

**易云nginx服务安装配置手册**

**V1.0**

**管理部门：\_\_\_**技术中心**\_\_\_**

**撰 写 人：\_\_\_**\_\_\_\_\_\_**\_\_\_\_\_\_**

**审 核 人：\_\_\_\_**\_\_\_\_\_\_**\_\_\_\_\_**

**日 期：\_\_\_**\_\_\_\_\_\_**\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **内容** | **修订时间** | **制定人** |
| **V 1.0** | **最初版本制定** | **2016-8-2** |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 环境准备 1](#_Toc457910311)

[1.1. 编译环境安装 1](#_Toc457910312)

[1.2. 所需软件 1](#_Toc457910313)

[2. 安装PCRE库 2](#_Toc457910314)

[3. 安装zlib库 2](#_Toc457910315)

[4. 安装openssl 2](#_Toc457910316)

[5. 安装 nginx-1.8.1 3](#_Toc457910317)

[6. 启动 4](#_Toc457910318)

[7. nginx常见配置 4](#_Toc457910319)

[7.1. 反向代理配置示例 4](#_Toc457910320)

[7.2. 负载均衡配置 4](#_Toc457910321)

[7.3. 负载均衡和反向代理配置示例 5](#_Toc457910322)

[7.4. 动静分离配置 6](#_Toc457910323)

[7.5. 日志配置 7](#_Toc457910324)

[7.6. check\_status配置 8](#_Toc457910325)

[7.7. 页面检查配置 8](#_Toc457910326)

[附录一：nginx.conf配置详解 9](#_Toc457910327)

[附录二：nginx起停脚本 12](#_Toc457910328)

[附录三：nginx日志切割脚本 12](#_Toc457910329)

# 环境准备

以nginx1.8.1安装与基本配置为例。

操作系统：CentOS6.7\_x64

## 编译环境安装

yum -y install gcc gcc-c++ automake autoconf libtool make cronolog git

## 所需软件

* **nginx-1.8.1.tar.gz**

- http://nginx.org/en/download.html

* **- pcre-8.36.tar.gz**

- 官网下载：https://sourceforge.net/projects/pcre/files/pcre/

* **- zlib-1.2.8.tar.gz**

- http://zlib.net/

* **- openssl-1.0.1c.tar.gz**

- https://www.openssl.org/source/

* **- nginx\_upstream\_check\_module-master.zip**

- https://codeload.github.com/yaoweibin/nginx\_upstream\_check\_module/zip/master

* **- fastdfs-nginx-module（FastDFS代理配置需要下载，http服务代理不需要）**

- git clone https://github.com/happyfish100/fastdfs-nginx-module.git ./fastdfs-nginx-module

上述软件本地存路径选定：/usr/local/src

cd /usr/local/src

wget http://exim.mirror.fr/pcre/pcre-8.36.tar.gz

wget http://zlib.net/zlib-1.2.8.tar.gz

wget http://www.openssl.org/source/openssl-1.0.1c.tar.gz

wget http://nginx.org/download/nginx-1.8.1.tar.gz

git clone https://github.com/yaoweibin/nginx\_upstream\_check\_module.git ./nginx\_upstream\_check\_module

git clone https://github.com/happyfish100/fastdfs-nginx-module.git ./fastdfs-nginx-module

# 安装PCRE库

官网下载地址：https://sourceforge.net/projects/pcre/files/pcre/

镜像下载：http://exim.mirror.fr/pcre/pcre-8.36.tar.gz

cd /usr/local/src

wget http://exim.mirror.fr/pcre/pcre-8.36.tar.gz

tar -zxf pcre-8.36.tar.gz

cd pcre-8.36

./configure

make

make install

cd ..

# 安装zlib库

cd /usr/local/src

wget http://zlib.net/zlib-1.2.8.tar.gz

tar -zxf zlib-1.2.8.tar.gz

cd zlib-1.2.8

./configure

make

make install

cd ..

# 安装openssl

cd /usr/local/src

wget http://www.openssl.org/source/openssl-1.0.1c.tar.gz

tar -zxf openssl-1.0.1c.tar.gz

cd openssl-1.0.1c/

./configure

make

make install

cd ..

