实验报告

课程名称：交通大数据管理

实验序号：实验 04　　实验项目名称：基于 Nginx 的动态负载均衡

学　　号	2210720131	姓　　名	薛文清	专业、班	22 大数据
实验地点	精工 1-406	指导教师	蔡钟淇	实验时间	2025-4-10

实验步骤（请给出关键代码、运行截图和分析）

1. 下载 consul，解压并将其移动到 usr/local/bin/文件夹

```
xwq@ubuntu:~$ wget https://releases.hashicorp.com/consul/1.15.3/consul_1.15.3_linux_amd64.
zip
--2025-04-09 23:57:14--  https://releases.hashicorp.com/consul/1.15.3/consul_1.15.3_linux_
amd64.zip
正在解析主机 releases.hashicorp.com (releases.hashicorp.com)... 198.18.0.146
正在连接 releases.hashicorp.com (releases.hashicorp.com)|198.18.0.146|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度： 53749844 (51M) [application/zip]
正在保存至: “consul_1.15.3_linux_amd64.zip”

consul_1.15.3_linux_am 100%[=========================>]  51.26M   893KB/s    in 41s

2025-04-09 23:57:56 (1.24 MB/s) - 已保存 “consul_1.15.3_linux_amd64.zip” [53749844/5374984
4])

xwq@ubuntu:~$ unzip consul_1.15.3_linux_amd64.zip
Archive:  consul_1.15.3_linux_amd64.zip
  inflating: consul
xwq@ubuntu:~$ sudo mv consul /usr/local/bin/
xwq@ubuntu:~$
```

2. 启动 consul 服务，指令如下

consul agent -server -bootstrap-expect=1 -data-dir=/tmp/consul -node=agent-one -client=0.0.0.0 -ui

参数说明：

-server：以服务器模式运行

-bootstrap-expect=1：单节点模式

-data-dir：数据存储路径

-client：允许外部访问 ip 地址

-ui 启动 webui 界面

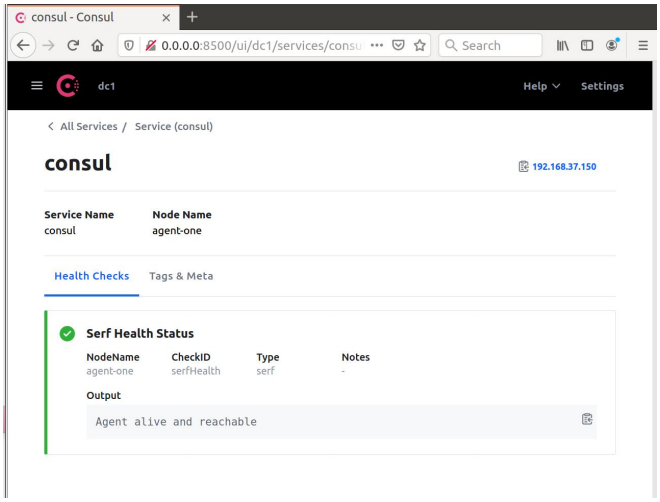
如图所示成功启动 consul

```
xwq@ubuntu:~$ consul agent -server -bootstrap-expect=1 -data-dir=/tmp/consul -node=agent-o
ne -client=0.0.0.0 -ui
==> Starting Consul agent...
           Version: '1.15.3'
         Build Date: '2023-06-01 20:40:32 +0000 UTC'
            Node ID: 'bfb75bd4-ca3f-56c8-e6e7-0fa6aa908625'
          Node name: 'agent-one'
         Datacenter: 'dc1' (Segment: '<all>')
             Server: true (Bootstrap: true)
        Client Addr: [0.0.0.0] (HTTP: 8500, HTTPS: -1, gRPC: -1, gRPC-TLS: 8503, DNS: 86
00)
       Cluster Addr: 192.168.37.150 (LAN: 8301, WAN: 8302)
    Gossip Encryption: false
     Auto-Encrypt-TLS: false
            HTTPS TLS: Verify Incoming: false, Verify Outgoing: false, Min Version: TLSv1_
2
             gRPC TLS: Verify Incoming: false, Min Version: TLSv1_2
     Internal RPC TLS: Verify Incoming: false, Verify Outgoing: false (Verify Hostname: fa
lse), Min Version: TLSv1_2

==> Log data will now stream in as it occurs:

2025-04-10T00:14:53.972-0700 [WARN]  agent: BootstrapExpect is set to 1; this is the same
as Bootstrap mode.
2025-04-10T00:14:53.972-0700 [WARN]  agent: bootstrap = true: do not enable unless necessa
ry
2025-04-10T00:14:53.976-0700 [WARN]  agent.auto_config: BootstrapExpect is set to 1; this
is the same as Bootstrap mode.
2025-04-10T00:14:53.976-0700 [WARN]  agent.auto_config: bootstrap = true: do not enable un
less necessary
```

