

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

Install nginx :

```
sudo apt install nginx
```

Make a directory test in /var/www/ and creating **index.html** in that directory

```
mkdir /var/www/test
```

```
vi /var/www/test/index.html
```

And paste the following sample html file

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
  <title>DevOps Assignment</title>
```

```
  <LINK href="styles.css" rel="stylesheet" type="text/css">
```

```
</head>
```

```
<body>
```

```
<h1>
```

```
  Hello NGINX !!
```

```
</h1>
```

```
<p>
```

```
From DevOps Internship !!
```

```
</p>
```

```
</body>
```

```
</html>
```

Save and exit

```
root@bibek-lfTechnology:~# ls /var/www/test/  
index.html  
root@bibek-lfTechnology:~# |
```

```
<!DOCTYPE html>  
  
<html>  
<head>  
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>  
  <title>DevOps Assignment</title>  
  <LINK href="styles.css" rel="stylesheet" type="text/css">  
</head>  
  
<body>  
  
<h1>  
  Hello NGINX !!  
</h1>  
<p>  
  From DevOps Internship !!  
</p>  
~  
</body>  
</html>
```

Index.html has the message to display **Hello Nginx**

### Make a test.conf file in sites-available

vi /etc/nginx/sites-available/test.conf

```
server {  
    listen 80;  
    server_name localhost;  
    root /var/www/test;  
    index index.html;  
    access_log /var/log/nginx/test.log;  
    error_log /var/log/nginx/test-error.log;  
}
```

Save and exit

Create soft link of test.conf file from **sites-available** to **sites-enabled**

**ln -s /etc/nginx/sites-available/test.conf /etc/nginx/sites-enabled/**

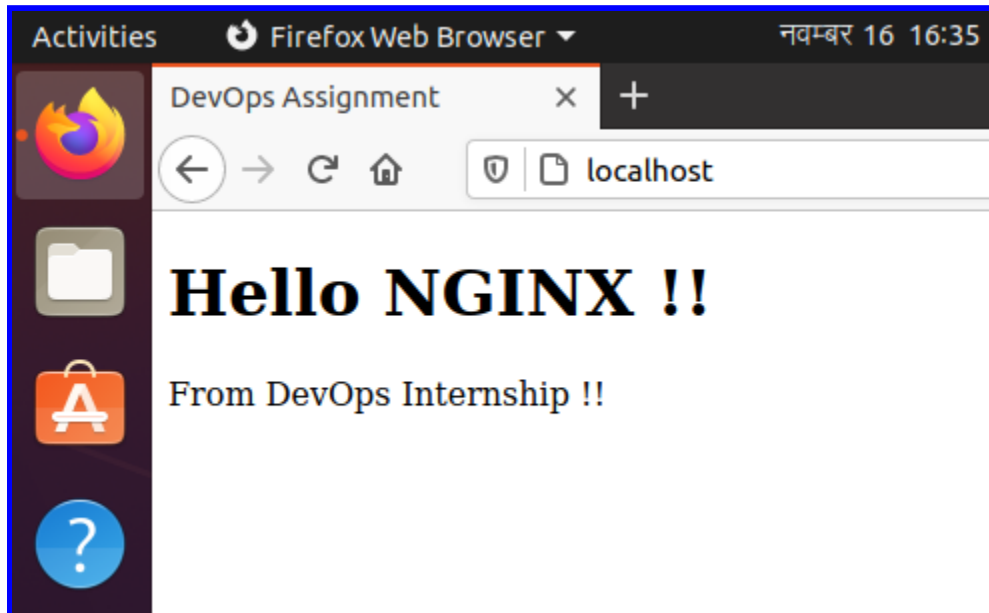
Check for the syntax error and if everything is ok restart the nginx service

**nginx -t**

**systemctl restart nginx**

```
root@bibek-lfTechnology:~# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@bibek-lfTechnology:~# systemctl restart nginx
```

If we browse local host, we can see the web page with required message



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

### Security Headers :

HTTP header fields are a list of linefeed-separated HTTP data being sent and received by both the client program and server on every HTTP request.

They define how information sent/received through the connection are encoded (as in Accept-Encoding),

- the session verification and identification of the client (as in browser cookies, IP address, user-agent) or their anonymity thereof (VPN or proxy masking, user-agent spoofing),
- how the server should handle data (as in Do-Not-Track),
- the age of the document being downloaded, amongst others.

HTTP security headers are a very important part of website security as they protect you against different types of attacks including, XSS, SQL injection, clickjacking, etc.

Some important security headers are

### 1. HSTS - HTTP Strict Transport Security

This header instructs a user agent to only use HTTPs connections and it also declared by Strict-Transport-Security

This will prevent web browsers from accessing web servers over non-HTTPS connections.

Currently all major web browsers support HTTP strict transport security.

We can implement HSTS in Nginx by adding the following entry in

/etc/nginx/sites-available/test.conf file:

```
add_header Strict-Transport-Security 'max-age=31536000; includeSubDomains; preload';
```

It caches this information for the max-age period (typically 31,536,000 seconds, equal to about 1 year).

The optional includeSubDomains parameter tells the browser that the HSTS policy also applies to all subdomains of the current domain.

## **2. X-Frame-Options**

The X-Frame-Options header can help prevent click-jacking attacks.

This header indicates whether the web page can be rendered in a frame.

X-Frame-Options headers allow you to tell the web browser not to allow embedding of your web pages in a frame.

Can be implemented as:

```
add_header X-Frame-Options "SAMEORIGIN" always;
```

- **SAMEORIGIN** – allow your webpages to be displayed in an iframe on the same website
- **ALLOW-FROM uri** – allow your webpages to be embedded in only specific domains/websites
- **DENY** – do not allow any website to embed your webpages in an iframe

## **3. Content Security Policy**

The Content-Security-Policy header is an improved version of the X-XSS-Protection header and provides an additional layer of security.

It is a very powerful header aimed to prevent XSS and data injection attacks.

The CSP header is a way of whitelisting the things that your site is allowed to run. This includes images, stylesheets, javascript, inline styles, frames.

### **Content-Security-Policy: policy**

Can be implemented as:

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

#### **4. X-XSS-Protection**

X-XSS also known as Cross Site Scripting header is used to defend against Cross-Site Scripting attacks.

Can be implemented as:

```
add_header X-XSS-Protection "1; mode=block" always;
```

- **X-XSS-Protection: 0** : This will disable the filter entirely.
- **X-XSS-Protection: 1** : This will enable the filter but only sanitizes potentially malicious scripts.
- **X-XSS-Protection: 1; mode=block** : This will enable the filter and completely block the page.

#### **5. X-Content-Type-Options**

The x-content-type header is also called "Browser Sniffing Protection" to tell the browser to follow the MIME types indicated in the header.

It is used to prevent web browsers such as Internet Explorer and Google Chrome from sniffing a response away from the declared Content-Type.

Can be implemented as:

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

#### **6. Feature-Policy**

The HTTP Feature-Policy header provides a mechanism to allow and deny the use of browser features in its own frame, and in content within any iframe elements in the document.

Syntax:

**Feature-Policy:** <directive> <allowlist>

Can be implemented as:

```
add_header Feature-Policy " microphone 'none' ; geolocation 'none' " ;
```

Some of Directives are:

Camera, midi, accelerometer, fullscreen,etc

### Allowlist values

- \*: The feature will be allowed in this document, and all nested browsing contexts (iframes) regardless of their origin.
- 'self': The feature will be allowed in this document, and in all nested browsing contexts (iframes) in the same origin. The feature is not allowed in cross-origin documents in nested browsing contexts.
- 'src': (In an iframe allow attribute only) The feature will be allowed in this iframe, as long as the document loaded into it comes from the same origin as the URL in the iframe's src attribute. Note: The 'src' origin is used in the iframe allow attribute only, and is the default allowlist value.
- 'none': The feature is disabled in top-level and nested browsing contexts.
- <origin(s)>: The feature is allowed for specific origins (for example, https://example.com). Origins should be separated by a space.

We add some of these headers into our **test.conf** file to implement those security

```
server {
    listen 80;
    server_name localhost;
    root /var/www/test;
    index index.html;
    access_log /var/log/nginx/test.log;
    error_log /var/log/nginx/test-error.log;

