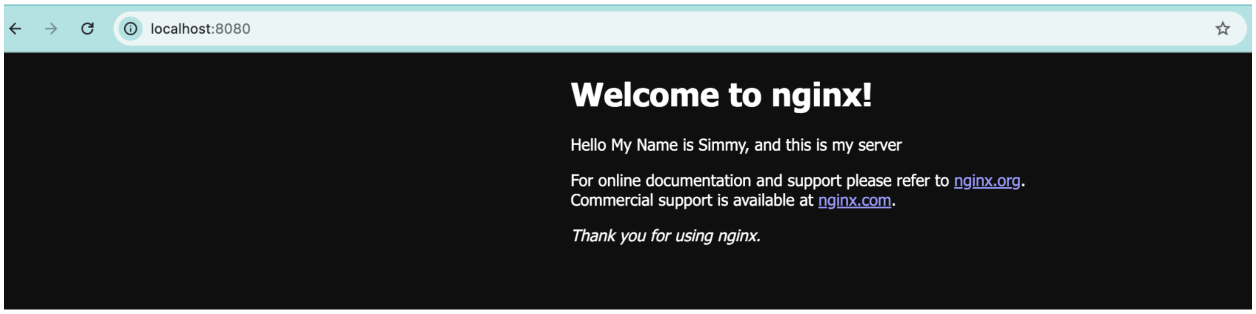
Assignment 1

Name: Harsimran Singh Dhillon
BNumber: B00983136

- Step 1:
  - Install NGINX using Homebrew
  - Terminal Command – brew install nginx -This will install nginx on your MacBook
  - Terminal Command – sudo nginx -This will start nginx
  - Enter the "localhost:8080" in the browser to check if nginx is started or not



Welcome to nginx!

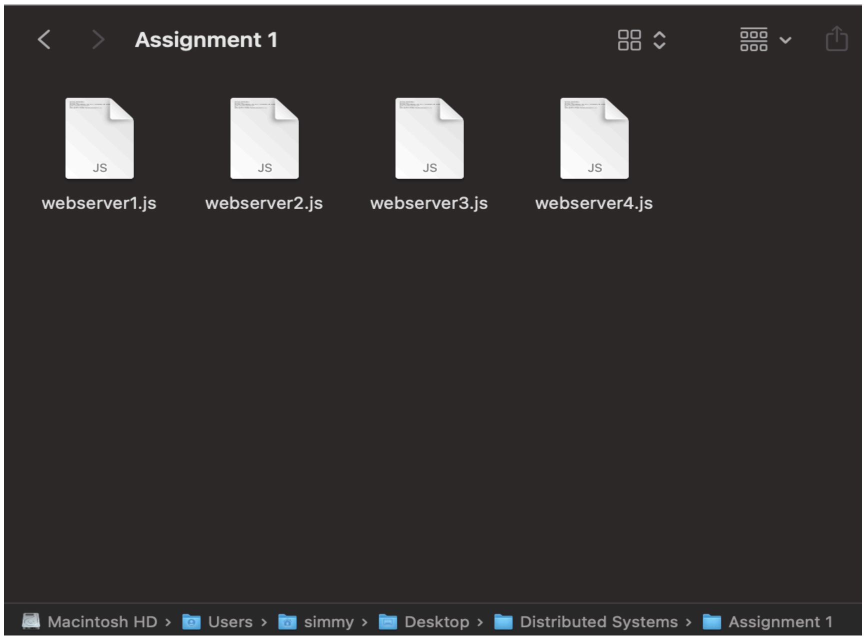
Hello My Name is Simmy, and this is my server

