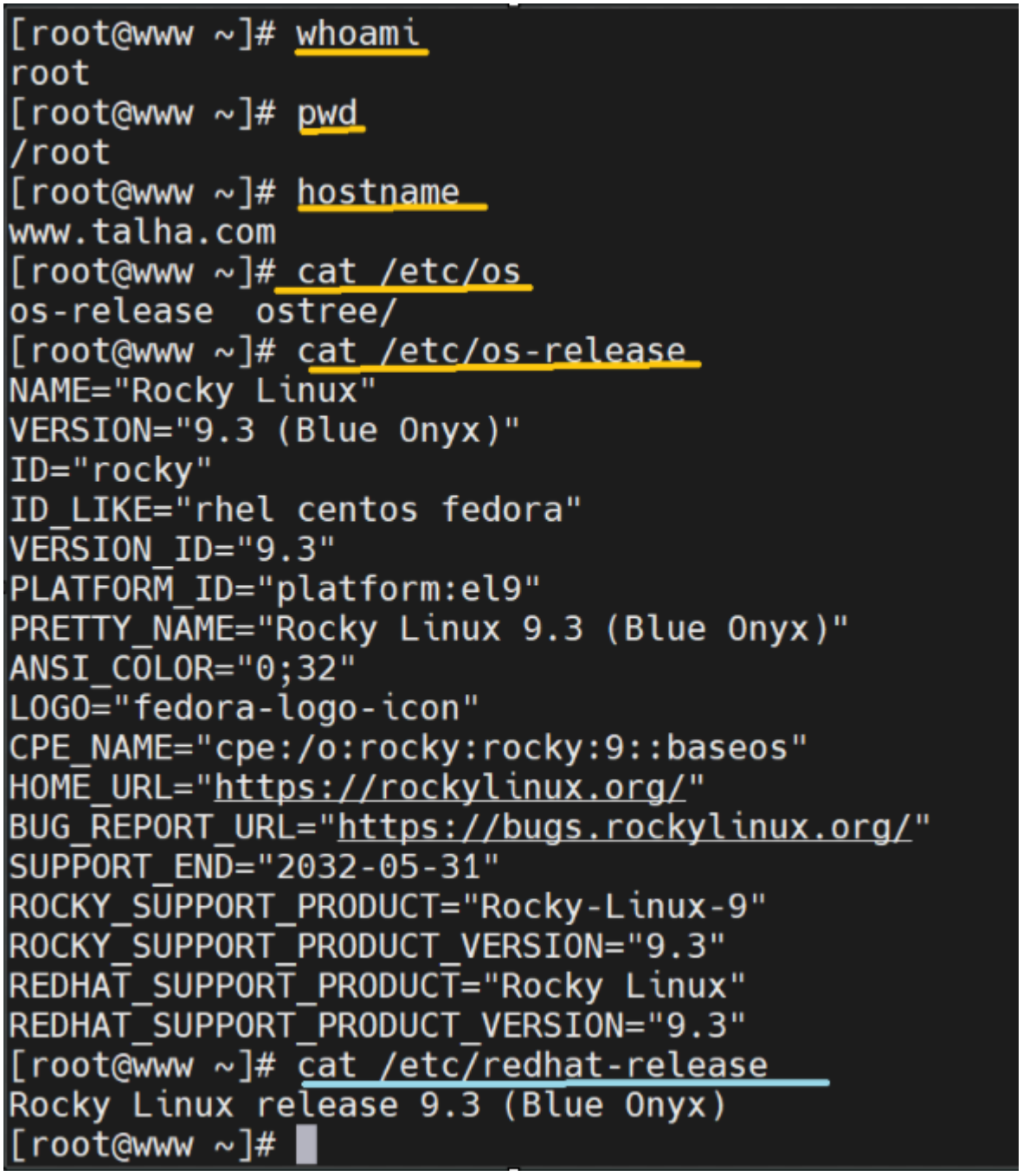
3. Hosting Website on NGINX Server

In this project I will deploy a website on the `NIGIX server`

STEP 1 (Checking System Information):

- I have started and fired up my `Linux Machine` and now i will run some basic command to learn about the `username` `current working directory` `hostname` `operating system info and details`

```
whoami   #Current user name information
pwd                       # Print the currentt directory in w
hostname                  # The hostname of your machine
cat /etc/os-release       # Detail information about your ope
cat /etc/redhat-release   # Full name of your current OS
```



- It gives me following information
  - user = root
  - pwd = /root
  - hostname = www.talha.com
  - `cat /etc/os-release` = Rocky Linux version 9.3 with version 9.3 and etc.
  - `cat /etc/redhat-release` = Rocky Linux release 9.3

- I want to change the `hostname` name to `www.nginixproject.com` . I will run the command

```
hostnamectl set-hostname www.nginixproject.com  #This will ch


# To verify i will run the command `hostname`
hostname
```

```
[root@www ~]# hostnamectl set-hostname www.nginixproject.com
[root@www ~]# hostname
www.nginixproject.com
[root@www ~]#
```

- It is always a good practice to know about the `ip address` of the machine you are using. Run the following command

```
ifconfig
```

```
[root@www ~]# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.10  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fee7:2777  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e7:27:77  txqueuelen 1000  (Ethernet)
        RX packets 12739  bytes 13850217 (13.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4911  bytes 373912 (365.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The current `ip address` of the machine is `192.168.1.10`

Step 2 (Updating OS and Installing Nginx):

Checking system updates and installing:

- It is a good practice to keep your `operating system` up to date. before doing any project or make it a `scheduled task` to check it periodically.
- We will first of all check is there any `updates` available for our current OS

```
dnf check-update  # A list of available updates for OS wi
```

```
yum.noarch                                                  4.14.0-9.el9
Obsoleting Packages
grub2-tools.x86_64                                          1:2.06-77.el9
    grub2-tools.x86_64                                      1:2.06-70.el9_3.1.rocky.0.2
grub2-tools-efi.x86_64                                      1:2.06-77.el9
    grub2-tools.x86_64                                      1:2.06-70.el9_3.1.rocky.0.2
grub2-tools-extra.x86_64                                    1:2.06-77.el9
    grub2-tools.x86_64                                      1:2.06-70.el9_3.1.rocky.0.2
grub2-tools-minimal.x86_64                                  1:2.06-77.el9
    grub2-tools.x86_64                                      1:2.06-70.el9_3.1.rocky.0.2
[root@www ~]# dnf check-update
```

We have a lot of updates available.

- To make the update we run the command

```
dnf update -y  # This is automatically startt updating our OS
```

```
 kernel-core                                 x86_64       5.14.0-427.18.1.el9_4
 kernel-modules                              x86_64       5.14.0-427.18.1.el9_4
 kernel-modules-core                         x86_64       5.14.0-427.18.1.el9_4
 pipewire-jack-audio-connection-kit-libs     x86_64       1.0.1-1.el9
Installing weak dependencies:
 evolution-data-server-ui                    x86_64       3.40.4-9.el9

