

Nginx

`sudo systemctl start nginx` 不报错就表示启动成功

`sudo systemctl status nginx` 验证 **Nginx** 是否正常运行

`ps -ef | grep nginx` 显示 Nginx 进程

然后只需要关注前两行，master，worker 进程

第二列 PID 进程 ID，unique

```
root      671      1  0 08:58 ?        00:00:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
www-data  672     671  0 08:58 ?        00:00:00 nginx: worker process
```

`lsof -i:80` 查看哪些进程正在监听 **80 端口**（HTTP 默认端口）

浏览器中输入

<http://localhost/>

或

<http://127.0.0.1>

`sudo nginx -s quit` 优雅停止

`sudo nginx -s stop` 立即停止

`sudo nginx -s reload` 重新配置文件

`sudo nginx -s reopen` 重新打开日志文件

`cat /proc/sys/kernel/pid_max` 查看当前 **PID** 范围

sudo nginx -V 显示 Nginx 的详细版本信息和编译参数

sudo nginx -t 测试配置文件语法是否正确，显示配置文件的加载路径

/etc/nginx/nginx.conf

sudo chown -R yingj /etc/nginx/ 升级权限

code /etc/nginx/nginx.conf 在 vscode 中打开 Nginx 主配置文件看

```
configure arguments: --with-cc-opt='-g -O2 -fno-omit-frame-pointer -mno-omit-leaf-frame-pointer -ffile-prefix-map=/build/nginx-5QYLpr/nginx-1.24.0=, -flto=auto -ffat-lto-objects -fstack-protector-strong -fstack-clash-protection -Wformat -Werror=format-security -fcf-protection -fdebbug-prefix-map=/build/nginx-5QYLpr/nginx-1.24.0=/usr/src/nginx-1.24.0-2ubuntu7.3 -fPIC -Wdate-time -D_FORTIFY_SOURCE=3' --with-ld-opt='-Wl,-Bsymbolic-functions -flto=auto -ffat-lto-objects -Wl,-z,relro -Wl,-z,now -fPIC' --prefix=/usr/share/nginx --conf-path=/etc/nginx/nginx.conf --http-log-path=/var/log/nginx/access.log --error-log-path=stderr --lock-path=/var/lock/nginx.lock --pid-path=/run/nginx.pid --modules-path=/usr/lib/nginx/modules --http-client-body-temp-path=/var/lib/nginx/body --http-fastcgi-temp-path=/var/lib/nginx/fastcgi --http-proxy-temp-path=/var/lib/nginx/proxy --http-scgi-temp-path=/var/lib/nginx/scgi --http-uwsgi-temp-path=/var/lib/nginx/uwsgi --with-compat --with-debug --with-pcre-jit --with-http_ssl_module --with-http_stub_status_module --with-http_realip_module --with-http_auth_request_module --with-http_v2_module --with-http_dav_module --with-http_slice_module --with-threads --with-http_addition_module --with-http_flv_module --with-http_gunzip_module --with-http_gzip_static_module --with-http_mp4_module --with-http_random_index_module --with-http_secure_link_module --with-http_sub_module --with-mail_ssl_module --with-stream_ssl_module --with-stream_ssl_preread_module --with-stream_realip_module --with-http_geoip_module=dynamic --with-http_image_filter_module=dynamic --with-http_perl_module=dynamic --with-http_xslt_module=dynamic --with-mail=dynamic --with-stream=dynamic --with-stream_geoip_module=dynamic
```

参数 (Parameter)	说明 (Description)
--prefix=	指定安装目录
--sbin-path=	指定Nginx可执行文件
--conf-path=	指定配置文件位置
--pid-path=	指定pid文件位置
--error-log-path=	指定错误日志文件
--http-log-path=	指定HTTP日志文件
--user=	指定运行Nginx的用户
--group=	指定运行Nginx的组
--with-pcre=	指定PCRE库的位置
--with-pcre-jit	开启PCRE的JIT（Just-in-time compilation）支持
--with-zlib=	指定zlib库的位置

