

# 极简Nginx入门教程

塔内pc网络部

培训时间：2020.0x.xx

# 目录

一

安装Nginx

二

nginx文件结构简要介绍

三

Nginx搭建简单服务器

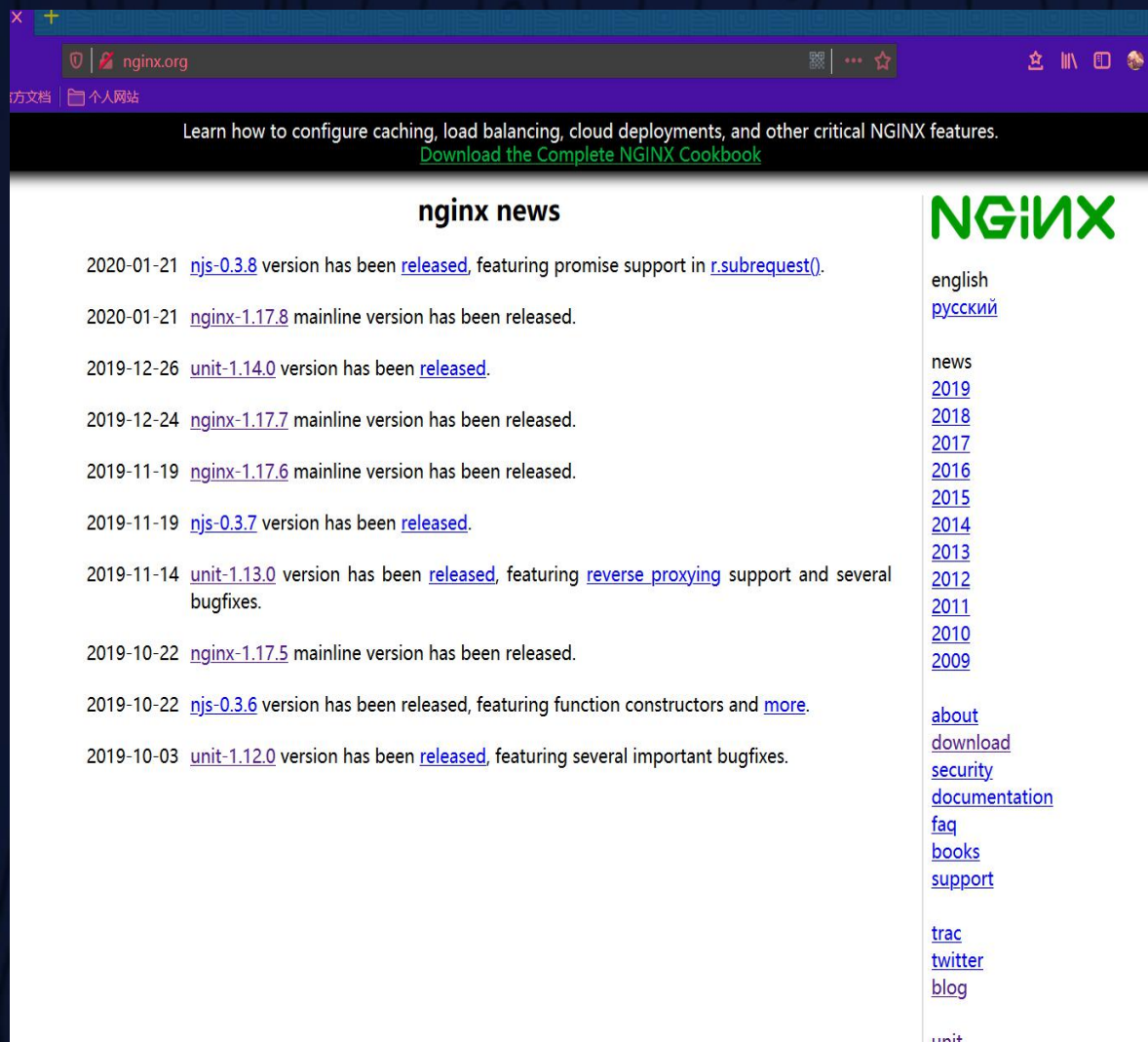




—

# 安装Nginx

# 1.到Nginx官网<http://nginx.org/>



下载

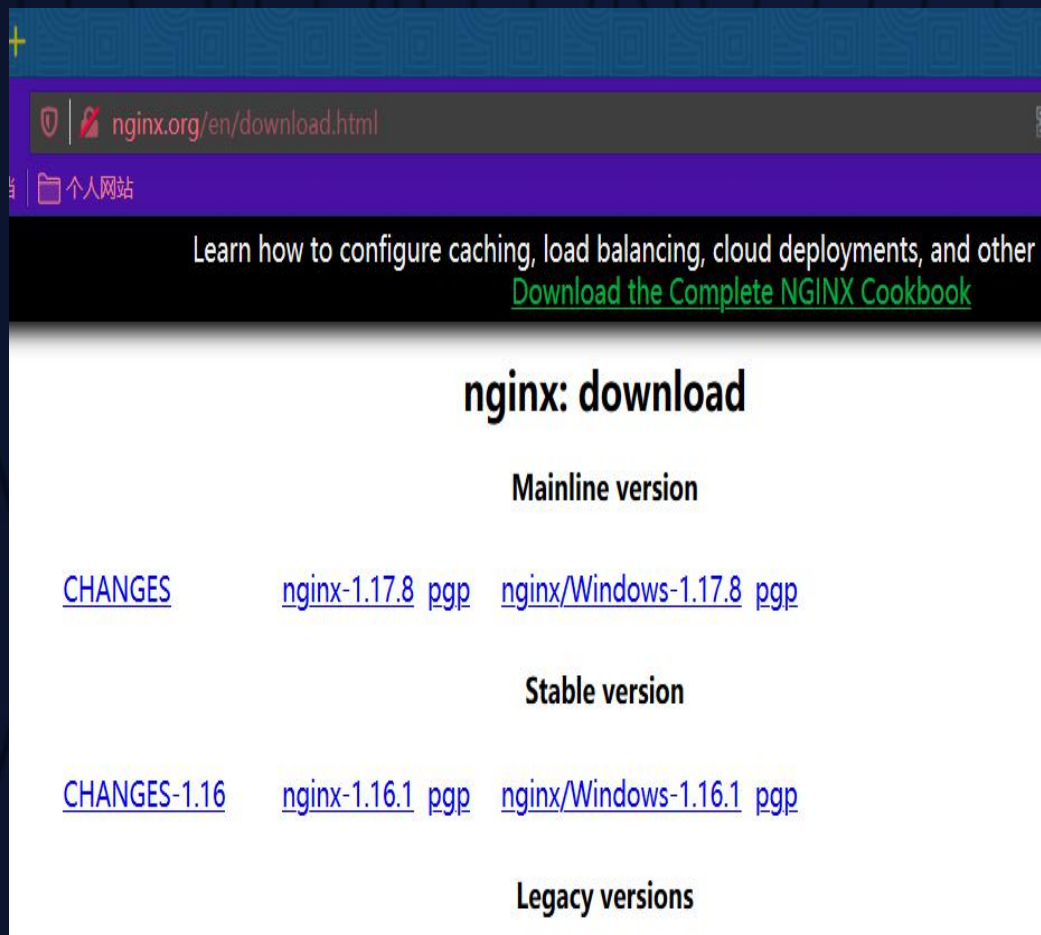
点击右边download,  
进入下载页面

帮助文档

点击右边的  
documentation



## 简要介绍



### Mainline Version

主流版本，即最新开发版本

### Stable Version

稳定版本

### Legacy Version

历史版本

### nginx-1.17.8

源码文件，要经过编译已生成二进制文件，常用语linux环境

nginx/Windows 用于windows系统

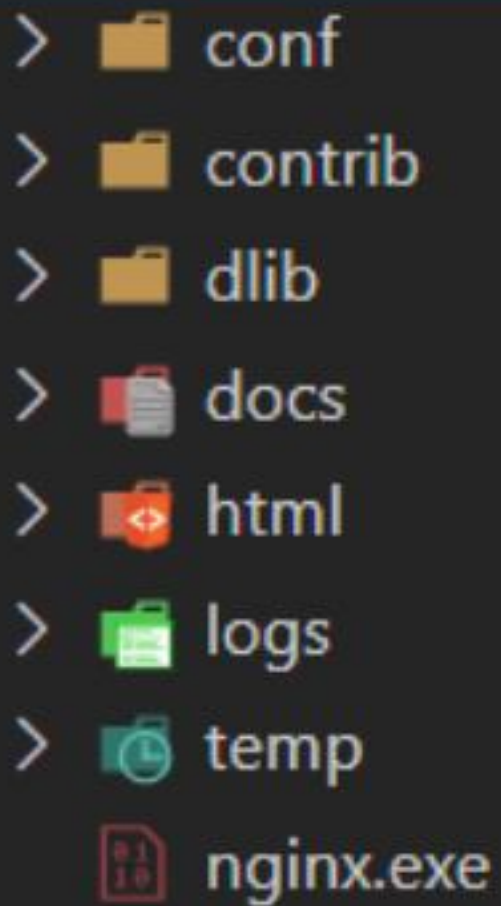










二

# nginx文件结构



# nginx文件结构



- >  conf
- >  contrib
- >  dlib
- >  docs
- >  html
- >  logs
- >  temp
-  nginx.exe