Transaction Summary
================================================================================
Install    8 Packages
Upgrade  432 Packages

Total download size: 1.2 G
Downloading Packages:
```

We have to download and install a total of `1.2G` update. Because we have used `-y` while running the command , it automatically started to download.

- After the downloading and installation of the the update. A `complete message` will be displayed

```
Installed:
  evolution-data-server-ui-3.40.4-9.el9.x86_64          grub2-tools-efi-1:2.06-77.el9.x86_64
  grub2-tools-extra-1:2.06-77.el9.x86_64                kernel-5.14.0-427.18.1.el9_4.x86_64
  kernel-core-5.14.0-427.18.1.el9_4.x86_64              kernel-modules-5.14.0-427.18.1.el9_4.x86_64
  kernel-modules-core-5.14.0-427.18.1.el9_4.x86_64      pipewire-jack-audio-connection-kit-libs-1.0.1-1.el9.x86_64

Complete!
```

or you can check it again by running the command `dnf update`

```
[root@www ~]# dnf update
Last metadata expiration check: 0:49:03 ago on Sun 02 Jun 2024 03:48:39 AM WIB.
Dependencies resolved.
Nothing to do.
Complete!
[root@www ~]#
```

Installing `Nginx Server`

- After we have completed our `system OS update` . We can now install `Nginx Server`.
- First of all we will check either it is already installed on our machine by running the `rpm` command :

```
rpm -qa | grep nginx
```

```
[root@www ~]# rpm -qa | grep nginx
[root@www ~]#
```

No message was returned this shows that , there is no `NGINX` is installed on it.

- To download and install the `nginx package` along with its other `dependencies` we will run the command :

```
dnf install -y nginx* # Using -y will install the package wit
                                                # wildcard will i
```

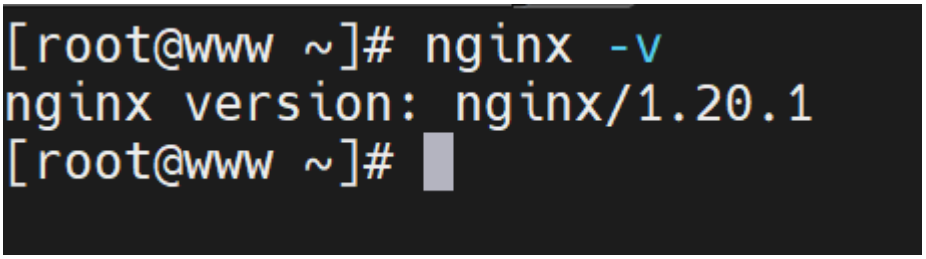
```
Installed:
  nginx-1:1.20.1-14.el9_2.1.x86_64                          nginx-all-modules-1:1.20.1-14.el9_2.1.noarch
  nginx-core-1:1.20.1-14.el9_2.1.x86_64                     nginx-filesystem-1:1.20.1-14.el9_2.1.noarch
  nginx-mod-http-image-filter-1:1.20.1-14.el9_2.1.x86_64    nginx-mod-http-perl-1:1.20.1-14.el9_2.1.x86_64
  nginx-mod-http-xslt-filter-1:1.20.1-14.el9_2.1.x86_64     nginx-mod-mail-1:1.20.1-14.el9_2.1.x86_64
  nginx-mod-stream-1:1.20.1-14.el9_2.1.x86_64               rocky-logos-httpd-90.15-2.el9.noarch

Complete!
[root@www ~]#
```

The package is installed to double check we can again run the command `rpm -qa | grep nginx`

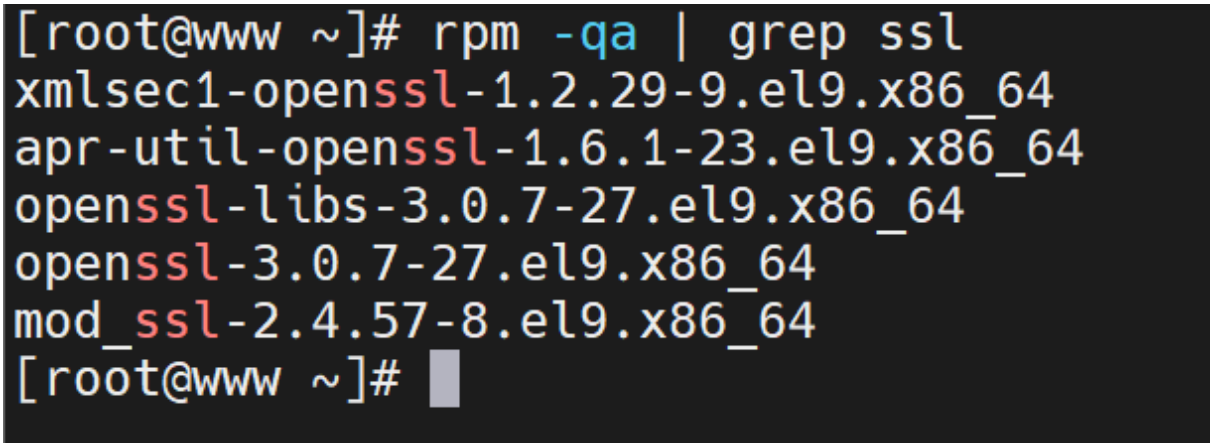
- We can also check the `version of nginx` which will also confirm its installation to our machine using the command `nginx -v`

```
nginx -v
```



- We also need to have package name `openSSL` and `mod_ssl` which will help us in generating keys if we want to install the `SSL/TLS` certificate for our website.
- We can check the package using the command:

```
rpm -qa | grep openssl
```



The image shows that the packages are installed already. If they are installed we should have installed them using the command :

```
dnf install openssl
dnf install mod_ssl
```

- Now `start` the `nginx server` , run the command:

```
systemctl start nginx
```

```
[root@www ~]# systemctl start nginx
```

if there is no message it means that the `service has been started`. It can be checked using the command

```
systemctl status nginx
```

```
[root@www ~]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; preset: disabled)
     Active: active (running) since Sun 2024-06-02 04:56:04 WIB; 10s ago
    Process: 80049 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Process: 80050 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 80051 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 80052 (nginx)
      Tasks: 6 (limit: 19286)
     Memory: 12.4M
        CPU: 572ms
     CGroup: /system.slice/nginx.service
             ├─80052 "nginx: master process /usr/sbin/nginx"
             ├─80053 "nginx: worker process"
             ├─80054 "nginx: worker process"
             ├─80055 "nginx: worker process"
             ├─80056 "nginx: worker process"
             └─80057 "nginx: worker process"
```

The `nginx` is activated. But if you look closely in the `3rd line` the `service is disabled`

- To enable the service of nginx , and to make it automatically start when we run our machine again we will execute the following command:

```
systemctl enable nginx.service --now  #
```

```
[root@www ~]# systemctl enable nginx.service --now
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[root@www ~]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Sun 2024-06-02 04:56:04 WIB; 4min 58s ago
   Main PID: 80052 (nginx)
      Tasks: 6 (limit: 19286)
     Memory: 12.4M
        CPU: 572ms
     CGroup: /system.slice/nginx.service
             ├─80052 "nginx: master process /usr/sbin/nginx"
             ├─80053 "nginx: worker process"
             ├─80054 "nginx: worker process"
             ├─80055 "nginx: worker process"
             ├─80056 "nginx: worker process"
             └─80057 "nginx: worker process"

Jun 02 04:56:04 www.nginixproject.com systemd[1]: Starting The nginx HTTP and reverse proxy server...
Jun 02 04:56:04 www.nginixproject.com nginx[80050]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Jun 02 04:56:04 www.nginixproject.com nginx[80050]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Jun 02 04:56:04 www.nginixproject.com systemd[1]: Started The nginx HTTP and reverse proxy server.
```

Step 3 (Testing The Nginx Server):

- In order to test the `nginix` we have few commands . First we will check either the `configuration` of the `nginix` is okay or have some issues and also test it using the command `nginx -t`:

```
nginix -t
```

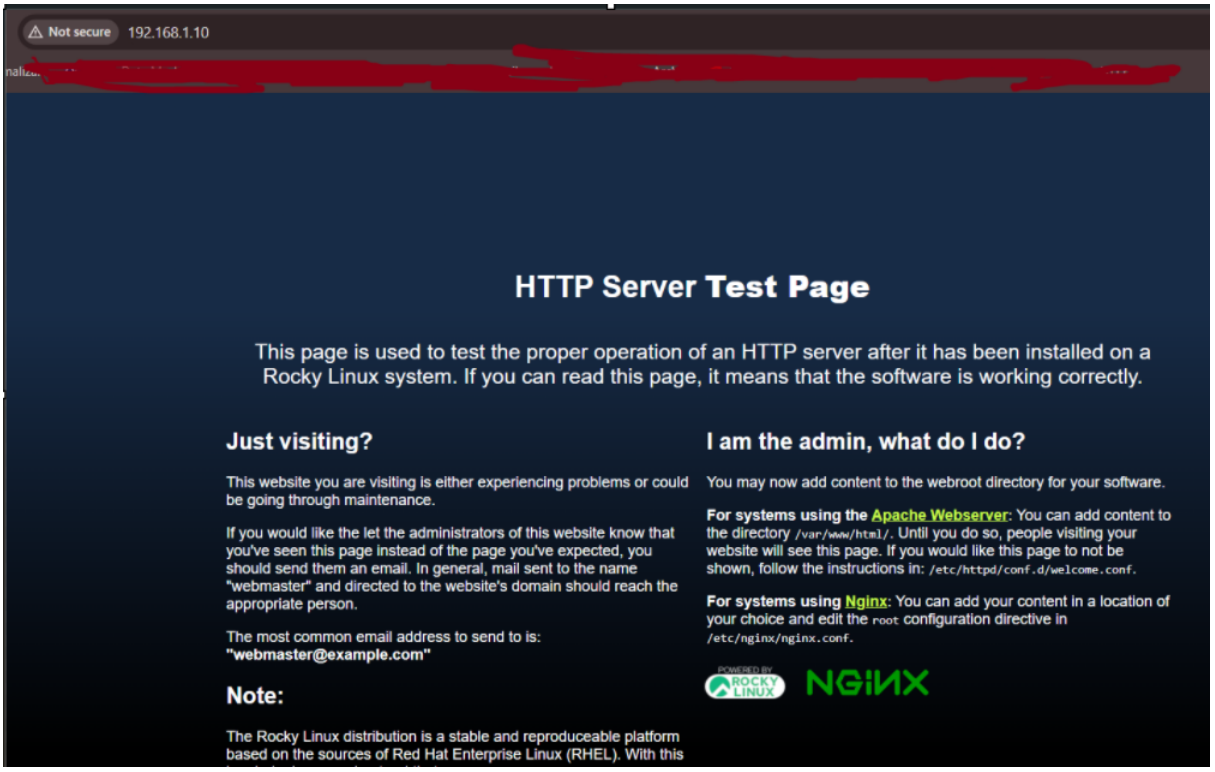
```
[root@www ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@www ~]#
```

- If we want to view the detail information and also wanted to print it we can use the command `nginx -T`
- -In order to view `all the command` associated with nginx and their functions we run the command `nginx -h`

```
[root@www ~]# nginx -h
nginx version: nginx/1.20.1
Usage: nginx [-?hvVtTq] [-s signal] [-p prefix]
             [-e filename] [-c filename] [-g directives]

Options:
  -?,-h         : this help
  -v            : show version and exit
  -V            : show version and configure options then exit
  -t            : test configuration and exit
  -T            : test configuration, dump it and exit
  -q            : suppress non-error messages during configuration testing
  -s signal     : send signal to a master process: stop, quit, reopen, reload
  -p prefix     : set prefix path (default: /usr/share/nginx/)
  -e filename   : set error log file (default: /var/log/nginx/error.log)
  -c filename   : set configuration file (default: /etc/nginx/nginx.conf)
  -g directives : set global directives out of configuration file
```

- We can `check and test` the nginx on our `web browser` by entering the `ip address` as we seen above the `ip address` of our machine is `192.168.1.10`



This shows that the `ngnix server` is up and running.

Step 4 (Nginx Files and Directories)

- After `installing package` of Nginx. The files of the `nginx` can be located in:

```
cd /etc/nginx
```

```
[root@www /]# cd /etc/nginx/
[root@www nginx]# ls
conf.d        fastcgi.conf.default    koi-utf     mime.types.default  scgi_params          uwsgi_params.default
default.d     fastcgi_params          koi-win     nginx.conf          scgi_params.default  win-utf
fastcgi.conf  fastcgi_params.default  mime.types  nginx.conf.default  uwsgi_params
[root@www nginx]#
```

- Here directory name `conf.d` is used to maintain the `virtual host` aka `v-shost` , so we can manage and `host` more then one website on a single machine.
- The main `configuration file of nginx` is `nginx.conf` . In the file we can do severla task e.g `set port number` ,

```
server {
    listen       80;
    listen       [::]:80;
    server_name  _;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    error_page 404 /404.html;
    location = /404.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

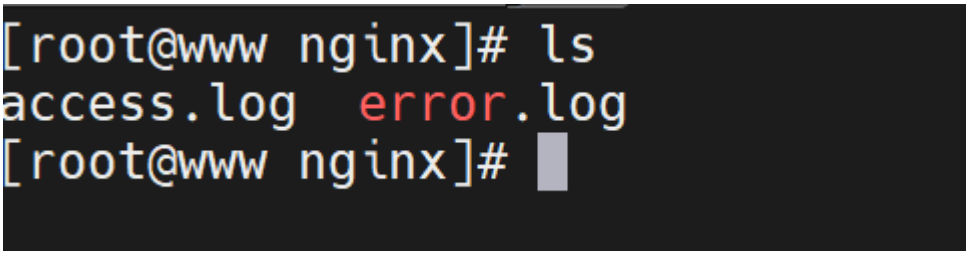
`view log file location` ,

```
access_log  /var/log/nginx/access.log  main;
```

and many more information is available. It is `highly recommended` not to make any changes until it is very important. As a safety always make a backup before making changes inn the configuration file.