--prefix: 指定 Nginx 的主目录，包含配置文件、HTML 页面、日志等

cd /usr/share/nginx

ls -ltr 显示目录下所有文件，长格式，按修改时间排，反转排序顺序

里有这个 html 目录

进去找到 html 文件

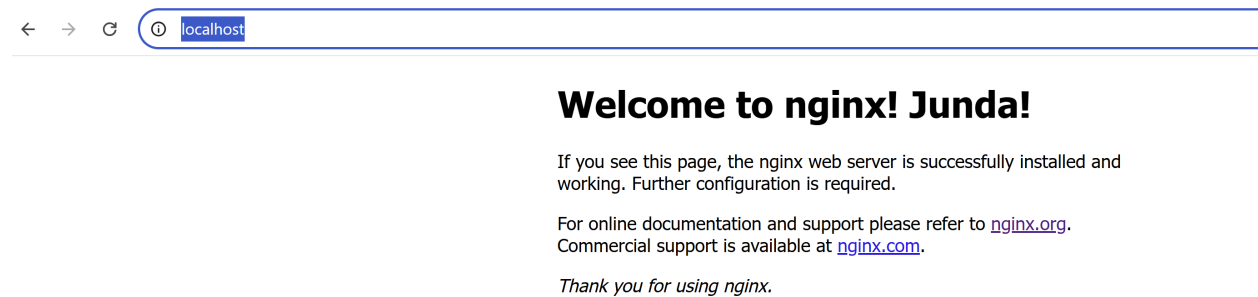
```
cd html
```

```
ls -ltr
```

sudo chown -R yingj /usr/share/nginx/html 给我升级权限去更改，到 html 文件夹

然后就不用 sudo 了

code index.html 网站显示欢迎界面的代码可以修改



回退权限成为 root（按需）

```
sudo chown -R root:root /usr/share/nginx/html
```

也可上传其他 html 到这个文件夹下在浏览器中展示

```
cd /usr/share/nginx/html
```

用 echo 命令快速生成一个简单的 HTML 内容（示例）

```
echo "<h1>Hello, New Page! </h1>" > new_page.html
```

浏览器中输入

```
http://localhost/new_page.html
```



localhost/new_page.html

Hello, New Page!

```
sudo apt install npm
```

安装 **Node.js** 包管理器 (npm)

用 **Hexo** 来生成简单的静态博客网站，把 markdown 转成静态页面

Tips:

Hexo是一个基于Node.js的博客框架。

安装 : npm install hexo-cli -g

初始化 : hexo init blog

安装依赖 : cd blog; npm install

本地运行 : hexo server / hexo s

修改 npm 全局路径

```
mkdir ~/.npm-global
```

```
npm config set prefix '~/.npm-global'
```

```
echo 'export PATH=~/.npm-global/bin:$PATH' >> ~/.bashrc
```

```
source ~/.bashrc
```

安装 `npm install hexo-cli -g`

更改 Nginx 目录所有者为当前用户

```
sudo chown -R yingj:yingj /usr/share/nginx
```

```
cd /usr/share/nginx
```

在当前目录下创建名为 blog-demo 的文件夹，并初始化 Hexo 项目结构

```
hexo init blog-demo
```

```
INFO Cloning hexo-starter https://github.com/hexojs/hexo-starter.git
fatal: unable to access 'https://github.com/hexojs/hexo-starter.git/': Failure when receiving data from the peer
WARN git clone failed. Copying data instead
INFO Install dependencies
INFO Start blogging with Hexo!
```

```
cd blog-demo / cd /usr/share/nginx/blog-demo
```

