1. **Nginx简介**

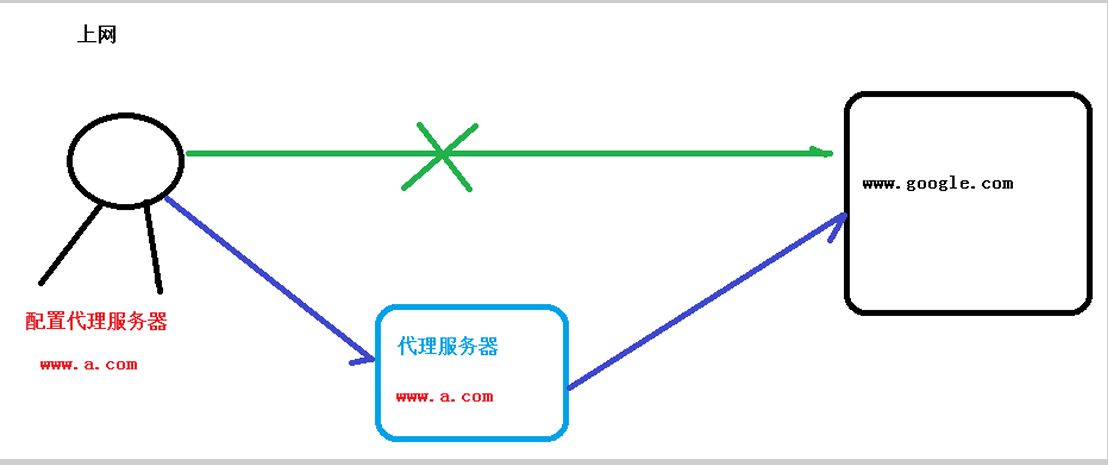
**1.Nginx概述**

Nginx (“engine x”) 是一个高性能的 HTTP 和 反向代理服务器，特点是占有内存少，并发能力强，能经受高负载的考验,有报告表明能支持高达50,000个并发连接数。

**2.正向代理**

正向代理：如果把局域网外的 Internet 想象成一个巨大的资源库，则局域网中的客户端要访问 Internet，则需要通过代理服务器来访问，这种代理服务就称为正向代理。

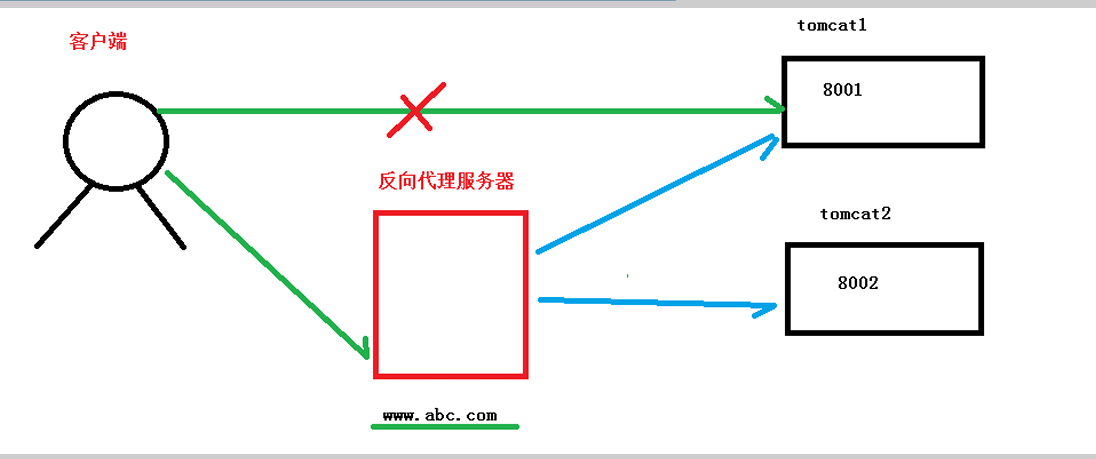
**【在浏览器配置代理服务器，通过代理服务器去访问这个网址】**



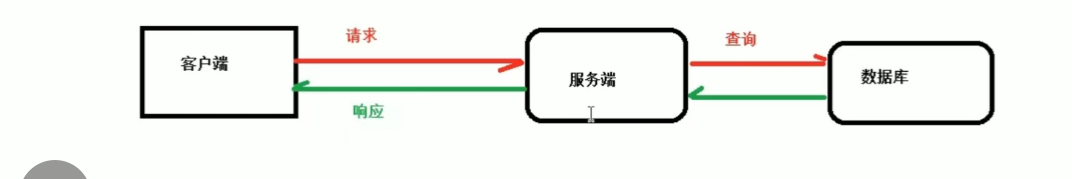
**3.反向代理**

反向代理，**其实客户端对代理是无感知的（和正向代理的区别，刚刚说正向代理是需要在客户端浏览器中配置代理服务器）**，因为客户端不需要任何配置就可以访问，我们只需要将请求发送到反向代理服务器，由反向代理服务器（9001）去选择目标服务器（8001）获取数据后，在返回给客户端，此时反向代理服务器和目标服务器对外就是一个服务器，暴露的是代理服务器地址（9001），隐藏了真实服务器 IP 地址（8001）。

反向代理服务器比如端口号是9001，真正的内部服务器是8001，但是客户端不知道，把后面的tomcat和反向代理服务器当作一个整体。

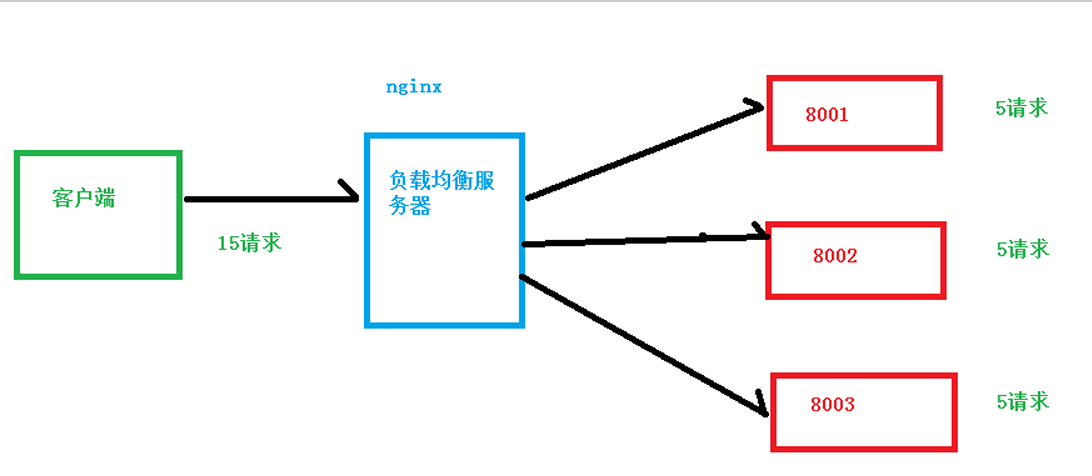
**4.负载均衡**

        一般请求



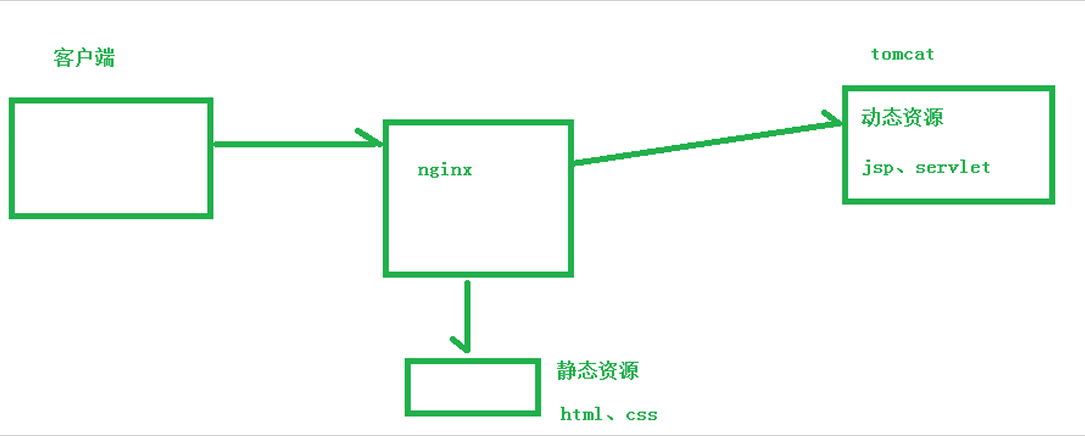
单个服务器解决不了，我们增加服务器的数量，然后将请求分发到各个服务器上，将原先请求**集中到单个服务器的情况**改为将**请求分发到多个服务器上**，将负载分发到不同的服务器，也就是所谓的负载均衡。

**把15个请求平均分发到3个服务器，每个服务器5个请求。**



**5、动静分离**

     为了加快网站的解析速度，可以把**动态页面**和**静态页面**由不同的服务器来解析，加快解析速度，降低原来单个服务器的压力。



**二、Nginx安装**

我这边ngnix就在windows上面安装【一般安装在linux系统中才能发挥它最大的效果】

下载安装包

[nginx: download](https://nginx.org/en/download.html)