- To view the `error logs` in your `nginx` we have to go to the directory :

```
cd /var/log/nginix
```



- The location to place our `website code` is :

```
    cd /var/www/html/
```

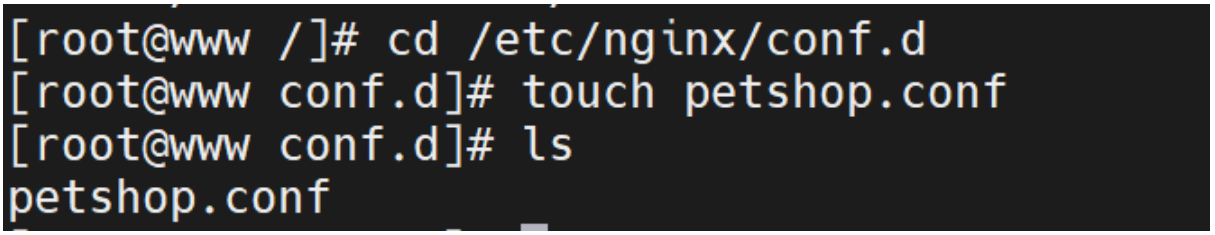
Step 5 (Configuring The Virtual Host)

- We are doing to deploy a website name `petshop` on our `nginix` server. To do so, we have to make a file and configure the file as per our requirement. We will go into the directory:

```
cd /etc/nginx/conf.d
```

- In that directory we will create a file name `petshop.conf` :

```
touch petshop.conf
```



> ***important : It is worth noting here that the file name we create should have an extension `.conf` . Apart from it it is consider as a good practice to have the website name as the file as well , so we can easily differentiate if we have two or more virtual host***

- Now we have to do some editing in the file `petshop.conf` . Here it is worth mentioning that we must understand the syntax of the `nginx` when we put some code in the configuration file.
- The `editing in nginx` is consist of two part .
    1. Block : A block in the `configuration file` start with a `variable or a name` e.g `server` or `http` etc. After the declaration of the name of the block , we make `curly brackets {}` . Inside the curly brackets . There can be multiple

blocks inside one block. Inside the block we put the second part. For now a block look like this :

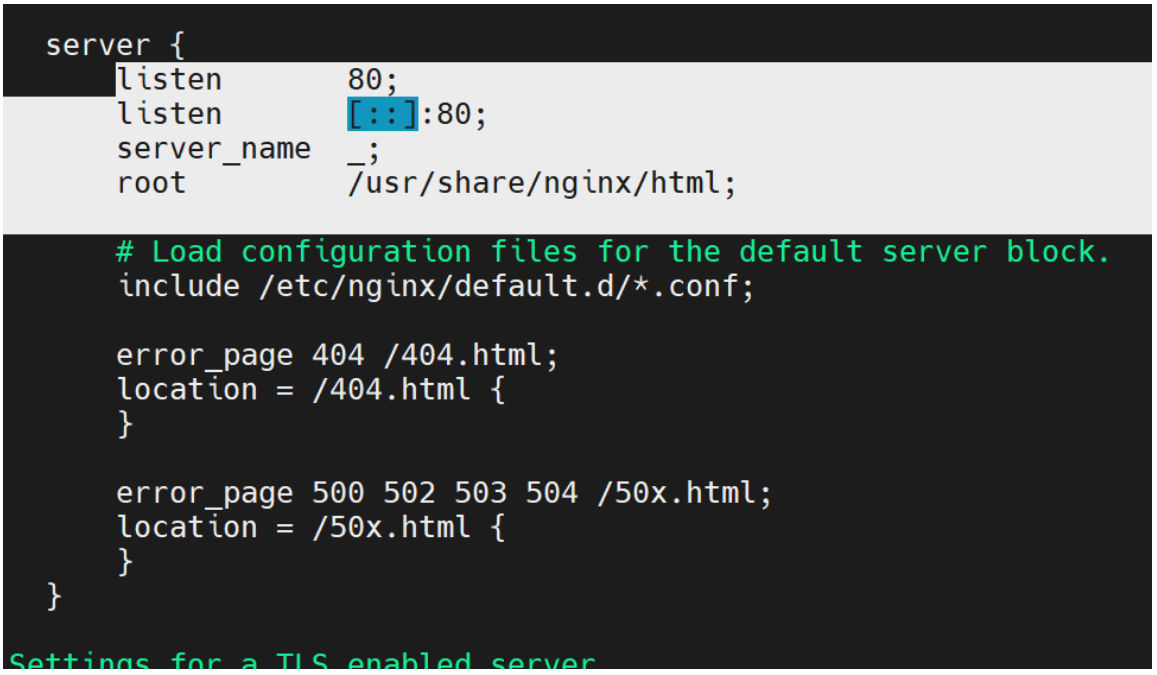
```
 server {
       listen       80;
       listen       [::]:80;
       server_name  _;
       root         /usr/share/nginx/html;

       # Load configuration files for the default server
       include /etc/nginx/default.d/*.conf;

       error_page 404 /404.html;
       location = /404.html {
       }

       error_page 500 502 503 504 /50x.html;
       location = /50x.html {
       }
   }
```

- Directives : The arguments we passe inside a block is called blocks. The can be multiple



The highlighted portion is called `directives` inside a `block`

- Now , coming back to our file name `petshop.conf` in the directory `cd /etc/nginx/conf.d`. Open the file in the `vim editior`

```
vim petshop.conf
```

We will configure the following settings:

```
server {
        listen 80 default_server;   #Define the port and also
        server_name nginxproject.local  www.nginxproject.com;
        index index.html index.php index.htm; #Defining the f
        root /var/www/petshop;  #Location where website code

}
```

Save the file and now `test nginx server` again using the command :

```
nginx -t
```

```
[root@www conf.d]# vim petshop.conf
[root@www conf.d]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@www conf.d]#
```

Our nginx server is configured successfully.