conf为配置文件

contrib有一些方便vim和linux操作的组件等，以及编码配置

dlib是我自己添加的组件

docs为一些文档

html存放一些html文件

logs为一些日志信息，以及开启nginx后.pid文件

temp存放临时文件

nginx.exe为可执行文件，注意不要直接点它

## nginx主要有下列模块

---

**Ningx.conf配置文件**  
控制Nginx的行为

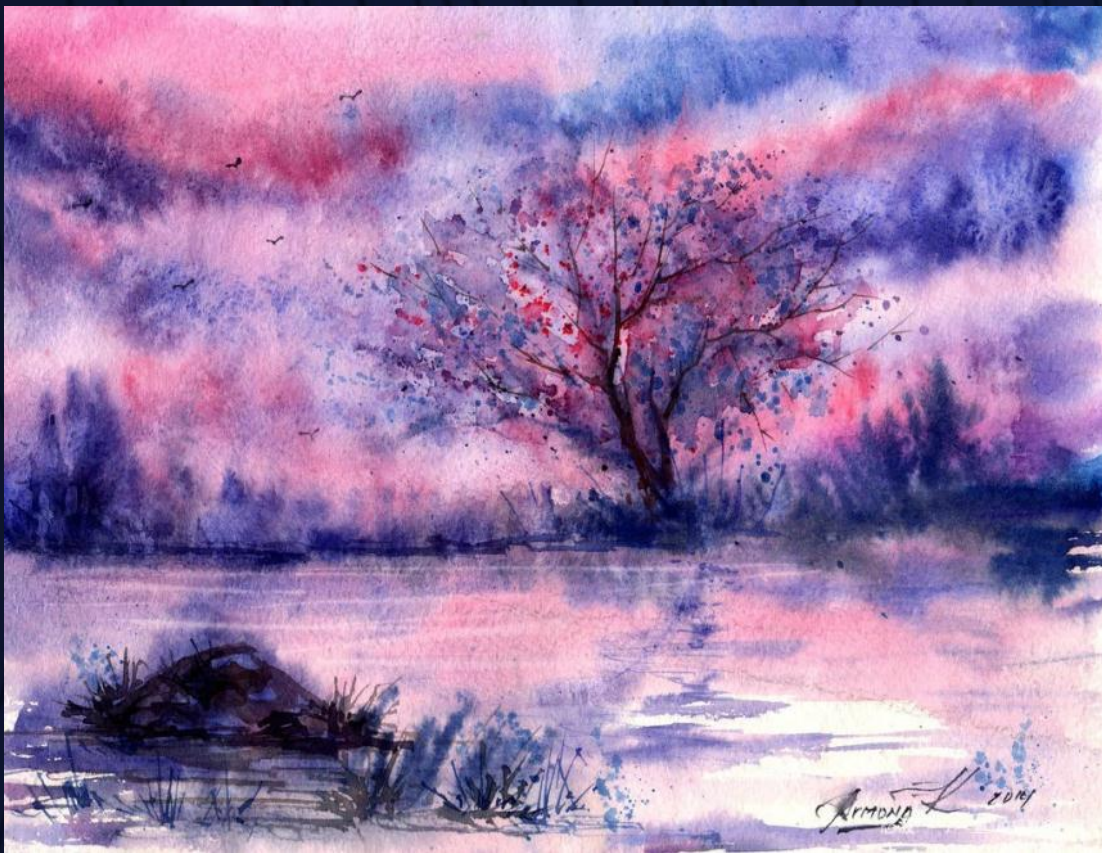
**Nginx二进制可执行文件**  
由各模块源码编译出的一个文件

**access.log访问日志**  
记录每一条http请求信息

**error.log错误日志**  
定位问题



## 相关nginx命令



**start nginx**  
开启nginx



**nginx -s reload**

修改了配置文件后重新开启



**nginx -s quit**  
优雅地退出

**nginx -s stop**  
强行退出





03

# 搭建服务器

一台web静态资源服务器

一台反向代理，2台web静态资源服务器



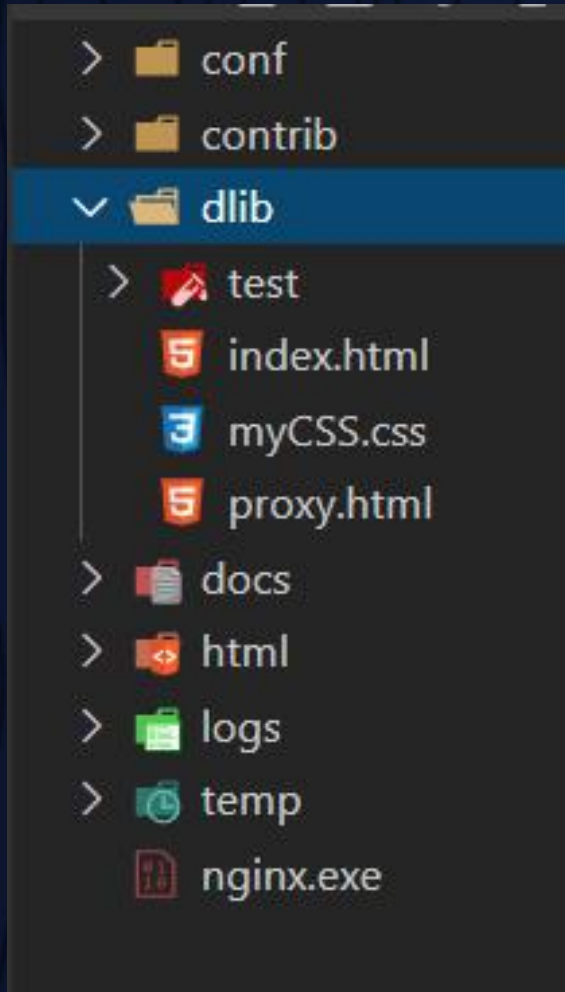
## 步骤

1. 在dlib下建好相应的文件



2. 打开conf文件夹，进入nginx.conf

修改配置文件



## nginx.conf

```
38  server {
39      listen          8080;
40      #                server_name localhost;
41
42      #charset         koi8-r;
43
44      #access_log      logs/host.access.log main;
45      proxy_set_header Host $host;
46      proxy_set_header X-Forwarder-For $remote_addr;
47
48      location / {
49          alias         dlib/;
50          index         index.html index.htm;
51      }
52
53      location /test {
54          alias         dlib/test/;
55          index         test.html;
56      }
```

这里我开的端口是8080,  
也可以其他端口,  
不要冲突即可

## location设置路由

proxy\_set\_head先不用管,  
这是配置反向代理的部分



## 补充一下常见的http状态码

1XX  
信息响应

2XX  
成功响应

3XX  
重定向

4XX  
客户端响应

5XX  
服务端响应



## 信息响应 1XX

1

### 100 Continue

这个临时响应表明，迄今为止的所有内容都是可行的，客户端应该继续请求，如果已经完成，则忽略它。

2

### 101 Switching Protocol

该代码是响应客户端的 **Upgrade** 标头发送的，并且指示服务器也正在切换的协议。

3

### 102 Processing

此代码表示服务器已收到并正在处理该请求，但没有响应可用。

4

