NGINX

What is NGINX?
NGINX is a versatile web server used for multiple tasks in application web hosting. It is known for its high performance, scalability, and efficiency. Below are its key functionalities:

Web Server

- Efficiently serves static content (e.g., HTML, images, CSS, and JavaScript).

Gateway

1. **Reverse Proxy**
    - Routes incoming client requests to backend servers (e.g., Node.js, Python, or other services).
2. **Caching**
    - Speeds up content delivery by storing frequently accessed data locally to reduce load on backend servers.
3. **Rate Limiting**
    - Controls the number of requests to prevent abuse or excessive traffic.
4. **Static Site Hosting**
    - Hosts static websites directly from the server.
5. **Multi-Site Hosting**
    - Supports hosting multiple websites on a single server using virtual hosts.

TLS Termination

- Manages secure HTTPS connections by handling encryption and decryption, reducing the load on backend servers.

Load Balancer

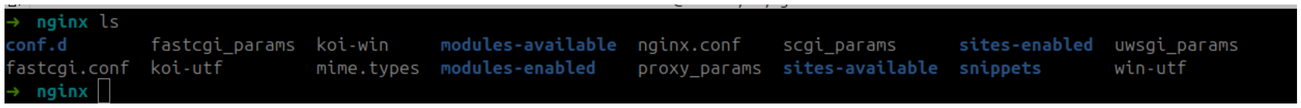
- Distributes incoming traffic across multiple servers using load-balancing algorithms such as:
    - Round-robin
    - Least connections
    - IP hash

Install Nginx

To install NGINX on a Linux system, you can simply use the following command:
bash

```
sudo apt install nginx -y
```

Once the installation is complete, you can find the NGINX configuration files in the /etc/nginx directory. This directory contains several files, and we will review the important ones and their purposes.

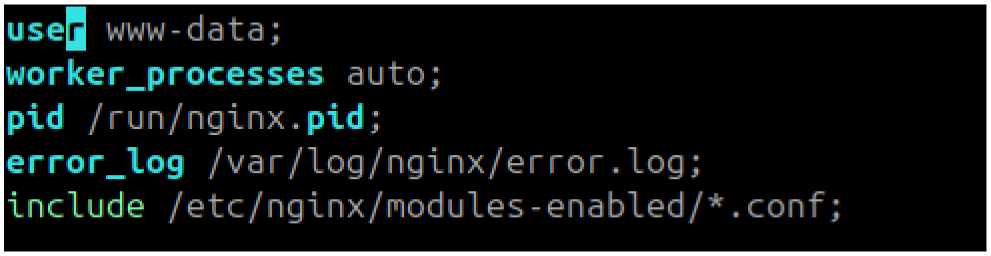


```
→ nginx ls
conf.d        fastcgi_params  koi-win     modules-available  nginx.conf    scgi_params      sites-enabled  uwsgi_params
fastcgi.conf  koi-utf         mime.types  modules-enabled    proxy_params  sites-available  snippets       win-utf
→ nginx
```

Nginx.conf

This is an important file where NGINX stores its main configuration details. You can modify this file to customize settings based on your requirements, such as server configurations, location blocks, and other critical directives.

Open this location, sudo vim /etc/nginx.conf



```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
error_log /var/log/nginx/error.log;
include /etc/nginx/modules-enabled/*.conf;
```

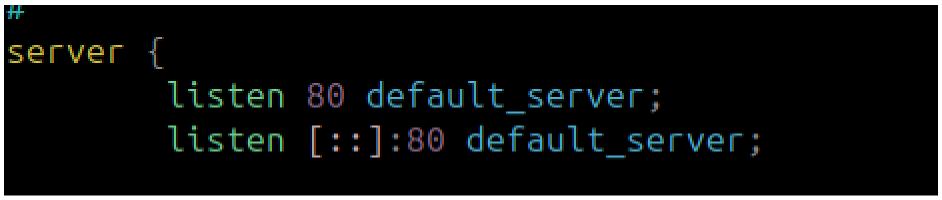
In the `/etc/nginx` directory, NGINX provides key configuration settings, including:

1. **`worker_processes`**
   - Specifies the number of worker processes that NGINX can run simultaneously.
   - It's usually set based on the number of CPU cores for optimal performance.

2. **`pid`**
   - Indicates the location of the file where the master process's PID (Process ID) is stored.
   - This file is used to manage the NGINX process, such as stopping or restarting it.
3. **`error_log`**
   - Specifies the file path for storing error logs.
   - Includes details on what types of errors to log, such as `warn`, `error`, or `crit` (critical errors).

