_autoindex_module
- --without-http_geo_module - 禁用ngx_http_geo_module
- --without-http_map_module - 禁用ngx_http_map_module
- --without-http_referer_module - 禁用ngx_http_referer_module
- --without-http_rewrite_module - 禁用ngx_http_rewrite_module。如果启用，需要PCRE包。
- --without-http_proxy_module - 禁用ngx_http_proxy_module
- --without-http_fastcgi_module - 禁用ngx_http_fastcgi_module
- --without-http_memcached_module - 禁用ngx_http_memcached_module
- --without-http_limit_zone_module - 禁用ngx_http_limit_zone_module
- --without-http_empty_gif_module - 禁用ngx_http_empty_gif_module
- --without-http_browser_module - 禁用ngx_http_browser_module
- --without-http_upstream_ip_hash_module - 禁用ngx_http_upstream_ip_hash_module
- --with-http_perl_module - 启用ngx_http_perl_module
- --with-perl_modules_path=PATH - 为perl模块设置路径
- --with-perl=PATH - 为perl库设置路径
- --http-client-body-temp-path=PATH - 为http连接的请求实体临时文件设置路径，如果没有指定，默认为<prefix>/client_body_temp

常见的Nginx安装配置选项-4

- --http-proxy-temp-path=PATH - 为http代理临时文件设置路径，如果没有指定，默认为<prefix>/proxy_temp
- --http-fastcgi-temp-path=PATH - 为http fastcgi临时文件设置路径，如果没有指定，默认为<prefix>/fastcgi_temp
- --without-http - 禁用HTTP服务
- --with-mail - 启用IMAP4/POP3/SMTP代理模块
- --with-mail_ssl_module - 启用ngx_mail_ssl_module
- --with-cc=PATH - 设置C编译器路径
- --with-cpp=PATH - 设置C预处理器路径
- --with-cc-opt=OPTIONS - 变量CFLAGS中附加的参数，用于FreeBSD中的PCRE库，同样需要指定－with-cc-opt=”-I /usr/local/include”，如果我们使用select()函数则需要同时增加文件描述符数量，可以通过－with-cc-opt=”-D FD_SETSIZE=2048”指定。
- --with-ld-opt=OPTIONS - 通过连接器的附加参数，用于FreeBSD中的PCRE库，同样需要指定－with-ld-opt=”-L /usr/local/lib”。
- --with-cpu-opt=CPU - 指定编译的CPU，可用的值为：pentium，pentiumpro，pentium3，pentium4，athlon，opteron，amd64，sparc32，sparc64，ppc64
- --without-pcre - 禁用PCRE库文件，同时将禁用HTTP rewrite 模块，如果要在”location”指令中使用正则表达式，同样需要PCRE库。

常见的Nginx安装配置选项-5

- --with-pcre=DIR - 设置PCRE库源文件路径。
- --with-pcre-opt=OPTIONS - 在编译时为PCRE设置附加参数。
- --with-md5=DIR - 设置md5库源文件路径。
- --with-md5-opt=OPTIONS - 在编译时为md5设置附加参数。
- --with-md5-asm - 使用md5汇编源。
- --with-sha1=DIR - 设置sha1库源文件路径。
- --with-sha1-opt=OPTIONS - 在编译时为sha1设置附加参数。
- --with-sha1-asm - 使用sha1汇编源。
- --with-zlib=DIR - 设置zlib库源文件路径。
- --with-zlib-opt=OPTIONS - 在编译时为zlib设置附加参数。
- --with-zlib-asm=CPU - 为指定的CPU使用zlib汇编源进行优化，可用值为: pentium, pentiumpro。
- --with-openssl=DIR - 设置openssl库源文件路径。
- --with-openssl-opt=OPTIONS - 在编译时为openssl设置附加参数。
- --with-debug - 启用debug记录。
- --add-module=PATH - 增加一个在PATH中的第三方模块。