### 103 Early Hints

此状态代码主要用于与 **Link** 链接头一起使用，以允许用户代理在服务器仍在准备响应时开始预加载资源。



## 成功响应 2XX

1

### 200 OK

请求成功。

2

### 201 Created

该请求已成功，并因此创建了一个新的资源。这通常是在POST请求，或是某些PUT请求之后返回的响应。

3

### 202 Accepted

请求已经接收到，但还未响应，没有结果。

### 203 Non-Authoritative Information

服务器已成功处理了请求，但返回的实体头部元信息不是在原始服务器上有效的确定集合，而是来自本地或者第三方的拷贝。

### 204 No Content

服务器成功处理了请求，但不需要返回任何实体内容，并且希望返回更新的元信息。

### 205 Reset Content

服务器成功处理了请求，且没有返回任何内容。

### 206 Partial Content

服务器已经成功处理了部分 GET 请求。

## 重定向 3XX

---

### 300 Multiple Choice

被请求的资源有一系列可供选择的回馈信息， 有自己特定的地址和浏览器驱动的商业信息。

### 301 Moved Permanently

被请求的资源已永久移动到新位置， 并且将来任何对此资源的引用都应该使用本响应返回的若干个 URI 之一。

### 302 Found

请求的资源现在临时从不同的 URI 响应请求。



## 客户端响应 4XX

---

### 400 Bad Request

1、语义有误，当前请求无法被服务器理解。 2、请求参数有误。

### 401 Unauthorized

当前请求需要用户验证。该响应必须包含一个适用于被请求资源的 WWW-Authenticate 信息头用以询问用户信息。

### 403 Forbidden

服务器已经理解请求，但是拒绝执行它。

### 404 Not Found

请求失败，请求所希望得到的资源未被在服务器上发现。

### 405 Method Not Allowed

请求行中指定的请求方法不能被用于请求相应的资源。

## 服务端响应 5XX

---

### 500 Internal Server Error

服务器遇到了不知道如何处理的情况。

### 501 Not Implemented

此请求方法不被服务器支持且无法被处理。

### 502 Bad Gateway

此错误响应表明服务器作为网关需要得到一个处理这个请求的响应，但是得到一个错误的响应。

### 503 Service Unavailable

服务器没有准备好处理请求。常见原因是服务器因维护或重载而停机。

### 504 Gateway Timeout

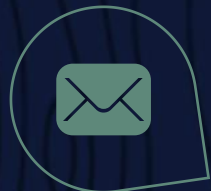
当服务器作为网关，不能及时得到响应时返回此错误代码。



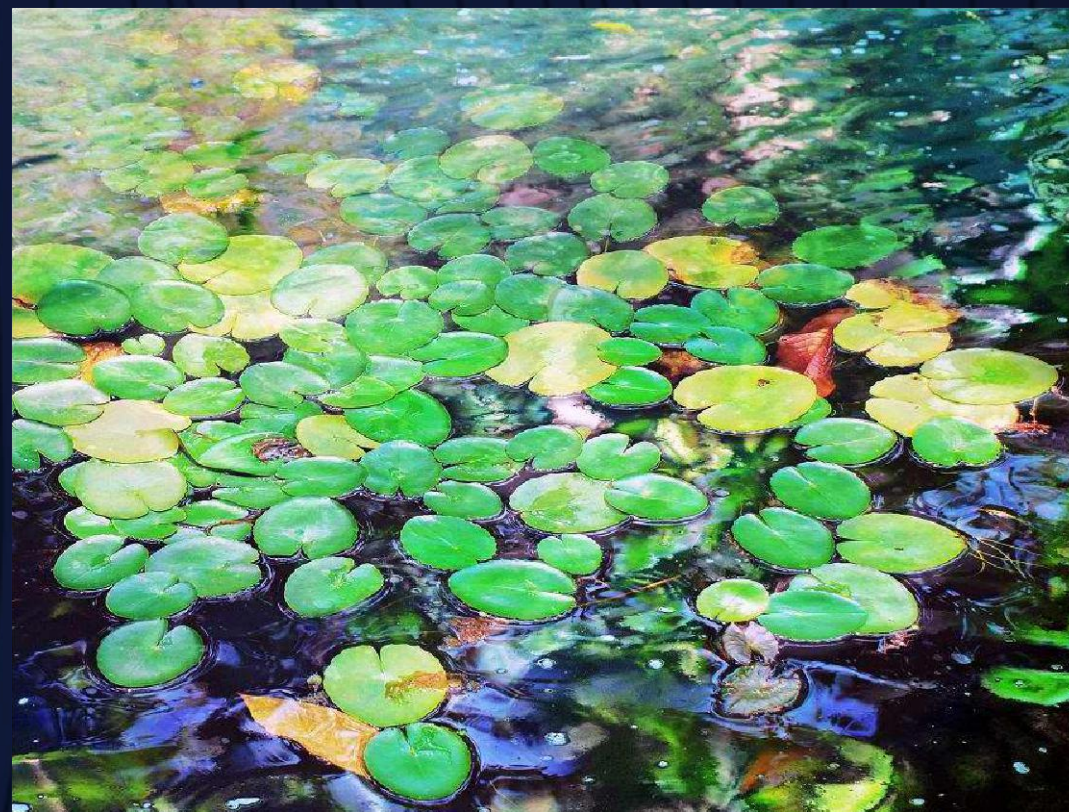
## 让我们回到配置中

---

我们刚刚在dlib下建好  
相应的文件



接着打开conf文件夹，  
进入nginx.conf，修改  
配置文件



## nginx.conf

```
#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;
}

error_page 404 /404.html;
location = /404.html {
    root            html;
}
```