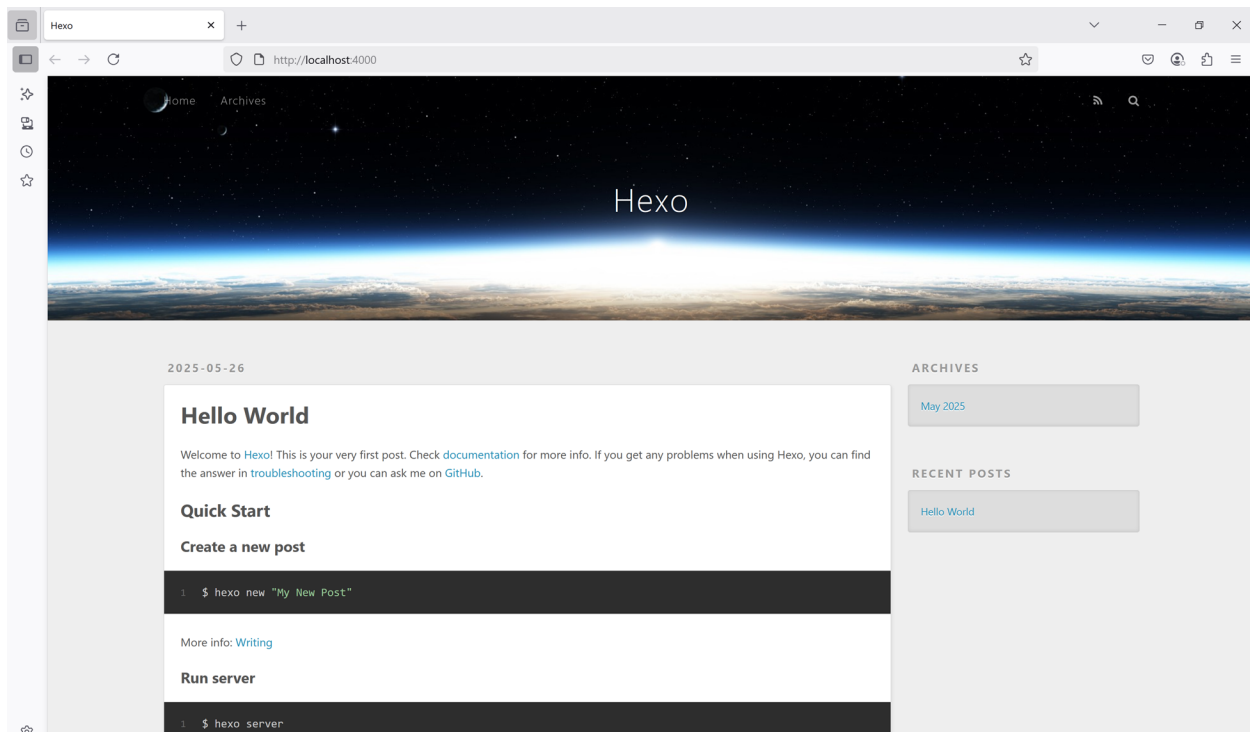
```
ls
```

生成静态页面：`hexo generate / hexo g`

```
INFO Validating config
INFO Start processing
INFO Files loaded in 168 ms
INFO Generated: archives/index.html
INFO Generated: index.html
INFO Generated: archives/2025/05/index.html
INFO Generated: fancybox/jquery.fancybox.min.css
INFO Generated: js/script.js
INFO Generated: css/style.css
INFO Generated: archives/2025/index.html
INFO Generated: fancybox/jquery.fancybox.min.js
INFO Generated: js/jquery-3.6.4.min.js
INFO Generated: css/images/banner.jpg
INFO Generated: 2025/05/26/hello-world/index.html
INFO 11 files generated in 176 ms
```

本地运行 `hexo server` / `hexo s`

在浏览器打开 `http://localhost:4000`，即可看到博客效果。



将 Hexo 生成的静态网站文件 复制到 Nginx 网站目录 中

```
cd public / cd /usr/share/nginx/blog-demo/public
```

因为 Hexo 执行 `hexo generate`（或 `hexo g`）后，会将 Markdown 文章转换为静态 HTML 文件，输出到项目根目录下的 `public` 文件夹中。

```
sudo cp -rf * /var/www/html/
```

- `-r`：递归复制目录及其子目录。
- `-f`：强制覆盖目标目录中已存在的文件。
- 路径解释：

- *: 表示复制 public 目录下的 所有文件和文件夹。

```
sudo rm /var/www/html/index.nginx-debian.html 删旧的
```

```
sudo nano /etc/nginx/sites-available/default
```

```
index 后面改成 index index.html;
```

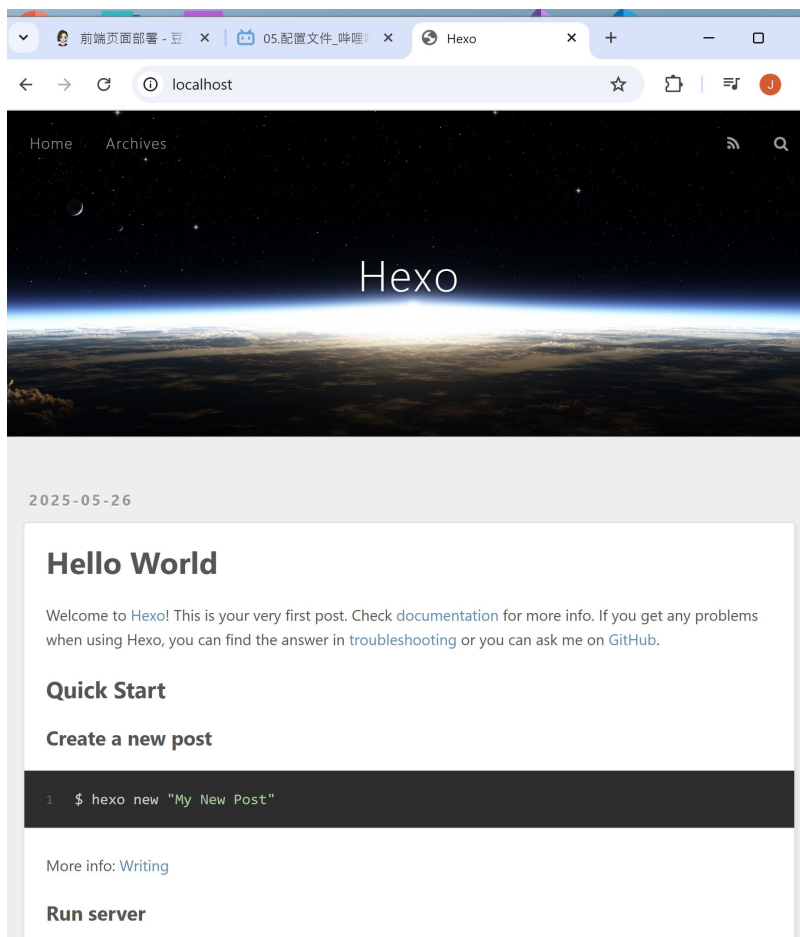
Nginx root 目录与 Hexo 项目目录(usr/share/nginx/blog-demo/public)必须分开

```
root /usr/share/nginx/html;改成 root /var/www/html/;
```

关闭后重新开下

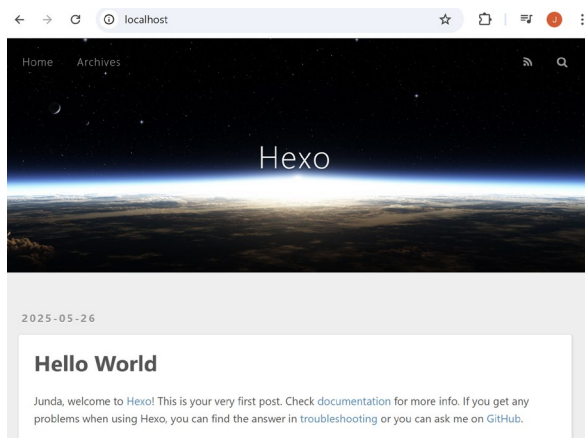
```
sudo systemctl stop nginx
```

```
sudo systemctl start nginx
```



sudo chown -R yingj:yingj /var/www/html 将 /var/www/html 目录的所有权改为当前用户

code index.html 稍微改下



hexo 部署到 github

```
cd /usr/share/nginx/blog-demo
```

```
_config.landscape.yml  db.json  package-lock.json  public  source
_config.yml           node_modules  package.json      scaffolds  themes
```

