Nginx用法指南

目录：如果只想知道怎么用的转：4.nginx的用法

1.Nginx是什么？----定义和关键概念理解
2.Nginx能干什么？----用途
1）静态内容的处理
2）做代理
3）反向代理负载均衡
4）服务分发
3.nginx的负载均衡策略
1）round-robin
2）round-robin+权重配置
3）第三种配置：ip Hash
4）least-conn
4.nginx的用法
1）nginx的下载
2）nginx的运行和操作
5.总结

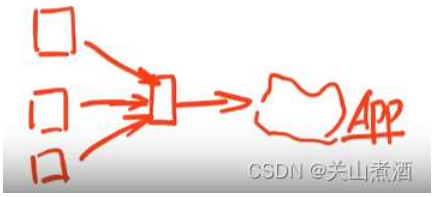
1.Nginx是什么？----定义和关键概念理解

nginx是一个http代理和反向代理服务器，下面是其英文说明。

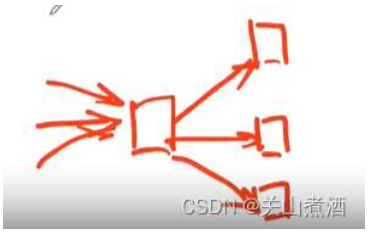
nginx [engine x] is an HTTP and reverse proxy server, as well as a mail proxy server, written by Igor Sysoev.

代理和反向代理的意思

代理是有一堆机器，比如大学宿舍的很多同学的电脑，然后因为没有那么多的ip，于是通过一个代理发送到互联网上的app请求资源，在App视角来看，这些请求都来自一个ip，即宿舍同学的机器都是被一个服务器给**代理**了，如下图：



而反向代理则是如下图，有一堆请求到一个代理服务器，代理服务器将这些请求发送给一个集群的服务器做负载均衡去处理请求，可以看到代理和反向代理的方向是反的。



2.Nginx能干什么？----用途

1）静态内容的处理

例如nginx的配置文件里有如下内容：

```
server {
 location / {
   root /data/www;
 }
 location /images/ {
   root /data;
 }
}
```

比如data/www目录下放了index.html和其他文本内容，data/images目录下有很多的图片，这个配置的意思是请求中带有/image的请求会和location中/image匹配，然后组装成/data/image,其余的请求则会到data/www目录下去找资源

```
location /images/ {
  root /data;
}
```


举个例子：

- This is already a working configuration of a server that listens on the standard port 80 and is accessible on the local machine at `http://localhost/`.
- In response to requests with URIs starting with /images/, the server will send files from the /data/images directory.
  - For example, in response to the `http://localhost/images/example.png` request nginx will send the `/data/images/example.png` file. If such file does not exist, nginx will send a response indicating the 404 error.
- Requests with URIs not starting with `/images/` will be mapped onto the `/data/www` directory.
  - For example, in response to the `http://localhost/some/example.html` request nginx will send the `/data/www/some/example.html` file.

假设server在localhost的80端口，带image的请求时这样匹配的：

For example, in response to the `http://localhost/images/example.png` request nginx will send the `/data/images/example.png` file. If such file does not exist, nginx will send a response indicating the 404 error.

而不是以image开始的请求则是这样匹配的：

Requests with URIs not starting with `/images/` will be mapped onto the `/data/www` directory.

- For example, in response to the `http://localhost/some/example.html` request nginx will send the `/data/www/some/example.html` file.

2）做代理

如下：

```
server {
 location / {
  proxy_pass http://localhost:8080/;
 }
 location ~ \.(gif|jpg|png)$ {
  root /data/images;
 }
}
```


这个配置的意思是凡是对8080请求，凡是以gif、jpg、png结尾的都会被代理到到/data/image （注意这里的第二个location那里写了**正则表达式**）

3）反向代理负载均衡

如下：upstream是一个上游服务器，有三台。server监听80端口的请求，并将所有请求转发到http://myapp1去处理

```
http {
 upstream myapp1 {
  server srv1.example.com;
  server srv2.example.com;
  server srv3.example.com;
 }
 server {
  listen 80;
  location / {
   proxy_pass http://myapp1;
  }
 }
}
```


这里的upstream没有写任何策略，则默认的是round robin轮询的，也就是一台一次的处理。

4）服务分发

比如有这样的服务器：

```
http {
upstream rtmp_backend {
    server 127.0.0.1:8090;
    server 127.0.0.1:9000;
}
  upstream flv_backend {
    server 127.0.0.1:8080;
}
```