这里处理错误部分的代码，注意在root下新建404.html,不然会404时对用户不友好



资源管理器

&gt; 打开的编辑器

NGINX1

conf

fastcgi\_params

fastcgi.conf

koi-utf

koi-win

mime.types

nginx.conf

scgi\_params

uwsgi\_params

win-utf

&gt; contrib

&gt; dlib

&gt; docs

html

50x.html

404.html

index.html

&gt; logs

&gt; temp

nginx.exe

404.html x

html &gt; 404.html &gt; html

1 &lt;!DOCTYPE html&gt;

2 &lt;html lang="en"&gt;

3 &lt;head&gt;

4 &lt;meta charset="UTF-8"&gt;

5 &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;

6 &lt;meta http-equiv="X-UA-Compatible" content="ie=edge"&gt;

7 &lt;title&gt;Not Found&lt;/title&gt;

8 &lt;/head&gt;

9 &lt;body&gt;

10 &lt;div class=""&gt;

11 &lt;p style="text-align: center;margin-top: 350px;"&gt;Not found.root&lt;/p&gt;

12 &lt;/div&gt;

13 &lt;/body&gt;

14 &lt;/html&gt;

## nginx框架处理大致流程(极简)



1.nginx框架接收  
用户请求



2.路由定位(成功与  
失败)



3.反向代理, 交给  
上游服务器处理并  
返回响应资源(可选)

4.返回响应头部及响应体



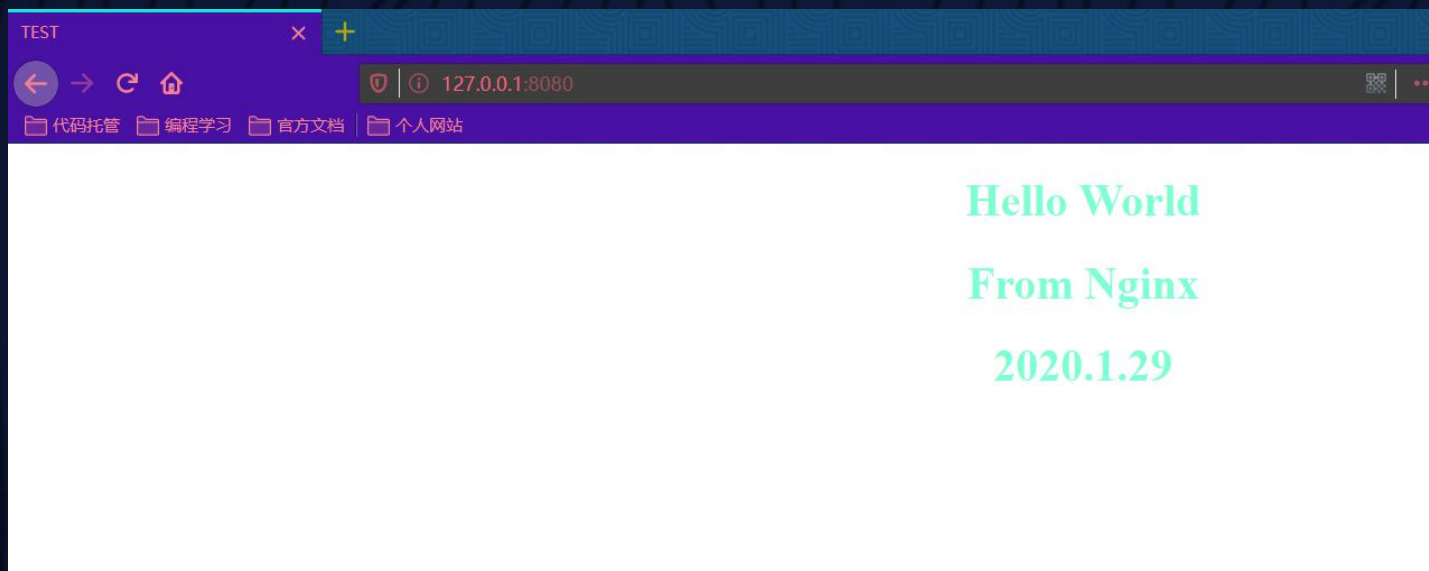
然后我们打开命令行，输入start nginx后回车，会出现一闪而过的nginx命令框(不管它)

```
问题  输出  调试控制台  终端
Microsoft Windows [版本 10.0.15063]
(c) 2017 Microsoft Corporation。保留所有权利。

D:\Nginx\nginx1>start nginx

D:\Nginx\nginx1>
```

在浏览器下输入  
127.0.0.1:8080  
(这里8080是我上一步在nginx.conf文件中设置的监听端口)