Nginx的基本运行

- 测试配置文件：

  安装路径下的/nginx/sbin/nginx -t

- 启动：

  安装路径下的/nginx/sbin/nginx

- 停止

  安装路径下的/nginx/sbin/nginx -s stop

  或者是： nginx -s quit

- 重启

  安装路径下的/nginx/sbin/nginx -s reload

- 查看进程

  ps -ef |grep nginx

- 安装过后，如果从外面访问不了，多半是被防火墙挡住了，可以关闭掉防火墙：

  /sbin/service iptables stop

Nginx的基本配置-1

- 默认启动Nginx时，使用的配置文件是： 安装路径/conf/nginx.conf 文件
  可以在启动nginx的时候，通过-c来指定要读取的配置文件
- 常见的配置文件有如下几个：
  nginx.conf：应用程序的基本配置文件
  mime.types：MIME类型关联的扩展文件
  fastcgi.conf：与fastcgi相关的配置
  proxy.conf：与proxy相关的配置
  sites.conf：配置Nginx提供的网站，包括虚拟主机
- Nginx的进程结构
  启动Nginx的时候，会启动一个Master进程，这个进程不处理任何客户端的请求，主要用来产生worker进程，一个worker进程用来处理一个request。
- Nginx模块分为：核心模块、事件模块、标准Http模块、可选Http模块、邮件模块、第三方模块和补丁等

Nginx的基本配置-2

- Nginx基本模块：所谓基本模块，指的是Nginx默认的功能模块，它们提供的指令，允许你使用定义Nginx基本功能的变量，在编译的时候不能被禁用，包括：
  核心模块：基本功能和指令，如进程管理和安全
  事件模块：在Nginx内配置网络使用的能力
  配置模块：提供包含机制
- 常见的核心模块指令，大部分都是放置在配置文件的顶部

  具体的指令，请参看nginx的官方文档，非常详细，参见：

  http://nginx.org/en/docs/ngx_core_module.html

  还有下面这个网站，也是非常不错的：

  http://www.howtocn.org/nginx:directiveindex

- 常见的events模块指令，大部分都是放置在配置文件的顶部

  具体的指令，在上面那个文档里面，命令的context为events的就是events模块指令，只能在events里面使用

Nginx常用的核心模块指令

- 核心模块指令，重点看看：error_log、include、pid、user、worker_cpu_affinity、worker_processes
- error_log

日志有6个级别：debug|info|notice|warn|error|crit

Nginx支持将不同的虚拟主机的日志记录在不同的地方，如下示例：

```
http{
    error_log logs/http_error.log error;
    server{
        server_name one;
        access_log logs/one_access.log;
        error_log logs/one_error.log error;
    }
    server{
        server_name two;
        access_log logs/two_access.log;
        error_log logs/two_error.log error;
    }
}
```

注意：error_log off不是禁用日志，而是创建一个名为off的日志，要禁用日志，可以这么写：error_log /dev/null crit;

Nginx常用的日志模块、事件模块指令

- 日志模块指令，几个都看看
- 事件模块指令，重点看看：use和worker_connections

Nginx的HTTP基本配置

- Nginx的HTTP配置主要包括三个区块，结构如下：

```
http {  //这个是协议级别
    include       mime.types;
    default_type  application/octet-stream;
    keepalive_timeout  65;
          gzip  on;
    server {  //这个是服务器级别
        listen       80;
        server_name  localhost;

        location / {  //这个是请求级别
            root   html;
            index  index.html index.htm;
        }
    }
}
```

- Nginx的HTTP核心模块，包括大量的指令和变量，大都很重要，具体参见：
  http://nginx.org/en/docs/http/ngx_http_core_module.html

Location区段-1

- Location区段，通过指定模式来与客户端请求的URI相匹配，基本语法如下：

```
location [=|~|~*|^~|@] pattern{……}
```

1：没有修饰符 表示：必须以指定模式开始，如：

```
server {
        server_name  sishuok.com;
        location  /abc {
            ……
        }
    }
```

那么，如下是对的：

http://sishuok.com/abc

http://sishuok.com/abc?p1=11&p2=22

http://sishuok.com/abc/

http://sishuok.com/abcde

Location区段-2

2：= 表示：必须与指定的模式精确匹配，如：

```
 server {
        server_name  sishuok.com;
        location = /abc {
            ......
        }
    }
```

那么，如下是对的：

http://sishuok.com/abc

http://sishuok.com/abc?p1=11&p2=22

如下是错的：

http://sishuok.com/abc/

http://sishuok.com/abcde

Location区段-3

3：~ 表示：指定的正则表达式要区分大小写，如：

```
 server {
        server_name  sishuok.com;
        location ~ ^/abc$ {
            ......
        }
    }
```

那么，如下是对的：

http://sishuok.com/abc

http://sishuok.com/abc?p1=11&p2=22

如下是错的：

http://sishuok.com/ABC

http://sishuok.com/abc/

http://sishuok.com/abcde

Location区段-4

4：~* 表示：指定的正则表达式不区分大小写，如：

```
 server {
        server_name  sishuok.com;
        location ~* ^/abc$ {
            ……
        }
    }
```

