nginx1.21.6测试与使用；nginx下载与安装。nginx作为高性能web服务器配置详解，比较重要的一个参数epoll。nginx静态资源服务器；nginx配置反向代理；nginx配置负载均衡。测试nginx反向代理到Apache（httpd）和Tomcat。

文末会提供一张nginx学习的思维导图。

正文

初次进入nginx官网，映入眼帘。给人的整体感觉就是简洁干净，一眼就能get到自己需要的资源。

nginx官网：http://nginx.org

nginx官方下载地址：http://nginx.org/en/download.html

在接下来的某些测试中，你可能看到使用的是root用户或者是nginx普通用户。使用root用户是为了方便演示，实际工作中一般你用的更多的是普通用户，一般只有管理员才有使用root用户的权限。

一、nginx快速安装

注意：nginx默认安装设置的sever是localhost，监听端口是80。

每一个代码块中都有详细注解进行解释，参考官方文档然后进行的翻译，并根据实际情况进行优化调整。

Windows版本的nginx官方文档地址：http://nginx.org/en/docs/windows.html

1、Windows下安装nginx

1.1、解压安装

```
#以管理员身份运行CMD窗口，进入D盘
d:
#新建work目录
mkdir work
#切换至work目录解压nginx
unzip nginx-1.21.6.zip
```

1.2、启动nginx

```
#进入nginx目录
cd work\nginx-1.21.6
#启动nginx服务
start nginx
```

1.3、查看nginx服务

nginx.conf文件中的参数：worker_processes 设置参数值为1，限制只能运行一个工作进程。

```
#查看nginx资源占用相关信息
d:\work\nginx-1.21.6>tasklist /fi "imagename eq nginx.exe"

映像名称                       PID 会话名              会话#       内存使用
========================= ======== ================ =========== ============
nginx.exe                     4108 Console                    6     11,232 K
nginx.exe                     4144 Console                    6     11,552 K
```

1.4、配置日志以及默认首页

在nginx解压目录中nginx-1.21.6\conf目录下打开nginx.conf配置文件，可以根据需求进行配置。此步骤下不做详细讲解，在接下来的配置说明进行详细讲解。

```
#配置日志与根目录首页
access_log   logs/site.log;
root         D:/web/html;
```

1.5、Windows下nginx命令管理

nginx在Windows中的一些常用使用命令，在Linux中一样可以使用stop、quit、reload命令管理nginx服务。

- nginx -s stop：快速关闭服务；
- nginx -s quit：优雅的关闭服务；
- nginx -s reload：改变配置，启动一个新的工作进程配置，优雅地关闭旧的工作进程；
- nginx -s reopen：重新打开日志文件；
- nginx -t -c nginx.conf ：检测nginx语法配置。

```
nginx -s stop   #快速关闭服务
nginx -s quit   #优雅的关闭服务
nginx -s reload #改变配置，启动一个新的工作进程配置，优雅地关闭旧的工作进程
nginx -s reopen #重新打开日志文件
nginx -t -c conf\nginx.conf #检测nginx语法配置
nginx: the configuration file D:\work\nginx-1.21.6/conf\nginx.conf syntax is ok
nginx: configuration file D:\work\nginx-1.21.6/conf\nginx.conf test is
successful
```

2、Linux下安装nginx

请先了解Linux中的基本权限和sudo权限。例如：我个人被管理的主机有两台：

- IP：192.168.245.146
- IP：192.168.245.147

但是，我只给其中一台147特定的管理权限。权限写的越具体，权限范围越小。测试使用sudoers文件管理普通用户：

```
# set other   测试授予所有或者某一个被管理主机拥有某些特定的权限，仅供参考
#nginx ALL=(ALL) /sbin/shutdown -r now
nginx 192.168.245.131=(ALL) /usr/local/nginx/sbin/nginx
nginx 192.168.245.147=(ALL) /usr/local/nginx-t/sbin/nginx
        实际管理，请先了解Linux权限知识和sudo权限
```

测试的主机为147，虽然给了131主机管理权限，但131并没部署，所以位于/usr/local/nginx/sbin/nginx文件是无法管理nginx服务的。

仅供参考，应该以实际工作场景为准，关于Linux中权限知识可以自己进行测试。

支持的Linux发行版也很丰富，同样可以参考官方文档，下载Debian系列，Ubuntu系列，Redhat系列以及Centos系列等等。部分制作成了表格形式，便于参考。

nginx支持操作系统（列出部分）	版本	支持平台
RHEL/CentOS	7.4+	x86_64, ppc64le, aarch64/arm64
RHEL/CentOS	8.x	x86_64, aarch64/arm64, s390x
Debian	10.x “buster”	x86_64, i386, aarch64/arm64
Debian	11.x “bullseye”	x86_64, aarch64/arm64
Ubuntu	18.04 “bionic”	x86_64, aarch64/arm64
Ubuntu	18.04 “bionic”	x86_64, aarch64/arm64, s390x
Ubuntu	21.10 “impish”	x86_64, aarch64/arm64

Linux下安装nginx，官网给出了便捷方式yum源、apt源等等：http://nginx.org/en/linux_packages.html

个人演示使用下载的源码包。Redhat7系列以及Centos7系列安装nginx-1.21.6，目前最新版本。可以使用nginx官方提供的yum源进行安装，或者使用wget命令进行下载安装。个人给出使用tar包（源码包）安装方式，下载到本机然后上传至虚拟机搭建的Linux环境Centos7.5服务器上。

2.1、安装依赖环境

安装需要的依赖环境，记住普通用户需要使用sudo提权，root用户则不需要。

```
[root@localhost ~]# yum install -y gcc pcre pcre-devel openssl openssl-devel gd gd-devel zlib-devel yum-utils
[nginx@localhost ~]$ sudo yum install -y gcc pcre pcre-devel openssl openssl-devel gd gd-devel zlib-devel yum-utils
```