# 安装 nginx-1.8.1

cd /usr/local/src

wget http://nginx.org/download/nginx-1.8.1.tar.gz

tar -zxf nginx-1.8.1.tar.gz

cd nginx-1.8.1

nginx反向代理应用服务编译命令：

./configure --prefix=/usr/local/nginx-1.8.1 --with-http\_ssl\_module --with-http\_gzip\_static\_module --with-pcre=/usr/local/src/pcre-8.36 --with-zlib=/usr/local/src/zlib-1.2.8 --with-openssl=/usr/local/src/openssl-1.0.1c --with-http\_stub\_status\_module --with-http\_realip\_module --add-module=/usr/local/src/nginx\_upstream\_check\_module

fastdfs代理编译命令：

./configure --prefix=/usr/local/nginx-1.8.1 --with-http\_ssl\_module --with-http\_gzip\_static\_module --with-pcre=/usr/local/src/pcre-8.36 --with-zlib=/usr/local/src/zlib-1.2.8 --with-openssl=/usr/local/src/openssl-1.0.1c --with-http\_stub\_status\_module --with-http\_realip\_module --add-module=/usr/local/src/fastdfs-nginx-module/src

说明：以下参数分别指的是对应的源码路径，编译nginx时，直接将这些程序编译到nginx里面。这样nginx启动时就不会采用动态加载的方式去load。这种方式会比动态加载有更高的效率。

--with-pcre=/usr/local/src/pcre-8.34 #指的是 pcre-8.36 的源码路径。

--with-zlib=/usr/local/src/zlib-1.2.7 #指的是 zlib-1.2.7 的源码路径。

--with-openssl=/usr/local/src/openssl-1.0.1c #指的是 openssl-1.0.1c的源码路径。

--add-module=/usr/local/src/nginx\_upstream\_check\_module #指 nginx\_upstream\_check\_module源码路径。

--add-module=/usr/local/src/fastdfs-nginx-module/src #指的是 fastdfs-nginx-module 的源码路径。

安装：

make

make install

cd ..

创建软连接：

ln -s /usr/local/nginx-1.8.1 /usr/local/nginx

# 启动

确保系统的 80 端口没被其他程序占用，运行如下命令来启动 Nginx：

启动nginx服务：

/usr/local/nginx/nginx

查看监听端口：

netstat -ano|grep 80

关闭命令：

/usr/local/nginx/nginx -s stop

优雅重启：

/usr/local/nginx/nginx -s reload

# nginx常见配置

修改部署目录下conf子目录的nginx.conf文件（如/usr/local/nginx/conf/nginx.conf）内容，可调整相关配置。

## 反向代理配置示例

|  |
| --- |
| location / {  #设置主机头和客户端真实地址，以便服务器获取客户端真实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;  #禁用缓存  proxy\_buffering off;  #设置反向代理的地址  proxy\_pass http://192.168.1.1;  } |

说明：代理地址根据实际情况修改。

## 负载均衡配置

nginx 的 upstream默认是以轮询的方式实现负载均衡，这种方式中，每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。

另外一种方式是ip\_hash：每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session的问题。

**负载均衡配置示例：**

|  |
| --- |
| upstream test{  #ip\_hash;  server 192.168.1.251;  server 192.168.1.252;  server 192.168.1.247;  }  server {  listen 80;  server\_name helloword;  location / {  #反向代理的地址  proxy\_pass http://test;  }  } |

Upstream命名和服务器地址根据实际情况修改。

## 负载均衡和反向代理配置示例

|  |
| --- |
| nginx.conf：  worker\_processes 1;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  sendfile on;  keepalive\_timeout 65;  upstream test{  #ip\_hash;  server 192.168.1.251;  server 192.168.1.252;  server 192.168.1.247;  }  server {  listen 80;  server\_name 2;  location / {  #设置主机头和客户端真实地址，以便服务器获取客户端真实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;  #禁用缓存  proxy\_buffering off;  #反向代理的地址  proxy\_pass http://test;  }  }  } |

