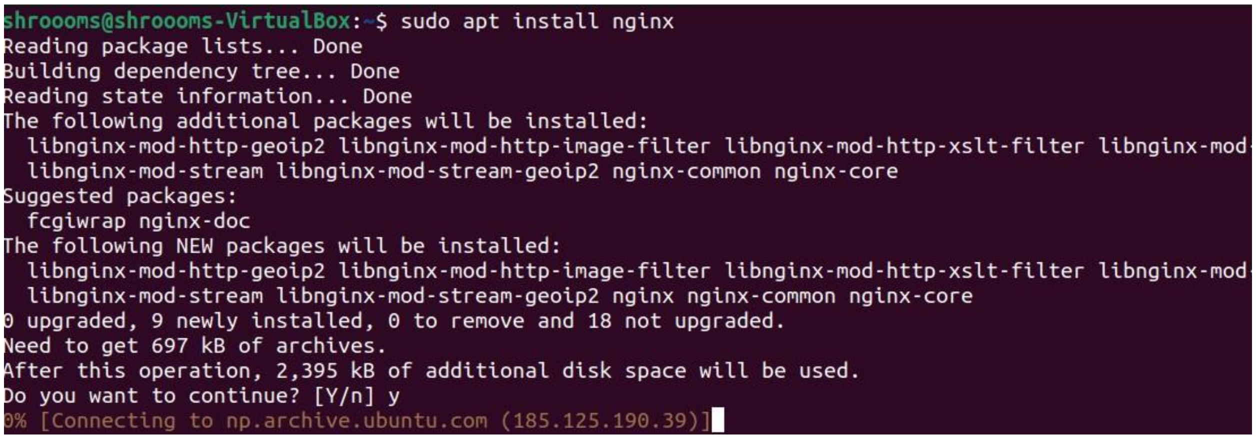
Assignment 4 - Sharams Kunwar – NginX

Hands-On

1. Installing NginX and Hosting Simple Index Page

Nginx was installed using command ‘sudo apt install nginx’.

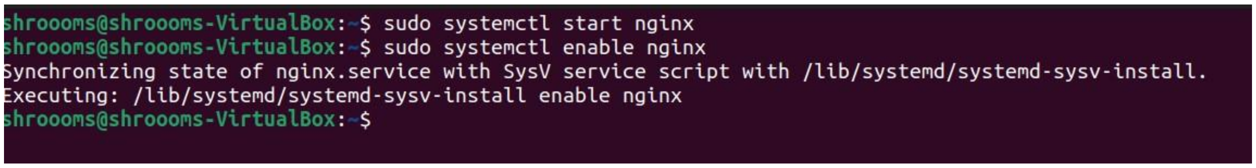
Screenshot:



```
shrooons@shrooons-VirtualBox:~$ sudo apt install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod
  libnginx-mod-stream libnginx-mod-stream-geoip2 nginx-common nginx-core
Suggested packages:
  fcgiwrap nginx-doc
The following NEW packages will be installed:
  libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod
  libnginx-mod-stream libnginx-mod-stream-geoip2 nginx nginx-common nginx-core
0 upgraded, 9 newly installed, 0 to remove and 18 not upgraded.
Need to get 697 kB of archives.
After this operation, 2,395 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
0% [Connecting to np.archive.ubuntu.com (185.125.190.39)]
```

To further set up Nginx, nginx service was started and it was made to enable on boot. Likewise, firewall had also been disabled to avoid issue during hands-on. However, it’s not the best practice.

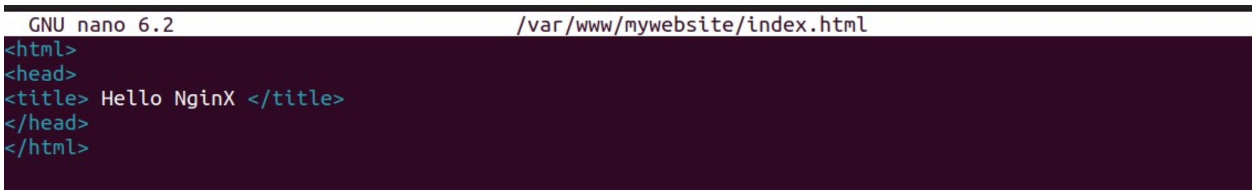
Screenshot:



```
shrooons@shrooons-VirtualBox:~$ sudo systemctl start nginx
shrooons@shrooons-VirtualBox:~$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
shrooons@shrooons-VirtualBox:~$
```

Then, a simple index.html page was set up inside var/www/mywebsite with the title ‘Hello Nginx’.

Screenshot:



```
  GNU nano 6.2                                /var/www/mywebsite/index.html
<html>
<head>
<title> Hello NginX </title>
</head>
</html>
```

Following Nginx Configuration was used to set up a server for mywebsite.

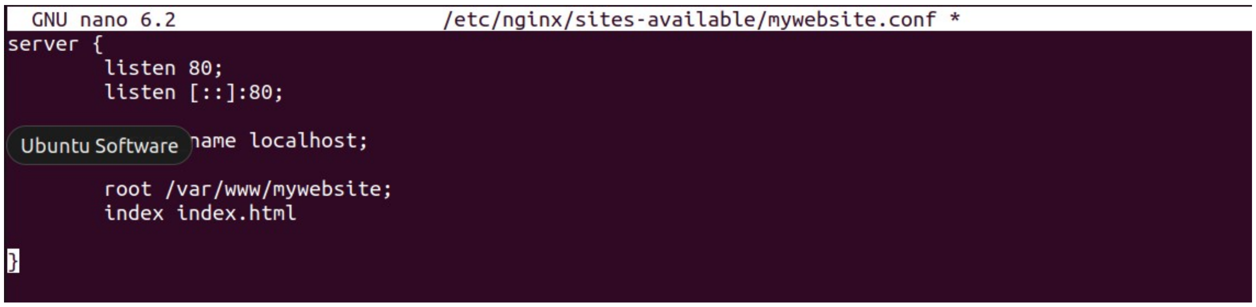
The server directive outlines a virtual server, which is simply a configuration block containing settings for the landing page.

The listen directive specifies the server will listen on port 80 for both Ipv4 and Ipv6 requests and the website will be accessible at localhost.

The root directive specifies the directory on server where the websites files are located, i.e., /var/www/mywebsite directory.

Likewise, the index directory specifies the name of the file to be served upon request to the root directory of the website.

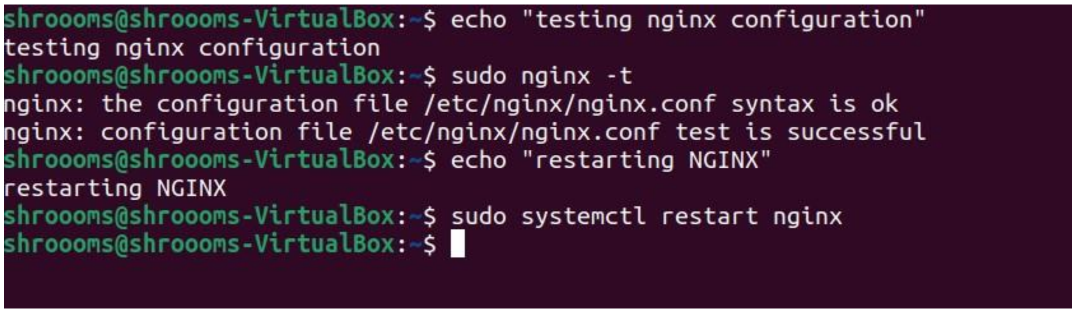
Screenshot:



```
  GNU nano 6.2                        /etc/nginx/sites-available/mywebsite.conf *
server {
        listen 80;
        listen [::]:80;

 Ubuntu Software name localhost;

        root /var/www/mywebsite;
        index index.html

}
```

The configuration was then tested for any syntax error and then the NginX service was restarted.