那么，如下是对的：

http://sishuok.com/abc

http://sishuok.com/ABC

http://sishuok.com/abc?p1=11&p2=22

如下是错的：

http://sishuok.com/abc/

http://sishuok.com/abcde

Location区段-5

5：^~ 类似于无修饰符的行为，也是以指定模式开始，不同的是，如果模式匹配，那么就停止搜索其他模式了。

6：@ ：定义命名location区段，这些区段客户段不能访问，只可以由内部产生的请求来访问，如try_files或error_page等

- 查找顺序和优先级
  1：带有“=“的精确匹配优先
  2：没有修饰符的精确匹配
  3：正则表达式按照他们在配置文件中定义的顺序
  4：带有“^~”修饰符的，开头匹配
  5：带有“~” 或“~*” 修饰符的，如果正则表达式与URI匹配
  6：没有修饰符的，如果指定字符串与URI开头匹配

Location区段匹配示例

```
location  = / {
  # 只匹配 / 的查询.
  [ configuration A ]
}
location  / {
  # 匹配任何以 / 开始的查询，但是正则表达式与一些较长的字符串将被首先匹配。
  [ configuration B ]
}
location ^~ /images/ {
  # 匹配任何以 /images/ 开始的查询并且停止搜索，不检查正则表达式。
  [ configuration C ]
}
location ~* \.(gif|jpg|jpeg)$ {
  # 匹配任何以gif, jpg, or jpeg结尾的文件，但是所有 /images/ 目录的请求将在Configuration C中处
    理。
  [ configuration D ]
}
```

各请求的处理如下例：

- / → configuration A
- /documents/document.html → configuration B
- /images/1.gif → configuration C
- /documents/1.jpg → configuration D

Http反向代理

- Nginx通常被用作后端服务器的反向代理，这样就可以很方便的实现动静分离，以及负载均衡，从而大大提高服务器的处理能力。

- Http Proxy模块，功能很多，最常用的是proxy_pass，最好还是都看看。

- 如果要使用proxy_cache的话，需要集成第三方的ngx_cache_purge模块，用来清除指定的URL缓存。这个集成需要在安装nginx的时候去做，形如：
  ./configure --add-module=../ngx_cache_purge-1.0 ……

动静分离

- Nginx实现动静分离，其实就是在反向代理的时候，如果是静态资源，那么就直接从Nginx发布的路径去读取，而不需要从后台服务器获取了。

- 但是要注意：这种情况下需要保证后端跟前端的程序保持一致，可以使用Rsync做服务端自动同步或者使用NFS、MFS分布式共享存储。

负载均衡

- Nginx通过upstream模块来实现简单的负载均衡
- 在upstream块内，定义一个服务器列表，默认的方式是轮询，如果要确定同一个访问者发出的请求总是由同一个后端服务器来处理，可以设置ip_hash，如：

```
upstream cctest1.com {
    ip_hash
    server 127.0.0.1:9080 weight=5;
    server 127.0.0.1:8080 weight=5;
    server 127.0.0.1:1111;
}
```

请注意：这个方法本质还是轮询，而且由于客户端的ip可能是不断变化的，比如动态ip，代理，翻墙等等，因此ip_hash并不能完全保证同一个客户端总是由同一个服务器来处理。

- 更多指令和配置，请参考Nginx的http负载均衡模块

Geo和GeoIP模块

- 这两个模块主要用于做全局的负载均衡，可以根据不同的客户端ip来访问不同的服务器，示例如下：

```
http{
    geo $geo{
            default default;
            202.103.10.1/24 A;
            179.9.0.3/24 B;                    }
    upstream default.server{
            server 192.168.0.100;              }
    upstream A.server{
            server 192.168.0.101;              }
    upstream B.server{
            server 192.168.0.102;              }
    server{
            listen 80;
            location / {
                        proxy_pass http://$geo.server$request_uri;
            }
    }
}
```

Rewrite模块配置-1

- Rewrite模块：用来执行URL重定向。这个机制有利于去掉恶意访问的url，也有利于搜索引擎优化(SEO)。
- Nginx使用的语法源于Perl兼容正则表达式（PCRE）库，基本语法如下：