3. 访问 consul UI,如图所示显示当前 NodeName 为我们刚才设置的 agent-one,自我检查健康



4. 安装 nginx 来作为我们的负载均衡器，设置端口为 80，并加入 upsync 模块实现代码如下：

wget http://nginx.org/download/nginx-1.20.2.tar.gz

git clone https://github.com/weibocom/nginx-upsync-module

tar -zxvf nginx-1.20.2.tar.gz

cd nginx-1.20.2

./configure --add-module=../nginx-upsync-module

make && sudo make install

其中编译 nginx 步骤中遇到了 PCRE 和 zlib 缺失，但是通过 apt 安装均已解决

编译并安装后，使用-v 命令检查是否安装成功

```
cp conf/fastcgi.conf '/usr/local/nginx/conf/fastcgi.conf.default'
test -f '/usr/local/nginx/conf/uwsgi_params' \
        || cp conf/uwsgi_params '/usr/local/nginx/conf'
cp conf/uwsgi_params \
        '/usr/local/nginx/conf/uwsgi_params.default'
test -f '/usr/local/nginx/conf/scgi_params' \
        || cp conf/scgi_params '/usr/local/nginx/conf'
cp conf/scgi_params \
        '/usr/local/nginx/conf/scgi_params.default'
test -f '/usr/local/nginx/conf/nginx.conf' \
        || cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf'
cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf.default'
test -d '/usr/local/nginx/logs' \
        || mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/logs' \
        || mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/html' \
        || cp -R html '/usr/local/nginx'
test -d '/usr/local/nginx/logs' \
        || mkdir -p '/usr/local/nginx/logs'
make[1]: Leaving directory '/home/xwq/nginx-1.20.2'
xwq@ubuntu:~/nginx-1.20.2$ /usr/local/nginx/sbin/nginx -v
nginx version: nginx/1.20.2
```

5. Nginx 动态负载均衡配置

   编辑/usr/local/nginx/conf/nginx.conf，添加如图所示内容

```
xwq@ubuntu:~/nginx-1.20.2$ sudo cat /usr/local/nginx/conf/nginx.conf
user  root;
worker_processes  auto;

error_log  /usr/local/nginx/logs/error.log warn;
pid        /usr/local/nginx/logs/nginx.pid;


events {
    worker_connections  1024;
}

http {
        include /usr/local/nginx/conf/mime.types;
    default_type  application/octet-stream;

        include conf.d/*.conf;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /usr/local/nginx/logs/access.log main;
    sendfile        on;
    tcp_nopush     on;
    keepalive_timeout  65;

    # 动态负载均衡核心配置
    upstream backend {
        # 从Consul获取服务列表（每500ms同步一次）
        upsync 127.0.0.1:8500/v1/kv/upstreams/backend upsync_type=consul
        upsync_interval=500ms;

        # 本地持久化路径（防止Consul不可用时使用缓存）
        upsync_dump_path /usr/local/nginx/conf/servers.conf;

        # 必须放在include之前
        server 127.0.0.1:8080 weight=1 backup;  # 添加weight参数
        # 包含从Consul同步的服务器列表
        include /usr/local/nginx/conf/servers.conf;
    }

    server {
        listen       80;
        server_name  localhost;

        location / {
            # 反向代理到动态upstream
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;

            # 负载均衡策略（可选）
            proxy_next_upstream error timeout http_500;
        }

        # 保留原有错误页面配置
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   /usr/share/nginx/html;
        }
    }

    # 可保留其他server配置（如HTTPS）
}
xwq@ubuntu:~/nginx-1.20.2$
```

检查 nginx 配置语法是否正确

```
xwq@ubuntu:~/nginx-1.20.2$ sudo /usr/local/nginx/sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
xwq@ubuntu:~/nginx-1.20.2$
```

6. 为了方便使用 nginx，创建符号链接以后直接使用 nginx 即可启动服务

```
xwq@ubuntu:~/nginx-1.20.2$ # 创建符号链接
xwq@ubuntu:~/nginx-1.20.2$ sudo ln -s /usr/local/nginx/sbin/nginx /usr/sbin/nginx
xwq@ubuntu:~/nginx-1.20.2$
xwq@ubuntu:~/nginx-1.20.2$ # 验证安装
xwq@ubuntu:~/nginx-1.20.2$ nginx -V 2>&1 | grep upsync  # 应该显示"--add-module=../nginx-u
psync-module"
configure arguments: --add-module=../nginx-upsync-module
xwq@ubuntu:~/nginx-1.20.2$
```

