[nginx服务器安装及配置文件详解](https://segmentfault.com/a/1190000002797601)

* [nginx](https://segmentfault.com/t/nginx/blogs)

[**seanlook**](https://segmentfault.com/u/seanlook) 2015年05月26日发布

* 推荐 **18** 推荐
* 收藏 **268** 收藏，72.6k 浏览

nginx在工作中已经有好几个环境在使用了，每次都是重新去网上扒博客，各种编译配置，今天自己也整理一份安装文档和nginx.conf配置选项的说明，留作以后参考。像负载均衡配置（包括健康检查）、缓存（包括清空缓存）配置实例，请参考 <http://segmentfault.com/a/1190000002873747> ，ssl加密请参考 <http://seanlook.com/2015/05/28/nginx-ssl/> 。

1. 安装nginx

1.1 选择稳定版本

我们编译安装nginx来定制自己的模块，机器CentOS 6.2 x86\_64。首先安装缺少的依赖包：

# yum -y install gcc gcc-c++ make libtool zlib zlib-devel openssl openssl-devel pcre pcre-devel

这些软件包如果yum上没有的话可以下载源码来编译安装，只是要注意编译时默认安装的目录，确保下面在安装nginx时能够找到这些动态库文件（ldconfig）。

从 <http://nginx.org/en/download.html> 下载稳定版nginx-1.6.3.tar.gz到/usr/local/src下解压。

为了后续准备我们另外下载2个插件模块：[nginx\_upstream\_check\_module-0.3.0.tar.gz](https://github.com/yaoweibin/nginx_upstream_check_module/releases) —— 检查后端服务器的状态，[nginx-goodies-nginx-sticky-module-ng-bd312d586752.tar.gz](https://bitbucket.org/nginx-goodies/nginx-sticky-module-ng/downloads)（建议在/usr/local/src下解压后将目录重命名为nginx-sticky-module-ng-1.2.5） —— 后端做负载均衡解决session sticky问题。

请注意插件与nginx的版本兼容问题，一般插件越新越好，nginx不用追新，稳定第一。nginx-1.4.7，nginx-sticky-module-1.1，nginx\_upstream\_check\_module-0.2.0，这个搭配也没问题。sticky-1.1与nginx-1.6版本由于更新没跟上编译出错。（可以直接使用Tengine，默认就包括了这些模块）

[root@cachets nginx-1.6.3]# pwd

/usr/local/src/nginx-1.6.3

[root@cachets nginx-1.6.3]# ./configure --prefix=/usr/local/nginx-1.6 --with-pcre \

> --with-http\_stub\_status\_module --with-http\_ssl\_module \

> --with-http\_gzip\_static\_module --with-http\_realip\_module \

> --add-module=../nginx-sticky-module-ng-1.2.5 --add-module=../nginx\_upstream\_check\_module-0.3.0

[root@cachets nginx-1.6.3]# make && make install

1.2 常用编译选项说明

nginx大部分常用模块，编译时./configure --help以--without开头的都默认安装。

* --prefix=PATH ： 指定nginx的安装目录。默认 /usr/local/nginx
* --conf-path=PATH ： 设置nginx.conf配置文件的路径。nginx允许使用不同的配置文件启动，通过命令行中的-c选项。默认为*prefix/conf/nginx.conf*
* --user=name： 设置nginx工作进程的用户。安装完成后，可以随时在nginx.conf配置文件更改user指令。默认的用户名是nobody。--group=name类似
* --with-pcre ： 设置PCRE库的源码路径，如果已通过yum方式安装，使用--with-pcre自动找到库文件。使用--with-pcre=PATH时，需要从PCRE网站下载pcre库的源码（版本4.4 - 8.30）并解压，剩下的就交给Nginx的./configure和make来完成。perl正则表达式使用在location指令和 ngx\_http\_rewrite\_module模块中。
* --with-zlib=PATH ： 指定 zlib（版本1.1.3 - 1.2.5）的源码解压目录。在默认就启用的网络传输压缩模块ngx\_http\_gzip\_module时需要使用zlib 。
* --with-http\_ssl\_module ： 使用https协议模块。默认情况下，该模块没有被构建。前提是openssl与openssl-devel已安装
* --with-http\_stub\_status\_module ： 用来监控 Nginx 的当前状态
* --with-http\_realip\_module ： 通过这个模块允许我们改变客户端请求头中客户端IP地址值(例如X-Real-IP 或 X-Forwarded-For)，意义在于能够使得后台服务器记录原始客户端的IP地址
* --add-module=PATH ： 添加第三方外部模块，如nginx-sticky-module-ng或缓存模块。每次添加新的模块都要重新编译（Tengine可以在新加入module时无需重新编译）

