1. Install nginx and host a simple index.html with message “hello nginx”.

Nginx is installed in our VM using command:

Sudo apt-get install nginx

After nginx is installed, Nginx services are to be allowed in the firewall.

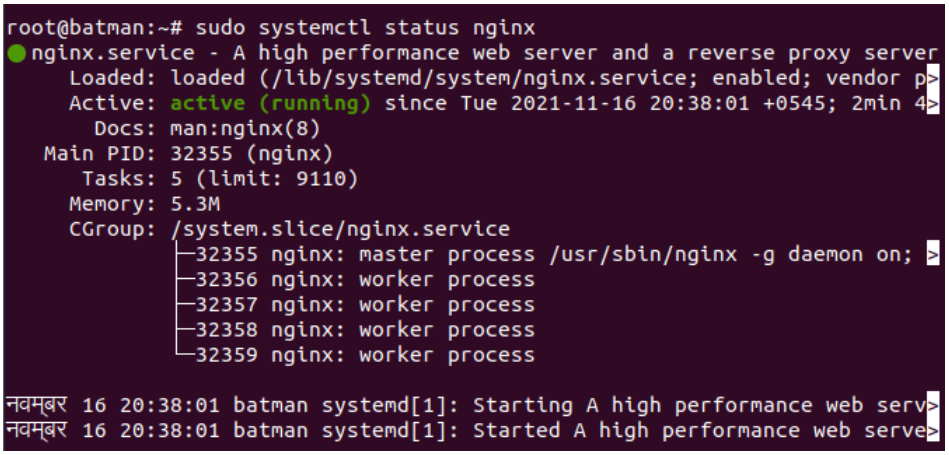
Using command: ***Sudo ufw app list*** we can list the application profile, and following command to allow the services in firewall specifically.

Ufw allow ‘Nginx Full’

Ufw allow ‘Nginx HTTP’

To check Nginx status:

Sudo systemctl status nginx



```
root@batman:~# sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor p>
     Active: active (running) since Tue 2021-11-16 20:38:01 +0545; 2min 4>
       Docs: man:nginx(8)
   Main PID: 32355 (nginx)
      Tasks: 5 (limit: 9110)
     Memory: 5.3M
     CGroup: /system.slice/nginx.service
             ├─32355 nginx: master process /usr/sbin/nginx -g daemon on; >
             ├─32356 nginx: worker process
             ├─32357 nginx: worker process
             ├─32358 nginx: worker process
             └─32359 nginx: worker process

नवम्बर 16 20:38:01 batman systemd[1]: Starting A high performance web serv>
नवम्बर 16 20:38:01 batman systemd[1]: Started A high performance web serve>
```

We can see that nginx is active.

Now, we copy the directory which contains test application which is ready to be deployed in the /var/www/html file using the command:

Cp NginxTask/ /var/www/html/

My directory was NginxTask which contains a simple index.html file.

Now my index.html file’s absolute path is /var/www/html/NginxTask/index.html

We use this absolute path quite often.

And that directory’s ownership was changed to current non root user:

Sudo chown -R $USER:$USER /var/www/html/NginxTask/

Now we create a configuration file for the application which is to be deployed.

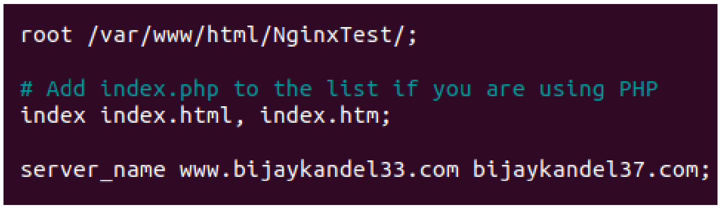
Inside the directory /etc/nginx/sites-available/, a default configuration file is present. We copy this file to a new file called NginxTask.conf inside the same directory using command:

Cp default NginxTask.conf

To edit the file:

Nano NginxTask.conf

Inside this file some configurations are edited: Inside server block, the server name was enlisted, also the absolute path of the index.html file was mentioned with root.