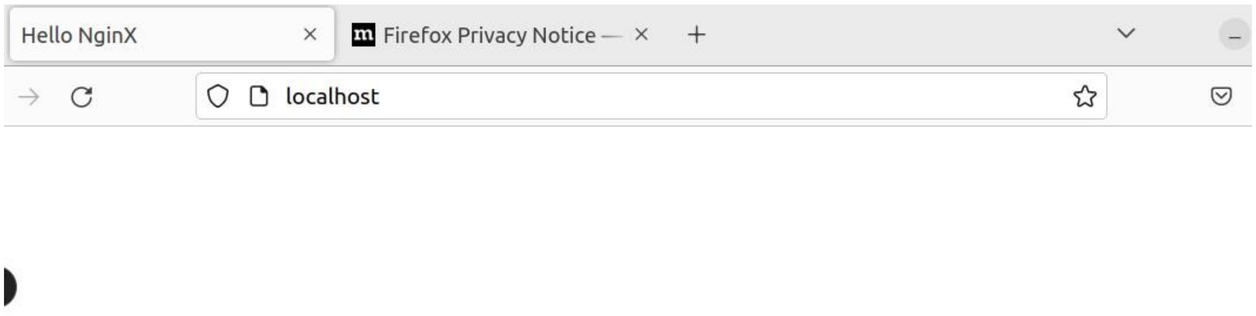
Screenshot:



```
shroooms@shroooms-VirtualBox:~$ echo "testing nginx configuration"
testing nginx configuration
shroooms@shroooms-VirtualBox:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
shroooms@shroooms-VirtualBox:~$ echo "restarting NGINX"
restarting NGINX
shroooms@shroooms-VirtualBox:~$ sudo systemctl restart nginx
shroooms@shroooms-VirtualBox:~$
```

Upon navigating the port 80 of localhost, using http://localhost:80 , we were landed on the index page.

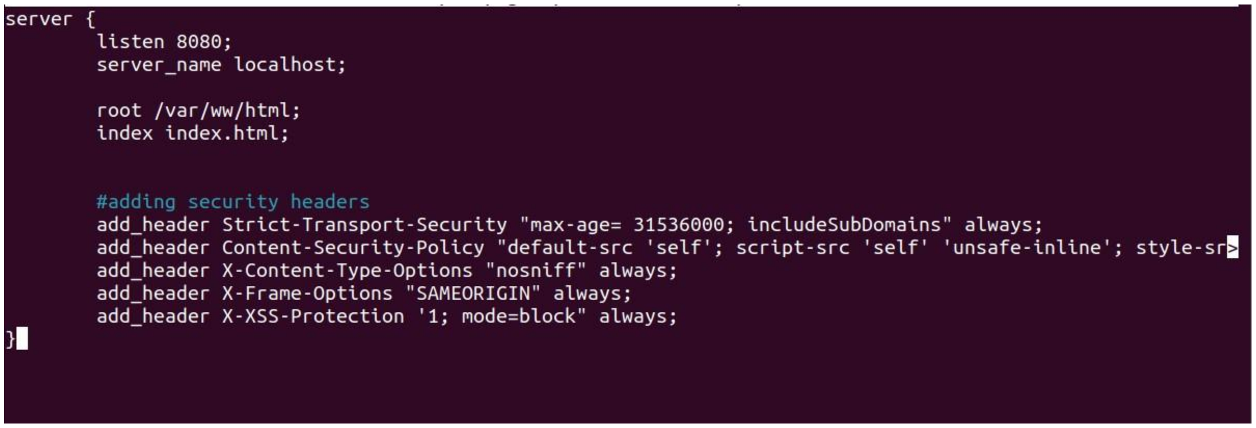
Screenshot:



2. NginX Header Security

Security headers were implemented on test.conf file. Let’s have a look on each purpose of the headers.

Screenshot:



```
server {
        listen 8080;
        server_name localhost;

        root /var/ww/html;
        index index.html;


        #adding security headers
        add_header Strict-Transport-Security "max-age= 31536000; includeSubDomains" always;
        add_header Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline'; style-sr>
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header X-XSS-Protection '1; mode=block" always;
}
```

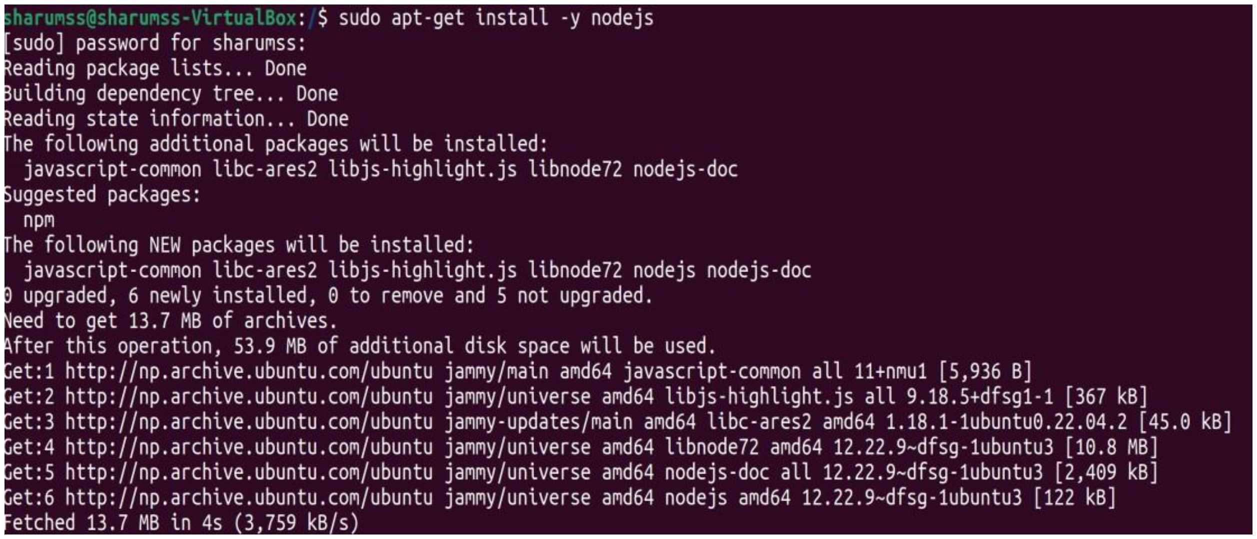
Security Headers and Description:

Header	Description
Strict-Transport-Security (HSTS)	Forces browsers to use HTTPS when connecting to the website. This helps to protect against man-in-the-middle attacks.
Content-Security-Policy (CSP)	Specifies which resources are allowed to be loaded by the website. This helps to prevent attackers from injecting malicious code into the website.
X-Content-Type-Options	Prevents browsers from sniffing the MIME type of resources. This helps to prevent attackers from injecting malicious code into the website.
X-Frame-Options	Prevents the website from being embedded in a frame on another website. This helps to prevent clickjacking attacks.
X-XSS-Protection	Enables XSS filtering in the browser. This helps to protect against XSS attacks.