- We also have to configure another file in :

```
vim /etc/hosts
```

The default setting in the file look like this :

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

We need to add the `ip address` and also the `hostname url`

```
192.168.1.10 www.nginxproject.com
```

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.1.10 www.nginxproject.com
~
```

Step 6 (Downloading the Website Code):

- We have mentioned that a directory name `petshop` is present in the location `cd /var/www/` . So we have to create a directory in `cd /var/www

```
cd /var/www/

mkdir petshop
```

```
[root@www var]# cd www
[root@www www]# ls
[root@www www]# mkdir petshop
[root@www www]# pwd
/var/www
[root@www www]# cd petshop/
[root@www petshop]# pwd
/var/www/petshop
[root@www petshop]#
```

- We have to download the code of the `website` from the url `https://www.free-css.com/assets/files/free-css-templates/download/page284/pet-shop.zip` `.` We will run the command

```
wget https://www.free-css.com/assets/files/free-css-templates
```

```
[root@www petshop]# wget https://www.free-css.com/assets/files/free-css-templates/download/page284/pet-shop.zip
--2024-06-02 06:26:42--  https://www.free-css.com/assets/files/free-css-templates/download/page284/pet-shop.zip
Resolving www.free-css.com (www.free-css.com)... 217.160.0.242, 2001:8d8:100f:f000::28f
Connecting to www.free-css.com (www.free-css.com)|217.160.0.242|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1061173 (1.0M) [application/zip]
Saving to: ‘pet-shop.zip.1’

pet-shop.zip.1             100%[=====================================>]   1.01M   166KB/s    in 7.1s

2024-06-02 06:26:57 (147 KB/s) - ‘pet-shop.zip.1’ saved [1061173/1061173]
```

- Unzip the file and now we have `index,html` and remaining content on our directory which we mention in the `petshop.conf` file

Part 7 (Checking port:80 is configured to NGINX):

- It is important to check that `port 80` is configured to `NGINX` because chances are previously you have configured it to `Apache web server`. In tha case, `nginx` will not work.
- So it is highly recommended to check and run the following command to check it :

```
lsof -i:80
```

```
[root@www ~]# lsof -i:80
COMMAND   PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
nginx   80052  root    6u  IPv4 137347      0t0  TCP *:http (LISTEN)
nginx   80052  root    7u  IPv6 137348      0t0  TCP *:http (LISTEN)
nginx   80053 nginx    6u  IPv4 137347      0t0  TCP *:http (LISTEN)
nginx   80053 nginx    7u  IPv6 137348      0t0  TCP *:http (LISTEN)
nginx   80054 nginx    6u  IPv4 137347      0t0  TCP *:http (LISTEN)
nginx   80054 nginx    7u  IPv6 137348      0t0  TCP *:http (LISTEN)
nginx   80055 nginx    6u  IPv4 137347      0t0  TCP *:http (LISTEN)
nginx   80055 nginx    7u  IPv6 137348      0t0  TCP *:http (LISTEN)
nginx   80056 nginx    6u  IPv4 137347      0t0  TCP *:http (LISTEN)
nginx   80056 nginx    7u  IPv6 137348      0t0  TCP *:http (LISTEN)
nginx   80057 nginx    6u  IPv4 137347      0t0  TCP *:http (LISTEN)
nginx   80057 nginx    7u  IPv6 137348      0t0  TCP *:http (LISTEN)
[root@www ~]#
```

It clearly shows that `port 80` is configured to `nginx`

We can also check it using the `netstat` command:

```
netstat -pan | grep nginx
```

```
[root@www ~]# netstat -pan | grep nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      80052/nginx: master
tcp6       0      0 :::80                   :::*                    LISTEN      80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135556   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135551   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135555   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135559   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135554   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135557   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135560   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135552   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135558   80052/nginx: master
unix  3      [ ]         STREAM     CONNECTED     135553   80052/nginx: master
```

Part 8 (Firewall Configuration):

- We have to add the `services` in the `firewall` to allow the `http` and `https` communication. We run the command :

```
firewall-cmd --permanent --add-service=http --zone=public
firewall-cmd --permanent --add-service=https --zone=public
```

```
[root@www ~]# firewall-cmd --permanent --add-service=http --zone=public
Warning: ALREADY_ENABLED: http
success
[root@www ~]#
```

```
[root@www ~]# firewall-cmd --permanent --add-service=https --zone=public
Warning: ALREADY_ENABLED: https
success
[root@www ~]#
```

- Both services are added already. We can also check the list of all the services configured on our firewall

```
firewall-cmd --list-services
```

```
[root@www ~]# firewall-cmd --list-services
cockpit dhcpv6-client http https ssh
```

- After setting up the services on `firewall` it is good practice to restart it

```
systemctl restart firewalld
```

- Also restart the `nginx`

```
systemctl restart ngnix
```

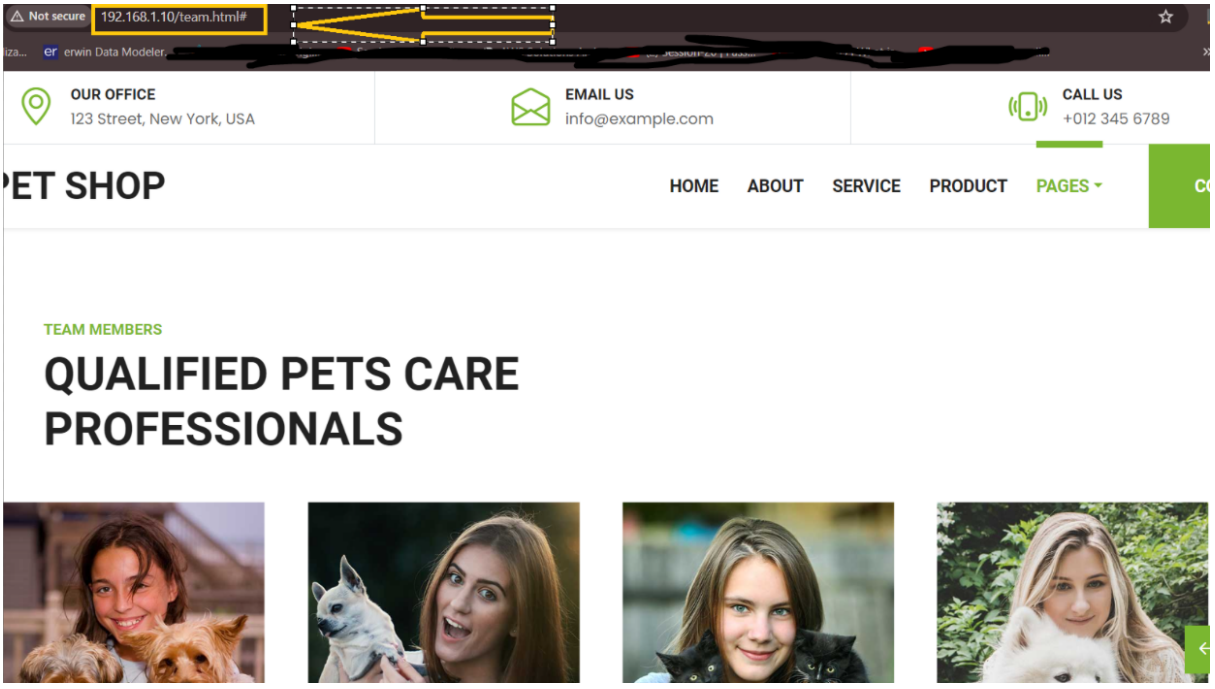
- Check the configuration again using the command