  ^ ：必须以^后的实体开头

  $ ：必须以$前的实体结尾

  . ：匹配任意字符

  [ ] ：匹配指定字符集内的任意字符

  [^] ：匹配任何不包括在指定字符集内的任意字符串

  | ：匹配 | 之前或之后的实体

  () ：分组，组成一组用于匹配的实体，通常会有|来协助
- 捕获子表达式，可以捕获放在（）之间的任何文本，比如：

  ^(.*)(hello|sir)$ 字符串为“hi sir” 捕获的结果： $1=hi $2=sir

  这些被捕获的数据，在后面就可以当变量一样使用了

Rewrite模块配置-2

- 内部请求

  外部请求是客户端的url，内部请求是Nginx通过特殊的指令触发。

  比如：error_page、index、rewrite、try_files、include等等

- 内部请求分成两种类型

  1：内部重定向：URI被改变，可能会匹配到其他的Location

  2：子请求：比如使用Addition模块，指令add_after_body允许你在原始的URI之后指定一个URI，会把该URI被处理后的结果，插入到原始的URI的body中。

- 内部重定向示例：

```
server {
        server_name  sishuok.com;
        location  /abc/ {
            rewrite ^/abc/(.*)$  /bcd/$1
        }
        location /bcd/{
            internal;
            root pages;
        }
    }
```

Rewrite模块配置-3

- 条件结构的基本语法：

  1：没有操作符：指定的字符串或者变量不为空，也不为0开始的字符串，取true

  2：= ， != ，例：if($request_method = POST){…}

  3：~，~*，!~，!~* ，例：if($uri ~* “\.jsp$”){…}

  4：-f,!-f :用来测试指定文件是否存在，例：if(-f $request_filename){…}

  5：-d,!-d ：用来测试指定目录是否存在

  6：-e,!-e：用来测试指定文件、目录或者符号链接是否存在

  7：-x,!-x：用来测试指定文件是否存在和是否可以执行

  8：break：跳出if块

  9：return：终止处理，并返回一个指定的http状态码

  10：set：初始化或者重定义一个变量

其它模块-1

- Http Index模块，都看看
- Http Referer模块，都看看，可用于防盗链
- Http Limit Zone模块，都看看，可用于会话的连接数控制，如限制每个IP的并发连接数等
- Http Access模块，用于简单的访问控制，都看看
- Http Charset模块，重点看看：charset
- Gzip模块，可以都看看

- Http Browser模块，用于按照请求头中的“User-agent”来创建一些变量，好为不同的浏览器创建不同的内容，暂时了解即可

- Memcached模块，这是把Nginx当作Memcached的客户端，用来连接Memcached的模块。暂时不用看
- Http Addition模块，可以在当前location内容之前或后添加内容，暂时不用看

其它模块-2

- Http Empty Gif模块，这个模块在内存中保存一个能够很快传递的1×1透明GIF，暂时不用看
- Http Auth Basic模块，基于Http Basic认证的方式来保护虚拟主机或目录，暂时不用看
- Http AutoIndex模块，用于提供自动目录列表，该模块只有在找不到默认的index文件的时候才启用，暂时不用看
- Http Fcgi模块，用于与FastCGI进程交互，暂时不用看
- FLV Stream模块，支持当Http下载方式播放Flv时，可以支持进度条拖放，暂时不用看。
- SSL模块，暂时不用看
- 邮件模块，暂时不用看
- 还有很多的模块，这里就不再一一介绍了

常见功能的配置片断-1

- server配置为监听ip和端口

```
server{
        listen 127.0.0.1:9080;
        server_name 127.0.0.1;
}
```

- server配置为监听域名和端口

```
server{
        listen  80;
        server_name  www.sishuok.com sishuok.com *.sishuok.com;
}
```

- 向后台服务器传递客户端的真实ip

```
location ~ \.(jsp|action|mvc)$ {
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_pass http://sishuok.com;
}
```

常见功能的配置片断-2

- 在负载均衡里面，实现后端服务器故障转移的配置

```
location ~ \.(jsp|action|mvc)$ {
        proxy_next_upstream http_502 http_504 timeout;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_pass http://server_pool;
}
```

- 简单的防盗链

```
location / {
        ……
        valid_referers blocked sishuok.com *.sishuok.com;
        if($invalid_referer){
                rewrite ^/  http://sishuok.com;
        }
}
```

