**A. Deploy a website on localhost using either apache2 or Nginx. Create a DNS name for this website as ‘awesomeweb’. You can use any web template you want or can write your own simple HTML code.**

**Step 1**: sudo systemctl start nginx

sudo systemctl enable nginx

**Create a directory for the new website**:

sudo mkdir -p /var/www/awesomeweb

**Set the correct permissions**:

sudo chmod 777 /var/www/awesomeweb

**Step: 2 - Create an index.html file** in the website directory:

You can either write your own HTML file or use this simple template.

nano /var/www/awesomeweb/index.html

Add **the** following HTML content:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Awesome Web</title>

</head>

<body>

<h1>Welcome to AwesomeWeb!</h1>

<p>This is a simple Nginx-hosted website on localhost with a custom DNS name.</p>

</body>

</html>

**Configure Nginx to Serve Your Website –**

**Step 3: Create a new server block configuration**:

Nginx stores configuration files in /etc/nginx/sites-available. We'll create a new configuration file for awesomeweb.

sudo nano /etc/nginx/sites-available/awesomeweb

**Add the following configuration to the file:**

server {

listen 80;

listen [::]:80;

server\_name awesomeweb;

root /var/www/awesomeweb;

index index.html;

location / {

try\_files $uri $uri/ =404;

}

}

**Enable the new configuration**:

sudo ln -s /etc/nginx/sites-available/awesomeweb /etc/nginx/sites-enabled/

**Test the Nginx configuration**:

sudo nginx -t

Reload Nginx

sudo systemctl reload nginx

**Step 4: Configure Local DNS (Set up Custom DNS Name)**

**Edit the /etc/hosts file**:

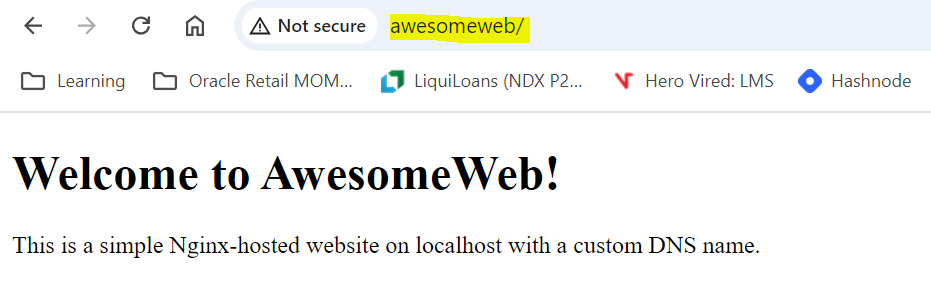
sudo nano /etc/hosts

Append the following line to map awesomeweb to localhost:

127.0.0.1 awesomeweb

**Step 5: Access the Website**

http://awesomeweb



**B.  A website can have many subdomains and different services are running on them. Write a Python script to check the status of the subdomains which are up or down. The script should automatically check the status every minute and should update it in tabular format on the screen.**

**Write documentation of it.**

**Requirements**

Before running the script, ensure you have Python installed on your machine. You will also need to install the **requests** and **prettytable** libraries. You can do this using pip:

pip install requests prettytable

Configuration-

**Subdomains List**:

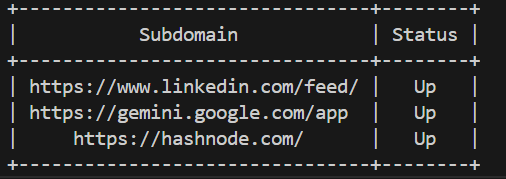
* Modify the SUBDOMAINS list at the beginning of the script to include the subdomains you want to monitor. You can add or remove subdomains as needed.
* SUBDOMAINS = [
* "example.com",
* "sub1.example.com",
* "sub2.example.com",

**How It Works**

1. **Check Status Function**:
   * The check\_subdomain\_status(subdomain) function makes an HTTP GET request to each subdomain. If the request is successful (HTTP status code 200), it returns the subdomain status as 'Up'. If there's a connection error or a timeout, it returns 'Down'.
2. **Display Function**:
   * The display\_status\_table(statuses) function uses the PrettyTable class to create a formatted table of subdomain statuses and clears the console before displaying it.
3. **Main Loop**:
   * The main() function runs an infinite loop that checks the status of all subdomains every minute (60 seconds). It updates the displayed status table each time.

**Running the Script**

1. Save the script to a file named subdomain\_checker.py.
2. Open a terminal or command prompt and navigate to the directory where the script is saved.
3. Run the script using the following command: python xyz.py



**C. Once the VM has been installed, visit**[**https://www.osboxes.org/**](https://www.osboxes.org/)**download a Ubuntu 22.04 image and start it through your VirtualBox.**

**Install Nginx inside the Ubuntu machine and host a website.**

**Come back to your host machine (windows/Linux/mac) and scan the virtual machine using Nmap. Create the documentation of the process and the output of the scan. Observe the ports which are open.**

**Phase 1: Installing Nginx and Hosting a Website in the Ubuntu Virtual Machine**

**1. Set Up a Virtual Machine**

* Launch your Ubuntu VM in a virtual environment such as VirtualBox, VMWare, or any cloud VM service.
* Ensure you have network connectivity between the host and virtual machine (use a Bridged or NAT Adapter with Port Forwarding).