创建 GitHub 仓库:

仓库名必须为 用户名.github.io

在 Hexo 项目的配置文件_config.yml 中，找到 deploy 部分，进行如下配置:

deploy:

type: git

repo: [git@github.com:jstyling/jstyling.github.io.git](https://github.com/jstyling/jstyling.github.io.git)

branch: main

```
cd /usr/share/nginx/blog-demo
```

部署到 GitHub/GitLab (Git 协议)


```
npm install hexo-deployer-git --save
```


hexo clean 清除缓存文件和已生成的静态文件。


hexo deploy 将 public 目录下的静态文件推送到 GitHub 仓库。


```
hexo clean && hexo d
```


☰

 jstying / jstying.github.io

 ▾

 ▾





<> Code

🕒 Issues

🔗 Pull requests

🔄 Actions


📁 Projects

📖 Wiki

🛡 Security

📊 Insights

⚙ Settings

 jstying.github.io Public

📌 Pin

👁 Watch 0 ▾

🍴 Fork 0 ▾

☆ Sta


🔗 main ▾

🔗

📁


+

<> Code ▾

 jstying Site updated: 2025-05-26 10:29:28 ● b4d5a36 · now 🕒 3 Commits

📁 2025/05/26/hello-world	Site updated: 2025-05-26 10:29:28	now
📁 archives	Site updated: 2025-05-26 10:21:49	8 minutes ago
📁 css	Site updated: 2025-05-26 10:21:49	8 minutes ago
📁 fancybox	Site updated: 2025-05-26 10:21:49	8 minutes ago
📁 js	Site updated: 2025-05-26 10:21:49	8 minutes ago
📄 index.html	Site updated: 2025-05-26 10:29:28	now

📖 README



Add a README

Help people interested in this repository understand your project by adding a README.

Add a README

About

No description, website, or topics

📈 Activity

☆ 0 stars

👁 0 watching

🍴 0 forks

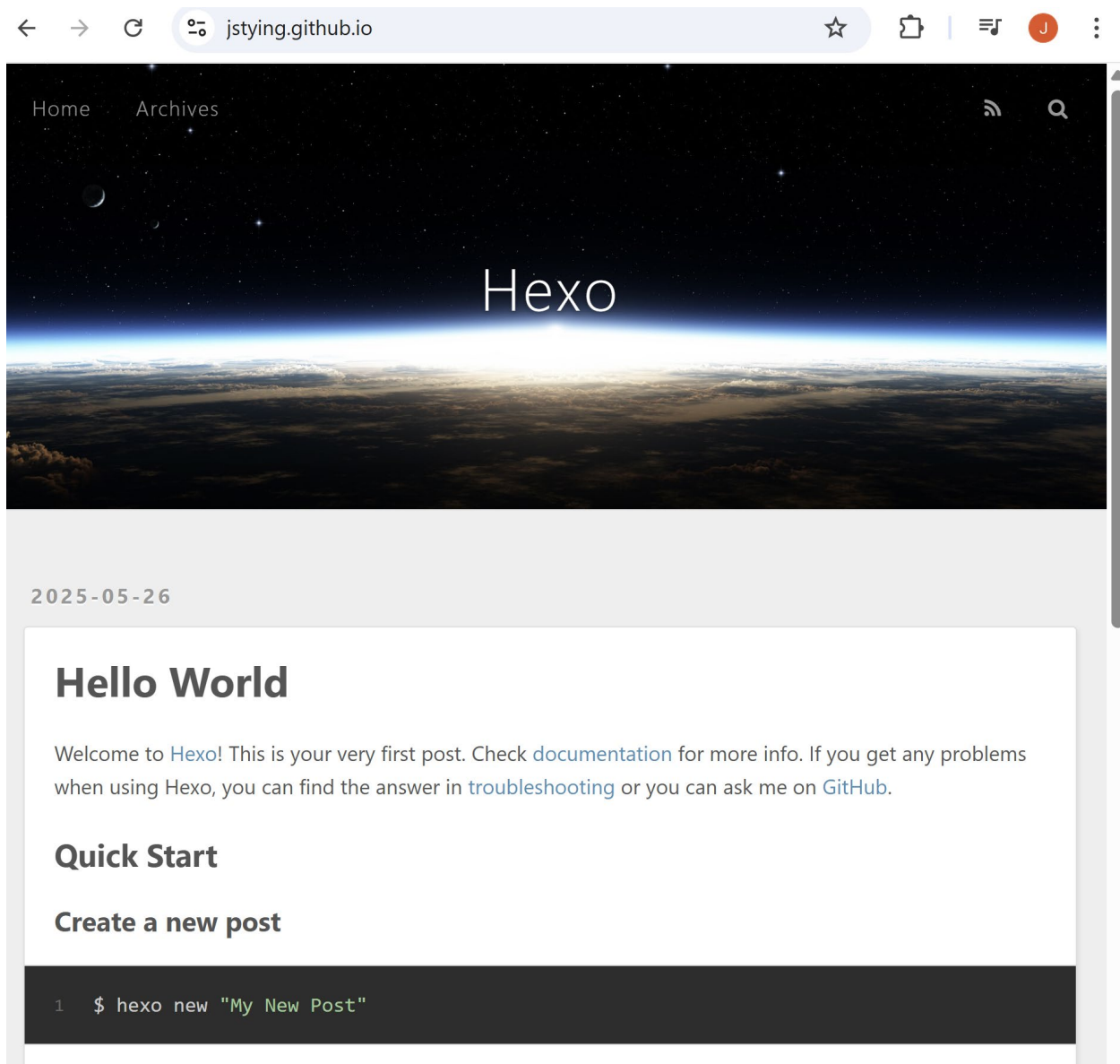
Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

