基于Nginx的负载均衡实验报告

# 实验目的

参考课堂演示的实验实现过程，自行实现负载均衡实验

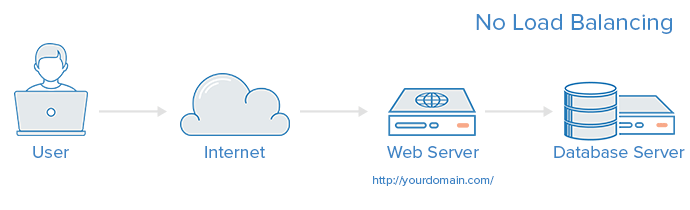
# 实验原理

## 什么是负载均衡（为什么需要负载均衡）？

负载均衡建立在现有网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。

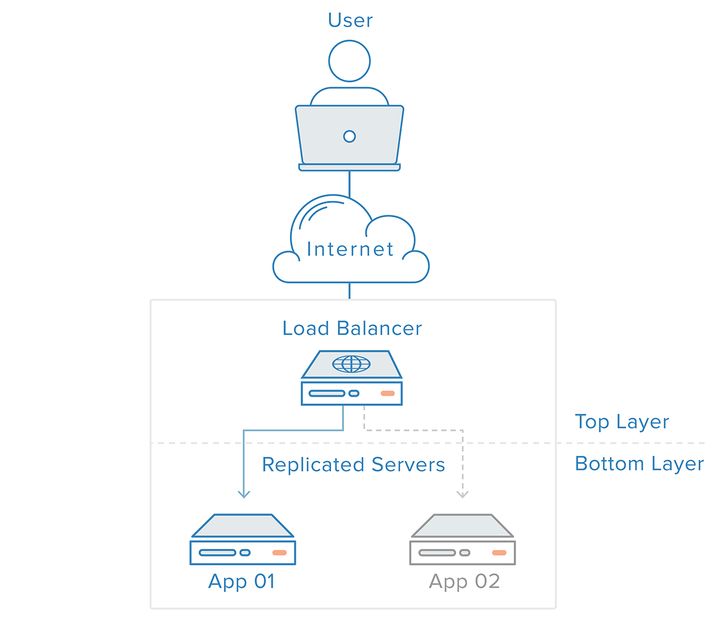
负载均衡*（Load Balance）*其意思就是分摊到多个操作单元上进行执行，例如Web服务器、FTP服务器、企业关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。

一个没有负载均衡的 web 架构类似下面这样：



这种情况下，用户是直连到 web 服务器，如果这个服务器宕机了，那么用户自然也就没办法访问了。另外，如果同时有很多用户试图访问服务器，超过了其能处理的极限，就会出现加载速度缓慢或根本无法连接的情况。

引入负载均衡之后，上述web架构需要额外的一个负载均衡器和web服务器，此时的web架构类似下面这样：



用户访问负载均衡器，再由负载均衡器将请求转发给后端服务器。在这种情况下，单点故障现在转移到负载均衡器上了，但是又可以通过引入第二个负载均衡器来缓解。

## 负载均衡的主要应用

**1、DNS负载均衡** 最早的负载均衡技术是通过DNS来实现的，在DNS中为多个地址配置同一个名字，因而查询这个名字的客户机将得到其中一个地址，从而使得不同的客户访问不同的服务器，达到负载均衡的目的。DNS负载均衡是一种简单而有效的方法，但是它不能区分服务器的差异，也不能反映服务器的当前运行状态。

**2、代理服务器负载均衡** 使用代理服务器，可以将请求转发给内部的服务器，使用这种加速模式显然可以提升静态网页的访问速度。然而，也可以考虑这样一种技术，使用代理服务器将请求均匀转发给多台服务器，从而达到负载均衡的目的。

**3、地址转换网关负载均衡** 支持负载均衡的地址转换网关，可以将一个外部IP地址映射为多个内部IP地址，对每次TCP连接请求动态使用其中一个内部地址，达到负载均衡的目的。

**4、协议内部支持负载均衡** 除了这三种负载均衡方式之外，有的协议内部支持与负载均衡相关的功能，例如HTTP协议中的重定向能力等，HTTP运行于TCP连接的最高层。

**5、NAT负载均衡** NAT（Network Address Translation网络地址转换）简单地说就是将一个IP地址转换为另一个IP地址，一般用于未经注册的内部地址与合法的、已获注册的Internet IP地址间进行转换。适用于解决Internet IP地址紧张、不想让网络外部知道内部网络结构等的场合下。

**6、反向代理负载均衡** 普通代理方式是代理内部网络用户访问internet上服务器的连接请求，客户端必须指定代理服务器，并将本来要直接发送到internet上服务器的连接请求发送给代理服务器处理。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，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。反向代理负载均衡技术是把将来自internet上的连接请求以反向代理的方式动态地转发给内部网络上的多台服务器进行处理，从而达到负载均衡的目的。