7. 启动 nginx

```
xwq@ubuntu:~/nginx-1.20.2$ sudo nginx
```

8. 在路径/usr/local/nginx/conf/conf.d/配置俩个端口分别为 8080 和 8081，文件名分别为 server1.conf 和 server2.conf，并将该路径写入 nginx.conf 中，运行 nginx，测试俩个服务器是否可以正常访问

```
xwq@ubuntu:~/nginx-1.20.2$ curl http://localhost:8080
<h1>Server 1 (8080)</h1>
xwq@ubuntu:~/nginx-1.20.2$ curl http://localhost:8081
<h1>Server 2 (8081)</h1>
```

9. 将俩个后端服务器添加到 consul 中，均显示 true

```
xwq@ubuntu:~/nginx-1.20.2$ curl -X PUT -d '{"weight":2, "max_fails":3, "fail_timeout":10}'
 \
> http://127.0.0.1:8500/v1/kv/upstreams/backend/127.0.0.1:8080
truexwq@ubuntu:~/nginx-1.20.2$ curl -X PUT -d '{"weight":3, "max_fails":3, "fail_timeout":
 \}'
> http://127.0.0.1:8500/v1/kv/upstreams/backend/127.0.0.1:8081
truexwq@ubuntu:~/nginx-1.20.2$
```

10. 验证 consul 存储的数据，使用如图所示指令，如图所示正常返回俩个服务器

```
xwq@ubuntu:~/nginx-1.20.2$ curl -s http://127.0.0.1:8500/v1/kv/upstreams/backend?recurse |
 jq .
[
  {
    "LockIndex": 0,
    "Key": "upstreams/backend/127.0.0.1:8080",
    "Flags": 0,
    "Value": "eyJ3ZWlnaHQiOjIsICJtYXhfZmFpbHMiOjMsICJmYWlsX3RpbWVvdXQiOjEwfQ==",
    "CreateIndex": 512,
    "ModifyIndex": 809
  },
  {
    "LockIndex": 0,
    "Key": "upstreams/backend/127.0.0.1:8081",
    "Flags": 0,
    "Value": "eyJ3ZWlnaHQiOjMsICJtYXhfZmFpbHMiOjMsICJmYWlsX3RpbWVvdXQiOjEwfQ==",
    "CreateIndex": 810,
    "ModifyIndex": 810
  }
]
xwq@ubuntu:~/nginx-1.20.2$
```

11. 再往 consul 中添加一个服务器，设置为 8082 端口，设置权重为 4,图中漏了一个步骤创建 nginx 的配置文件 sudo nano /usr/local/nginx/conf/conf.d/server3.conf

```
xwq@ubuntu:~/nginx-1.20.2$ sudo mkdir -p /var/www/server3
xwq@ubuntu:~/nginx-1.20.2$ echo "<h1>Backend Server 3 (8082)</h1>" | sudo tee /var/www/ser
ver3/index.html
<h1>Backend Server 3 (8082)</h1>
xwq@ubuntu:~/nginx-1.20.2$ curl -X PUT -d '{"weight":4}' \
> http://127.0.0.1:8500/v1/kv/upstreams/backend/127.0.0.1:8082
xwq@ubuntu:~/nginx-1.20.2$ curl -s http://127.0.0.1:8500/v1/kv/upstreams/backend?recurse |
 jq .
[
  {
    "LockIndex": 0,
    "Key": "upstreams/backend/127.0.0.1:8080",
    "Flags": 0,
    "Value": "eyJ3ZWlnaHQiOjIsICJtYXhfZmFpbHMiOjMsICJmYWlsX3RpbWVvdXQiOjEwfQ==",
    "CreateIndex": 512,
    "ModifyIndex": 809
  },
  {
    "LockIndex": 0,
    "Key": "upstreams/backend/127.0.0.1:8081",
    "Flags": 0,
    "Value": "eyJ3ZWlnaHQiOjMsICJtYXhfZmFpbHMiOjMsICJmYWlsX3RpbWVvdXQiOjEwfQ==",
    "CreateIndex": 810,
    "ModifyIndex": 810
  },
  {
    "LockIndex": 0,
    "Key": "upstreams/backend/127.0.0.1:8082",
    "Flags": 0,
    "Value": "eyJ3ZWlnaHQiOjR9",
    "CreateIndex": 848,
    "ModifyIndex": 848
  }
]
xwq@ubuntu:~/nginx-1.20.2$
```

12. 将 8081 的权重修改为 6

```
xwq@ubuntu:~/nginx-1.20.2$ curl -X PUT -d '{"weight":6}' \
> http://127.0.0.1:8500/v1/kv/upstreams/backend/127.0.0.1:8081
```