3. Reverse Proxy all http requests to NodeJS API

Node.js runtime environment was installed on Ubuntu Linux to develop and run JavaScript applications.

Screenshot:

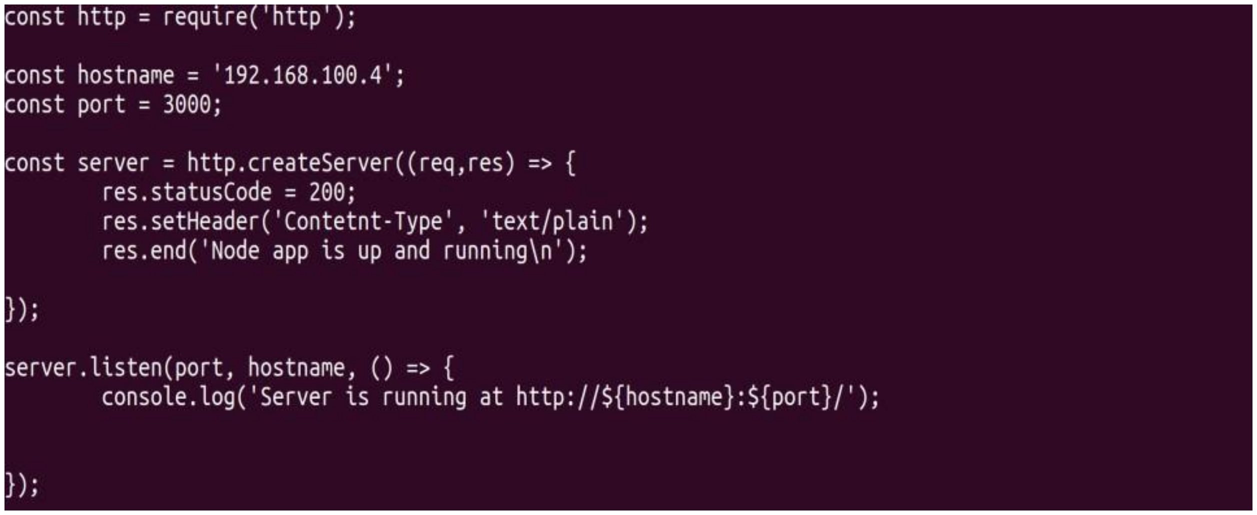


```
sharumss@sharumss-VirtualBox:/$ sudo apt-get install -y nodejs
[sudo] password for sharumss:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  javascript-common libc-ares2 libjs-highlight.js libnode72 nodejs-doc
Suggested packages:
  npm
The following NEW packages will be installed:
  javascript-common libc-ares2 libjs-highlight.js libnode72 nodejs nodejs-doc
0 upgraded, 6 newly installed, 0 to remove and 5 not upgraded.
Need to get 13.7 MB of archives.
After this operation, 53.9 MB of additional disk space will be used.
Get:1 http://np.archive.ubuntu.com/ubuntu jammy/main amd64 javascript-common all 11+nmu1 [5,936 B]
Get:2 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 libjs-highlight.js all 9.18.5+dfsg1-1 [367 kB]
Get:3 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libc-ares2 amd64 1.18.1-1ubuntu0.22.04.2 [45.0 kB]
Get:4 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 libnode72 amd64 12.22.9~dfsg-1ubuntu3 [10.8 MB]
Get:5 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 nodejs-doc all 12.22.9~dfsg-1ubuntu3 [2,409 kB]
Get:6 http://np.archive.ubuntu.com/ubuntu jammy/universe amd64 nodejs amd64 12.22.9~dfsg-1ubuntu3 [122 kB]
Fetched 13.7 MB in 4s (3,759 kB/s)
```

Then, a simple Node.js Server listening on port 3000 was set up, which responds with 'Node app is up and running' upon requests. In detail, following is what happens:

- Client makes requests and the server starts processing it upon receiving.
- Server then creates a response object and sets up response headers.
- It also writes response body to response object and then sends response back to client.
- Client then displays it to user.

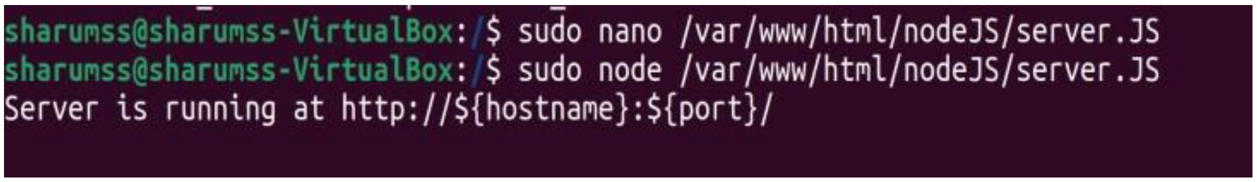
Screenshot:



```
const http = require('http');

const hostname = '192.168.100.4';
const port = 3000;

const server = http.createServer((req,res) => {
        res.statusCode = 200;
        res.setHeader('Contetnt-Type', 'text/plain');
        res.end('Node app is up and running\n');

});

server.listen(port, hostname, () => {
        console.log('Server is running at http://${hostname}:${port}/');


});
```

Then, the created server was run.

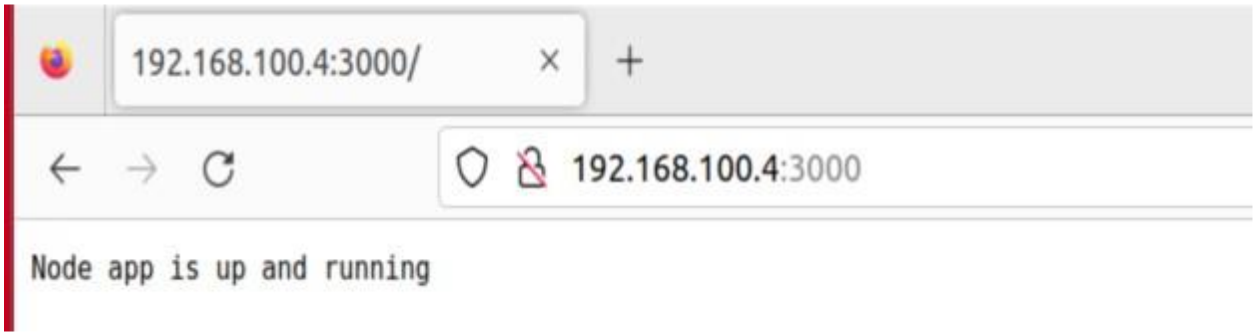
Screenshot:



```
sharumss@sharumss-VirtualBox:/$ sudo nano /var/www/html/nodeJS/server.JS
sharumss@sharumss-VirtualBox:/$ sudo node /var/www/html/nodeJS/server.JS
Server is running at http://${hostname}:${port}/
```

Upon navigating http://192.168.100.4:3000/ we get following result:

Screenshot:

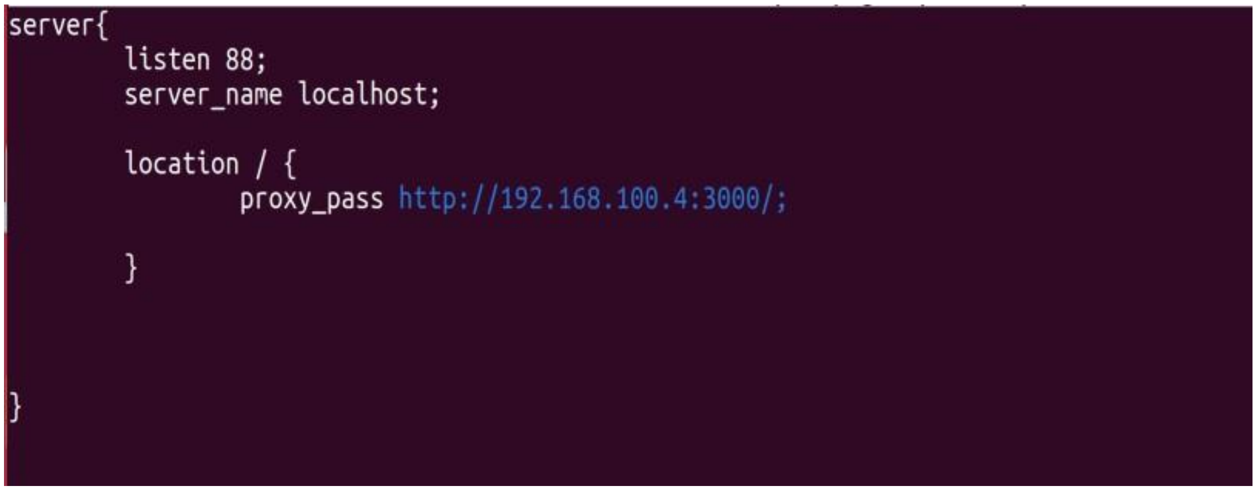


```
Node app is up and running
```

Then, the following configuration was used to set up a reverse proxy for the Node.js server.

The configuration tells NginX to listen for requests on port 88 and then forward all requests to server running on port 3000.

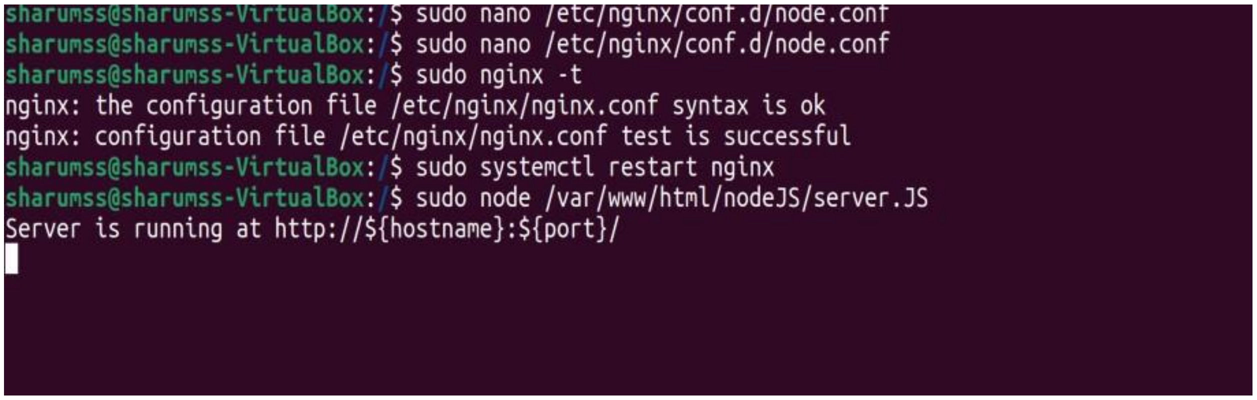
Screenshot:



```
server{
        listen 88;
        server_name localhost;

        location / {
                proxy_pass http://192.168.100.4:3000/;

        }



}
```

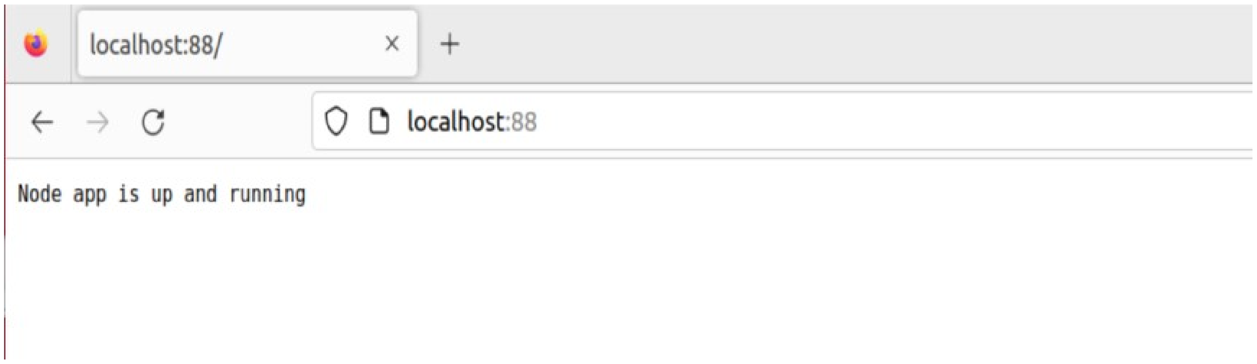
Then it was tested and and nginx service was restarted.

Screenshot:



```
sharumss@sharumss-VirtualBox:/$ sudo nano /etc/nginx/conf.d/node.conf
sharumss@sharumss-VirtualBox:/$ sudo nano /etc/nginx/conf.d/node.conf
sharumss@sharumss-VirtualBox:/$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
sharumss@sharumss-VirtualBox:/$ sudo systemctl restart nginx
sharumss@sharumss-VirtualBox:/$ sudo node /var/www/html/nodeJS/server.JS
Server is running at http://${hostname}:${port}/
```

Upon navigating port 88 of localhost, we can get following results.
Screenshot:



```
Node app is up and running
```

Hence, reverse proxy was set up.

4. Use of Proxies and Types

Proxies have two types:

- Forward Proxy:
  It sits between client and internet intercepting all requests from client and forwarding them to appropriate server on internet. The server then sends response back to the forward proxy, which then forwards to client.

- Reverse Proxy:

  It sits between client and the server intercepting all requests from clients and forwarding them to appropriate server. The server then sends the response back to the proxy, which then forwards it to the client.

Main difference is notable upon their use cases.

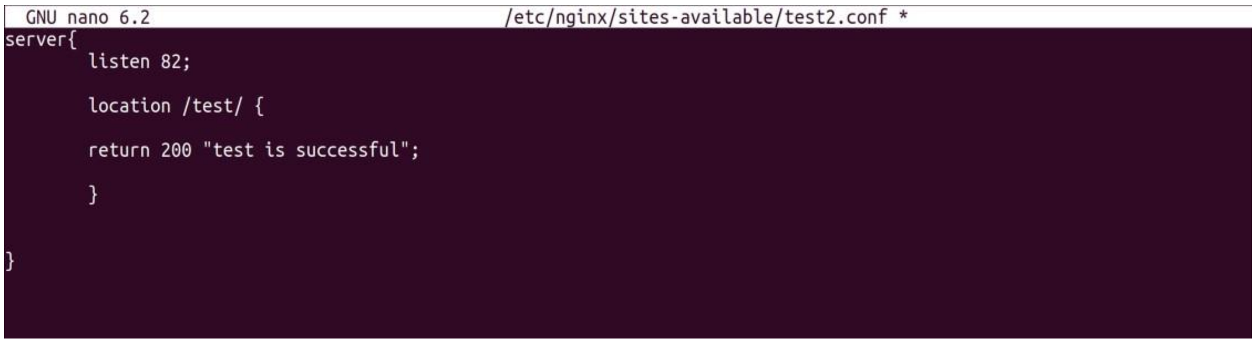
Forward proxy is used to hide the client's IP address from the server, while reverse proxy is used to distribute traffic to multiple servers, ensuring load balancing and additional security.