    #header_Security
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Feature-Policy " microphone 'none' ; geolocation 'none' " ;
}
```

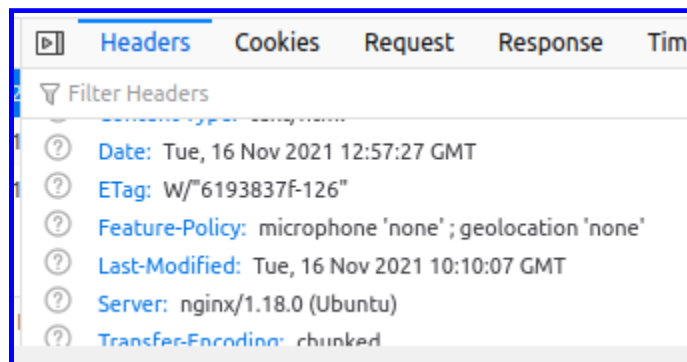
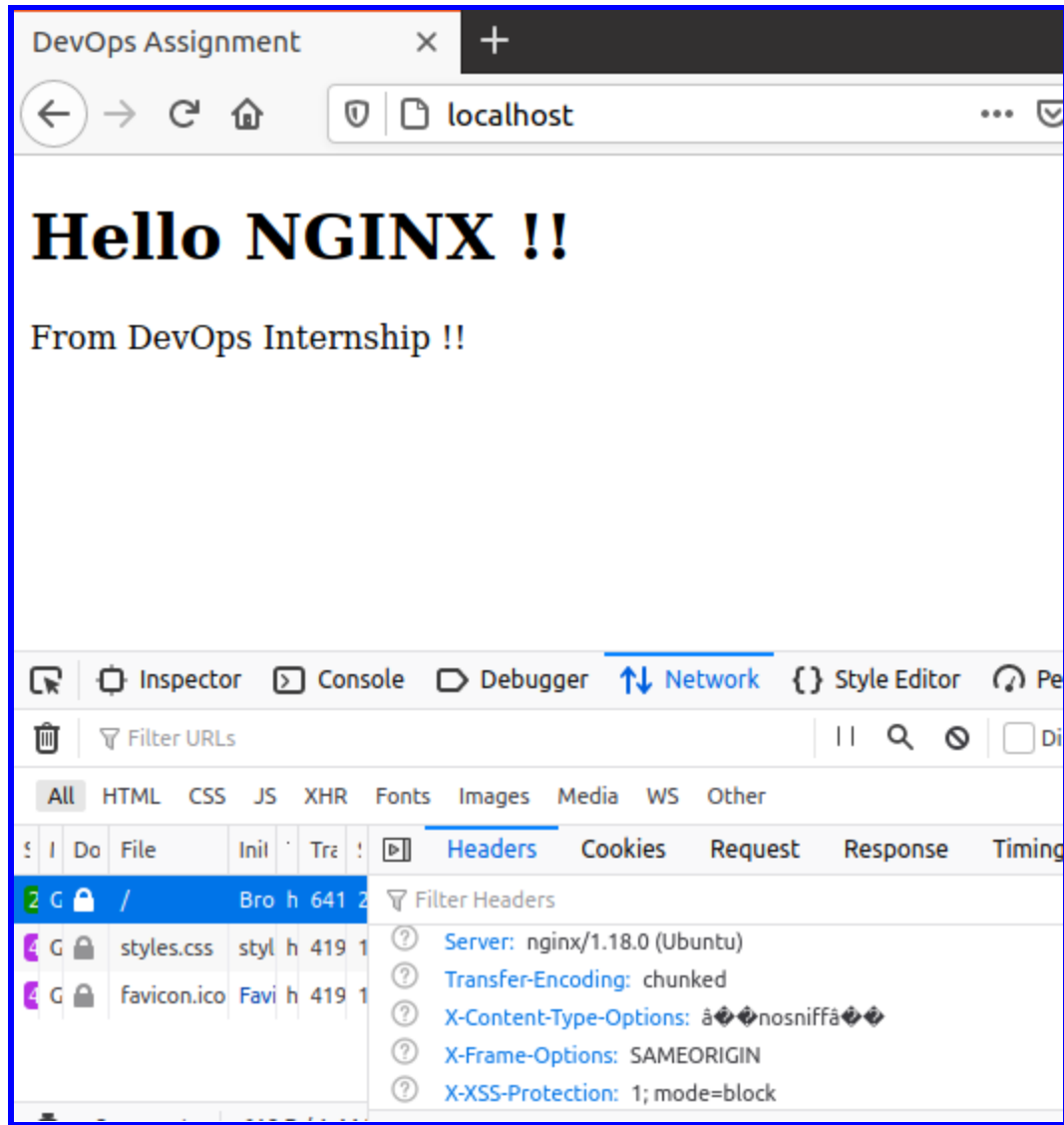
Checked the syntactical error

**nginx -t**

And restart the nginx service

**systemctl restart nginx**

And we see the output in browser with implemented header securities



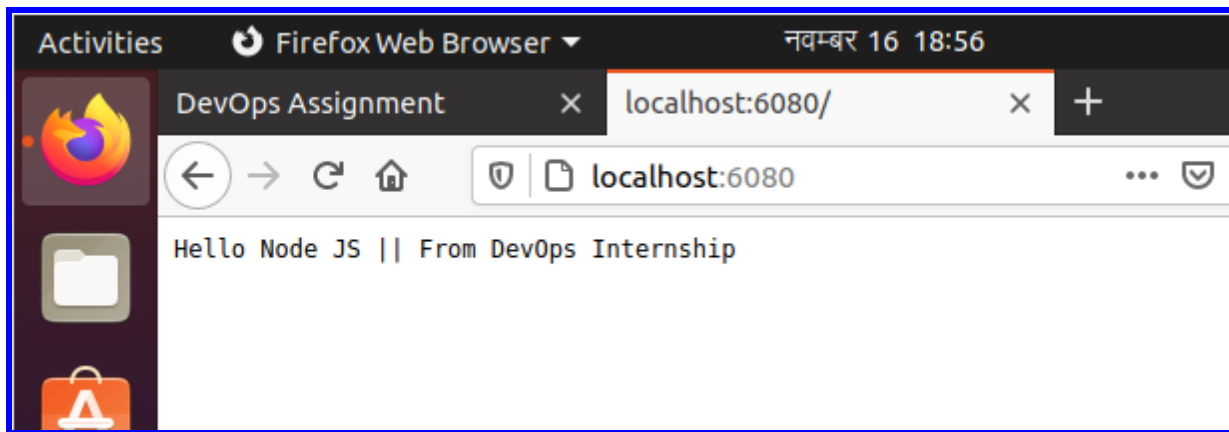
**3. Nginx Reverse proxy all http requests to nodes js api.**



Started the main.js service with pm2 tool

