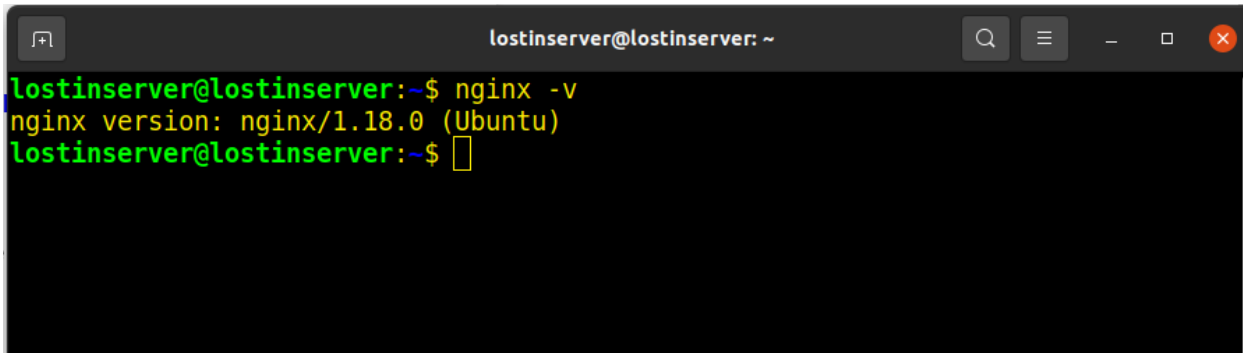
Installing nginx and hosting a simple html page

Nginx is one of the most popular web servers in the world and is responsible for hosting some of the largest and highest-traffic sites on the internet. It is a lightweight choice that can be used as either a web server or reverse proxy.

To install **nginx** use,

$ sudo apt install nginx



Check version of **nginx**,

$ nginx -v

Start the nginx service and check whether it is running or not,

$ sudo systemctl start nginx

$ sudo systemctl status nginx

```
lostinserver@lostinserver: ~
lostinserver@lostinserver:~$ sudo systemctl start nginx
[sudo] password for lostinserver:
lostinserver@lostinserver:~$ 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 preset:>
     Active: active (running) since Wed 2021-11-17 10:58:13 +0545; 6h ago
       Docs: man:nginx(8)
   Main PID: 831 (nginx)
      Tasks: 5 (limit: 9364)
     Memory: 6.4M
     CGroup: /system.slice/nginx.service
             ├─831 nginx: master process /usr/sbin/nginx -g daemon on; master_p>
             ├─832 nginx: worker process
             ├─833 nginx: worker process
             ├─834 nginx: worker process
             └─835 nginx: worker process

नवम्बर 17 10:58:13 lostinserver systemd[1]: Starting A high performance web serv>
नवम्बर 17 10:58:13 lostinserver systemd[1]: Started A high performance web serve>
lines 1-16/16 (END)
```

Now, let us host a simple **index.html** page with the message *"hello nginx"* ,

a) We first go to the location **"/var/www/html"** and see the files there ,

$ ls /var/www/html

```
lostinserver@lostinserver:~$ cd /var/www/html/
lostinserver@lostinserver:/var/www/html$ ls
index.nginx-debian.html
lostinserver@lostinserver:/var/www/html$
```

b) We initially have another .html file here, we rename that to **home.html** for now,

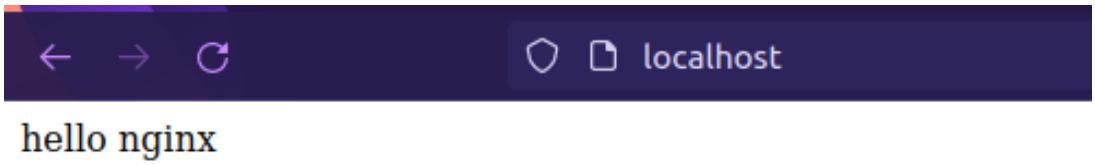
$ sudo mv index.nginx-debian.html home.html

c) Now, we create our own **index.html** page as,

$ sudo nano index.html

d) We add the content as below in this created file,

e) Now we check in our local ip (localhost) on default port of **nginx** i.e. 80 ,



hello nginx

We can see the site has been successfully hosted.

nginx header security and its uses

The nginx security HTTP headers are the response HTTP headers that the server can add in order to harden the security of HTTP exchange (browsing).

