AWS Infrastructure Automation using Ansible

AUTOMATE. DEPLOY. REPLICATE.

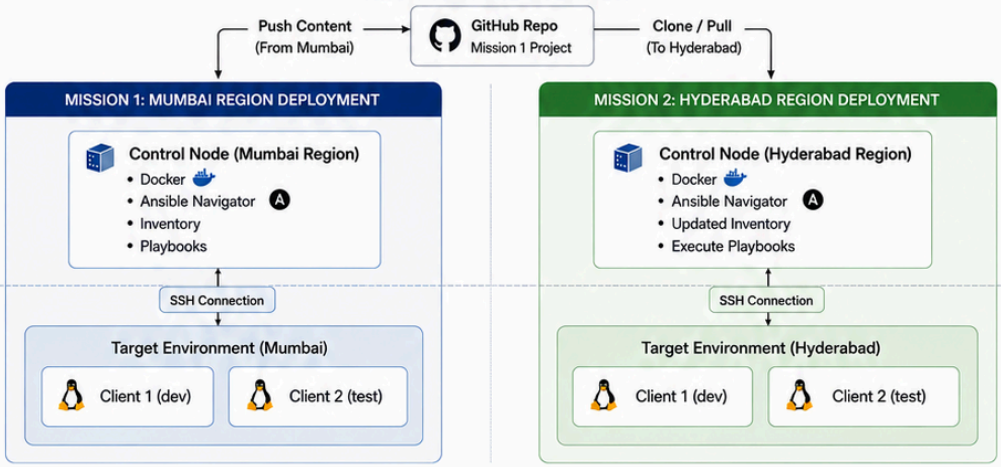
OBJECTIVE

This project demonstrates end-to-end automation of AWS infrastructure using Ansible. It covers provisioning, configuration management, software installation, and web server setup. The automation code is version-controlled with Git and can be reused to replicate the same environment in another AWS region with minimal manual effort.

TECHNOLOGIES USED

AWS EC2 | Amazon Linux 2023 | Docker | Ansible Navigator | Git | GitHub | SSH

ARCHITECTURE OVERVIEW



PROJECT HIGHLIGHTS

INFRASTRUCTURE AUTOMATION
Automate provisioning and configuration of AWS EC2 instances.

CONFIGURATION MANAGEMENT
Use Ansible playbooks, roles, and templates for consistent setups.

REUSABLE & REPLICABLE
Recreate identical environments in another AWS region with ease.

VERSION CONTROL
Store and manage automation code using Git & GitHub.

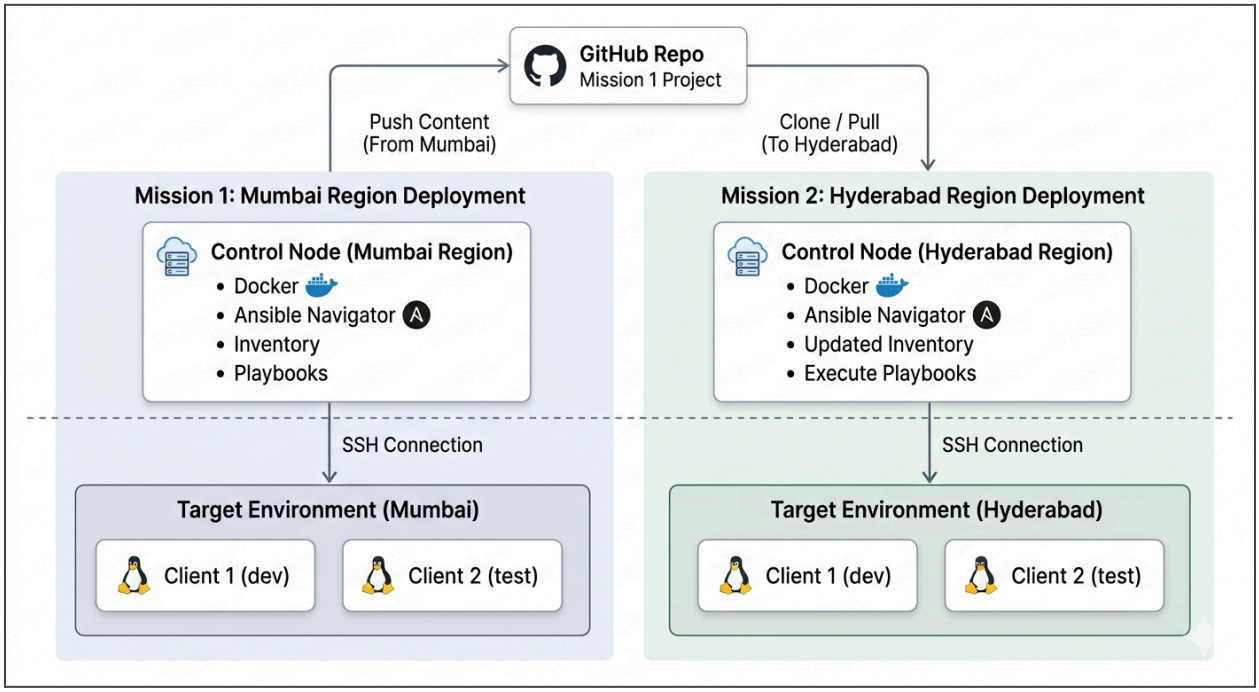
END-TO-END AUTOMATION
Install packages, configure web server, and manage content automatically.

Prepared by Nandu Sivadas
cloud & devops enthusiast

Objective

The main objective of this project is to gain practical experience in AWS infrastructure deployment and infrastructure automation using Ansible. The project aims to automate software installation, web server configuration, content management, and server administration while using Git for version control. It also demonstrates how an existing automation project can be reused to recreate an identical environment in another AWS region with minimal manual effort.

Architecture Diagram



Prerequisites

- AWS EC2 Instance
- Amazon Linux 2023
- Docker
- Ansible Navigator
- Git & GitHub
- SSH

Workflow Explanation

MISSION 1

1. Launch three **AWS EC2** instances in the **Mumbai Region**.
2. Create the **devops** user and configure **SSH** key-based authentication.
3. Install **Docker** and **Ansible Navigator** on the Control Node.
4. Configure the Ansible project using the **inventory** and **ansible.cfg** files.
5. Create **Ansible playbooks**, **roles**, and **Jinja2** templates to automate server configuration.
6. Execute all playbooks using **Ansible Navigator** inside a **Docker container**.
7. Verify package installation, Apache configuration, and web server output.
8. **Push** the automation project to the **GitHub** repository.