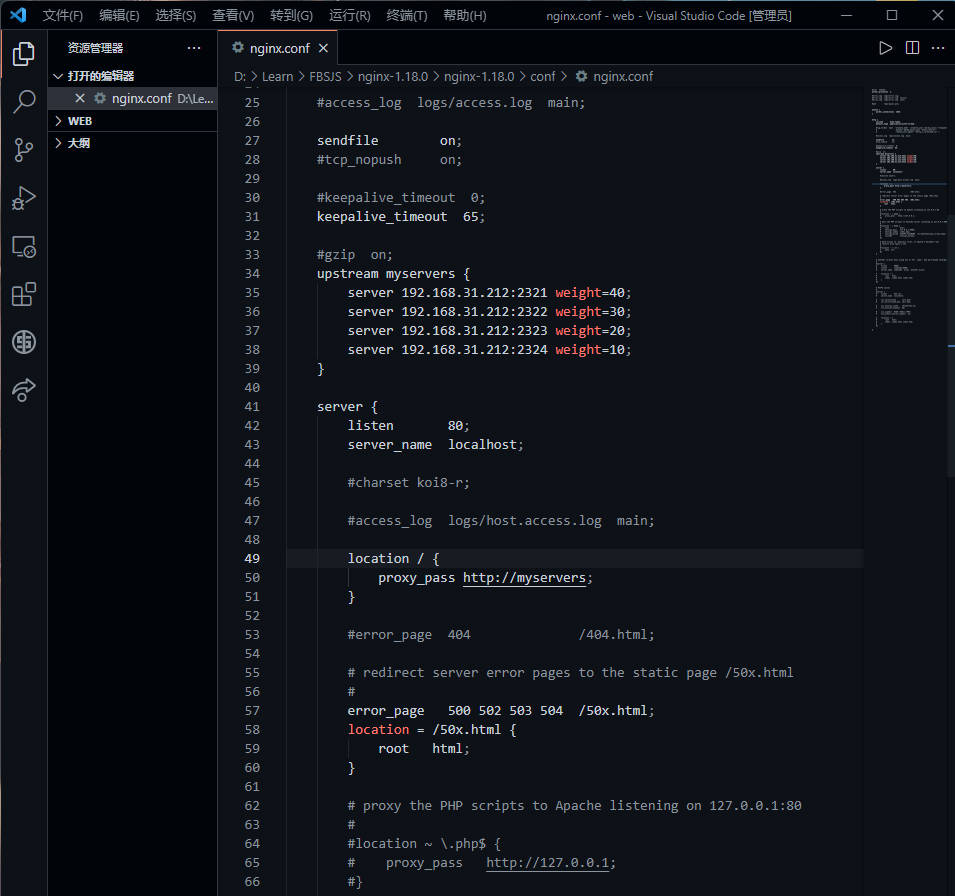
**7、混合型负载均衡** 在有些大型网络，由于多个服务器群内硬件设备、各自的规模、提供的服务等的差异，可以考虑给每个服务器群采用最合适的负载均衡方式，然后又在这多个服务器群间再一次负载均衡或群集起来以一个整体向外界提供服务（即把这多个服务器群当做一个新的服务器群），从而达到最佳的性能。将这种方式称之为混合型负载均衡。此种方式有时也用于单台均衡设备的性能不能满足大量连接请求的情况下。 [3]

## 实验过程

实验环境：

Windows11 + Nginx 1.18.0 + postman + nodejs 14.17.6

Nginx配置文件修改：



upstream 模块默认采用轮询法，根据每个web端口的权重会访问不同的次数

配置webserver文件充当web服务器：



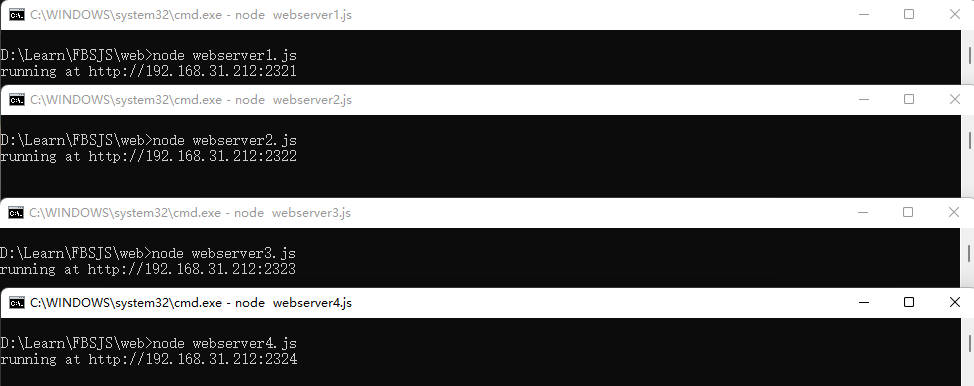
如图为webserver1端口为2321，webserver2、3、4的配置类似，端口不同。

## 实验结果

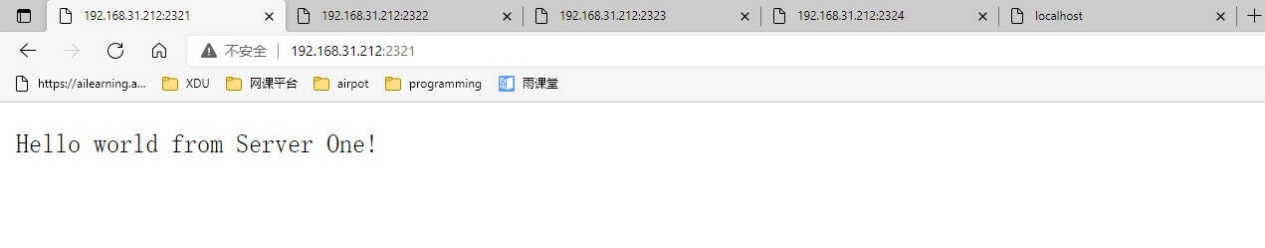
启动nginx并重载修改配置文件：



启动webserver服务：

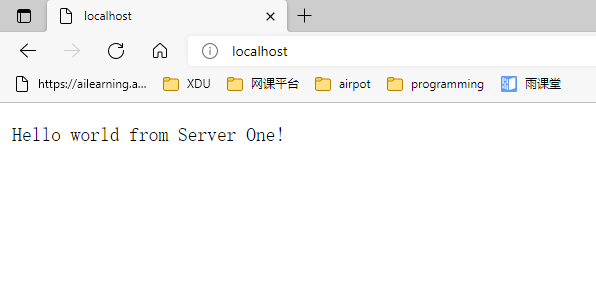


edge浏览器充当客户端，验证各个webserver服务正确启动：



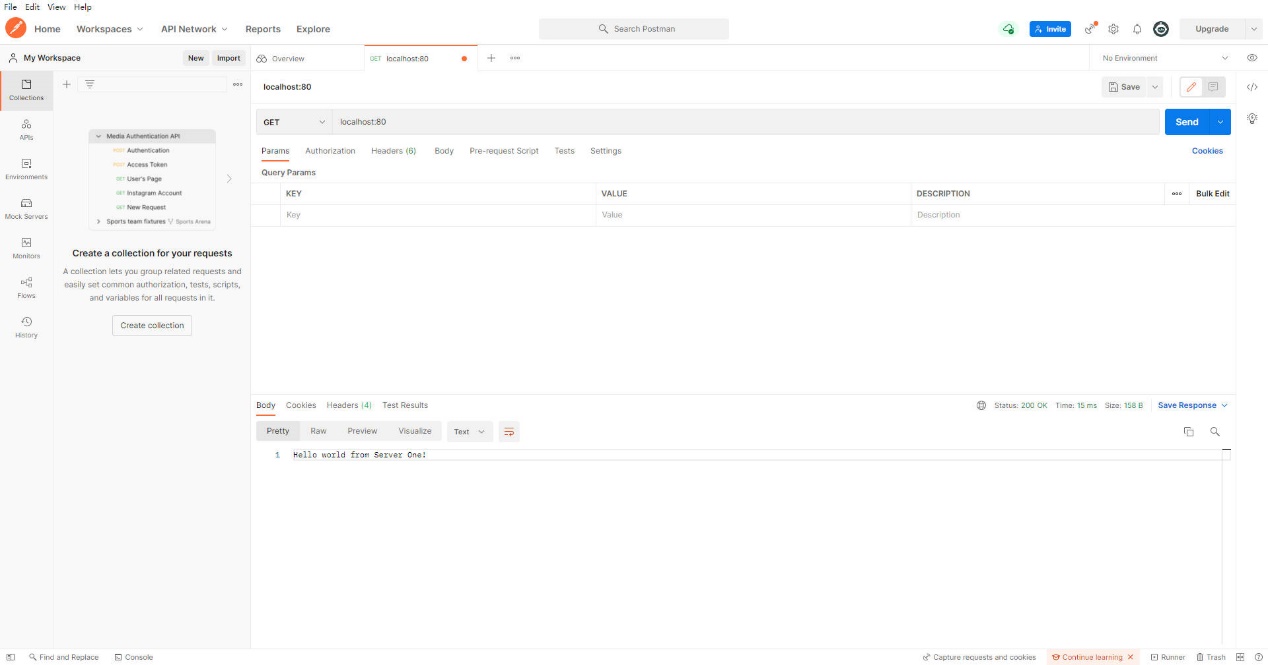


Edge浏览器充当客户端访问：



每次刷新会重新访问，得到的信息显示访问是轮询访问但顺序是随机的，若权重相同则完全随机，权重不同则大致按照各个webserver权重占比访问。

Postman充当客户端时：



通过访问localhost：80端口，得到的信息显示访问是轮询访问但是顺序是随机的，与使用浏览器访问结果相似，与课堂上使用curl充当客户端时，轮询访问是按照顺序进行访问的结果不同。