2.2、解压安装nginx

配置可以参考nginx的官方文档，很详细。

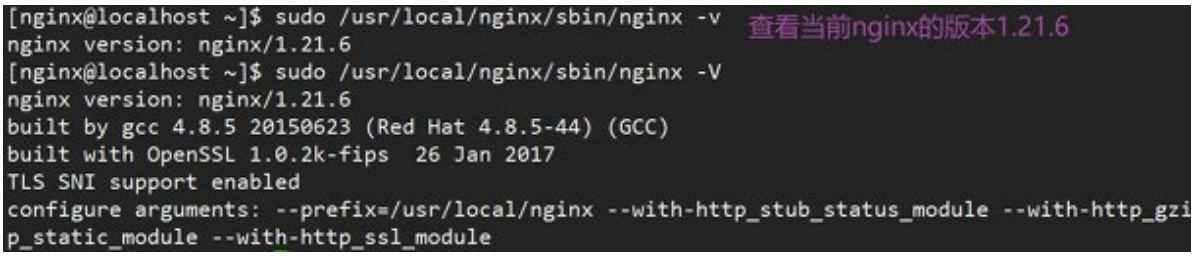
http://nginx.org/en/docs/configure.html

解压tar包，编译指定路径。不指定安装路径，默认安装到/usr/local/nginx，源码包默认安装位置。配置--prefix参数，指定安装路径以及需要的模块(module)，使用make && make install命令编译并安装。

```
#01、解压tar包
tar -zxvf nginx-1.21.6.tar.gz
#02、编译指定路径，不指定一样默认安装到/usr/local,源码包默认安装位置
cd nginx-1.21.6/
#03、配置，--prefix指定安装路径以及需要的模块(module)
./configure --prefix=/usr/local/nginx-t --with-http_stub_status_module --with-
http_gzip_static_module --with-http_ssl_module
#04、编译并安装
make && make install
```

查看nginx的版本：

- 参数-v：nginx -v 命令查看nginx中间件的版本；
- 参数-V：nginx -V命令查看nginx版本以及系统使用GCC版本、OpenSSL版本和配置的--prefix参数。




```
[nginx@localhost ~]$ sudo /usr/local/nginx/sbin/nginx -v
nginx version: nginx/1.21.6
[nginx@localhost ~]$ sudo /usr/local/nginx/sbin/nginx -V
nginx version: nginx/1.21.6
built by gcc 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)
built with OpenSSL 1.0.2k-fips  26 Jan 2017
TLS SNI support enabled
configure arguments: --prefix=/usr/local/nginx --with-http_stub_status_module --
with-http_gzip_static_module --with-http_ssl_module
```

2.3、管理nginx服务

安装nginx后的目录/usr/local/nginx/，使用ls以及ll命令查看安装后的文件。然后以绝对路径方式启动nginx服务： sudo /usr/local/nginx/sbin/nginx，在测试环境root用户下无需加sudo提权。新建用户，使用root用户身份权限新建。改变nginx安装目录所有者和所属组，赋予给nginx用户，此时登录nginx用户也可进行管理。

如何区分你使用的是超级管理用户root还是普通用户。

- #：带有#前缀符号则是超级管理员用户；
- $：带有$前缀符号则是普通用户。

```
#新建用户,使用root用户身份权限新建
[root@localhost ~]# useradd nginx #新建nginx用户
[root@localhost ~]# passwd nginx  #修改密码
#仅供参考，应该以实际应用场景为准。
[root@localhost ~]# chown -R root:nginx /usr/local/nginx/
[root@localhost nginx]# ls /usr/local/nginx/
client_body_temp    conf    fastcgi_temp    html    logs
proxy_temp  sbin    scgi_temp   uwsgi_temp
#安装nginx后的目录
[root@localhost ~]# ls /usr/local/nginx/
conf  html  logs  sbin
```

授予普通用户nginx管理的权限。使用`visudo`或者`vim /etc/sudoers`，在文件末尾加上`nginx ALL=(ALL) /usr/local/nginx/sbin/nginx`。作用是给nginx用户使用nginx脚本命令的权限（使用sudo）。给用户的权限范围越精确，用户权限则越小。**在你赋予权限的时候，理应思考是否合理。**开个小玩笑，一不小心将服务器拱手让人了，哈哈。我经常在说的一句话，你能够将Linux的权限玩的明明白白，就已经领先很大一部分人。

```
# visudo
# vim /etc/sudoers
nginx ALL=(ALL) /usr/local/nginx/sbin/nginx
```

启动nginx服务。root身份则无需提权，以绝对路径形式启动服务。普通用户，则需要使用sudo权限提权管理nginx服务。sudo的用法，可以使用man帮助命令查看。简单的提一下，使用命令visudo添加普通用户，或者编辑/etc/sudoers文件加入普通用户可执行的命令。

```
#root身份启动nginx服务
[root@localhost ~]# /usr/local/nginx/sbin/nginx
#普通用户身份启动nginx服务
[nginx@localhost ~]$ sudo /usr/local/nginx/sbin/nginx
```

优雅的关闭nginx服务，实际上找的是nginx.pid文件中存储的pid号。可以通过cat查看/usr/local/nginx/logs/nginx.pid。在nginx.conf配置文件中去找到相应的设置，将#注释去掉：

```
pid        logs/nginx.pid;
[root@localhost ~]# cat /usr/local/nginx/logs/nginx.pid
```

```
[root@localhost ~]# cat /usr/local/nginx/logs/nginx.pid
2398
[root@localhost ~]# ps -ef | grep nginx
root       2395      1  0 18:17 ?        00:00:00 nginx: master process /usr/local/nginx-t/sb
in/nginx
nobody     2396   2395  0 18:17 ?        00:00:00 nginx: worker process
root       2398      1  0 18:18 ?        00:00:00 nginx: master process /usr/local/nginx/sbin
/nginx
```

优雅的关闭nginx服务，nginx -s quit

```
[root@localhost ~]# /usr/local/nginx/sbin/nginx -s quit
[nginx@localhost ~]$ sudo /usr/local/nginx/sbin/nginx -s quit
```

重载nginx服务，nginx -s reload

```
[root@localhost ~]# /usr/local/nginx/sbin/nginx -s reload
[nginx@localhost ~]$ sudo /usr/local/nginx/sbin/nginx -s reload
```