<https://jstying.github.io> 公开可访问的



blog-demo/ # 项目根目录（包含源文件）

└─ source/ # 文章源文件（.md）

 └─ _posts/ # 博客文章

└─ themes/ # 主题

└─ public/ # 生成的静态文件（部署到服务器）

 └─ _config.yml # 配置文件

```
cd /usr/share/nginx/blog-demo
```

```
hexo new Junda_1stBlog    # 创建新文章
```

```
yingj@LAPTOP-AHTL5GV8:/usr/share/nginx/blog-demo$ hexo new Junda_1stBlog
INFO Validating config
INFO Created: /usr/share/nginx/blog-demo/source/_posts/Junda-1stBlog.md
```

```
code source/_posts/Junda-1stBlog.md
```

```
hexo clean && hexo deploy
```

Hexo

2025-05-26

Junda_1stBlog

大家好！我是 Junda（也可以叫我 Justin），很高兴在这里开启我的第一篇博客。

#自我介绍

🔗 Share

2025-05-26

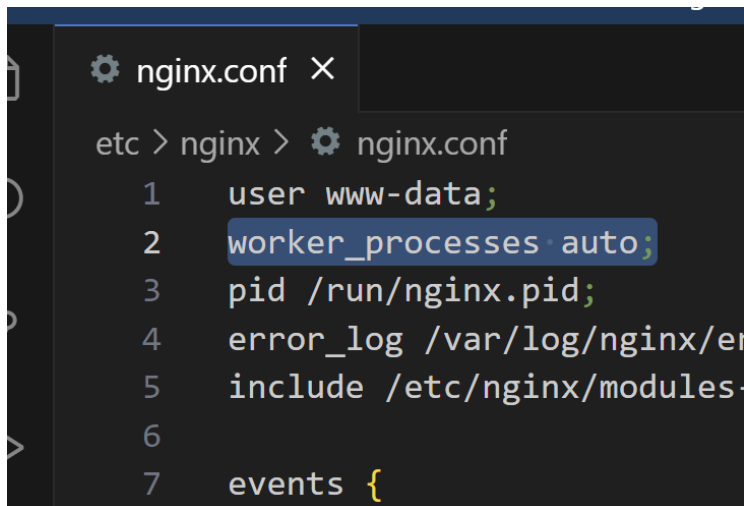
Hello World

Welcome to [Hexo](#)! This is your very first post. Check [documentation](#) for more info. If you get any problems when using Hexo, you can find the answer in [troubleshooting](#) or you can ask me on [GitHub](#).

配置文件

cd /etc/nginx/ 配置文件位置!!

code nginx.conf



```
nginx.conf X
etc > nginx > nginx.conf
1 user www-data;
2 worker_processes auto;
3 pid /run/nginx.pid;
4 error_log /var/log/nginx/error.log;
5 include /etc/nginx/modules-enabled/*.conf;
6
7 events {
```

worker 进程的数量现在是根据内核分配数量也可以自己改

worker_processes 10;

sudo nginx -t 看配置文件是否正确

```
2025/05/26 11:22:09 [emerg] 10742#10742: invalid number of arguments in "worker_processes" directive in /etc/nginx/nginx.conf:3
nginx: configuration file /etc/nginx/nginx.conf test failed
```

提示问题出在第三行

sudo nginx -s reload 重新配置文件

ps -ef | grep nginx 显示 Nginx 进程

```
root      5690      1  0 09:57 ?        00:00:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
www-data  11001     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11002     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11003     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11004     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11005     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11006     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11007     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11008     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11009     5690  0 11:23 ?        00:00:00 nginx: worker process
www-data  11010     5690  0 11:23 ?        00:00:00 nginx: worker process
yingji    11090     389  0 11:23 pts/0    00:00:00 grep --color=auto nginx
```