直接解压即可



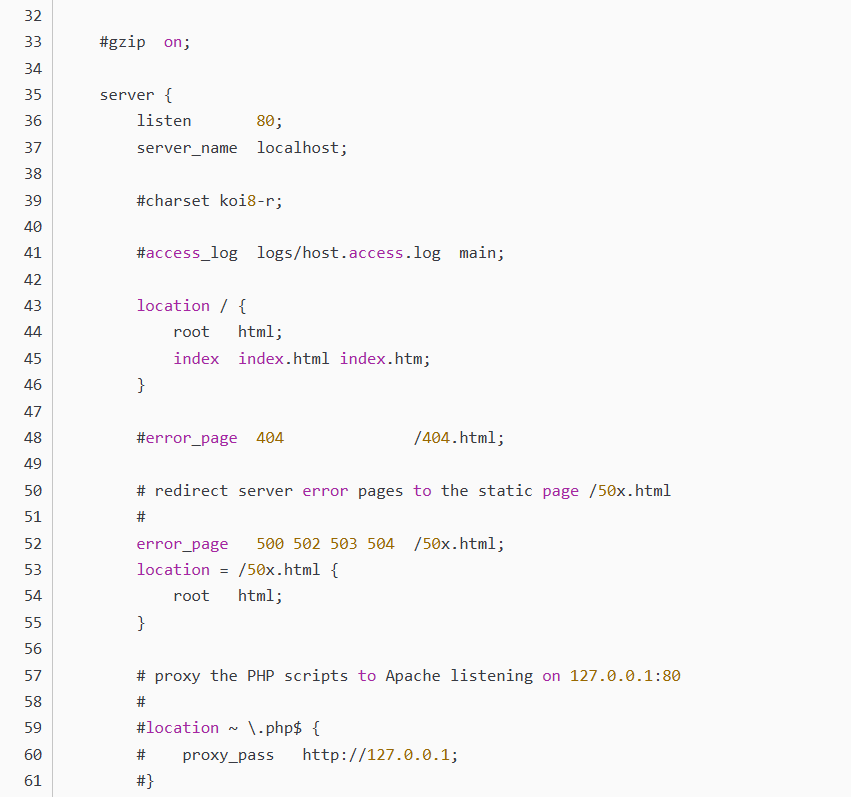
**三、Ngnix配置文件**

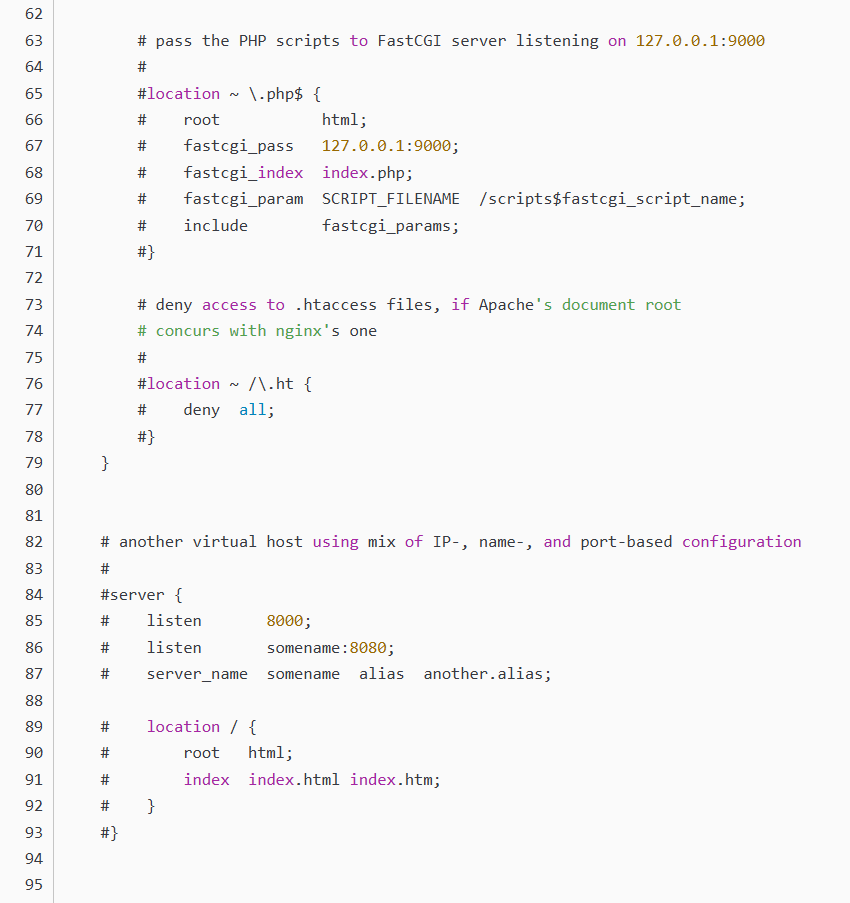
我们这边就直接看配置文件

D:\nginx-1.26.2\conf\ngnix.conf

一般打开会有一些内容，但是很多地方都有注释

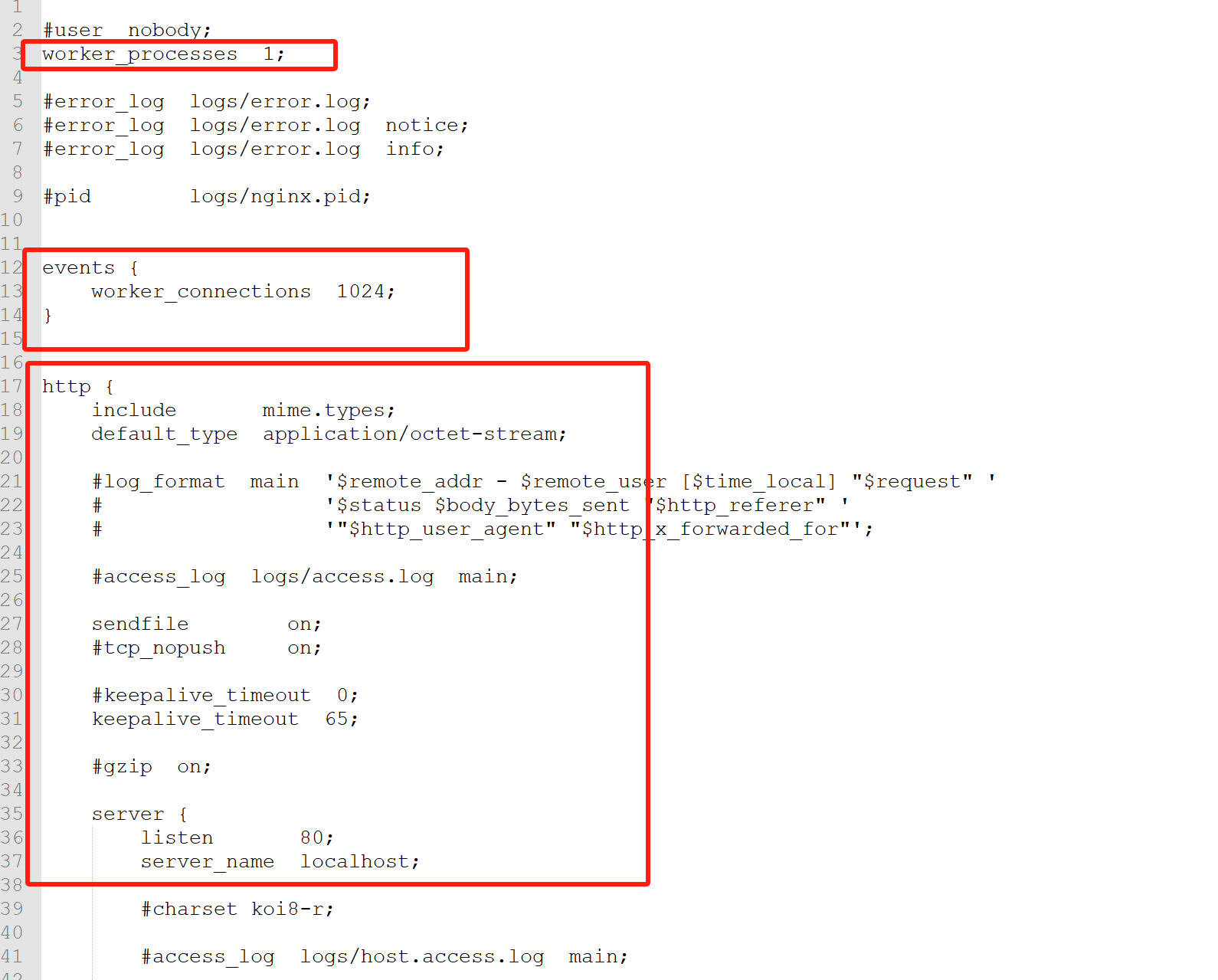








总共三部分组成，大致如下：



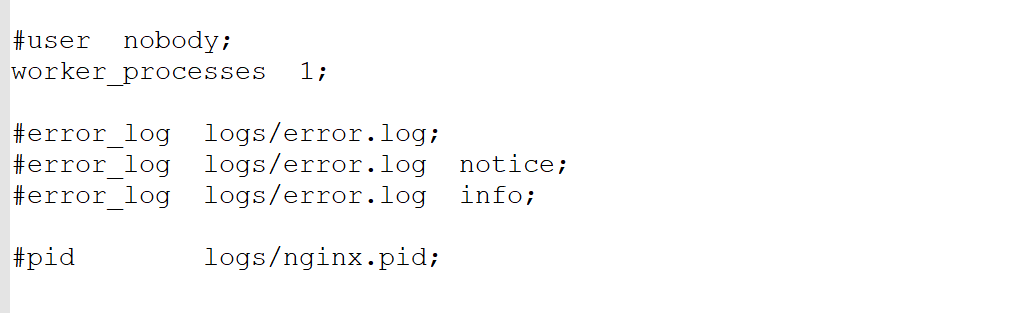
这是我去掉之前注释后剩下的部分：

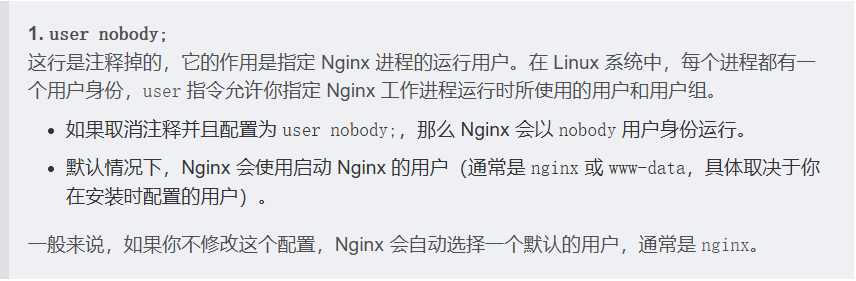


第一部分：全局块

从配置文件开始到 events 块之间的内容，主要会设置一些影响 nginx 服务器整体运行的配置指令，主要包括配置运行 Nginx 服务器的用户（组）、允许生成的 worker process 数，进程 PID 存放路径、日志存放路径和类型以及配置文件的引入等。

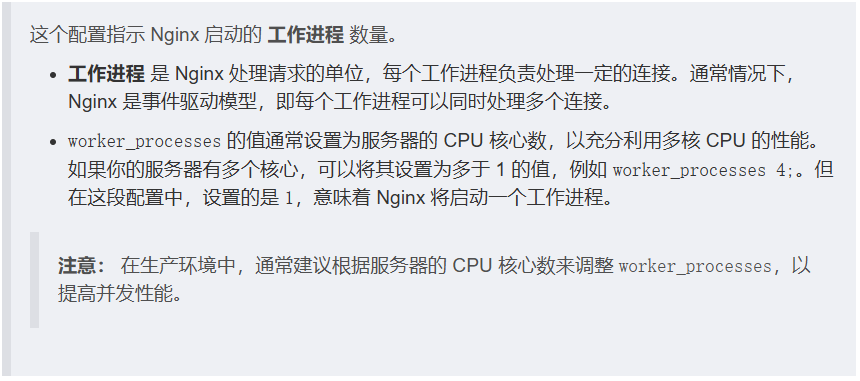
————————————————

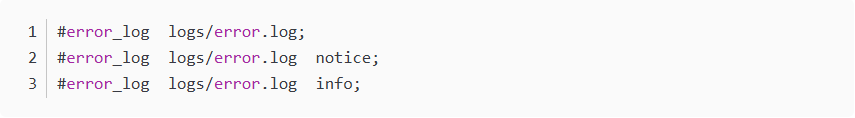




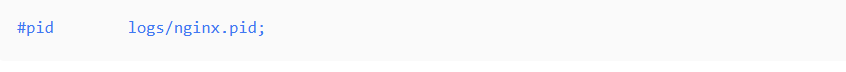


