> 技术要点：Nginx、Firewalld

前言

作为一名开发者，若要往全栈工程师或前端架构师的方向发展，后续几章都要认真学习。部署一个属于自己的服务器，接着都会在该服务器中搞事情。

通过前两章的学习，服务器与域名都部署完毕就能立即访问网站吗？当然不是，访问网站不是简单地将网站源码丢到服务器，还需 站服务器 的加持，这样用户才能访问网站。该过程具体如何实现？本章将带领你**基于Nginx反向代理应用与服务**，通过 Nginx 完成一些日常 站服务器 的工作，解锁个人官网、域名映射、跨域问题、动静分离、反向代理等服务端姿势。

背景：站服务器是什么

站服务器又称 Web服务器 或 网站服务器 ，简称 WS ，指驻留在因特网中的网络应用程序，用于处理客户端的请求并返回响应，可存放资源文件供全世界浏览，可存放数据文件供全世界下载。

站服务器 主要功能是向客户端提供文档服务，只要是遵循 HTTP 设计的网络应用程序都可认为是 站服务器 。其工作原理分为三个过程，分别是 连接过程 、 请求过程 和 应答过程 。

熟悉 HTTP服务器 的同学可能发现 站服务器 的工作原理与其类似。其实在软件开发与使用时， 站服务器 等同于 HTTP服务器 。虽然不同 站服务器 间有细节上的差异，但其基础原理都是一样的。

- **Step1**：用户在地址栏中输入网址并按下回车键
- **Step2**：客户端与服务端建立 TCP连接
- **Step3**：客户端将用户行为根据 HTTP格式 打包成数据包
- **Step4**：服务端接收到数据包后，以 HTTP格式 解析并获取客户端意图
- **Step5**：客户端确认服务端可写，将数据包通过因特网递交到服务端
- **Step6**：服务端获取客户端意图后，将数据包分类处理并返回处理结果
- **Step7**：服务端将处理结果装入缓冲区
- **Step8**：服务端将处理结果根据 HTTP格式 打包成数据包

- **Step9**：服务端确定客户端可写，将数据包通过因特网递交到客户端
- **Step10**：客户端接收到数据包后，以 HTTP格式 解析并获取处理结果
- **Step11**：客户端处理相关数据并展示在网页中

通过梳理不难发现，这是一个简单的网络通信过程。简而言之，就是一个简单的 发送数据 、 接受数据 和 处理数据 的过程。更高级的 站服务器 ，无非就是将上述三个过程的内容划分得更详细一点而已。若非得对 HTTP服务器 与 站服务器 做一个差异化对比，那 HTTP服务器 是建立在 HTTP 上用于提供文档浏览的服务器，更多的是提供静态文件的浏览功能，而 站服务器 则涵盖了 HTTP服务器 涉及的功能，不仅能存储文档还能在用户通过客户端提供信息的基础上运行其他脚本。

常见 站服务器 有 Nginx 、 Apache 和 Tomcat 。

Nginx

Nginx是一个轻量级开源的 站服务器 软件，具备反向代理、负载均衡、缓存服务等功能，为高并发网站的应用场景而设计。随着技术发展与业务扩张，其逐渐受到关注，在国内一线互联网公司，例如阿里、腾讯、百度、网易、新浪等都开始广泛使用其满足一些高并发业务场景。

Apache

Apache是一个广泛流行的 站服务器 软件，具备开源、跨平台、安全稳定等特性。其伴随互联网的大力发展并经过多年的技术沉淀与积累，已变得成熟稳定且具备大量可扩展功能模块，但在设计之初对性能与资源的消耗并未做过多关注，导致其在应对高并发业务场景时被一些轻量级的高性能 站服务器 赶超。

Tomcat

Tomcat是一个运行 Servlet/JSP 的 站服务器 ，主要用于 JavaWeb 环境，与 Apache 一样都是由 Apache软件基金会 运作的开源项目。其本身可作为一个单独的 站服务器 使用，主要用于处理动态请求，但在静态资源与高并发方面的性能较弱，因此常与 Apache 等软件搭配使用，实现动静态请求分离。