There are a few, and as the web evolves, more are being added. Each security header serves its own purpose. The nginx security headers are as,

- HTTP Strict Transport Security (HSTS)
- Public Key Pinning Extension for HTTP (HPKP)
- X-Frame-Options
- X-XSS-Protection
- X-Content-Type-Options
- Content-Security-Policy
- X-Permitted-Cross-Domain-Policies
- Referrer-Policy
- Expect-CT
- Feature-Policy

In most cases, HTTP security headers are added to responses, so that the browsers behave in a more secure way.

For example:

X-Content-Type-Options: nosniff

When a security header is sent in a response, it prevents browsers from trying to “guess” MIME types and such, forcing them to use what the server tells them.

This helps to harden security because a maliciously changed file on a compromised website has fewer chances to be run as an executable, thus preventing the infecting of the client machines.

There are several methods to assign headers. Let's assign some of the **nginx** security headers in our config file using **add_header** directive.

The **add_header** is the built-in directive in NGINX. However, it is the least intuitive in the way it is inherited, as well as limited in how it can work.

Let's use it and see in our browser for ourselves,

a) We edit our file,

$ sudo vim /etc/nginx/nginx.conf

b) We will add some of the response headers now. We add the following in our file,

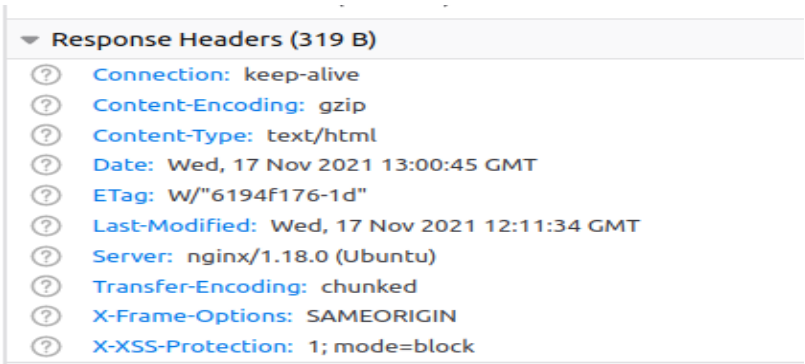
add_header X-Frame-Options SAMEORIGIN;
add_header X-XSS-Protection "1; mode=block";

We can place the above statement in http, server or location block.

```
# Adding Headers

add_header X-Frame-Options SAMEORIGIN;
add_header X-XSS-Protection "1; mode=block";
```

c) Now we can see in our browser as,



nginx reverse proxy to all http requests to node js api

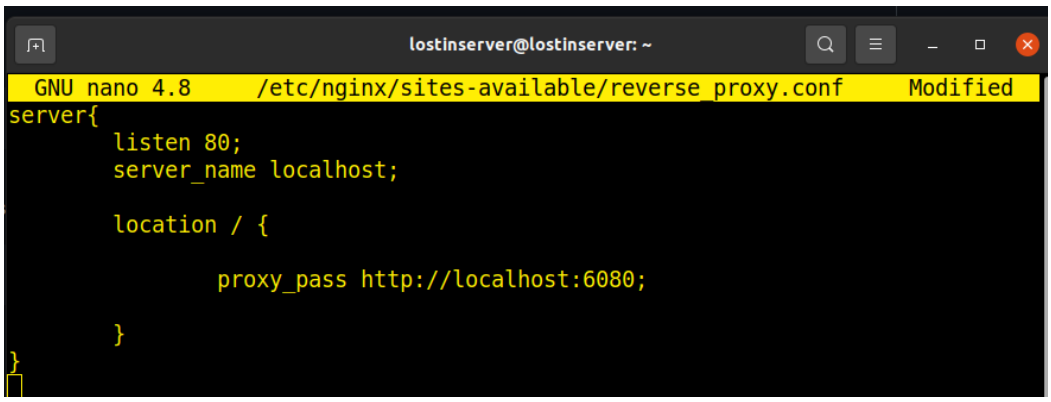
Our application is currently running and listening on **localhost**, but we need to set up a way for our users to access it. We set up the Nginx web server as a reverse proxy for this purpose.