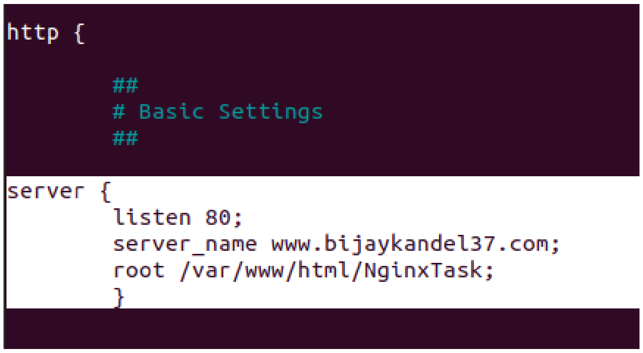
```
root /var/www/html/NginxTest/;

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

server_name www.bijaykandel33.com bijaykandel37.com;
```

And the file was saved and exited.

Now nginx.conf file was edited using command:

Nano /etc/nginx/nginx.conf

Inside http block, a Server block is added with some configurations as shown:



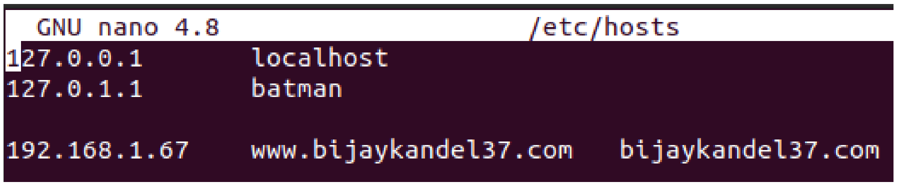
```
http {

        ##
        # Basic Settings
        ##

server {
        listen 80;
        server_name www.bijaykandel37.com;
        root /var/www/html/NginxTask;
        }
```

Symlink was created with the NginxTask.conf located in sites-available directory and that of sites-enabled directory using command:

In -s /etc/nginx/sites-available/NginxTask.conf /etc/nginx/sites-enabled/

Now host ip was configured using command:

Sudo nano /etc/hosts and listing my IP and domainname as:



```
  GNU nano 4.8                          /etc/hosts
127.0.0.1       localhost
127.0.1.1       batman

192.168.1.67    www.bijaykandel37.com   bijaykandel37.com
```

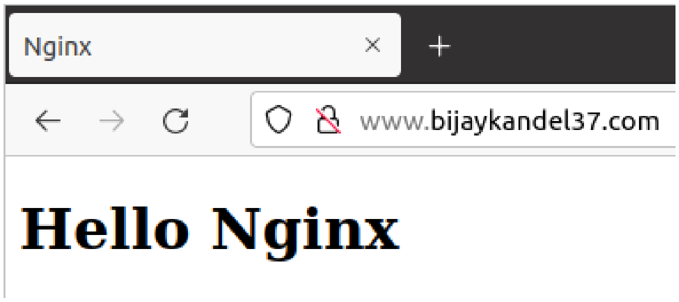
To test the system configurations,

nginx -t

Since no errors were generated, nginx was restarted using command:

Sudo systemctl restart nginx

And when we go to the browser and enter the domain name which is provided before (www.bijaykandel37.com) in my case and we can get this result:



Hello Nginx

2. What are nginx header security and its uses. Also implement in the test.conf file.

The HTTP headers are the additional information passed between the client (browsers) and web servers. The security HTTP headers are the response HTTP headers that the server can add in order to increase the security of HTTP exchange. Each security header serves its own purpose. Here are a few http security headers, with their uses:

- **HTTP Strict Transport Security (HSTS)** - It is a way to deal with potential vulnerability by instructing the browser that a domain can only be accessed using HTTPS. It can be implemented as:

  add_header Strict-Transport-Security "max-age=1000; includeSubDomains" always;

  The *max-age* is the time in seconds for which the information is cached. The *includeSubDomains* option tells that this rule is applied to all subdomains of the main domain.

- **X-Frame-Options** - It is used to defend our website from clickjacking attacks by disabling iframes on our site. There are three ways to configure X-Frame-Options:

  DENY **:** This will disable iframe features completely.

  SAMEORIGIN **:** iframe can be used only by someone of the same origin.

  ALLOW-FROM **:** This will allow pages to be put in iframes only from specific URLs.