testRounter



127.0.0.1:8080/test/

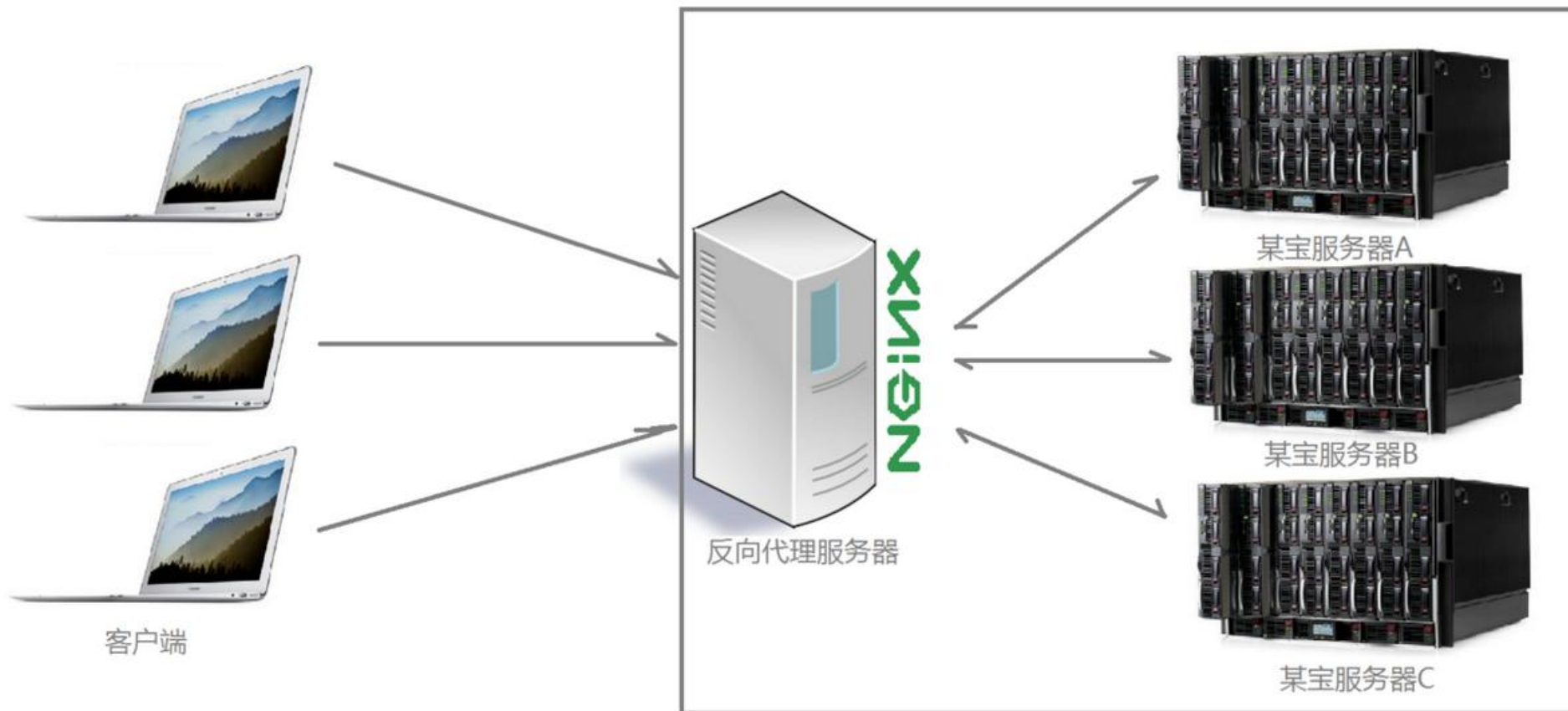



代码托管 编程学习 官方文档 个人网站

Hello haha.



最后我们来看看反向代理





反向代理，可以使服务器的请求分给上游的服务器。

网上的解释：

反向代理（Reverse Proxy）方式是指以代理服务器来接受Internet上的连接请求，然后将请求转发给内部网络上的服务器；

目的是为了实**现负载均衡**，提高服务器整体的性能。

负载均衡可以这样简单理解，就是每个服务器的cpu数目和cpu内部核数不一定相同，根据一些负载均衡算法，可以让他们工作的分量和他们的能力相匹配。

\*下面我以一台nginx服务器作为反向代理服务器，另外两台作为上游服务器进行演示



## nginx1 (反向代理服务器)

在nginx.conf中，http模块内,添加upstream模块，配置上有服务器集群

```
32 upstream upgroup {  
33     server 127.0.0.1:9000 weight=2;  
34     server 127.0.0.1:9090 ;  
35     #nodejs  
36     # server 127.0.0.1:3000;  
37 }
```

## nginx1 (反向代理服务器)

在nginx.conf中，server模块内,添加location模块，配置上路由

```
59     location /proxy {
60         proxy_pass      http://upgroup;
61         #                alias dlib;
62         #                index proxy.html;
63     }
64
65     location /proxy2 {
66         proxy_pass      http://upgroup;
67     }
68
```



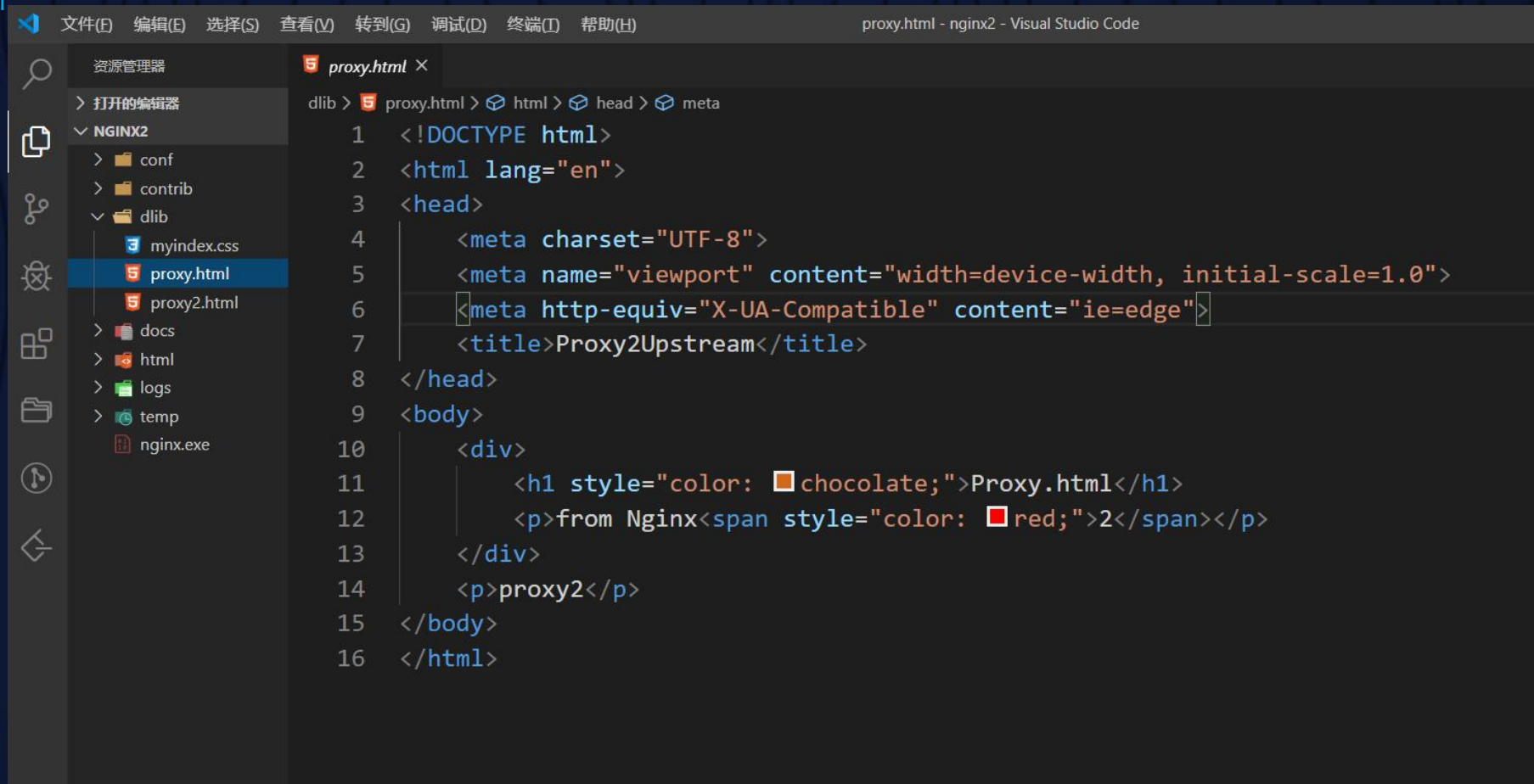
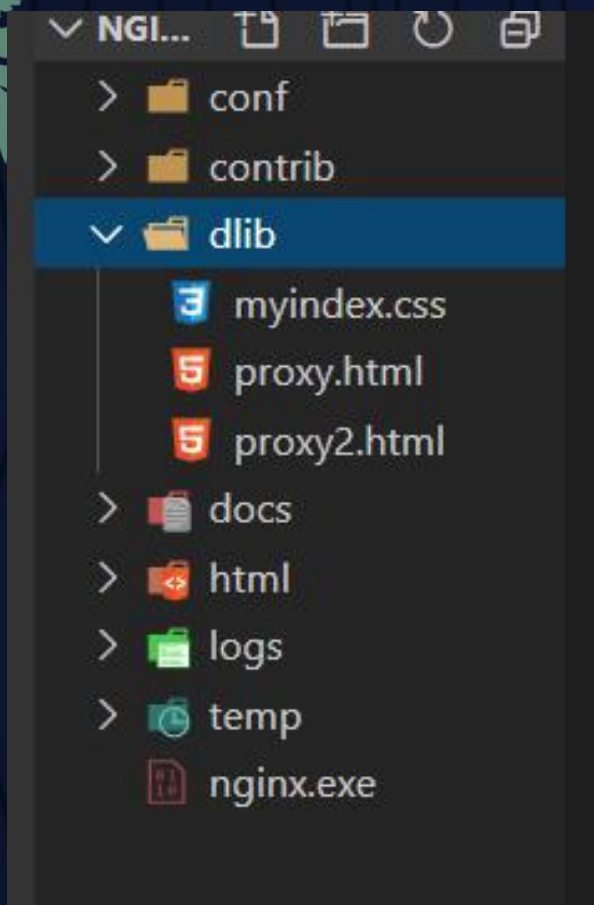
## nginx2 (上游服务器)

