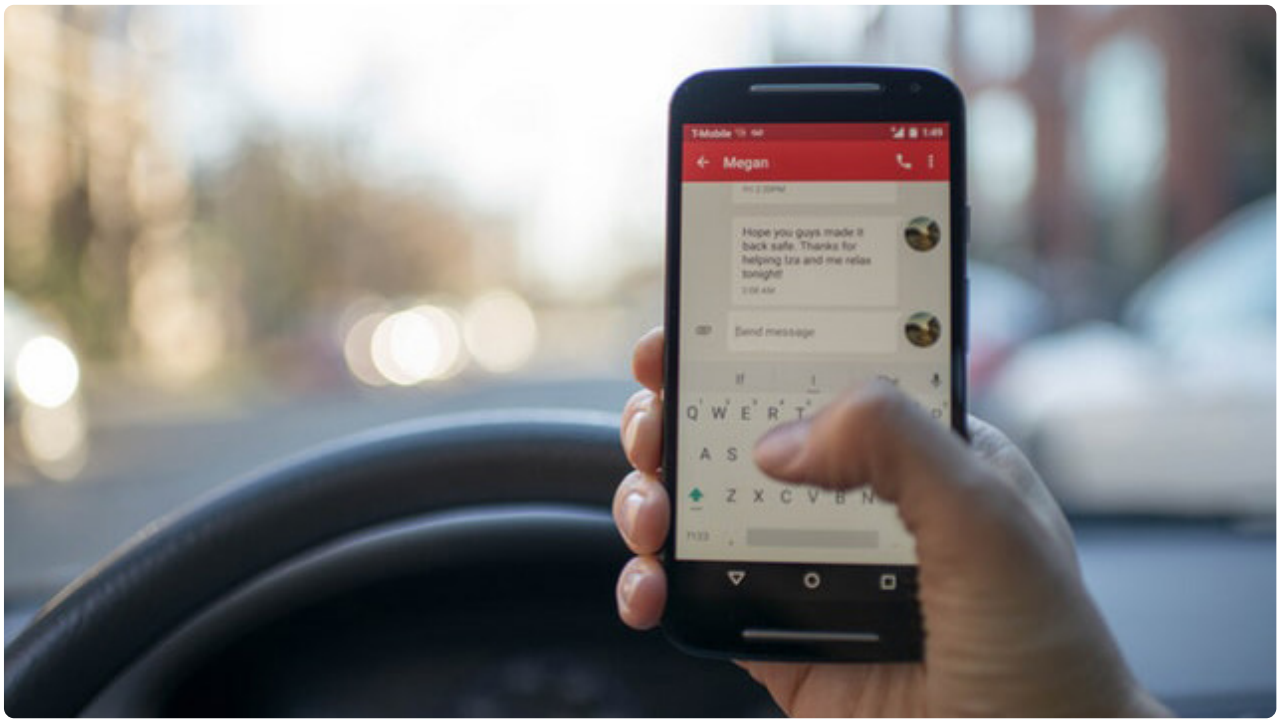
21 配置负载均衡

更新时间：2020-02-21 10:07:42



> 只有在那崎岖的小路上不畏艰险奋勇攀登的人,才有希望达到光辉的顶点。——马克思

前言

前面几篇文章中，我们讲解了 反向代理 ， 负载均衡 等内容，其实这些理论知识挺枯燥的，有些地方还比较生硬。我们在这篇文章中通过实例来深入理解一下这些概念。

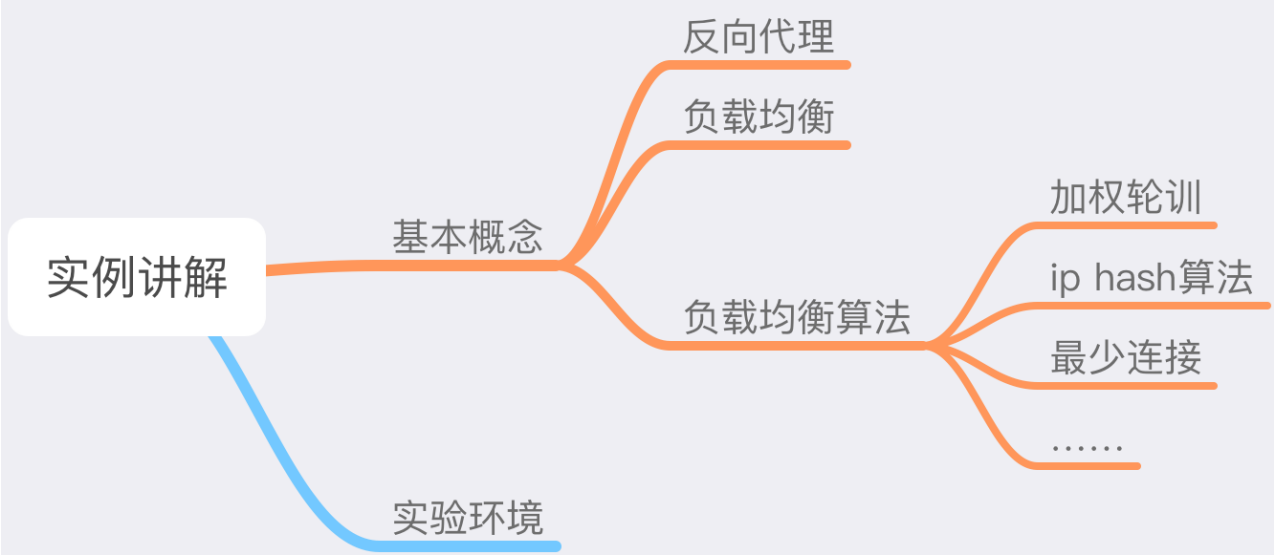
内容回顾

先简单的回顾一下前面几篇文章中介绍到的概念。

负载均衡：多台服务器按照一定的策略分担负载，最大效率的利用服务器的性能。

反向代理：对外提供统一的接口，屏蔽真实提供请求的服务器。和负载均衡一起配合，构成了现在服务架构的基础。

负载均衡策略：这是一种算法，按照这种算法分发用户的请求，保证最大化的利用服务器性能。



实验环境

> 纸上得来终觉浅，绝知此事要躬行

只有通过动手做实验，才能真正的理解上面介绍的几个概念。

首先，我们要搭建一个测试环境，我们推荐使用 docker 来搭建，非常的方便快捷。

环境准备

由于要练习负载均衡，所以要多台服务器，但是在测试环境中，我们可以通过域名的方式来模拟多台服务器。

在 /etc/hosts 中定义了三个域名，分别是 localhost， www.backend1.com， www.backend2.com 。

```
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      d9caa3dc65d6
127.0.0.1       www.backend1.com  www.backend2.com
```

定义两个 nginx 配置文件，分别为 nginx.conf.8081 和 nginx.conf.8082，后面的数字分别代表两个服务器监听的端口号，配置内容如下：

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    access_log /usr/local/nginx/logs/access.log.8081;

    server {
        listen       8081;
        server_name  www.backend1.com;

        location /proxy {
                return 200 "server www.backend1.com:8081    \n";
        }


    }
```

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    access_log /usr/local/nginx/logs/access.log.8082;

    server {
        listen       8082;
        server_name  www.backend2.com;

        location /proxy {
                return 200 "www.backend2.com:8082     \n";
        }
    }
```


我们在这两个配置文件中，对同一个服务 "/proxy" 返回了不同的内容，真正的线上服务器不是这样的。他们应该是相同的，我们这里主要是为了区分负载均衡策略到底选择了哪个后端服务器。

定义前端服务器配置文件 nginx.conf，这个服务器主要是用来接收用户端的请求，然后根据负载均衡策略选择不同的服务器。

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    access_log /usr/local/nginx/logs/access.log;

    upstream backend {
        server www.backend1.com:8081 weight=3;
        server www.backend2.com:8082;
    }
    server {
        listen       80;
        server_name  localhost;