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

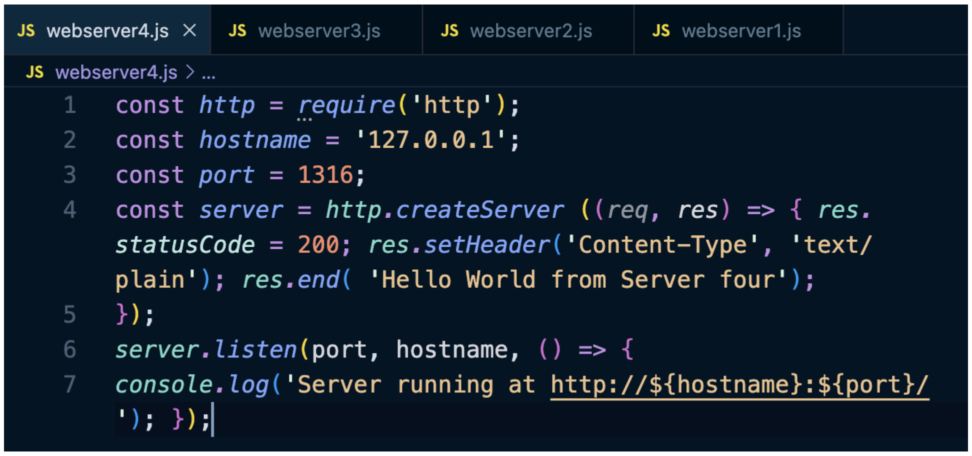
- Step 2:
  - Install Node.js using Homebrew
  - Terminal Command – brew install node - This will install Node.js on your MacBook

- Step 3:
    - Create a folder to store 4 web server files with ports 1313, 1314, 1315, and 1316, using server.listen() to listen to the function
    - When the response request is returned, the string "Hello world from server {number}" is written into HTTP as a request and displayed on the browser's web page



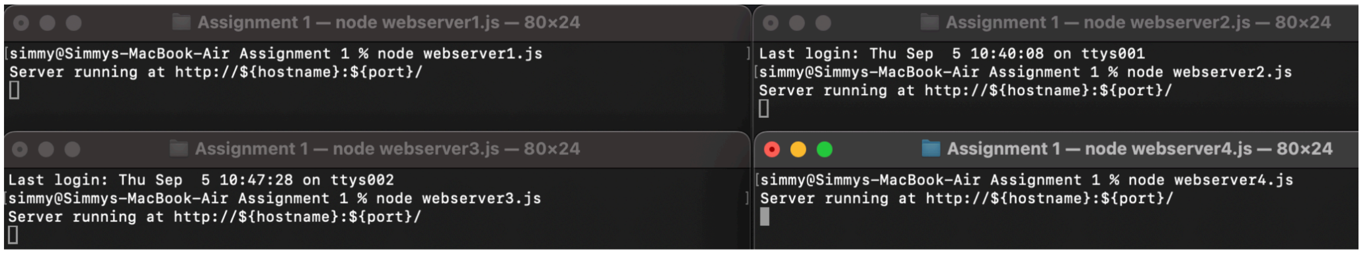
Assignment 1
webserver1.js
webserver2.js
webserver3.js
webserver4.js
Macintosh HD › Users › simmy › Desktop › Distributed Systems › Assignment 1

- Step 4:
    - Please add these lines to the above files mentioned
    - Your files should look like this (eg webserver4.js file)



```
const http = require('http');
const hostname = '127.0.0.1';
const port = 1316;
const server = http.createServer ((req, res) => { res.
statusCode = 200; res.setHeader('Content-Type', 'text/
plain'); res.end( 'Hello World from Server four');
});
server.listen(port, hostname, () => {
console.log('Server running at http://${hostname}:${port}/
'); });
```

- Step 5:
    - Enter the below command for each web server javascript file to run them


Assignment 1 — node webserver1.js — 80×24
Assignment 1 — node webserver2.js — 80×24
Assignment 1 — node webserver3.js — 80×24
Assignment 1 — node webserver4.js — 80×24

