

Reverse proxy

* Request goes to nginx server. It fetches the data from servers and sends it back to client

Single point encryption

* All security checks are handled at Nginx server

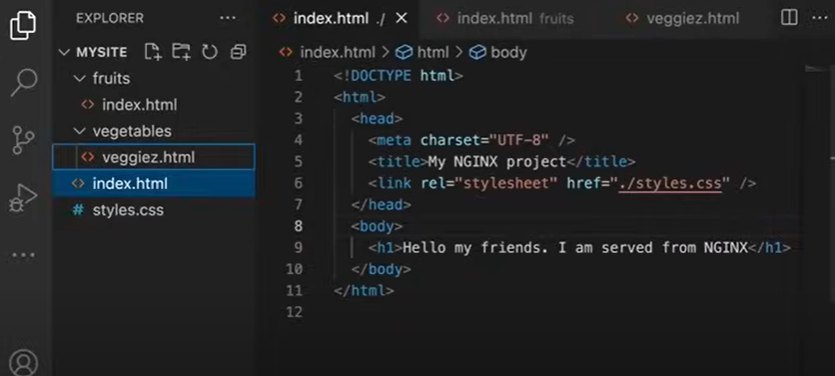
Load balancer

* It uses round robin or other algorithm to regulate traffic

Advantage over Apache

* Faster
* Lightweight
* Simple configuration
* Popular in container world

**nginx.conf**

****

worker\_processes 1; //line 16

events {

worker\_connections 1024; //line 17

}

http {

types { //line 1

text/css css; //line 2

text/html html; //line 3

} //line 4

upstream backendserver { //line 14

least\_conn; //line 19

server 127.0.0.1:1111

server 127.0.0.1:2222

server 127.0.0.1:3333

server 127.0.0.1:4444

}

location / {

proxy\_pass <http://backendserver/>; //line 15

proxy\_set\_header Host $host; //line 18

proxy\_set\_header X-Real-IP $remote\_addr; //line 19

}

server {

listen 8080;

root /Users/shakthi/Desktop/mysite; //line 6

location /fruits {

root /Users/shakthi/Desktop/mysite; //line 7

}

location /carbs {

alias /Users/shakthi/Desktop/mysite/fruits; //line 8

}

location /vegetables {

root /Users/shakthi/Desktop/mysite; //line 9

try\_files /vegetables/veggiez.html /index.html =404; //line 10

}

location ~\* /count/[0-9] { //line 11

root /Users/shakthi/Desktop/mysite;

try\_files /index.html =404;

}

location /crops {

return 307 /fruits; //line 12

}

rewrite ^number/(\w+) /count/$1; //line 13

}

//necessary. Else .css file will be sent to browser but will not be utilised as it cannot read the file type

//line 1 – 4 can be replaced by (include mime.types) , mime.types is a default file provided by nginx that has all types

//line 6 - content in mysite/index.html will be rendered when user hits (localhost:8080)

//line 7 - content in mysite/fruits/index.html will be rendered when user hits (localhost:8080/fruits)

//line 8 - content in mysite/fruits/index.html will be rendered when user hits (localhost:8080/carbs)

//line 9 - content in mysite/vegetables/index.html will be rendered when user hits (localhost:8080/vegetables)

//line 10 – if mysite/vegetables/index.html can not be found display mysite/vegetables/veggiez.html,

If mysite/vegetables/veggiez.html cannot be found display mysite/index.html,

If mysite/index.html can not be found display 404

//line 11 – display mysite/index.html when user hits (localhost/count/(any no. between 0 - 9))

//line 12 – redirect to /fruits when user hits /crops

//line 13 – redirect to count/1

//line 14 – use the four servers in a round robin technique

//line 15 – specifying the servers

// line 16 – specifies how many worker process should the server start with

Should be tuned according to CPU and traffic

Therefore leave it as worker\_processes auto

//line 17 – how many connections / worker\_process

Increases memory usage

worker\_connections 1024 – meaning we can serve 1024 clients simultaneously

//line 18 – send all info from client to server

//line 19 – send client IP to server

Because server sees request being sent from nginx, not from client, so, nginx needs to send all information from client to server. IP of client too, as it will be needed for logging or other such puroposes

//line 6 – 13 (server {}) should not be ideally used and we should let the servers handle the request

//line 19 – scheduling algorithm, can be ignored, defaults to round robin