  It can be implemented as:

  Add_header X-Frame-Options “DENY” always;

- **X-XSS-Protection** - It is used to defend against Cross-Site Scripting attacks. This header stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. It can be implemented as: add_header X-XSS-Protection "1; mode=block";

  X-XSS-Protection: 0 : It will disable the filter entirely.

  X-XSS-Protection: 1 : It enables the filter but only sanitizes potentially malicious scripts.

X-XSS-Protection: 1; mode=block : It enables the filter and completely blocks the page.

- **X-Content-Type-Options** - It is used to prevent web browsers from sniffing a response away from the declared Content-Type. It can be implemented as:

  ```
  add_header X-Content-Type-Options nosniff;
  ```

- **Content-Security-Policy** - It is an improved version of the X-XSS-Protection header and provides an additional layer of security which is used to prevent XSS and data injection attacks. It is used to filter the file types that are accepted. It can be implemented as:

  ```
  add_header Content-Security-Policy "default-src 'self'; font-src *;img-src * data:; script-src *; style-src *";
  ```

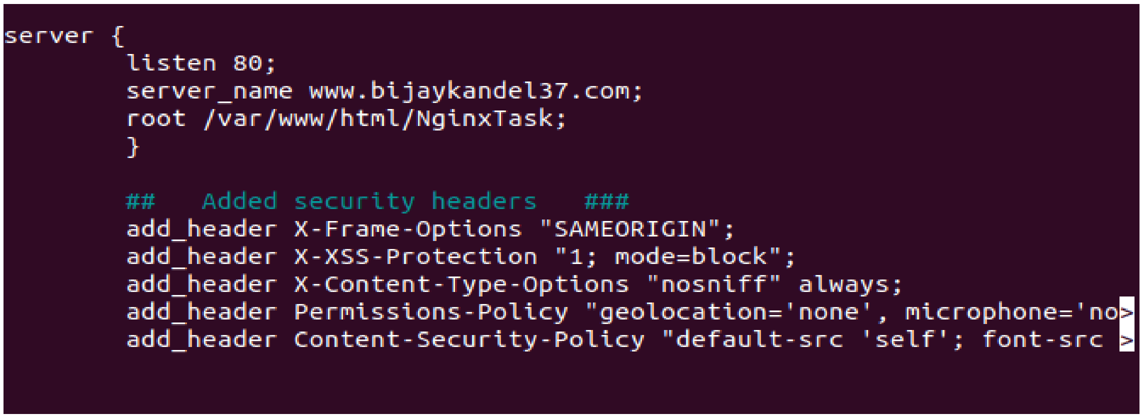
- **Permissions-Policy** - It allows a site to control which APIs or features can be used in the browser. Like controls of individual hardwares like mic and webcam. It can be implemented as:

  ```
  add_header Permissions-Policy "geolocation=(), midi=(), sync-xhr=(), microphone=(), camera=(), fullscreen=(self), payment=()";
  ```

- **X-Permitted-Cross-Domain-Policies** - used to allow PDF and Flash documents for cross-domain requests.
- **Others** : There are many more security headers like Public Key Pinning Extension for HTTP (HPKP), Referrer policy and so on.

In our NginxTask.conf file, we applied following security headers:

Nano /etc/nginx/nginx.conf



```
server {
        listen 80;
        server_name www.bijaykandel37.com;
        root /var/www/html/NginxTask;
        }

        ##   Added security headers   ###
        add_header X-Frame-Options "SAMEORIGIN";
        add_header X-XSS-Protection "1; mode=block";
        add_header X-Content-Type-Options "nosniff" always;
        add_header Permissions-Policy "geolocation='none', microphone='no>
        add_header Content-Security-Policy "default-src 'self'; font-src >
```