```
root@bibek-lfTechnology:/home/bibek# cd nodejs
root@bibek-lfTechnology:/home/bibek/nodejs# ls
main.js  package.json
root@bibek-lfTechnology:/home/bibek/nodejs# pm2 start main.js
[PM2] Starting /home/bibek/nodejs/main.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	u	status	cpu	memory
0	main	fork	0	online	0%	30.3mb



Now creating reverse\_proxy.conf file in sites-available for reverse proxy

**vi /etc/nginx/sites-available/reverse\_proxy.conf**

```
server {
    listen 81;
    server_name localhost;

    location / {
        proxy_pass http://localhost:6080;
    }
}
```

To create soft-link and enable this reverseproxy

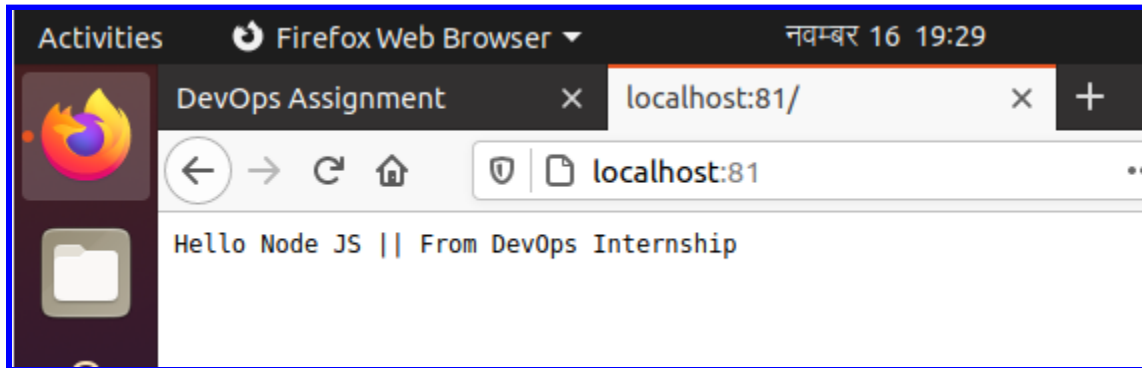
**ln -rs /etc/nginx/sites-available/reverse\_proxy.conf /etc/nginx/sites-enabled/**

```
root@bibek-lfTechnology:~# ls /etc/nginx/sites-enabled/
reverse_proxy.conf  test.conf
root@bibek-lfTechnology:~#
```

To check the syntactical error and restart nginx

```
root@bibek-lfTechnology:/etc/nginx/sites-enabled# ls
reverse_proxy.conf
root@bibek-lfTechnology:/etc/nginx/sites-enabled# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@bibek-lfTechnology:/etc/nginx/sites-enabled# systemctl restart nginx
root@bibek-lfTechnology:/etc/nginx/sites-enabled#
```

View in the browser



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

I made the test1 folder inside previous root folder test

```
root@bibek-lfTechnology:/var/www/test/test1# pwd
/var/www/test/test1
root@bibek-lfTechnology:/var/www/test/test1# ls
index.html
root@bibek-lfTechnology:/var/www/test/test1#
```