2.4、查看nginx进程

可以使用`ps`命令配合`grep`搜素命令查看nginx服务进程状态，然后查看启动后的nginx目录多出了**client_body_temp**、**fastcgi_temp**、**proxy_temp**、**scgi_temp**、**uwsgi_temp**模块。

```
[nginx@localhost ~]$ ps -aux | grep nginx
root       7355  0.0  0.0  45992  1136 ?        Ss    nginx: master process
/usr/local/nginx/sbin/nginx
nobody     7356  0.0  0.1  48528  1988 ?        S     nginx: worker process
root       7368  0.0  0.0 112720   972 pts/1    S+    grep --color=auto nginx
[nginx@localhost ~]$ ls /usr/local/nginx/
client_body_temp  conf  fastcgi_temp  html  logs  proxy_temp  sbin  scgi_temp
uwsgi_temp
```

2.5、验证nginx服务

使用netstat命令查看监听到的nginx服务，默认使用的是80端口，一般80是不对外开放的。为了演示，使用firewalld命令开启80端口，然后使用`firewall-cmd --reload`命令重载防火墙。

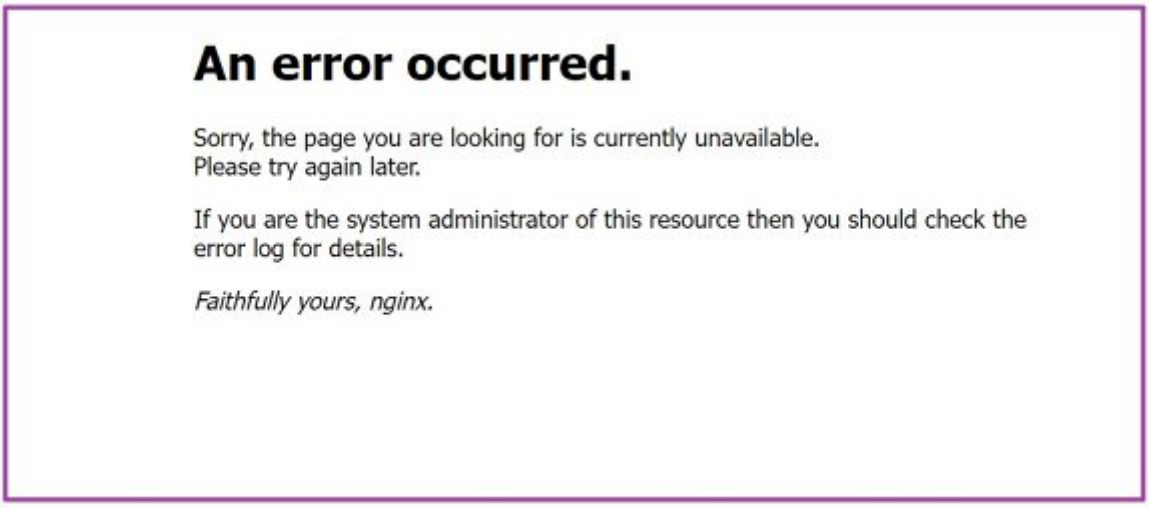
```
[root@localhost ~]# netstat -tlunp | grep nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
 7355/nginx: master
[root@localhost ~]# firewall-cmd --zone=public --add-port=80/tcp --permanent
success
[root@localhost ~]# firewall-cmd --zone=public --add-port=8081/tcp --permanent
success
[root@localhost ~]# firewall-cmd --reload
success
```

使用Chrome浏览器登录nginx，比如我个人使用虚拟环境搭建的，访问即可看到nginx服务启动完毕。即可看到，欢迎访问nginx服务。这个index.html页面默认存放在nginx的安装目录中html目录下。

```
http://192.168.245.147/
```

访问出现错误页面则为50x.html静态页面的内容，**直接在url后拼接访问50x.html文件即可进行测试。**

```
http://192.168.245.147/50x.html
```



至此，在Windows与Linux服务器上安装nginx服务以及服务的管理介绍完毕。在Linux版本中，我介绍的是比较详细的，这也是为了照顾初学者。

二、nginx做静态资源web服务器

1、nginx中常见的错误码

http消息	状态码	含义
已移动	http 301	请求的数据具有新的位置，并且永久更改。
已找到	http 302	请求的数据临时具有不同URI。
请参阅其它	http 303	可在另一URI下找到对请求的响应，并且使用get请求检索。
未修改	http 304	未按照预期修改文档。
使用代理	http 305	必须通过位置字段中提供的代理来访问请求的资源。
未使用	http 306	不再使用，但保留此代码以便将来使用。
无法找到网页	http 400	可以连接到web服务器，但由于web地址（URL）的问题无法找到网页。
网站拒绝显示此网页	http 403	可以连接到网站，但Internet Explorer没有访问网页文件的权限。
无法找到网页	http 404	可以连接到网站，但找不到网页。可能是网页暂不可用或者已被删除。
网站无法显示此网页	http 405	可以连接到网站，但网页内容无法下载到用户的计算机。可能是网页编码格式问题。
无法读取此网页格式	http 406	能从网站接收信息，但Internet Explorer无法识别格式，不能正确地显示消息。
网站忙，无法显示此网页	http 408或409	服务器显示网页时间过长，或对同一网页请求过多。
网页不复存在	http 410	可以连接到网站，但找不到网页。此错误为永久性的，而且由网站管理员打开。
网站无法显示该页面	http 500	正在访问的网站出现服务器问题，阻止此网页显示。正在维护或者交互程序出错。
未执行	http 501	没有将正在访问的网站设置为显示浏览器所请求的内容。
不支持的版本	http 505	该网站不支持浏览器用于请求网页的http协议。

