**1、安装Nginx**  
在安装Nginx之前，需确保系统已经安装了gcc、 openssl-devel、 pcre-devel和zlib-devel软件库。  
下面是Nginx安装过程：

wget http://nginx.org/download/nginx-1.0.14.tar.gz

tar zxvf nginx-1.0.14.tar.gz

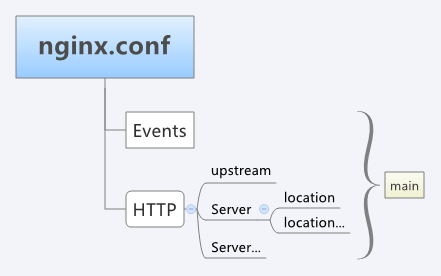
./configure --with-http\_stub\_status\_module --prefix=/opt/nginx

cd nginx-1.0.14

make

make install

其中， –with-http\_stub\_status\_module 可以用来启用 Nginx 的 NginxStatus 功能，以监控 Nginx 的运行状态。  
想要了解更多的模块的情况可以通过 ./configure –help 选项查看。

**2、Nginx的配置文件结构**  
Nginx的配置文件nginx.conf位于其安装目录的conf目录下。  
nginx.conf由多个块组成，最外面的块是main，main包含Events和HTTP，HTTP包含upstream和多个Server，Server又包含多个location：  


main（全局设置）、server（主机设置）、upstream（负载均衡服务器设置）和 location（URL匹配特定位置的设置）。

* main块设置的指令将影响其他所有设置；
* server块的指令主要用于指定主机和端口；
* upstream指令主要用于负载均衡，设置一系列的后端服务器；
* location块用于匹配网页位置。

这四者之间的关系式：server继承main，location继承server，upstream既不会继承其他设置也不会被继承。  
在这四个部分当中，每个部分都包含若干指令，这些指令主要包含Nginx的主模块指令、事件模块指令、HTTP核心模块指令，同时每个部分还可以使用其他HTTP模块指令，例如Http SSL模块、HttpGzip Static模块和Http Addition模块等。

**2.1 Nginx的全局配置**

代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | user nobody nobody;  worker\_processes 2;  error\_log logs/error.log notice;  pid logs/nginx.pid;  worker\_rlimit\_nofile 65535;    events{  use epoll;  worker\_connections 65536;  } |

每个配置选项的含义解释如下：

* user是个主模块指令，指定Nginx Worker进程运行用户以及用户组，默认由nobody账号运行。
* worker\_processes是个主模块指令，指定了Nginx要开启的进程数。每个Nginx进程平均耗费10M~12M内存。建议指定和CPU的数量一致即可。
* error\_log是个主模块指令，用来定义全局错误日志文件。日志输出级别有debug、info、notice、warn、error、crit可供选择，其中，debug输出日志最为最详细，而crit输出日志最少。
* pid是个主模块指令，用来指定进程pid的存储文件位置。
* worker\_rlimit\_nofile用于绑定worker进程和CPU， Linux内核2.4以上可用。

events事件指令是设定Nginx的工作模式及连接数上限：  
use是个事件模块指令，用来指定Nginx的工作模式。Nginx支持的工作模式有select、poll、kqueue、epoll、rtsig和/dev/poll。其中select和poll都是标准的工作模式，kqueue和epoll是高效的工作模式，不同的是epoll用在Linux平台上，而kqueue用在BSD系统中。对于Linux系统，**epoll工作模式是首选。**  
worker\_connections也是个事件模块指令，用于定义Nginx每个进程的最大连接数，默认是1024。最大客户端连接数由worker\_processes和worker\_connections决定，即Max\_client=worker\_processes\*worker\_connections。  
在作为反向代理时，max\_clients变为：max\_clients = worker\_processes \* worker\_connections/4。  
进程的最大连接数受Linux系统进程的最大打开文件数限制，在执行操作系统命令“ulimit -n 65536”后worker\_connections的设置才能生效