Reverse proxy can also be used to cache static content improving performance and content/ traffic filtering as well.

5. Creating a test2.conf listening on port 82 and to location /test/ and reverse proxy all traffic of port 82 to port 85

Test2.conf file was created, which was set up to listen on port 82 and display message 'test is successful' upon receiving status code 200.

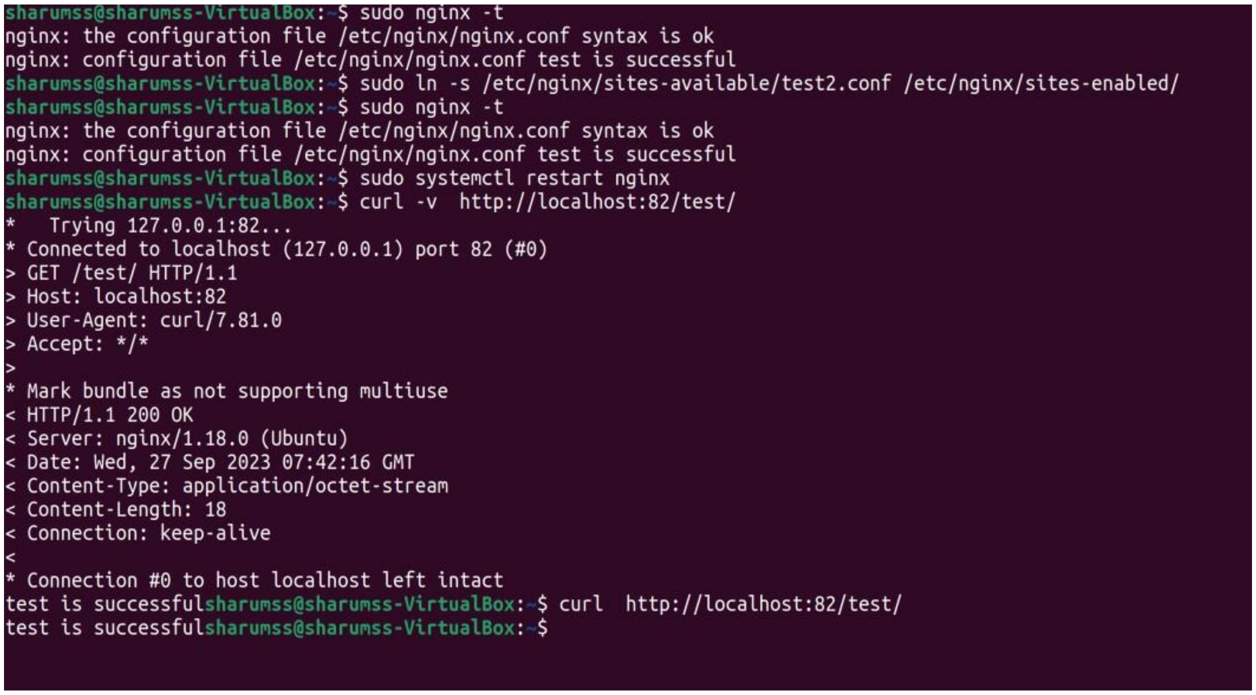
Screenshot:



```
  GNU nano 6.2                              /etc/nginx/sites-available/test2.conf *
server{
        listen 82;

        location /test/ {

        return 200 "test is successful";

        }


}
```

Using curl, the message was successfully retrieved.

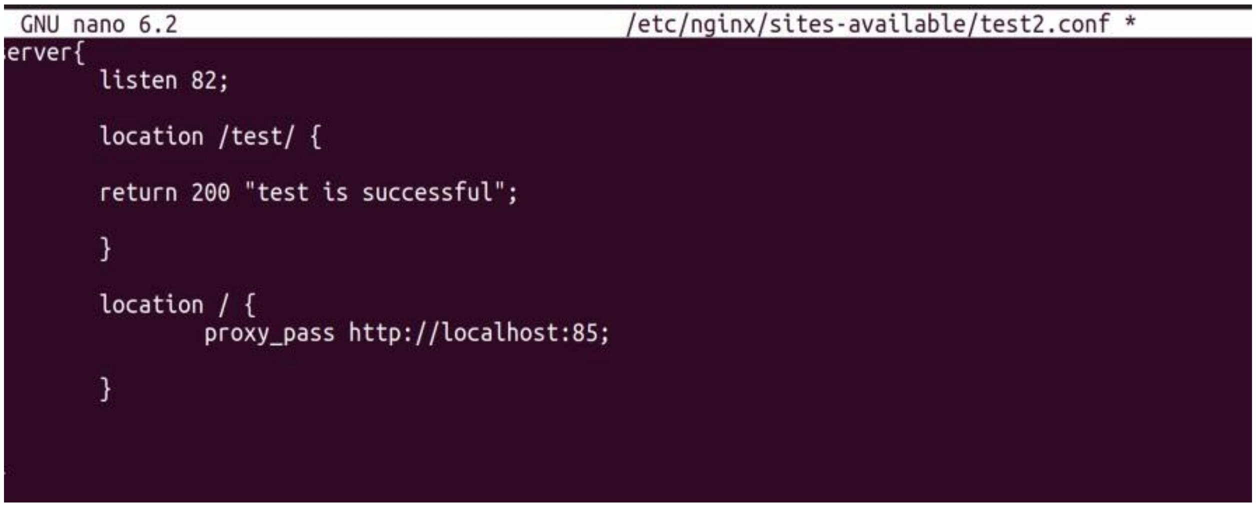
Screenshot:



```
sharumss@sharumss-VirtualBox:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
sharumss@sharumss-VirtualBox:~$ sudo ln -s /etc/nginx/sites-available/test2.conf /etc/nginx/sites-enabled/
sharumss@sharumss-VirtualBox:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
sharumss@sharumss-VirtualBox:~$ sudo systemctl restart nginx
sharumss@sharumss-VirtualBox:~$ curl -v  http://localhost:82/test/
*   Trying 127.0.0.1:82...
* Connected to localhost (127.0.0.1) port 82 (#0)
> GET /test/ HTTP/1.1
> Host: localhost:82
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.18.0 (Ubuntu)
< Date: Wed, 27 Sep 2023 07:42:16 GMT
< Content-Type: application/octet-stream
< Content-Length: 18
< Connection: keep-alive
<
* Connection #0 to host localhost left intact
test is successfulsharumss@sharumss-VirtualBox:~$ curl  http://localhost:82/test/
test is successfulsharumss@sharumss-VirtualBox:~$
```

Then, reverse proxy was set up to forward all http traffic from port 82 to port 85.