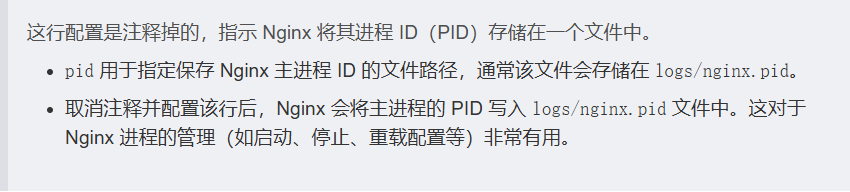
**这是Ngnix服务器并发处理服务的关键配置，worker\_processes值越大，可以支持的并发处理量越多，但是会受到硬件、软件等设备的制约。**

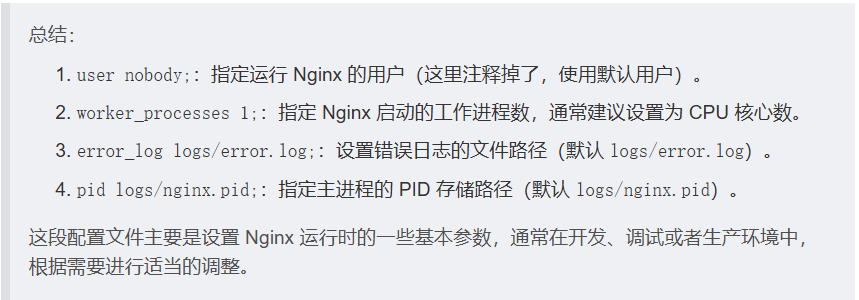








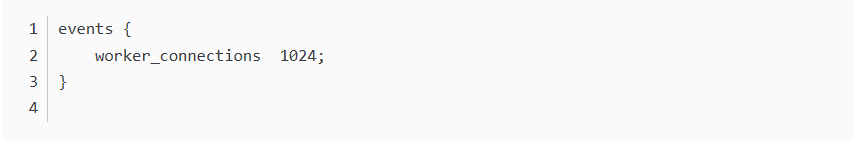




第二部分：events块

events 块涉及的指令主要影响 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 work process 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，选取哪种事件驱动模型来处理连接请求，每个 word process 可以同时支持的最大连接数等。

下图例子就表示每个 work process 支持的最大连接数为 1024. 这部分的配置对 Nginx 的性能影响较大，在实际中应该灵活配置。



第三部分：http块

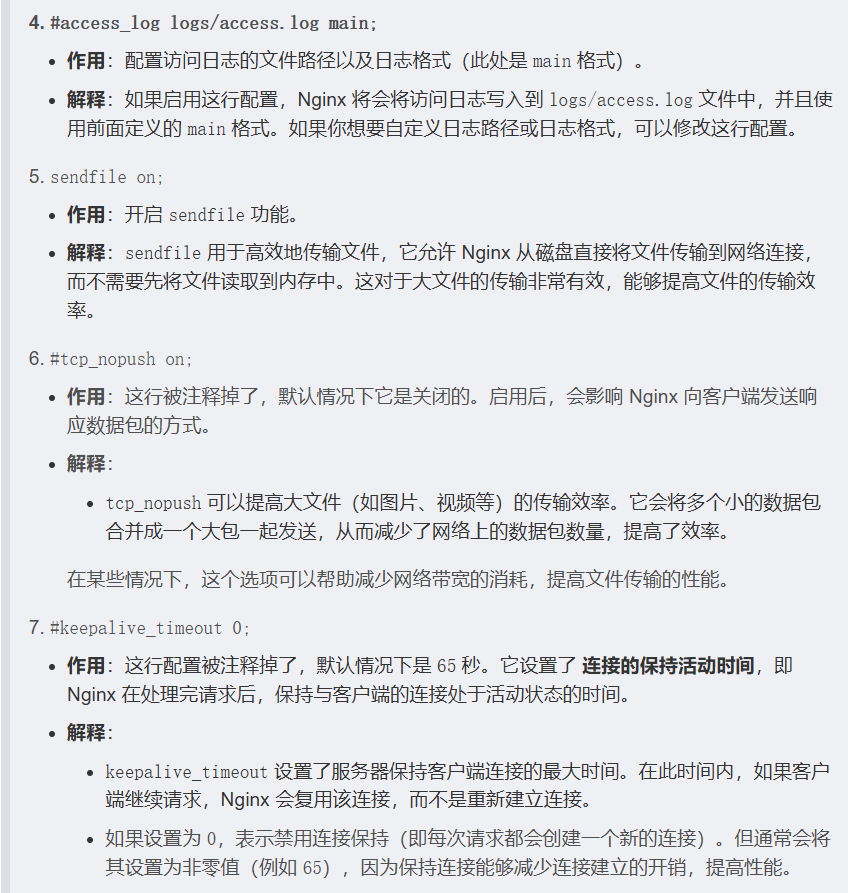
这算是 Nginx 服务器配置中最频繁的部分，代理、缓存和日志定义等绝大多数功能和第三方模块的配置都在这里。

