PROJECT

TITLE: SETTING UP A PERSONAL WEB SERVER.

CONTENT:

- Introduction
- Objective
- Prerequisites
- Step-by-Step Guide
- Troubleshooting
- Additional Features
- Conclusion
- References and Resources

INTRODUCTION:

Install and configure Apache or Nginx on a Linux machine to host a simple website or web application. Learn about basic web server configuration, file permissions, and serving static content. A personal web server allows you to host your own website, store files, and manage various online services from your own hardware. Setting up a personal web server is a great way to learn about web hosting, server management, and web development. This guide will walk you through the steps to set up a personal web server, covering the essential software, configurations, and security measures.

By the end of this project, you will have a fully operational web server that you can customize and expand according to your needs, providing you with a robust platform for learning, experimentation, and online publishing. Whether your goal is to run a personal blog, host a portfolio, or develop web applications, this project will empower you to take control of your online presence with confidence.

OBJECTIVES:

The objective of the project "Setting Up a Personal Web Server" is to equip users with the knowledge and skills necessary to successfully install, configure, and manage a personal web server using software such as Apache or Nginx. This project aims to provide hands-on experience in setting up essential components, including static IP addresses, domain name settings, and virtual hosts, while also teaching users how to install a database management system and server-side scripting language. Additionally, the project emphasizes the importance of implementing basic security measures, such as firewalls and SSL/TLS certificates, ensuring a secure and functional web hosting environment.

Describe how to configure the web server software, static IP addresses, domain name settings, and virtual hosts, among other necessary components. Show users how to set up and maintain the required software packages, such as a server-side programming language (PHP, Python) and a database management system (e.g., MySQL). Assist the user in setting up a web server (like Apache or Nginx) on a desktop or server in order to host a website or web application.

PREREQUISITES:

- **Hardware Requirements:** Specify the minimum hardware needed (e.g., 4GB RAM, stable internet connection, modern processor).
- **Software Requirements:** List essential software (Linux/Windows Server OS, Apache/Nginx web server, MySQL database, PHP).
- **Basic Knowledge:** Outline the knowledge required, such as familiarity with command-line interfaces, networking, and web technologies.
- **Internet Connection:** A stable and reasonably fast internet connection is crucial for serving web pages to users globally. If hosting from home, consider the upload speed of your connection, as it will affect website performance.
- **Domain Name:** If you plan to make your server accessible over the internet, acquiring a domain name and configuring DNS settings will enhance accessibility and professionalism.

STEP-BY-STEP GUIDE:

1. **Choose an Operating System:**

   - Explain the options available, such as Linux distributions (Ubuntu, Debian) or Windows Server.

2. **Install Apache Web Server:**

   - By default, the Apache package is included in the Fedora 34 default repo. You can install it with the following command.

```
dnf update -y
```

```
dnf install httpd -y
```

   - After the successful installation, you can see detailed information about Apache with the following command.

```
rpm -qi httpd
```

- You will get the following output.

```
Name        : httpd
Version     : 2.4.53
Release     : 1.fc34
Architecture: x86_64
Install Date: Thursday 13 April 2023 10:58:31 PM
Group       : Unspecified
Size        : 4932592
License     : ASL 2.0
Signature   : RSA/SHA256, Thursday 17 March 2022 01:00:49 PM, Key ID 1
Source RPM  : httpd-2.4.53-1.fc34.src.rpm
Build Date  : Thursday 17 March 2022 12:43:53 PM
Build Host  : buildvm-x86-27.iad2.fedoraproject.org
Packager    : Fedora Project
Vendor      : Fedora Project
URL         : https://httpd.apache.org/
Bug URL     : https://bugz.fedoraproject.org/httpd
Summary     : Apache HTTP Server
Description :
The Apache HTTP Server is a powerful, efficient, and extensible
web server.
```

3. Manage Apache Service:

- By default, the Apache service is managed by systemd. You can easily start, stop and enable the Apache service via systemctl command.

- To start the Apache service, run the following command.

```
systemctl start httpd
```

- To enable the Apache service, run the following command.

```
systemctl enable httpd
```

- To check the status of the Apache service, run the following command.

```
systemctl status httpd
```

- You will get the following output.

```
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled;
     Active: active (running) since Thu 2023-04-13 22:58:51 EDT; 4s ag
       Docs: man:httpd.service(8)
   Main PID: 1539 (httpd)
     Status: "Started, listening on: port 80"
      Tasks: 177 (limit: 4666)
     Memory: 14.2M
        CPU: 99ms
     CGroup: /system.slice/httpd.service
             ├─1539 /usr/sbin/httpd -DFOREGROUND
             ├─1540 /usr/sbin/httpd -DFOREGROUND
             ├─1541 /usr/sbin/httpd -DFOREGROUND
             ├─1542 /usr/sbin/httpd -DFOREGROUND
             └─1543 /usr/sbin/httpd -DFOREGROUND

Apr 13 22:58:51 fedora systemd[1]: Starting The Apache HTTP Server...
```

- To stop the Apache service, run the following command.

```
systemctl stop httpd
```