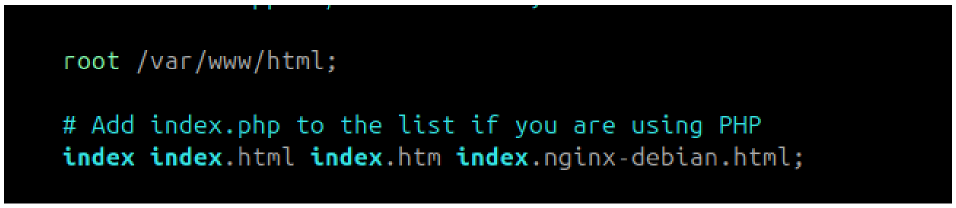
Sites-Enabled

To check and modify server-related information, navigate to /etc/nginx/sites-available/default. This file contains various directives, such as server and location, which can be customized to meet your requirements.



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

As we can see, the server directive specifies the port on which the server will listen for requests. The first 80 is for IPv4 addresses, while the second [::]:80 is for IPv6 addresses.



```
    root /var/www/html;

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

This is another directive called root, which tells the server where to fetch the index files and specifies their names. This location is used to serve the index files of a web server.

In Ubuntu, NGINX provides two locations for server configuration:

1. `/etc/nginx/sites-enabled`
   - This is the traditional location where server configurations are linked for specific websites or virtual hosts.
2. `/etc/nginx/conf.d`
   - This directory is intended for storing all server configurations and is considered the best practice by the NGINX documentation.
   - Keeping server configurations in `conf.d` helps maintain a cleaner and more organized structure, as it allows for better separation of server settings, especially in larger setups.

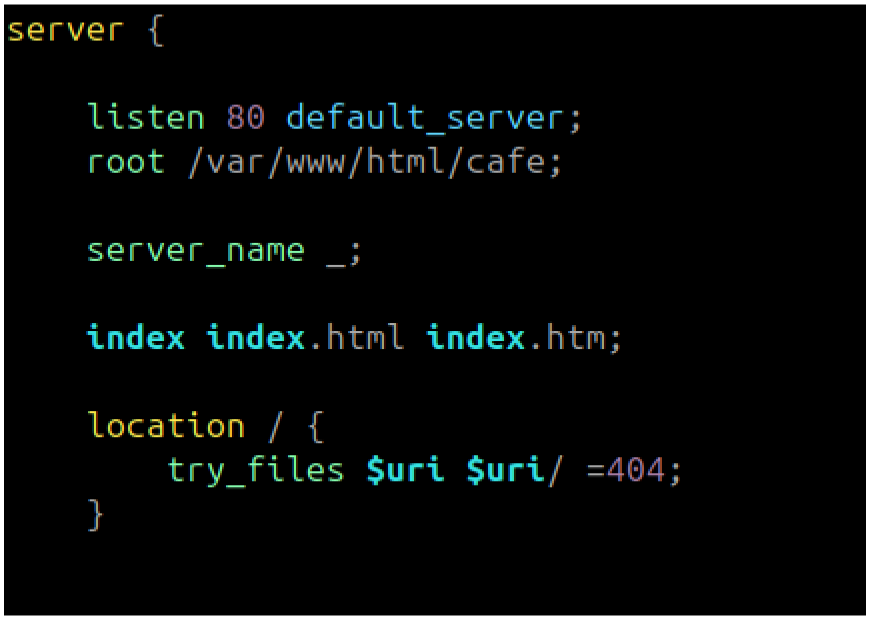
The recommended practice, according to the NGINX documentation, is to use `conf.d` for server configurations to ensure better organization and readability.

Creating Web Server

Now let us create virtual server which is use to serve static websites through nginx.

As we are going to make changes to the conf.d folder,

1. Open the conf.d folder and create one file with your choice name I give a name to the file as a cafe.conf, and the file name should end with.conf always and now make server configuration as shown below.



```
server {

    listen 80 default_server;
    root /var/www/html/cafe;

    server_name _;

    index index.html index.htm;