Screenshot:



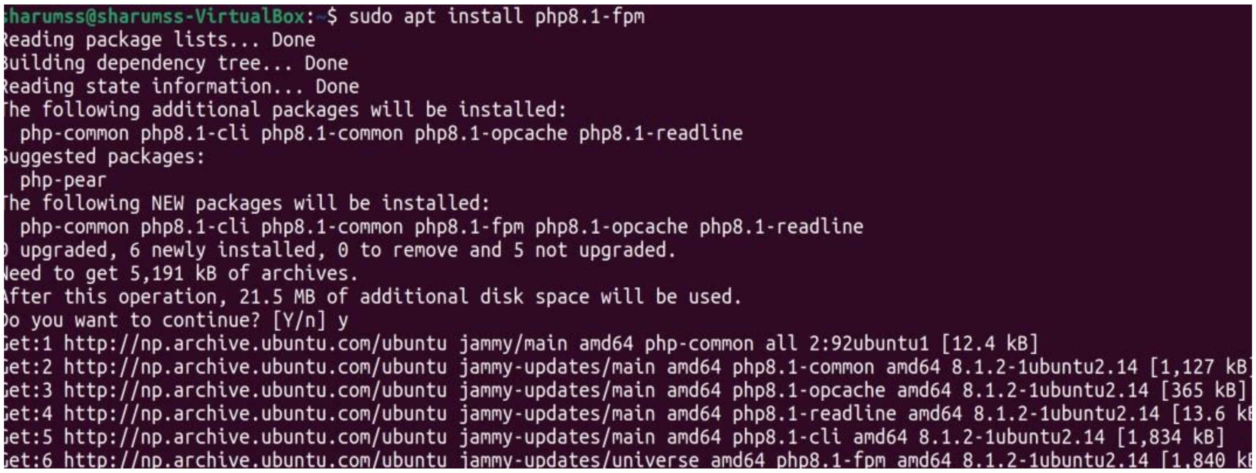
```
 GNU nano 6.2                                  /etc/nginx/sites-available/test2.conf *
erver{
        listen 82;

        location /test/ {

        return 200 "test is successful";

        }

        location / {
                proxy_pass http://localhost:85;

        }
```

6. Installing LEMP Stack without installing mysql and opening info.php on port 80 and printing message info.php

Php8.1-fpm was first installed on ubuntu machine. PHP FPM is a FastCGI process manager for PHP. It manages the PHP processes handling requests from web servers to run PHP applications in production environments, which is much efficient than running PHP as CGI or Apache module because of features like process management, logging, and security.

PHP FPM can pool PHP processes, i.e., it can reuse existing PHP processes to handle requests instead of creating and destroying new PHP processes for each request. It is also highly scalable and handle large numbers of concurrent requests. It also has security features like isolation and sandboxing.

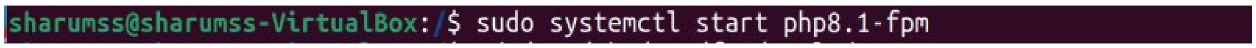
Screenshot:



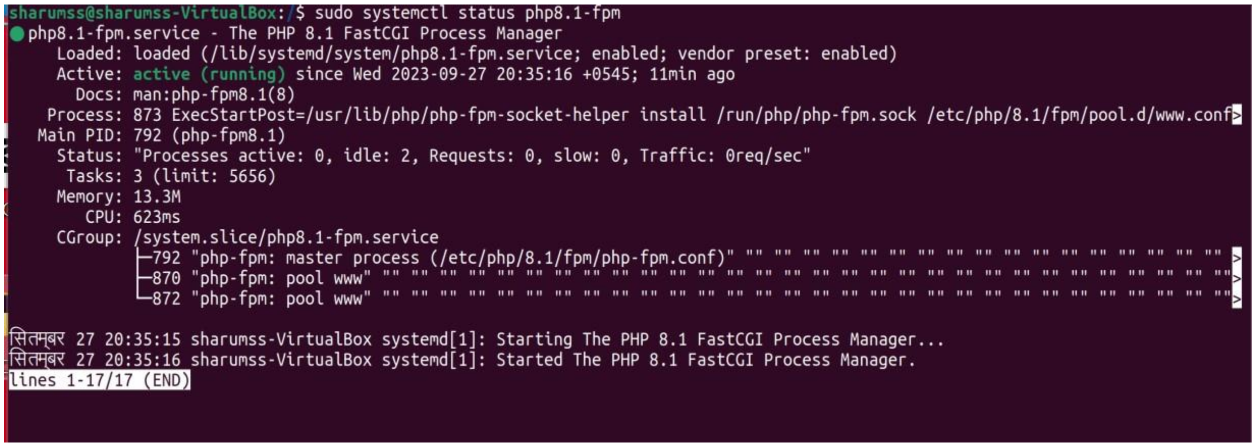
```
harumss@sharumss-VirtualBox:~$ sudo apt install php8.1-fpm
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
he following additional packages will be installed:
 php-common php8.1-cli php8.1-common php8.1-opcache php8.1-readline
uggested packages:
 php-pear
he following NEW packages will be installed:
 php-common php8.1-cli php8.1-common php8.1-fpm php8.1-opcache php8.1-readline
 upgraded, 6 newly installed, 0 to remove and 5 not upgraded.
eed to get 5,191 kB of archives.
fter this operation, 21.5 MB of additional disk space will be used.
o you want to continue? [Y/n] y
et:1 http://np.archive.ubuntu.com/ubuntu jammy/main amd64 php-common all 2:92ubuntu1 [12.4 kB]
et:2 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-common amd64 8.1.2-1ubuntu2.14 [1,127 kB
et:3 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-opcache amd64 8.1.2-1ubuntu2.14 [365 kB]
et:4 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-readline amd64 8.1.2-1ubuntu2.14 [13.6 k
et:5 http://np.archive.ubuntu.com/ubuntu jammy-updates/main amd64 php8.1-cli amd64 8.1.2-1ubuntu2.14 [1,834 kB]
et:6 http://np.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 php8.1-fpm amd64 8.1.2-1ubuntu2.14 [1,840 k
```

Then, it was run, and its status was verified.

Screenshot:



```
sharumss@sharumss-VirtualBox:/$ sudo systemctl start php8.1-fpm
```