MISSION 2

9. Create a new AWS environment in the **Hyderabad Region**.
10. **Clone** the **GitHub** repository and update the inventory.
11. Re-execute all playbooks to recreate the infrastructure.
12. Verify the successful deployment on the new environment.

MISSION 1: Build and Configure an Ansible Automation Environment

Phase 1: Infrastructure Setup

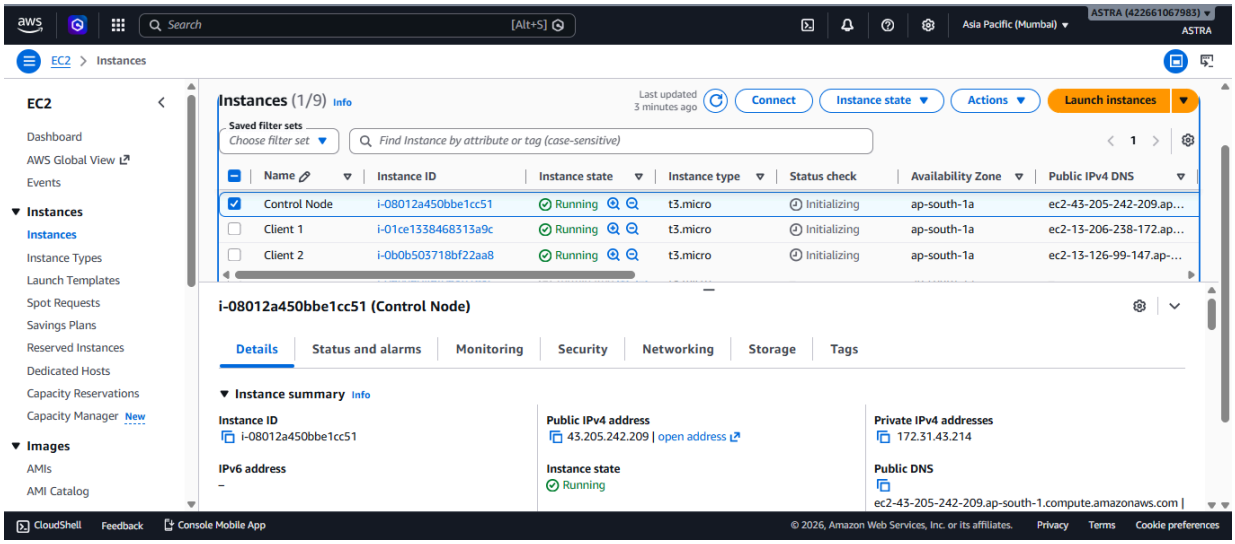
Task 1: Create AWS Infrastructure

Requirements

1. Created three AWS EC2 instances in the Mumbai Region: one Control Node (mumbai-control) and two Client Nodes (mumbai-client1 and mumbai-client2).
2. Create a user named devops on all machines.
3. Configure SSH key-based authentication for the devops user.
4. Use the following project directory on the control node: /home/devops/ansible
5. All Ansible playbooks must be executed using**:**
    - Docker
    - ansible-navigator

Step 01: Created three AWS EC2 instances in Mumbai Region

Control Node**:** mumbai-control **|** Client Nodes**:** mumbai-client 1, mumbai-client 2



Step 02: Create a user named devops on all machines.

Created a `devops` user on all three EC2 instances and added it to the `wheel` group to grant sudo privileges.

ec2-user@ip-172-31-43-214:~ **Control Node**

```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-43-214 ~]$ sudo useradd devops
[ec2-user@ip-172-31-43-214 ~]$ sudo usermod -aG wheel devops
[ec2-user@ip-172-31-43-214 ~]$ id devops
uid=1001(devops) gid=1001(devops) groups=1001(devops),10(wheel)
[ec2-user@ip-172-31-43-214 ~]$
```

ec2-user@ip-172-31-42-176:~ **Client 1**

```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'

[ec2-user@ip-172-31-42-176 ~]$
[ec2-user@ip-172-31-42-176 ~]$ sudo useradd devops
[ec2-user@ip-172-31-42-176 ~]$ sudo usermod -aG wheel devops
[ec2-user@ip-172-31-42-176 ~]$ id devops
uid=1001(devops) gid=1001(devops) groups=1001(devops),10(wheel)
[ec2-user@ip-172-31-42-176 ~]$
```

ec2-user@ip-172-31-46-37:~ **Client 2**

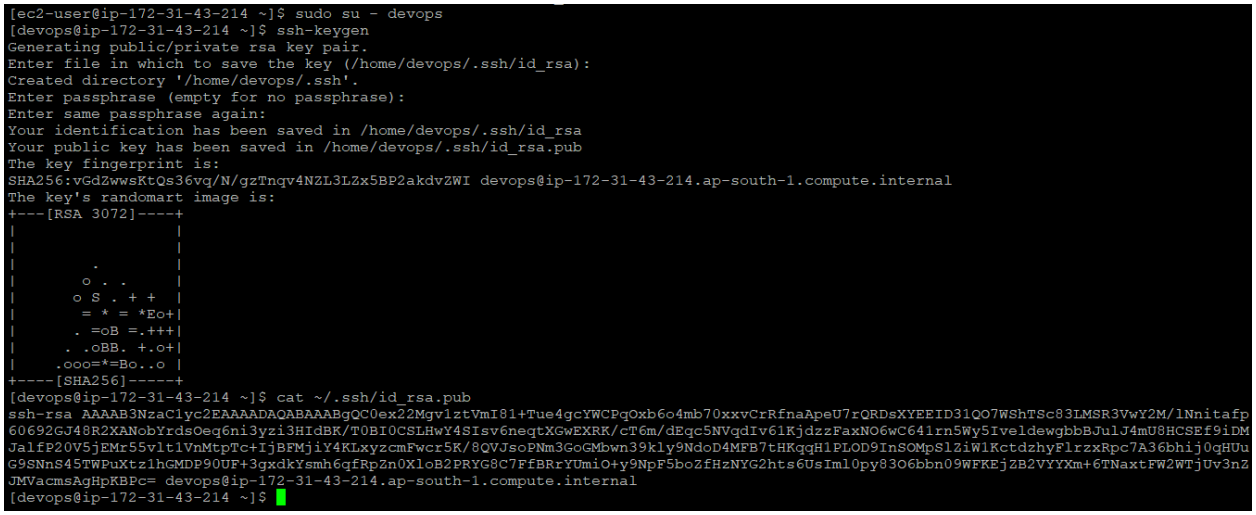
```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-46-37 ~]$ sudo useradd devops
[ec2-user@ip-172-31-46-37 ~]$ sudo usermod -aG wheel devops
[ec2-user@ip-172-31-46-37 ~]$ id devops
uid=1001(devops) gid=1001(devops) groups=1001(devops),10(wheel)
[ec2-user@ip-172-31-46-37 ~]$
```

Step 2: Configure SSH Key-Based Authentication

Configured SSH key-based authentication for the **devops** user to enable passwordless communication between the Control Node and Client Nodes.