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

🌐 How to Serve Static Content for a Website

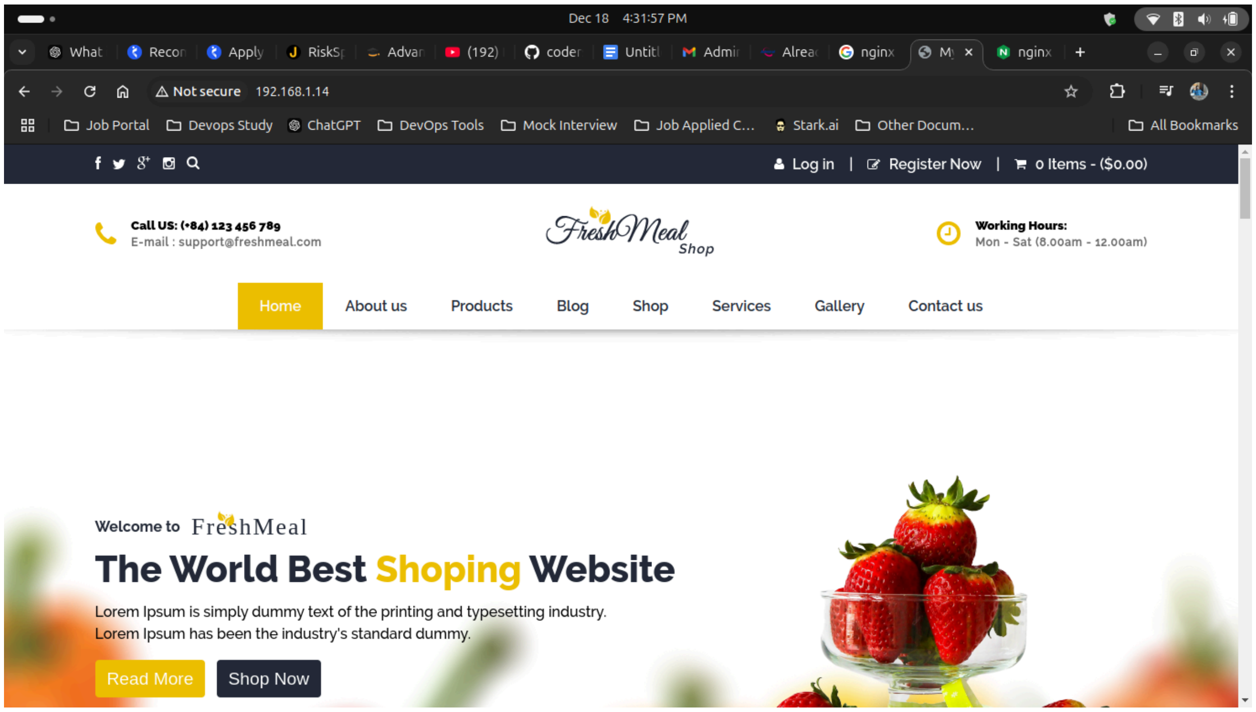
Want to host a static website easily? Follow these steps:

1. Clone the Repository

You can get started by cloning this repository, which contains all the required static files:
👉 https://github.com/AmittAshok/Cafe-website.git .

Once all configurations are complete, test the Nginx setup using the `nginx -t` command, and then reload the service with `systemctl reload nginx`.

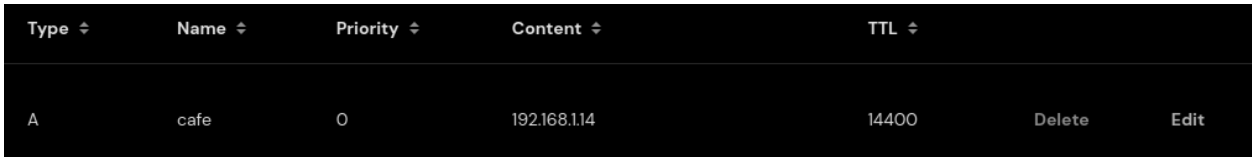
You can see the static website is running on a local IP address.


Not secure 192.168.1.14
Log in | Register Now | 0 Items - ($0.00)
Call US: (+84) 123 456 789
E-mail : support@freshmeal.com
FreshMeal Shop
Working Hours:
Mon - Sat (8.00am - 12.00am)
Home
About us
Products
Blog
Shop
Services
Gallery
Contact us
Welcome to FreshMeal
The World Best Shoping Website
Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy.
Read More
Shop Now

Now let’s associate a domain name with the website.