2、hexo+nginx静态资源服务器

2.1、hexo的使用

Windows下首先安装node环境，然后使用npm再安装hexo模块。这里只介绍Windows下安装hexo环境：

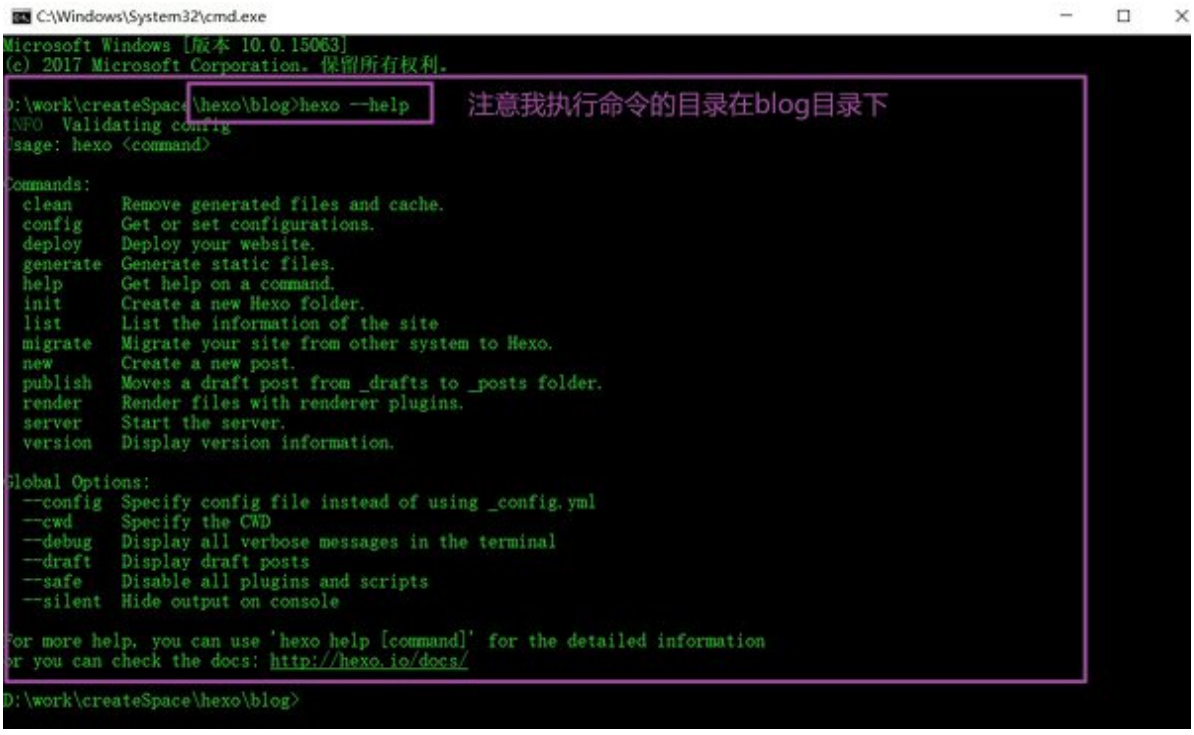
- 01、安装node环境；
- 02、在node环境下安装hexo，打开cmd命令窗口执行：`npm install -g hexo-cli`
- 03、继续在cmd窗口命令安装：`npm install hexo`

具体其它平台安装hexo可以参考官网中文文档：https://hexo.io/zh-cn/docs/

在node环境下安装hexo后生成的blog文件目录：

使用hexo命令，hexo new命令生成文件，hexo server命令启动服务，通过http://localhost:4000访问hexo。

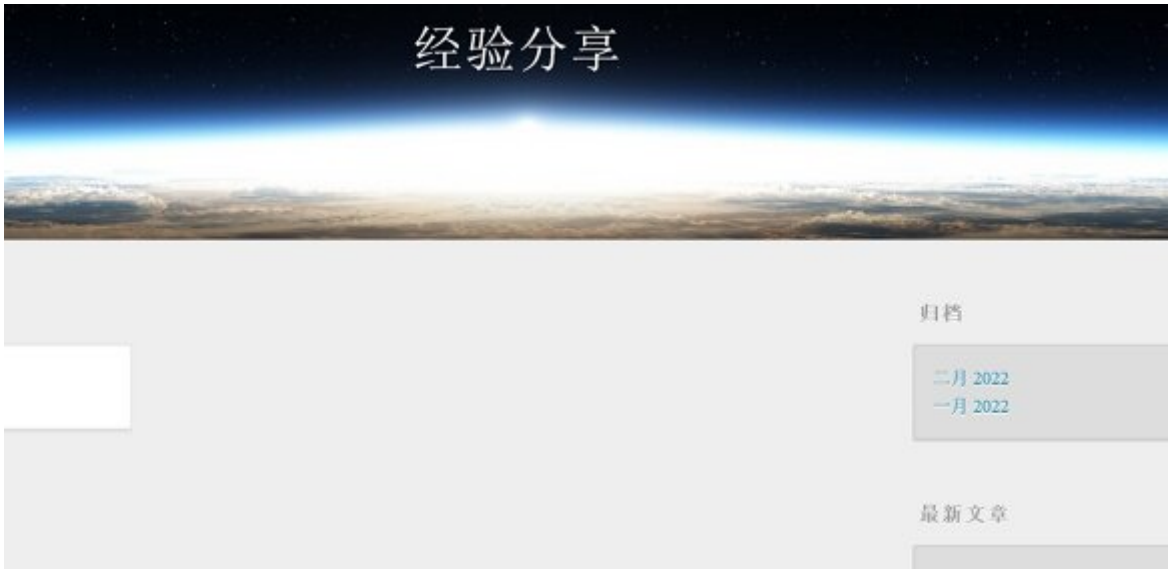
```
D:\work\createSpace\hexo\blog>hexo new "你要生成的md文件名"
hexo generate #生成静态文件
hexo server #启动服务
```



2.2、在nginx中访问

将hexo生成的静态html文件上传到虚拟环境中nginx服务器的html目录下，默认的设置的根目录和和首页配置不变。在虚拟机搭建的nginx静态资源服务器，并且使用了反向代理，代理了默认的80端口开启的nginx服务：

```
http://192.168.245.147:8081/archives/2022/02/
```



如果想看具体效果可以访问我在github上搭建的测试环境：

```
https://cnwangk.github.io/archives/2022/02/
```

你可以使用hexo、jekyl以及hugo去生成静态网页，然后部署到nginx服务器上。如果买了云服务器，可以利用起来。再入手一个域名，申请蓝色的幕布，然后进行备案使用https解析，nginx同样也是支持ssl（解析https协议）的。

我同时开启了两个nginx服务，使用其中一个反向代理另一个nginx服务。配置文件如下设置，**反向代理使用到关键字为proxy_pass**：

```
#在http模块中配置
http{
    upstream test {
        server 192.168.245.147;
    }
    server {
            location / {
                proxy_pass http://test;
                root   html;
                index  index.html index.htm;
            }
     }
}
```