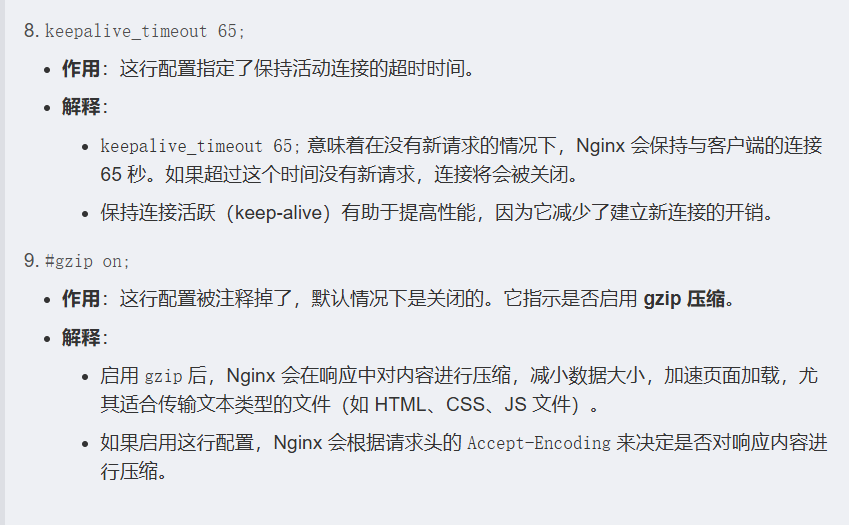
需要注意的是：http 块也可以包括 http 全局块、server 块。

①、http 全局块

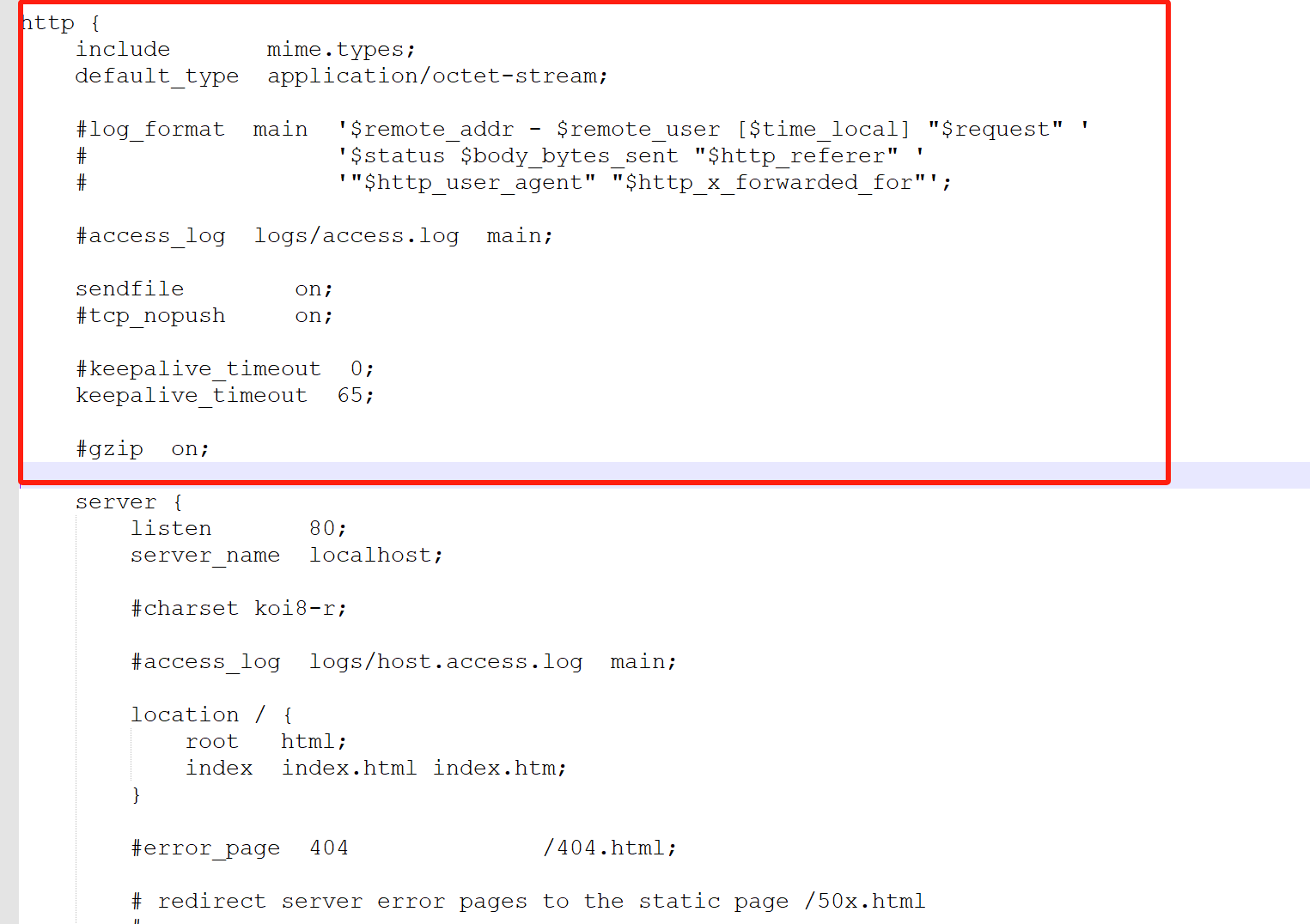
http 全局块配置的指令包括文件引入、MIME-TYPE 定义、日志自定义、连接超时时间、单链接请求数上限等。如下





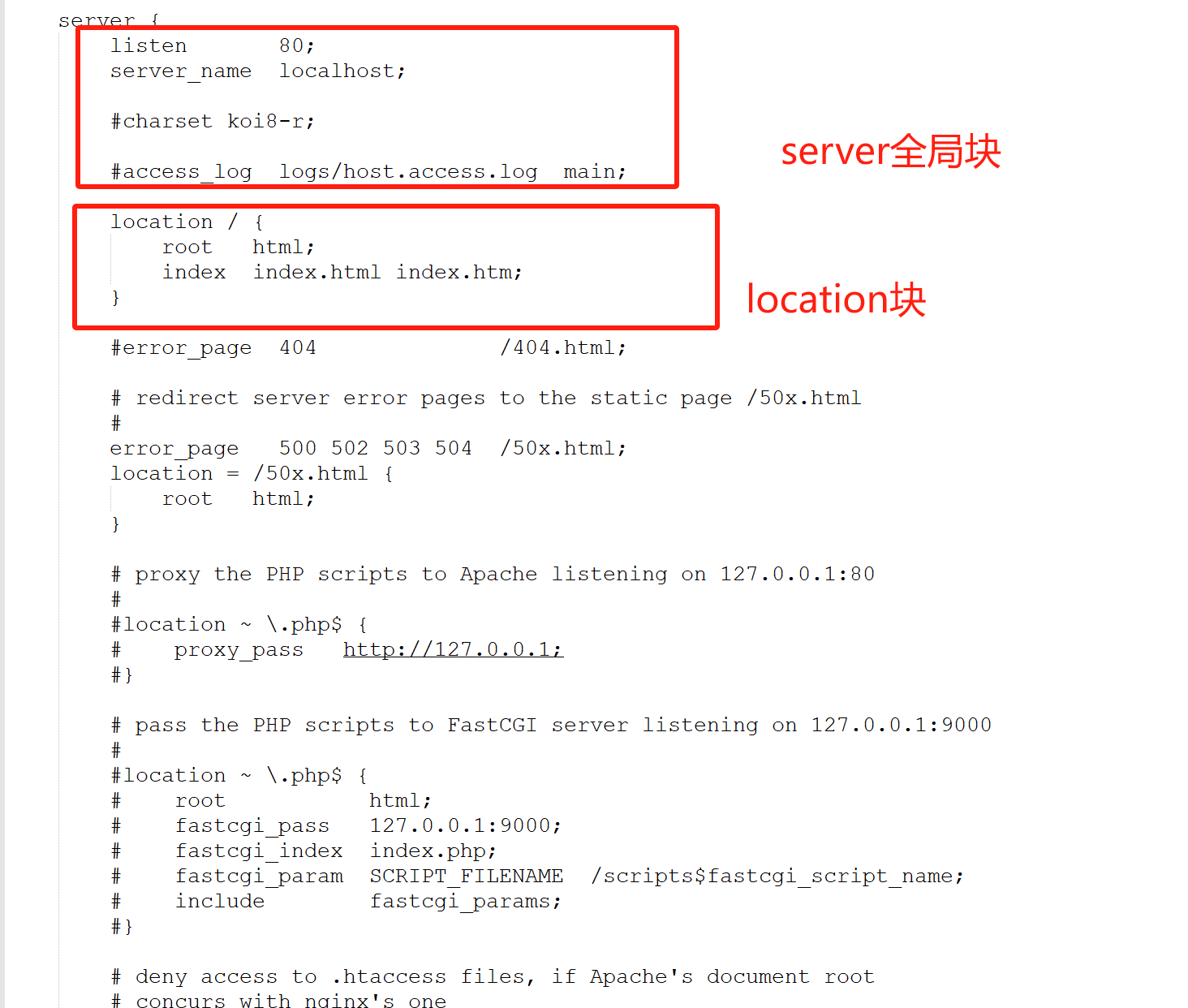


如下图



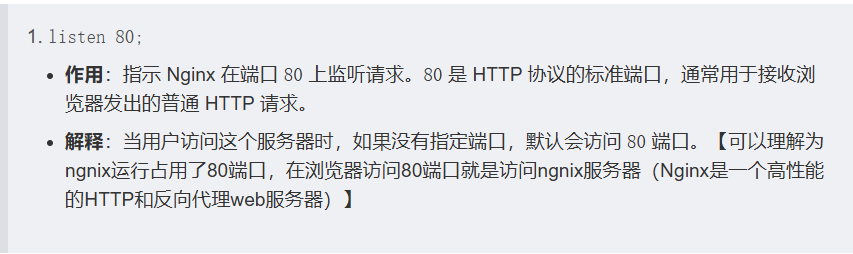
②、server 块

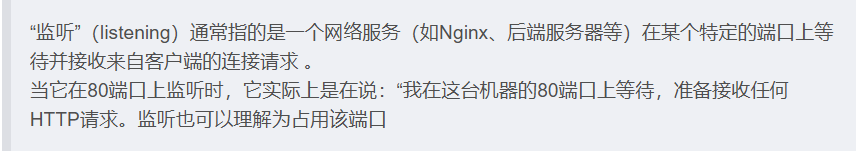
这块和虚拟主机有密切关系，虚拟主机从用户角度看，和一台独立的硬件主机是完全一样的，该技术的产生是为了节省互联网服务器硬件成本。每个 http 块可以包括多个 server 块，而每个 server 块就相当于一个虚拟主机。而每个 server 块也分为全局 server 块，以及可以同时包含多个 locaton 块。