## 动静分离配置

|  |
| --- |
| worker\_processes 1;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  sendfile on;  keepalive\_timeout 65;  upstream test {  #ip\_hash;  server 192.168.1.251;  server 192.168.1.252;  server 192.168.1.247;  }  server {  listen 80;  server\_name 2;  #配置Nginx动静分离，定义的静态页面直接从Nginx发布目录读取。  location ~ .\*\.(html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css)$ {  root /usr/local/nginx/html/myloan;  #expires定义用户浏览器缓存的时间为7天，如果静态页面不常更新，可以设置更长，这样可以节省带宽和缓解服务器的压力  expires 7d;  }  #所有jsp、do的动态请求都交给后面的tomcat处理  location ~ (\.jsp)|(\.do)$ {  #tomcat地址  proxy\_pass http://test;  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;  client\_max\_body\_size 10m;  client\_body\_buffer\_size 128k;  proxy\_connect\_timeout 90;  proxy\_send\_timeout 90;  proxy\_read\_timeout 90;  proxy\_buffer\_size 4k;  proxy\_buffers 4 32k;  proxy\_busy\_buffers\_size 64k;  proxy\_temp\_file\_write\_size 64k;  }  }  } |

## 日志配置

为便于ELK采集分析nginx日志，建议将nginx日志配置为json格式，具体配置方法如下：

log\_format logstash\_json '{ "@timestamp": "$time\_iso8601", '

'"client\_ip": "$remote\_addr", '

'"resp\_len": "$body\_bytes\_sent", '

'"waster\_time": "$request\_time", '

'"status": "$status", '

'"request\_path": "$request", '

'"request\_method": "$request\_method", '

'"upstream": "$upstream\_addr", '

'"upstream\_status": "$upstream\_status", '

'"ups\_resp\_time": "$upstream\_response\_time" }';

#打开nginx访问日志：

access\_log logs/access\_ json.log logstash\_json;

**输出日志：**

{"logtime": "30/Jul/2016:17:09:06 +0800", "client\_ip": "10.59.34.140", "resp\_len": "44", "waster\_time": "0.009", "status": "200", "request\_path": "POST /mes/rpc/mesMessageServiceRpc HTTP/1.1", "request\_method": "POST", "request\_url": "-", "http\_forwarded": "-", "user\_agent": "-" ,"upstream": "10.59.2.40:7000","upstream\_status": "200"}

**json美化后显示为：**

{

"logtime": "15/Jul/2016:13:51:17 +0800",

"client\_ip": "10.59.34.140",

"resp\_len": "44",

"waster\_time": "0.008",

"status": "200",

"request\_path": "POST /mes/rpc/mesMessageServiceRpc HTTP/1.1",

"request\_method": "POST",

"request\_url": "-",

"body\_bytes\_sent": "44",

"http\_forwarded": "-",

"user\_agent": "-",

"upstream": "10.59.2.40:7000",

"upstream\_status": "200"

}

**上述可以直接配置到logstach中进行日志采集和分析，而无需进行正则匹配和过滤，节约服务器资源，logstach中只需指定日志格式输出为json：**

**codec => "json"**

## check\_status配置

|  |
| --- |
| location /check-status {  check\_status;  access\_log off;  #allow SOME.IP.ADD.RESS;  #deny all;  } |

## 页面检查配置

|  |
| --- |
| upstream mes {  server 10.59.2.40:7000;  server 10.59.2.41:7000;  check interval=3000 rise=2 fall=5 timeout=1000 type=http;  check\_http\_send "GET /mes/index.jsp HTTP/1.0\r\n\r\n";  } |

# 附录一：nginx.conf配置详解

#运行用户

user nginx;

#启动进程，通常设置成和cpu的数量相等

worker\_processes 1;

#全局错误日志及PID文件

error\_log /var/log/nginx/error.log;

pid /var/run/nginx.pid;

#工作模式及连接数上限

events {

use epoll; #epoll是多路复用IO(I/O Multiplexing)中的一种方式,但是仅用于linux2.6以上内核,可以大大提高nginx的性能

worker\_connections 1024;#单个后台worker process进程的最大并发链接数

# multi\_accept on;

}

#设定http服务器，利用它的反向代理功能提供负载均衡支持

http {

#设定mime类型,类型由mime.type文件定义

include /etc/nginx/mime.types;

default\_type application/octet-stream;