Here, let's reverse proxy all http requests to our created node js api.

a) Initially create a **reverse_proxy.conf** file in the location */etc/nginx/sites-available* using,

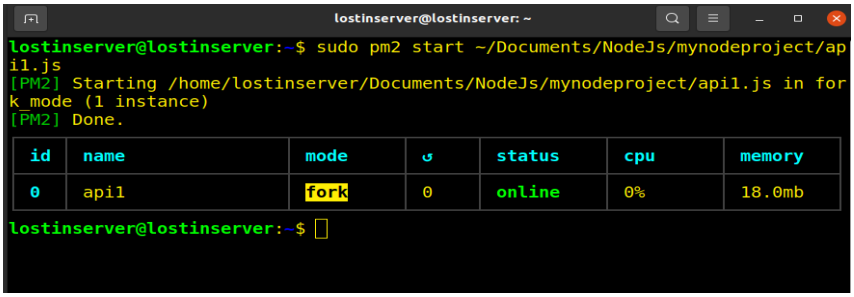
$ sudo nano /etc/nginx/sites-available/reverse_proxy.conf

b) Add the following in the file as shown ,



c) Start pm2 for api1.js i.e for 6080 port with,

$ **sudo pm2 start ~/Documents/NodeJs/mynodeproject/api1.js**



d) Now,we create a link from it to the sites-enabled directory,

$ sudo ln -rs /etc/nginx/sites-available/reverse_proxy.conf /etc/nginx/sites-enabled

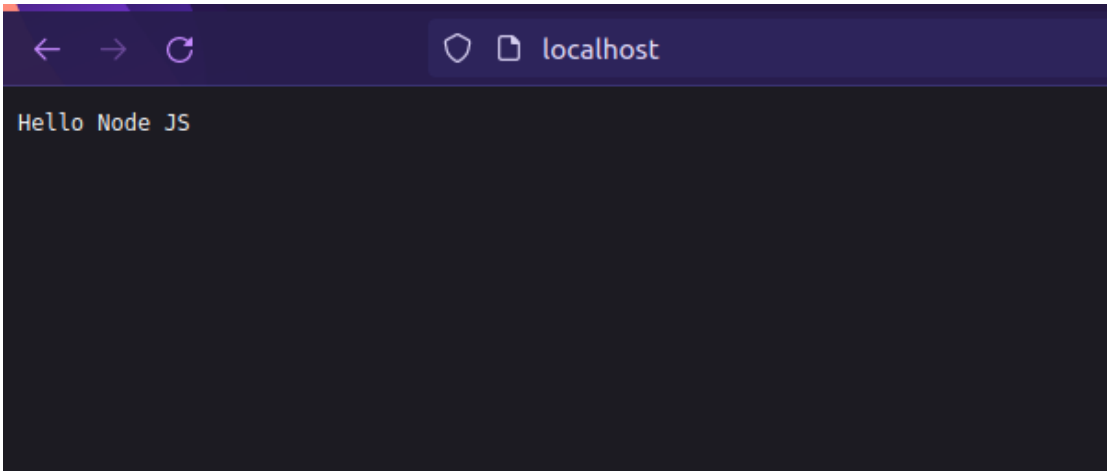
e) Now, for testing ,

$ sudo nginx -t

f) Restart the nginx to take effect,

$ sudo systemctl restart nginx

Now, we can see in our browser by browsing to **localhost:80** ,



Creating a test2.conf and listening on port 82 and to “location /test/” with message “test is successful”

So inorder to do this, we first create directories using,

$ sudo mkdir /var/www/test2/html/test

So here **test2, html** and **test** directories will be created as suggested above.

And we create a **index.html** page here with message *“test is successful”* as below,

Lets create a **test2.conf** file in */etc/nginx/sites-available* ,

$ sudo nano /etc/nginx/sites-available/test2.conf

We include the following in the file,

```
server {
        listen 82;
        listen [::]:82;
        root /var/www/test2/html;
        index index.html index.htm index.nginx-debian.html;
        server_name localhost;

        location /test/ {
        index index.html;
        }
}
```

We now create link for file as,

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

Test and restart the server to get it working,

$ sudo nginx -t
$ sudo systemctl restart nginx

We can see, we successfully received our message in **localhost:82/test/**

Reverse proxy all http traffic of port 82 to port 85

We create file as **reverse_proxy2.conf** inside */etc/nginx/sites-available* as,