在nginx.conf中, server模块内, 添加location模块, 配置上路由

```
32     server {
33         listen      127.0.0.1:9000;
34         #          server_name localhost;
35
36         #charset    koi8-r;
37
38         #access_log  logs/host.access.log main;
39         location /proxy {
40             alias    dlib/;
41             index    proxy2.html;
42         }
43
44
45         location /proxy2 {
46             alias    dlib/;
47             index    proxy.html;
48         }
49 }
```

## nginx2 （上游服务器1）

建好相应文件





## nginx3 (上游服务器2)

在nginx.conf中, server模块内, 添加location模块, 配置上路由, 建好dlib文件夹及其子文件(与nginx2类似)

```
32     server {
33         listen      127.0.0.1:9090;
34         server_name localhost;
35
36         #charset    koi8-r;
37
38         #access_log  logs/host.access.log main;
39         location / {
40             root     html;
41             index     index.html index.htm;
42         }
43
44         location /proxy {
45             alias     dlib/;
46             index     proxy.html;
47         }
48
49         location /proxy2 {
50             alias     dlib/;
51             index     proxy2.html;
52         }
53     }
```

## 依次运行

### 运行上游服务器

问题 输出 调试控制台 终端

Microsoft Windows [版本 10.0.15063]  
(c) 2017 Microsoft Corporation. 保留所有权利。

D:\Nginx\nginx2>start nginx

D:\Nginx\nginx2>█

问题 输出 调试控制台 终端

Microsoft Windows [版本 10.0.15063]  
(c) 2017 Microsoft Corporation. 保留所有权利。

D:\Nginx\nginx3>start nginx

D:\Nginx\nginx3>█

### 运行反向代理服务器

问题 输出 调试控制台 终端

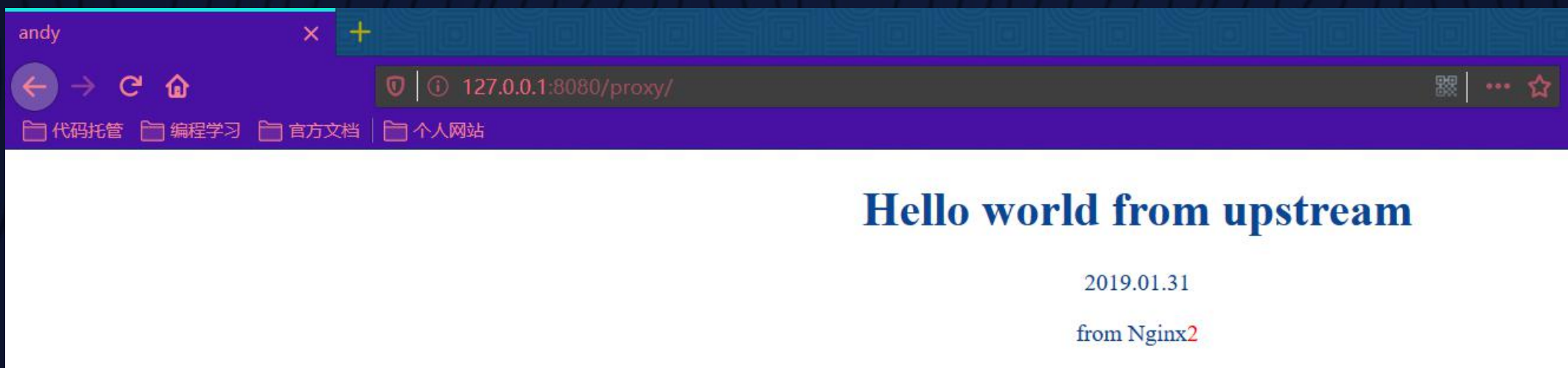
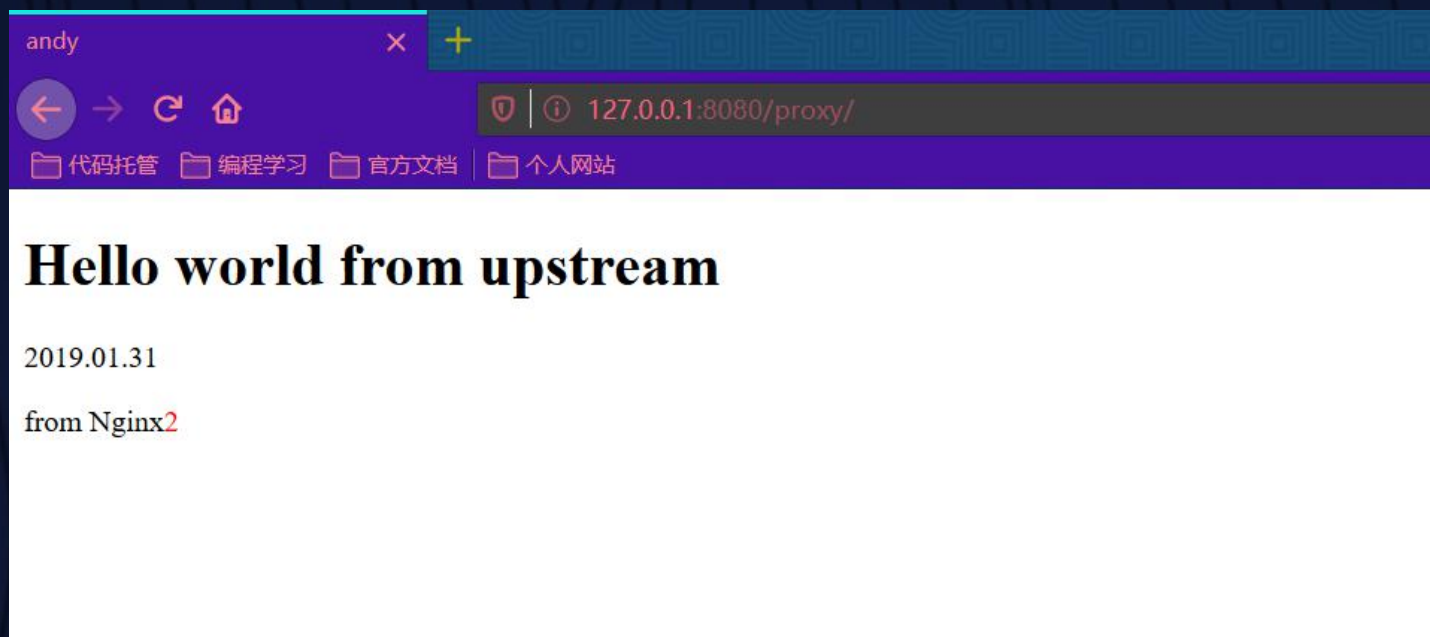
Microsoft Windows [版本 10.0.15063]  
(c) 2017 Microsoft Corporation. 保留所有权利。