And an index.html file with message “test is successful”

```

<!DOCTYPE html>

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title>DevOps Assignment</title>
  <LINK href="styles.css" rel="stylesheet" type="text/css">
</head>

<body>

<h1>
  TEST is successful !!
</h1>
<p>
  From DevOps Internship !!
</p>

</body>
</html>

```

Created test2.conf file in sites-available

vi /etc/nginx/sites-available/test2.conf

```

server {
    listen 82;
    server_name localhost;
    root /var/www/test;
    index index.html;
    access_log /var/log/nginx/test2.log;
    error_log /var/log/nginx/test2-error.log;

    location /test1/ {
        index index.html;
    }

    #header_Security
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Feature-Policy " microphone 'none' ; geolocation 'none' " ;
}

```

Created soft-link in sites-enabled

**ln -rs /etc/nginx/sites-available/test2.conf /etc/nginx/sites-enabled/**

```

root@bibek-lfTechnology:/var/www/test/test1# ls /etc/nginx/sites-enabled/
reverse_proxy.conf test2.conf test.conf
root@bibek-lfTechnology:/var/www/test/test1# █

```

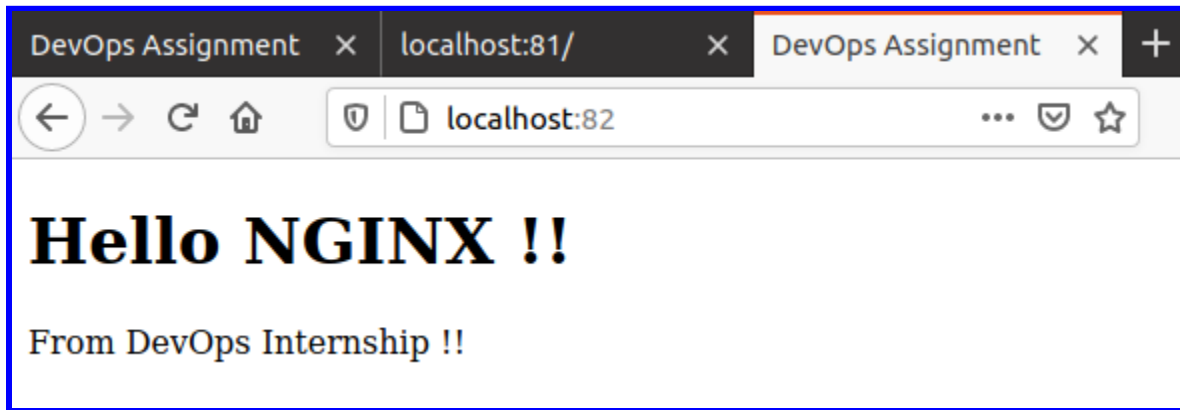
To check the syntax and restart the nginx

**nginx -t**

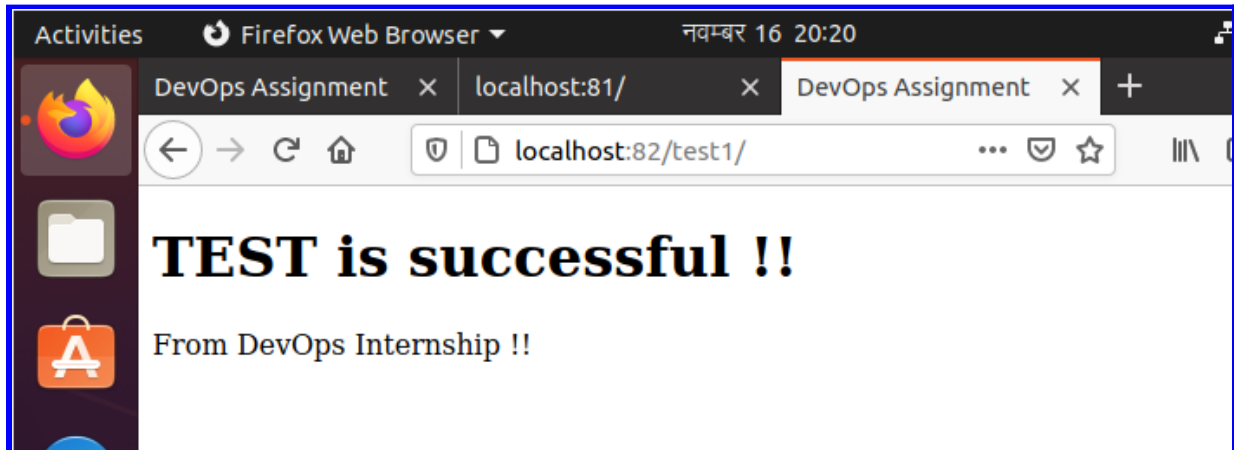
**systemctl restart nginx**

```
root@bibek-lfTechnology:/var/www/test/test1# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@bibek-lfTechnology:/var/www/test/test1# systemctl restart nginx
root@bibek-lfTechnology:/var/www/test/test1# pwd
```