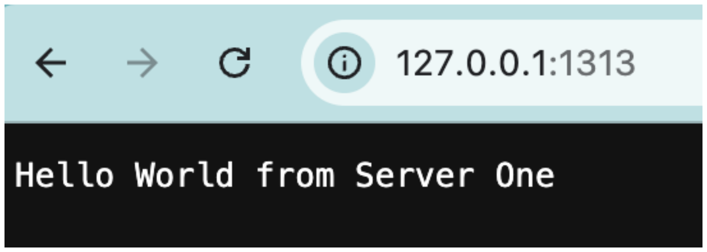
```
simmy@Simmys-MacBook-Air Assignment 1 % node webserver1.js
Server running at http://${hostname}:${port}/
```

```
Last login: Thu Sep  5 10:40:08 on ttys001
simmy@Simmys-MacBook-Air Assignment 1 % node webserver2.js
Server running at http://${hostname}:${port}/
```

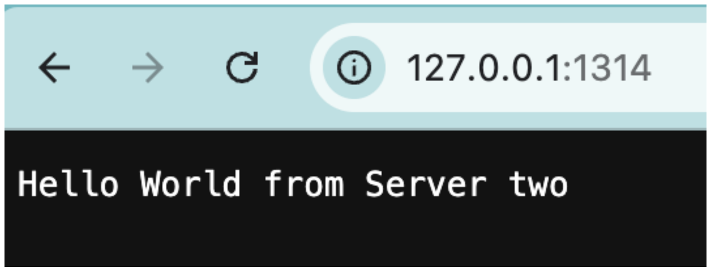
```
Last login: Thu Sep  5 10:47:28 on ttys002
simmy@Simmys-MacBook-Air Assignment 1 % node webserver3.js
Server running at http://${hostname}:${port}/
```

```
simmy@Simmys-MacBook-Air Assignment 1 % node webserver4.js
Server running at http://${hostname}:${port}/
```

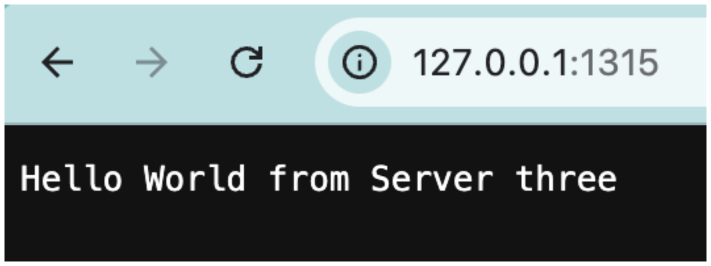
- Step 6:
    - To check whether our server is running or not please type in the below IP address and Port number on your web browser



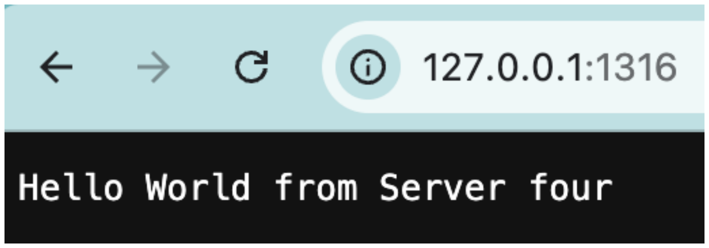
```
Hello World from Server One
```



```
Hello World from Server two
```

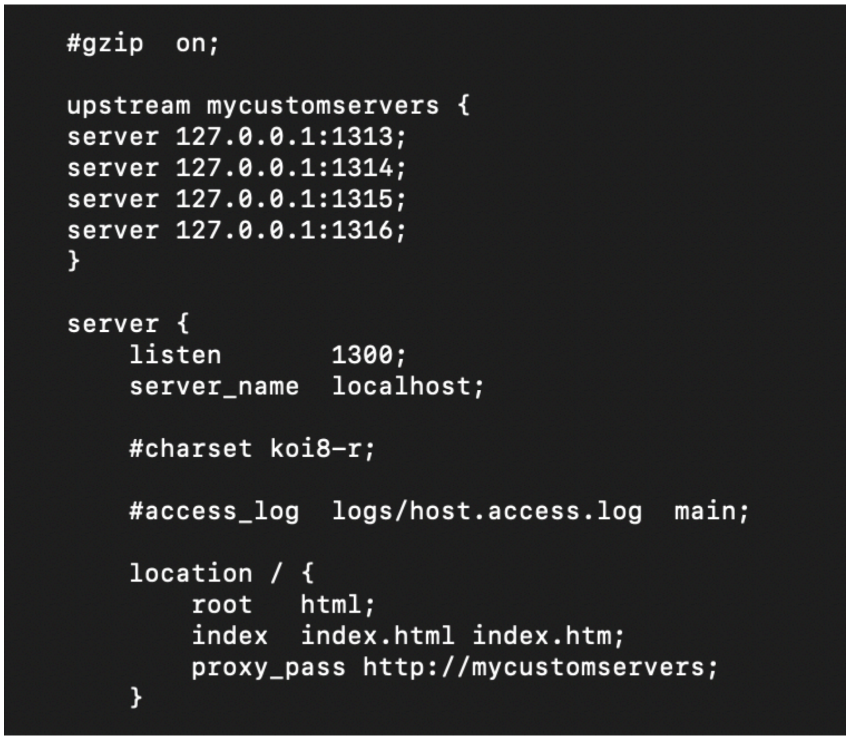


```
Hello World from Server three
```



```
Hello World from Server four
```