```
nginx -t
```

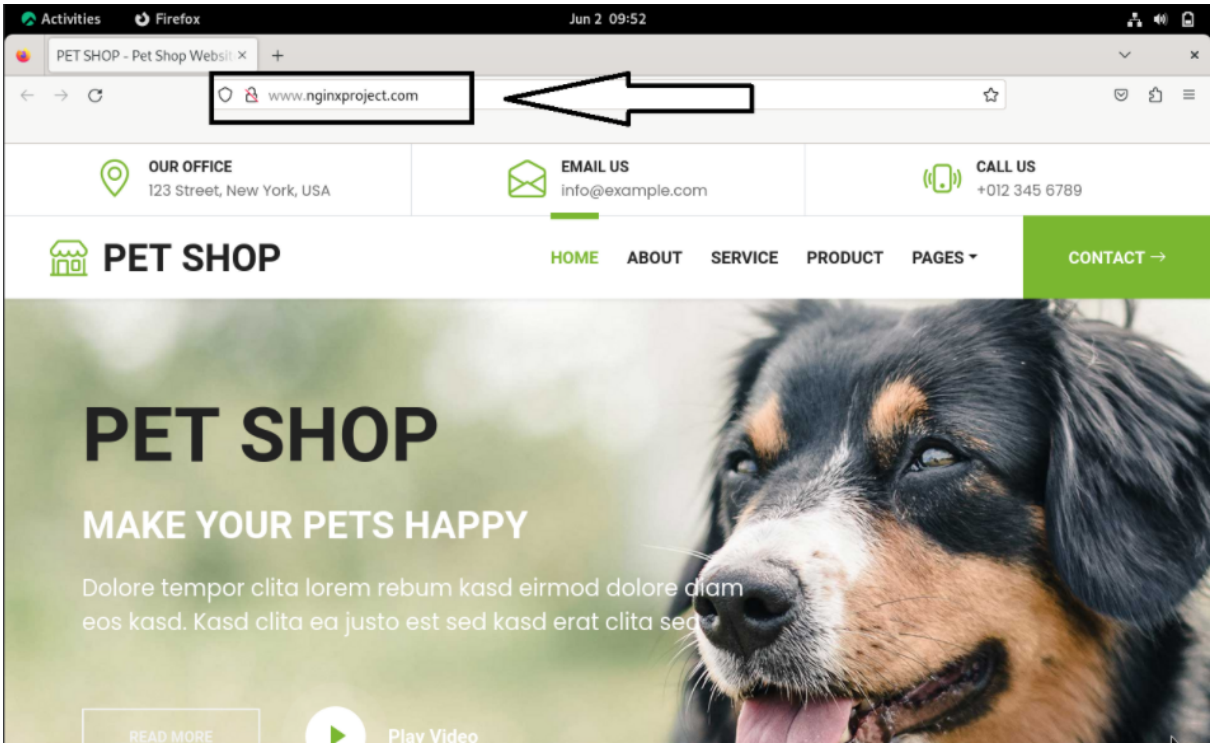
If there is no error. Fire up the `web brwwer` and put test your website

Step 9 (Testing Website)

- To test your website which you have hosted . Open the web browser and type in the ip address of your machine `192.168.1.10`



- Also , use the `url` `www.nginxproject.com`

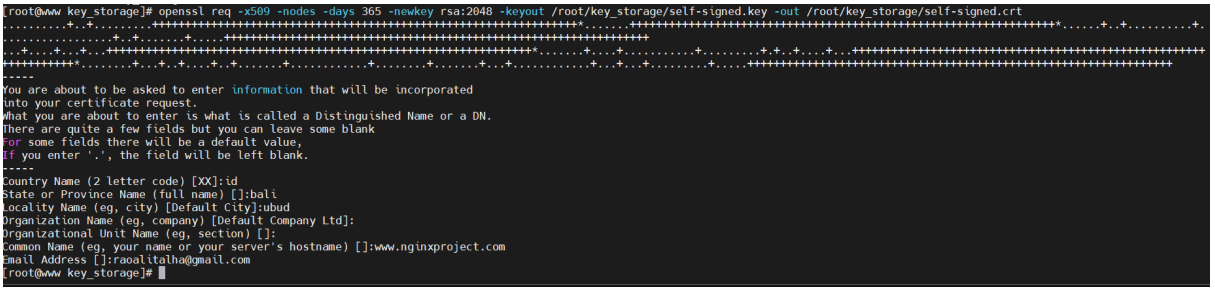


Hosting Another Website On same Server

Step 10 (Creating Self signed SSL/TLS certificate):

- Now we will host another website on the same server.
- Since we are going to use the `local machine` to host website with `SSL/TLS certificate` . And we do not have own website for which we have bought `domain` . So , we have to use `self-signed certificate` for testing.
- First of all we will create a directory named `key_storage` at `cd /root/key_storage`
- To generate a self key we will run the command:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /
```



Few question will be asked answer them and the key will be generated.

