Automate the setup of Linux, Apache, MySQL, and PHP on multiple servers

You're a DevOps engineer at CodeBite Ltd. Your dev team needs identical LAMP environments across dev, staging, and prod for testing their new CRM web app. Manual setup is error-prone, so you decide to automate LAMP stack provisioning using Ansible across Ubuntu-based VMs.

Objectives

The **LAMP Stack Deployment using Ansible** experiment aims to automate the installation and configuration of the LAMP stack (Linux, Apache, MySQL, and PHP) across multiple web servers, streamlining the setup process and ensuring consistency. By creating modular Ansible roles for Apache, MySQL, PHP, and firewall configuration, this project simplifies server management, reduces manual configuration errors, and enhances security by properly configuring firewall rules. The automation of the entire deployment process ensures high availability, optimized performance, and enables quick scalability, making it easier to manage and scale infrastructure. Additionally, this experiment demonstrates best practices in Infrastructure as Code (IaC) by utilizing Ansible to manage server infrastructure and application deployment efficiently.

If your using VMware

You will need **at least two virtual machines** for the deployment:

1. **Control Machine** (where Ansible runs).
2. **Target Machine(s)** (where LAMP stack will be deployed).

1. Control Machine (Ansible Host)

- **Number of Machines**: 1
- **Purpose**: This machine will run Ansible to manage and configure the target machine(s).
- **OS**: Linux-based (Ubuntu, CentOS, etc.).
- **Software**: Ansible installed.
- **Network Access**: This VM will need network access to communicate with the target machine(s) via SSH.

2. Target Machines (Web Servers)

- **Number of Machines**: 1 or more (depending on your use case).
- **Purpose**: These are the servers where the **LAMP stack** (Linux, Apache, MySQL, PHP) will be deployed.
- **OS**: Linux-based (Ubuntu, CentOS, etc.).
- **Software**: Apache, MySQL, PHP, UFW (firewall).
- **Network Access**: These VMs need to be accessible via SSH from the **control machine**.

System requirements

Operating System:

- Linux-based OS (e.g., Ubuntu, CentOS, Debian).

Ansible:

- Version 2.9 or higher.
- Installed on the local machine.

SSH Client:

- Installed on the local machine for remote server access.

Target Servers:

- **Apache** (web server).
- **MySQL** (database server).
- **PHP** (scripting language).
- **UFW** (for firewall management).

Text Editor/IDE:

- For editing playbooks and configuration files (e.g., VS Code, Sublime Text).

Project structure

```
lamp-ansible-project/
├── inventory/
│   └── hosts.ini            # Inventory file listing the web servers (IP addresses)
├── roles/
│   ├── apache/
│   │   ├── tasks/
│   │   │   └── main.yml     # Tasks for Apache installation and configuration
│   │
│   ├── mysql/
│   │   ├── tasks/
│   │   │   └── main.yml     # Tasks for MySQL installation and configuration
│   │
│   ├── php/
│   │   ├── tasks/
```

```
│   │   └── main.yml       # Tasks for PHP installation and configuration
│   └── files/
│       └── index.php      # Sample PHP page to be deployed
│
└── firewall/
    └── tasks/
        └── main.yml       # Tasks for firewall configuration (UFW setup)
│
├── site.yml               # Main playbook that includes all roles (Apache, MySQL, PHP, Firewall)
└── README.md              # Project documentation (optional, can include setup instructions)
```

Step 1: Create a project folder

- mkdir lamp-ansible-project && cd lamp-ansible-project

Step 2 Create an inventory file

- Create an inventory file (hosts.ini) that lists the IP addresses or hostnames of your target web servers under the [webservers] group.
- Create a folder and add your web servers' IPs.
- You will need to add the IP address of the web server which is your end point

- Examples inventory/hosts.ini
  [webservers]
  192.168.1.101
  192.168.1.102

Step 3: Create the site playbook

- Create a playbook (site.yml) that includes roles for Apache, MySQL, PHP, and firewall configuration to automate the installation and setup.
- Install Apache on Servers:
- In the apache role, define tasks to install Apache (apache2), ensure it is started, and enable it to start on boot.

This is the main playbook that ties all roles together.

You will be required to create roles in the playbook inorder for the site to execute

name: Install LAMP stack on web servers

hosts: webservers

become: true

roles:

- apache

- mysql

- php

- firewall

Step 4: Create roles directory

mkdir -p roles/{apache,mysql,php,firewall}/tasks

Step 5: Add Apache role

You will need to edit the yml file

name: Install Apache

apt:

name: XXXX -→ you will need to edit the file

state: XXXX -→ you will need to edit the file

update_cache: yes

Step 6: Add MySQL role

In the mysql role, define tasks to install MySQL (mysql-server), start the service, and enable it to run on boot.

Like the pervious step create the yml file enable the role for services

Step 7: Add PHP role

- In the php role, install PHP and required extensions, and deploy a sample PHP file (index.php) to the Apache web server's document root.

Step 8: Add Firewall role

- In the firewall role, configure UFW to allow incoming HTTP (port 80) and HTTPS (port 443) traffic on the target servers, then enable UFW.

Step 9: Run the playbook

Code to run the playbook **ansible-playbook -i inventory/hosts.ini site.yml**

Expected output

- **Apache is installed and running.**
- **MySQL is installed and running.**
- **PHP is configured.**

- **Your test PHP page (phpinfo()) is live at:**
  examples (http://192.168.1.101/index.php and http://192.168.1.102/index.php)