**2.2 HTTP服务器配置**  
Nginx对HTTP服务器相关属性的配置代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | http{  http{  include conf/mime.types;  default\_type application/octet-stream;  log\_format main '$remote\_addr - $remote\_user [$time\_local] '  '"$request" $status $bytes\_sent '  '"$http\_referer" "$http\_user\_agent" '  '"$gzip\_ratio"';  log\_format download '$remote\_addr - $remote\_user [$time\_local] '  '"$request" $status $bytes\_sent '  '"$http\_referer" "$http\_user\_agent" '  '"$http\_range" "$sent\_http\_content\_range"';  client\_max\_body\_size 20m;  client\_header\_buffer\_size 32K;  large\_client\_header\_buffers 4 32k;  Sendfile on;  tcp\_nopush on;  tcp\_nodelay on;  keepalive\_timeout 60;  client\_header\_timeout 10;  client\_body\_timeout 10;  send\_timeout 10; |

下面详细介绍下这段代码中每个配置选项的含义。  
include是个主模块指令，实现对配置文件所包含的文件的设定，可以减少主配置文件的复杂度。类似于Apache中的include方法。  
default\_type属于HTTP核心模块指令，这里设定默认类型为二进制流，也就是当文件类型未定义时使用这种方式，例如在没有配置PHP环境时，Nginx是不予解析的，此时，用浏览器访问PHP文件就会出现下载窗口。

下面的代码实现对日志格式的设定：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | log\_format main '$remote\_addr - $remote\_user [$time\_local] '  '"$request" $status $bytes\_sent '  '"$http\_referer" "$http\_user\_agent" '  '"$gzip\_ratio"';  log\_format download '$remote\_addr - $remote\_user [$time\_local] '  '"$request" $status $bytes\_sent '  '"$http\_referer" "$http\_user\_agent" '  '"$http\_range" "$sent\_http\_content\_range"'; |

log\_format是Nginx的HttpLog模块指令，用于指定Nginx日志的输出格式。main为此日志输出格式的名称，可以在下面的access\_log指令中引用。

* client\_max\_body\_size用来设置允许客户端请求的最大的单个文件字节数；
* client\_header\_buffer\_size用于指定来自客户端请求头的headerbuffer大小。对于大多数请求，1K的缓冲区大小已经足够，如果自定义了消息头或有更大的Cookie，可以增加缓冲区大小。这里设置为32K；
* large\_client\_header\_buffers用来指定客户端请求中较大的消息头的缓存最大数量和大小， “4”为个数，“128K”为大小，最大缓存量为4个128K；
* sendfile参数用于开启高效文件传输模式。将tcp\_nopush和tcp\_nodelay两个指令设置为on用于防止网络阻塞；
* keepalive\_timeout设置客户端连接保持活动的超时时间。在超过这个时间之后，服务器会关闭该连接；
* client\_header\_timeout设置客户端请求头读取超时时间。如果超过这个时间，客户端还没有发送任何数据，Nginx将返回“Request time out（408）”错误；
* client\_body\_timeout设置客户端请求主体读取超时时间。如果超过这个时间，客户端还没有发送任何数据，Nginx将返回“Request time out（408）”错误，默认值是60；
* send\_timeout指定响应客户端的超时时间。这个超时仅限于两个连接活动之间的时间，如果超过这个时间，客户端没有任何活动，Nginx将会关闭连接。

**2.3 HttpGzip模块配置**  
下面配置Nginx的HttpGzip模块。这个模块支持在线实时压缩输出数据流。  
看是否安装了HttpGzip模块：

[root@vps ~]# /opt/nginx/sbin/nginx -V

nginx version: nginx/1.0.14

built by gcc 4.4.6 20110731 (Red Hat 4.4.6-3) (GCC)

configure arguments: --with-http\_stub\_status\_module --with-http\_gzip\_static\_module --prefix=/opt/nginx

通过/opt/nginx/sbin/nginx -V命令可以查看安装Nginx时的编译选项，由输出可知，我们已经安装了HttpGzip模块。

下面是HttpGzip模块在Nginx配置中的相关属性设置：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | gzip on;  gzip\_min\_length 1k;  gzip\_buffers 4 16k;  gzip\_http\_version 1.1;  gzip\_comp\_level 2;  gzip\_types text/plain application/x-javascript text/css application/xml;  gzip\_vary on; |