On 82 port we get the root folder index.html messege



And with localhost:82/test1 we got the required message “Test is successful”



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

Created reverse\_proxy2.conf in sites-available directory

`vi /etc/nginx/sites-available/reverse_proxy2.conf`

```
server {
    listen 85;
    server_name localhost;

    location / {
        proxy_pass http://localhost:82/test1/;
    }
}
```

Created soft link in sites-enabled

`ln -rs /etc/nginx/sites-available/reverse_proxy2.conf /etc/nginx/sites-enabled/`

Checked the syntactical error

`nginx -t`

And restart nginx

`systemctl restart nginx`

```
root@bibek-lfTechnology:/var/www/test/test1# vi /etc/nginx/sites-available/reverse_proxy2.conf
root@bibek-lfTechnology:/var/www/test/test1# ln -rs /etc/nginx/sites-available/reverse_proxy2.conf /etc/nginx/sites-enabled/
root@bibek-lfTechnology:/var/www/test/test1# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@bibek-lfTechnology:/var/www/test/test1# systemctl restart nginx
```

And we can see the test is successful at localhost:85



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

LEMP stack

L = Linux

E = Nginx

M = MySQL (Not-needed for this assignment)

P = PHP

Our system is Ubuntu(Linux) and We have already installed nginx,  
We have to print info.php

PHP

Installing php-fpm

**apt install -y php-fpm**

```
Setting up php-fpm (2:7.4+75) ...  
Processing triggers for man-db (2.9.1-1) ...  
Processing triggers for systemd (245.4-4ubuntu3.4) ...  
Processing triggers for php7.4-cli (7.4.3-4ubuntu2.7) ...  
Processing triggers for php7.4-fpm (7.4.3-4ubuntu2.7) ...  
root@bibek-lfTechnology:/var/www/test#
```

php7.4 -fpm has been installed by default

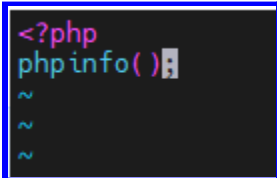
No need to install php-mysql as we are not integrating MySQL database

### Creating php folder in /var/www

**mkdir /var/www/php**

Creating file info.php

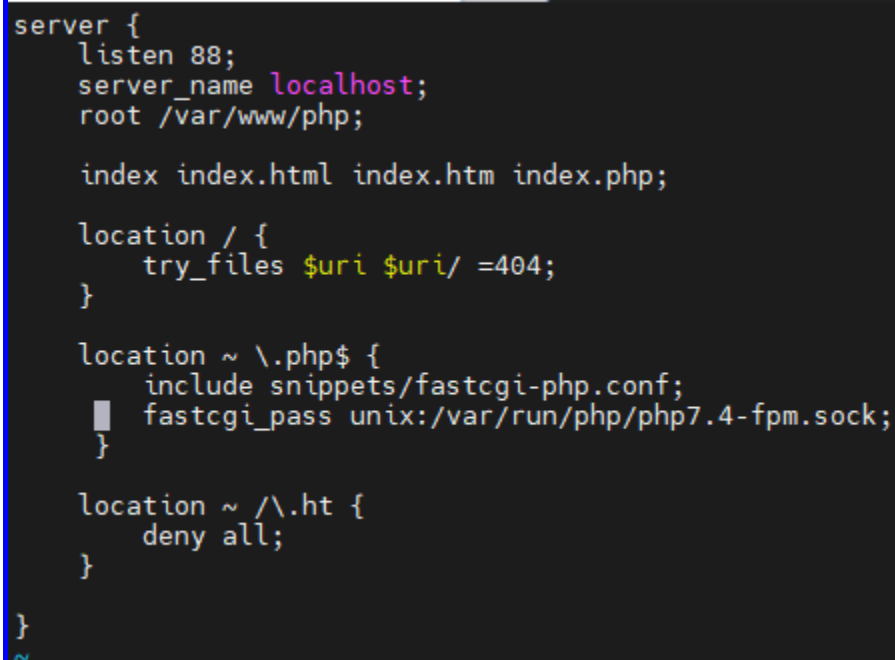
**vi /var/www/php/info.php**



```
<?php
phpinfo();
~
~
~
```