为了方便理解，我梳理了一个表格，简单聊聊它们的特性与区别。

站服务器	开源	性能	安全性
Nginx	✔	较高	较高
Apache	✔	一般	一般
Tomcat	✔	一般	一般

综上所述，为了给网站营造一个长期稳定且安全高效的运行环境，推荐使用 Nginx 作为 站服务器 ， 毕竟其免费功能也能满足网站的日常使用。

方案：基于Nginx反向代理应用与服务

Nginx不仅具备反向代理、负载均衡、缓存服务等功能，还支持热部署。它几乎可做到 7*24 小时不间断运行，即使运行几个月也无需重启，还能在不间断运行的情况下热更新软件版本。 Nginx 主打高性能，其占用内存少、并发能力强、能支持高达 50000 个并发连接数，满足日常使用不在话下。

对于开发者来说 Node 并不陌生， Nginx 与 Node 有着很多相似理念，例如 HTTP服务器 、 事件驱动 、 非阻塞式I/O模型 等，而 Nginx 很多功能使用 Node 也可实现，但两者并不冲突。 Nginx 擅长底层服务器资源的处理， Node 擅长顶层具体业务逻辑的处理，两者可完美组合互相助力。

安装

打开 CMD工具 ， 登录服务器。第5章使用 yum 安装了 Nginx ， 执行 nginx -v ， 发现其版本不是最新版本。 CentOS8 自带的 Nginx版本 是 1.14 ， 因此自行创建高版本的 Nginx源 ， 在 yum install nginx 时才会安装最新版本的 Nginx 。

执行 vim /etc/yum.repos.d/nginx.repo ， 加入以下内容。

```
[nginx-stable]
name=nginx stable repo
baseurl=https://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
```

```
module_hotfixes=true

[nginx-mainline]
name=nginx mainline repo
baseurl=https://nginx.org/packages/mainline/centos/$releasever/$basearch/
gpgcheck=1
enabled=0
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
```

执行 `yum info nginx` 查看最新版本。

```
上次元数据过期检查：1:09:43 前，执行于 2022年05月29日 星期日 04时56分26秒。
已安装的软件包 # 当前版本
名称         : nginx
时期         : 1
版本         : 1.14.0
...

可安装的软件包 # 最新版本
名称         : nginx
时期         : 1
版本         : 1.22.0
...
```

再执行 `yum install nginx` 重新安装 `Nginx` ，再执行 `nginx -v` ，输出版本与上述信息一样表示更新成功。

打开 `FTP工具` ，发现很多 `Nginx配置文件` 存放在 `/ect/nginx` 目录中。虽然配置文件众多，但只需关注 `/etc/nginx/conf.d` 目录，该目录用于存放用户自定义的子配置文件。

主配置文件 `nginx.conf` 中有一行 `include /etc/nginx/conf.d/*.conf` 代码，其作用是当 `Nginx` 运行时，主配置文件会默认加载 `/etc/nginx/conf.d` 目录中所有子配置文件。

操作

启动 `Nginx` 后，得益其安全稳定的特性，若未遇到特殊情况几乎都不会再重启，只需掌握以下命令就能操作 `nginx` 。

命令	功能
nginx	启动进程
nginx -t	验证配置
nginx -s reload	重启进程
nginx -s stop	杀掉进程
`ps -ef | grep nginx`	查看进程

语法

Nginx 主配置文件是 /etc/nginx/nginx.conf ，可用 vim /etc/nginx/nginx.conf 查看配置。以下是 nginx.conf 的主体结构。

```txt
nginx.conf # 全局配置
├── events # 配置影响：Nginx服务器与用户的网络连接
├── http   # 配置功能：代理、缓存、日志等功能
│   ├── upstream # 配置后端地址：负载均衡不可或缺的部分
│   ├── server   # 配置虚拟主机：一个http块可包括多个server块
│   ├── server
│   │   ├── location # 一个server块可包括多个location块
│   │   ├── location # location块指令用于匹配URI
│   │   └── ...
│   └── ...
└── ...
```