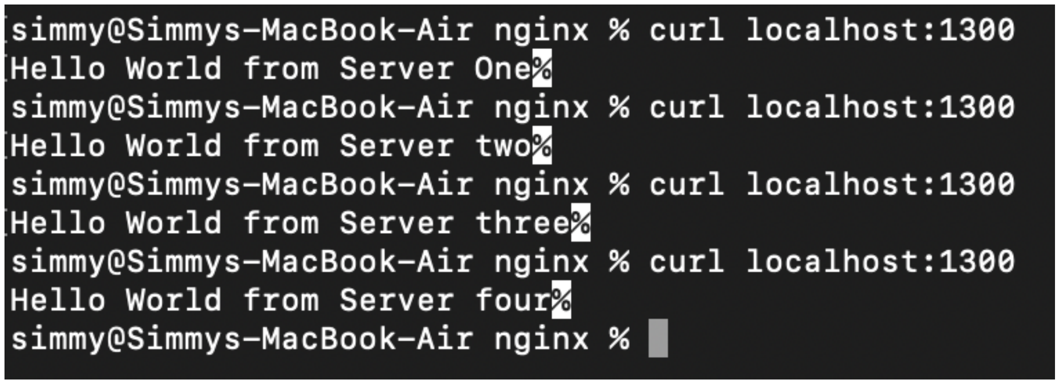
- Step 7:
  - Open the Nginx.conf file from the address of /opt/homebrew/etc/nginx
  - You can use the “vi nginx.conf” command to edit the file
  - Add the upstream mycustomservers function in the file
  - Change the server listen port number from 8080 to any of your choice (In the above example I changed it to 1300)



```
#gzip  on;

upstream mycustomservers {
server 127.0.0.1:1313;
server 127.0.0.1:1314;
server 127.0.0.1:1315;
server 127.0.0.1:1316;
}

server {
    listen       1300;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
        proxy_pass http://mycustomservers;
    }
```

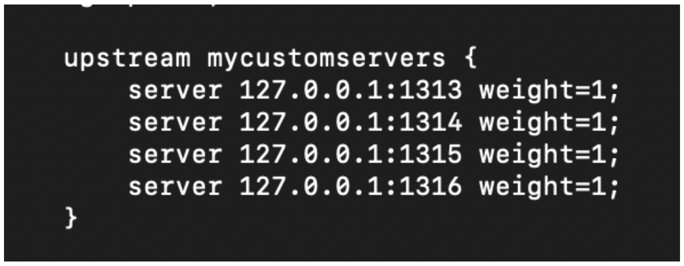
- Step 8:
  - Reload the nginx server by using the command “sudo nginx -s reload” every time when you want to change a load balancing strategy

- Step 9:
  - Run “curl localhost:1300” in the command prompt to send an HTTP request
  - The NGINX proxy server directs this request to a server in the cluster, which then sends the response back through the proxy
  - NGINX uses Round Robin technique by default