#设定日志格式

access\_log /var/log/nginx/access.log;

#sendfile 指令指定 nginx 是否调用 sendfile 函数（zero copy 方式）来输出文件，对于普通应用，

#必须设为 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uptime.

sendfile on;

#tcp\_nopush on;

#连接超时时间

#keepalive\_timeout 0;

keepalive\_timeout 65;

tcp\_nodelay on;

#开启gzip压缩

gzip on;

gzip\_disable "MSIE [1-6]\.(?!.\*SV1)";

#设定请求缓冲

client\_header\_buffer\_size 1k;

large\_client\_header\_buffers 4 4k;

include /etc/nginx/conf.d/\*.conf;

include /etc/nginx/sites-enabled/\*;

#设定负载均衡的服务器列表

upstream mysvr {

#weigth参数表示权值，权值越高被分配到的几率越大

#本机上的Squid开启3128端口

server 192.168.8.1:3128 weight=5;

server 192.168.8.2:80 weight=1;

server 192.168.8.3:80 weight=6;

}

server {

#侦听80端口

listen 80;

#定义使用www.xx.com访问

server\_name www.xx.com;

#设定本虚拟主机的访问日志

access\_log logs/www.xx.com.access.log main;

#默认请求

location / {

root /root; #定义服务器的默认网站根目录位置

index index.php index.html index.htm; #定义首页索引文件的名称

fastcgi\_pass www.xx.com;

fastcgi\_param SCRIPT\_FILENAME $document\_root/$fastcgi\_script\_name;

include /etc/nginx/fastcgi\_params;

}

# 定义错误提示页面

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

location = /50x.html {

root /root;

}

#静态文件，nginx自己处理

location ~ ^/(images|javascript|js|css|flash|media|static)/ {

root /var/www/virtual/htdocs;

#过期30天，静态文件不怎么更新，过期可以设大一点，如果频繁更新，则可以设置得小一点。

expires 30d;

}

#PHP 脚本请求全部转发到 FastCGI处理. 使用FastCGI默认配置.

location ~ \.php$ {

root /root;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME /home/www/www$fastcgi\_script\_name;

include fastcgi\_params;

}

#设定查看Nginx状态的地址

location /NginxStatus {

stub\_status on;

access\_log on;

auth\_basic "NginxStatus";

auth\_basic\_user\_file conf/htpasswd;

}

#禁止访问 .htxxx 文件

location ~ /\.ht {

deny all;

}

}

}

#####################################################################

nginx漏洞：nginx before 1.8.1 and 1.9.x before 1.9.10

漏洞描述：

Clickjacking（点击劫持）是由互联网安全专家罗伯特·汉森和耶利米·格劳斯曼在2008年首创的。

是一种视觉欺骗手段，在web端就是iframe嵌套一个透明不可见的页面，让用户在不知情的情况下，点击攻击者想要欺骗用户点击的位置。服务器端没有返回X-Frame-Options头部，导致该站点存在点击劫持攻击的威胁。X-Frame-Options HTTP 响应头，可以指示浏览器是否应该加载一个iframe中的页面。网站可以通过设置X-Frame-Options阻止站点内的页面被其他页面嵌入从而防止点击劫持。如果您确定不需要将网站内容嵌入其他站点中，请开启选项。

解决方案:在站点配置文件httpd.conf中添加如下配置，限制只有站点内的页面才可以嵌入iframe。

Header always append X-Frame-Options SAMEORIGIN

配置之后重启apache使其生效。该配置方式对IBM HTTP Server同样适用。

如果同一apache服务器上有多个站点，只想针对一个站点进行配置，可以修改.htaccess文件，添加如下内容：

Header append X-FRAME-OPTIONS "SAMEORIGIN"

到 nginx/conf文件夹下，修改nginx.conf ，添加如下内容：

add\_header X-Frame-Options "SAMEORIGIN";

重启Nginx服务。

# 附录二：nginx起停脚本



# 附录三：nginx日志切割脚本