**再提供一种编译方案**：

./configure \

> --prefix=/usr \

> --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 \

> --pid-path=/var/run/nginx/nginx.pid \

> --lock-path=/var/lock/nginx.lock \

> --user=nginx \

> --group=nginx \

> --with-http\_ssl\_module \

> --with-http\_stub\_status\_module \

> --with-http\_gzip\_static\_module \

> --http-client-body-temp-path=/var/tmp/nginx/client/ \

> --http-proxy-temp-path=/var/tmp/nginx/proxy/ \

> --http-fastcgi-temp-path=/var/tmp/nginx/fcgi/ \

> --http-uwsgi-temp-path=/var/tmp/nginx/uwsgi \

> --with-pcre=../pcre-7.8

> --with-zlib=../zlib-1.2.3

1.3 启动关闭nginx

## 检查配置文件是否正确

# /usr/local/nginx-1.6/sbin/nginx -t

# ./sbin/nginx -V # 可以看到编译选项

## 启动、关闭

# ./sbin/nginx # 默认配置文件 conf/nginx.conf，-c 指定

# ./sbin/nginx -s stop

或 pkill nginx

## 重启，不会改变启动时指定的配置文件

# ./sbin/nginx -s reload

或 kill -HUP `cat /usr/local/nginx-1.6/logs/nginx.pid`

当然也可以将 nginx 作为系统服务管理，下载 [nginx](http://sean-images.qiniudn.com/nginx) 到/etc/init.d/，修改里面的路径然后赋予可执行权限。

# service nginx {**start**|**stop**|**status**|restart|reload|configtest}

1.4 yum安装

---- 2015-05-22更新 yum安装rpm包会比编译安装简单很多，默认会安装许多模块，但缺点是如果你想以后安装第三方模块那就没办法了。

# vi /etc/yum.repo.d/nginx.repo

[nginx]

name=nginx repo

baseurl=http://nginx.org/packages/centos/$releasever/$basearch/

gpgcheck=0

enabled=1

剩下的就yum install nginx搞定，也可以yum install nginx-1.6.3安装指定版本（前提是你去packages里看到有对应的版本，默认是最新版稳定版）。

2. nginx.conf配置文件

Nginx配置文件主要分成四部分：main（全局设置）、server（主机设置）、upstream（上游服务器设置，主要为反向代理、负载均衡相关配置）和 location（URL匹配特定位置后的设置），每部分包含若干个指令。main部分设置的指令将影响其它所有部分的设置；server部分的指令主要用于指定虚拟主机域名、IP和端口；upstream的指令用于设置一系列的后端服务器，设置反向代理及后端服务器的负载均衡；location部分用于匹配网页位置（比如，根目录“/”,“/images”,等等）。他们之间的关系式：server继承main，location继承server；upstream既不会继承指令也不会被继承。它有自己的特殊指令，不需要在其他地方的应用。

