Project: Automate the Configuration of a Server

This project demonstrates how to automate the setup and maintenance of a web server and a database server using Vagrant and Ansible. The web server will serve a static HTML page using Nginx, while the database server will be set up to run a simple database service.

Key Components:

- **Vagrant**: Manages the virtual machines.
- **VirtualBox**: Provides virtualization.
- **Ansible**: Automates server configuration and maintenance.

Final Vagrantfile and Ansible Playbooks

Vagrantfile

```
# Vagrantfile Vagrant.configure("2") do |config| # Configuration for the web server VM
config.vm.define "web_server" do |web| web.vm.box = "ubuntu/bionic64" web.vm.provider
"virtualbox" do |vb| vb.memory = "1536" # 1.5 GB RAM vb.cpus = 1 # 1 CPU end web.vm.network
"private_network", ip: "192.168.33.10" web.vm.provision "ansible" do |ansible| ansible.playbook =
"web_server_playbook.yml" end end # Configuration for the database server VM config.vm.define
"database_server" do |db| db.vm.box = "ubuntu/bionic64" db.vm.provider "virtualbox" do |vb|
vb.memory = "1536" # 1.5 GB RAM vb.cpus = 1 # 1 CPU end db.vm.network "private_network", ip:
"192.168.33.20" db.vm.provision "ansible" do |ansible| ansible.playbook =
"database_server_playbook.yml" end end end
```

Web Server Playbook (web_server_playbook.yml)

```
# web_server_playbook.yml --- - hosts: web_server become: true tasks: - name: Update apt
package index apt: update_cache: yes - name: Install Nginx apt: name: nginx state: present -
name: Start Nginx service service: name: nginx state: started enabled: yes - name: Create a
simple index.html copy: content: | <html> <head> <title>Welcome to the Web Server!</title> </head>
<body> <h1>Hello from the Web Server!</h1> </body> </html> dest: /var/www/html/index.html
owner: www-data group: www-data mode: '0644' - name: Ensure cron is installed apt: name:
cron state: present - name: Add cron job for automatic package updates cron: name: "Automatic
package update" minute: "0" hour: "0" job: "/usr/bin/apt-get update && /usr/bin/apt-get -y
upgrade" state: present
```

Database Server Playbook (database_server_playbook.yml)

```
# database_server_playbook.yml --- - hosts: database_server become: true tasks: - name: Update
apt package index apt: update_cache: yes - name: Install MySQL server apt: name: mysql-server
state: present - name: Start MySQL service service: name: mysql state: started enabled: yes -
name: Ensure cron is installed apt: name: cron state: present - name: Add cron job for
automatic package updates cron: name: "Automatic package update" minute: "0" hour: "0" job:
"/usr/bin/apt-get update && /usr/bin/apt-get -y upgrade" state: present
```

How to Run the Project

1. Create a project directory and navigate into it:

   ```
   mkdir automate-server-configuration cd automate-server-configuration
   ```

2. Create the `Vagrantfile`:

   ```
   nano Vagrantfile
   ```

3. Create the web server playbook:

   ```
   nano web_server_playbook.yml
   ```

4. Create the database server playbook:

   ```
   nano database_server_playbook.yml
   ```

5. Start the Vagrant environment:

   ```
   vagrant up
   ```

Project Result

Upon completing this project, you will have two virtual machines (VMs) efficiently set up and maintained using Vagrant and Ansible.

1. **Web Server VM**:
   - Runs Nginx, a high-performance web server.
   - Serves a static HTML page, demonstrating basic web server functionality.
   - Allocated 1.5 GB of RAM and 1 CPU, ensuring optimal resource usage.
   - Includes a cron job for automatic daily updates of installed packages, enhancing security and stability.
2. **Database Server VM**:
   - Runs MySQL, a popular database server.
   - Configured to start automatically and ready for database operations.

- Also allocated 1.5 GB of RAM and 1 CPU.
- Includes a similar cron job for daily package updates, ensuring it remains up-to-date and secure.

By using Vagrant, the creation and configuration of these VMs are automated and consistent, reducing the chances of manual setup errors. Ansible playbooks ensure that the necessary software is installed and configured correctly, providing a repeatable and reliable setup process. The automation of updates via cron jobs ensures that both servers remain secure and perform optimally over time.

This project showcases how infrastructure as code (IaC) can streamline and enhance server management, making it an excellent example of modern DevOps practices. The skills demonstrated here are valuable for managing scalable and maintainable server environments.

Explanation

"In this project, I automated the configuration of two servers using Vagrant and Ansible. The first server is a web server running Nginx, and the second server is a database server running MySQL. Both VMs are allocated 1.5 GB of RAM and 1 CPU each. The Vagrantfile defines the VMs and their configurations, while the Ansible playbooks handle the installation and setup of the required software, along with scheduling automatic updates using cron jobs."

In summary

This project automates the configuration of two virtual machines (VMs) using Vagrant and Ansible. The first VM is a web server running Nginx, and the second VM is a database server running MySQL. Each VM is allocated 1.5 GB of RAM and 1 CPU.

The `Vagrantfile` defines the setup of both VMs, specifying their resources and network settings. It also links to Ansible playbooks for provisioning. The `web_server_playbook.yml` installs and configures Nginx, sets up a simple HTML page, and schedules a cron job for daily package updates. The `database_server_playbook.yml` installs and configures MySQL, and similarly sets up a cron job for updates.

By running `vagrant up`, Vagrant creates and configures the VMs based on the instructions in the Vagrantfile and playbooks. This approach ensures consistent and automated setup, making server management efficient and reproducible.