查看nginx进程，发现有两个不同路径的进程，分别是nginx文件和nginx-t文件：

```
[nginx@localhost nginx-1.21.6]$ ps -aux | grep nginx
root      15241  0.0  0.0  46004  1132 ?        Ss    nginx: master process
/usr/local/nginx-t/sbin/nginx
nobody    15242  0.0  0.1  48528  2488 ?        S     nginx: worker process
root      15274  0.0  0.0  45992  1136 ?        Ss    nginx: master process
/usr/local/nginx/sbin/nginx
nobody    15275  0.0  0.1  48532  2240 ?        S     nginx: worker process
root      15302  0.0  0.0 112724   968 pts/2    S+    grep --color=auto nginx
```

进行测试演示。监听的端口，Redhat7系列使用firewall-cmd命令启用了80和8081端口。

```
[nginx@localhost nginx-1.21.6]$ netstat -tlunp | grep nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
 15274/nginx: master
tcp        0      0 0.0.0.0:8081            0.0.0.0:*               LISTEN
 15241/nginx: master
```

三、nginx代理服务

谈到代理，能联想到的有生活中的代理商，还有平时想翻山越海其实也是利用代理服务。国内某大厂原创game虽然火不久，但代理出了名，估计大家也猜出来了。在我们的nginx中间件中一样可以实现正向代理和反向代理，反向代理恰恰是nginx服务的重要功能之一。通过图形化可以更直观的理解代理。

图1-1：正向代理

nginx服务配置正向代理的3个指令：

- resolver：用于指定DNS服务器的IP地址。
- resolver_timeout：用于设置DNS服务器域名解析超时时间。
- proxy_pass：用于设置代理协议，同时也是配置反向代理的指令。

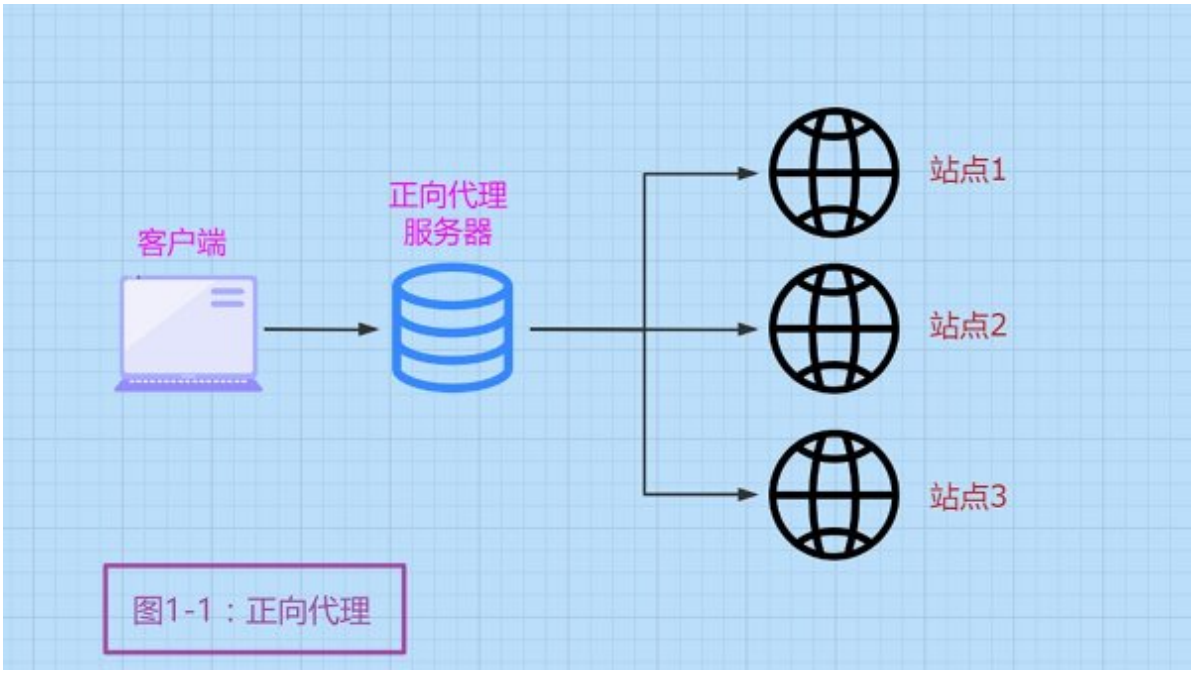
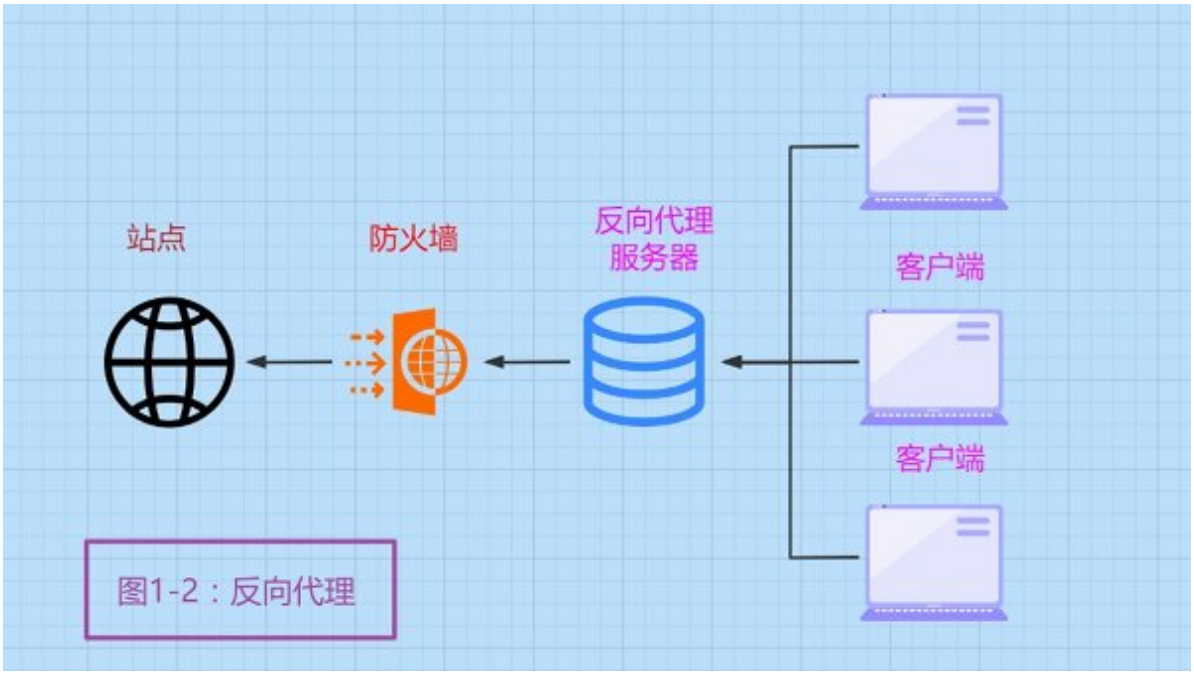