我设置了这样的匹配规则：这里的意思是请求带上了8000?ttype=...,会根据参数ttype的值将请求分发给不同的服务器去处理，比如ttype为flv则分发给flv_backend

```
 map "$arg_ttype" $backend_server {
        "flv" flv_backend;
        "rtmp" rtmp_backend;
        default "172.16.1.2";
    }
```

可以做微服务的分发。

3.nginx的负载均衡策略

假设我写了两个简单的server，处理请求的逻辑如下，就是简单的返回了字符串+session，一个在8080端口服务，一个在8090端口

```
@GetMapping("/")
public String getHome() {
    if (hs.getAttribute( s: "token") == null){
        hs.setAttribute( s: "token",  o: "New");
        System.out.println("token is null");
    }
    else {
        System.out.println("token is NOT null");
        hs.setAttribute( s: "token",  o: "Old");
    }
    return "Let' start! Server One: " + hs.getAttribute( s: "token");
}
```


server.port=8080 server.port=8090

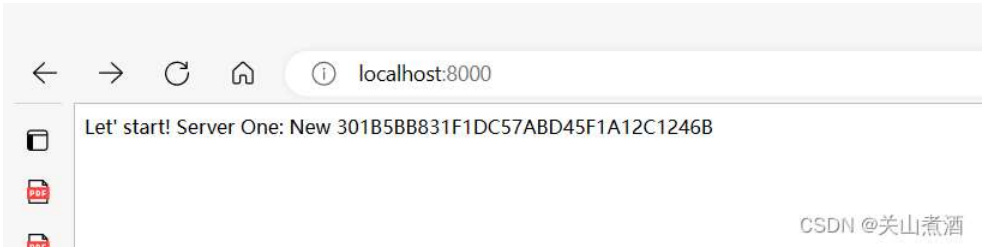
如下配置：

```
upstream myapp{
 server 127.0.0.1:8080;#相当于本机的localhost 8080
 server 127.0.0.1:8090;
 }
server {
#这里是在监听8000端口的请求
        listen      8000;
        server_name  localhost;
        #charset koi8-r;
        #access_log  logs/host.access.log  main;
        location / {
                root   html;
            proxy_pass http://myapp;
            index  index.html index.htm;
        }
```