**2. Update Package Repositories**

Run the following commands in your Ubuntu VM terminal to update the package lists:

bash

Copy code

sudo apt update

sudo apt upgrade -y

**3. Install Nginx**

To install Nginx on the Ubuntu machine, run:

bash

Copy code

sudo apt install nginx -y

**4. Start and Enable Nginx**

After installation, start and enable the Nginx service:

bash

Copy code

sudo systemctl start nginx

sudo systemctl enable nginx

**5. Verify Nginx Status**

Ensure that Nginx is running:

bash

Copy code

sudo systemctl status nginx

**6. Configure the Firewall**

If you have a firewall (such as UFW), allow traffic on HTTP and HTTPS ports (80 and 443):

bash

Copy code

sudo ufw allow 'Nginx Full'

**7. Create a Basic Web Page**

You can modify the default Nginx web page or create a new one. Let’s create a simple HTML file:

bash

Copy code

sudo nano /var/www/html/index.html

Add the following content:

html

Copy code

<!DOCTYPE html>

<html>

<head>

<title>My Nginx Website</title>

</head>

<body>

<h1>Hello, welcome to my website hosted on Nginx!</h1>

</body>

</html>

Save and exit (CTRL+X, Y, then Enter).

**8. Test the Web Page**

In your Ubuntu machine, you can verify the hosted web page by opening the browser and navigating to:

arduino

Copy code

http://localhost

If everything is set up correctly, you should see your "Hello, welcome to my website" message.

You can also verify by checking your machine's IP address:

bash

Copy code

hostname -I

Use this IP address on the host machine’s browser:

arduino

Copy code

http://<vm\_ip\_address>

**Phase 2: Scan the Virtual Machine from the Host Machine Using Nmap**

**1. Install Nmap on the Host Machine**

If you don't have Nmap installed on your host machine, you can install it as follows:

**For Windows:**

* Download the Nmap installer from https://nmap.org/download.html and follow the installation steps.

**For Linux (Debian-based):**

bash

Copy code

sudo apt install nmap

**For macOS:** You can install it using Homebrew:

bash

Copy code

brew install nmap

**2. Scan the Virtual Machine**

Once Nmap is installed, use the following command to scan the virtual machine from your host machine:

bash

Copy code

nmap <vm\_ip\_address>

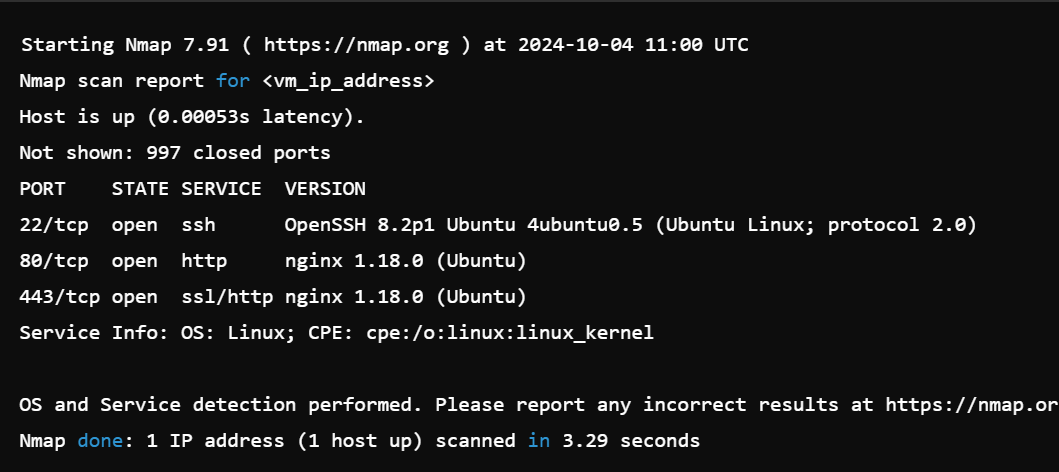
For a more detailed scan, you can run:

bash

Copy code

nmap -A <vm\_ip\_address>

The -A option enables OS detection, version detection, script scanning, and traceroute.



**Explanation of the Nmap Output:**

1. **Host is up:** This shows the target machine is reachable.
2. **Ports:**
   * **22/tcp (ssh):** The port is used for Secure Shell (SSH). It is open, meaning that the VM allows remote connections.
   * **80/tcp (http):** This port is used for HTTP traffic. Since Nginx was installed and is running, this port is open, hosting the website.
   * **443/tcp (https):** This port is used for HTTPS traffic. Nginx is also listening on this port for secure connections.
3. **Service Information:**
   * The report provides information about the services running on the detected ports, including the version of Nginx and OpenSSH.