- 配置文件由 指令 与 指令块 组成
- 指令以 分号 结尾，指令与参数以 空格 分隔
- 指令块以 大括号 将多条指令组织在一起
- 使用 $ 表示变量，提高复用性
- 使用 # 加入注释，提高可读性
- 部分指令的参数支持正则表达式
- include 语句允许组合多个配置文件以提升配置的可维护性

Nginx变量 众多，可查看Nginx预定义变量，在此不深入讲述了。

防火墙

启动 Nginx 后，若 CentOS 开启防火墙，还需在防火墙中加入需开放的端口让网站正常访问。 CentOS 默认安装了 firewalld ，可操作 firewalld 控制指定端口是否开放。

firewalld 的基本使用可通过 systemctl 管理。 systemctl 是 CentOS 服务管理工具中的主要工具，其融合 service 与 chkconfig 的功能于一体，只需掌握以下命令就能操作 firewalld 。

命令	功能
systemctl start firewalld	开启防火墙
systemctl stop firewalld	关闭防火墙
systemctl status firewalld	查看防火墙状态
systemctl disable firewalld	开机禁用防火墙
systemctl enable firewalld	开机启用防火墙

那如何开放一个指定端口？跟着我的步伐，通过执行以下命令就可开放或关闭一个指定端口，以 9999 端口为例。

- 查看防火墙版本，确保防火墙已安装： firewall-cmd --version
- 查看防火墙状态，确保防火墙已开启： firewall-cmd --state
- 查看所有打开的端口，若端口已开放则无需继续执行命令： firewall-cmd --zone=public --list-ports
- 开放指定端口： firewall-cmd --zone=public --add-port=9999/tcp --permanent
- 重载防火墙配置： firewall-cmd --reload
- 查看指定端口是否已开放： firewall-cmd --zone=public --query-port=9999/tcp
- 删除指定端口： firewall-cmd --zone=public --remove-port=9999/tcp --permanent

开放指定端口时追加 `--permanent` 表示开放端口永久生效，无该参数则重启防火墙会失效。在默认情况下， `CentOS` 的防火墙是关闭的，因此无需主动开放指定端口。