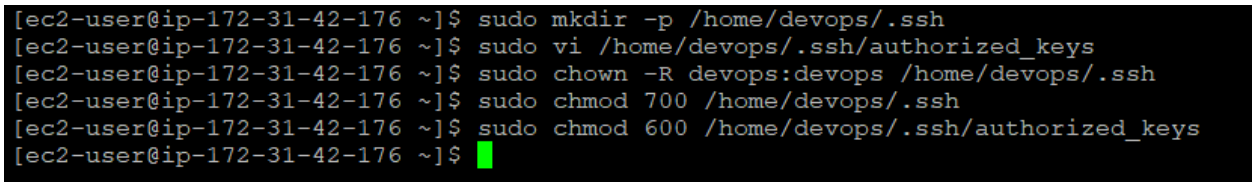
In Control Node

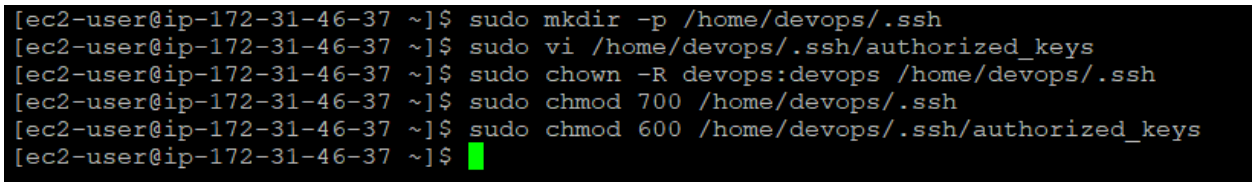
- Generated an SSH key pair on the Control Node.
- Copied the public key to both Client Nodes.



In Client Node - 1 & 2

- Creating `authorized_keys` and setting permissions





Verify passwordless SSH connectivity from the Control Node to the Client Node using SSH.

From Control Node - # ssh devops@<public ip_Client 1>

```
[devops@ip-172-31-43-214 ~]$ ssh devops@13.206.238.172
The authenticity of host '13.206.238.172 (13.206.238.172)' can't be established.
ED25519 key fingerprint is SHA256:Ity1mQ4yKP6WwEoHXt9hx/6Hed2tnfxQ7MhHBCRS/4M.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.206.238.172' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[devops@ip-172-31-42-176 ~]$
```

➔ Successful SSH login to Client Node 1 ✅

From Control Node - # ssh devops@<public ip_Client 2>

```
[devops@ip-172-31-43-214 ~]$ ssh devops@13.126.99.147
The authenticity of host '13.126.99.147 (13.126.99.147)' can't be established.
ED25519 key fingerprint is SHA256:ikHJ1o+tTnUAzZtqcdVwsKeXxiz8cqJd6SHgGrcHi+k.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.126.99.147' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[devops@ip-172-31-46-37 ~]$
```

➔ Successful SSH login to Client Node 2 ✅

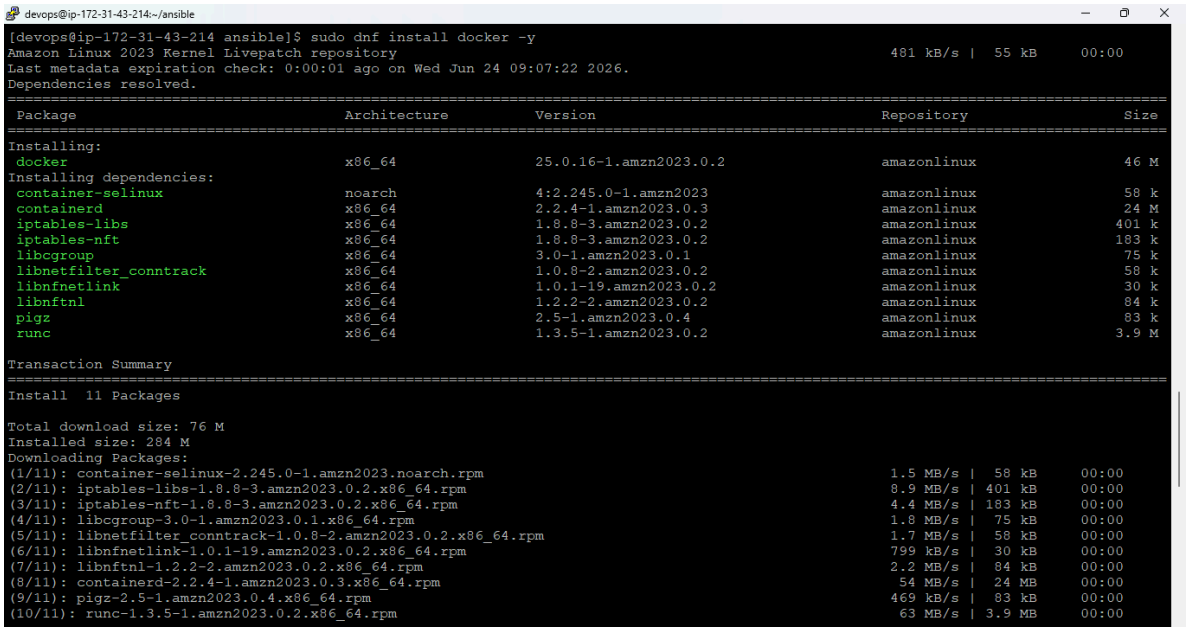
Verified SSH connectivity successfully.