Creating php.conf to host php directory and print info.php

**vi /etc/nginx/sites-available**



```
server {
    listen 88;
    server_name localhost;
    root /var/www/php;

    index index.html index.htm index.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 I have used 88 port as 80 port is already occupied or we have to unlink the previous conf file.

And Index.html is also not created in root directory so it will throw error 404 error through try\_files

#### **location ~ \.php\$ :**

This location block handles the actual PHP processing by pointing Nginx to the fastcgi-php.conf configuration file and the php7.4-fpm.sock file, which declares what socket is associated with php-fpm and serves info.php in the root directory.

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

The last location block deals with .htaccess files, which Nginx does not process.

By adding the deny all directive, if any .htaccess files happen to find their way into the document root ,they will not be served to visitors.

Now creating symlink of php.conf from sites-available to sites-enabled

**ln -sr /etc/nginx/sites-available/php.conf /etc/nginx/sites-enabled/**

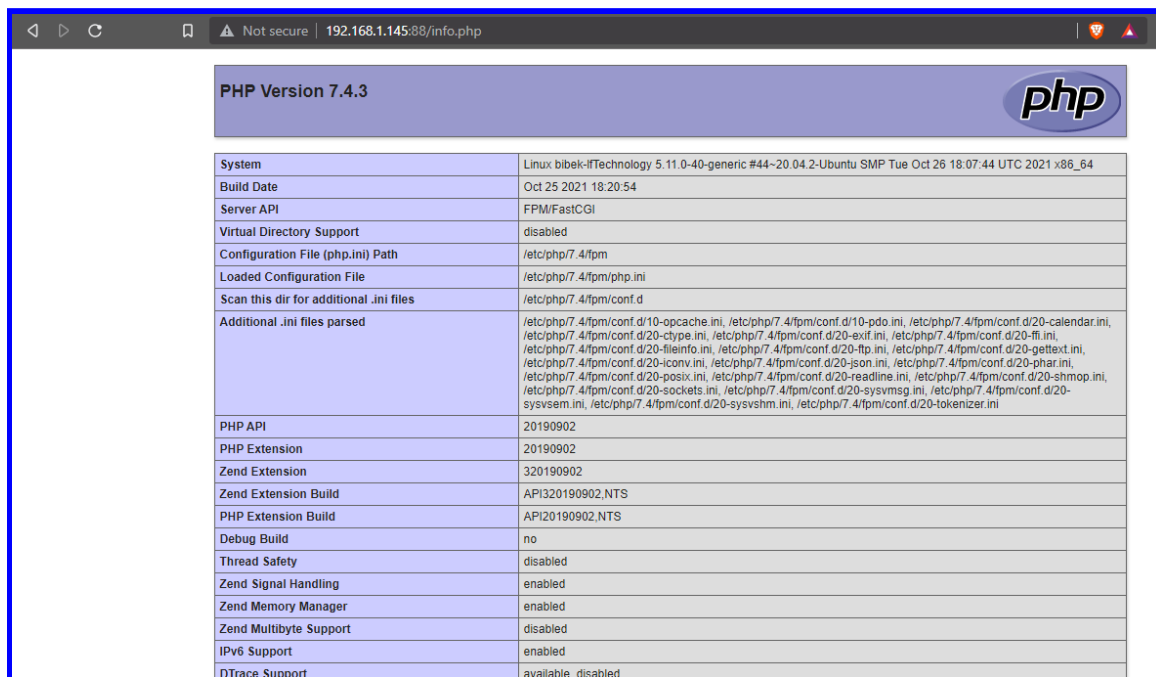
Checking for syntactical error

**nginx -t**

And restarting nginx