应用：解锁Nginx各种姿势

以下示例及后续章节涉及的域名 `yangzw.vip` ，记得改成你的域名哈！

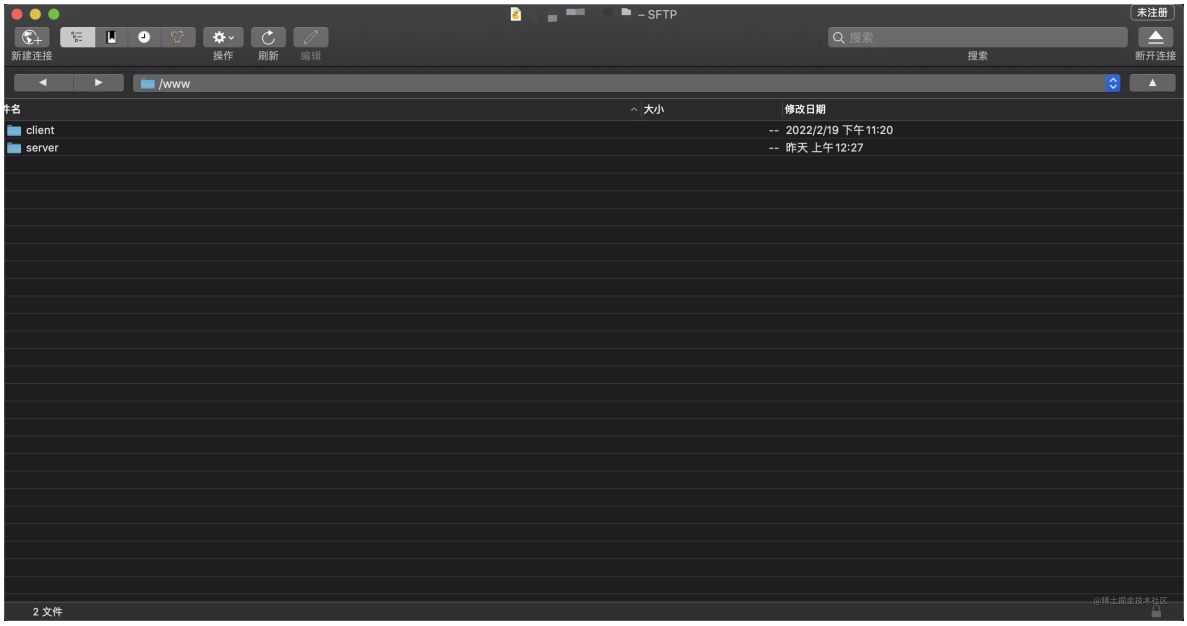
以下示例及后续章节涉及的域名 `yangzw.vip` ，记得改成你的域名哈！

以下示例及后续章节涉及的域名 `yangzw.vip` ，记得改成你的域名哈！

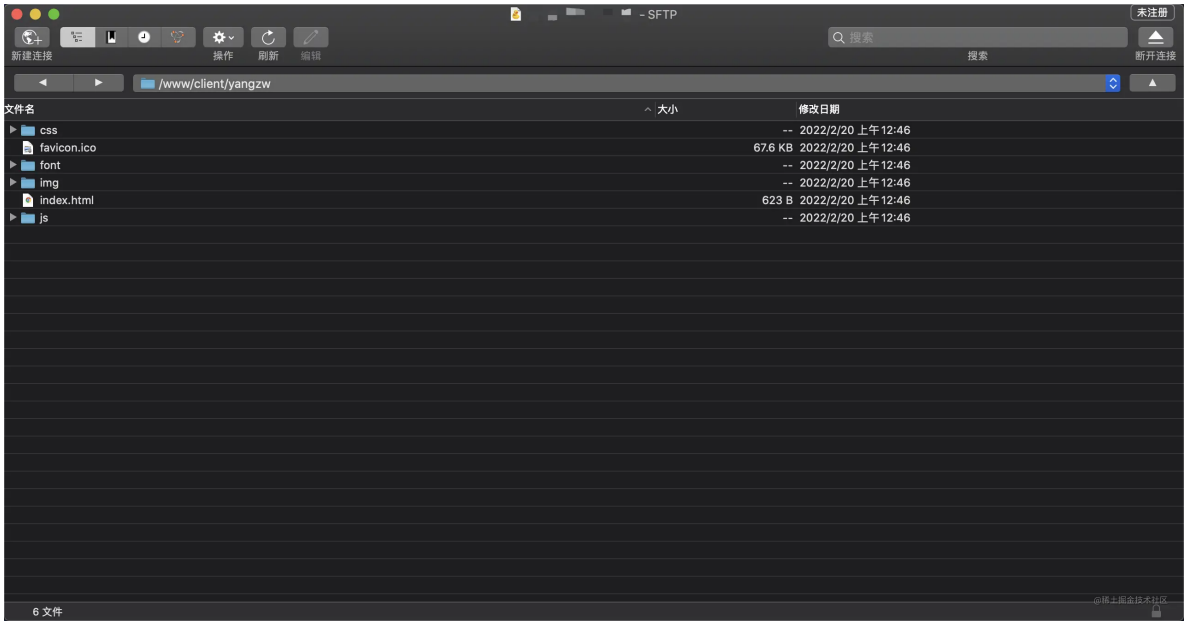
重要事情只说三遍，后续我不再提示把每章的 `yangzw.vip` 改成你的域名了。

个人官网

在根目录中创建 `www` 文件夹，该文件夹中再创建两个文件夹，分别是 `client` 与 `server` 。 `client` 用于存放 `Web应用源码` ， `server` 用于存放 `Node应用源码` 。