Step 3: Create the Ansible Project Directory

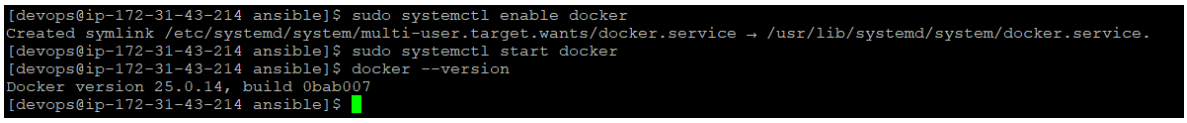
```
[devops@ip-172-31-43-214 ~]$ mkdir -p /home/devops/ansible
[devops@ip-172-31-43-214 ~]$ cd /home/devops/ansible
[devops@ip-172-31-43-214 ansible]$ pwd
/home/devops/ansible
[devops@ip-172-31-43-214 ansible]$
```

Step 4: Set Up the Ansible Execution Environment

1. Installed Docker on the Control Node. # Sudo dnf install docker -y



Verified the Docker installation



2. **Ansible Navigator** installed on Control Node (Install Python pip and Install Ansible Navigator) ✅

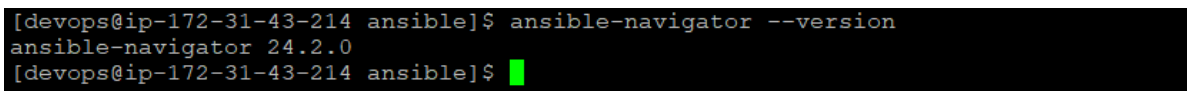
sudo dnf install python3-pip -y

- Installed **Python pip**, which is required to install Ansible Navigator. .

pip3 install ansible-navigator

- Installed **Ansible Navigator** using `pip3` to execute Ansible playbooks in the required container-based environment.

3. Verified the **Ansible Navigator** installation.



Task 2: Install and Configure Ansible Environment

Requirements

1. Create the inventory file in the Ansible project directory.
    - Add mumbai-client1 to the dev group.
    - Add mumbai-client2 to the test group.
2. Create the ansible.cfg configuration file.
3. Verify connectivity between the Control Node and Client Nodes using Ansible.

Step 1: Create the Inventory File

Created the inventory file and added the client nodes to the dev and test groups with the required SSH user.

```
[devops@ip-172-31-43-214 ansible]$ vi inventory
[devops@ip-172-31-43-214 ansible]$ cat inventory
[dev]
client1 ansible_host=172.31.42.176

[test]
client2 ansible_host=172.31.46.37

[all:vars]
ansible_user=devops
[devops@ip-172-31-43-214 ansible]$
```

Step 2: Configure the Ansible Configuration File

Configured the ansible.cfg file by specifying the inventory file and disabling host key checking.

```
[devops@ip-172-31-43-214 ansible]$ vi ansible.cfg
[devops@ip-172-31-43-214 ansible]$ cat ansible.cfg
[defaults]
inventory=inventory
host_key_checking=False
[devops@ip-172-31-43-214 ansible]$
```

Step 3: Create the Ansible Test Playbook

Created a simple ping playbook to verify connectivity between the Control Node and the Client Nodes.

```
[devops@ip-172-31-43-214 ansible]$ vi ping.yml
[devops@ip-172-31-43-214 ansible]$ cat ping.yml
---
- hosts: all
  tasks:
    - ping:
[devops@ip-172-31-43-214 ansible]$
```

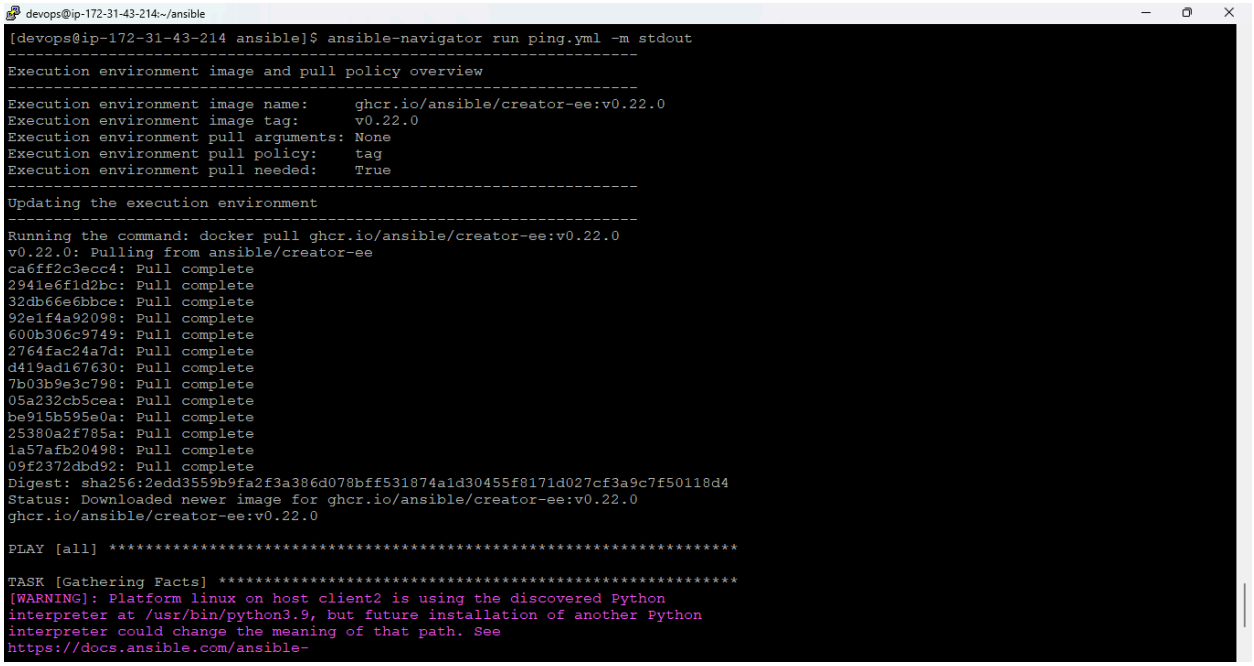
Step 4: Configure Docker Permissions

Added the devops user to the docker group to allow Docker commands without requiring root privileges.

```
[devops@ip-172-31-43-214 ansible]$ sudo usermod -aG docker devops
[devops@ip-172-31-43-214 ansible]$ exit
logout
[ec2-user@ip-172-31-43-214 ~]$ sudo su - devops
Last login: Wed Jun 24 09:06:20 UTC 2026 on pts/1
[devops@ip-172-31-43-214 ~]$ groups
devops wheel docker
[devops@ip-172-31-43-214 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[devops@ip-172-31-43-214 ~]$ cd /home/devops/ansible
```

Step 5: Verify Ansible Connectivity

Executed the ping playbook using Ansible Navigator and verified successful communication with all managed nodes.



```
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [client2]
[WARNING]: Platform linux on host client1 is using the discovered Python
interpreter at /usr/bin/python3.9, but future installation of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [client1]

TASK [ping] ********************************************************************
ok: [client1]
ok: [client2]

PLAY RECAP *********************************************************************
client1                    : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
client2                    : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[devops@ip-172-31-43-214 ansible]$
```

Task 3: Package Management Automation

Requirements

1. Create the `packages.yml` playbook in the Ansible project directory.
2. Install the `mariadb` and `php` `packages` on the **dev** and **test** groups.
3. Install the **Development Tools** package group on the **dev** group.
4. Update all installed packages on the **dev** group.
5. Verify successful playbook execution and package installation.

