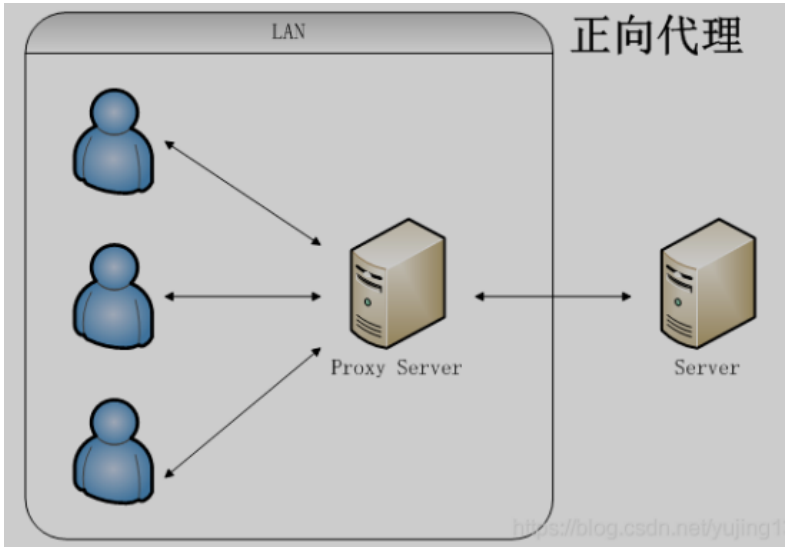
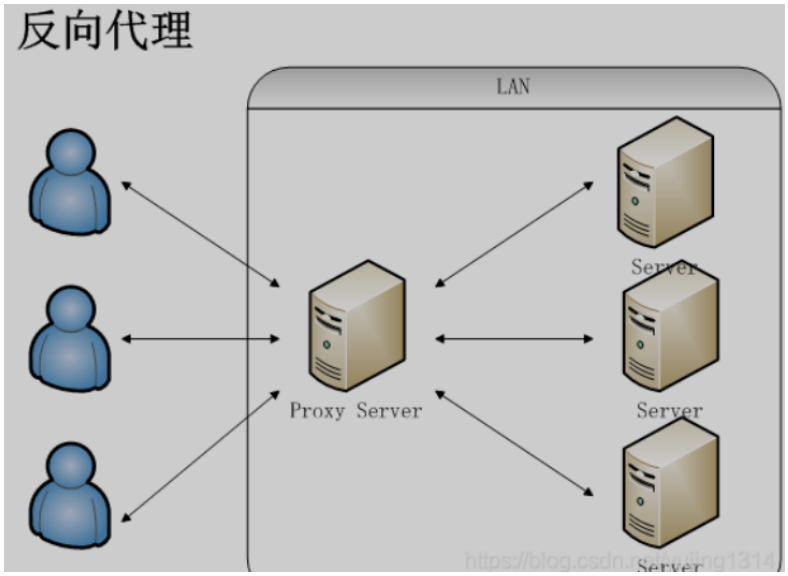
Nginx学习

- *@author : sunshijiang*
- *@Wechat :1294777193*
- ☆**发音**： [ˈendʒɪnks]
- 基本**概念:**
  - Nginx：是一款自由的、开源的、高性能的 HTTP 服务器和反向代理服务器；同时也是一个 IMAP、POP3、SMTP 代理服务器。
  - 代理：就是一个代表，一个渠道；涉及到2个角色：被代理角色 + 目标角色
    - 概念：被代理角色通过这个代理访问目标角色完成一些任务的过程称为代理操作过程；(eg.如同生活中的专卖店，客人到 adidas 专卖店买了一双鞋，这个专卖店就是代理，被代理角色就是 adidas 厂家，目标角色就是用户)
    - 正向代理:
      - 概念:
        - 局域网中的电脑用户想要直接访问网络是不可行的，只能通过代理服务器来访问，这种代理服务就被称为正向代理。



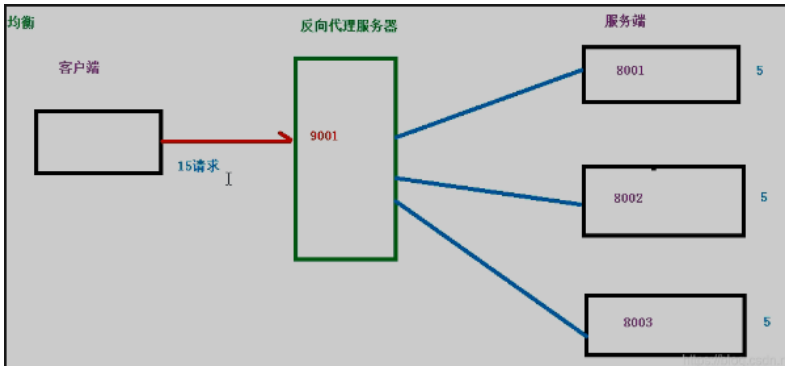
      - 原理:
        - 正向代理，"它代理的是客户端"，是一个位于客户端和原始服务器（Origin Server）之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标（原始服务器）。然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。
      - 最大特性:
        - （1）客户端非常明确要访问的服务器地址；但服务器只清楚请求来自哪个代理服务器，而不清楚来自哪个具体的客户端。
        - （2）正向代理模式屏蔽或者隐藏了真实客户端信息。

- 👇（3）客户端和代理是同一个环境，客户端对代理这个事是有感知的。
- 👉用途：
  - 👇（1）访问原来无法访问的资源，如：Google。
  - 👇（2）可以做缓存，加速访问资源。
  - 👇（3）对客户访问授权，上网进行认证。
  - 👇（4）代理可以记录用户访问记录，对外隐藏用户信息。
- 👁反向代理：
  - 👉概念：
    - 👇客户端无法感知代理，因为客户端访问网络不需要配置，只要把请求发送到反向代理服务器，由反向代理服务器去选择目标服务器获取数据，然后再返回到客户端。



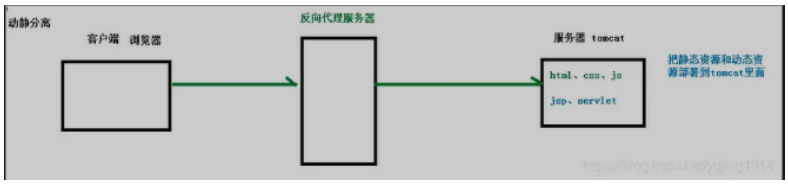
  - 👉原理：
    - 👇反向代理，"它代理的是服务端"，主要用于服务器集群分布式部署的情况下，反向代理隐藏了服务器的信息。
  - 👉最大特性：
    - 👇（1）多个客户端给服务器发送的请求，Nginx 代理服务器接收到之后，按照一定的规则分发给了后端的业务处理服务器进行处理了。此时请求的来源也就是客户端是明确的，但是请求具体由哪台服务器处理的并不明确了，Nginx 扮演的就是一个反向代理角色。
    - 👇（2）服务端和代理是同一个环境，客户端对代理这个事是无感知的，访问者并不知道自己访问的是一个代理。
    - 👇（3）反向代理模式暴露的是代理服务器地址，屏蔽或隐藏了真实服务器的信息。
    - 👇（4）客户端不需要任何配置就可以访问。
  - 👉用途：

    - 👇（1）保证内网的安全，通常将反向代理作为公网访问地址，Web 服务器是内网。
    - 👇（2）负载均衡，通过反向代理服务器来优化网站的负载。
- 👁负载量：客户端发送的、Nginx 反向代理服务器接收到的请求数量。
- 👁均衡规则：请求数量（负载量）按照一定的规则进行分发到不同的服务器处理的规则。
- 👁负载均衡：
    - 👉概念：
        - 👇增加服务器的数量，构建集群，将请求量按照规则分发到各个服务器上的过程，这个过程就叫做负载均衡。（假设有 15 个请求发送到代理服务器，那么由代理服务器根据服务器数量，平均分配，每个服务器处理 5 个请求，这个按照规则分发的过程就是负载均衡）

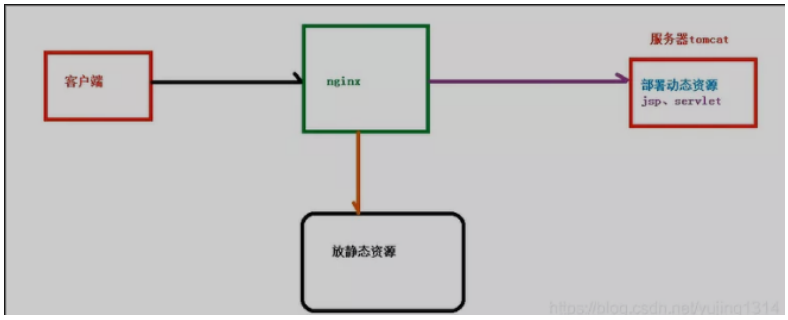


    - 👉原理：
        - 👇将原来请求集中到单个服务器的情况改为请求分发到多个服务器。
        - 👇负载均衡是针对反向代理的一种负载量分发规则。
    - 👉调度算法：
        - 👇（1）权重轮询（默认）：
            - 🔍接收到的请求按照顺序逐一分配到不同的后端服务器，即使在使用过程中，某一台后端服务器宕机，Nginx 会自动将该服务器剔除出队列，请求受理情况不会受到任何影响。
        - 👇（2）ip_hash：
            - 🔍每个请求按照发起客户端的 ip 的 hash 结果进行匹配，这样的算法下一个固定 ip 地址的客户端总会访问到同一个后端服务器，这也在一定程度上解决了集群部署环境下 Session 共享的问题。
        - 👇（3）fair：
            - 🔍智能调整调度算法，动态的根据后端服务器的请求处理到响应的时间进行均衡分配。响应时间短处理效率高的服务器分配到请求的概率高，响应时间长处理效率低的服务器分配到的请求少，它是结合了前两者的优点的一种调度算法。
        - 👇（4）url_hash：

        - 🔍按照访问的 URL 的 hash 结果分配请求，每个请求的 URL 会指向后端固定的某个服务器，可以在 Nginx 作为静态服务器的情况下提高缓存效率。
    - 👁动静分离：
        - 👉概念：
            - 👇为了加快网站的解析速度，可以把动态请求和静态请求交给不同的服务器来解析，加快解析的速度，降低由单个服务器的压力。（不能理解成只是单纯地把动态页面和 静态页面物理分离。严格意义上说应该是动态请求跟静态请求分开，**可以理解成使用 Nginx 处理静态页面，Tomcat 处理动态页面**）
        - 👉原理：
            - 👇动静分离之前的状态：



            - 👇动静分离之后的状态：



        - 👉分类：
            - 👇纯粹把静态文件独立成单独的域名，放在独立的服务器上，也是目前主流推崇的方案
            - 👇动态跟静态文件混合在一起发布，通过 nginx 来分开
- 🖨**特点**：
    - 占用内存少，并发能力强（专为性能优化而开发）
- 📝常用命令：
    - *使用nginx操作命令，必须进入 nginx 目录中才可以：cd /usr/local/nginx/sbin*
    - ❗查看版本：
        - ./nginx -v
    - ❗查看Nginx的状态：
        - ps -ef | grep nginx
    - ❗启动Nginx：
        - ./nginx
    - ❗关闭Nginx：

    - ./nginx -s stop　或
    - ./nginx -s quit
  - ❗比较野蛮直接杀死进程：
    - ./killall nginx
  - ❗重新加载Nginx配置：
    - ./nginx -s reload
- ⚙配置文件：
  - 👁配置文件位置：/usr/local/nginx/conf/nginx.conf
  - 👁包含三部分内容：
    - 👉全局块：
      - 👇从配置文件开始到 events 块之间，主要设置一些影响 Nginx 服务器整体运行的配置指令。
      - 👇并发处理服务的配置，值越大，可以支持的并发处理量越多，但是会受到硬件、软件等设备的制约。

```
# 第一部分：全局块：配置服务器整体运行的配置指令
#运行用户，默认即是nginx，可以不进行设置
user  nginx;
#Nginx进程，一般设置为和CPU核数一样 -=> 处理并发数的配置
worker_processes  1;
#错误日志存放目录
error_log  /var/log/nginx/error.log warn;
#进程pid存放位置
pid        /var/run/nginx.pid;
```

    - 👉events块：
      - 👇影响 Nginx 服务器与用户的网络连接，常用的设置包括是否开启对多 workprocess 下的网络连接进行序列化，是否允许同时接收多个网络连接等等。

```
# 第二部分：events 块：影响 Nginx 服务器与用户的网络连接
events {
    worker_connections  1024; # 单个后台进程的最大并发数
}
```

    - 👉HTTP块：（包含http块自身和server块）
      - 👇诸如反向代理和负载均衡都在此配置。

```
# 第三部分：http 块，还包含两部分：http 全局块和server 块
http {
    include       /etc/nginx/mime.types;   #文件扩展名与类型映射表
    default_type  application/octet-stream;  #默认文件类型
    #设置日志模式
    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  /var/log/nginx/access.log  main;   #nginx访问日志存
放位置

    sendfile        on;   #开启高效传输模式
    #tcp_nopush     on;    #减少网络报文段的数量

    keepalive_timeout  65;  #保持连接的时间，也叫超时时间

    #gzip  on;  #开启gzip压缩

    include /etc/nginx/conf.d/*.conf; #包含的子配置项位置和文件
[server快]
```

- 👇include子文件配置项default.conf内容：

```
server {
    listen       80;   #配置监听端口
    server_name  localhost;  //配置域名

    #charset koi8-r;
    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;     #服务默认启动目录
        index  index.html index.htm;    #默认访问文件
    }

    #error_page  404              /404.html;   # 配置404页面

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;   #错误状态码的显示页
面，配置后需要重启
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    # ~ \.php$ 是正则表达式，匹配以.php结尾的所有文件
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1; // 反向代理
    #}

    # pass the PHP scripts to FastCGI server listening on
127.0.0.1:9000
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME
/scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}
```

- 🌍配置实例：（一）反向代理：
  - 🗑反向代理指令：proxy_pass

```
server{
        listen 80; // 浏览器端访问地址的端口号
        # server_name 可以是网址域名; // 浏览器端访问的地址
        # server_name 192.168.191.34; // 浏览器端访问的地址
        # location / {
                # proxy_pass http://123.com; // nginx 反向代理的地
址，可以是域名
                # proxy_pass 192.168.23.45.8080; // nginx 反向代理的
地址
        # };

        # ~ /edu/ 是正则表达式，区分大小写，匹配的是edu目录
        # ~* /edu/ 是正则表达式，不区分大小写，匹配的是edu目录
        # = /edu/ 是正则表达式，严格匹配，匹配的是edu目录
        location ~ /edu/ {
             proxy_pass 192.168.23.45.8081; // nginx 反向代理的地址
        };

        # ~ /vod/ 是正则表达式，匹配的是vod目录
        location ~ /vod/ {
             proxy_pass 192.168.23.45.8082; // nginx 反向代理的地址
        }
}
```

- 🗑️其他反向代理指令：
  - 👉proxy_set_header：
    - 在将客户端请求发送给后端服务器之前，更改来自客户端的请求头信息
  - 👉proxy_connect_timeout:
    - 配置Nginx与后端代理服务器尝试建立连接的超时时间
  - 👉proxy_read_timeout：
    - 配置Nginx向后端服务器组发出read请求后，等待相应的超时时间
  - 👉proxy_send_timeout：
    - 配置Nginx向后端服务器组发出write请求后，等待相应的超时时间
  - 👉proxy_redirect：
    - 用于修改后端服务器返回的响应头中的Location和Refresh
  - 👉关于proxy代理指令的详细信息： www.nginx.cn/doc/mail/ma…
- 🌍配置实例：（二）配置负载均衡：
  - 🗑️实现负载均衡主要在http块和server块里面配置：
    - 👉http块配置如下：

```
http {
        # ... 省略http块默认带有的配置
        upstream myserver {
            server 115.28.52.63:8080 ;
            server 115.28.52.63:8081 ;
        }
}
```

    - 👉server块的配置如下：

```
aerver {
        location / {
        ... 省略location里面默认带有的配置
        proxy_pass http://myserver; //myserver是上面创建的myserver
服务名
        proxy_connect_timeout 10;
    }
}
```

- 🗑️**不同调度算法的配置方式：**
  - 👉权重：（weight 代表权,重默认为 1,权重越高被分配的客户端越多）

```
upstream server_pool {
  server 192.168.5.21 weight=10; // weight 代表权,重默认为 1,权重越高
被分配的客户端越多
  server 192.168.5.22 weight=10;
}
```

  - 👉ip_hash：

```
upstream server_pool {
  ip_hash;
  server 192.168.5.21:80;
  server 192.168.5.22:80;
}
```

  - 👉fair（第三方）：

```
upstream server_pool {
  server 192.168.5.21:80;
  server 192.168.5.22:80;
  fair;
}
```

- 🌍配置实例：（三）配置动静分离：
  - 🗑️在 liunx 系统中准备静态资源，用于进行访问，在nginx里面，新建一个文件夹 data
  - 🗑️主要在server块的配置如下：

```
aerver {
        server 80;
    server_name 192.168.17.129;

    #chartset koi8-r;
    #access_log log/host.access.log main;

    # 通过nginx实现静态资源跳转
    location /www/ {
        # /data/是静态资源目录
        root /data/;
        index index.html index.htm;
    }

    # 通过nginx实现静态资源跳转
    location /image/ {
        # /data/是静态资源目录
        root /data/;
        # autoindex on是列出访问目录，可以不加
        autoindex on;
    }
}
```

以上内容整理于 幕布文档