4. **Access the Apache Test Page:**

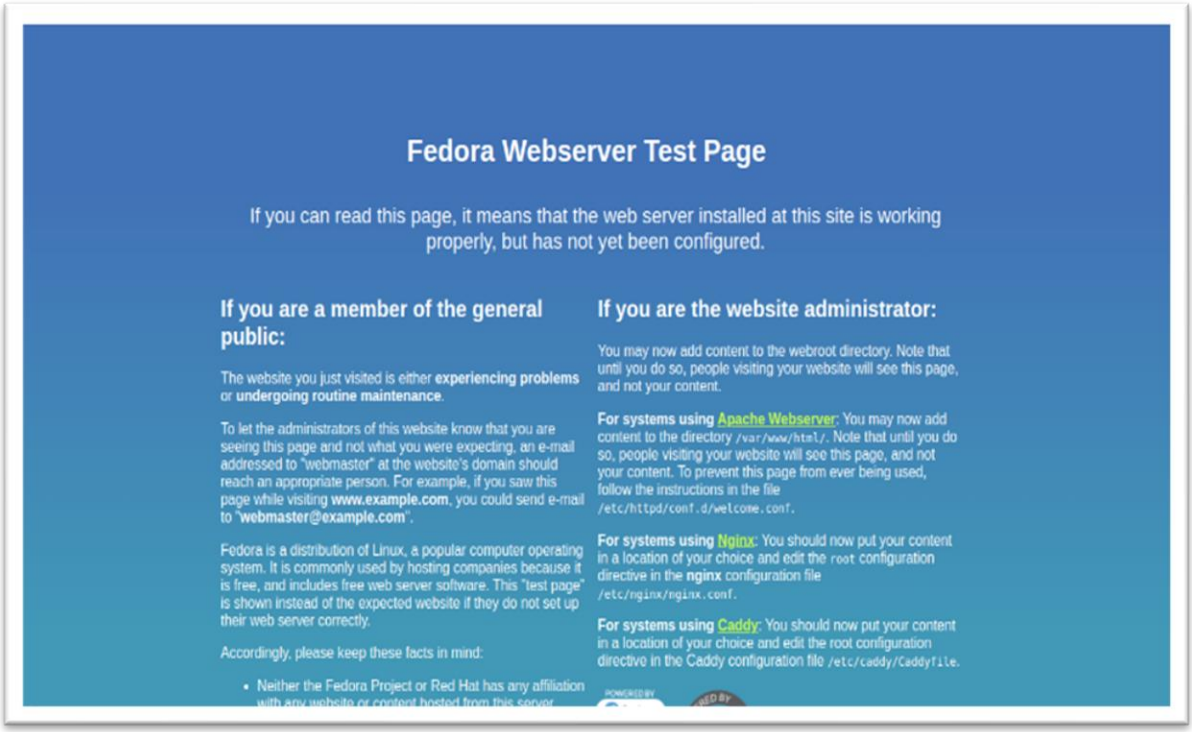
- At this point, the Apache web server is started and listening on port 80. You can verify it with the following command**.**

```
ss -antpl | grep :80
```

- You should see the Apache listening port in the following output.

```
LISTEN 0      511                *:80              *:*    users:(("htt
```

- Now, open your web browser and access the Apache test page using the URL **http://your-server-ip.**



5. Create and Host a Simple Website with Apache:

- First, create an Apache website directory using the following command.

```
mkdir /var/www/html/website
```

- Next, create an index.html file:

```
nano /var/www/html/website/index.html
```

- Add the following content:

```
<html>
  <head>
    <h2>Welcome to Apache Web Server</title>
  </h2>
  <body>
    <h3>Success! Apache server on Fedora is working!</h3>
  </body>
</html>
```

- Next, set the permissions and ownership to the Apache website directory.

```
chown -R apache:apache /var/www/html/website
chmod -R 755 /var/www/html/website
```

- Next, create an Apache virtual host configuration file.

```
nano /etc/httpd/conf.d/web.conf
```

- Add the following configuration:

```
<VirtualHost *:80>
    ServerName web.example.com
    DocumentRoot /var/www/html/website
    DirectoryIndex index.html
    ErrorLog /var/log/httpd/example.com_error.log
    CustomLog /var/log/httpd/example.com_requests.log combined
</VirtualHost>
```

- Save and close the file, then restart the Apache service to apply the changes.

```
systemctl restart httpd
```

6. **Access Apache Website:**

At this point, the Apache web server is configured to serve HTML websites. You can now access it using the URL **http://web.example.com.**

You should see your website on the following screen.

TROUBLESHOOTING:

- **Common Issues and Solutions:** Address potential problems users may encounter, such as server not responding, permission errors, or security configuration issues.
- **Debugging Techniques:** Provide tips for diagnosing server issues using log files and command-line tools.

ADDITIONAL FEATURES:

- **Setting Up FTP/SFTP Access:** Explain how to configure file transfer protocols for uploading website content.
- **Hosting Multiple Websites:** Guide on configuring multiple domains on the same server using virtual hosts.
- **Automating Backups:** Discuss setting up automated scripts for server backups.

CONCLUSION:

In conclusion, setting up a personal web server provides a valuable learning experience that enables users to host websites, run web applications, and manage online services with full control over their data. This project equips you with the skills to configure essential components, such as web server software, databases, and security measures, while offering hands-on experience in server management and networking. By following the steps outlined, you will have a secure and functional web server that serves as a robust platform for experimentation, learning, or personal projects, empowering you to take ownership of your online presence and explore further possibilities in web development and hosting.

It goes beyond just installing software this project enables you to dive into critical aspects such as server configuration, network management, and data security, which are essential skills in today's digital world. By learning to troubleshoot common issues and implement best practices for securing your server, you gain a practical skill set that extends beyond hosting a website, opening doors to opportunities in web development, system administration, and IT management.

REFERENCES AND RESOURCES:

- **Documentation Links:** Provide links to official documentation for software used (e.g., Apache, Nginx, MySQL).

- **Learning Platforms:** Recommend online courses or tutorials for deeper understanding (e.g., Coursera, Udemy, edX and many more).

- **Books:** Recommend many books like “The Linux Command Line" by William Shotts, "Linux Server Administration" by Craig Hunt, "Learning PHP, MySQL & JavaScript" by Robin Nixon and many more.

- **YouTube Channels:** Recommend many You Tube Channels like “Tech with Tim”, “The Net Ninja”, “Network Chuck”.