**Problem Statement**

You have been asked to:

● Create a script which can add text “This text has been added by custom script” to /tmp.1.txt

● Run this script using Ansible on all the hosts

**Solution Approach**

**Step 1: Create and Setup ec2 instances**

* We will create 3 ec2 instance: Master, Slave 1 and Slave 2
* We will set up the connections such that we can ssh from the Master into both Slave 1 and 2

**Step 2: Setting up Master-Slave architecture**

Used the following set of codes to set up master slave architecture between the Master and 2 Slaves

1. which python3 # check if python is installed

2.

3. # Install Ansible only in the master node

4. sudo yum install -y ansible

5.

6. # Create a new user with the name "ansible" in the MASTER node

7. sudo useradd ansible

8. sudo passwd ansible # Generate a password for the user

9. su – ansible # log in to the user

10.

11. # Create a new user with the name "ansible" in the both slave nodes

12. sudo useradd ansible

13. sudo passwd ansible # Generate a password for the user

14. su - ansible

15.

16. # Provide sudo access to the ansible user in the both the slave nodes

17. cd /etc/ # go to folder

18. sudo vi sudoers # open sudoers in a text editor

19. # add this to wheel: ansible ALL=(ALL) NOPASSWD: ALL

20.

21. # Generate a public and private key in the MASTER node

22. su - ansible # log in to the user

23. ssh-keygen # command to generate keypairs

24.

25. # In order to establish the ssh connection, we need the public IP of the Slave nodes

26. curl ifconfig.me

27.

28. # Run this in SLAVE node to enable ssh port

29. cd /etc/ssh # go to ssh folder

30. sudo vi sshd\_config # the file sshd\_config has details that needs to be changed

31. # Search for the setting "PasswordAuthentication" and change the setting from "no" to "yes"

32. sudo systemctl restart sshd # restart sshd service

33.

34. # Now try gaining remote access to both servers from Master

35. ssh ansible@3.111.149.212 # Slave 1

36. ssh ansible@ 65.0.122.233 # Slave 2

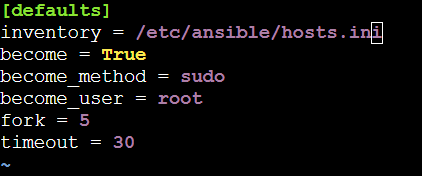
37.

38. # Copy public key from master to remote server

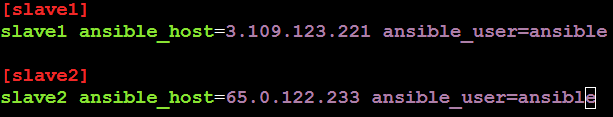
39. ssh-copy-id ansible@{slave public ip}

**Step 3: Creating ansible configuration and inventory file**

1. Creating the configuration file as follows:



1. Creating the inventory file as follows:



**Step 4: Creating the Ansible Roles**

The 2 Ansible Roles will be created as required in the question

1. sudo ansible-galaxy init /etc/ansible/roles/role1 # creating role 1 for apache

2. sudo ansible-galaxy init /etc/ansible/roles/role2 # creating role 2 for nginx

This is what the directory structure looks like after creating the roles

ansible-roles/

├── inventory

├── site.yml

├── roles/

│ ├── apache2/

│ │ ├── tasks/

│ │ │ └── main.yml

│ │ └── defaults/

│ │ └── main.yml

│ ├── nginx/

│ ├── tasks/

│ │ └── main.yml

│ └── defaults/

│ └── main.yml

**Step 5: Updating the main.yml files in the respective roles**

1. The *main.yml* file in role1:

1. ---

2. - name: Install Apache2

3. apt:

4. name: httpd.x86\_64

5. state: present

6. become: yes

1. The *main.yml* file in role2:

1. ---

2. - name: Install Nginx

3. apt:

4. name: nginx

5. state: present

6. become: yes

**Step 6: Creating the Ansible Playbook with the New Roles Defined**

This is the ansible playbook that will be triggered.

1. ---

2. - name: Install Apache2 on slave1

3. hosts: slave1

4. roles:

5. - role1

6.

7. - name: Install Nginx on slave2

8. hosts: slave2

9. roles:

10. - role2

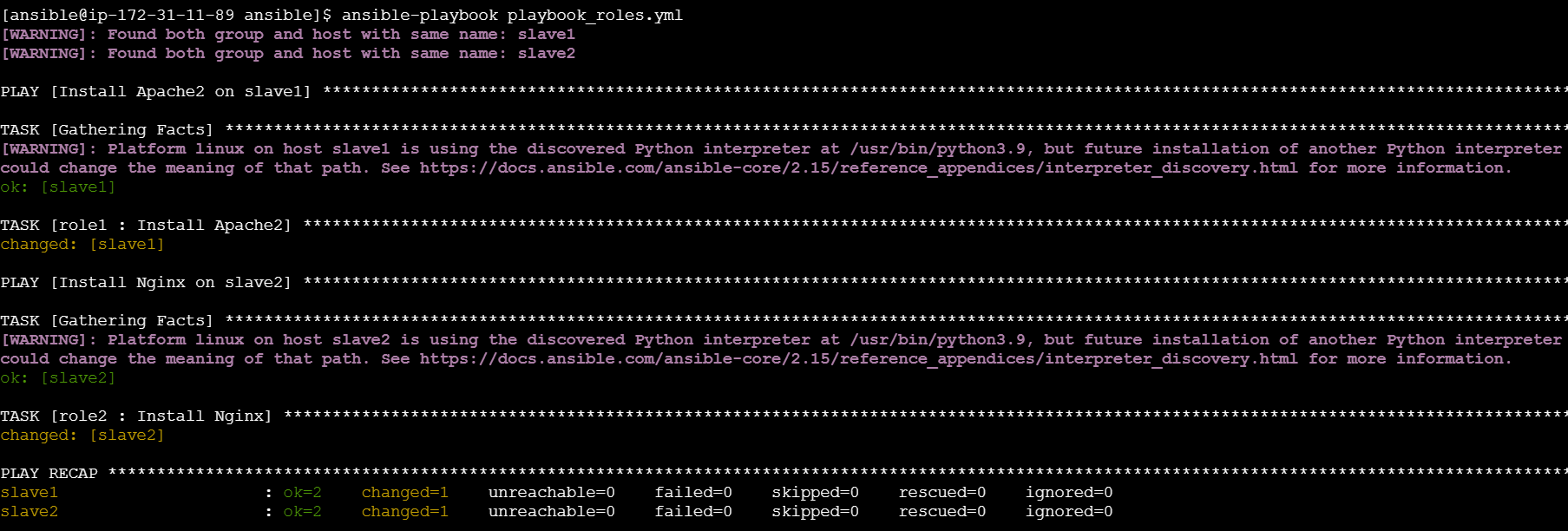
**Step 7: Executing the Ansible Playbook**

The ansible playbook was executed using the below commands:

1. ansible-playbook playbook\_role.yml --syntax-check   # code to check for syntax error

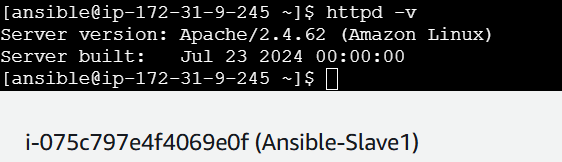
2. ansible-playbook playbook\_role.yml --check          # dry run on terminal

3. ansible-playbook playbook\_role.yml                  # final run on the server



**Step 8: Checking if the libraries have been installed in the respective slave modules**

Slave 1:



Slave 2:

