

# 江 蘇 大 學

JIANGSU UNIVERSITY

## 计算机网络实验报告



实验名称: Web 服务的配置与管理

学院名称: 计算机科学与通信工程学院

专业班级: 物联网工程 2303

学生姓名: 邱佳亮

学生学号: 3230611072

教师姓名: 李峰

报告日期: 2024/11/21

# 目录

目录 .....	1
1 Nginx 的配置 .....	2
1.1 实验目的 .....	2
1.2 实验思路 .....	2
1.3 实验步骤 .....	2
1.3.1 安装配置 Nginx .....	2
1.3.2 配置负载均衡 .....	3
1.3.3 配置运行多个网站 .....	5
1.3.4 配置访问控制 .....	6
1.4 思考与提高 .....	7
2 总结和收获 .....	8

# 1 Nginx 的配置

## 1.1 实验目的

- (1) 了解 Nginx 的工作原理，掌握 Nginx 的安装与配置方法；
- (2) 掌握在一台 nginx 服务器上配置运行多个网站的方法；
- (3) 了解 Nginx 负载均衡的实现原理，能够利用 Nginx 进行负载均衡；
- (4) 了解 Nginx 访问控制的实现原理，能够利用 Nginx 进行访问控制。

## 1.2 实验思路

- (1) 安装配置 Nginx，并验证已经安装成功；
- (2) 在一台 nginx 服务器上基于不同端口配置运行多个网站；
- (3) 在 nginx 服务器配置负载均衡，并测试其效果；
- (4) 在 nginx 服务器配置访问控制，并测试其效果。

## 1.3 实验步骤

### 1.3.1 安装配置 Nginx

访问 nginx 官网，下载 nginx：

nginx: 下载			 <a href="#">中文</a> <a href="#">新闻</a> <a href="#">关于</a> <a href="#">下载nginx</a> <a href="#">安全</a> <a href="#">nginx文档</a>
主线版本			
CHANGES	<a href="#">nginx-1.25.1</a>	<a href="#">pgp</a>	
	<a href="#">nginx/Windows-1.25.1</a>	<a href="#">pgp</a>	
稳定版本			
CHANGES-1.24	<a href="#">nginx-1.24.0</a>	<a href="#">pgp</a>	
	<a href="#">nginx/Windows-1.24.0</a>	<a href="#">pgp</a>	

图 1 下载 nginx

解压：

名称	修改日期	类型
conf	2024/11/22 8:29	文件夹
contrib	2024/11/22 8:29	文件夹
docs	2024/11/22 8:29	文件夹
html	2024/11/22 8:29	文件夹
logs	2023/6/13 20:01	文件夹
temp	2023/6/13 20:01	文件夹
nginx.exe	2024/11/22 8:29	应用程序

图 2 解压

在/conf下找到配置文件并修改端口为 81:

```
#gzip on;

server {
    listen      81;
    server_name localhost;

    #charset koi8-r;

    #access_log logs/host.access.log main;

    location / {
        root    html;
        index   index.html index.htm;
    }
}
```

图 3 修改端口

通过命令提示符进入 nginx 文件夹，启动 nginx 服务:

```
D:\nginx-1.25.1>start nginx

D:\nginx-1.25.1>
```

图 4 启动 nginx

访问 localhost:81，显示访问成功:



图 5 访问成功

### 1.3.2 配置负载均衡

配置负载均衡，在 http 块中定义后端服务器，并设置负载均衡规则，设置了两个负载服务器 127.0.0.1:8080 和 127.0.0.1:8081，其 root 目录分别为 backend1 和 backend2:

```

}
server {
    listen 8080;

    location / {
        root D:/nginx-1.25.1/backend1;
        index index.html;
    }
}

server {
    listen 8081;

    location / {
        root D:/nginx-1.25.1/backend2;
        index index.html;
    }
}

server {
    listen 81;
    server_name localhost;

    #charset koi8-r;

    #access_log logs/host.access.log main;

    location / {
        proxy_pass http://backend;
    }

    #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;
    }
}

```

图 6 配置负载均衡

通过命令行测试 nginx 配置文件是否正确，并重新加载 nginx：

```

D:\nginx-1.25.1>nginx -t
nginx: the configuration file D:\nginx-1.25.1\conf\nginx.conf syntax is ok
nginx: configuration file D:\nginx-1.25.1\conf\nginx.conf test is successful

D:\nginx-1.25.1>nginx -s reload

```

图 7 重新加载 nginx

在两个服务器的根目录下创建 index.html 文件：

```

D: > nginx-1.25.1 > backend1 > index.html > h1
1  <h1>Response from Server 8080</h1>

```

图 8 index 文件

通过命令行启动两个服务：

```

D:\nginx-1.25.1>python -m http.server 8080

D:\nginx-1.25.1>python -m http.server 8081

```

图 9 启动两个服务

此时访问 localhost:81，两个后端服务会交替响应：

```
D:\nginx-1.25.1>curl http://localhost:81
<h1>Response from Server 8080</h1>
D:\nginx-1.25.1>curl http://localhost:81
<h1>Response from Server 8081</h1>
D:\nginx-1.25.1>curl http://localhost:81
<h1>Response from Server 8080</h1>
D:\nginx-1.25.1>curl http://localhost:81
<h1>Response from Server 8081</h1>
D:\nginx-1.25.1>
```

图 10 交替响应

### 1.3.3 配置运行多个网站

创建两个网站的 root 目录:

site1	2024/11/22 9:00	文件夹
site2	2024/11/22 9:01	文件夹

图 11 创建目录

在目录下创建网站的 index:

```
D: > nginx-1.25.1 > site1 > index.html > ...
1 <h1>Welcome to Site 1</h1>
2 |
```

图 12 index

修改配置文件，为两个网站添加 server 配置:

```
#gzip on;
server {
    listen 8080;
    server_name localhost;

    location / {
        root D:/nginx-1.25.1/site1;
        index index.html;
    }
}

server {
    listen 8081;
    server_name localhost;

    location / {
        root D:/nginx-1.25.1/site2;
        index index.html;
    }
}
```

图 13 修改配置

检查配置文件并重启服务：

```
D:\nginx-1.25.1>nginx -t
nginx: the configuration file D:\nginx-1.25.1/conf/nginx.conf syntax is ok
nginx: configuration file D:\nginx-1.25.1/conf/nginx.conf test is successful

D:\nginx-1.25.1>nginx -s reload
```

图 14 重启服务

可以访问两个网站的内容：

```
D:\nginx-1.25.1>curl http://localhost:8080
<h1>Welcome to Site 1</h1>

D:\nginx-1.25.1>curl http://localhost:8081
<h1>Welcome to Site 2</h1>
```

图 15 访问网站

### 1.3.4 配置访问控制

编辑配置文件，允许特定 ip 地址访问并拒绝其他 ip 地址访问：

```
location / {
    root    html;
    index  index.html index.htm;

    allow 192.168.1.100;
    allow 127.0.0.1;

    deny all;
}
```

图 16 修改配置文件

检查配置文件并重启服务：

```
D:\nginx-1.25.1>nginx -t
nginx: the configuration file D:\nginx-1.25.1/conf/nginx.conf syntax is ok
nginx: configuration file D:\nginx-1.25.1/conf/nginx.conf test is successful

D:\nginx-1.25.1>nginx -s reload
```

图 17 重启服务

此时非特定 ip 无法访问 nginx 服务：

```
D:\nginx-1.25.1>curl -H "X-Forwarded-For: 192.168.1.100" http://localhost:81
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.25.1</center>
</body>
</html>
```

图 18 访问失败

## 1.4 思考与提高

跨域问题是指在 B/S 架构中，当浏览器发起请求的协议、域名或端口与当前页面的地址不一致时，受到浏览器同源策略的限制而导致请求被阻止的现象。浏览器的同源策略是一种安全机制，旨在防止恶意网站窃取用户数据或发送未经授权的请求。Nginx 可以通过配置 CORS 支持或使用反向代理转发请求解决跨域问题。