```
root@bibeK-lfTechnology:/var/www/test# ln -sr /etc/nginx/sites-available/php.conf /etc/nginx/sites-enabled/
root@bibeK-lfTechnology:/var/www/test# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@bibeK-lfTechnology:/var/www/test# systemctl restart nginx
```

My vm is in bridge mode, so if I hit my vm ip with 88 port/info.php, we can see the following output

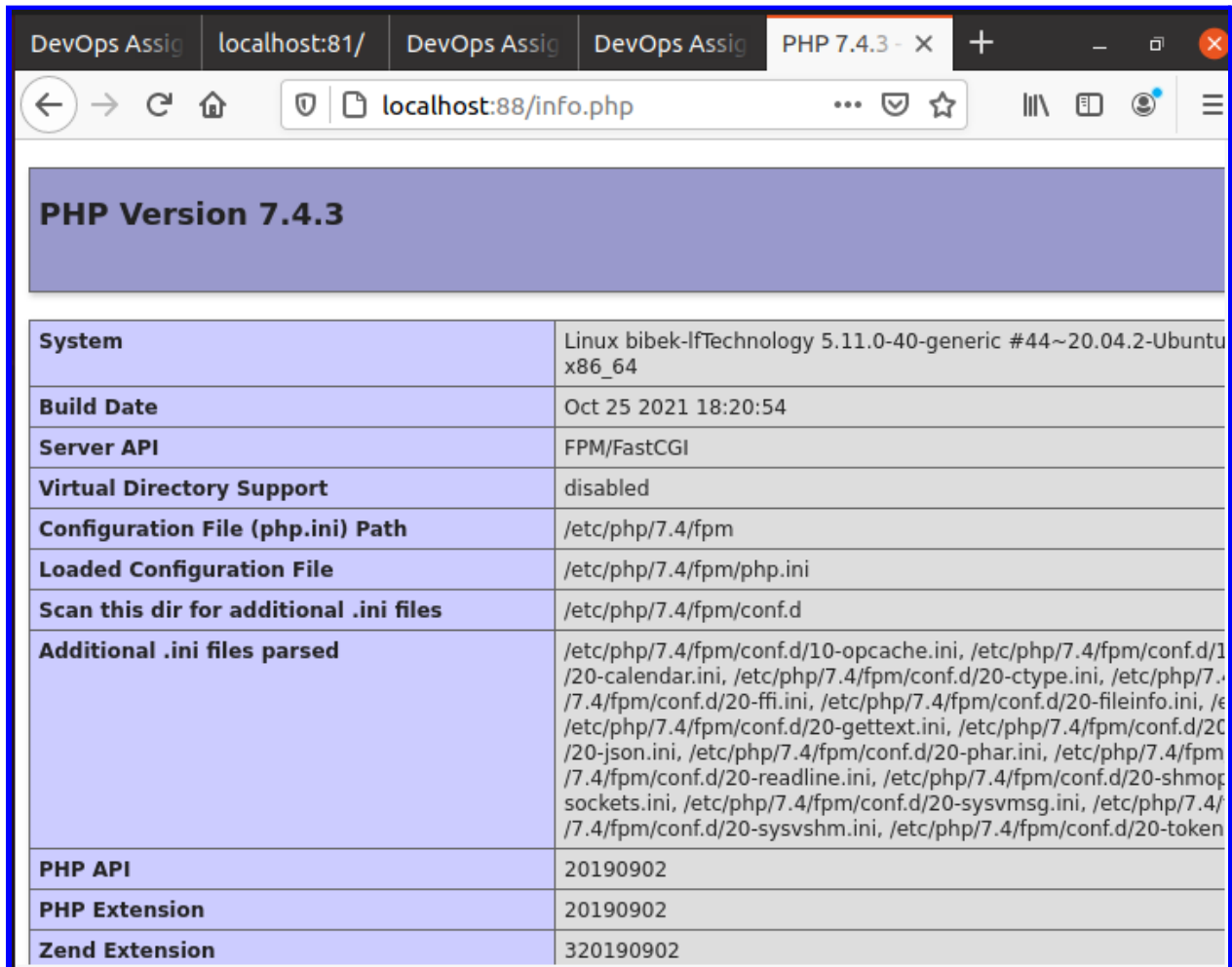


PHP Version 7.4.3

System	Linux bibeK-lfTechnology 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-fileinfo.ini, /etc/php/7.4/fpm/conf.d/20-iconv.ini, /etc/php/7.4/fpm/conf.d/20-ldap.ini, /etc/php/7.4/fpm/conf.d/20-mbstring.ini, /etc/php/7.4/fpm/conf.d/20-openssl.ini, /etc/php/7.4/fpm/conf.d/20-pdo_mysql.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
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902.NTS
PHP Extension Build	API20190902.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled



If we browse localhost on port 88 with info.php, We can see the same output



PHP Version 7.4.3	
System	Linux bibek-lfTechnology 5.11.0-40-generic #44~20.04.2-Ubuntu 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/12-calendar.ini, /etc/php/7.4/fpm/conf.d/20-ctype.ini, /etc/php/7.4/fpm/conf.d/20-fileinfo.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-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-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-sysvshm.ini, /etc/php/7.4/fpm/conf.d/20-tokenizer.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902