13. 删除 8080 端口

```
xwq@ubuntu:~/nginx-1.20.2$ curl -s http://127.0.0.1:8500/v1/kv/upstreams/backend?recurse |
 jq .
[
  {
    "LockIndex": 0,
    "Key": "upstreams/backend/127.0.0.1:8081",
    "Flags": 0,
    "Value": "eyJ3ZWlnaHQiOjZ9",
    "CreateIndex": 810,
    "ModifyIndex": 856
  },
  {
    "LockIndex": 0,
    "Key": "upstreams/backend/127.0.0.1:8082",
    "Flags": 0,
    "Value": "eyJ3ZWlnaHQiOjR9",
    "CreateIndex": 848,
    "ModifyIndex": 848
  }
]
xwq@ubuntu:~/nginx-1.20.2$
```

最终端口如下所示

```
xwq@ubuntu:~/nginx-1.20.2$ cat /usr/local/nginx/conf/servers.conf
server 127.0.0.1:8082 weight=4 max_fails=2 fail_timeout=10s;
server 127.0.0.1:8081 weight=6 max_fails=2 fail_timeout=10s;
```

14. 测试请求分发，循环 100 次，查看是否遵循 6：4 的权重分发，如图所示

```
xwq@ubuntu:~/nginx-1.20.2$ for i in {1..100}; do curl -s http://localhost; done | sort | u
niq -c
     40 <h1>Backend Server 3 (8082)</h1>
     60 <h1>Server 2 (8081)</h1>
```

15. 重新添加 8080 端口服务器，设置权重为 1，访问一百次，如图所示，正常分发

```
truexwq@ubuntu:~/nginx-1.20for i in {1..100}; do curl -s http://localhost; done | sort | u
niq -c
     36 <h1>Backend Server 3 (8082)</h1>
      9 <h1>Server 1 (8080)</h1>
     55 <h1>Server 2 (8081)</h1>
xwq@ubuntu:~/nginx-1.20.2$
```

16. 打印最近 5 条访问日志

```
xwq@ubuntu:~/nginx-1.20.2$ sudo tail -n 5 /usr/local/nginx/logs/access.log
127.0.0.1 - - [14/Apr/2025:05:33:22 -0700] "GET / HTTP/1.1" 200 25 "-" "curl/7.47.0" "-"
127.0.0.1 - - [14/Apr/2025:05:33:22 -0700] "GET / HTTP/1.1" 200 25 "-" "curl/7.47.0" "-"
127.0.0.1 - - [14/Apr/2025:05:33:22 -0700] "GET / HTTP/1.1" 200 25 "-" "curl/7.47.0" "-"
127.0.0.1 - - [14/Apr/2025:05:33:22 -0700] "GET / HTTP/1.1" 200 33 "-" "curl/7.47.0" "-"
127.0.0.1 - - [14/Apr/2025:05:33:22 -0700] "GET / HTTP/1.1" 200 25 "-" "curl/7.47.0" "-"
xwq@ubuntu:~/nginx-1.20.2$
```

解释如下:

字段值	字段名	含义	示例值解释
127.0.0.1	$remote_addr	客户端的 IP 地址	本机通过 localhost 访问，IP 为 127.0.0.1
第一个 -	$remote_user	客户端认证的用户名（未认证则为 -）	无用户认证，显示为 -
第二个 -	（占位符）	保留字段（无实际意义，仅分隔符）	默认占位符，保持日志格式对齐
[14/Apr/2025:05:33:22 -0700]	$time_local	服务器接收请求的时间（带时区）	请求时间为 2025 年 4 月 14 日 05:33:22，时区为 UTC-7
"GET / HTTP/1.1"	$request	客户端请求的完整内容（方法 + URL + 协议）	使用 GET 方法访问根路径/，协议为 HTTP/1.1
200	$status	HTTP 响应状态码	200 表示请求成功
25	$body_bytes_sent	发送给客户端的响应体字节数	本次响应返回 25 字节的内容
第三个 -	$http_referer	请求来源页面（直接访问或无来源时为 -）	直接通过地址栏访问或无来源页面，显示为 -
"curl/7.47.0"	$http_user_agent	客户端使用的工具或浏览器信息	使用 curl 7.47.0 命令行工具发起请求
第四个 -	$http_x_forwarded_for	客户端原始 IP（当请求经过代理时记录，无代理则为 -）	请求未经过代理服务器，显示为 -

实验中出现的问题：

(1) 没有理解 apt 安装 nginx 与手动编译安装 nginx 的区别，由于实验要求安装 Nginx-upsync-model 模块，而 apt 安装无法实现该要求，所以必须使用手动安装

教师评语 签名： 日期：	成绩