## Nginx相关背景

Nginx与Apache类似，也是一个高性能的HTTP和反向代理服务器软件，还是一个IMAP/POP3/SMTP代理服务器。

Nginx与Apache相比有以下优势：在性能上，它占用很少的系统资源，能支持更多的并发连接，达到更高的访问效率；在功能上，Nginx是优秀的代理服务器和负载均衡服务器；在安装配置上，Nginx安装简单、配置灵活。

而且Nginx还支持热部署（**敲黑板，这个很方便**），启动速度特别迅速，可以在不间断服务的情况下对软件版本或配置进行升级，即使运行数月也无需重新启动。

## Nginx安装

* **安装依赖**
  1. yum install gcc
  2. yum install pcre-devel
  3. yum install zlib zlib-devel
  4. yum install openssl openssl-devel
  5. //一键安装上面四个依赖

yum -y install gcc zlib zlib-devel pcre-devel openssl openssl-devel

**注：如果安装的依赖不成功或者不全，会影响Nginx相关功能的使用。**

* **下载nginx的tar包**

1. //创建一个文件夹

cd /usr/local

mkdir nginx

cd nginx

1. //下载tar包

wget http://[nginx.org/download/nginx-1.13.7.tar.gz](http://nginx.org/download/nginx-1.13.7.tar.gz)

**注：如果weget一直不能成功，直接打开浏览器输入上面网址，浏览器会自动下载压缩包，然后在rz到对应服务器即可。**

tar -xvf nginx-1.13.7.tar.g

* **安装nginx**

1. //进入nginx目录

cd /usr/nginx/nginx-1.13.7

1. //执行命令

./configure --user=www --group=www --prefix=/usr/local/nginx --with-http\_stub\_status\_module --with-http\_ssl\_module（着了直接安装项目所用的第三方模块）

1. //执行make命令

make

1. //执行make install命令

make install

## Nginx常用命令

如果你上面是按照流程一步一步来的，理论上是安装成功的。我们可以用nginx自带命令首先测试配置文件

//测试配置文件

安装路径下的/nginx/sbin/nginx -t

//启动命令

安装路径下的/nginx/sbin/nginx

//停止命令

安装路径下的/nginx/sbin/nginx -s stop

或者 : nginx -s quit（**优雅的退出，会执行完内部相关操作才会退出，推荐使用这个命令**）

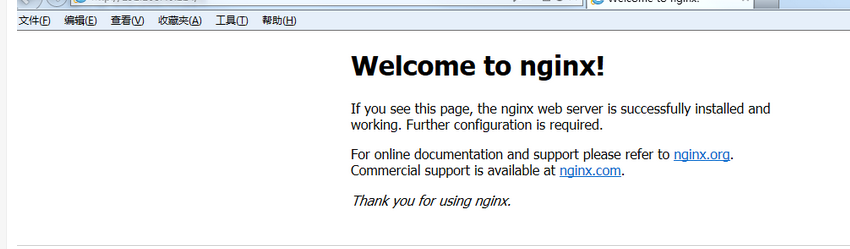
//热加载

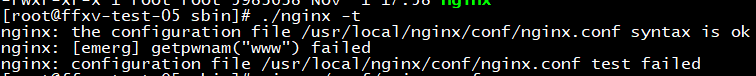
安装路径下的/nginx/sbin/nginx -s reload

**修改完nginx相关配置以后，执行此命令，即可实现nginx的平滑启动，无需重启，很方便。如果你经历过每次服务器更新，都要杀死对应进程，然后利用脚本重启服务器，查看日志是否报错等一系列流程，你看见这个命令就会会心一笑^\_^。。**

* **检验nginx是否真正安装启动成功**

Nginx 默认启动端口是80，我们在浏览器上输入你服务器的ip（**http协议，可以不输入端口号，因为我们知道http默认端口好就是80**）,如果你看到如下界面，恭喜你安装并且启动成功，你已经向使用Nginx的道路上迈出了一大步。



注：如果报错为：

则把/usr/local/nginx/conf/nginx.conf配置文件中的第一行#user nobody的注释去掉，然后重新启动nginx即可。

## Nginx的配置文件

实现Nginx负载均衡、正反向代理等功能都是通过修改其配置文件来实现的，所以可见其配置文件的重要性。在此只做简单的配置（项目所涉及）介绍，其他复杂功能实现，请自行Google其参数意义。

配置文件路径(如果你是一路默认安装)：/usr/local/nginx/conf/**nginx.conf，如下为Nginx的配置文件。**

########### 每个指令必须有分号结束。#################

#user administrator administrators; #配置用户或者组，默认为nobody nobody。

user nobody;

#4个进程数 一般与机器的cpu个数相同 最大效率的利用每个cpu

worker\_processes 4;

worker\_cpu\_affinity 0001 0010 0100 1000; #每个进程对应分配每个cpu

events {

worker\_connections 65535; #最大连接数，默认为512

}

http {

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，默认为text/plain

keepalive\_timeout 65; #连接超时时间，默认为75s，可以在http，server，location块。

upstream mysvr {

server1 127.0.0.1:7878;

server2 127.0.0.1:7878;

……….

}

**注：这里可以配置多个server的iP与端口，也是负载均衡的详细配置，如果要修改跳转地址，就是改此处。Mysvr只是个代号的存在，可以任意起名，当然鼓励见名知意。**

error\_page 404 https://www.baidu.com; #错误页

server {

listen 4545; #监听端口

server\_name 127.0.0.1; #监听地址

location ~\*^.+$ { #请求的url过滤，正则匹配，~为区分大小写，~\*为不区分大小写

proxy\_pass http://mysvr; #**请求转向mysvr 定义的服务器列表**

**注：这里实现了引用上面配置的mysvr模块，就实现负载均衡，如果负载均衡的是https协议，直接填写https均可。**

ssl\_certificate \*.crt;

ssl\_certificate\_key \*.key;

**注：上述参数是为Nignx开启https需要配置的参数，也是我们开始安装的依赖相关。就是指明协议需要的证书相关文件，而如果不需要付费申请证书的话，可以用Linux自带的openssl命令自行生成与签发，达到的效果与付费签发个人认为没有什么区别，就是别的浏览器会不信任你自己所生成的证书，从而弹出警告。**

}

}

}

上边所述涉及的负载均衡、还有为Nigxn开启配置https基本上就是项目上所涉及的关键点，也达到了需求的效果。

本身Nignx涉及到的关键点非常多，这里只是对它的简单应用，如果需要更为复杂的实现（比如资源静态分离、集群部署），需要进一步深究，暂无探索，如有兴趣的同学可以自行探索~。红色字体都是我在安装部署过程中查阅相关资料自己遇到的一些问题以及心得和浅薄的理解，如有错误以及不妥之处，欢迎指出修改。