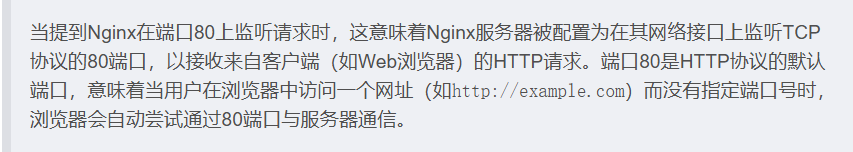


**1、全局 server 块**

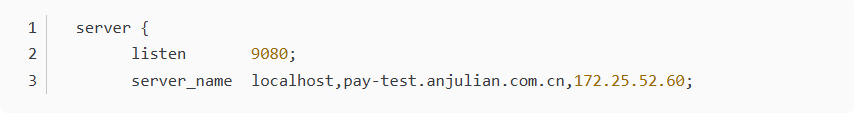
        最常见的配置是**本虚拟机主机的监听配置**和**本虚拟主机的名称**或 **IP 配置**。











也可能有这种配置：这个server\_name虽然有三个名字，不过都代表一台服务器。

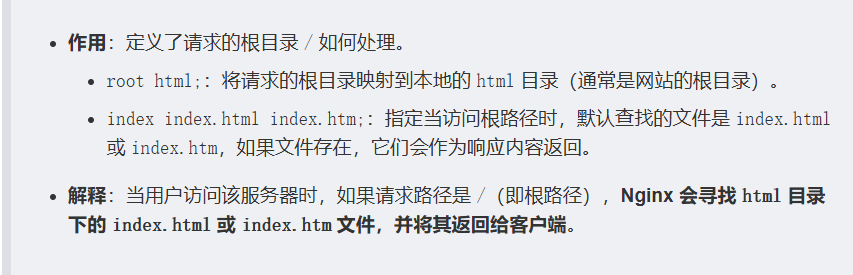


2、location 块

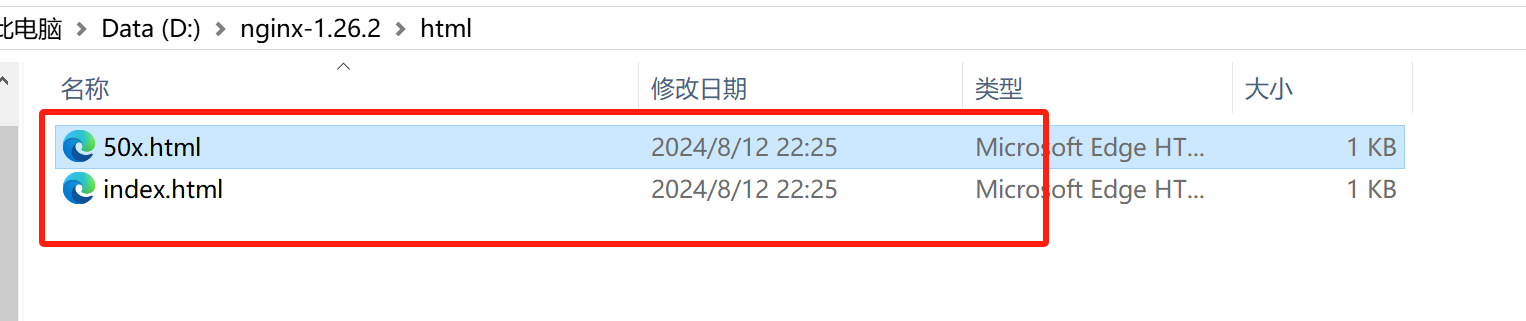
一个 server 块可以配置多个 location 块。

这块的主要作用是基于 Nginx 服务器接收到的请求字符串（例如 server\_name/uri-string），对虚拟主机名称（也可以是 IP 别名）之外的字符串（例如 前面的 /uri-string）进行匹配，对特定的请求进行处理。地址定向、数据缓存和应答控制等功能，还有许多第三方模块的配置也在这里进行。



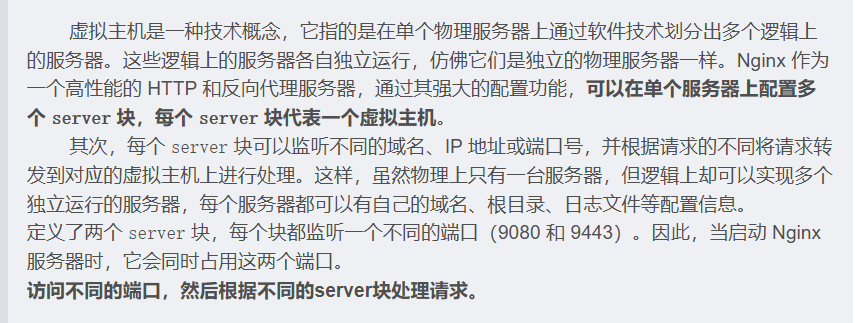






**后面注释的地方不用管了**

**③、总结：**







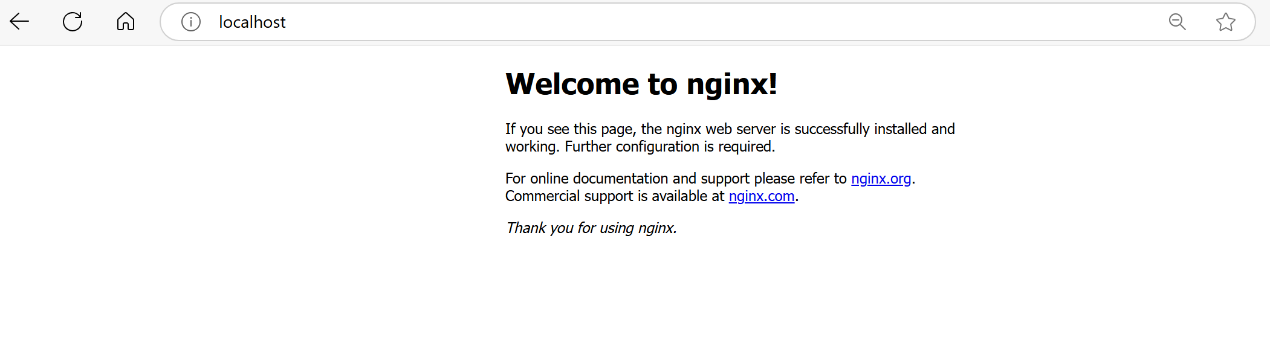
**四、Nginx启动**

我们先就默认配置启动，双击一下，然后如果看到画面一闪而过，代表启动成功



我们直接浏览器输入localhost 就可以进入，这个画面其实在前面说过，就是下面第二张图中html目录下面的index.html(默认配置就是访问的这个界面），如果启动失败那应该会访问50x.html。

**出现下面界面代表启动成功**





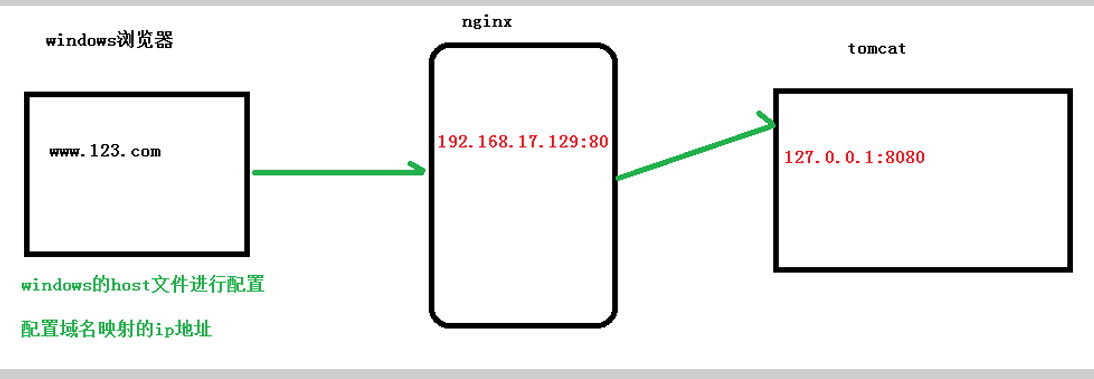
**五、Nginx配置实战**

**1、Nginx配置实例-反向代理**

**将请求转发到后端服务器**

**实例一**

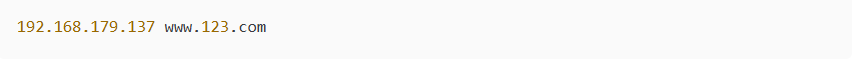
一般我们这里是用的打包后的前端项目，然后指定到后端sprinboot项目（自带tomcat）