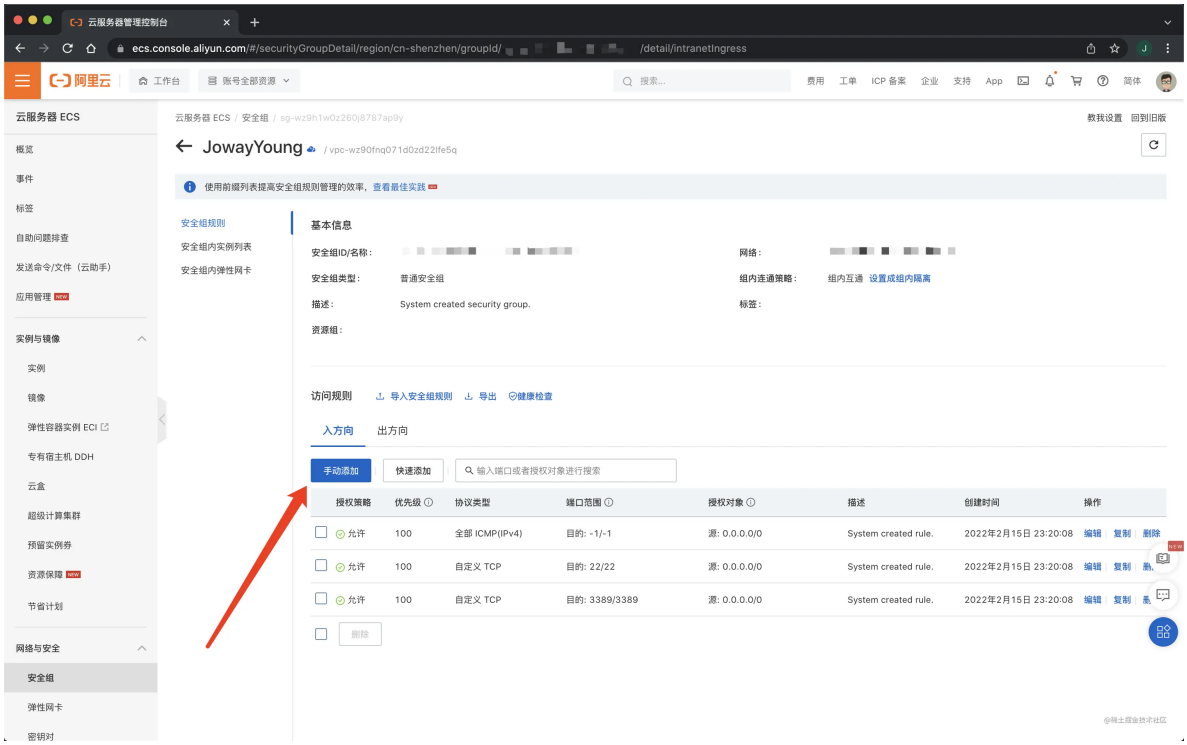
准备好个人官网源码，在 `client` 文件夹中创建 `yangzw` 文件夹，把源码拖到 `yangzw` 中，最终的入口文件路径是 `/www/client/yangzw/index.html` 。



在 `/etc/nginx/conf.d` 目录中创建 `yangzw.vip.conf` 文件，执行 `vim /etc/nginx/conf.d/yangzw.vip.conf` ，加入以下内容。

```
server {
        listen 80;
        server_name yangzw.vip www.yangzw.vip;
        location / {
                root /www/client/yangzw;
                index index.html;
        }
}
```

打开阿里云官网，选择 右上角的控制台 → 左上角的菜单 → 云服务器ECS → 安全组 → 配置规则 。再点击 手动添加 ，增加安全组的配置。



手动增加 `80` 端口。执行 `nginx -t` 验证 `Nginx` 配置，再执行 `nginx -s reload` 重启 `Nginx` 进程。在浏览器地址栏中输入yangzw.vip，就可正常访问个人官网了。

域名映射

通过第6章的方式在服务器中配置一个全新的 `static.yangzw.vip` 二级域名，该域名用于托管所有 静态资源 。

在 `www` 文件夹中创建 `static` 文件夹，该文件夹用于存放静态资源。

在 `/etc/nginx/conf.d` 目录中创建 `static.yangzw.vip.conf` 文件，执行 `vim /etc/nginx/conf.d/static.yangzw.vip.conf` ，加入以下内容。

```
server {
        listen 80;
        server_name static.yangzw.vip;
        location / {
                root /www/static;
                index index.html;
        }
}
```

在 `/www/static` 目录中创建 `index.html` 文件，随便编码。

编辑完毕执行 `nginx -t` 验证 `Nginx` 配置，再执行 `nginx -s reload` 重启 `Nginx` 进程。在浏览器地址栏中输入static.yangzw.vip，就可直接跳转到刚才创建的 `/www/static` 目录中。