图1-2：反向代理

- proxy_pass：配置反向代理的主要参数，注意指明传输协议。
- proxy_hide_header：用于隐藏一些头域信息。
- proxy_pass_header：用于处理发送响应报文时接收一些date、server、x-accel头域信息。
- proxy_set_header：用于更改nginx服务器接收到客户端请求的请求头信息。

关于反向代理指令就介绍这几个，更多的可以参考ngx_http_proxy_module，nginx官网proxy模块。我也列出我在工作中实际应用到配置：

```
proxy_pass_header User-Agent;
proxy_set_header Host $http_host;
proxy_next_upstream error timeout invalid_header http_500 http_502 http_503;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto https;
```

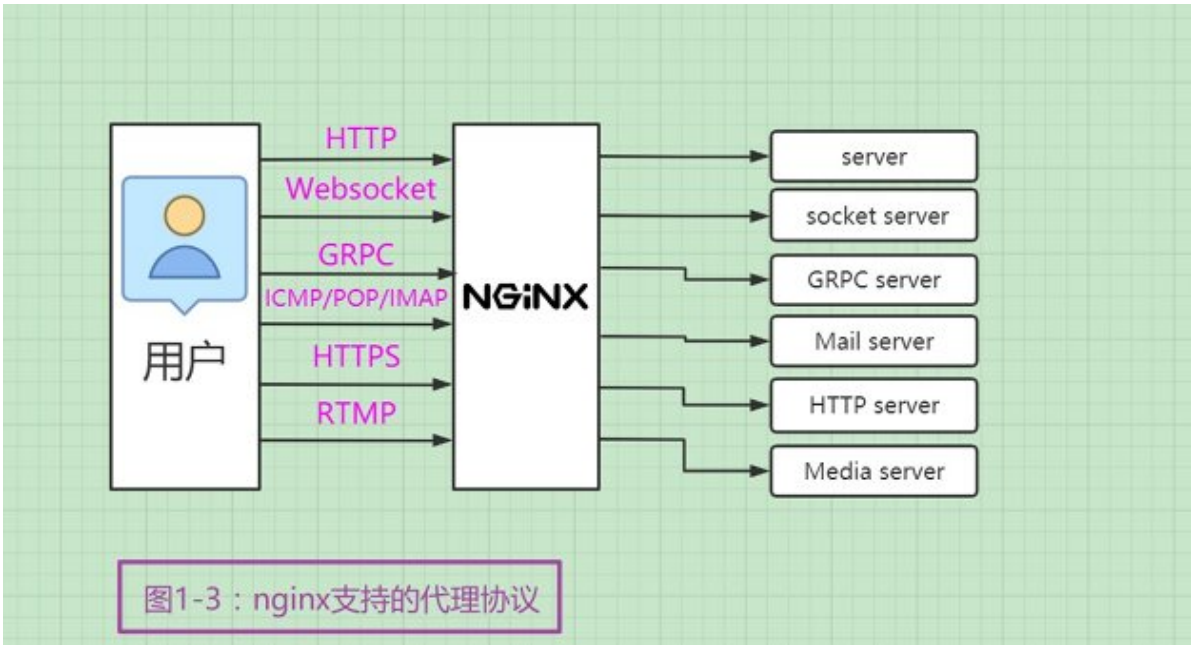


代理区别：形式上服务的对象不一样。

- 正向代理代理的对象是客户端，**为客户端服务**；
- 反向代理代理的对象是服务端，**为服务端服务。**

为了演示进行测试，开启了**nginx**服务、**httpd**服务以及**Tomcat**服务。

在使用nginx代理服时，看看nginx支持哪些代理协议，如图1-3：



主要演示工作中一些场景：

- nginx代理到nginx；
- nginx代理到Apache；
- nginx代理到Tomcat；

1、nginx目录结构简介

nginx目录作用：

- conf目录：主要存放nginx的配置文件，主要的控制文件。
- html目录：存放静态资源目录。

- logs目录：存放nginx生成的日志文件（包含错误日志）以及nginx.pid文件存放nginx分进程pid号。
- sbin目录：nginx服务脚本，需要使用root管理员身份管理服务，或者使用sudo提权。

```
[nginx@localhost ~]$ ls /usr/local/nginx
conf    html    logs    sbin
```

2、ngin.conf文件的介绍

初学者在配置nginx.conf文件中的参数时，往往会遇到语法错误，可以**使用nginx提供命令进行检测语法配置**：

```
[nginx@localhost ~]$ sudo /usr/local/nginx/sbin/nginx -t
[nginx@localhost ~]$ sudo /usr/local/nginx/sbin/nginx -t -c
/usr/local/nginx/conf/nginx.conf
```

语法配置正确，则会有以下提示：

```
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
```

2.1、用户与进程配置区

- user参数：用来配置用户以及用户组，如果配置为nobody代表不限制用户。
- worker_processes：worker_processes参数配置工作进程。可以根据CPU核心数配置，比如4核配置4个工作进程，提高并发。

```
#user  nobody;
worker_processes  1;
```

2.2、日志与pid配置区

- error_log：配置错误日志；
- pid：配置nginx存储的pid号，临时的，服务关闭就消失了。可以根据pid号去杀死进程。

```
#配置日志区
error_log  logs/error.log;
error_log  logs/error.log  notice;
error_log  logs/error.log  info;
#pid存储位置，可以根据pid号去杀死进程
pid        logs/nginx.pid;
```

