Setting Up and Deploying a Basic Web Page on Debian

Introduction

This worksheet will guide you through several steps in configuring a Debian server, installing a web server, setting up a user account, and deploying a static website. Here's a quick overview of the tools and concepts you'll work with:

- **Linux**: An open-source, Unix-like operating system widely used for servers, desktops, and embedded systems. Known for its stability, security, and flexibility, Linux powers most of the world's servers and cloud infrastructure.
- **Debian Bookworm**: Debian is a popular Linux distribution known for its stability and community-driven development. "Bookworm" is the current stable release of Debian. It's widely used for servers and software development due to its extensive package repositories and long-term support.
- **Nginx**: A high-performance web server and reverse proxy server. Nginx (pronounced "Engine X") is known for handling high volumes of traffic with efficiency, making it suitable for hosting websites and applications. In this exercise, you'll use Nginx to serve both static HTML pages and a Hugo-generated website.
- **sudo**: A Linux command that stands for "superuser do." It allows users with appropriate permissions to execute commands as the superuser (root), providing elevated privileges needed for system administration tasks. For security, standard user accounts typically don't have root access, but by using sudo, they can temporarily gain these permissions.
- **Hugo**: A popular static site generator designed to build websites quickly from plain text files, often written in Markdown. Hugo is widely used for generating simple, fast-loading websites, such as blogs and documentation sites. In this exercise, you'll build and deploy a basic website using Hugo.

Objective

This worksheet will guide you through: - Updating the OS - Installing and configuring Nginx and curl - Setting up a user account with sudo permissions and a home directory - Creating and deploying a static site using Hugo

---

Prerequisites

You should have access to a Debian Bookworm server. Log in with an account that has sudo privileges.

Steps

---

1. Update the Operating System

1. **Update package lists**:

   ```
   sudo apt update
   ```

2. **Upgrade installed packages**:

   ```
   sudo apt upgrade -y
   ```

3. **Remove unused packages**:

   ```
   sudo apt autoremove -y
   ```

2. Install Nginx and Curl

1. **Install Nginx and curl**:

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

2. **Start and enable Nginx** to ensure it runs on startup:

   ```
   sudo systemctl start nginx
   sudo systemctl enable nginx
   ```

3. **Verify Nginx is running** by visiting the server's IP address in a web browser or running:

```
curl http://localhost
```

---

3. Add a Standard User Account with Home Directory and Sudo Permissions

1. **Create a new user with a home directory** (replace `newuser` with the desired username):

   ```
   sudo adduser --create-home newuser
   ```

2. **Follow the prompts** to set a password and provide other user details.

3. **Add the user to the sudo group** to grant sudo privileges:

   ```
   sudo usermod -aG sudo newuser
   ```

4. **Verify sudo permissions** by logging in as the new user and running a sudo command:

   ```
   su - newuser
   sudo ls /root
   ```

   If prompted, enter the new user's password. You should see the contents of the `/root` directory if permissions are set correctly.

---

4. Set Up Nginx for User's Home Page

1. **Create a public_html directory** in the user's home:

   ```
   sudo mkdir /home/newuser/public_html
   sudo chown -R newuser:newuser /home/newuser/public_html
   ```

2. **Create a sample HTML page** in this directory:

   ```
   echo '<h1>Welcome to the home page for newuser!</h1>' > /home/newuser/public_html/index.html
   ```

3. **Configure Nginx to Serve User Directories**:

   - Open the Nginx configuration file:

     ```
     sudo nano /etc/nginx/sites-available/default
     ```

   - Add the following lines within the server block to set up the user directory:

     ```
     location /~newuser/ {
         alias /home/newuser/public_html/;
         autoindex on;
     }
     ```

4. **Check for syntax errors** in the configuration:

   ```
   sudo nginx -t
   ```

5. **Restart Nginx** to apply the changes:

   ```
   sudo systemctl restart nginx
   ```

6. **Test**: Navigate to `http://your_server_ip/~newuser/` in a browser. You should see the user's home page.

---

5. Log Into the User Account

1. **Switch to the user**:

   ```
   su - newuser
   ```

2. You are now logged in as `newuser`. Use this session to complete the next steps.

---

6. Install Hugo

1. **Download Hugo**:

   ```
   sudo apt install hugo -y
   ```

2. **Verify Hugo installation**:

   ```
   hugo version
   ```

---

7. Create and Build a Basic Hugo Site

1. **Initialize a new Hugo site**:

   ```
   hugo new site myhugosite
   ```

2. **Navigate to the site directory**:

   ```
   cd myhugosite
   ```

3. **Add a theme** (example: `ananke`):

   ```
   git init
   git submodule add https://github.com/theNewDynamic/gohugo-theme-ananke.git themes/ananke
   ```

4. **Configure the theme** in `config.toml`:

   ```
   echo 'theme = "ananke"' >> config.toml
   ```

5. **Create a new content page**:

   ```
   hugo new posts/my-first-post.md
   ```

6. **Edit the post** to add content:

   ```
   nano content/posts/my-first-post.md
   ```

7. **Build the Hugo site**:

   ```
   hugo
   ```

   This command will generate the static files in the `public` directory.

---

8. Deploy the Hugo Site to User's Home Directory

1. **Copy the Hugo-generated site** to `public_html`:

   ```
   cp -r public/* /home/newuser/public_html/
   ```

2. **Set permissions** so that Nginx can read the files:

   ```
   sudo chown -R newuser:newuser /home/newuser/public_html
   ```

3. **Test the deployment** by visiting `http://your_server_ip/~newuser/` in a browser. You should see the Hugo site displayed.

Recap and Questions

- What did you learn about managing web servers and deploying static sites?
- What additional configurations would you apply for a production environment?
- What are the potential security risks of allowing users to serve content from their home directories, and how might these risks be mitigated?
- What are some advantages and challenges of using static site generators like Hugo compared to dynamic content management systems (CMS) like WordPress?
- Why is it beneficial to use Infrastructure as Code (IaC) tools (like Ansible or Terraform) to automate setting up environments, and how would this apply to deploying Nginx and Hugo in a larger-scale environment?

- What factors would you consider when configuring firewall rules or access controls to secure a server like the one you set up?
- If the Hugo site content needed to be updated frequently, what approach could you use to automate updates and redeployment?**
- What are some benefits of adding monitoring tools (like Prometheus and Grafana) to track the performance and uptime of services such as Nginx?
- How could you leverage version control (such as Git) to manage and track changes to your Hugo site or Nginx configuration files, especially when working in a team?