nginx笔记：

### 1、静态HTTP服务器

首先，Nginx是一个HTTP服务器，可以将服务器上的静态文件（如HTML、图片）通过HTTP协议展现给客户端。

配置：

**[plain]** [view plain](https://blog.csdn.net/zhongguozhichuang/article/details/52816887) [copy](https://blog.csdn.net/zhongguozhichuang/article/details/52816887)

1. server {
2. listen80; # 端口号
3. location / {
4. root /usr/share/nginx/html; # 静态文件路径
5. }
6. }

### 2、反向代理服务器

什么是反向代理？

客户端本来可以直接通过HTTP协议访问某网站应用服务器，网站管理员可以在中间加上一个Nginx，客户端请求Nginx，Nginx请求应用服务器，然后将结果返回给客户端，此时Nginx就是反向代理服务器。

配置：

**[html]** [view plain](https://blog.csdn.net/zhongguozhichuang/article/details/52816887) [copy](https://blog.csdn.net/zhongguozhichuang/article/details/52816887)

1. server {
2. listen80;
3. location / {
4. proxy\_pass http://192.168.20.1:8080; # 应用服务器HTTP地址
5. }
6. }

### 3、负载均衡

当网站访问量非常大，网站[站长](http://zz.2cto.com/)开心赚钱的同时，也摊上事儿了。因为网站越来越慢，一台服务器已经不够用了。于是将同一个应用部署在多台服务器上，将大量用户的请求分配给多台机器处理。同时带来的好处是，其中一台服务器万一挂了，只要还有其他服务器正常运行，就不会影响用户使用。

Nginx可以通过反向代理来实现负载均衡。

**[html]** [view plain](https://blog.csdn.net/zhongguozhichuang/article/details/52816887) [copy](https://blog.csdn.net/zhongguozhichuang/article/details/52816887)

1. upstream myapp {
2. server192.168.20.1:8080; # 应用服务器1
3. server192.168.20.2:8080; # 应用服务器2
4. }
5. server {
6. listen80;
7. location / {
8. proxy\_pass http://myapp;
9. }
10. }

以上配置会将请求轮询分配到应用服务器，也就是一个客户端的多次请求，有可能会由多台不同的服务器处理。可以通过ip-hash的方式，根据客户端ip地址的hash值将请求分配给固定的某一个服务器处理。

配置：

**[html]** [view plain](https://blog.csdn.net/zhongguozhichuang/article/details/52816887) [copy](https://blog.csdn.net/zhongguozhichuang/article/details/52816887)

1. upstream myapp {
2. ip\_hash; # 根据客户端IP地址Hash值将请求分配给固定的一个服务器处理
3. server192.168.20.1:8080;
4. server192.168.20.2:8080;
5. }
6. server {
7. listen80;
8. location / {
9. proxy\_pass http://myapp;
10. }
11. }

另外，服务器的硬件配置可能有好有差，想把大部分请求分配给好的服务器，把少量请求分配给差的服务器，可以通过weight来控制。

配置：

**[html]** [view plain](https://blog.csdn.net/zhongguozhichuang/article/details/52816887) [copy](https://blog.csdn.net/zhongguozhichuang/article/details/52816887)

1. upstream myapp {
2. server192.168.20.1:8080weight=3; # 该服务器处理3/4请求
3. server192.168.20.2:8080; # weight默认为1，该服务器处理1/4请求
4. }
5. server {
6. listen80;
7. location / {
8. proxy\_pass http://myapp;
9. }
10. }

### 4、虚拟主机

有的网站访问量大，需要负载均衡。然而并不是所有网站都如此出色，有的网站，由于访问量太小，需要节省成本，将多个网站部署在同一台服务器上。

例如将www.aaa.com和www.bbb.com两个网站部署在同一台服务器上，两个域名解析到同一个IP地址，但是用户通过两个域名却可以打开两个完全不同的网站，互相不影响，就像访问两个服务器一样，所以叫两个虚拟主机。

配置：

**[html]** [view plain](https://blog.csdn.net/zhongguozhichuang/article/details/52816887) [copy](https://blog.csdn.net/zhongguozhichuang/article/details/52816887)

1. server {
2. listen80default\_server;
3. server\_name \_;
4. return444; # 过滤其他域名的请求，返回444状态码
5. }
6. server {
7. listen80;
8. server\_name www.aaa.com; # www.aaa.com域名
9. location / {
10. proxy\_pass http://localhost:8080; # 对应端口号8080
11. }
12. }
13. server {
14. listen80;
15. server\_name www.bbb.com; # www.bbb.com域名
16. location / {
17. proxy\_pass http://localhost:8081; # 对应端口号8081
18. }
19. }

在服务器8080和8081分别开了一个应用，客户端通过不同的域名访问，根据server\_name可以反向代理到对应的应用服务器。

虚拟主机的原理是通过HTTP请求头中的Host是否匹配server\_name来实现的，有兴趣的同学可以研究一下HTTP协议。

另外，server\_name配置还可以过滤有人恶意将某些域名指向你的主机服务器。

5、FastCGI

Nginx本身不支持[PHP](http://www.2cto.com/kf/web/php/)等语言，但是它可以通过FastCGI来将请求扔给某些语言或框架处理（例如PHP、[Python](http://www.2cto.com/kf/web/Python/)、Perl）。

**[html]** [view plain](https://blog.csdn.net/zhongguozhichuang/article/details/52816887) [copy](https://blog.csdn.net/zhongguozhichuang/article/details/52816887)

1. server {
2. listen80;
3. location ~ \.php$ {
4. include fastcgi\_params;
5. fastcgi\_param SCRIPT\_FILENAME /PHP文件路径$fastcgi\_script\_name; # PHP文件路径
6. fastcgi\_pass127.0.0.1:9000; # PHP-FPM地址和端口号
7. # 另一种方式：fastcgi\_pass unix:/var/run/php5-fpm.sock;
8. }
9. }

配置中将.php结尾的请求通过FashCGI交给PHP-FPM处理，PHP-FPM是PHP的一个FastCGI管理器。有

反向代理的解释：

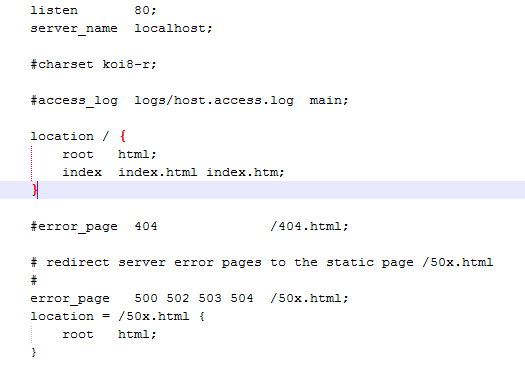
这里讲得很直白。反向代理方式实际上就是一台负责转发的代理服务器，貌似充当了真正服务器的功能，但实际上并不是，代理服务器只是充当了转发的作用，并且从真正的服务器那里取得返回的数据。这样说，其实nginx完成的就是这样的工作。我们让nginx监听一个端口，譬如80端口，但实际上我们转发给在8080端口的tomcat，由它来处理真正的请求，当请求完成后，tomcat返回，但数据此时没直接返回，而是直接给nginx，由nginx进行返回，这里，我们会以为是nginx进行了处理，但实际上进行处理的是tomcat。

说到上面的方式，也许很多人又会想起来，这样可以把静态文件交由nginx来进行处理。对，很多用到nginx的地方都是作为静态伺服器，这样可以方便缓存那些静态文件，比如CSS，JS，html，htm等文件。

闲话就不多说了，我们直接来看看nginx怎么使用。

nginx的nginx.conf文件的部分解释

2）我们可以看到nginx文件夹内有一个conf文件夹，其中有好几个文件，其他先不管，我们打开nginx.conf，可以看到一段：



  这段代码在server里面，相当于一个代理服务器，当然可以配置多个。

下面我们仔细来分析一下：

listen：表示当前的代理服务器监听的端口，默认的是监听80端口。注意，如果我们配置了多个server，这个listen要配置不一样，不然就不能确定转到哪里去了。

server\_name：表示监听到之后需要转到哪里去，这时我们直接转到本地，这时是直接到nginx文件夹内。

location：表示匹配的路径，这时配置了/表示所有请求都被匹配到这里

root：里面配置了root这时表示当匹配这个请求的路径时，将会在这个文件夹内寻找相应的文件，这里对我们之后的静态文件伺服很有用。

index：当没有指定主页时，默认会选择这个指定的文件，它可以有多个，并按顺序来加载，如果第一个不存在，则找第二个，依此类推。

下面的error\_page是代表错误的页面，这里我们暂时不用，先不管它。

那我们知道了具体的配置了，怎么让它访问localhost时转到tomcat时。实际上就修改两个地方：

使用nginx访问两个tomcat的方法，可以访问这个网页：

https://blog.csdn.net/u010921373/article/details/71038509

LInux:

weget http://nginx.org/download/nginx-1.15.2.tar.gz

tar -zxvf nginx-1.15.2.tar.gz

cd nginx-1.15.2.tar.gz

yum install -y gcc-c++

yum install -y pcre pcre-devel

yum install -y zlib zlib-devel

yum install -y openssl openssl-devel

./configure --prefix=/app/nginx \

--with-pcre \

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

--with-http\_realip\_module \

--with-http\_addition\_module \

--with-http\_ssl\_module \

--with-file-aio \

--with-debug \

执行上述操作会生成MakeFile

编译安装:

make && make install