kill命令格式：

**kill** 是向进程发送信号的命令。

**Nginx的信号**

1）、TERM、INT 快速关闭

2）、QUIT从容关闭

3）、HUP平滑重启，重新加载配置文件

4）、USR1 重新打开日志文件

5）、USR2 平滑升级可执行程序

**KILL 9 强制终止，直接杀**

上课前，回顾下昨天的作业：

location /rout {

rewrite ^/rout/(.\*)  /static/$1.html break;

root   html/;

index  index.html index.htm;

}

curl linux中发http请求，查看返回值

**Nginx管理虚拟主机**

虚拟主机使用的是特殊的软硬件技术，它把一台运行在因特网上的服务器主机分成一台台“虚拟”的主机，每台虚拟主机都可以是一个独立的网站，可以具有独立的域名，具有完整的Intemet服务器功能（WWW、FTP、Email等），同一台主机上的虚拟主机之间是完全独立的。从网站访问者来看，每一台虚拟主机和一台独立的主机完全一样。

利用虚拟主机，不用为每个要运行的网站提供一台单独的Nginx服务器或单独运行一组Nginx进程。虚拟主机提供了在同一台服务器、同一组Nginx进程上运行多个网站的功能。

**基于域名的虚拟主机**

https://common.cnblogs.com/images/copycode.gifserver {

#监听端口 80

listen 80;

#监听域名abc.com;

server\_name abc.com;

location / {

# 相对路径，相对nginx根目录。也可写成绝对路径

root abc;

# 默认跳转到index.html页面

index index.html;

}

}

**基于端口的虚拟主机配置**

https://common.cnblogs.com/images/copycode.gifserver {

listen 2022;

server\_name abc.com;

location / {

root /home;

index index.html;

}

}

**基于IP地址虚拟主机配置**

server {

listen 80;

server\_name 192.168.197.142;

location / {

root ip;

index index.html;

}

}

**Location语法规则**

**1.1 Location规则**

语法规则： location [=|~|~\*|^~] /uri/ {… }

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。

|  |  |
| --- | --- |
| 符号 | 含义 |
| = | = 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!~\* | !~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 |
| / | 用户所使用的代理（一般为浏览器） |
| $http\_x\_forwarded\_for | 可以记录客户端IP，通过代理服务器来记录客户端的ip地址 |
| $http\_referer | 可以记录用户是从哪个链接访问过来的 |

**Location解析过程**

server {

listen       80;

server\_name  location.enjoy.com;

location /a {

        rewrite ^/  /a.html break;

        root   html/static/;

}

location /b/a {

        rewrite ^/  /b.html break;

        root   html/static/;

}

location /b/d/a {

        rewrite ^/  /d.html break;

        root   html/static/;

}

location ^~/b/c/a {

        rewrite ^/  /d.html break;

        root   html/static/;

}

location ~ /b/d {

        rewrite ^/  /c.html break;

        root   html/static/;

}

location ~ /b/d/a {

        rewrite ^/  /a.html break;

        root   html/static/;

}

}

**Rewrite的使用**

**rewirte的 语法**

rewrite regex replacement [flag];

rewrite ^/rout/(.\*)  $1.html break;

**负载均衡配置**

1、轮询（默认）

upstream nginx {

server 172.17.0.4:8081;

server 172.17.0.5:8081;

}

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。

2、weight

指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。down 暂时不参与负载

例如：

upstream nginx {

server 172.17.0.4:8081 weight=2;

server 172.17.0.5:8081 weight=1;

}

2、ip\_hash

每个请求按访问ip的hash结果分配，这样同一客户端的请求总是发往同一个后端服务器，可以解决session的问题。

例如：

upstream nginx {

ip\_hash;

server 172.17.0.4:8081;

server 172.17.0.5:8081;

}

**实战**

**配置：**

**172.17.0.2作为代理nginx**

**172.17.0.3作为静态服务器，读html文件**

**172.17.0.4为后台服务器1，提供web服务**

**172.17.0.5为后台服务器2，提供web服务**

**172.17.0.2作为反向代理，有以下配置：**

**后台代理：**

upstream nginx {

#       ip\_hash;

server 172.17.0.4:8081 weight=2;

server 172.17.0.5:8081 weight=1;

}

server {

        listen       80;

        server\_name  www.enjoy.com;

        location /proxy {

                proxy\_pass <http://172.17.0.4:8081/nginx/;>

        }

        location /nginx {

                proxy\_pass http://nginx;

        }

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

        location = /50x.html {

            root   html;

        }

    }

**html代理：**

location /static {

proxy\_pass <http://rewrite.enjoy.com/;>

}

**location和rewrite进阶：**

**nginx运行阶段：** **rewrite 阶段、access 阶段以及 content 阶段**

**不按代码顺序执行，是按阶段执行，顺序如下：**

**先执行命中的所有rewrite层指令（下面的set），再执行access，再执行content（下面的echo）**

**语法：**

**location  = / {**

**set $a 32;**

**echo $a;**

**set $a 64;**

**echo $a;**

**}**