当前nginx支持的几个指令上下文：

2.1 通用

下面的nginx.conf简单的实现nginx在前端做反向代理服务器的例子，处理js、png等静态文件，jsp等动态请求转发到其它服务器tomcat：

**user** www www;

**worker\_processes** 2;

**error\_log** logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

**pid** logs/nginx.pid;

**events** {

**use** epoll;

**worker\_connections** 2048;

}

**http** {

**include** mime.types;

**default\_type** application/octet-stream;

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

# '$status $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log logs/access.log main;

**sendfile** on;

# tcp\_nopush on;

**keepalive\_timeout** 65;

# gzip压缩功能设置

**gzip** on;

**gzip\_min\_length** 1k;

**gzip\_buffers** 4 16k;

**gzip\_http\_version** 1.0;

**gzip\_comp\_level** 6;

**gzip\_types** text/html text/plain text/css text/javascript application/json application/javascript application/x-javascript application/xml;

**gzip\_vary** on;

# http\_proxy 设置

**client\_max\_body\_size** 10m;

**client\_body\_buffer\_size** 128k;

**proxy\_connect\_timeout** 75;

**proxy\_send\_timeout** 75;

**proxy\_read\_timeout** 75;

**proxy\_buffer\_size** 4k;

**proxy\_buffers** 4 32k;

**proxy\_busy\_buffers\_size** 64k;

**proxy\_temp\_file\_write\_size** 64k;

**proxy\_temp\_path** /usr/local/nginx/proxy\_temp 1 2;

# 设定负载均衡后台服务器列表

**upstream** backend {

#ip\_hash;

**server** 192.168.10.100:8080 max\_fails=2 fail\_timeout=30s ;

**server** 192.168.10.101:8080 max\_fails=2 fail\_timeout=30s ;

}

# 很重要的虚拟主机配置

**server** {

**listen** 80;

**server\_name** itoatest.example.com;

**root** /apps/oaapp;

**charset** utf-8;

**access\_log** logs/host.access.log main;

#对 / 所有做负载均衡+反向代理

**location** / {

**root** /apps/oaapp;

**index** index.jsp index.html index.htm;

**proxy\_pass** http://backend;

**proxy\_redirect** off;

# 后端的Web服务器可以通过X-Forwarded-For获取用户真实IP

**proxy\_set\_header** Host $host;

**proxy\_set\_header** X-Real-IP $remote\_addr;

**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;

}

#静态文件，nginx自己处理，不去backend请求tomcat

**location** ~\* /download/ {

**root** /apps/oa/fs;

}

**location** ~ .\*\.(gif|jpg|jpeg|bmp|png|ico|txt|js|css)$

{

**root** /apps/oaapp;

**expires** 7d;

}

**location** /nginx\_status {

**stub\_status** on;

**access\_log** off;

**allow** 192.168.10.0/24;

**deny** all;

}

**location** ~ ^/(WEB-INF)/ {

**deny** all;

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

**error\_page** 500 502 503 504 /50x.html;

**location** = /50x.html {

**root** html;

}

}

## 其它虚拟主机，server 指令开始

}

2.2 常用指令说明

2.2.1 main全局配置

nginx在运行时与具体业务功能（比如http服务或者email服务代理）无关的一些参数，比如工作进程数，运行的身份等。

* woker\_processes 2  
  在配置文件的顶级*main*部分，worker角色的工作进程的个数，master进程是接收并分配请求给worker处理。这个数值简单一点可以设置为cpu的核数grep ^processor /proc/cpuinfo | wc -l，也是 auto 值，如果开启了ssl和gzip更应该设置成与逻辑CPU数量一样甚至为2倍，可以减少I/O操作。如果nginx服务器还有其它服务，可以考虑适当减少。
* worker\_cpu\_affinity  
  也是写在*main*部分。在高并发情况下，通过设置cpu粘性来降低由于多CPU核切换造成的寄存器等现场重建带来的性能损耗。如worker\_cpu\_affinity 0001 0010 0100 1000; （四核）。
* worker\_connections 2048  
  写在*events*部分。每一个worker进程能并发处理（发起）的最大连接数（包含与客户端或后端被代理服务器间等所有连接数）。nginx作为反向代理服务器，计算公式 最大连接数 = worker\_processes \* worker\_connections/4，所以这里客户端最大连接数是1024，这个可以增到到8192都没关系，看情况而定，但不能超过后面的worker\_rlimit\_nofile。当nginx作为http服务器时，计算公式里面是除以2。
* worker\_rlimit\_nofile 10240  
  写在*main*部分。默认是没有设置，可以限制为操作系统最大的限制65535。
* use epoll  
  写在events部分。在Linux操作系统下，nginx默认使用epoll事件模型，得益于此，nginx在Linux操作系统下效率相当高。同时Nginx在OpenBSD或FreeBSD操作系统上采用类似于epoll的高效事件模型kqueue。在操作系统不支持这些高效模型时才使用select。