全局块: worker process 数量, 指定运行的用户等等

events 块: 服务器和客户端网络连接的配置

http 块: server 块/虚拟主机

include /etc/nginx/mime.types;把这个文件包含进来

```
types {
    text/html          html htm shtml;
    text/css           css;
    text/xml           xml;
    image/gif          gif;
    image/jpeg         jpeg jpg;
    application/javascript js;
    application/atom+xml atom;
    application/rss+xml rss;

    text/mathml        mml;
    text/plain         txt;
    text/vnd.sun.j2me.app-descriptor jad;
    text/vnd.wap.wml   wml;
    text/x-component   htc;

    image/avif         avif;
    image/png          png;
    image/svg+xml      svg svgz;
    image/tiff         tif tiff;
    image/vnd.wap.wbmp wbmp;
    image/webp         webp;
```

各种文件的后缀

还可详见 teachersNotes

反向代理和负载均衡

正向代理-代理客户端 Client – vpn

代理服务器知道客户端的身份，但目标服务器不知道客户端的真实 IP

反向代理-代理服务端 Server

用户在浏览器输入 google.com，请求先到达反向代理服务器，再给真实服务器

静态页面（Static Web Page）是指内容预先编写并存储在服务器上，每次被访问时内容固定不变、直接返回给用户的网页。它不依赖服务器端动态处理逻辑，也不与数据库交互

go 语言搭建 3 个 web 服务器，然后用 nginx 做反向代理

```
go main.go  X  go main-8001.go  go main-8002.go  nginx.conf  $ default

nginxDemo > go main.go
1  package main
2
3  import (
4      "fmt"
5      "net/http"
6  )
7
8  func main() {
9      http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
10         // 设置响应头，指定内容类型为HTML
11         w.Header().Set("Content-Type", "text/html")
12
13         // 使用正确的HTML标签
14         fmt.Fprintf(w, "<h1>Port 8000!</h1>\n")
15     })
16
17     // 启动服务器，监听8000端口
18     fmt.Println("服务器启动，访问 http://localhost:8000")
19     http.ListenAndServe(":8000", nil)
20 }
```


总共有 3 个，go 文件，只改下 8000，8001，8002

全部打开 每个都要 go run main.go 开 3 个终端 vscode ctrl shift `

```
go run main-8001.go
```

```
go run main-8002.go
```

之后

负载均衡：将客户端请求均匀分配到多个后端服务器，提高可用性和性能。

在 nginx.conf 里的 http 块里加上

```
upstream backend{ #backend 是任取的名字
```

```
    server 127.0.0.1:8000;
```

```
    server 127.0.0.1:8001;
```

```
    server 127.0.0.1:8002;
```

```
}
```

在 server 块/default 文件里

```
location /app{
```

```
    proxy_pass http://backend; # 将请求转发到名为 backend（负载均衡）
```

```
}
```

默认是**轮询 Round Robin**) 选择一个后端服务器 不断刷新 localhost/app 会出现 8000，8001，8002，8000。。。。

```
upstream backend{
```

```
    ip_hash; # 基于客户端 IP 的哈希值，怎么刷新始终访问同一服务器
```

```
    server 127.0.0.1:8000;
```

```
    server 127.0.0.1:8001;
```

server 127.0.0.1:8002 weight=2; # weight 参数设置权重，此处 8002 端口被选中的概率是其他服务器的 2 倍

}

它接收客户端的请求，然后将请求转发给后端服务器，并将后端的响应返回给客户端。客户端无需知道后端服务器 8000-8003 的存在，只与反向代理服务器（如 Nginx）通信。

反向代理的英文是 **reverse proxy**，负载均衡的英文是 **load balancing**

https 配置-----

http port 80

https port 443 – SSL 证书

使用 openssl 生成证书

cd etc/nginx

输入这 3 个命令

Tips: 使用openssl生成证书

生成私钥文件 (private key)

```
openssl genrsa -out private.key 2048
```

根据私钥生成证书签名请求文件 (Certificate Signing Request, 简称CSR文件)

```
openssl req -new -key private.key -out cert.csr
```

使用私钥对证书申请进行签名从而生成证书文件 (pem文件)

```
openssl x509 -req -in cert.csr -out cacert.pem -signkey private.key
```

```
openssl genrsa -out private.key 2048
```

```
openssl req -new -key private.key -out cert.csr
```

```
openssl x509 -req -in cert.csr -out cacert.pem -signkey private.key
```

```
Certificate request self-signature ok
subject=C = CN, ST = Shanghai, L = city, O = yjd, OU = 1, CN = junda, emailAddr
ess = justinying2006@gmail.com
```

填写完后生成 2 个文件

cacert.pm private.key

加到 server 块里

```
# SSL configuration
```

```
#SSL 开这个
```

```
listen 443 ssl default_server;
```

```
listen [::]:443 ssl default_server;
```

```
# 证书文件名称
```

```
ssl_certificate /etc/nginx/cacert.pem;
```

```
# 证书私钥文件名称
```

```
ssl_certificate_key /etc/nginx/private.key;
```

ssl 验证配置

ssl_session_timeout 5m; # 缓存有效期

安全链接可选的加密协议

ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;

配置加密套件 / 加密算法，写法遵循 openssl 标准。

ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4;