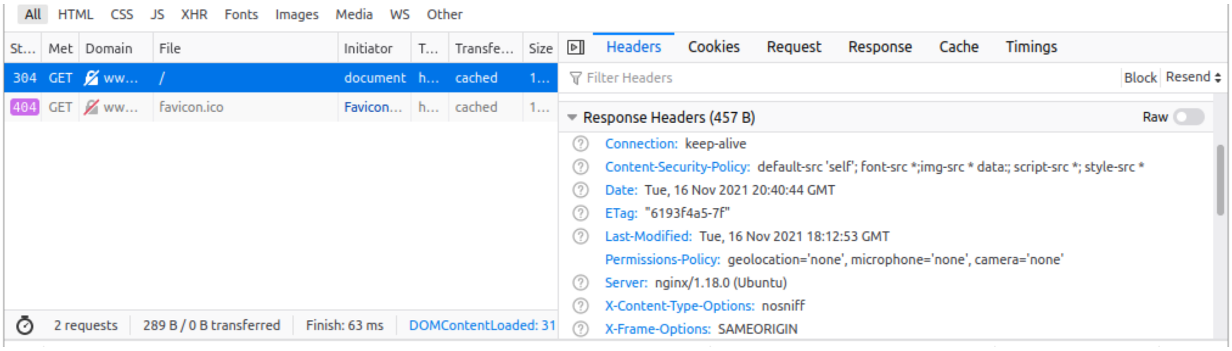
To test the updated configurations:

Nginx -t

And to restart nginx:

Systemctl restart nginx

Now when the browser is opened with the same link, we can see the activated security headers by inspecting.



St…	Met	Domain	File	Initiator	T…	Transfe…	Size
304	GET	ww…	/	document	h…	cached	1…
404	GET	ww…	favicon.ico	Favicon…	h…	cached	1…

2 requests | 289 B / 0 B transferred | Finish: 63 ms | DOMContentLoaded: 31

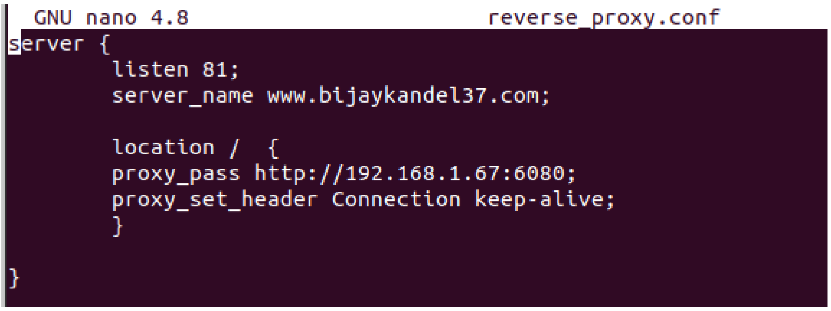
Response Headers (457 B)

```
Connection: keep-alive
Content-Security-Policy: default-src 'self'; font-src *;img-src * data:; script-src *; style-src *
Date: Tue, 16 Nov 2021 20:40:44 GMT
ETag: "6193f4a5-7f"
Last-Modified: Tue, 16 Nov 2021 18:12:53 GMT
Permissions-Policy: geolocation='none', microphone='none', camera='none'
Server: nginx/1.18.0 (Ubuntu)
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
```

3. Nginx Reverse proxy all http requests to nodes js api.

Firstly, a reverse_proxy.conf file is created in /sites-available/ directory with following content:

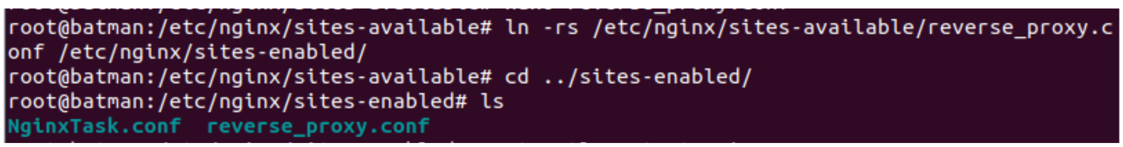
Nano reverse_proxy.conf



```
  GNU nano 4.8                          reverse_proxy.conf
server {
        listen 81;
        server_name www.bijaykandel37.com;

        location /  {
        proxy_pass http://192.168.1.67:6080;
        proxy_set_header Connection keep-alive;
        }

}
```

Then a symlink is created using command:

Ln -rs /etc/nginx/sites-available/reverse_proxy.conf /etc/nginx/sites-enabled/



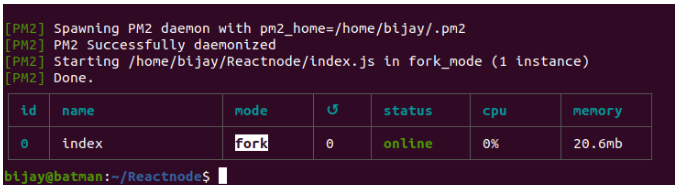
```
root@batman:/etc/nginx/sites-available# ln -rs /etc/nginx/sites-available/reverse_proxy.c
onf /etc/nginx/sites-enabled/
root@batman:/etc/nginx/sites-available# cd ../sites-enabled/
root@batman:/etc/nginx/sites-enabled# ls
NginxTask.conf  reverse_proxy.conf
```

Now the config is tested using command ***nginx -t***, and restarted nginx using:

Systemctl restart nginx

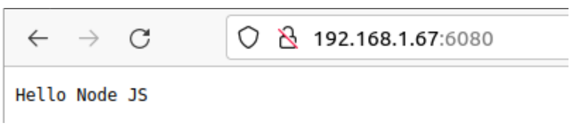
Now, node app is started using pm2, with command:

Pm2 start index.js

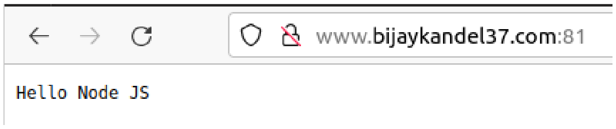


```
[PM2] Spawning PM2 daemon with pm2_home=/home/bijay/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/bijay/Reactnode/index.js in fork_mode (1 instance)
[PM2] Done.
┌────┬────────────────────┬──────────┬──────┬───────────┬──────────┬──────────┐
│ id │ name               │ mode     │ ↺    │ status    │ cpu      │ memory   │
├────┼────────────────────┼──────────┼──────┼───────────┼──────────┼──────────┤
│ 0  │ index              │ fork     │ 0    │ online    │ 0%       │ 20.6mb   │
└────┴────────────────────┴──────────┴──────┴───────────┴──────────┴──────────┘
bijay@batman:~/Reactnode$
```

We can verify by running the app in browser, and if we enter the domain of nginx server with port 81, we can see the same node app:



Hello Node JS



Hello Node JS

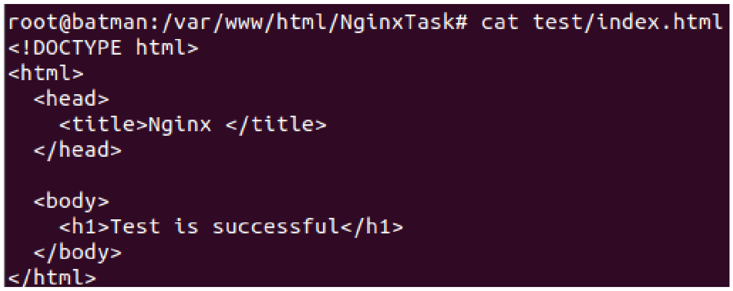
4. Create a test2.conf and listen on port 82 and to “ location /test/” with the message “ test is successful”.

First of all, I made a subdirectory ‘test’ inside the previous directory NginxTask,

The sample html page was copied to test from NginxTask using command:

Cp index.html test/

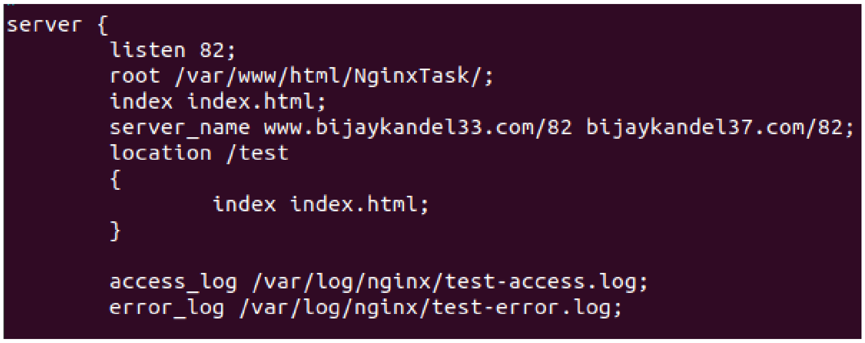
The message in index.html inside test/ was edited to ‘Test Was Successful’.



```
root@batman:/var/www/html/NginxTask# cat test/index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Nginx </title>
  </head>

  <body>
    <h1>Test is successful</h1>
  </body>
</html>
```