* gzip用于设置开启或者关闭gzip模块，“gzip on”表示开启GZIP压缩，实时压缩输出数据流；
* gzip\_min\_length设置允许压缩的页面最小字节数，页面字节数从header头的Content-Length中获取。默认值是0，不管页面多大都进行压缩。建议设置成大于1K的字节数，小于1K可能会越压越大；
* gzip\_buffers表示申请4个单位为16K的内存作为压缩结果流缓存，默认值是申请与原始数据大小相同的内存空间来存储gzip压缩结果；
* gzip\_http\_version用于设置识别HTTP协议版本，默认是1.1，目前大部分浏览器已经支持GZIP解压，使用默认即可；
* gzip\_comp\_level用来指定GZIP压缩比，1 压缩比最小，处理速度最快；9 压缩比最大，传输速度快，但处理最慢，也比较消耗cpu资源；
* gzip\_types用来指定压缩的类型，无论是否指定，“text/html”类型总是会被压缩的；
* gzip\_vary选项可以让前端的缓存服务器缓存经过GZIP压缩的页面，例如用Squid缓存经过Nginx压缩的数据。

**2.4 负载均衡配置**  
下面设定负载均衡的服务器列表：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | upstream cszhi.com{  ip\_hash;  server 192.168.8.11:80;  server 192.168.8.12:80 down;  server 192.168.8.13:8009 max\_fails=3 fail\_timeout=20s;  server 192.168.8.146:8080;  } |

upstream是Nginx的HTTP Upstream模块，这个模块通过一个简单的调度算法来实现客户端IP到后端服务器的负载均衡。  
在上面的设定中，通过upstream指令指定了一个负载均衡器的名称cszhi.com。这个名称可以任意指定，在后面需要的地方直接调用即可。

Nginx的负载均衡模块目前支持4种调度算法，下面进行分别介绍，其中后两项属于第三方的调度方法。

* 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，如果后端某台服务器宕机，故障系统被自动剔除，使用户访问不受影响；
* Weight：指定轮询权值，Weight值越大，分配到的访问机率越高，主要用于后端每个服务器性能不均的情况下；
* ip\_hash：每个请求按访问IP的hash结果分配，这样来自同一个IP的访客固定访问一个后端服务器，有效解决了动态网页存在的session共享问题；
* fair：比上面两个更加智能的负载均衡算法。此种算法可以依据页面大小和加载时间长短智能地进行负载均衡，也就是根据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。Nginx本身是不支持fair的，如果需要使用这种调度算法，必须下载Nginx的upstream\_fair模块；
* url\_hash：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，可以进一步提高后端缓存服务器的效率。Nginx本身是不支持url\_hash的，如果需要使用这种调度算法，必须安装Nginx 的hash软件包。

在HTTP Upstream模块中，可以通过server指令指定后端服务器的IP地址和端口，同时还可以设定每个后端服务器在负载均衡调度中的状态。常用的状态有：

* down：表示当前的server暂时不参与负载均衡；
* backup：预留的备份机器。当其他所有的非backup机器出现故障或者忙的时候，才会请求backup机器，因此这台机器的压力最轻；
* 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：在经历了max\_fails次失败后，暂停服务的时间。max\_fails可以和fail\_timeout一起使用。

注意，当负载调度算法为ip\_hash时，后端服务器在负载均衡调度中的状态不能是weight和backup。

**2.5 server虚拟主机配置**  
下面介绍对虚拟主机的配置。  
建议将对虚拟主机进行配置的内容写进另外一个文件，然后通过include指令包含进来，这样更便于维护和管理。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | server{  listen 80;  server\_name 192.168.8.18 cszhi.com;  index index.html index.htm index.php;  root /wwwroot/www.cszhi.com  charset gb2312;  access\_log logs/www.ixdba.net.access.log main; |

server标志定义虚拟主机开始，listen用于指定虚拟主机的服务端口，server\_name用来指定IP地址或者域名，多个域名之间用空格分 开。index用于设定访问的默认首页地址，root指令用于指定虚拟主机的网页根目录，这个目录可以是相对路径，也可以是绝对路径。Charset用于 设置网页的默认编码格式。access\_log用来指定此虚拟主机的访问日志存放路径，最后的main用于指定访问日志的输出格式。