2.2.2 http服务器

与提供http服务相关的一些配置参数。例如：是否使用keepalive啊，是否使用gzip进行压缩等。

* sendfile on  
  开启高效文件传输模式，sendfile指令指定nginx是否调用sendfile函数来输出文件，减少用户空间到内核空间的上下文切换。对于普通应用设为 on，如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络I/O处理速度，降低系统的负载。
* keepalive\_timeout 65 : 长连接超时时间，单位是秒，这个参数很敏感，涉及浏览器的种类、后端服务器的超时设置、操作系统的设置，可以另外起一片文章了。长连接请求大量小文件的时候，可以减少重建连接的开销，但假如有大文件上传，65s内没上传完成会导致失败。如果设置时间过长，用户又多，长时间保持连接会占用大量资源。
* send\_timeout : 用于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。
* client\_max\_body\_size 10m  
  允许客户端请求的最大单文件字节数。如果有上传较大文件，请设置它的限制值
* client\_body\_buffer\_size 128k  
  缓冲区代理缓冲用户端请求的最大字节数

**模块http\_proxy：**  
这个模块实现的是nginx作为反向代理服务器的功能，包括缓存功能（另见[文章](http://segmentfault.com/a/1190000002873747)）

* proxy\_connect\_timeout 60  
  nginx跟后端服务器连接超时时间(代理连接超时)
* proxy\_read\_timeout 60  
  连接成功后，与后端服务器两个成功的响应操作之间超时时间(代理接收超时)
* proxy\_buffer\_size 4k  
  设置代理服务器（nginx）从后端realserver读取并保存用户**头**信息的缓冲区大小，默认与proxy\_buffers大小相同，其实可以将这个指令值设的小一点
* proxy\_buffers 4 32k  
  proxy\_buffers缓冲区，nginx针对单个连接缓存来自后端realserver的**响应**，网页平均在32k以下的话，这样设置
* proxy\_busy\_buffers\_size 64k  
  高负荷下缓冲大小（proxy\_buffers\*2）
* proxy\_max\_temp\_file\_size  
  当proxy\_buffers放不下后端服务器的响应内容时，会将一部分保存到硬盘的临时文件中，这个值用来设置最大临时文件大小，默认1024M，它与proxy\_cache没有关系。大于这个值，将从upstream服务器传回。设置为0禁用。
* proxy\_temp\_file\_write\_size 64k  
  当缓存被代理的服务器响应到临时文件时，这个选项限制每次写临时文件的大小。proxy\_temp\_path（可以在编译的时候）指定写到哪那个目录。

proxy\_pass，proxy\_redirect见 location 部分。

**模块http\_gzip：**

* gzip on : 开启gzip压缩输出，减少网络传输。  
    
  + gzip\_min\_length 1k ： 设置允许压缩的页面最小字节数，页面字节数从header头得content-length中进行获取。默认值是20。建议设置成大于1k的字节数，小于1k可能会越压越大。
  + gzip\_buffers 4 16k ： 设置系统获取几个单位的缓存用于存储gzip的压缩结果数据流。4 16k代表以16k为单位，安装原始数据大小以16k为单位的4倍申请内存。
  + gzip\_http\_version 1.0 ： 用于识别 http 协议的版本，早期的浏览器不支持 Gzip 压缩，用户就会看到乱码，所以为了支持前期版本加上了这个选项，如果你用了 Nginx 的反向代理并期望也启用 Gzip 压缩的话，由于末端通信是 http/1.0，故请设置为 1.0。
  + gzip\_comp\_level 6 ： gzip压缩比，1压缩比最小处理速度最快，9压缩比最大但处理速度最慢(传输快但比较消耗cpu)
  + gzip\_types ：匹配mime类型进行压缩，无论是否指定,”text/html”类型总是会被压缩的。
  + gzip\_proxied any ： Nginx作为反向代理的时候启用，决定开启或者关闭后端服务器返回的结果是否压缩，匹配的前提是后端服务器必须要返回包含”Via”的 header头。
  + gzip\_vary on ： 和http头有关系，会在响应头加个 Vary: Accept-Encoding ，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。。