通过上述方式可映射各种资源文件，例如在 `/www/static` 目录中创建 `img` 文件夹并复制一张图像 `nice.png` ，可通过 `http://static.yangzw.vip/img/nice.png` 访问该图像了。

跨域问题

当产生跨域时，配置 `Nginx` 处理该问题，就可无感知地继续保持开发进度，无需把实际访问的后端地址改成前端地址，这样适用性更高。

例如 `nice.yangzw.vip` 访问 `static.yangzw.vip` ，根据同源策略的定义，协议域名端口其一不同就是跨域。可通过配置 `Nginx` 的 `header` 解决跨域问题，继续以上述配置文件为例。

```
server {
        listen 80;
        server_name static.yangzw.vip;
        # 新增部分-开始
        add_header "Access-Control-Allow-Origin" $http_origin; # 当前请求域名，不支持携带C
        add_header "Access-Control-Allow-Credentials" "true"; # 请求可携带Cookie
        add_header "Access-Control-Allow-Methods" "GET, POST, OPTIONS"; # 允许的请求方式
        add_header "Access-Control-Allow-Headers" $http_access_control_request_headers;
        add_header "Access-Control-Expose-Headers" "Content-Length,Content-Range";
        if ($request_method = "OPTIONS") {
                add_header "Access-Control-Max-Age" 18000000; # 请求的有效期：在有效期内无
                add_header "Content-Length" 0;
                add_header "Content-Type" "text/plain; charset=utf-8";
        }
        # 新增部分-结束
        location / {
                root /www/static;
                index index.html;
        }
}
```

编辑完毕执行 `nginx -t` 验证 `Nginx` 配置，再执行 `nginx -s reload` 重启 `Nginx` 进程。跨域问题完美解决。

动静分离

目前动静分离的方式有两种解决方案。在生产条件充足的情况下，还是推荐使用第一种方案，当然第二种方案也能满足个人官网做动静分离的需求。

- [x] 将静态资源存放到独立的二级域名中，例如上述创建的 `static.yangzw.vip`
- [x] 动态跟静态资源混合发布，通过 `Nginx` 配置区分

若使用 `Nginx` 配置动静分离，就无需使用二级域名了。上述在 `www` 文件夹中创建的 `client` 文件夹用于存放 `Web` 应用源码，创建的 `static` 文件夹用于存放静态资源。

```
server {
        listen 80;
        location / {
                root /www/client; # 存放动态资源(Web应用)
                index index.html;
        }
        location /static/ {
                root /www/staic; # 存放静态资源
                autoindex on; # 开启资源目录
        }
}
```

编辑完毕执行 `nginx -t` 验证 `Nginx` 配置，再执行 `nginx -s reload` 重启 `Nginx` 进程。动静分离完美解决。

反向代理

`反向代理` 经常被用于处理跨域问题，以下问题就是最直接的体现。

- 将请求转发到本机的另一个服务中
- 根据访问路径跳转到不同端口的服务中

创建二级域名 `api.yangzw.vip` 用于管理接口，在服务器的 `9999` 端口运行一个提供接口功能的 `Node服务` 。现在需将两者关联在一起，即在客户端中通过 `api.yangzw.vip` 可访问到服务端中 `127.0.0.1:9999` 的接口功能。

在 `/etc/nginx/conf.d` 目录中创建 `api.yangzw.vip.conf` 文件，执行 `vim /etc/nginx/conf.d/api.yangzw.vip.conf` ，加入以下内容。

```
server {
        listen 80;
        server_name api.yangzw.vip;
        location / {
                proxy_pass http://127.0.0.1:9999;
        }
}
```