**2.6 location URL匹配配置**  
URL地址匹配是进行Nginx配置中最灵活的部分。 location支持正则表达式匹配，也支持条件判断匹配，用户可以通过location指令实现Nginx对动、静态网页进行过滤处理。使用location URL匹配配置还可以实现反向代理，用于实现PHP动态解析或者负载负载均衡。  
以下这段设置是通过location指令来对网页URL进行分析处理，所有扩展名以.gif、.jpg、.jpeg、.png、.bmp、.swf结尾的静态文件都交给nginx处理，而expires用来指定静态文件的过期时间，这里是30天。

|  |  |
| --- | --- |
| 1  2  3  4 | location ~ .\*\.(gif|jpg|jpeg|png|bmp|swf)$ {  root /wwwroot/www.cszhi.com;  expires 30d;  } |

以下这段设置是将upload和html下的所有文件都交给nginx来处理，当然，upload和html目录包含在/web/wwwroot/www.cszhi.com目录中。

|  |  |
| --- | --- |
| 1  2  3  4 | location ~ ^/(upload|html)/ {  root /web/wwwroot/www.cszhi.com;  expires 30d;  } |

在最后这段设置中，location是对此虚拟主机下动态网页的过滤处理，也就是将所有以.jsp为后缀的文件都交给本机的8080端口处理。

|  |  |
| --- | --- |
| 1  2  3  4 | location ~ .\*.php$ {  index index.php;  proxy\_pass http://localhost:8080;  } |

**2.7 StubStatus模块配置**  
StubStatus模块能够获取Nginx自上次启动以来的工作状态，此模块非核心模块，需要在Nginx编译安装时手工指定才能使用此功能。  
以下指令实指定启用获取Nginx工作状态的功能。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | location /NginxStatus {  stub\_status on;  access\_log logs/NginxStatus.log;  auth\_basic "NginxStatus";  auth\_basic\_user\_file ../htpasswd;  } |

stub\_status设置为“on”表示启用StubStatus的工作状态统计功能。access\_log 用来指定StubStatus模块的访问日志文件。auth\_basic是Nginx的一种认证机制。auth\_basic\_user\_file用来指定认证的密码文件，由于Nginx的auth\_basic认证采用的是与Apache兼容的密码文件，因此需要用Apache的htpasswd命令来生成密码文件，例如要添加一个test用户，可以使用下面方式生成密码文件：

/usr/local/apache/bin/htpasswd -c /opt/nginx/conf/htpasswd test

然后输入两次密码后确认之后添加用户成功。

要查看Nginx的运行状态，可以输入http://ip/NginxStatus，输入创建的用户名和密码就可以看到Nginx的运行状态：

Active connections: 1

server accepts handled requests

34561 35731 354399

Reading: 0 Writing: 3 Waiting: 0

Active connections表示当前活跃的连接数，第三行的三个数字表示 Nginx当前总共处理了34561个连接， 成功创建次握手， 总共处理了354399个请求。最后一行的Reading表示Nginx读取到客户端Header信息数， Writing表示Nginx返回给客户端的Header信息数  
，“Waiting”表示Nginx已经处理完，正在等候下一次请求指令时的驻留连接数。

在最后这段设置中，设置了虚拟主机的错误信息返回页面，通过error\_page指令可以定制各种错误信息的返回页面。在默认情况下，Nginx会在主目录的html目录中查找指定的返回页面，特别需要注意的是，这些错误信息的返回页面大小一定要超过512K，否者会被ie浏览器替换为ie默认的错误页面。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | error\_page 404 /404.html;  error\_page 500 502 503 504 /50x.html;  location = /50x.html {  root html;  }  }  } |