Now Inside the /sites-available/ directory, a test2.conf file was made with following content:

Nano test2.conf



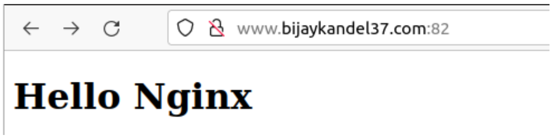
```
server {
        listen 82;
        root /var/www/html/NginxTask/;
        index index.html;
        server_name www.bijaykandel33.com/82 bijaykandel37.com/82;
        location /test
        {
                index index.html;
        }

        access_log /var/log/nginx/test-access.log;
        error_log /var/log/nginx/test-error.log;
```

And a symlink was created using command:

Ln -rs /etc/nginx/sites-available/test2.conf ../sites-enabled/

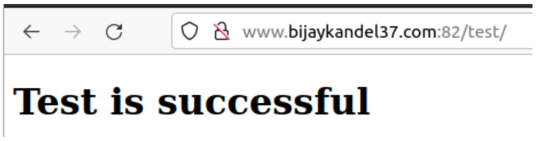
nginx -t to test and

systemctl restart nginx to restart the nginx and we can see the result in browser:



www.bijaykandel37.com:82

Hello Nginx



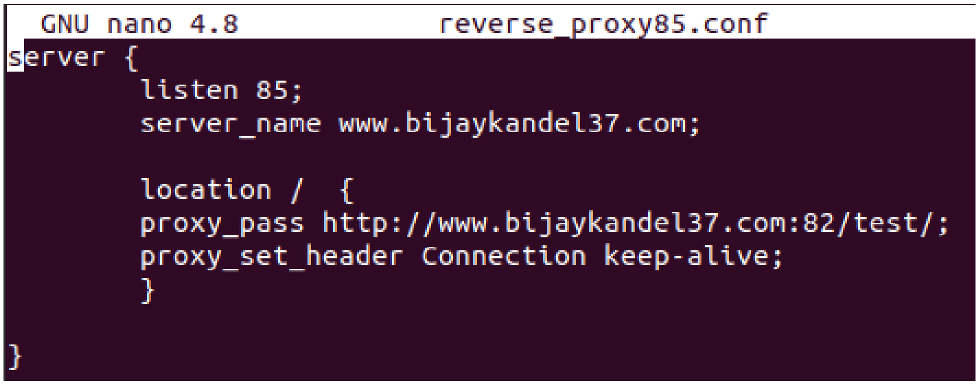
www.bijaykandel37.com:82/test/

Test is successful

5. Reverse proxy all http traffic of port 82 to port 85.

For this, we created a new reverse_proxy85.conf file inside /sites-available/ directory and entered this content:

Nano reverse_proxy85.conf



```
  GNU nano 4.8             reverse_proxy85.conf
server {
        listen 85;
        server_name www.bijaykandel37.com;

        location /  {
        proxy_pass http://www.bijaykandel37.com:82/test/;
        proxy_set_header Connection keep-alive;
        }

}
```

And a symlink was created using command:

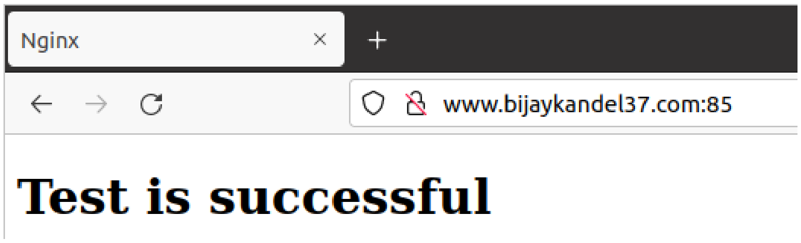
Ln -rs /etc/nginx/sites-available/reverse_proxy85.conf ../sites-enabled/

Now to test and restart nginx:

Nginx -t

Systemctl restart nginx

And in the browser we can see the results:



Test is successful

6. Install LEMP stack (avoid installing mysql) and open info.php on port 80 and print message info.php.

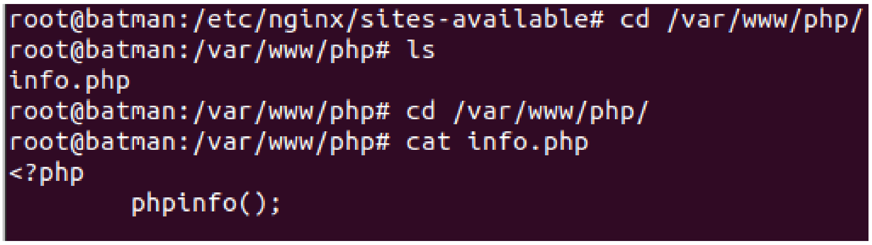
First php-fpm is installed in our system:

Sudo apt install php-fpm

We already have a Linux system and Nginx (Pronounced as EngineX), since we have to avoid installing mysql, we now have a LEMP stack.

Now, a file, info.php is made in directory /var/www/php/ with following content:

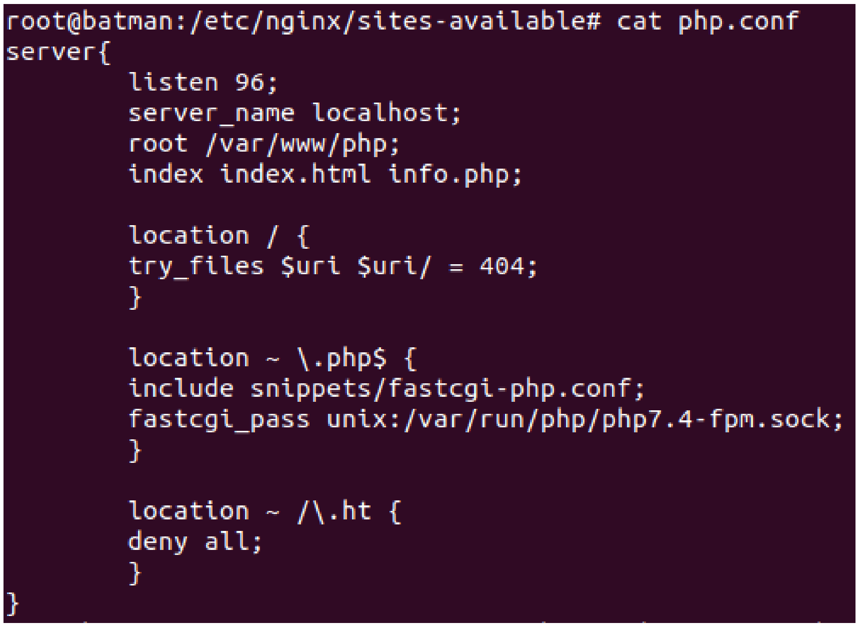
Nano /var/www/php/info.php



```
root@batman:/etc/nginx/sites-available# cd /var/www/php/
root@batman:/var/www/php# ls
info.php
root@batman:/var/www/php# cd /var/www/php/
root@batman:/var/www/php# cat info.php
<?php
        phpinfo();
```

And a file php.conf is made in directory /etc/nginx/sites-available/ with following content:

Nano /etc/nginx/sites-available/php.conf



```
root@batman:/etc/nginx/sites-available# cat php.conf
server{
        listen 96;
        server_name localhost;
        root /var/www/php;
        index index.html info.php;

        location / {
        try_files $uri $uri/ = 404;
        }

        location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        }

        location ~ /\.ht {
        deny all;
        }
}
```

Here port number 96 is used to avoid collision since port 80 is used by the previous task. If we wish to use port 80, we can simply remove the previous conf file or change its port to any port other than 80.