D:\Nginx\nginx1>start nginx

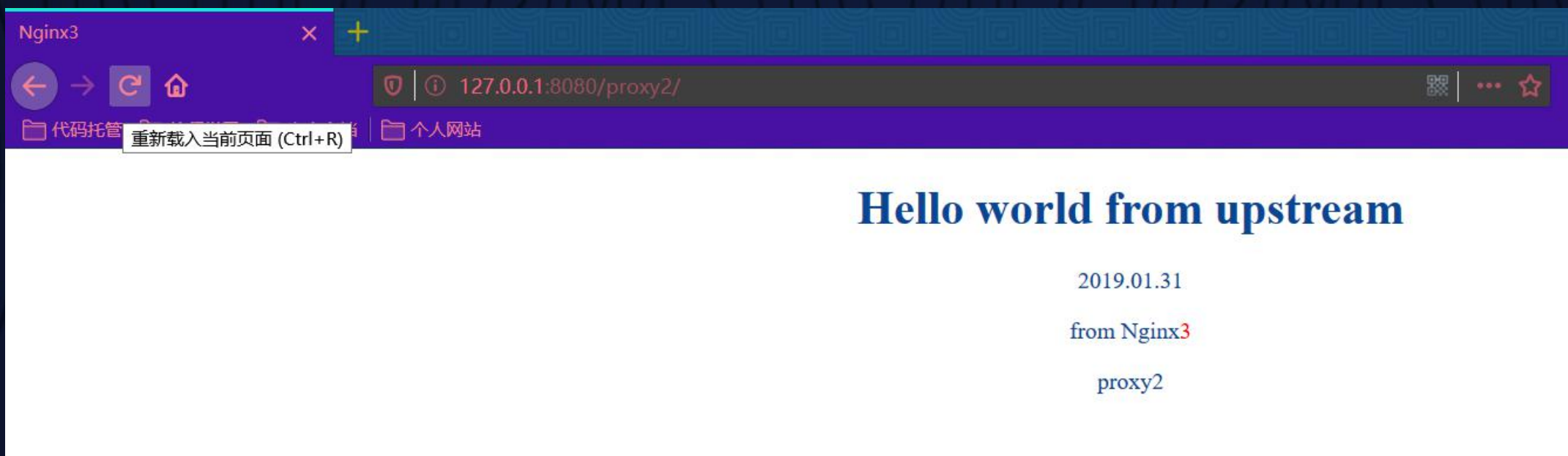
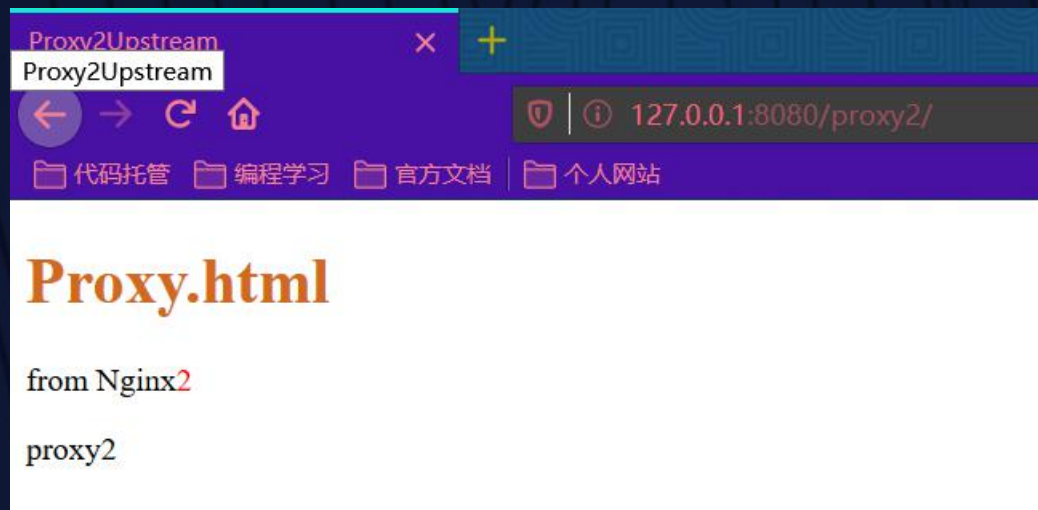
D:\Nginx\nginx1>█



效果



效果







最后依次优雅关闭服务器。

```
D:\Nginx\nginx3>nginx -s quit
```

```
D:\Nginx\nginx2>nginx -s quit
```

由于这里我还要演示一个例子，反向代理服务器就先不关闭了

\*其实，上游服务器也可以是其他服务器。比如nodejs或Django搭建的服务器,Tomcat等

```
32 upstream upgroup {  
33     server 127.0.0.1:9000 weight=2 down;  
34     server 127.0.0.1:9090 down;  
35     #nodejs  
36     server 127.0.0.1:3000;  
37 }
```