- Now we have to move the `key` and `certificate` generated .Move the `self-signed.key` to the directory `/etc/pki/tls/private/` and `self-signed.cert` to `/etc/pki/tls/cert/` .

```
mv self-signed.key etc/pki/tls/private/

mv self-signed.crt /etc/pki/tls/cert/
```

Step 11 (Making changes to the configuration file and creating new files)

- To enable `https` and `SLL/TLS` certification we need to first make some changes in the original `configuration` file of the `nginx` . Open the `nginx.conf` file

```
vi /etc/nginx/nginx.conf
```

- Scroll down and find the `# Settings for a TLS enabled server.` If the setting are `commented out` remove the `#` from that like in the screenshot below.

```
# Settings for a TLS enabled server.
#
    server {
        listen       443 ssl http2;
        listen       [::]:443 ssl http2;
        server_name  _;
        root         /usr/share/nginx/html;

        ssl_certificate "/etc/pki/tls/certs/self-signed.crt";
        ssl_certificate_key "/etc/pki/tls/private/self-signed.key";
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_ciphers PROFILE=SYSTEM;
        ssl_prefer_server_ciphers on;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        error_page 404 /404.html;
            location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
        }
    }

}
```

- Our second website which we are going to host is a `cars website` . We will create a new file name name `buycars.conf` in the directory of :

```
cd /etc/nginx/conf.d/

touch buycars.conf
```

- Open the file in `vi editor` to edit:

```
# HTTP server configuration for www.buycars.com and buycars.c
server {
    listen 80;
    listen [::]:80;
    server_name www.buycars.com buycars.com;

    # Redirect all HTTP requests to HTTPS
    return 301 https://$host$request_uri;
}

# HTTPS server configuration for www.buycars.com and buycars.
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name www.buycars.com buycars.com;

    # Path to the self-signed SSL certificate and key
    ssl_certificate /etc/pki/tls/certs/self-signed.crt;
    ssl_certificate_key /etc/pki/tls/private/self-signed.key;

    # Additional SSL settings
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHAC

    root /var/www/buycars;
    index index.html index.php index.htm;
    location / {
        try_files $uri $uri/ =404;
    }

    # Error pages
    error_page 404 /404.html;
    location = /404.html {
        internal;
    }

    error_page 500 502 503 504 /50x.html;
```

```
    location = /50x.html {
        internal;
    }
}
```

- As we are now configuring our sites on `https` we also have to make changes to the configuration files of our `first website` which was `www.nginxproject.com`
- Open the file in vi editor `vi /etc/nginx/conf.d/petshop.conf` and make changes the file text will be:

```
# HTTP server configuration for nginxproject.local and www.ng
server {
    listen 80;
    listen [::]:80;
    server_name nginxproject.local www.nginxproject.com;

    # Redirect all HTTP requests to HTTPS
    return 301 https://$host$request_uri;
}

# HTTPS server configuration for nginxproject.local and www.n
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name nginxproject.local www.nginxproject.com;

    # Path to the self-signed SSL certificate and key
    ssl_certificate /etc/pki/tls/certs/self-signed.crt;
    ssl_certificate_key /etc/pki/tls/private/self-signed.key;

    # Additional SSL settings
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHAC

root /var/www/petshop;
```

```
    index index.html index.htm;

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

- Save the file and after that run the `nginx` test by running the command

```
nginx -t
```

Step 12 (Copying the website code to the directory)

- We have not created the directory to put our code in `cd /var/www/` . In our `configuration file` for `buycars` we have mention the website location `cd /var/www/buycars` .
- So create a directory in `/var/www/` with the name of `buycars`

```
mkdir -p buycars
```

```
[root@www conf.d]# cd /var/www/
[root@www www]# ls
petshop
[root@www www]# mkdir -p buyfurniture
[root@www www]# ls
buyfurniture  petshop
[root@www www]#
```

- Open the directory `buy cars` and download the website code using command

```
wget https://www.free-css.com/assets/files/free-css-templates
```

- Unzip the file and copy the entire files to `/var/www/buycars/` and delete other files which no longer needed.

Step 13 (Adding the url into host file)

- Open the file in directory `/etc/hosts` use the command :

```
vim /etc/hosts
```

```
[root@www nginx]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.1.10 www.nginxproject.com
192.168.1.10 www.buycars.com buycars.com

[root@www nginx]#
```

add `192.168.1.10 www.buycars.com`

Save the file

Step 14 (Final Checking)

- Run the command to check the status of `nginx`

```
nginx -t
```

```
[root@www buycars]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@www buycars]#
```

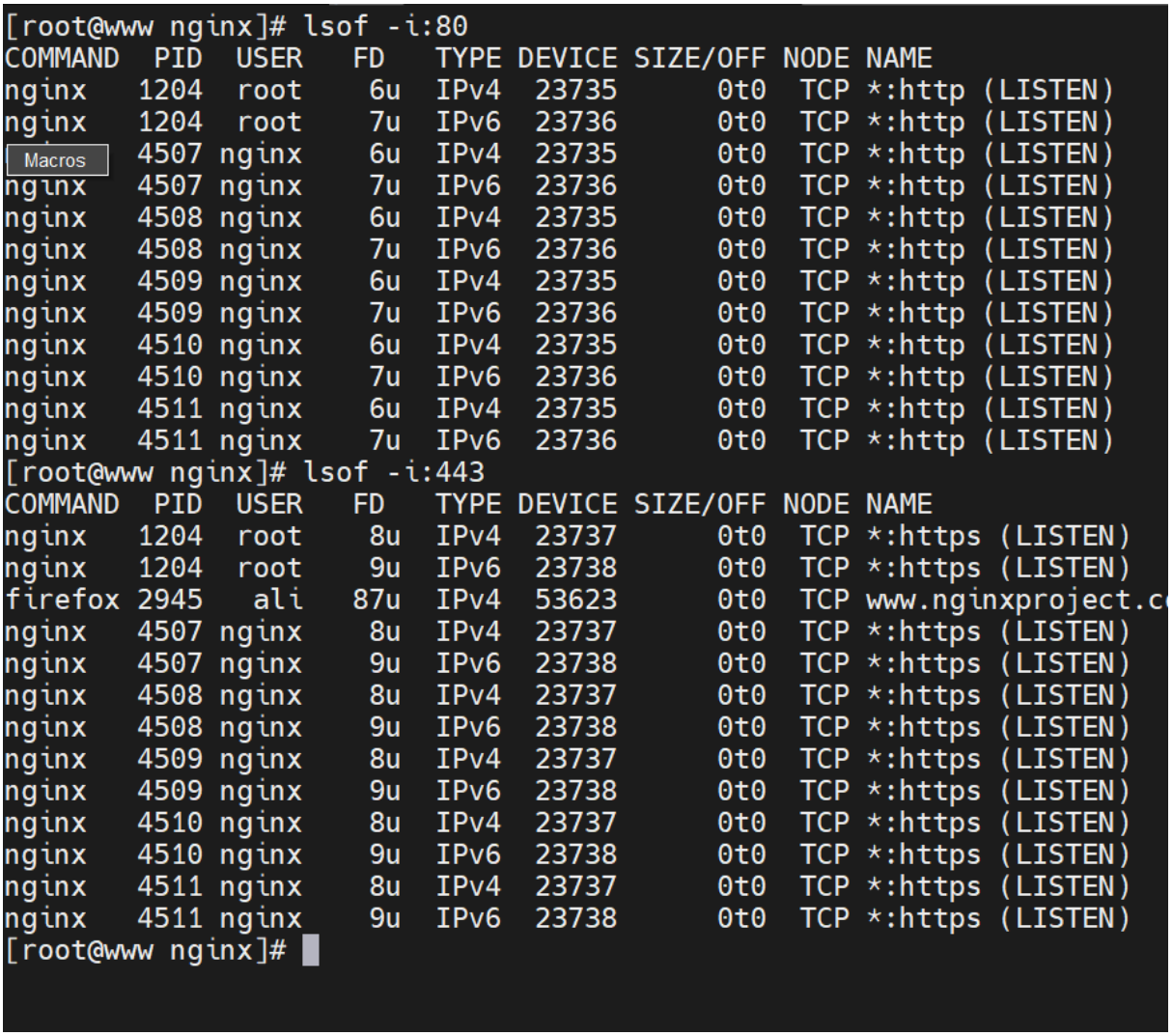
- Reload the `nginx server`