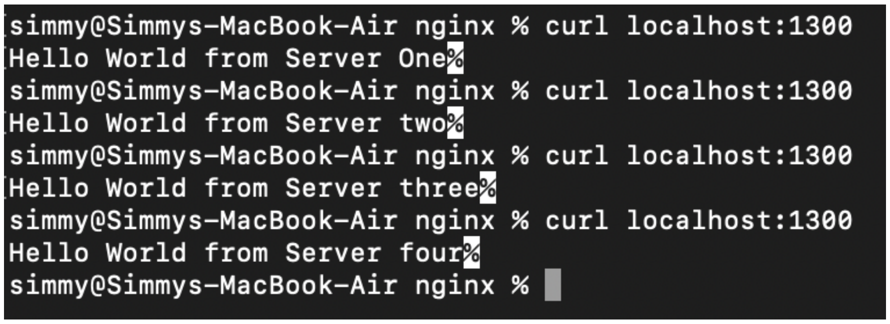


```
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server One%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server two%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server three%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server four%
simmy@Simmys-MacBook-Air nginx %
```

- Step 10:
  - **Load Balancing Strategies:**
    1. Round Robin:



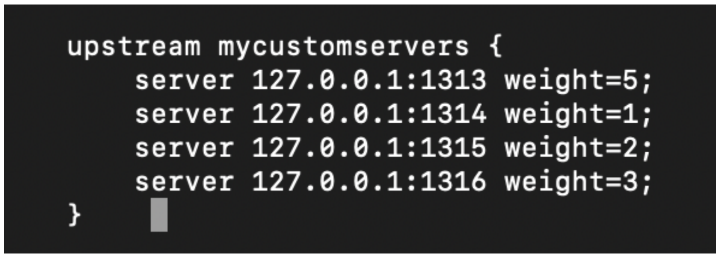
```
    upstream mycustomservers {
        server 127.0.0.1:1313 weight=1;
        server 127.0.0.1:1314 weight=1;
        server 127.0.0.1:1315 weight=1;
        server 127.0.0.1:1316 weight=1;
    }
```



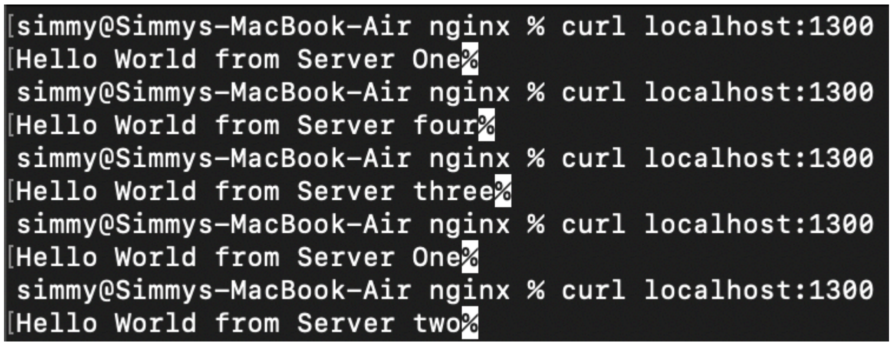
```
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server One%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server two%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server three%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server four%
simmy@Simmys-MacBook-Air nginx %
```

    2. **Weighted Round Robin:**

a. Change the weights to your desired values

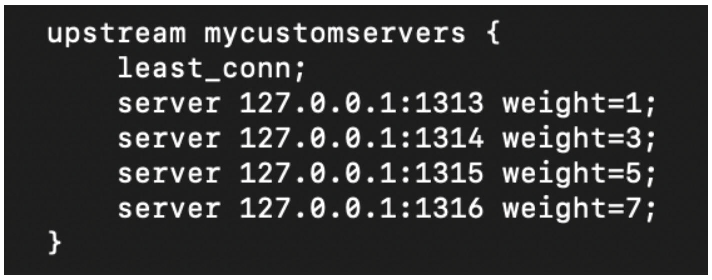


```
upstream mycustomservers {
    server 127.0.0.1:1313 weight=5;
    server 127.0.0.1:1314 weight=1;
    server 127.0.0.1:1315 weight=2;
    server 127.0.0.1:1316 weight=3;
}
```

