Nginx安装

一、安装Nginx（Apache）

每一台storage server都需要安装Nginx。

1、下载文件

- 查看GitHub上面最新的nginx release 版本

  https://github.com/nginx/nginx/releases
- 下载各个版本的nginx的地址：

  http://nginx.org/download/
- 下载Nginx

```
wget http://nginx.org/download/nginx-1.15.6.tar.gz
```

- 上传nginx压缩包

```
yum install -y lrzsz
rz
```

通过rz命令上传文件：nginx-1.15.6.tar.gz

2、安装第三方软件

2.1、安装PCRE

PCRE(Perl Compatible Regular Expressions)是一个Perl库，包括 perl 兼容的正则表达式库。Nginx的http模块使用pcre来解析正则表达式，所以需要在linux上安装pcre库。

```
yum install -y pcre-devel
```

> 注：pcre-devel是使用pcre开发的一个二次开发库。Nginx也需要此库。

2.2、安装ZLIB

zlib库提供了很多种压缩和解压缩的方式，Nginx使用zlib对http包的内容进行gzip，所以需要在linux上安装zlib库。

```
yum install -y zlib-devel
```

2.3、安装OPENSSL

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用。

Nginx不仅支持http协议，还支持https（即在ssl协议上传输http），所以需要在linux安装openssl库。

```
yum install -y openssl-devel
```

3、解压缩

```
tar -xf nginx-1.15.6.tar.gz
```

4、执行configure配置

```
cd nginx-1.15.6/

./configure \
--prefix=/kkb/server/nginx \
--pid-path=/var/run/nginx/nginx.pid \
--lock-path=/var/lock/nginx.lock \
--error-log-path=/var/log/nginx/error.log \
--http-log-path=/var/log/nginx/access.log \
--http-client-body-temp-path=/var/temp/nginx/client \
--http-proxy-temp-path=/var/temp/nginx/proxy \
--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \
--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \
--http-scgi-temp-path=/var/temp/nginx/scgi \
--with-http_gzip_static_module
```

> **注意：**
>
> prefix=`/kkb/server/nginx` 中的 `/kkb/server/nginx` 指的是要安装的nginx的路径

5、创建nginx/client目录

```
mkdir -p /var/temp/nginx/client
```

6、创建临时目录

上面执行的configure命令，设置了一些配置参数，其中的一些参数指定的目录一定要存在。

```
mkdir /var/temp/nginx -p
```

7、编译安装

```
make && make install
```

8、启动Nginx

切换到nginx/bin目录

```
./nginx
```

六、Nginx附加资料

location配置详解

语法规则：

```
location [=|~|~*|^~]  /uri/  { … }
```

语法说明：

- = 开头表示精确匹配
- ^~ 开头表示uri以某个常规字符串开头，理解为匹配 url路径即可。nginx不对url做编码，因此请求为/static/20%/aa，可以被规则^~ /static/ /aa匹配到（注意是空格）。
- **~ 开头表示区分大小写的正则匹配**
- **~* 开头表示不区分大小写的正则匹配**
- **!~和!~*分别为区分大小写不匹配及不区分大小写不匹配 的正则**
- / 通用匹配，任何请求都会匹配到。

多个location配置的情况下匹配顺序：

- 首先匹配 =
- 其次匹配^~
- 其次是按文件中顺序的正则匹配
- 最后是交给 / 通用匹配
- 当有匹配成功时候，停止匹配，按当前匹配规则处理请求。

例子，有如下匹配规则：

```
location = / {
   #规则A
}
location = /login {
   #规则B
```

```
}
location ^~ /static/ {
   #规则C
}
location ~ \.(gif|jpg|png|js|css)$ {
   #规则D
}
location ~* \.png$ {
   #规则E
}
location !~ \.xhtml$ {
   #规则F
}
location !~* \.xhtml$ {
   #规则G
}
location / {
   #规则H
}
```

那么产生的效果如下：

```
访问根目录/， 比如http://localhost/ 将匹配规则A

访问 http://localhost/login 将匹配规则B，http://localhost/register 则匹配规则H

访问 http://localhost/static/a.html 将匹配规则C

访问 http://localhost/a.gif, http://localhost/b.jpg 将匹配规则D和规则E，但是规则D顺序优先，规则E不起作用，而 http://localhost/static/c.png 则优先匹配到 规则C

访问 http://localhost/a.PNG 则匹配规则E， 而不会匹配规则D，因为规则E不区分大小写。

访问 http://localhost/a.xhtml 不会匹配规则F和规则G，http://localhost/a.XHTML不会匹配规则G，因为不区分大小写。规则F，规则G属于排除法，符合匹配规则但是不会匹配到，所以想想看实际应用中哪里会用到。

访问 http://localhost/category/id/1111 则最终匹配到规则H，因为以上规则都不匹配，这个时候应该是nginx转发请求给后端应用服务器，比如FastCGI（php），tomcat（jsp），nginx作为方向代理服务器存在。
```

所以实际使用中，通常至少有三个匹配规则定义，如下：

```
#直接匹配网站根，通过域名访问网站首页比较频繁，使用这个会加速处理，官网如是说。
#这里是直接转发给后端应用服务器了，也可以是一个静态首页
# 第一个必选规则
location = / {
    proxy_pass http://tomcat:8080/index
}

# 第二个必选规则是处理静态文件请求，这是nginx作为http服务器的强项
# 有两种配置模式，目录匹配或后缀匹配,任选其一或搭配使用
location ^~ /static/ {
```

```
    root /webroot/static/;
}
location ~* \.(gif|jpg|jpeg|png|css|js|ico)$ {
    root /webroot/res/;
}

#第三个规则就是通用规则，用来转发动态请求到后端应用服务器
#非静态文件请求就默认是动态请求，自己根据实际把握
#毕竟目前的一些框架的流行，带.php,.jsp后缀的情况很少了
location / {
    proxy_pass http://tomcat:8080/
}
```

rewrite语法

- **语法命令：**

```
rewrite     <regex>     <replacement>     [flag]

关键字          正则          替代内容           flag标记
```

- **命令解释：**
  - 关键字：重写语法关键字
  - 正则：perl兼容正则表达式语句进行规则匹配
  - 替代内容：将正则匹配的内容替换成replacement
  - flag标记：rewrite支持的flag标记
- **flag标记说明：**

```
* last  #本条规则匹配完成后，继续向下匹配新的location URI规则

* break  #本条规则匹配完成即终止，不再匹配后面的任何规则

* redirect  #返回302临时重定向，浏览器地址会显示跳转后的URL地址

* permanent  #返回301永久重定向，浏览器地址栏会显示跳转后的URL地址
```