Step 1: Create and Configure the Package Management Playbook

Create the `packages.yml` playbook in the Ansible project directory and configure it to:

```
[devops@ip-172-31-43-214 ansible]$ vi packages.yml
[devops@ip-172-31-43-214 ansible]$ cat packages.yml
---
- hosts: dev
  become: yes

  tasks:

    - name: Update all packages
      dnf:
        name: "*"
        state: latest

    - name: Install mariadb
      dnf:
        name: mariadb
        state: present

    - name: Install php
      dnf:
        name: php
        state: present

    - name: Install Development Tools
      dnf:
        name: "@Development Tools"
        state: present

- hosts: test
  become: yes

  tasks:

    - name: Install mariadb
      dnf:
        name: mariadb
        state: present
```

```
    - name: Install php
      dnf:
        name: php
        state: present
[devops@ip-172-31-43-214 ansible]$
```

- Install `mariadb` and `php` on the dev and test groups.
- Install the Development Tools package group on the dev group.
- Update all packages on the dev group.

Step 2: Execute and Verify the Playbook

Run the playbook using Ansible Navigator to install and update the required packages on the target nodes. Verify successful execution by checking the task results and the final PLAY RECAP, ensuring all hosts completed successfully without failures.

```
[devops@ip-172-31-43-214 ansible]$ ansible-navigator run packages.yml -m stdout

PLAY [dev] *********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host client1 is using the discovered Python
interpreter at /usr/bin/python3.9, but future installation of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [client1]

TASK [Update all packages] *****************************************************
ok: [client1]

TASK [Install mariadb] *********************************************************
ok: [client1]

TASK [Install php] *************************************************************
ok: [client1]

TASK [Install Development Tools] ***********************************************
ok: [client1]

PLAY [test] ********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host client2 is using the discovered Python
interpreter at /usr/bin/python3.9, but future installation of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [client2]

TASK [Install mariadb] *********************************************************
changed: [client2]
```

```
TASK [Install php] *************************************************************
changed: [client2]

PLAY RECAP *********************************************************************
client1                    : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
client2                    : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[devops@ip-172-31-43-214 ansible]$
```

Task 4: Create and Use Roles

Requirements

1. Create the Ansible role `myrole` in the project directory.
2. Configure the role to install and configure the Apache Web Server.
3. Create the `index.j2` template to display the hostname and IP address.

4. Create the `myrole.yml` playbook to execute the role.
5. Configure the playbook to run only on the **test** group.
6. Verify the web server deployment by accessing the web page using a browser or `curl`.

Step 1: Configure the Ansible Role

Configure the `myrole` role by creating the `tasks/main.yml` file to install and start the Apache web server, then deploy the web page using a Jinja2 template.

```
[devops@ip-172-31-43-214 ansible]$ vi roles/myrole/tasks/main.yml
[devops@ip-172-31-43-214 ansible]$ cat roles/myrole/tasks/main.yml
---
- name: Install Apache
  dnf:
    name: httpd
    state: present

- name: Start Apache
  service:
    name: httpd
    state: started
    enabled: yes

- name: Deploy index page
  template:
    src: index.j2
    dest: /var/www/html/index.html
[devops@ip-172-31-43-214 ansible]$
```

Step 2: Create the Jinja2 Template

Create the `index.j2` template to display the hostname and IP address on the web page.

```
[devops@ip-172-31-43-214 ansible]$ vi roles/myrole/templates/index.j2
[devops@ip-172-31-43-214 ansible]$ cat roles/myrole/templates/index.j2
<h1>Welcome to {{ inventory_hostname }}</h1>
<h2>Managed by Ansible Role</h2>
[devops@ip-172-31-43-214 ansible]$
```

Step 3: Create and Execute the Playbook

Create the `myrole.yml` playbook to execute the `myrole` role on the **test** group, then run the playbook using Ansible Navigator.

```
[devops@ip-172-31-43-214 ansible]$ vi myrole.yml
[devops@ip-172-31-43-214 ansible]$ cat myrole.yml
---
- hosts: test
  become: yes

  roles:
    - myrole
[devops@ip-172-31-43-214 ansible]$
```

```
[devops@ip-172-31-43-214 ansible]$ ansible-navigator run myrole.yml -m stdout

PLAY [test] ********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host client2 is using the discovered Python
interpreter at /usr/bin/python3.9, but future installation of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [client2]

TASK [myrole : Install Apache] *************************************************
ok: [client2]

TASK [myrole : Start Apache] ***************************************************
changed: [client2]

TASK [myrole : Deploy index page] **********************************************
changed: [client2]

PLAY RECAP *********************************************************************
client2                    : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[devops@ip-172-31-43-214 ansible]$
```

Step 4: Verify the Deployment

Verify that the Apache web server is running and confirm the web page is accessible using the `curl` command.

`# curl http://localhost`

```
[ec2-user@ip-172-31-46-37 ~]$ curl http://localhost
<h1>Welcome to client2</h1>
<h2>Managed by Ansible Role</h2>
[ec2-user@ip-172-31-46-37 ~]$
```

Task 5: Content Modification

Requirements

1. Create the issue.yml playbook in the Ansible project directory.
2. Modify the contents of the /etc/issue file.
3. Set the content to **"development"** for the **dev** group.
4. Set the content to **"test"** for the **test** group.
5. Verify the updated /etc/issue content on all target nodes.

Step 1: Create the Playbook

Create the issue.yml playbook and configure it to update the /etc/issue file with **"development"** for the **dev** group and **"test"** for the **test** group.

```
[devops@ip-172-31-43-214 ansible]$ vi issue.yml
[devops@ip-172-31-43-214 ansible]$ cat issue.yml
---
- hosts: dev
  become: yes

  tasks:
    - name: Set issue file for dev group
      copy:
        content: "development\n"
        dest: /etc/issue

- hosts: test
  become: yes

  tasks:
    - name: Set issue file for test group
      copy:
        content: "test\n"
        dest: /etc/issue
[devops@ip-172-31-43-214 ansible]$
```

Step 2: Execute the Playbook

Run the `issue.yml` playbook using Ansible Navigator to apply the changes on the target nodes and verify that the playbook executes successfully.

```
[devops@ip-172-31-43-214 ansible]$ ansible-navigator run issue.yml -m stdout

PLAY [dev] *********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host client1 is using the discovered Python
interpreter at /usr/bin/python3.9, but future installation of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [client1]

TASK [Set issue file for dev group] ********************************************
changed: [client1]

PLAY [test] ********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host client2 is using the discovered Python
interpreter at /usr/bin/python3.9, but future installation of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [client2]

TASK [Set issue file for test group] *******************************************
changed: [client2]

PLAY RECAP *********************************************************************
client1                    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
client2                    : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[devops@ip-172-31-43-214 ansible]$
```

Step 3: Verify the Configuration

Verify the updated `/etc/issue` content on both the **dev** and **test** nodes using the `cat /etc/issue` command.