[反向代理配置](https://so.csdn.net/so/search?q=%E5%8F%8D%E5%90%91%E4%BB%A3%E7%90%86%E9%85%8D%E7%BD%AE&spm=1001.2101.3001.7020)  
        修改host文件（地址直接参照以下图片）

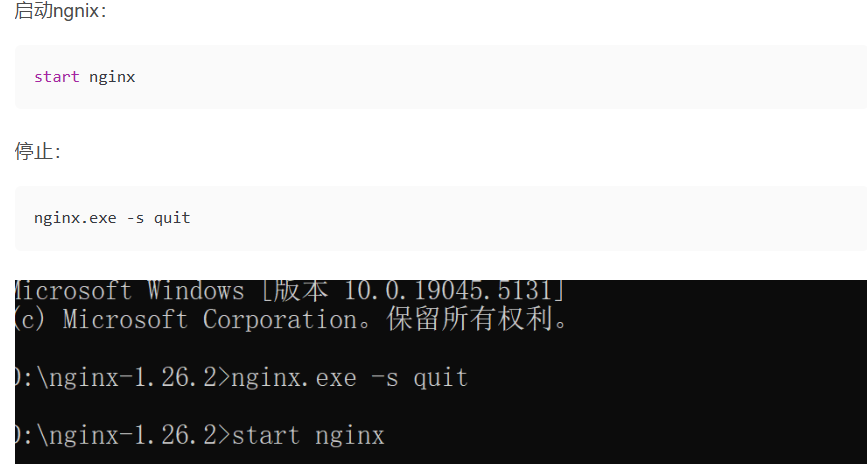


   加上一行：将www.123.com映射到192.168.179.137（ip是Linux地址）



**因为我这里是window是所以这里就不用了**





总结：

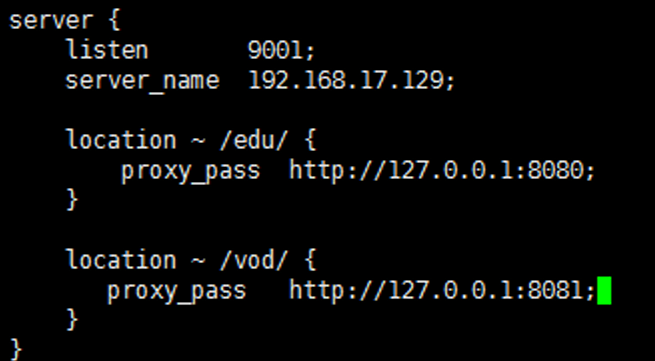
**因为我们这里是测试的反向代理，我这里本地启动了一个springbooot服务、localhost:8080，我在浏览器输入这个即可进入到一个界面，然后我用ngnix做反向代理，我输入的是localhost:80即可进入这个界面**

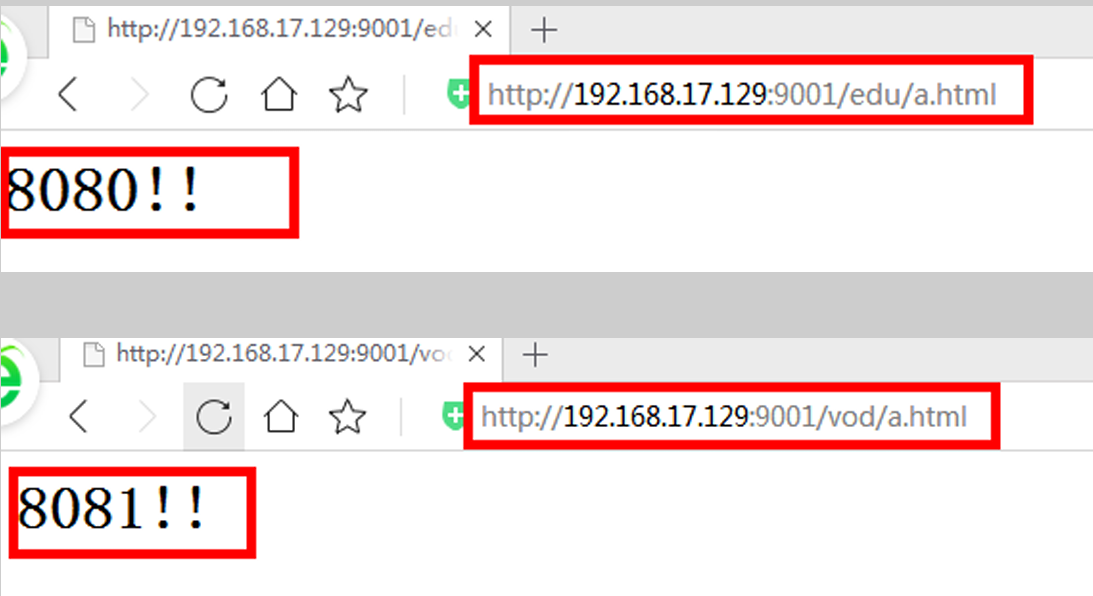
实例二

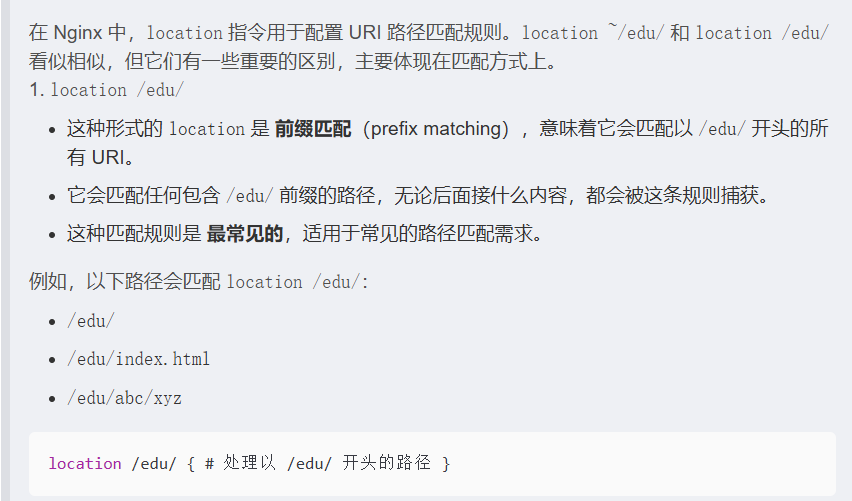
这里就描述一下：

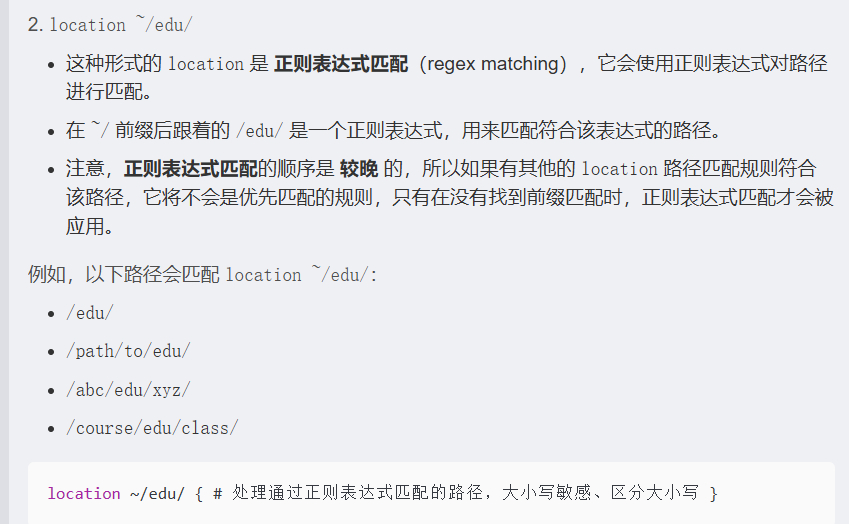
（1）准备两个tomcat服务器，一个8080端口，一个8081端口

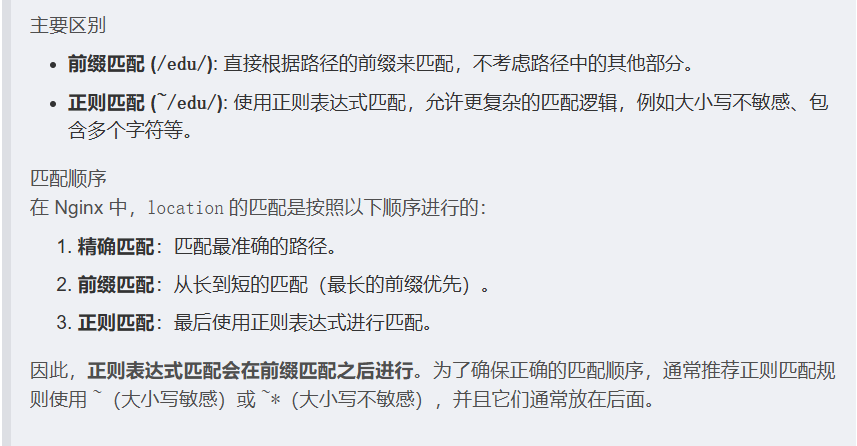
~代表正则表达式的形式，有edu就8080，有vod就8081







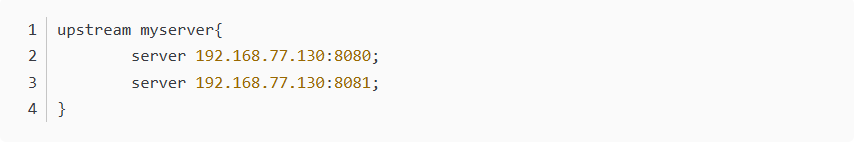




**2、Nginx负载均衡**

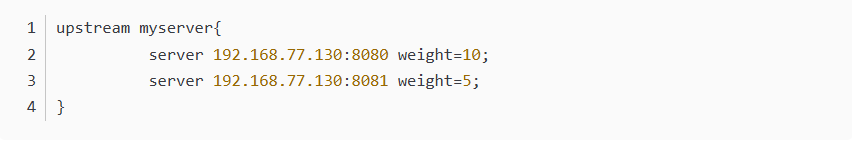


**大概是实现效果就是每次请求80端口刷新，轮询请求一个服务器，可能是8080，可能是8081**  
      
1、轮询（默认）  
        每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器 down 掉，能自动剔除。