在 conf 文件中添加 cors 配置：

```
server {
    listen      81;
    server_name localhost;

    #charset koi8-r;

    #access_log logs/host.access.log main;

    location / {
        root    html;
        index   frontend.html;
    }
    location /api {
        proxy_pass http://127.0.0.1:5000; # 转发到后端服务
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;

        # 添加 CORS 响应头
        add_header Access-Control-Allow-Origin *;
        add_header Access-Control-Allow-Methods "GET, POST, OPTIONS";
        add_header Access-Control-Allow-Headers "Content-Type, Authorization";

        # OPTIONS 请求直接返回
        if ($request_method = 'OPTIONS') {
            return 204;
        }
    }
}
```

图 19 添加配置

前端文件如下：

```
D: > nginx-1.25.1 > html > frontend.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>CORS Test</title>
6      <script>
7          async function fetchData() {
8              try {
9                  const response = await fetch('http://localhost:81/api/data'); // 跨域请求
10                 const data = await response.json();
11                 console.log(data);
12             } catch (error) {
13                 console.error('Error:', error);
14             }
15         }
16     </script>
17 </head>
18 <body>
19     <h1>CORS Test</h1>
20     <button onclick="fetchData()">Fetch Data</button>
21 </body>
22 </html>
```

图 20 前端文件

访问页面，点击 fetch data 按钮进行跨域请求：



# CORS Test

[Fetch Data](#)

图 21 跨域请求

查看浏览器控制台网络部分，发现收到后端返回数据：

▼ 常规	
请求 URL:	http://localhost:81/api/data
Request Method:	GET
状态代码:	200 OK
远程地址:	127.0.0.1:81
引用站点策略:	strict-origin-when-cross-origin

图 22 数据包

查看响应标头，看到 cors 运行成功：

标头	预览	响应	发起程序	计时	Cookie
原始					
Accept:	*/*				
Accept-Encoding:	gzip, deflate, br, zstd				
Accept-Language:	zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6				
Connection:	keep-alive				
Cookie:	username=localhost-8888=2[1]0[10:1729747884]23:username=localhost-8888[2]12:eyJ1c2VybmFzZSI6IjYzkyNWQ3YjIwNjU0NWYzYTNkZDk2ZWJjYzU5MmE1OCIsCiJ1YVW1IjogIjFub255bW91cy8QaGlsb3Bocm9zeW5lliwgIl6IG51bGx9Jm91dD7265ee736ef5b85cde0a608c0aed8c1e25c9a1e9886e4830b6b2a6f,_xsrf=2[2]19f2c58[17478e3f4ea17c9b4eb7aebbeb2075]1				
Host:	localhost:81				
Referer:	http://localhost:81/				
Sec-Ch-Ua:	"Microsoft Edge";v="131", "Chromium";v="131", "Not_A_Brand";v="24"				
Sec-Ch-Ua-Mobile:	?0				
Sec-Ch-Ua-Platform:	"Windows"				
Sec-Fetch-Dest:	empty				
Sec-Fetch-Mode:	cors				
Sec-Fetch-Site:	same-origin				
User-Agent:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36 Edg/131.0.0.0				

图 23 响应标头

## 2 总结和收获

通过本次实验，我深入了解了 Nginx 的安装、配置及其多种功能的实现过程。首先，成功完成了 Nginx 的安装与启动，并通过修改配置文件调整服务端口和站点路径，验证了服务的正常运行。接着，在一台 Nginx 服务器上配置了基于不同端口运行多个网站，验证了多站点同时运行的效果。同时，通过设置负载均衡，将用户请求分发至不同的后端服务器，实现了轮询交替响应的功能。此外，还配置了访问控制策略，通过允许特定 IP 地址访问并拒绝其他 IP 地址的方式，验证了访问权限的有效性。通过本次实验，不仅掌握了 Nginx 的基本使用方法，还对其灵活的配置和强大的功能有了更深入的理解，为后续实际应用奠定了良好的基础。