2.2.3 server虚拟主机

http服务上支持若干虚拟主机。每个虚拟主机一个对应的server配置项，配置项里面包含该虚拟主机相关的配置。在提供mail服务的代理时，也可以建立若干server。每个server通过监听地址或端口来区分。

* listen  
  监听端口，默认80，小于1024的要以root启动。可以为listen \*:80、listen 127.0.0.1:80等形式。
* server\_name  
  服务器名，如localhost、www.example.com，可以通过正则匹配。

**模块http\_stream**  
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，upstream后接负载均衡器的名字，后端realserver以 host:port options; 方式组织在 {} 中。如果后端被代理的只有一台，也可以直接写在 proxy\_pass 。

2.2.4 location

http服务中，某些特定的URL对应的一系列配置项。

* root /var/www/html  
  定义服务器的默认网站根目录位置。如果locationURL匹配的是子目录或文件，root没什么作用，一般放在server指令里面或/下。
* index index.jsp index.html index.htm  
  定义路径下默认访问的文件名，一般跟着root放
* proxy\_pass http:/backend  
  请求转向backend定义的服务器列表，即反向代理，对应upstream负载均衡器。也可以proxy\_pass http://ip:port。
* proxy\_redirect off;  
  proxy\_set\_header Host $host;  
  proxy\_set\_header X-Real-IP $remote\_addr;  
  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
  这四个暂且这样设，如果深究的话，每一个都涉及到很复杂的内容，也将通过另一篇文章来解读。

关于location匹配规则的写法，可以说尤为关键且基础的，参考文章 [nginx配置location总结及rewrite规则写法](http://segmentfault.com/a/1190000002797606);

2.3 其它

2.3.1 访问控制 allow/deny

Nginx 的访问控制模块默认就会安装，而且写法也非常简单，可以分别有多个allow,deny，允许或禁止某个ip或ip段访问，依次满足任何一个规则就停止往下匹配。如：

**location** /nginx-status {

**stub\_status** on;

**access\_log** off;

# auth\_basic "NginxStatus";

# auth\_basic\_user\_file /usr/local/nginx-1.6/htpasswd;

**allow** 192.168.10.100;

**allow** 172.29.73.0/24;

**deny** all;

}

我们也常用 httpd-devel 工具的 htpasswd 来为访问的路径设置登录密码：

**# htpasswd -c htpasswd admin**

New passwd:

Re-type **new** password:

Adding password **for** user admin

**# htpasswd htpasswd admin //修改admin密码**

**# htpasswd htpasswd sean //多添加一个认证用户**

这样就生成了默认使用CRYPT加密的密码文件。打开上面nginx-status的两行注释，重启nginx生效。

2.3.2 列出目录 autoindex

Nginx默认是不允许列出整个目录的。如需此功能，打开nginx.conf文件，在location，server 或 http段中加入autoindex on;，另外两个参数最好也加上去:

* autoindex\_exact\_size off; 默认为on，显示出文件的确切大小，单位是bytes。改为off后，显示出文件的大概大小，单位是kB或者MB或者GB
* autoindex\_localtime on;  
  默认为off，显示的文件时间为GMT时间。改为on后，显示的文件时间为文件的服务器时间

**location** /images {

**root** /var/www/nginx-default/images;

**autoindex** on;

**autoindex\_exact\_size** off;

**autoindex\_localtime** on;

}