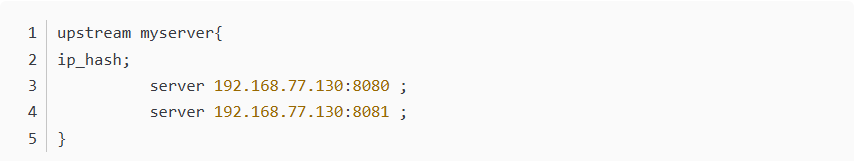


 2、weight  
        weight 代表权重默认为 1,权重越高被分配的客户端越多 。

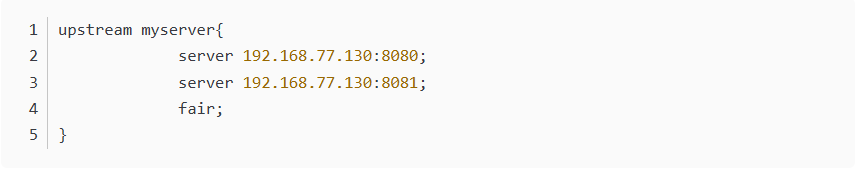
        指定轮询几率，weight 和访问比率成正比，用于后端服务器性能不均的情况。 例如：



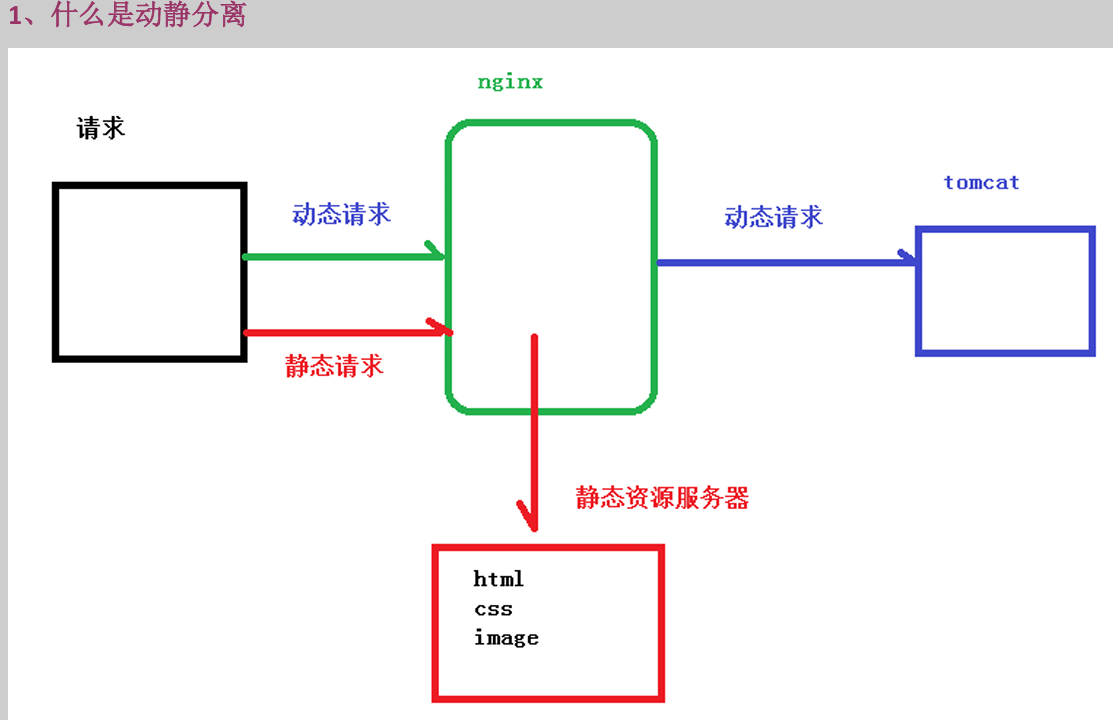
3、 ip\_hash  
     每个请求按访问 ip 的 hash 结果分配，这样每个访客固定访问一个后端服务器，可以解决 session 的问题。也就是说，客户第一次访问这个服务器，之后访问的都是这个服务器。



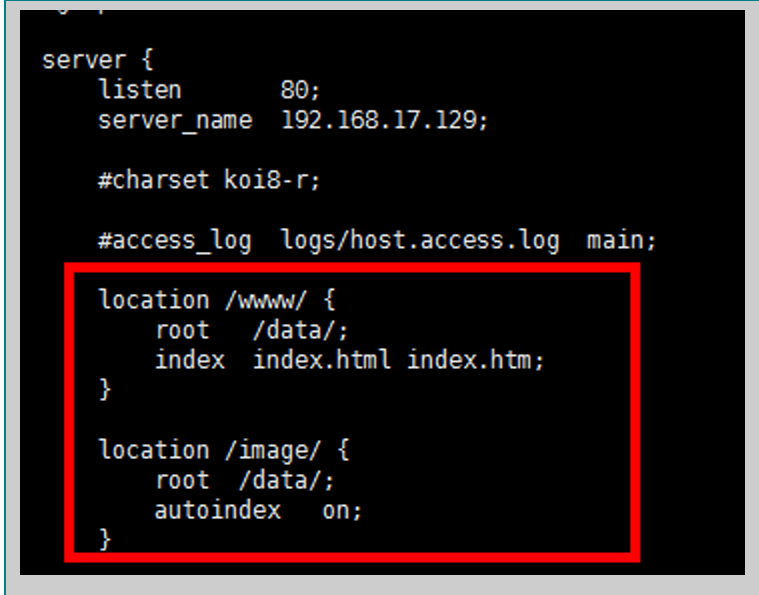
4、fair  
       按后端服务器的响应时间来分配请求，响应时间短的优先分配。



**3、Nginx动静分离**



**实现效果:不同资源访问不同服务器。**



一般在linux系统中，这个root就是根目录，所以可以直接用data

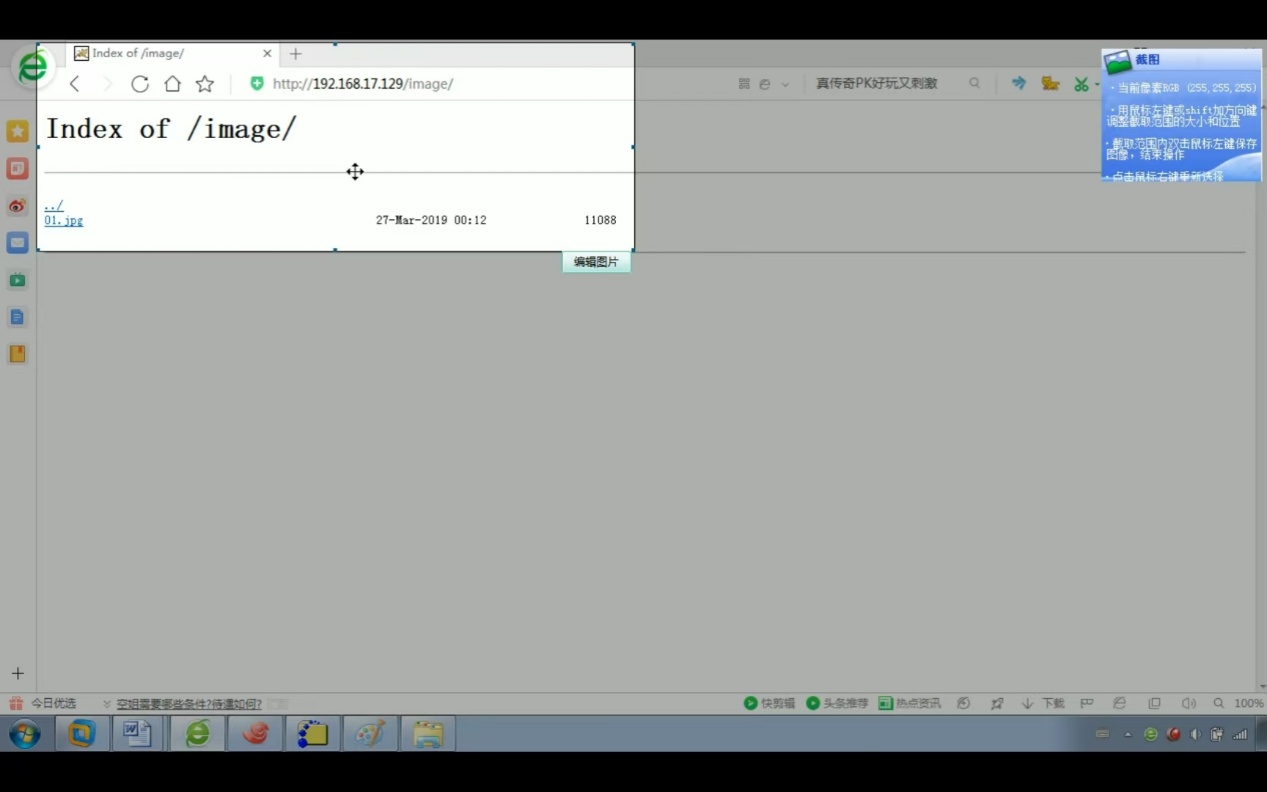
**如果是在windows系统中，之前有一个默认示例，就是html中的静态页面，root应该就是从nginx的目录下开始**