$ sudo nano /etc/nginx/sites-available/reverse_proxy2.conf

Modify the file as below,

```
  GNU nano 4.8    /etc/nginx/sites-available/reverse_proxy2.conf
# Reverse proxy all http traffic of port 82 to 85

server{

        listen 85;
        server_name localhost;

        location / {

                proxy_pass http://localhost:82/;

        }
}
```

Now similar to above create link, test it and restart the **nginx** server,

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

$ sudo nginx -t

$ sudo systemctl restart nginx

test is successful

We can see the site is now running on **localhost:85/test** which was previously running on **localhost:82/test.**

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

LEMP is an open-source web application stack used to develop web applications. The term LEMP is an acronym that represents L for the Linux Operating system, Nginx (pronounced as engine-x, hence the E in the acronym) web server, M for MySQL database, and P for PHP scripting language.

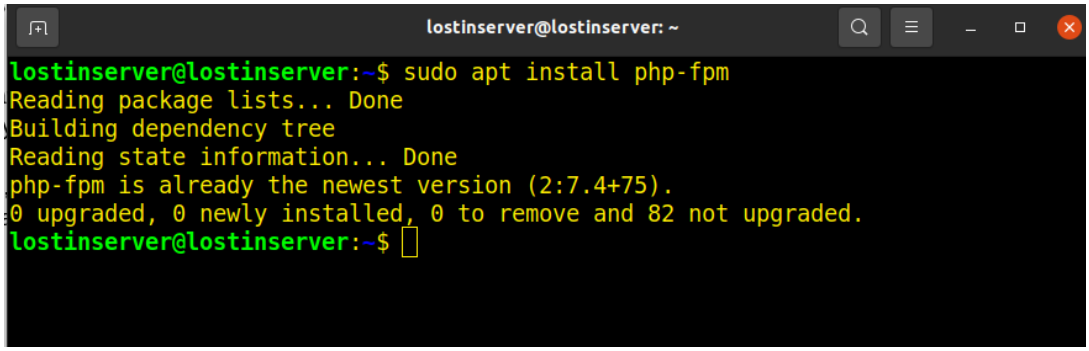
So, we already are using Linux OS and have installed nginx, we now proceed further for installing PHP (excluding mysql) as below :

a) Firstly we will need to install **php-fpm**, which stands for “fastCGI process manager” as nginx doesn’t contain native PHP processors.

   $ sudo add-apt-repository universe

b) Install **php-fpm** as,

   $ sudo apt install php-fpm



c) Let’s add info.php file in our */var/www/php* location ,

   $ sudo mkdir /var/www/php
   $ sudo touch /var/www/php/info.php

d) Let's add the below to the **info.php** ,

```
lostinserver@lostinserver: /etc/nginx/sites-available
  GNU nano 4.8                /var/www/php/info.php                Modified
<?php
phpinfo();
?>
```

e) Now we create another php.conf file inside /etc/nginx/sites-available/ as,

$ sudo nano /etc/nginx/sites-available/php.conf

f) Add the following to the config file,

```
  GNU nano 4.8          /etc/nginx/sites-available/php.conf          Modified
        listen 80;
        root /var/www/php;
        index index.php index.html index.htm index.nginx-debian.html;
        server_name localhost;

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

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
        }

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

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell
```

Config File

```
server {
  listen 80;
  root /var/www/php;
  index index.php index.html index.htm index.nginx-debian.html;
  server_name localhost;

   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;
      }
   }
```

g) We now change our default server **port 80** of **default** from */etc/nginx/sites-available* to **port 83** and **reverse_proxy.conf** from */etc/nginx/sites-available/* to **port 8089** for now as we have assigned **port 80** for **php.conf** above.

```
#
server {
        listen 83 ;
        listen [::]:83 ;
```

```
  GNU nano 4.8     /etc/nginx/sites-available/reverse_proxy.conf
server{
        listen 8089;
        server_name localhost;

        location / {

                proxy_pass http://localhost:6080;

        }
}
```

h) Create a link for the file, test it and restart the nginx server,

$ sudo ln -rs /etc/nginx/sites-available/php.conf /etc/nginx/sites-enabled/

$ sudo nginx -t

$ sudo systemctl restart nginx

i) Now we can see in our browser **localhost/info.php** as,

localhost/info.php

PHP Version 7.4.3

System	Linux lostinserver 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
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
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

Configuration

calendar