1）round-robin

即一人一次

点击两次结果如下：



解释：RR策略就是轮流来，依次用server1,2,1,2。由于服务器不断切换，所以每次都需要new一个session出来，所以session每次都是新的。

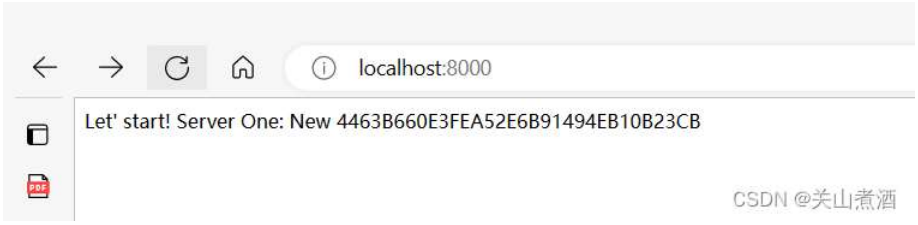
2）round-robin+权重配置

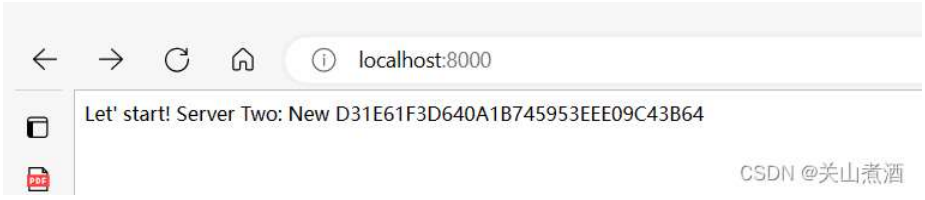
不同的server处理请求的能力不同，则给予不同的处理请求的权重

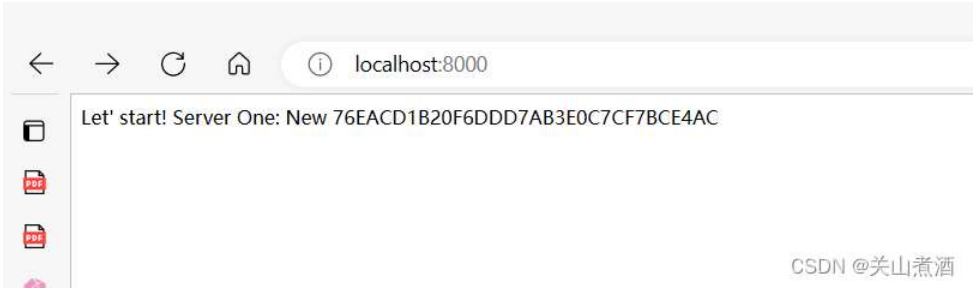
修改配置如下：

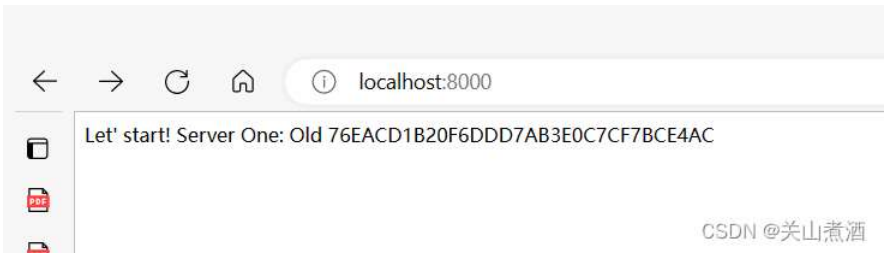
```
http {
upstream myapp{
 server 127.0.0.1:8080  weight=2;#增加权重
 server 127.0.0.1:8090;
 }
```

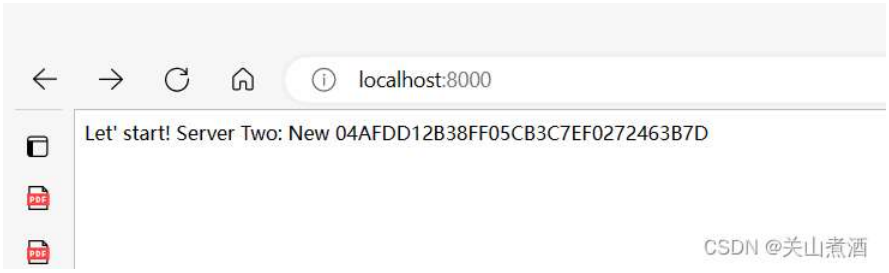
运行6次结果如下：

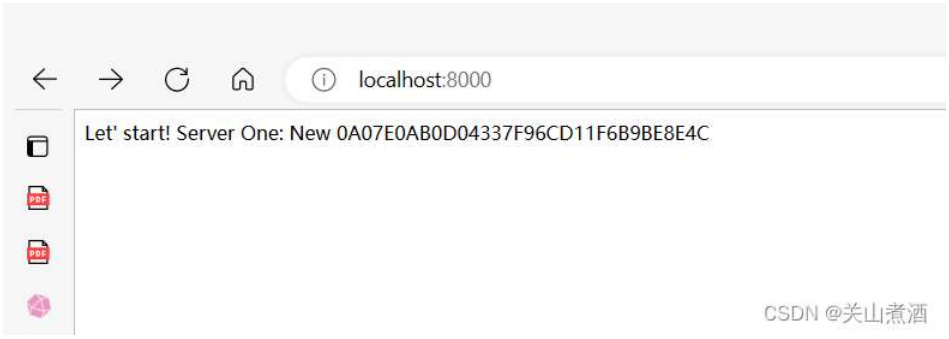












解释：由于在8080端口的server1权重为2，而8090端口的server2权重默认为1，所以6次运行是1,2,1,1,2,1,server1出现的比重为2/3。且第三第四次之间因为没有发生server的切换，所以session还是旧的。

3）第三种配置：ip Hash

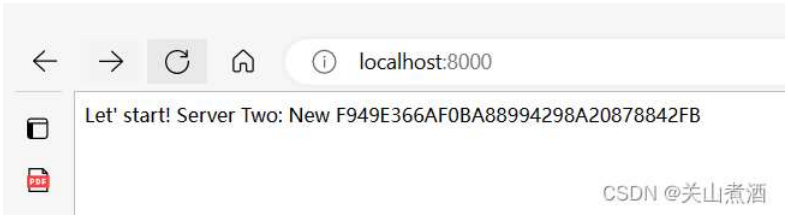
修改配置如下：

```
http {
upstream myapp{
ip_hash;
 server 127.0.0.1:8080  weight=2;
 server 127.0.0.1:8090;
 }
```

