# [Nginx 之一：编译安装nginx 1.8.1 及配置](http://www.cnblogs.com/zhang-shijie/p/5294162.html)

安装

1、环境准备：先安装准备环境

yum install gcc gcc-c++ automake pcre pcre-devel zlip zlib-devel openssl openssl-devel

　　gcc为GNU Compiler Collection的缩写，可以编译C和C++源代码等，它是GNU开发的C和C++以及其他很多种语言 的编译器（最早的时候只能编译C，后来很快进化成一个编译多种语言的集合，如Fortran、Pascal、Objective-C、Java、Ada、 Go等。）

　　gcc 在编译C++源代码的阶段，只能编译 C++ 源文件，而不能自动和 C++ 程序使用的库链接（编译过程分为编译、链接两个阶段，注意不要和可执行文件这个概念搞混，相对可执行文件来说有三个重要的概念：编译（compile）、链接（link）、加载（load）。源程序文件被编译成目标文件，多个目标文件连同库被链接成一个最终的可执行文件，可执行文件被加载到内存中运行）。因此，通常使用 g++ 命令来完成 C++ 程序的编译和连接，该程序会自动调用 gcc 实现编译。

　　gcc-c++也能编译C源代码，只不过把会把它当成C++源代码，后缀为.c的，gcc把它当作是C程序，而g++当作是c++程序；后缀为.cpp的，两者都会认为是c++程序，注意，虽然c++是c的超集，但是两者对语法的要求是有区别的。

　　automake是一个从Makefile.am文件自动生成Makefile.in的工具。为了生成Makefile.in，automake还需用到perl，由于automake创建的发布完全遵循GNU标准，所以在创建中不需要perl。libtool是一款方便生成各种程序库的工具。

　　pcre pcre-devel：在Nginx编译需要 PCRE(Perl Compatible Regular Expression)，因为Nginx 的Rewrite模块和HTTP 核心模块会使用到PCRE正则表达式语法。

　　zlip zlib-devel：nginx启用压缩功能的时候，需要此模块的支持。

　　openssl openssl-devel：开启SSL的时候需要此模块的支持。

 2、下载nginx 安装包：  官网地址：http://nginx.org/

截止得到当前，最新的版本为1.8.1，在linux使用wget下载:

[root@Server1 ~]# wget http://nginx.org/download/nginx-1.8.1.tar.gz

--2016-04-23 10:22:55-- http://nginx.org/download/nginx-1.8.1.tar.gz

Resolving nginx.org (nginx.org)... 206.251.255.63, 95.211.80.227, 2001:1af8:4060:a004:21::e3, ...

Connecting to nginx.org (nginx.org)|206.251.255.63|:80... connected.

HTTP request sent, awaiting response... 200 OK

Length: 833473 (814K) [application/octet-stream]

Saving to: ‘nginx-1.8.1.tar.gz.1’

100%[==============================================================================================================>] 833,473 251KB/s in 3.2s

2016-04-23 10:23:00 (251 KB/s) - ‘nginx-1.8.1.tar.gz.1’ saved [833473/833473]

3、解压安装包:

[root@Server1 ~]# tar xvf nginx-1.8.1.tar.gz

[root@Server1 ~]# cd nginx-1.8.1

[root@Server1 nginx-1.8.1]$ ls

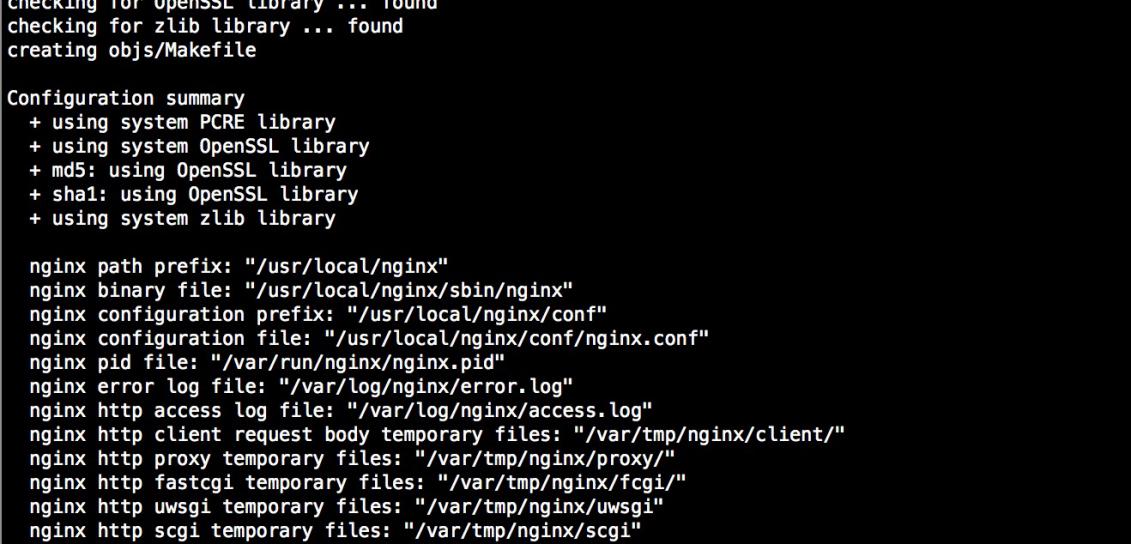
auto CHANGES CHANGES.ru conf configure contrib html LICENSE man README src