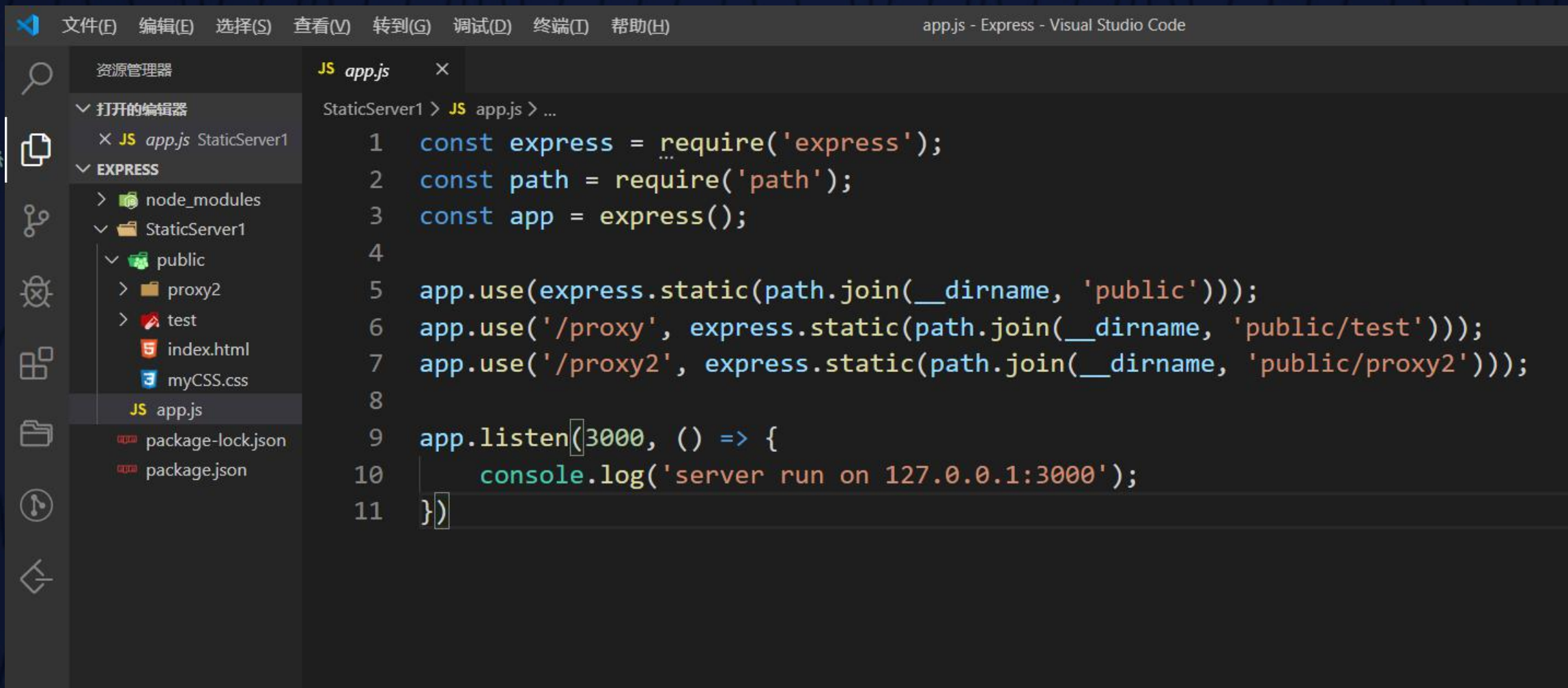
这里down是让其下线的意思，即现在这个上游服务器集群我们只让3000端口的进行工作。

改完保存然后reload一下

```
D:\Nginx\nginx1>nginx -s reload
```



nodejs端我们用express框架搭建一个服务器，用来作为上游服务器



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'StaticServer1' folder containing a 'public' subfolder with 'index.html' and 'myCSS.css' files. The code editor displays the 'app.js' file with the following JavaScript code:

```
1  const express = require('express');
2  const path = require('path');
3  const app = express();
4
5  app.use(express.static(path.join(__dirname, 'public')));
6  app.use('/proxy', express.static(path.join(__dirname, 'public/test')));
7  app.use('/proxy2', express.static(path.join(__dirname, 'public/proxy2')));
8
9  app.listen(3000, () => {
10    console.log('server run on 127.0.0.1:3000');
11  })
```

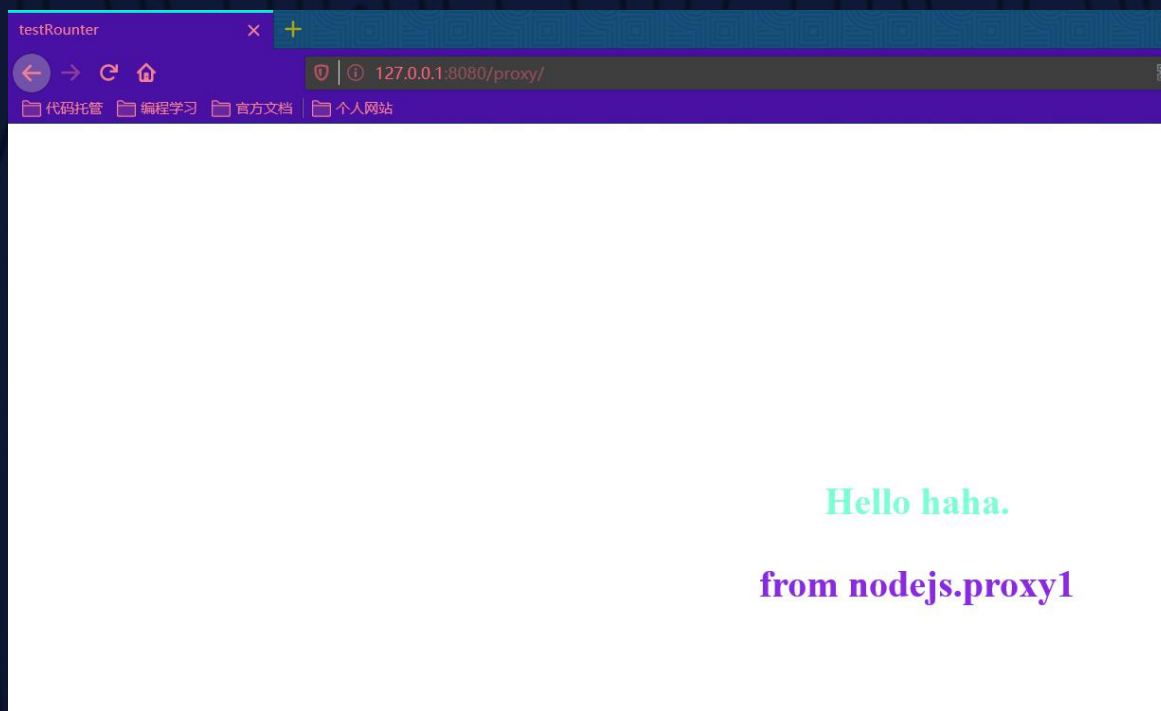
## 运行上游服务器

问题 输出 调试控制台 终端

```
Microsoft Windows [版本 10.0.15063]  
(c) 2017 Microsoft Corporation. 保留所有权利。
```

```
d:\Desktop\vscodeProject\Express\StaticServer1>node app.js  
server run on 127.0.0.1:3000
```

## 查看效果





A decorative border featuring stylized flowers and leaves in shades of green, yellow, orange, and pink, framing the central text.

加油努力向上奋进!

谢谢观看