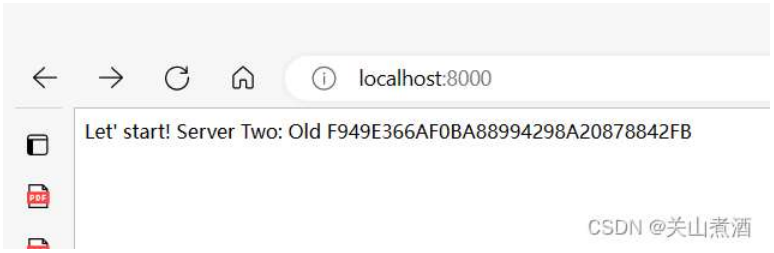
然后reload一下：

运行n次结果如下：

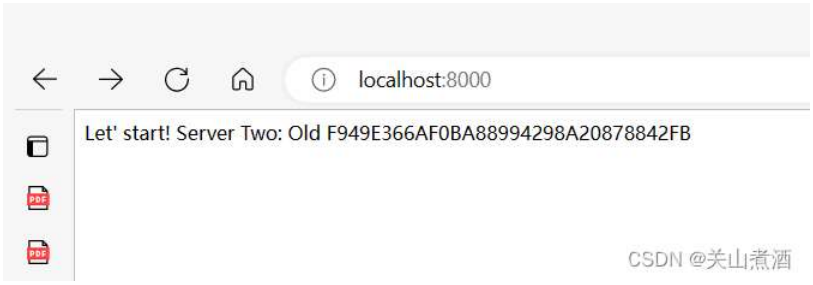
第一次：



第二次到第n次：



……



解释：

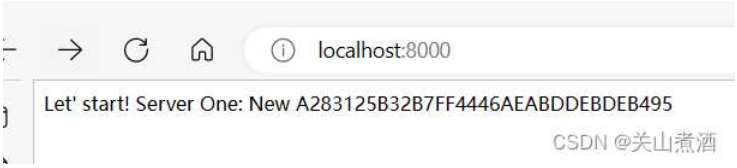
ip hash会根据访问的用户ip计算一个hash值，然后根据hash值来分配服务器，这里ip没有变，所以第一次访问到server2，出现new session之后就一直是用这个server处理请求的，server也不会改变。

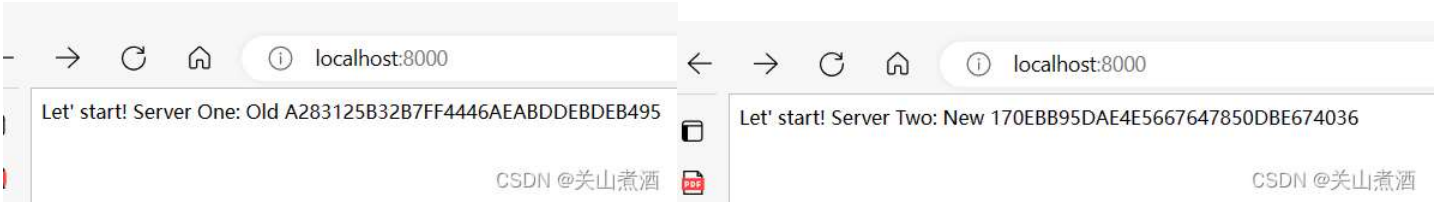
4）least-conn

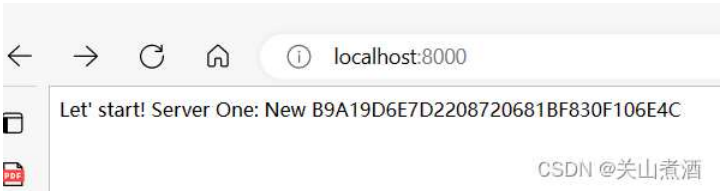
修改配置如下：

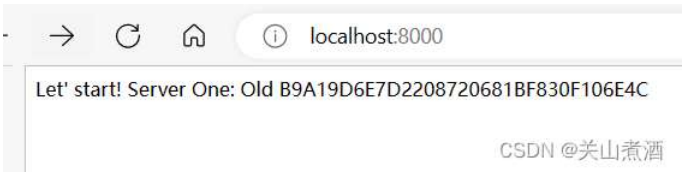
```
http {
upstream myapp{
#ip_hash;
least_conn;
 server 127.0.0.1:8080  weight=2;
 server 127.0.0.1:8090;
 }
```