准备工作

在/data里面准备两个文件夹/image和/www，里面分别放入01.jpg和a.html来模拟静态资源。

Niginx具体配置

autoindex on：目的是为了在访问 /image 时，能够显示目录里面的内容，当然这里也可以通过expire设置缓存过期时间 。

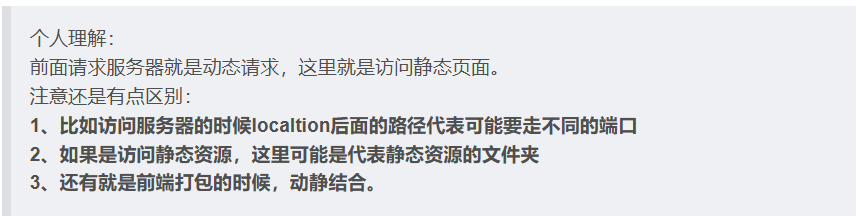


 或者直接

http://192.168.17.129/image/01.jpg

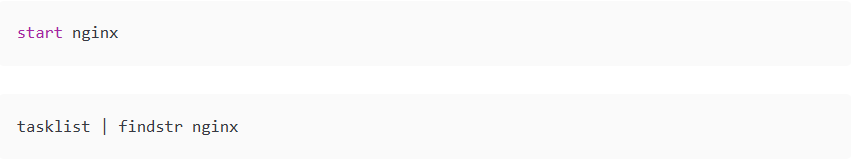
如果是访问http://192.168.17.129/www/没有autoindex on，所以是404

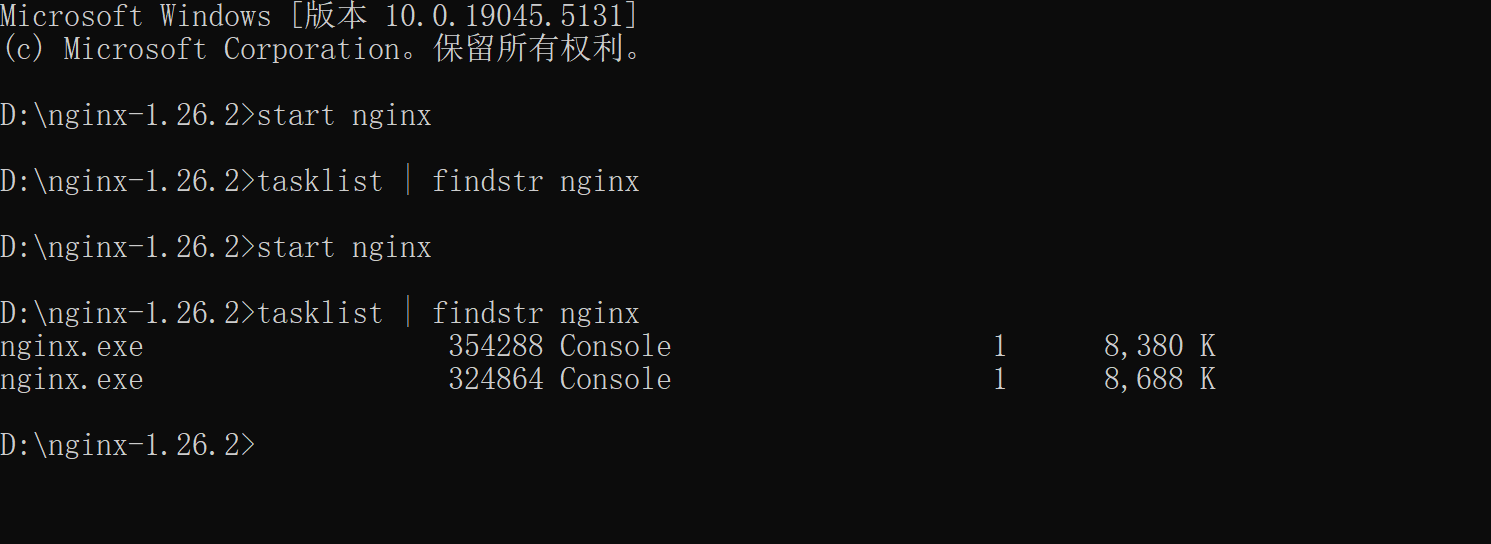
但是可以http://192.168.17.129/www/a.html。可以看到页面



**六、项目打包实战**

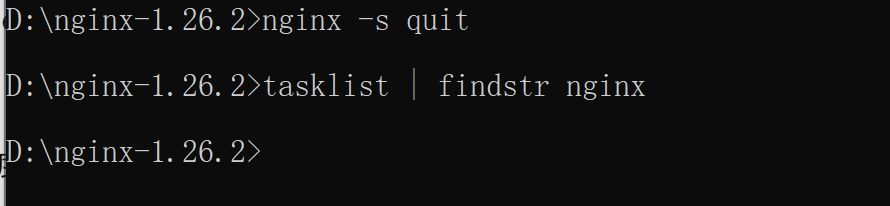
**1、测试**

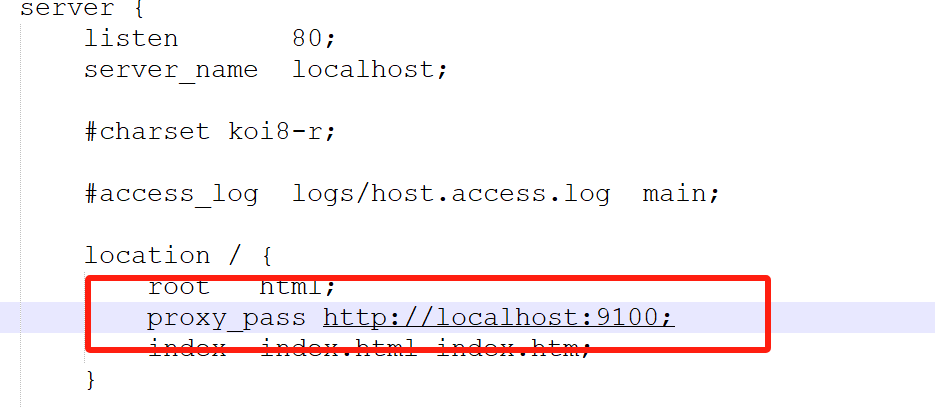


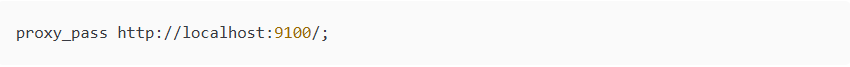




不返回结果，代表关闭

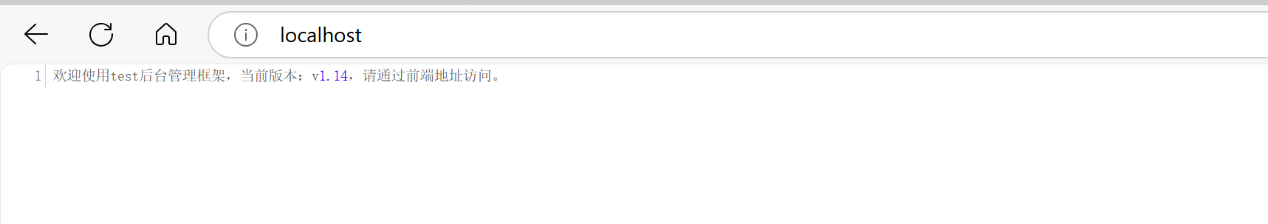






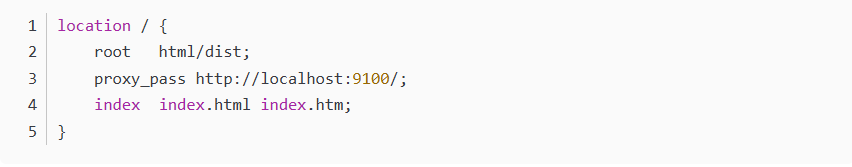
**注意路径后面有一个“/"**

和前面的反向代理一样。这里可以直接访问到后端服务器，但是我们一般用ngnix部署前端



2、若依打包部署

不能这样写：



即使你配置了 root html/dist; 来提供静态文件，proxy\_pass会将请求转发到 http://localhost:9100，而不是直接访问静态文件。







原文链接：https://blog.csdn.net/qq\_56947957/article/details/144711277