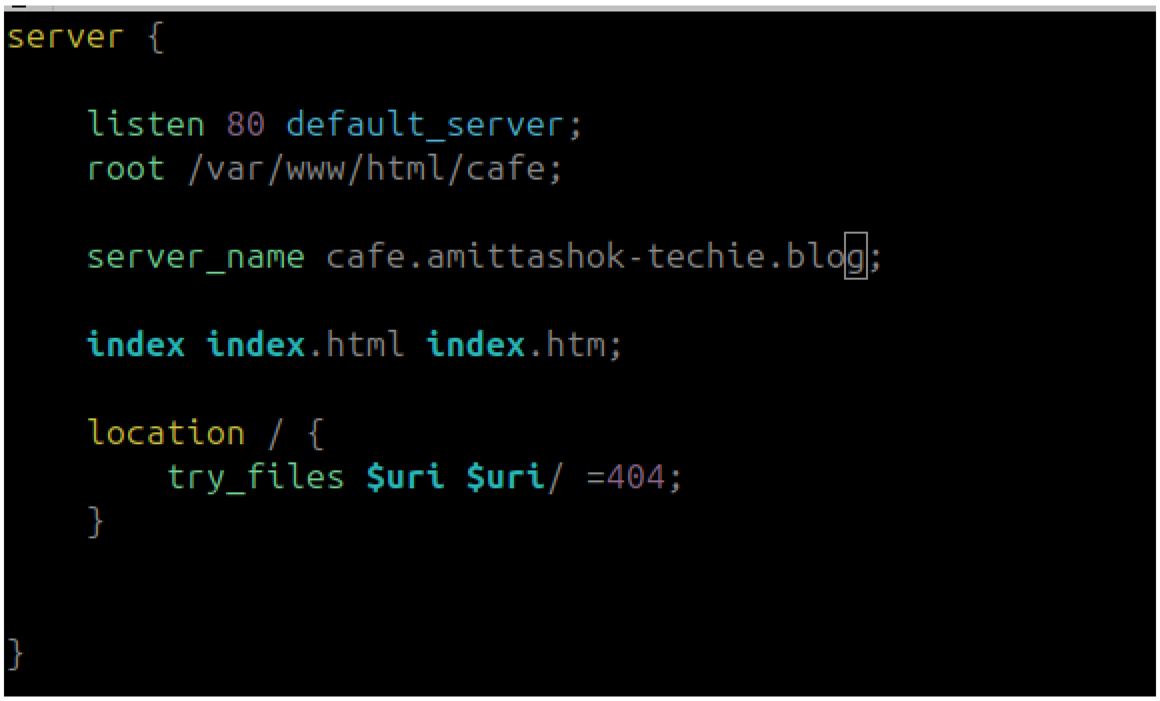
Currently, our website is being served from the localhost IP address, but we want to use a custom domain name. To achieve this:

<u>Step 1</u>: Register your domain with a domain registrar.
I have registered my domain through Hostinger with the name `amittashok-techie.blog`. I then added a new DNS record for my café website as `cafe.amittashok-techie.blog` using an A record.



Type	Name	Priority	Content	TTL		
A	cafe	0	192.168.1.14	14400	Delete	Edit

<u>Step 2:</u> Now, open the main configuration file located in the `conf.d` folder where we created the `cafe.conf` file. Add your domain name in the `server_name` directive within the configuration.
For example:



```
server {

    listen 80 default_server;
    root /var/www/html/cafe;

    server_name cafe.amittashok-techie.blog;

    index index.html index.htm;

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


}
```

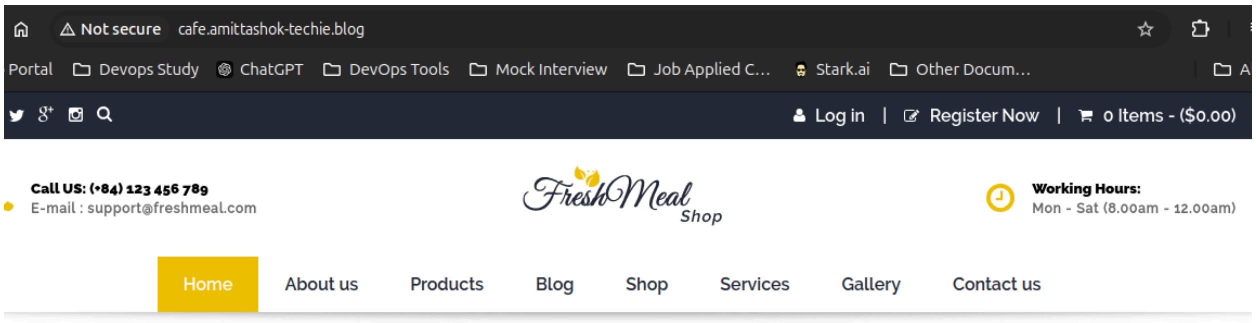
<u>Step 3</u> Once you have updated the `cafe.conf` file with the correct configuration:

1. Save the File
   Save the changes to the `cafe.conf` file in the `conf.d` folder.

   Reload the NGINX Server
   Run the following command to reload the NGINX configuration:

```
sudo systemctl reload nginx
```

2. Test the Configuration
   Open a web browser and enter your domain name (e.g., `http://cafe.amittashok-techie.blog`).
   - If everything is configured correctly, your website should load as expected.



Thank You….