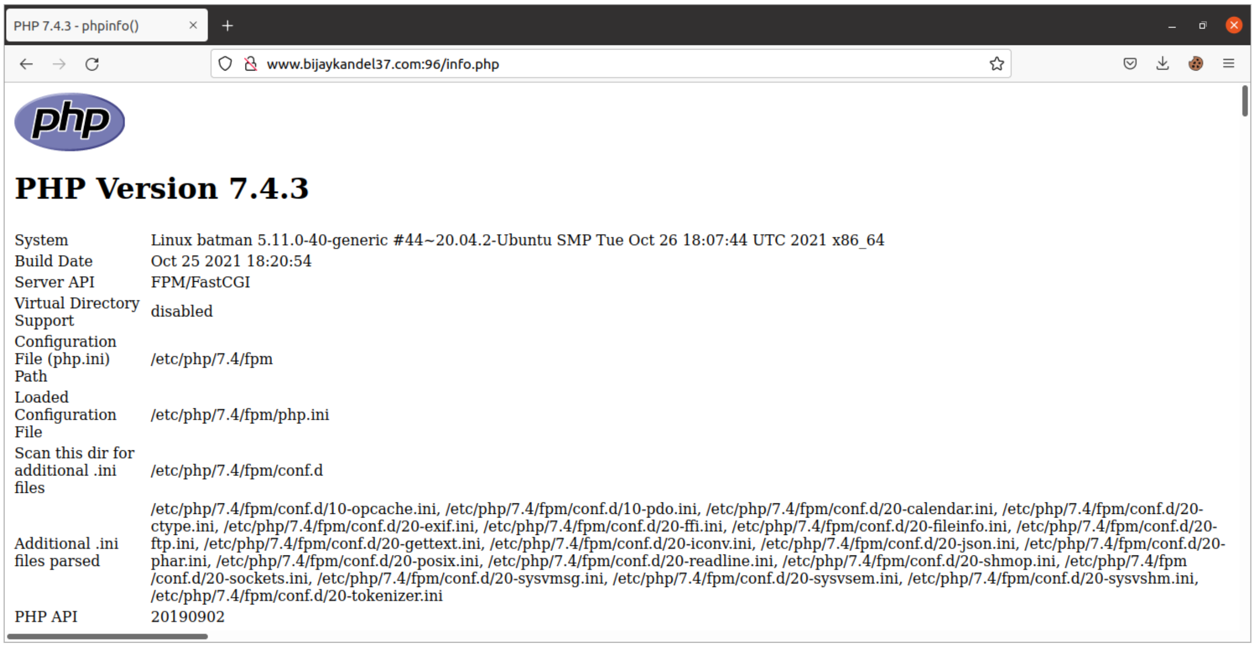
A symlink is created using following command:

```
Ln -rs /etc/nginx/sites-available/php.conf /etc/nginx/sites-enabled/
```

Now, to test and restart:

```
nginx -t
Systemctl restart nginx
```

Now when we access the browser with url we can get the result:



PHP Version 7.4.3

System	Linux batman 5.11.0-40-generic #44~20.04.2-Ubuntu SMP Tue Oct 26 18:07:44 UTC 2021 x86_64
Build Date	Oct 25 2021 18:20:54
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/fpm
Loaded Configuration File	/etc/php/7.4/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/fpm/conf.d
Additional .ini files parsed	/etc/php/7.4/fpm/conf.d/10-opcache.ini, /etc/php/7.4/fpm/conf.d/10-pdo.ini, /etc/php/7.4/fpm/conf.d/20-calendar.ini, /etc/php/7.4/fpm/conf.d/20-ctype.ini, /etc/php/7.4/fpm/conf.d/20-exif.ini, /etc/php/7.4/fpm/conf.d/20-ffi.ini, /etc/php/7.4/fpm/conf.d/20-fileinfo.ini, /etc/php/7.4/fpm/conf.d/20-ftp.ini, /etc/php/7.4/fpm/conf.d/20-gettext.ini, /etc/php/7.4/fpm/conf.d/20-iconv.ini, /etc/php/7.4/fpm/conf.d/20-json.ini, /etc/php/7.4/fpm/conf.d/20-phar.ini, /etc/php/7.4/fpm/conf.d/20-posix.ini, /etc/php/7.4/fpm/conf.d/20-readline.ini, /etc/php/7.4/fpm/conf.d/20-shmop.ini, /etc/php/7.4/fpm/conf.d/20-sockets.ini, /etc/php/7.4/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.4/fpm/conf.d/20-sysvsem.ini, /etc/php/7.4/fpm/conf.d/20-sysvshm.ini, /etc/php/7.4/fpm/conf.d/20-tokenizer.ini
PHP API	20190902