```
systemctl reload nginx
```

- Check the list of `ports enable`

```
lsof -i :80

lsof -i:443
```

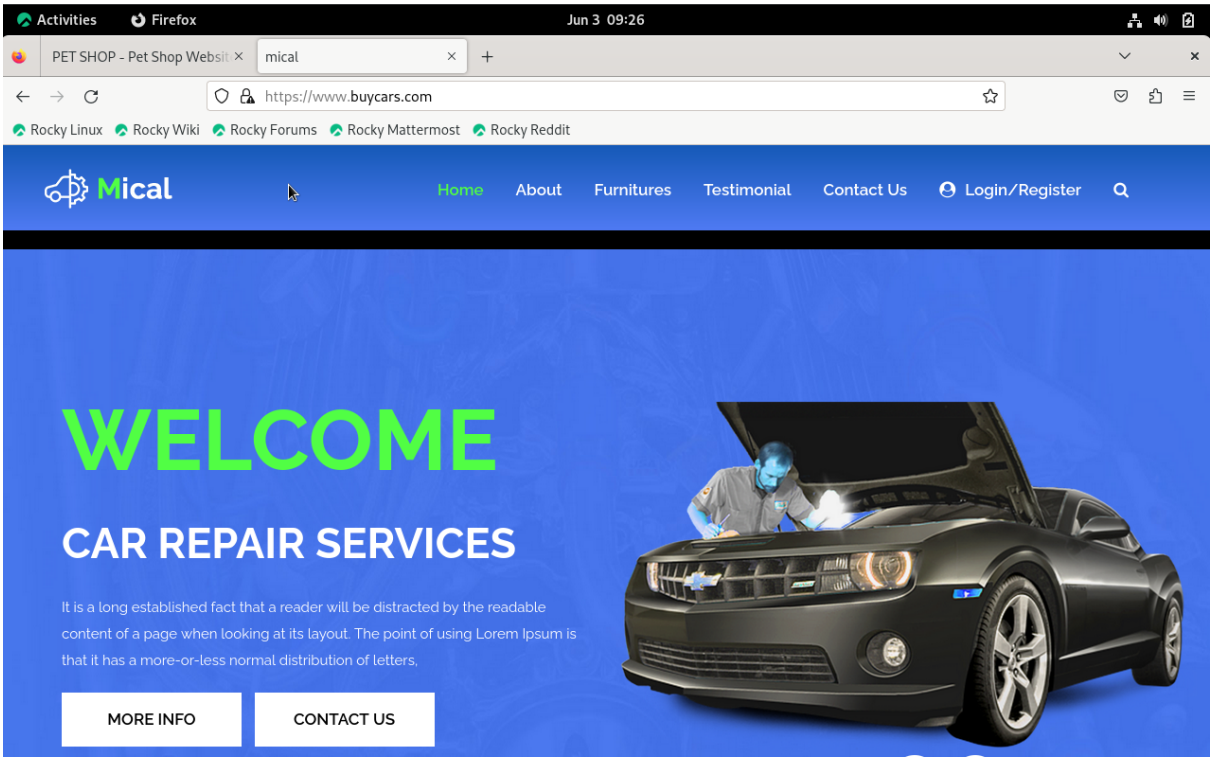


Step 15 (Testing On web Browser)

- open the `firefox browser` on you Linux machine and type the address:

```
www.nginxproject.com

www.buycars.com
```



- Both websites are working fine

Step 16 (Creating Individual LOG Directories for Individual Website)

- We know that by default the log files of the `nginx` is located in:

```
cd /var/log/nginx
```

```
[root@www nginx]# ls -ltr
total 144
-rw-r--r--. 1 root  root  7797 Jun  2 19:58 error.log-20240603
-rw-r--r--. 1 root  root 90563 Jun  2 19:58 access.log-20240603
-rw-r-----. 1 nginx root 32758 Jun  3 08:59 access.log
-rw-r-----. 1 nginx root  8884 Jun  3 09:36 error.log
[root@www nginx]#
```

- All the upstream websites information and error are placed in these logs. But it is a single file for all the websites. We can also create `individual log`

`files` for `each website` and the use the `default log files` as a `consolidated` file for all the track record.

- To configure independent log we have to `change the configuration file` for each website we hosted in `/etc/nginx/conf.d` . First we will configure the `petshop website`

```
vi /etc/nginx/conf.d/petshop.conf
```

In the server block add the following line

```
    access_log /var/log/nginx/nginxproject.local.access.log;
    error_log /var/log/nginx/nginxproject.localnginxproject.l
```

```
# HTTP server configuration for nginxproject.local and www.nginxproject.com
server {
    listen 80;
    listen [::]:80;
    server_name nginxproject.local www.nginxproject.com;

    # Redirect all HTTP requests to HTTPS
    return 301 https://$host$request_uri;


    access_log /var/log/nginx/nginxproject.local.access.log;
    error_log /var/log/nginx/nginxproject.localnginxproject.log;
}

# HTTPS server configuration for nginxproject.local and www.nginxproject.com
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name nginxproject.local www.nginxproject.com;

    # Path to the self-signed SSL certificate and key
    ssl_certificate /etc/pki/tls/certs/self-signed.crt;
    ssl_certificate_key /etc/pki/tls/private/self-signed.key;
```

- Restart the `nginx` using the command `nginx -t`
- And go to the directory `/var/log/nginx/` you will see new log files will be created all the information for the website individually updated here

```
[root@www nginx]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@www nginx]# ls -ltr
total 144
-rw-r--r--. 1 root  root  7797 Jun  2 19:58 error.log-20240603
-rw-r--r--. 1 root  root 90563 Jun  2 19:58 access.log-20240603
-rw-r-----. 1 nginx root 32758 Jun  3 08:59 access.log
-rw-r-----. 1 nginx root  8884 Jun  3 09:36 error.log
-rw-r--r--. 1 root  root     0 Jun  3 09:49 nginxproject.localnginxproject.log
-rw-r--r--. 1 root  root     0 Jun  3 09:49 nginxproject.local.access.log
[root@www nginx]#
```

- Do the same for the other website if you wish to