Client 1 (dev) → (`cat /etc/issue` → `development`)

```
[ec2-user@ip-172-31-42-176 ~]$ cat /etc/issue
development
[ec2-user@ip-172-31-42-176 ~]$
```

Client 2 (test) → (`cat /etc/issue` → `test`)

```
[ec2-user@ip-172-31-46-37 ~]$ cat /etc/issue
test
[ec2-user@ip-172-31-46-37 ~]$
```

Task 6: Custom-Based Web Server

Requirements

1. Create the `custom.yml` playbook in the Ansible project directory.
2. Configure the Apache Web Server on the **dev** group.
3. Set the document root to `/webdev`.
4. Create a symbolic link from `/webdev` to `/var/www/html`.
5. Configure the web page to display the content **"development"**.
6. Verify the web page and confirm the displayed content.

Step 1: Create the Playbook

Create the `custom.yml` playbook to install and configure the Apache Web Server, create the `/webdev` document root, add the web page content, and create a symbolic link from `/webdev` to `/var/www/html`.

```
[devops@ip-172-31-43-214 ansible]$ vi custom.yml
[devops@ip-172-31-43-214 ansible]$ cat custom.yml
---
- hosts: dev
  become: yes

  tasks:

    - name: Install Apache
      dnf:
        name: httpd
        state: present

    - name: Start Apache
      service:
        name: httpd
        state: started
        enabled: yes

    - name: Create webdev directory
      file:
        path: /webdev
        state: directory

    - name: Create index page
      copy:
        content: "development\n"
        dest: /webdev/index.html

    - name: Remove default html directory
      file:
        path: /var/www/html
        state: absent
```

```
    - name: Create symbolic link
      file:
        src: /webdev
        dest: /var/www/html
        state: link
[devops@ip-172-31-43-214 ansible]$
```

Step 2: Execute the Playbook

Run the `custom.yml` playbook using Ansible Navigator and verify that all tasks complete successfully without errors.

```
[devops@ip-172-31-43-214 ansible]$ ansible-navigator run custom.yml -m stdout

PLAY [dev] *********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host client1 is using the discovered Python
interpreter at /usr/bin/python3.9, but future installation of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [client1]

TASK [Install Apache] **********************************************************
ok: [client1]

TASK [Start Apache] ************************************************************
changed: [client1]

TASK [Create webdev directory] *************************************************
changed: [client1]

TASK [Create index page] *******************************************************
changed: [client1]

TASK [Remove default html directory] ********************************************
changed: [client1]

TASK [Create symbolic link] ****************************************************
changed: [client1]

PLAY RECAP *********************************************************************
client1                    : ok=7    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[devops@ip-172-31-43-214 ansible]$
```

Step 3: Verify the Web Server Configuration

Verify that the symbolic link is created, the web page content is available, and the Apache Web Server is serving the page successfully.

```
# ls -ld /var/www/html
# cat /webdev/index.html
# systemctl status httpd
```

```
[ec2-user@ip-172-31-42-176 ~]$ ls -ld /var/www/html
lrwxrwxrwx. 1 root root 7 Jun 24 10:10 /var/www/html -> /webdev
[ec2-user@ip-172-31-42-176 ~]$ cat /webdev/index.html
development
```

```
[ec2-user@ip-172-31-42-176 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Wed 2026-06-24 10:10:27 UTC; 1min 50s ago
       Docs: man:httpd.service(8)
   Main PID: 43031 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:   0 B/sec"
      Tasks: 177 (limit: 1059)
     Memory: 18.3M
        CPU: 172ms
     CGroup: /system.slice/httpd.service
             ├─43031 /usr/sbin/httpd -DFOREGROUND
             ├─43033 /usr/sbin/httpd -DFOREGROUND
             ├─43034 /usr/sbin/httpd -DFOREGROUND
             ├─43035 /usr/sbin/httpd -DFOREGROUND
             └─43066 /usr/sbin/httpd -DFOREGROUND

Jun 24 10:10:27 ip-172-31-42-176.ap-south-1.compute.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Jun 24 10:10:27 ip-172-31-42-176.ap-south-1.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Jun 24 10:10:27 ip-172-31-42-176.ap-south-1.compute.internal httpd[43031]: Server configured, listening on: port 80
[ec2-user@ip-172-31-42-176 ~]$ curl http://localhost
development
[ec2-user@ip-172-31-42-176 ~]$
```

Task 7: Git Integration

Requirements

1. Configure the local Git repository.
2. Add the required Ansible files and role to the repository.
3. Commit the project files with an appropriate commit message.
4. Push the committed changes to the remote Git repository.
5. Verify the uploaded files and Git commit history in the repository.

Step 1: Create and Configure the GitHub Repository

Initialize the local GitHub repository, add the required project files, and commit the changes with an appropriate commit message.

```
# git commit -m "Initial Ansible Automation Project”
```

Creates a commit and saves the project changes to the local Git repository

```
[devops@ip-172-31-43-214 ansible]$ git commit -m "Initial Ansible Automation Project"
[master (root-commit) f68515e] Initial Ansible Automation Project
 8 files changed, 130 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 custom.yml
 create mode 100644 inventory
 create mode 100644 issue.yml
 create mode 100644 myrole.yml
 create mode 100644 packages.yml
 create mode 100644 roles/myrole/tasks/main.yml
 create mode 100644 roles/myrole/templates/index.j2
[devops@ip-172-31-43-214 ansible]$
```

Step 2: Push the Project to GitHub

Configure the remote repository and push the committed project files to the GitHub repository.

```
# git push -u origin main
```

Pushes the committed project files from the local repository to the GitHub repository

```
[devops@ip-172-31-43-214 ansible]$ git branch -M main
[devops@ip-172-31-43-214 ansible]$ git push -u origin main
Username for 'https://github.com': nandusivadas98@gmail.com
Password for 'https://nandusivadas98%40gmail.com@github.com':
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 2 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (14/14), 1.61 KiB | 1.61 MiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: This repository moved. Please use the new location:
remote:   https://github.com/nandusivadas/Ansible-Automation-Project.git
To https://github.com/nandusivadas/ansible-automation-project.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
[devops@ip-172-31-43-214 ansible]$
```