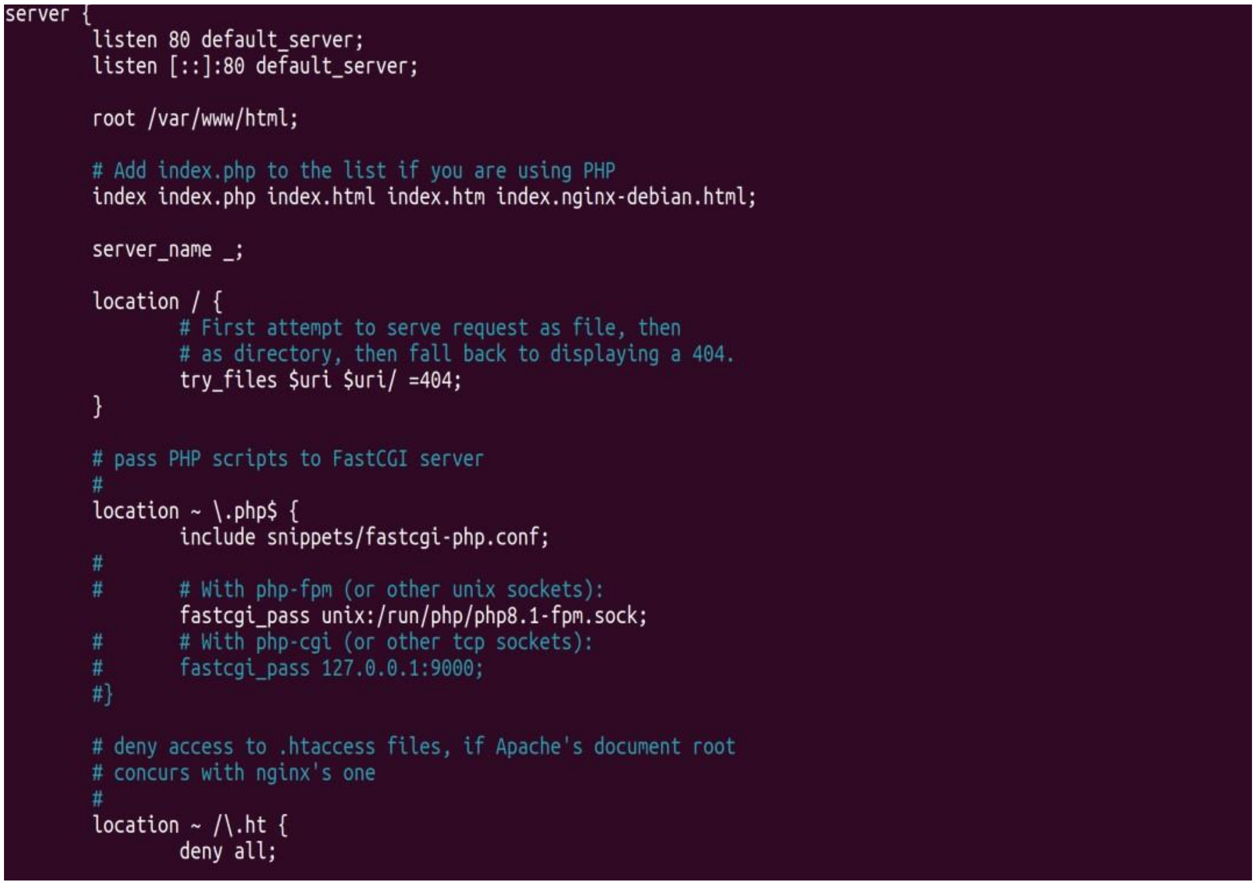
```
sharumss@sharumss-VirtualBox:/$ sudo systemctl status php8.1-fpm
● php8.1-fpm.service - The PHP 8.1 FastCGI Process Manager
     Loaded: loaded (/lib/systemd/system/php8.1-fpm.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2023-09-27 20:35:16 +0545; 11min ago
       Docs: man:php-fpm8.1(8)
    Process: 873 ExecStartPost=/usr/lib/php/php-fpm-socket-helper install /run/php/php-fpm.sock /etc/php/8.1/fpm/pool.d/www.conf>
   Main PID: 792 (php-fpm8.1)
     Status: "Processes active: 0, idle: 2, Requests: 0, slow: 0, Traffic: 0req/sec"
      Tasks: 3 (limit: 5656)
     Memory: 13.3M
        CPU: 623ms
     CGroup: /system.slice/php8.1-fpm.service
             ├─792 "php-fpm: master process (/etc/php/8.1/fpm/php-fpm.conf)" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" >
             ├─870 "php-fpm: pool www" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "">
             └─872 "php-fpm: pool www" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "">

सितम्बर 27 20:35:15 sharumss-VirtualBox systemd[1]: Starting The PHP 8.1 FastCGI Process Manager...
सितम्बर 27 20:35:16 sharumss-VirtualBox systemd[1]: Started The PHP 8.1 FastCGI Process Manager.
lines 1-17/17 (END)
```

Then, the configuration was edited as below.

The server block defines a virtual server, specifying, the server listens on port 80 and the index.php will served upon request.

Location ~\.php$ block specifies that all requests for files with .php extension shall be forwarded to PHP FPM. The ‘include snippets/fastcgi-php.conf;’ directive includes the default FastCGI configuration for PHP. The ‘fastcgi_pass unix:/run/php/php8.1-fpm.sock;’ directive specifies the path to the PHP FPM socket file.

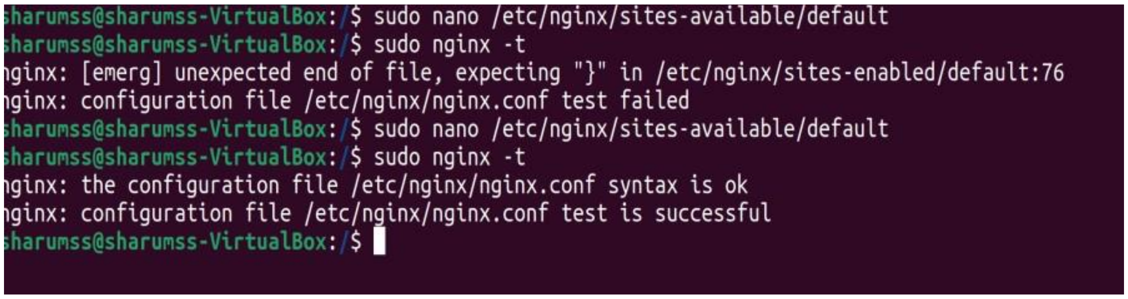
Screenshot:



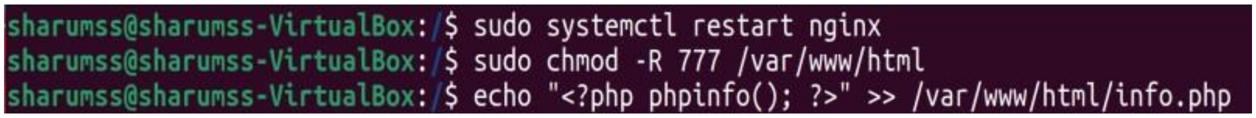
```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.php index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }

        # pass PHP scripts to FastCGI server
        #
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/run/php/php8.1-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        location ~ /\.ht {
                deny all;
```

The configuration was then tested, and service was restarted, and info.php was created.