2.3、events配置区

events事件配置区，配置全局的。worker_connections参数默认配置的1024，可以根据系统去优化设置最大的工作连接数。

```
#配置连接数
events {
    worker_connections  1024;
}
```

2.4、http模块

- 包含server模块，可以配置多个。
- 包含location模块，同样可以配置多个。
- 负载均衡upstream同样配置在http模块中，server模块之外。

```
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  0;
    keepalive_timeout  65;
    #配置负载均衡
    upstream tomcat {
        #可以是域名,或者是ip加端口
        server www.example.com;
        server 192.168.245.147:8888;
    }
 }
```

2.5、server模块

- listen：配置nginx服务监听端口，默认为80端口，可以根据实际需求更改；
- server_name：配置服务名，可以是IP地址也可以是域名；
- charset：配置字符集；
- access_log：访问服务接收的日志所在主要目录；
- location：包含location设置，主要有主页以及代理请求头等等参数配置。

```
 server {
        listen 8088;
        #listen       8443 ssl;
        #server_name  192.168.0.233;
        server_name 127.0.0.1;
        #charset koi8-r;
        access_log  logs/host.access.log  main;
        #读取根目录
        location / {
           proxy_pass http://test;
           #设置读取的目录
           root   html;
           index  index.html index.htm;
        }
        #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;
        }
```

2.6、location配置区，初次安装默认只有root配置根目录和index配置首页。proxy_pass是配置代理，我后面加的。

```
    #读取根目录
    location / {
          #配置代理
          proxy_pass http://192.168.245.233;
          #设置读取的目录
          root   html;
          index  index.html index.htm;
     }
```

2.7、配置多个conf文件

在复杂的场景下可能会配置多个conf文件，使用include关键字包含其它的配置文件。

```
#配置多个conf文件包含进来
include  conf/*.conf;
```

3、nginx反向代理配置

反向代理恰恰是nginx服务的重要功能之一，着重演示一下反向代理过程。其实配置参数很简单，使用proxy_pass即可配置反向代理，http://后面配置的可以是ip也可以是域名。

```
http{
    server{
        location / {
            proxy_pass http://192.168.245.233:88;
        }
    }
}
```

3.1、反向代理到Apache

3.1.1、在Redhat7系列直接使用yum命令安装Apache，通过rpm命令**验证是否安装httpd服务。**

```
[nginx@localhost ~]$ rpm -qa | grep httpd
httpd-2.4.6-97.el7.centos.4.x86_64
httpd-tools-2.4.6-97.el7.centos.4.x86_64
```

3.1.2、安装Apache服务

Redhat系列使用yum命令安装httpd。

```
$ sudo yum -y install httpd
# yum -y install httpd
```

httpd安装后的目录，主要配置文件存放在conf目录下：

- conf：httpd.conf配置文件目录；
- conf.d：其它配置文件，比如用户存储目录配置；
- logs：接收的日志文件access_log 、error_log；
- modules：代理、请求以及重写规则等模块。

```
[nginx@localhost ~]$ ls /etc/httpd/
conf  conf.d  conf.modules.d  logs  modules  run
[nginx@localhost ~]$ ls /etc/httpd/conf
httpd.conf  magic
```

使用yum命令安装后的命令脚本，使用whereis命令查看httpd，默认路径在/usr/sbin/httpd。

```
[nginx@localhost ~]$ whereis httpd
httpd: /usr/sbin/httpd /usr/lib64/httpd /etc/httpd /usr/share/httpd
/usr/share/man/man8/httpd.8.gz
[nginx@localhost ~]$ ll /usr/sbin/httpd
-rwxr-xr-x. 1 root root 523640  /usr/sbin/httpd
```

3.1.3、修改Apache服务监听的端口，修改Listen后面的参数为81端口进行测试。


```
# vim /etc/httpd/conf/httpd.conf
Listen 81
```

访问的html页面在`/usr/share/httpd/noindex`目录下：

```
[nginx@localhost ~]$ ls /usr/share/httpd/noindex/
css  images  index.html
```

3.1.4、反向代理Apache

配置负载均衡，**设置了参数weight权重。**在我们安装的nginx服务中进行配置代理，测试使用的nginx服务设置监听8081端口。并且给反向代理的Apache服务配置的权重为2，执行3次有两次会显示Apache服务页面。

```
http{
        upstream test {
            server 192.168.245.147:81 weight=2;
            server 192.168.245.147;
    }
        server{
          listen       8081;
          server_name  localhost;
          location / {
                proxy_pass http://test;
                root   html;
                index  index.html index.htm;
            }
       }
}
```

在Chrome浏览器访问。使用upstream配置了负载均衡，访问3次有两次定位Apache页面，一次定位hexo搭建静态页面：

```
http://192.168.245.147:8081
```

3.2、反向代理到Tomcat

开启Tomcat服务默认使用端口8080，可以根据实际情况修改。**加入到防火墙规则**

```
[root@localhost conf]# firewall-cmd --zone=public --add-port=8080/tcp --permanent
success
```

重载防火墙

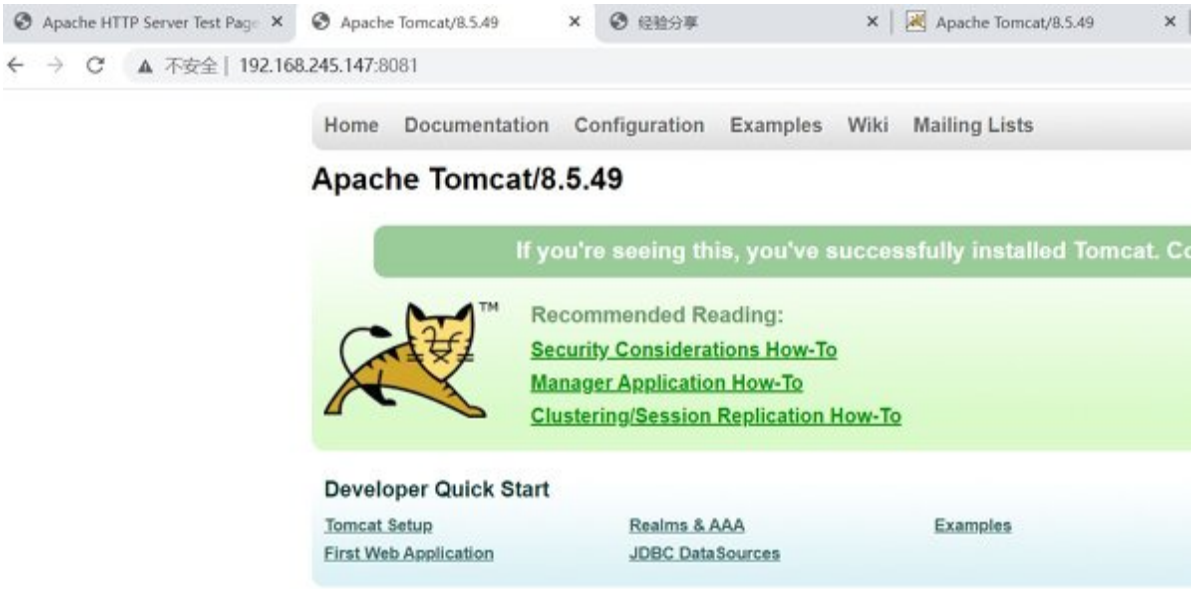
```
[root@localhost conf]# firewall-cmd --reload
success
```