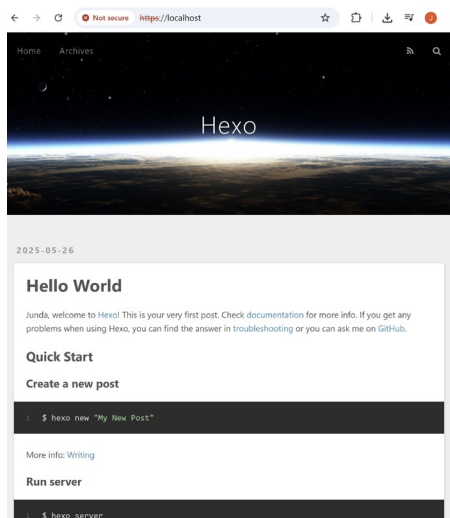
使用服务器端的首选算法

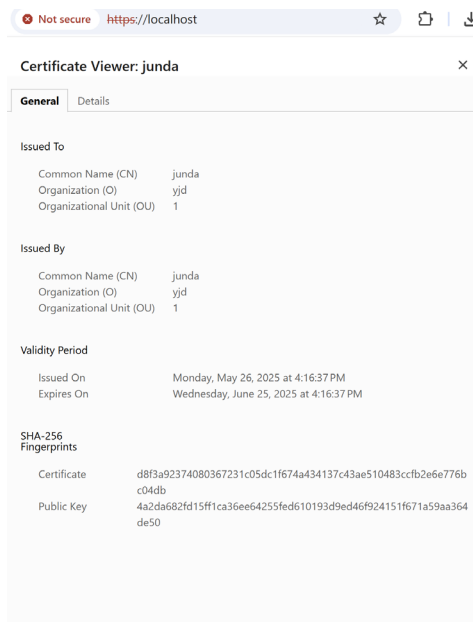
ssl_prefer_server_ciphers on;

写完后

sudo nginx -s reload

浏览器这次用 https





自签名的不被认证

```
server {  
  
    listen 80;  
  
    server_name localhost; # 只匹配 localhost  
  
    return 301 https://localhost$request_uri; # 明确指定重定向到 https://localhost  
  
}
```

重定向成功，但是可能是证书不行，浏览器上无法 http 的 localhost 转到 https 的

```
yingj@LAPTOP-AHTL5GV8:/etc/nginx/sites-enabled$ curl -I http://localhost  
HTTP/1.1 301 Moved Permanently  
Server: nginx/1.24.0 (Ubuntu)  
Date: Mon, 26 May 2025 08:54:27 GMT  
Content-Type: text/html  
Content-Length: 178  
Connection: keep-alive  
Location: https://localhost/
```

虚拟主机

每个 server 块都是个虚拟主机

server_name 来指导虚拟主机的域名

一台服务器上布置多个网站，部署 vue3 站点，然后用虚拟主机的方式布置到 nginx 上

```
yingjj@LAPTOP-AHTL5GV8:/etc/nginx$ npm create vite
Need to install the following packages:
  create-vite@6.5.0
Ok to proceed? (y) y

◇ Project name:
  vue-demo

◇ Select a framework:
  Vue

◇ Select a variant:
  TypeScript

◇ Scaffolding project in /etc/nginx/vue-demo...

Done. Now run:

cd vue-demo
npm install
npm run dev

yingjj@LAPTOP-AHTL5GV8:/etc/nginx$ cd vue-demo
yingjj@LAPTOP-AHTL5GV8:/etc/nginx/vue-demo$ npm install
```

```
npm run build
```

ls -ltr 一行行，按时间反向，旧的在前面

```

yingj@LAPTOP-AHTL5GV8:/etc/nginx/vue-demo$ ls -l
total 88
-rw-r--r--  1 yingj yingj  155 May 27 13:48 vite.config.ts
-rw-r--r--  1 yingj yingj  630 May 27 13:48 tsconfig.node.json
-rw-r--r--  1 yingj yingj  119 May 27 13:48 tsconfig.json
-rw-r--r--  1 yingj yingj  424 May 27 13:48 tsconfig.app.json
drwxr-xr-x  4 yingj yingj 4096 May 27 13:48 src
drwxr-xr-x  2 yingj yingj 4096 May 27 13:48 public
-rw-r--r--  1 yingj yingj  411 May 27 13:48 package.json
-rw-r--r--  1 yingj yingj  362 May 27 13:48 index.html
-rw-r--r--  1 yingj yingj  442 May 27 13:48 README.md
-rw-r--r--  1 yingj yingj 44615 May 27 13:49 package-lock.json
drwxr-xr-x 40 yingj yingj 4096 May 27 13:52 node_modules
drwxr-xr-x  3 yingj yingj 4096 May 27 13:52 dist

yingj@LAPTOP-AHTL5GV8:/etc/nginx/vue-demo$ cd dist
yingj@LAPTOP-AHTL5GV8:/etc/nginx/vue-demo/dist$ ls
assets  index.html  vite.svg
yingj@LAPTOP-AHTL5GV8:/etc/nginx/vue-demo/dist$ pwd
/etc/nginx/vue-demo/dist

```

pwd 打印工作目录

新建 vue.conf

touch vue.conf 创建新文件

- **HTTP 协议**：默认端口为 **80**，因此 http://localhost 等价于 http://localhost:80。
- **HTTPS 协议**：默认端口为 **443**，因此 https://localhost 等价于 https://localhost:443。