编辑完毕执行 `nginx -t` 验证 `Nginx` 配置，再执行 `nginx -s reload` 重启 `Nginx` 进程。反向代理完美解决。

负载均衡

`负载均衡` 是把负载均匀合理地分发到多个服务器中，实现压力分流的作用。

`Nginx` 提供以下 `负载均衡` 方式，默认为 `轮询` 。

- **轮询**：无需配置，每个请求根据时间顺序逐一分配到不同服务器，若其中一个服务挂了会自动被剔除
- **weight**：根据权重分配，指定每个服务器的轮询几率，权重越高其被访问的概率越大，可解决服务器性能不均的问题
- **ip_hash**：根据访问 `IP` 的 `Hash结果` 分配，每个访客固定访问一个服务器，可解决动态网页 `Session共享` 的问题
- **fair**：根据服务器响应时间分配，响应时间短的服务器会优先分配，需安装 `nginx-upstream-fair`

```
http {
        upstream api.yangzw.vip {
                # ip_hash; # IpHash方式
                # fair; # Fair方式
                server 127.0.0.1:9999; # 负载均衡目的服务地址：可设置多个服务器
                server 127.0.0.1:8888;
                server 127.0.0.1:7777 weight=10; # 配置权重：不配置默认为1
        }
        server {
                location / {
                        proxy_pass api.yangzw.vip;
                        proxy_connect_timeout 10;
                }
        }
}
```

编辑完毕执行 `nginx -t` 验证 Nginx 配置，再执行 `nginx -s reload` 重启 Nginx 进程。负载均衡完美解决。

总结

Nginx 在 前端工程化 中是一个很有分量的工具，其作为 站服务器 能解决很多请求或代理的网络问题。在后续实战中还会继续结合项目衍生 Nginx 的其他应用场景，如 深度压缩 、 终端适配 、 图像防盗 、 路由刷新 、 请求过滤 、 文件缓存 等应用场景。

当然本章内容也是作为后续章节的基础，会配合更多工具完善 Nginx 的进阶，建议你好好拿上述配置流程练手，以熟悉 Nginx 的语法结构与配置技巧。

本章内容到此为止，希望能对你有所启发，欢迎你把自己的学习心得打到评论区！

- [x] 示例项目：fe-engineering
- [x] 正式项目：bruce

留言

输入评论（Enter换行，Ctrl + Enter发送）

发表评论

全部评论（14）

伽蓝 JY.3 前端开发 2月前

ftp 如何连接远程的啊?

点赞 回复

btbrad JY.3 搬砖 @ 沪郊某工地 2月前

Serviet? servlet?

点赞 1

JowayYoung （作者） 2月前

笔误了，稍后修复下

点赞 回复

爱敲代码 JY.3 2月前

请问大佬跨域问题 这块儿内容层级写的是不是有些问题？ 能否将有出入的内容更新下?

点赞 1

NewName 2月前

确实, 我试了一下也不好使， 放到location / 下面好使了

点赞 回复

静听花亦落 JY.3 老年前端 @ 阿巴阿巴 2月前

找到了一些报错的地方，我自己百度了半天（对ngnix一窍不通），给大家填个坑：
1 跨域配置，if不能放在server里，会报错，我是放到了location里（不知道对不对，等作者），反正不报错了
2 负载均衡那里，如果放到api.xxx.conf中，需要把http拿掉，而且少了点东西
proxy_pass http:// api.yangzw.vip; http:// 不加会报错
3 还是负载均衡，一开始api.conf里配的反向代理到了负载均衡里 是需要保留还是直接...

展开

点赞 1

JowayYoung （作者） 2月前

由于是手搓代码，可能存在一些错误，大家遇到问题可以像这位同学这样先留言，我整改好后再通知大家

点赞 回复

爱敲代码 JY.3 3月前

请问大佬 负载均衡 nginx 自定义自配置文件 是api.yangzw.vip吧？ 需要加上server_name吗?

点赞 1

JowayYoung （作者） 3月前

要的，我这里忘记加了

点赞 回复

爱敲代码 JY.3 3月前

请问大佬 动静分离 server_name 是 yangzw.vip www.yangzw.vip吧?

点赞 1

JowayYoung （作者） 3月前

是的，就是主域名

点赞 回复

CarelessHim LV.1 JY.4 高级前端CV工程师 4月前

防火墙那些命令里的 --zone=public是什么作用呢

点赞 2

CarelessHim 4月前

我之前配置防火墙也用了那几个query和add的命令，但是没有加这个--zone=public，也实现了同样的效果，好奇这个属性是做什么用的

点赞 回复

JowayYoung （作者） 3月前

提供工作作用域

点赞 回复