原始启动tomcat服务默认server.xml配置的是8080端口，我进行了反向代理使用8081访问。**启动tomcat服务**

```
[root@localhost conf]# /usr/local/apache-tomcat-8.5.49/bin/catalina.sh run &
```

加入tomcat服务的ip地址到负载均衡。设置访问Apache服务的权重weight=2，访问两个站点3次，其中两次会出现Apache页面，第三次则会出现Apache Tomcat页面。

```
upstream test {
      server 192.168.245.147:81 weight=2;
      #server 192.168.245.147;
      server 192.168.245.147:8080;
  }
```



实际上你可以在nginx上再套一层nginx，nginx反向代理nginx，只是没有代理其它中间件来的那么直观。

四、nginx负载均衡

其实我在演示上面的反向代理过程中，就已经用到了负载均衡。在测试的过程中，**请加入需要的防火墙规则**，避免造成不必要的麻烦。

tips：server后面可以接ip，也可以接域名。

1、负载均衡的几种模式

1.1、负载均衡默认配置

默认的负载均衡设置，**采用轮询的形式**，权重是均衡的。如果想测试建议配置多个nginx监听服务，然后进行测试。非要问个为什么，那就是**nginx很轻很小**，**但是功能很强大**！

```
#默认负载均衡（轮询）
upstream proxy_demo1{
    server 192.168.245.233:8086;
    server 192.168.245.233:8087;
    server 192.168.245.233:8088;
}
```

1.2、负载均衡加权轮询

做5次刷新访问的页面测试，其中有3次会定位到设置权重为3的8087端口对应的ip上，剩余两次分别定位到8086和8088上。

```
#加权轮询负载均衡
upstream proxy_demo2{
    server 192.168.245.233:8086;
    server 192.168.245.233:8087 weight=3;
    server 192.168.245.233:8088;
}
```

1.3、负载均衡基于ip_hash

ip_hash配置很简单：http://nginx.org/en/docs/http/ngx_http_upstream_module.html#ip_hash

```
#配置语法
Syntax: ip_hash;    Default:    —       Context:    upstream
upstream backend {
    ip_hash;#加入参数即可
    server backend1.example.com;
    server backend2.example.com;
    server backend3.example.com down;
    server backend4.example.com;
}
```

ip_hash策略：是将前端的访问IP进行hash操作，然后根据hash结果将请求分配给不同的后端节点。可以将这种策略看成一种特殊的轮询策略，每个前端访问IP会固定访问一个后端节点。**优势**：避免考虑前端用户的session在后端多个节点上共享的问题。

```
#基于ip的hash
upstream proxy_demo3{
    ip_hash;
    server 192.168.245.233:8086;
    server 192.168.245.233:8087;
    server 192.168.245.233:8088;
}
```

1.4、负载均衡基于url的hash

url_hash策略和ip_hash类似，属于第三方扩展模块。不同点在于ip_hash策略是对前端访问IP进行hash操作；url_hash策略是对前端请求的url进行了hash操作。**url_hash优势**：如果后端有缓存服务器，他能够高缓存效率，同时解决session的问题。缺点是后端节点出现异常，不能自动排除此节点。说到web缓存，相信有不少web后端开发者对Squid服务器有所了解，**经典组合方式nginx缓存功能配合Squid服务。**

```
#基于url的hash
upstream proxy_demo4{
    #url_hash;
    hash $request_uri;
    server 192.168.245.233:8086;
    server 192.168.245.233:8087;
    server 192.168.245.233:8088;
}
```

2、nginx优化

nginx的高级配置，针对内核、cpu、网络连接以及事件驱动模型进行配置的思考：

- ipv4内核7个参数；
- cpu配置优化；
- 网络连接配置4个相关指令；
- 事件驱动模型8个指令。

3、nginx其它应用场景

nginx做缓存服务器

- nginx服务器基于proxy store的缓存机制；
- nginx服务器基于memcached的缓存机制；
- nginx服务器基于proxy cache的缓存机制。
- nginx与squid服务器组合配置。

感兴趣的可以对nginx的Rewrite功能、gzip模块；时间管理、内存管理以及工作进程进行深度学习。

文末提供一张nginx思维导图，这个链接可以直接访问，就不以大图形式展示了：

https://gitee.com/dywangk/img/raw/master/images/nginx%E5%85%A5%E9%97%A8%E5%88%B0%E5%AE%9E%E8%B7%B501.png

总结

以上就是本次nginx安装与配置的全部内容，希望能对你的工作与学习有所帮助。感觉写的好，就拿出你的一键三连。在公众号上更新的可能要快一点，公众号目前还在完善中。**能看到这里的，都是帅哥靓妹**。如果感觉总结的不到位，也希望能留下您宝贵的意见，我会在文章中进行调整优化。

MySQL系列文章：《**MySQL开发篇，存储引擎的选择真的很重要吗？**》已经上传至github和gitee仓库SQL-study。个人github仓库地址，一般会先更新PDF文件，然后再上传markdown文件。如果访问github太慢，可以使用gitee进行克隆。

作者：龙腾万里sky