4、编译nginx：make

编译是为了检查系统环境是否符合编译安装的要求，比如是否有gcc编译工具，是否支持编译参数当中的模块，并根据开启的参数等生成Makefile文件为下一步做准备：

[root@Server1 nginx-1.8.1]# ./configure --prefix=/usr/local/nginx --sbin-path=/usr/local/nginx/sbin/nginx --conf-path=/usr/local/nginx/conf/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 --http-scgi-temp-path=/var/tmp/nginx/scgi --with-pcre

结果如下:

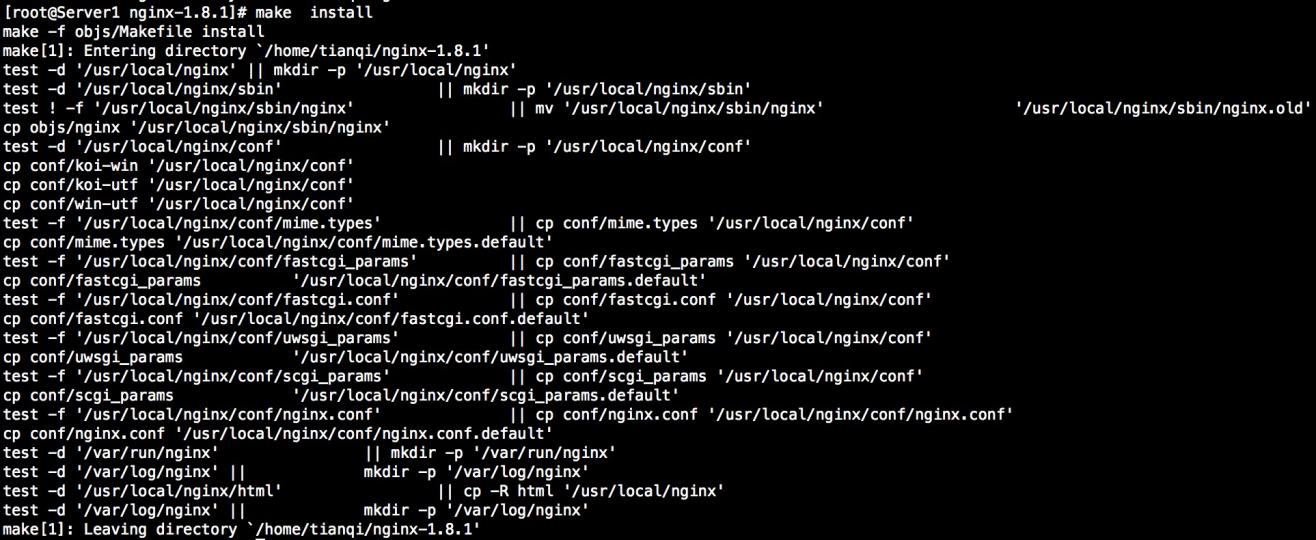


5、生成脚本及配置文件：make

编译步骤，根据Makefile文件生成相应的模块



6、安装：make install

创建目录，并将生成的模块和文件复制到相应的目录： 

备注：nginx完成安装以后，有四个主要的目录：

conf：保存nginx所有的配置文件，其中nginx.conf是nginx服务器的最核心最主要的配置文件，其他的.conf则是用来配置nginx相关的功能的，例如fastcgi功能使用的是fastcgi.conf和fastcgi\_params两个文件，配置文件一般都有个样板配置文件，是文件名.default结尾，使用的使用将其复制为并将default去掉即可。

html目录中保存了nginx服务器的web文件，但是可以更改为其他目录保存web文件,另外还有一个50x的web文件是默认的错误页面提示页面。

logs：用来保存nginx服务器的访问日志错误日志等日志，logs目录可以放在其他路径，比如/var/logs/nginx里面。

sbin：保存nginx二进制启动脚本，可以接受不同的参数以实现不同的功能。

7、启动：

将监听端口改为8090，避免80端口冲突：

listen 8090;