- 简单的限制下载速度

```
location / {
        limit_rate 256K;     }
```

常见功能的配置片断-3

- 使用proxy_cache的配置

```
http{#下面这两个path指定的路径必须在同一个分区
        proxy_temp_path   /cachetemp/proxy_temp_path;
        #设置名称为mycache，内存缓存100m，自动清除1天未使用的内容，硬盘缓存空间1g
        proxy_cache_path /cachetemp/proxy_cache_path levels=1:2 keys_zone=mycache:100m
inactive=1d max_size=1g;
        server{
                location ~ .*\.(gif|jpg|html|js|css)$ {
                        proxy_cache mycache; #使用名称为mycache的缓存
                        #对不同的Http状态码设置不同的缓存时间
                        proxy_cache_valid 200 304 24h;
                        proxy_cache_valid 301 302 10m;
                        proxy_cache_valid any 1m;
                        #设置缓存的key值
                        proxy_cache_key $host$uri$is_args$args;
                }
        }
}
```

Nginx的配置优化-1

- 如果没有足够的实力和必要去自己改写Nginx，那么Nginx的优化主要就是：优化Nginx的配置，做到合理高效的使用
- 优化的方向和目标，无外乎：
  1：尽量提高单台机器处理效率
  2：尽量降低单台机器的负载
  3：尽量降低磁盘I/O
  4：尽量降低网络I/O
  5：尽量减少内存使用
  6：尽量高效利用CPU
- 生产环境下，应该使Nginx模块最小化，就是用到哪几个就开哪几个，这个需要在编译安装Nginx的时候做。

Nginx的配置优化-2

- 用户和组，生产环境下，最好是专为Nginx创建用户和组，并单独设置权限，这样会更安全。例如： user nginx nginx
- worker_processes :通常配置成cpu的总核数，或者其2倍，性能会更好。这可以减少进程间切换带来的消耗。
- 还可以同时使用worker_cpu_affinity来绑定cpu，使得每个worker进程独享一个cpu，实现完全的并发，性能更好，不过这个只对linux系统有效。
- events里面的事件模型，Linux推荐使用epoll模型，FreeBSD推荐采用kqueue
- worker_rlimit_nofile：描述一个nginx进程打开的最多的文件数目。配置成跟linux内核下文件打开数一致就可以了。可以通过ulimit -n 来查看，新装的系统默认是1024，CentOS中可以如下方式进行修改：

  在/etc/security/limits.conf最后增加：

```
* soft nofile 65535
* hard nofile 65535
* soft nproc 65535
* hard nproc 65535
```

Nginx的配置优化-3

- worker_connections：每个进程允许的最多连接数，默认是1024，可以设置大一些。
  理论上并发总数是worker_processes和worker_connections的乘积，
  worker_connections值的设置跟物理内存大小有关，因为系统可以打开的最大文件数和内存大小成正比，一般1GB内存的机器上可以打开的文件数大约是10万左右，所以，worker_connections 的值需根据 worker_processes 进程数目和系统可以打开的最大文件总数进行适当地进行设置。
- keepalive_timeout：设置到65左右就可以
- client_header_buffer_size：设置请求的缓存，设置为4k，通常为系统分页大小的整数倍，可以通过getconf PAGESIZE 来查看系统分页大小。
- 对打开文件设置缓存
  open_file_cache max=建议设置成和每个进程打开的最大文件数一致 inactive=60s;
  open_file_cache_valid 90s;
  open_file_cache_min_uses 2;
  open_file_cache_errors on;
- 尽量开启Gzip压缩，gzip_comp_level通常设置成3-5，高了浪费CPU
- Error日志优化：运行期间设置为crit，可以减少I/O

Nginx的配置优化-4

- access日志优化：如果使用了其他统计软件，可以关闭日志，来减少磁盘写，或者写入内存文件，提高I/O效率。
- sendfile指令指定 nginx 是否调用 sendfile 函数（zero copy 方式）来输出文件，通常应设置成on，如果是下载等应用磁盘IO重负载应用，可设置为 off
- Buffers size优化：如果buffer size太小就会到导致nginx使用临时文件存储response，这会引起磁盘读写IO，流量越大问题越明显。

  client_body_buffer_size 处理客户端请求体buffer大小。用来处理POST提交数据，上传文件等。client_body_buffer_size 需要足够大以容纳需要上传的POST数据。同理还有后端的buffer数据。
- worker_priority进程优先级设置：Linux系统中，优先级高的进程会占用更多的系统资源，这里配置的是进程的静态优先级，取值范围-20到+19，-20级别最高。因此可以把这个值设置小一点，但不建议比内核进程的值低（通常为-5）
- 合理设置静态资源的浏览器缓存时间，尽量用浏览器缓存
- 负载均衡锁accept_mutex，建议开启，默认就是开启的
- 如果使用SSL的话，而且服务器上有SSL硬件加速设备的话，请开启硬件加速。