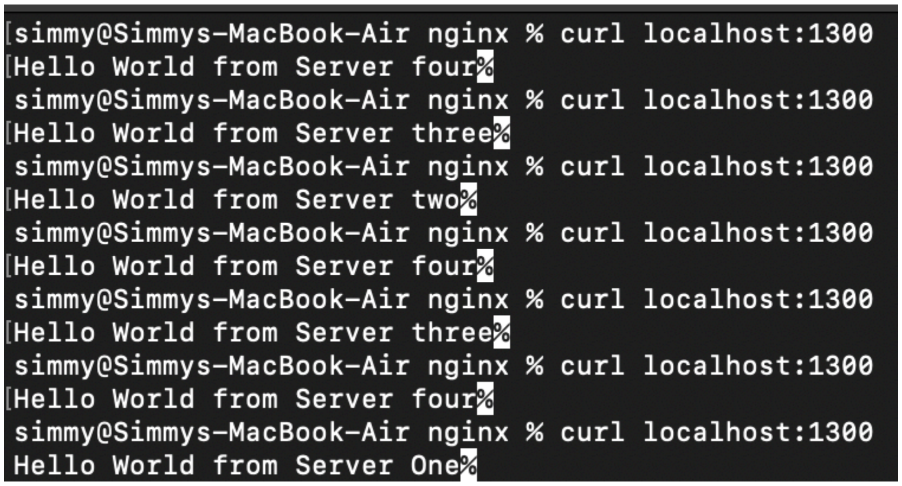


```
[simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server One%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server four%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server three%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server One%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server two%
```

3. **Least Connection:**
   a. For the least connection, we just add **least_conn** on the first line and **weights** also with each server



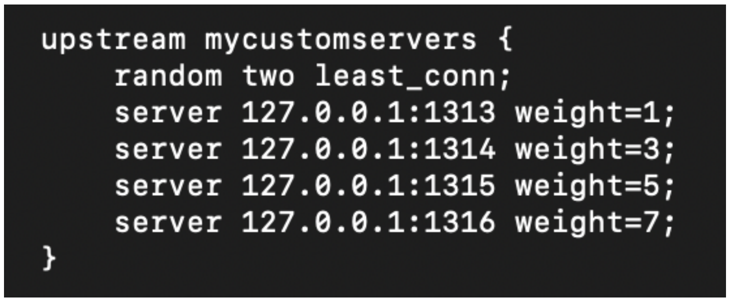
```
upstream mycustomservers {
    least_conn;
    server 127.0.0.1:1313 weight=1;
    server 127.0.0.1:1314 weight=3;
    server 127.0.0.1:1315 weight=5;
    server 127.0.0.1:1316 weight=7;
}
```



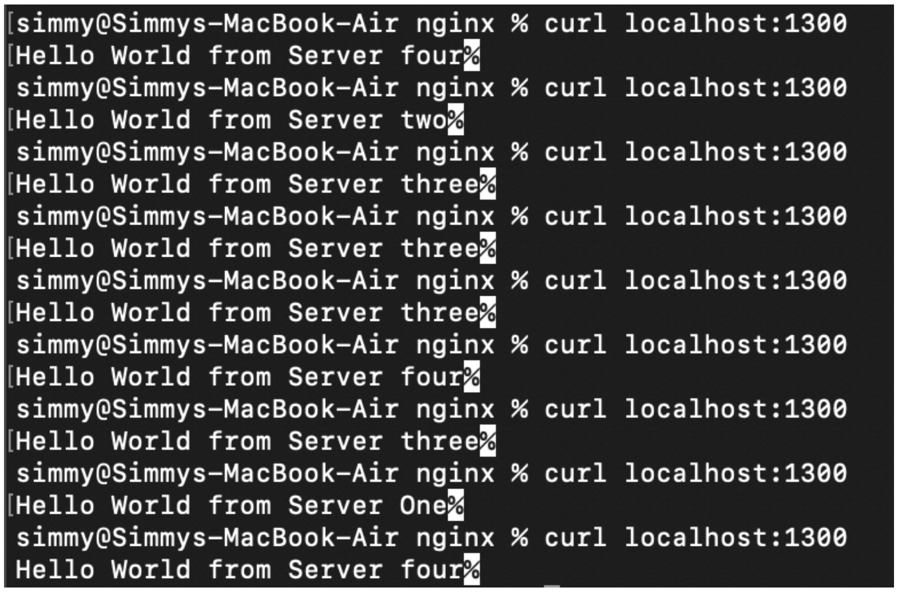
```
[simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server four%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server three%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server two%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server four%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server three%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server four%
 simmy@Simmys-MacBook-Air nginx % curl localhost:1300
 Hello World from Server One%
```

