****This study guide is based on the video lesson available on TrainerTests.com****

Configuration Management with Ansible on AWS Study Guide

Introduction to Configuration Management

Configuration management is a key component of modern IT operations, enabling automated control over software, hardware, and infrastructure. Tools like Ansible allow administrators to define configurations declaratively and manage servers through automated scripts, reducing manual effort and ensuring consistency across environments.

In this chapter, we will walk through setting up and using **Ansible**—a popular configuration management tool—on AWS, where we will create multiple servers (EC2 instances) and use Ansible to manage and configure them automatically. By the end of this chapter, you'll understand how to:

- Set up an AWS environment for Ansible.
- Install and configure Ansible.
- Use Ansible playbooks to automate server configurations.

Setting Up AWS for Ansible

To follow along, you need an active AWS account. Once you have access, the first step is to prepare your AWS environment by creating several virtual servers using **EC2 (Elastic Compute Cloud)** and ensuring they can communicate with Ansible.

Virtual Private Cloud (VPC)

AWS uses **VPC (Virtual Private Cloud)** to logically isolate network environments. Most AWS regions have a **default VPC** configured, which includes basic networking components like subnets and security groups. In this lesson, we use the default VPC to simplify networking setup.

1. **Log in to the AWS Management Console**.
2. **Navigate to VPC**: Check if a default VPC exists for your region (e.g., North California). This default VPC will ensure that the necessary networking is in place for the servers we create.
3. **Review Subnets**: Ensure that your default VPC has subnets and IP address ranges for hosting instances. AWS automatically manages this.

Creating EC2 Instances

The next step is to create the virtual servers on which we will install and manage services. We'll create three **EC2 instances**, two for web servers and one for control.

1. **Launch EC2 Instances**:
    - Navigate to **EC2** and click **Launch Instance**.
    - Choose **Ubuntu** as the operating system.
    - Configure the default settings, but create a new key pair for SSH access. Call this key pair "Ansible."
    - Create **three instances** with the following roles:
        - **Web1**: A web server.
        - **Web2**: Another web server.
        - **DB1 (Control VM)**: The machine from which Ansible will be installed and used to control the other instances.
    - Enable **HTTP/HTTPS traffic** for the web servers.
2. **Capture IP Addresses**:
    - Once the instances are launched, record the **private and public IP addresses** for both **Web1** and **Web2**. These will be required later for configuration and communication.
    - The **Control VM** also needs its private IP address for our inventory file, which will allow Ansible to communicate with other instances.

Setting Up Ansible on the Control VM

Ansible requires installation on a machine from which it can control other nodes (servers). This control node will issue commands to the other nodes based on **playbooks**.

1. **Connect to Control VM**:
    - Use **EC2 Instance Connect** to access the command prompt of the **Control VM**.
2. **Update the System**:
    - Run the following command to update the system's package list:

```
bash
Copy code
sudo apt update
```

3. **Install Ansible**:
    - To install Ansible, execute:

```
bash
Copy code
sudo apt install ansible -y
```

Setting Up the Inventory File

The **inventory file** in Ansible specifies which servers it will manage. The inventory is organized by groups, such as web servers or database servers.

1. **Create the Inventory File**:
    - Use a text editor such as `nano` to create the file:

```
bash
Copy code
nano inventory
```

- In this file, define the groups and the IP addresses of the servers:

```
ini
Copy code
[web]
172.31.25.137  # Web1 Private IP
172.31.22.189  # Web2 Private IP

[control]
172.31.26.199  # Control VM Private IP
```

2. **Save the File**:
   - Exit and save the file using **Ctrl+X**, then press **Y** to confirm changes.

Preparing SSH Access for Ansible

For Ansible to manage the servers, it needs SSH access to them. When you created the instances, AWS provided a key pair for secure access.

1. **Transfer the Key Pair**:
   - Transfer the key pair from your local computer to the Control VM. Use the following command in your terminal:

```
bash
Copy code
scp -i ansible-key.pem ansible-key.pem ubuntu@<Control_VM_Public_IP>:~/
```

   - Replace `<Control_VM_Public_IP>` with the public IP of your Control VM.
2. **Set Correct Permissions**:
   - Once transferred, set the correct permissions for the key:

```
bash
Copy code
chmod 400 ansible-key.pem
```

3. **Test SSH Connection**:
   - Test your SSH connection from the Control VM to **Web1** and **Web2**:

```
bash
Copy code
ssh -i ansible-key.pem ubuntu@<Web1_Private_IP>
```

Writing and Executing Ansible Playbooks

Ansible uses **playbooks** written in YAML to define the desired configuration on managed servers. A playbook specifies a series of tasks for Ansible to execute.

Example Playbook: Setting Up Apache

1. **Create a Playbook File**:
   - In the Control VM, create a new playbook file:

```
bash
Copy code
nano setup-web-server.yml
```

2. **Define the Playbook**:
    - Use the following YAML code to define the steps for installing **Apache** on the web servers:

```
yaml
Copy code
---
- hosts: web
  become: yes
  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: present
    - name: Start Apache
      service:
        name: apache2
        state: started
        enabled: yes
```

3. **Save the File**:
    - Exit and save the file using **Ctrl+X**, then press **Y** to confirm changes.

Running the Playbook

Now that we have our playbook and inventory set up, it's time to run the playbook and configure our web servers.

1. **Execute the Playbook**:
    - Run the playbook with the following command:

```
bash
Copy code
ansible-playbook -i inventory setup-web-server.yml --private-key ansible-
key.pem
```

2. **Verify Apache Installation**:
    - Open a web browser and enter the public IP of **Web1** and **Web2**. You should see the default Ubuntu Apache webpage, confirming that Apache was successfully installed and running.

Cleaning Up

To avoid unnecessary charges from AWS, ensure that you terminate the EC2 instances when finished.

1. **Terminate Instances**:
    - Go to the **EC2 Dashboard**.
    - Select the instances (Web1, Web2, Control VM) and click **Terminate**.
2. **Verify Termination**:
    - Ensure that the status of the instances changes to **terminated**.

Conclusion

In this chapter, we learned how to set up configuration management using Ansible on AWS. We created EC2 instances, installed Ansible on a control node, wrote an inventory file and a playbook, and automated the installation of Apache on multiple web servers. Ansible simplifies configuration management by automating tasks across multiple servers, ensuring consistency and efficiency in managing IT environments.