8、通过命令启动和关闭nginx：

[root@Server1 sbin]# /usr/local/nginx/sbin/nginx/nginx

nginx: [emerg] getpwnam("nginx") failed #没有nginx用户

[root@Server1 sbin]# /usr/local/nginx/sbin/nginx/nginx

nginx: [emerg] mkdir() "/var/tmp/nginx/client/" failed (2: No such file or directory) #目录不存在

[root@Server1 sbin]# /usr/local/nginx/sbin/nginx/nginx #直到没有报错，才算启动完成

[IMG_268](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

9、重读配置文件和关闭服务：

[root@Server1 local]# /usr/local/nginx/sbin/nginx/nginx #启动 服务

[root@Server1 local]# /usr/local/nginx/sbin/nginx/nginx -s reload #不停止服务重读配置文件

[root@Server1 local]# /usr/local/nginx/sbin/nginx/nginx -s stop #停止服务 #停止服务

10.验证端口是否开启：

[IMG_269](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

[root@Server1 sbin]# ps -ef | grep nginx

root 13228 1 0 Apr23 ? 00:00:00 nginx: master process /usr/local/nginx/sbin/nginx/nginx #nginx的主进程,只有一个主进程

nginx 13229 13228 0 Apr23 ? 00:00:00 nginx: worker process #nginx工作进程，默认只有一个，可以通过修改nginx.conf中的worker\_processes 1; 参数启动多个工作进程

root 13295 1400 0 00:01 pts/0 00:00:00 grep --color=auto nginx

[root@Server1 local]# lsof -i:8090 #显示占用8090的进程

COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME

nginx 13337 root 6u IPv4 5932680 0t0 TCP \*:8090 (LISTEN)

nginx 13338 nginx 6u IPv4 5932680 0t0 TCP \*:8090 (LISTEN)

[IMG_270](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

 11、通过给nginx的主进程ID号发送信号启动或停止nginx：

获取nginx主进程号的办法：

[root@Server1 nginx]# cat /var/run/nginx/nginx.pid #查看nginx的pid文件，此文件保存的就是nginx的主进程id13337 #次ID是随机的，每次启动都不一样的

[root@Server1 nginx]# ps -ef | grep nginx #过滤nginx的进程号

root 13337 1 0 00:05 ? 00:00:00 nginx: master process /usr/local/nginx/sbin/nginx/nginx

nginx 21568 13337 0 10:58 ? 00:00:00 nginx: worker process

支持的信号：

[IMG_271](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

[root@Server1 nginx]# kill -QUIT 13337 #平缓关闭Nginx，即不再接受新的请求，但是等当前请求处理完毕后再关闭Nginx。

[root@Server1 nginx]# kill -TERM 21665 #快速停止Nginx服务

[root@Server1 nginx]# kill -HUP 21703 #使用新的配置文件启动进程然后平缓停止原有的nginx进程，即平滑重启。

[root@Server1 nginx]# kill -USR1 21703 #重新打开配置文件，用于nginx 日志切割

日期切割的脚本：

#!/bin/bash

PID=`cat /var/run/nginx/nginx.pid`mv /var/log/nginx/access.log /var/log/nginx/`date +%Y\_%m\_%d:%H:%M:%S`.access.logkill -USR1 $PID

[root@Server1 nginx]# kill -USR2 21703 #使用新版本的nginx文件启动服务，然后在平缓停止原有的nginx服务，即平滑升级。

[root@Server1 nginx]# kill -WINCH 21703 #平滑停止nginx的工作进程，用于nginx平滑升级。

[IMG_272](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

三：nginx 主配置文件:nginx.conf

3.1：默认配置：配置文件默认保存在path/conf当中，默认的配置文件为nginx.conf，以下是编译安装后的默认配置：

[IMG_273](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

[root@Server1 conf]# grep -v "#" nginx.conf | grep -v "^$"

　　#全局生效，主要设置nginx的启动用户/组，启动的工作进程数量，Nginx的PID路径，日志路径等。

worker\_processes 1; #默认启动一个工作进程

events { #events设置快，主要影响nginx服务器与用户的网络连接，比如是否允许同时接受多个网络连接，使用哪种事件驱动模型处理请求，每个工作进程可以同时支持的最大连接数，是否开启对多工作进程下的网络连接进行序列化等。

worker\_connections 1024; #设置nginx可以接受的最大并发，多个进程只和

}

http { #http块是Nginx服务器配置中的重要部分，缓存、代理和日志格式定义等绝大多数功能和第三方模块都可以在这设置，http块可以包含多个server块，而一个server块中又可以包含多个location块，server块可以配置文件引入、MIME-Type定义、日志自定义、是否启用sendfile、连接超时时间和单个链接的请求上限等。