Step 3: Verify the Repository

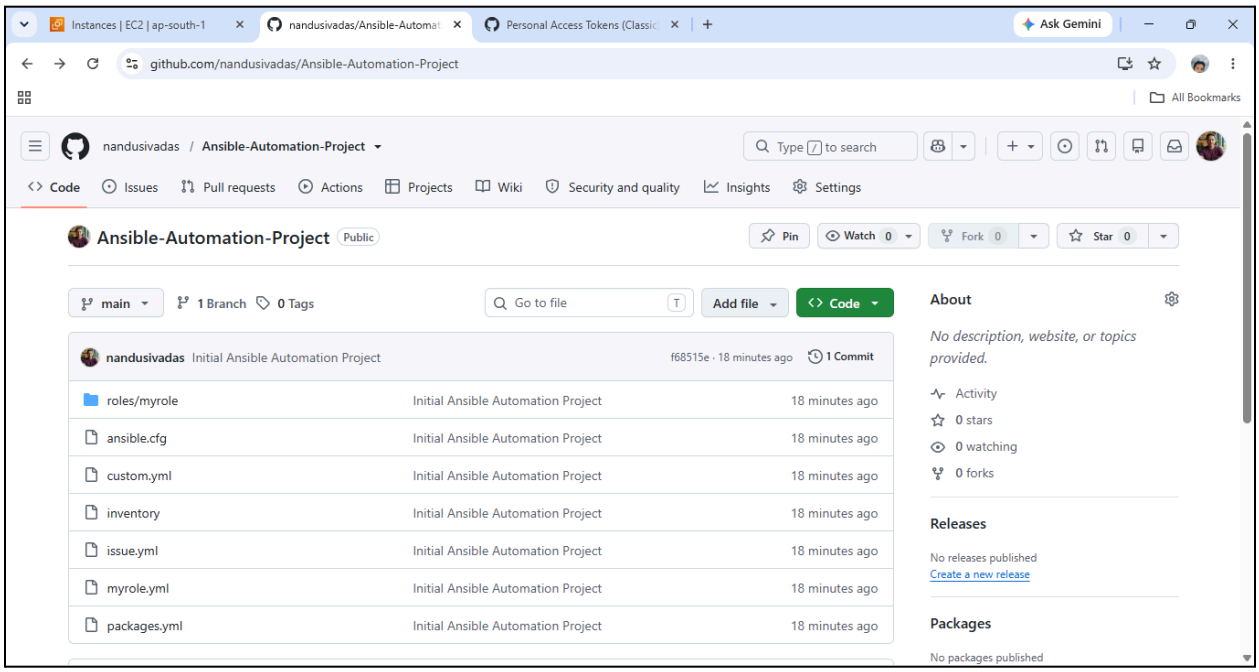
Verify the Git commit history and confirm that all project files have been uploaded successfully to the GitHub repository.

```
# git log --oneline
```

Displays the Git commit history in a single-line format to verify the commit.

```
[devops@ip-172-31-43-214 ansible]$ git log --oneline
f68515e (HEAD -> main, origin/main) Initial Ansible Automation Project
[devops@ip-172-31-43-214 ansible]$
```

GitHub repository page showing the uploaded files



Mission 1 Completion

Mission 1 was successfully completed by implementing all the required tasks and verifying the expected outputs. The Ansible environment was configured successfully, connectivity between the control node and client nodes was established, and all project requirements were achieved.

MISSION 2 – Infrastructure Recreation Using Git

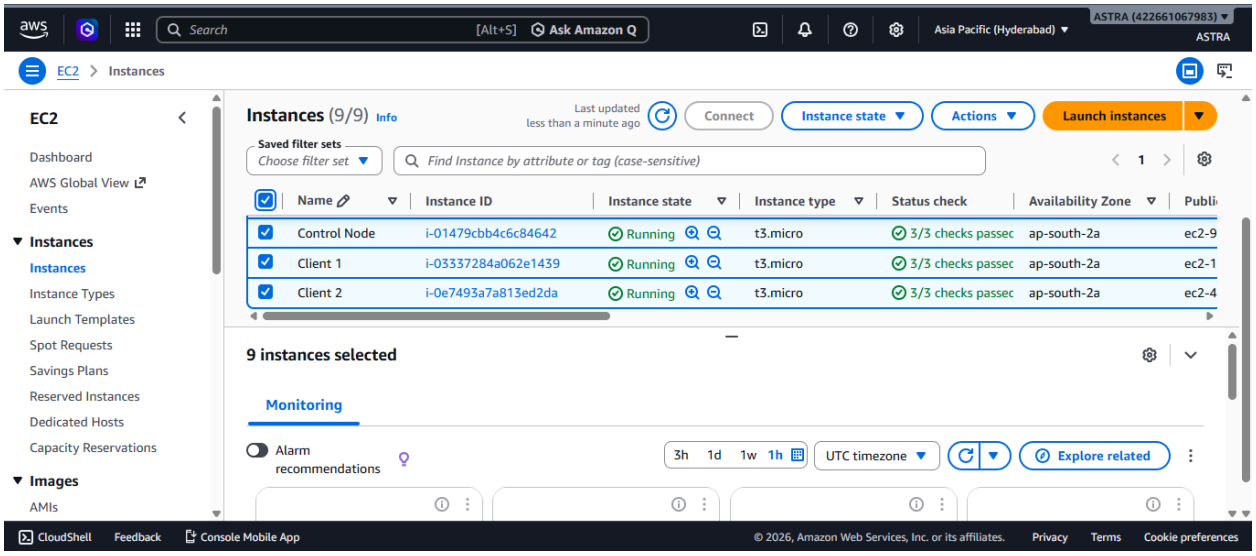
Phase 1: Create a New Environment

Requirements

1. Launch a new AWS environment in the **Hyderabad Region**.
2. Create one control node and two client nodes.
3. Assign the required instance names:
    - `hyderabad-control`
    - `hyderabad-client1`
    - `hyderabad-client2`
4. Create a user named **`clone`** on all machines.
5. Create the Ansible project directory at `/home/clone/ansible`

Step 1: Launch the AWS Environment

Launch one control node and two client nodes in the **AWS Hyderabad Region** using the required naming convention.



Step 2: Create the `clone` User

Create the `clone` user on the **control node** and **both client nodes**, then add the user to the `wheel` group for administrative privileges. Verify the user creation using the `id clone` command.

```
ec2-user@ip-172-31-6-129:~
login as: ec2-user
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-6-129 ~]$ sudo useradd clone
[ec2-user@ip-172-31-6-129 ~]$ sudo usermod -aG wheel clone
[ec2-user@ip-172-31-6-129 ~]$ id clone
uid=1001(clone) gid=1001(clone) groups=1001(clone),10(wheel)
[ec2-user@ip-172-31-6-129 ~]$
```

Step 3: Configure Passwordless SSH Authentication

Generate an SSH key pair on the control node and configure passwordless SSH access to both client nodes.