        location /proxy {
                proxy_pass http://backend;
        }

    }
```

后端集群

反向代理到后端集群

在这里我们定义了包含了两个后端服务器的集群。我们使用了默认的加权轮训策略，因为第一台服务器 www.backend1.com 的权重是 3，第二台服务器的权重是 1，所以正常情况下每四个请求中会有三个请求被第一个后端服务器处理，剩下的一个被第二个后端服务器处理。

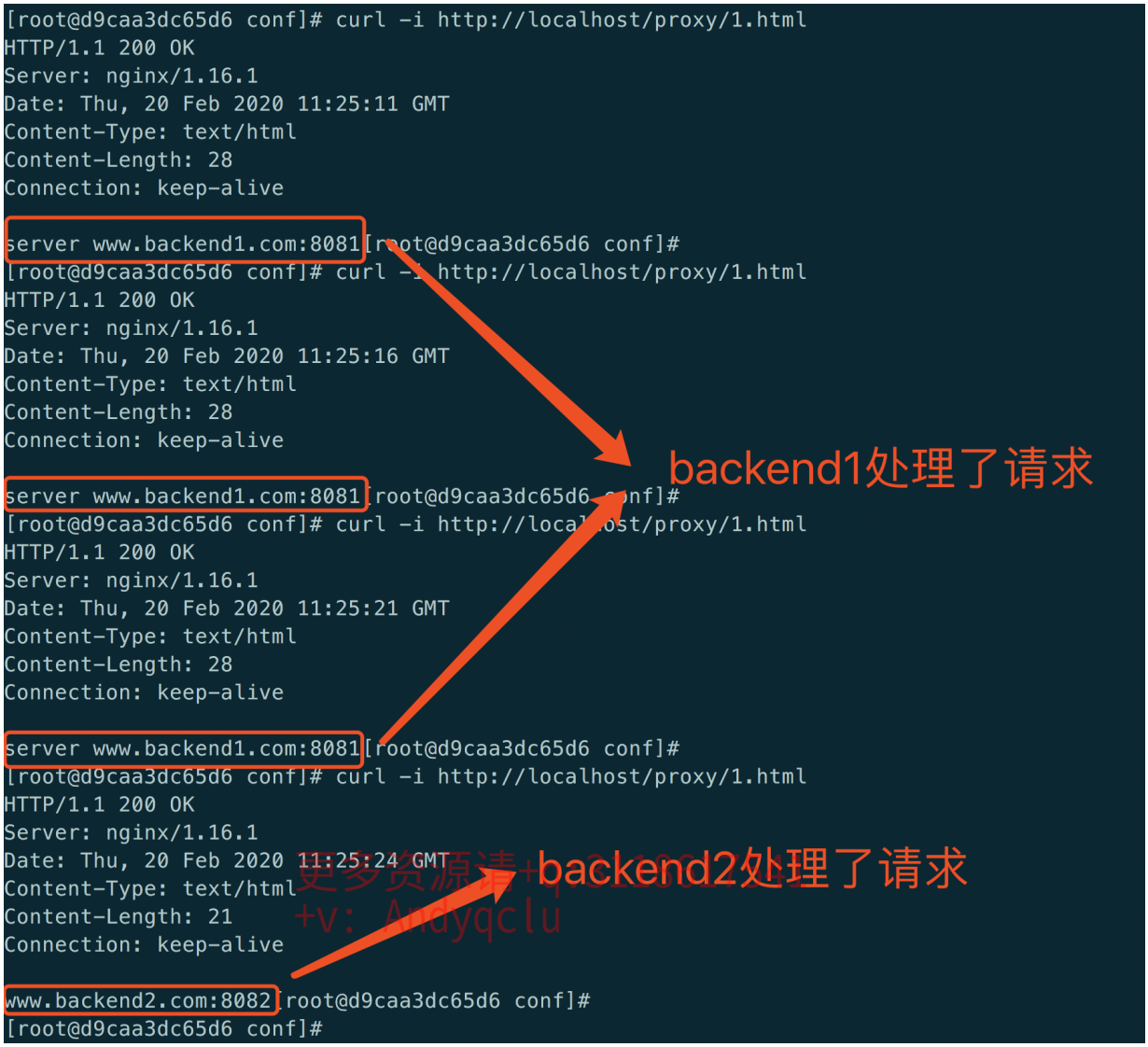
分别启动这三个服务器，如下图所示：

```
root        99     0  0 06:52 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf.8081
nobody     100    99  0 06:52 ?        00:00:00 nginx: worker process
root       103     0  0 06:52 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf.8082
nobody     104   103  0 06:52 ?        00:00:00 nginx: worker process
root       118     0  0 06:59 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx
nobody     219   118  0 10:45 ?        00:00:00 nginx: worker process
root       237    16  0 11:22 pts/1    00:00:00 grep --color=auto nginx
```

可以看到，共有三个 master 进程，三个 worker 进程，说明我们的服务已经全部启动成功。

测试请求

我们使用 curl 命令进行测试，进行四次请求，结果如下：



我们请求了四次，其中三次是被 backend1 进行了处理，一次被 backend2 进行了处理，这和我们的权重是相符合的。

我们这里只是使用了比较常见的轮询策略，还有其他的各种负载均衡策略，他们应用到不同的场合，大家可以动手试一试，加深理解。

总结

我为什么要单独的写这一篇文章呢？当时我在学习的时候，因为刚开始学习编程，不知道怎么入手，看着书上的概念，完全处于一种懵逼状态。不知道动手练习，也不知道该如何下手，导致了学习的速度特别的慢，我在这里就是想告诉初学者，一定要多动手，安装一个 docker 环境，然后随便的折腾，动手操作之后真的能更深刻的理解这些概念。

}

20 负载均衡模块搭建

22 压缩压缩再压缩