Screenshot:



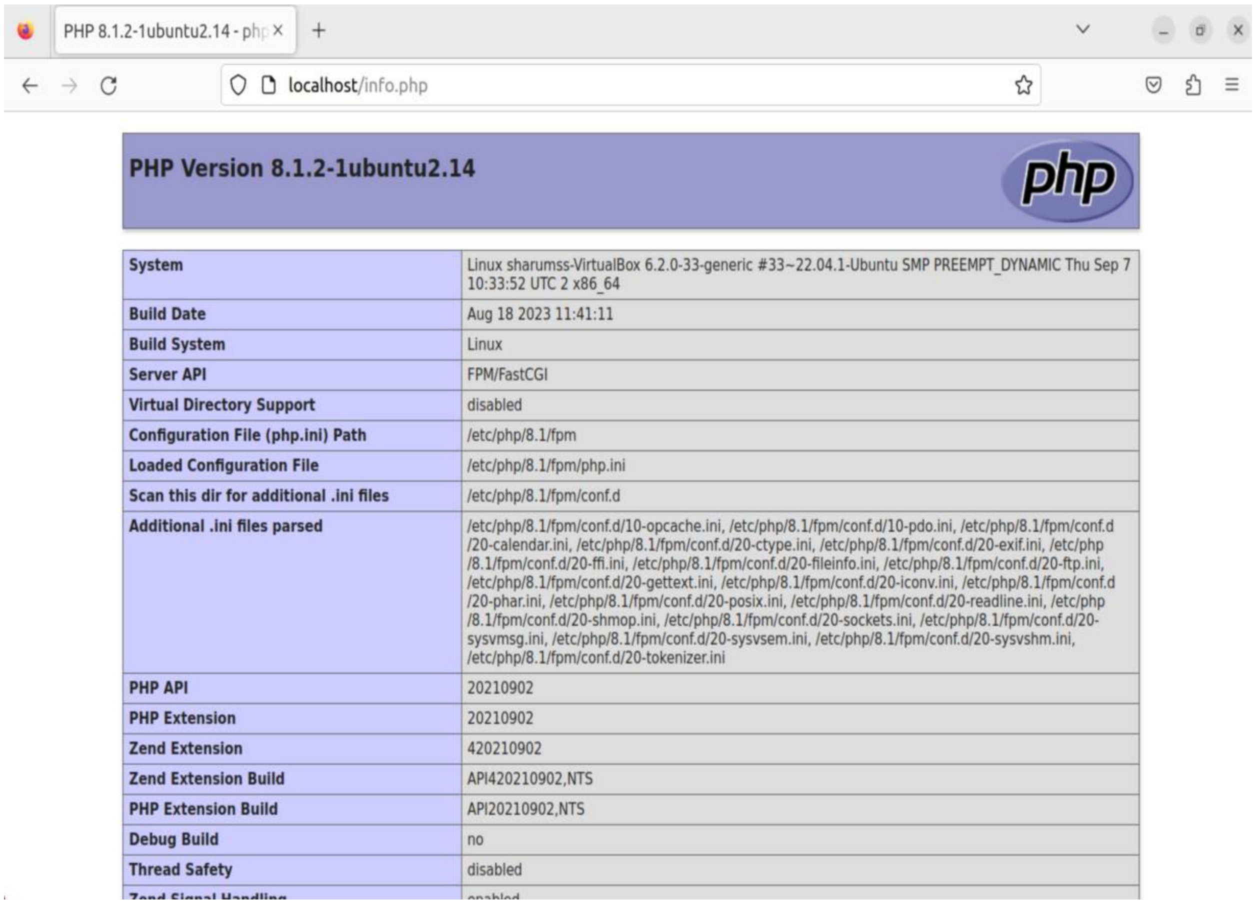
```
sharumss@sharumss-VirtualBox:/$ sudo nano /etc/nginx/sites-available/default
sharumss@sharumss-VirtualBox:/$ sudo nginx -t
nginx: [emerg] unexpected end of file, expecting "}" in /etc/nginx/sites-enabled/default:76
nginx: configuration file /etc/nginx/nginx.conf test failed
sharumss@sharumss-VirtualBox:/$ sudo nano /etc/nginx/sites-available/default
sharumss@sharumss-VirtualBox:/$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
sharumss@sharumss-VirtualBox:/$
```



```
sharumss@sharumss-VirtualBox:/$ sudo systemctl restart nginx
sharumss@sharumss-VirtualBox:/$ sudo chmod -R 777 /var/www/html
sharumss@sharumss-VirtualBox:/$ echo "<?php phpinfo(); ?>" >> /var/www/html/info.php
```

Upon navigating, http://localhost/info.php , following was diaplayed.

Screenshot:



PHP Version 8.1.2-1ubuntu2.14

System	Linux sharumss-VirtualBox 6.2.0-33-generic #33~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Sep 7 10:33:52 UTC 2 x86_64
Build Date	Aug 18 2023 11:41:11
Build System	Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/fpm
Loaded Configuration File	/etc/php/8.1/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/fpm/conf.d
Additional .ini files parsed	/etc/php/8.1/fpm/conf.d/10-opcache.ini, /etc/php/8.1/fpm/conf.d/10-pdo.ini, /etc/php/8.1/fpm/conf.d/20-calendar.ini, /etc/php/8.1/fpm/conf.d/20-ctype.ini, /etc/php/8.1/fpm/conf.d/20-exif.ini, /etc/php/8.1/fpm/conf.d/20-ffi.ini, /etc/php/8.1/fpm/conf.d/20-fileinfo.ini, /etc/php/8.1/fpm/conf.d/20-ftp.ini, /etc/php/8.1/fpm/conf.d/20-gettext.ini, /etc/php/8.1/fpm/conf.d/20-iconv.ini, /etc/php/8.1/fpm/conf.d/20-phar.ini, /etc/php/8.1/fpm/conf.d/20-posix.ini, /etc/php/8.1/fpm/conf.d/20-readline.ini, /etc/php/8.1/fpm/conf.d/20-shmop.ini, /etc/php/8.1/fpm/conf.d/20-sockets.ini, /etc/php/8.1/fpm/conf.d/20-sysvmsg.ini, /etc/php/8.1/fpm/conf.d/20-sysvsem.ini, /etc/php/8.1/fpm/conf.d/20-sysvshm.ini, /etc/php/8.1/fpm/conf.d/20-tokenizer.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no
Thread Safety	disabled

7. Key Concepts

- Differences between Nginx and Apache

Nginx	Apache
Reverse proxy, load balancer, mail proxy, HTTP cache	Web server, mail server, FTP server
Highly scalable and performant, especially for serving static content	Less scalable and performant than Nginx, but more flexible
Supports HTTP/2, reverse proxying, load balancing, mail proxying, and HTTP caching	Supports a wide range of features, including CGI, Perl, PHP, Python, and Ruby
Relatively simple to configure	More complex to configure than Nginx

- **nginx -s reload vs. systemctl restart nginx.**

- The nginx -s reload command reloads the Nginx configuration without restarting the Nginx process. This means that any changes to the Nginx configuration will be picked up without interrupting service.

- The systemctl restart nginx command restarts the Nginx process. This means that all connections to Nginx will be closed, and Nginx will be restarted from scratch. This can be used to restart Nginx if it is not working properly or if you need to make changes to the Nginx configuration that require Nginx to be restarted.

- What is TLS/SSL?

- TLS stands for Transport Layer Security. SSL stands for Secure Sockets Layer. TLS and SSL are cryptographic protocols that are used to provide secure communication over a computer network.

- TLS and SSL are used to protect a variety of types of traffic, including web traffic, email traffic, and file transfer traffic. TLS and SSL are also used to protect traffic between different applications, such as between a web browser and a web server.

- Differences

  TLS is more secure than SSL and it supports several new features, such as support for new cryptographic algorithms and support for elliptic curve cryptography. SSL is no longer considered to be secure, and it is not recommended for use. If you are using a website that uses SSL, you should upgrade to TLS.

- Proxy_pass
  The proxy_pass directive in Nginx is used to forward requests to another server. This can be used to load balance traffic across multiple servers or to improve performance by serving static content from a cache server.

- Fastcgipass
  The fastcgipass directive in Nginx is used to forward requests to a FastCGI application server. FastCGI is a protocol that is used to improve the performance of dynamic web applications.