4. **Random:**

a. In the random load balancing strategy you add the random keyword and two in this example specifies the number of parameters.
b. For the first parameter, NGINX randomly selects two servers taking into account server weights
c. For the second parameter we have specified a condition of least_conn, so it will take into consideration the least connections.



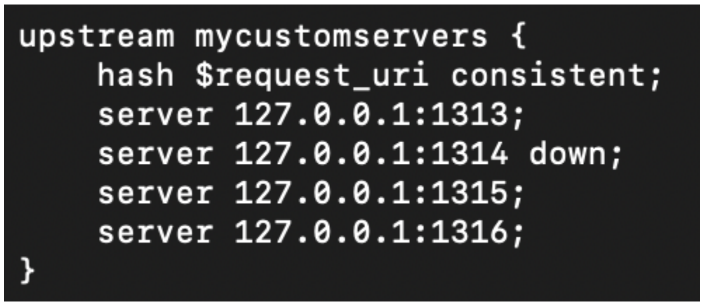
```
upstream mycustomservers {
    random two least_conn;
    server 127.0.0.1:1313 weight=1;
    server 127.0.0.1:1314 weight=3;
    server 127.0.0.1:1315 weight=5;
    server 127.0.0.1:1316 weight=7;
}
```



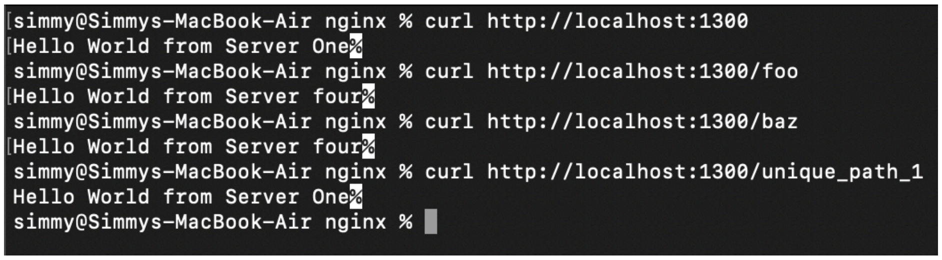
```
[simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server four%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server two%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server three%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server three%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server three%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server four%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server three%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
[Hello World from Server One%
simmy@Simmys-MacBook-Air nginx % curl localhost:1300
Hello World from Server four%
```

5. Generic Hash:

a. For Generic Hash, we add the hash keyword and $request_uri consistent in front of it
b. We have made server 2 down by adding a down keyword in front of it



```
upstream mycustomservers {
    hash $request_uri consistent;
    server 127.0.0.1:1313;
    server 127.0.0.1:1314 down;
    server 127.0.0.1:1315;
    server 127.0.0.1:1316;
}
```



```
[simmy@Simmys-MacBook-Air nginx % curl http://localhost:1300
[Hello World from Server One%
 simmy@Simmys-MacBook-Air nginx % curl http://localhost:1300/foo
[Hello World from Server four%
 simmy@Simmys-MacBook-Air nginx % curl http://localhost:1300/baz
[Hello World from Server four%
 simmy@Simmys-MacBook-Air nginx % curl http://localhost:1300/unique_path_1
 Hello World from Server One%
 simmy@Simmys-MacBook-Air nginx %
```