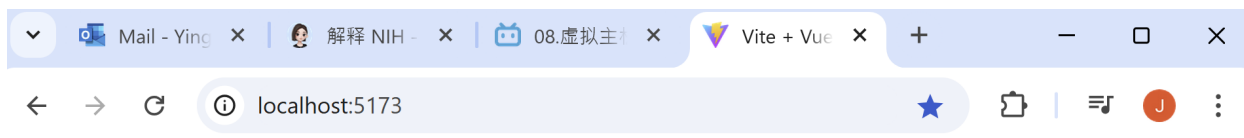
所以我们自定义 5173 是要 localhost:5173

vue.conf

```
1  server{
2      listen 5173;
3      server_name localhost; #域名 http://localhost:5173
4      location / {
5          root /etc/nginx/vue-demo/dist; #vue项目的根目录
6          index index.html index.htm; #寻找展示的html文件
7      }
8  }
```

nginx.conf

```
#include vue.conf 虚拟主机
include /etc/nginx/vue.conf;
```

Vite + Vue

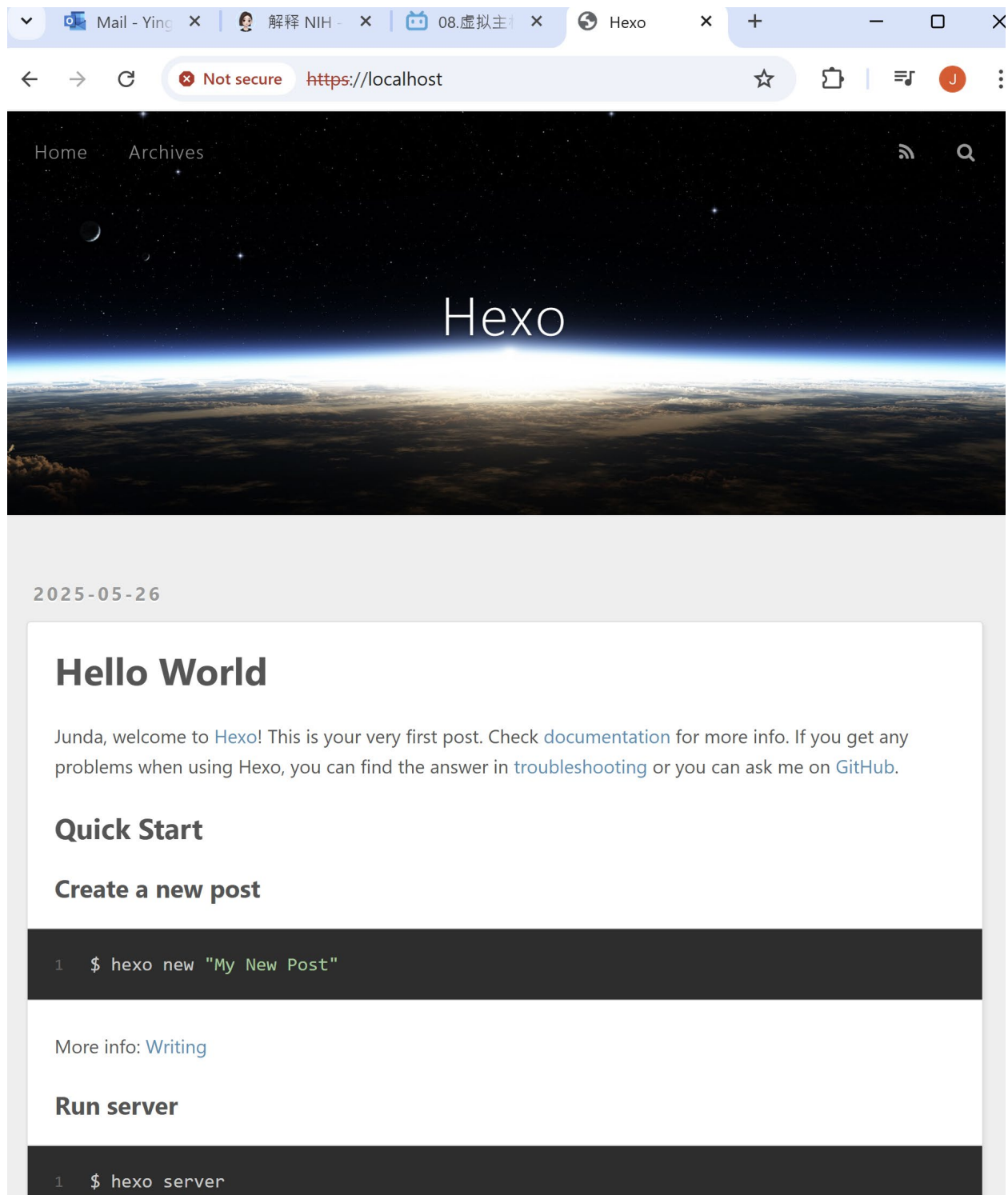
count is 7

Edit components/HelloWorld.vue to test HMR

Check out [create-vue](#), the official Vue + Vite starter

Learn more about IDE Support for Vue in the [Vue Docs Scaling up Guide](#).

Click on the Vite and Vue logos to learn more



也在