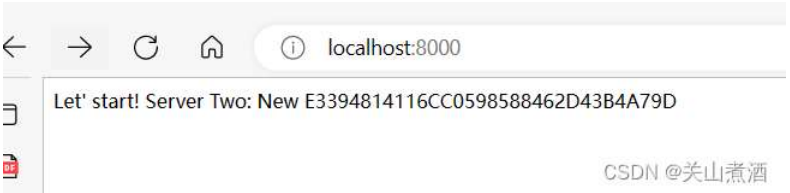
运行多次结果如下：











解释：

如果有些请求占用的时间很长，会导致其所在的后端负载较高。在这种场景下，把请求转发给连接数较少的后端，能够达到更好的负载均衡效果，这就是least_conn算法。

least_conn算法很简单，首选遍历后端集群，比较每个后端的conns/weight，选取该值最小的后端。

如果有多个后端的conns/weight值同为最小的，那么对它们采用加权轮询算法。

这里后端连接数都是1，且请求的时间都很短暂，所以在不修改权重的情况下和第二种配置出现的效果差不多。

4.nginx的用法

1）nginx的下载

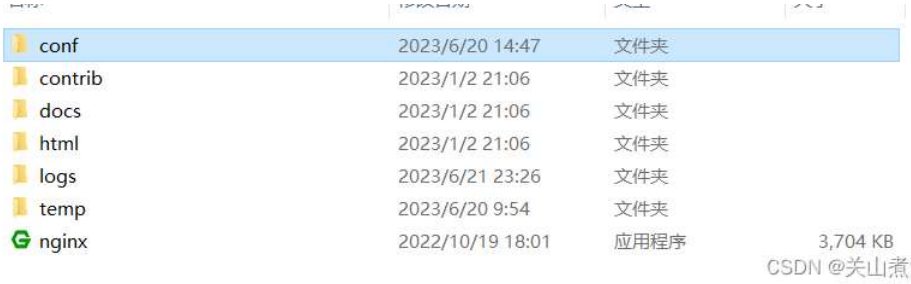
不同操作系统如下：

- Mac OS
  - $ sudo chown -R `whoami` /usr/local/Homebrew/
  - $ sudo chown -R $(whoami) $(brew --prefix)/*
  - $ sudo mkdir /usr/local/Frameworks
  - $ sudo chown -R `whoami` /usr/local/Frameworks/
  - $ brew install nginx
  - $ nginx

- Windows
  - Download and Unzip the stable version of Nginx on http://nginx.org/en/download.html
  - Run nginx.exe

window是直接去官网下载即可nginx官网

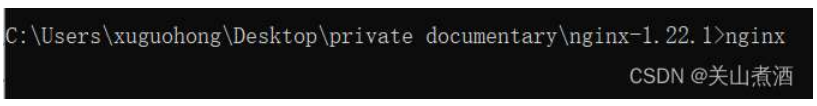
解压后到这个目录：（注意下载的文件所在路径最好不要有中文）



2）nginx的运行和操作

运行nginx可以直接点击图中的绿色的nginx.exe，也可以在当前目录下输入cmd打开命令行

然后：输入nginx即可



然后浏览器输入http:://localhost:8000看看是否可以看到：如果能看到则成功了！

然后其他的命令如下：

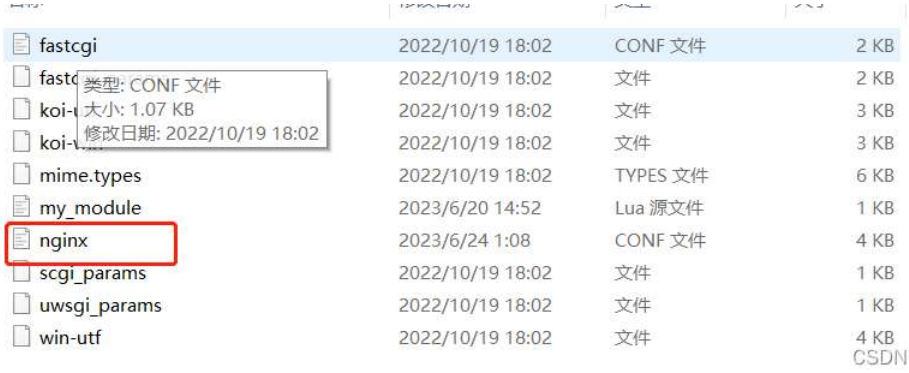
- `nginx -s signal` Where *signal* may be one of the following:
  - `stop` — fast shutdown
  - `quit` — graceful shutdown
  - `reload` — reloading the configuration file
  - `reopen` — reopening the log files

举例如下：

```
C:\Users\xuguohong\Desktop\private documentary\nginx-1.22.1>nginx -s quit

C:\Users\xuguohong\Desktop\private documentary\nginx-1.22.1>
```


然后配置文件在conf文件夹下的：



点开即可修改对应部分的配置。

记住每次修改配置之后都要运行一下nginx -s reload 配置才会生效。

5.总结

以上就是nginx的一个简单的用法说明，是笔者结合实验室实习+老师上课内容总结而成，如果对你有所帮助的话不妨收藏一下！谢谢！