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

sendfile on; #是否调用 sendfile 函数（zero copy -->零copy方式）来输出文件，普通应用打开，可以大幅提升nginx的读文件性能，如果服务器是下载的就需要关闭，

keepalive\_timeout 65; #长连接超时时间，单位是秒

server { #设置一个虚拟机主机，可以包含自己的全局快，同时也可以包含多个locating模块。比如本虚拟机监听的端口、本虚拟机的名称和IP配置，多个server 可以使用一个端口，比如都使用8090端口提供web服务、

listen 8090; #server的全局配置，配置监听的端口

server\_name localhost; #本server的名称，当访问此名称的时候nginx会调用当前serevr内部的配置进程匹配。

location / { #location其实是server的一个指令，为nginx服务器提供比较多而且灵活的指令，都是在location中提现的，主要是基于nginx接受到的请求字符串，对用户请求的UIL进行匹配，并对特定的指令进行处理，包括地址重定向、数据缓存和应答控制等功能都是在这部分实现，另外很多第三方模块的配置也是在location模块中配置。

root html; #相当于默认页面的目录名称，默认是相对路径，可以使用绝对路径配置。

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html; #错误页面的文件名称

location = /50x.html { #location处理对应的不同错误码的页面定义到/50x.html，这个跟对应其server中定义的目录下。

root html; #定义默认页面所在的目录

}

}

}

[IMG_274](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

 3.2：配置nginx 主进程的启动用户和工作进程数：

user xxxx; #每一条指令都要以分号结尾

worker\_processes 1; #可以指定启动的固定nginx进程数，或使用auto，auto是启动与当前CPU 线程相同的进程数，如CPU是四核八线程的就启动八个进程的Nginx工作进程。

3.3：绑定Nginx 工作进程到不同的CPU上:

　　默认Nginx是不进行保定的，绑定并不能是当前nginx进程独占以一核心CPU，但是可以保证此进程不会运行在其他核心上，这就极大减少了nginx 工作进程在不同cpu上的跳转，减少了CPU对进程的资源分配与回收，因此可以有效的提升nginx服务器的性能，配置如下：

[IMG_275](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

[root@Server1 nginx]# grep process /proc/cpuinfo | wc -l #确认CPU的核心数量4

四个线程CPU的配置：

worker\_processes 4;

worker\_cpu\_affinity 0001 0010 0100 1000;

八个线程CPU的配置：

worker\_processes 8;

worker\_cpu\_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000；

[IMG_276](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

3.4：PID和错误日志文件路径：

[IMG_277](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

#pid logs/nginx.pid; #可以指定绝对路径

#error\_log logs/error.log; #指定错误日志路径

#error\_log logs/error.log notice; #指定一个日志记录级别

#error\_log logs/error.log info;

支持的日志级别：

语法:

error\_log file [ debug | info | notice | warn | error | crit ] | [{ debug\_core | debug\_alloc | debug\_mutex | debug\_event | debug\_http | debug\_mail | debug\_mysql } ]

日志级别 = 错误日志级别 | 调试日志级别; 或者

日志级别 = 错误日志级别;

错误日志的级别: emerg, alert, crit, error, warn, notic, info, debug,

调试日志的级别: debug\_core, debug\_alloc, debug\_mutex, debug\_event, debug\_http, debug\_mail, debug\_mysql

error\_log 指令的日志级别配置分为错误日志级别和调试日志级别，错误日志只能设置一个级别，而且错误日志必须书写在调试日志级别的前面，另外调试日志可以设置多个级别，其他配置方法可能无法满足需求。

[IMG_278](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

 3.5：配置文件的引入：include

[IMG_279](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);" \o "复制代码)

include file； #file是要导入的文件，支持相对路径，一般在html目录里面

导入一个conf文件，并配置不同主机名的页面,编辑nginx.conf主配置文件:

include /usr/local/nginx/conf.d/samsung.conf; #在最后一个大括号里面加入一项，\*是导入任何以conf结尾的配置文件

在/usr/local/nginx/conf.d/创建一个samsung.conf，内容如下：

[root@Server1 nginx]# grep -v "#" conf.d/samsung.conf | grep -v "^$"

server {

listen 8090;

server\_name samsung.chinacloudapp.cn;

location / {

root html;

index index1.html index.htm;

}

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

location = /50x.html {

root html;

}

}