In Control Node:

```
# ssh-keygen
```

```
[ec2-user@ip-172-31-6-129 ~]$ sudo su - clone
[clone@ip-172-31-6-129 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/clone/.ssh/id_rsa):
Created directory '/home/clone/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/clone/.ssh/id_rsa
Your public key has been saved in /home/clone/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:kQEgr9onht9iYRwM5YdRYbops89GEuLQo5QNwBGg/7M clone@ip-172-31-6-129.ap-south-2.compute.internal
The key's randomart image is:
+---[RSA 3072]----+
|*+*.=o...        |
|o+ B     o       |
|.+B o   o        |
|++=*     .       |
|B+=o    S        |
|oO=.             |
|+++.+            |
| =++ o           |
| o+.E            |
+----[SHA256]-----+
[clone@ip-172-31-6-129 ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDnZc3r3WOGYN00sTZs+pdCeD/ZvGzCb7GkS+tilWXA5mXMG3qzNdR83LkWPz2/6YWXVWiOVlNJuwA8UVhjtjfy5n5DS4wmoh
UE6EWqulq9Xlbemj2GBCIeipylBOGA2cybuzydWPYsIyGKjgjsZ1Rxtn2d37N+bVhtVQN+i2NV9WuK852hzJZWms0zIlbrqTeXHIWuCTfsMO1Uqs90LQaTJLjxl0Ny8+tLGiw7
mXPzKioYz+r27h8LShtnUzt3rdzIc+EkFa6pWdDihqAMZ0VqwMDbO8BFvJAs9bpsIXOygTQrverTP9sqVMzQb0p1T19eYMbKDJP8cVCKUj9lya3qC8nqAyo/HYaYdpA7qmSN14
p9EYuk4ktoSzJ7iN7pqWVUQrBBglhf1E/yeEikP3e2VZGuYxU6GtZ1+jBEii66ZHxPwufORGGXTmrhHHh72F5BhALgL7h0KRiABEWifrMRhdYmmtulVjzYr5Ov1PQ/rGbay6X1
VkXNwGqgBVEJONs= clone@ip-172-31-6-129.ap-south-2.compute.internal
[clone@ip-172-31-6-129 ~]$
```

Step 4: Configure SSH Authentication on Client Nodes

Create the `.ssh` directory and configure the `authorized_keys` file on the client nodes to enable passwordless SSH authentication.

```
[ec2-user@ip-172-31-15-188 ~]$ sudo su - clone
[clone@ip-172-31-15-188 ~]$ mkdir -p ~/.ssh
[clone@ip-172-31-15-188 ~]$ chmod 700 ~/.ssh
[clone@ip-172-31-15-188 ~]$ vim ~/.ssh/authorized_keys
[clone@ip-172-31-15-188 ~]$ chmod 600 ~/.ssh/authorized_keys
[clone@ip-172-31-15-188 ~]$
```

Successful SSH login from Control Node to a Client node

```
[clone@ip-172-31-6-129 ~]$ ssh clone@16.112.61.172
The authenticity of host '16.112.61.172 (16.112.61.172)' can't be established.
ED25519 key fingerprint is SHA256:WJn0ZsXmCjYWfpMErcN/oFZifrx/tC0MCiKeixR9srg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '16.112.61.172' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Jun 25 07:27:41 2026
[clone@ip-172-31-15-188 ~]$
```

Step 5: Create the Ansible Project Directory

Create the project directory `/home/clone/ansible` on the control node.

```
[clone@ip-172-31-6-129 ~]$ mkdir ~/ansible
[clone@ip-172-31-6-129 ~]$ cd ~/ansible
```

Phase 2: Clone Existing Automation Project

Requirements

1. Clone the Git repository into the Ansible project directory.
2. Update the inventory file to match the new environment.
3. Verify SSH connectivity between the control node and client nodes.
4. Execute all Ansible playbooks using Ansible Navigator.
5. Verify the successful execution of all playbooks:
    - `packages.yml`
    - `myrole.yml`
    - `issue.yml`

- `custom.yml`

Step 1: Configure Docker Environment

Installed Docker on the control node, enabled and started the Docker service, and verified that it was running successfully to support the Ansible Execution Environment.

```
[clone@ip-172-31-6-129 ansible]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[clone@ip-172-31-6-129 ansible]$ sudo systemctl start docker
[clone@ip-172-31-6-129 ansible]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
     Active: active (running) since Thu 2026-06-25 07:42:00 UTC; 14s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Process: 28375 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
    Process: 28376 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Main PID: 28377 (dockerd)
      Tasks: 9
     Memory: 30.0M
        CPU: 336ms
     CGroup: /system.slice/docker.service
             └─28377 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Jun 25 07:41:59 ip-172-31-6-129.ap-south-2.compute.internal systemd[1]: Starting docker.service - Docker Application Container Engine>
Jun 25 07:41:59 ip-172-31-6-129.ap-south-2.compute.internal dockerd[28377]: time="2026-06-25T07:41:59.687076404Z" level=info msg="Sta>
Jun 25 07:41:59 ip-172-31-6-129.ap-south-2.compute.internal dockerd[28377]: time="2026-06-25T07:41:59.757358535Z" level=info msg="Loa>
Jun 25 07:42:00 ip-172-31-6-129.ap-south-2.compute.internal dockerd[28377]: time="2026-06-25T07:42:00.296831210Z" level=info msg="Loa>
Jun 25 07:42:00 ip-172-31-6-129.ap-south-2.compute.internal dockerd[28377]: time="2026-06-25T07:42:00.330468670Z" level=info msg="Doc>
Jun 25 07:42:00 ip-172-31-6-129.ap-south-2.compute.internal dockerd[28377]: time="2026-06-25T07:42:00.330789990Z" level=info msg="Dae>
Jun 25 07:42:00 ip-172-31-6-129.ap-south-2.compute.internal dockerd[28377]: time="2026-06-25T07:42:00.359211057Z" level=info msg="API>
Jun 25 07:42:00 ip-172-31-6-129.ap-south-2.compute.internal systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-22/22 (END)
```

Step 2: Install and Verify Ansible

Installed Ansible and Ansible Navigator on the control node and verified the installation

```
[clone@ip-172-31-6-129 ansible]$ ansible --version
ansible [core 2.15.13]
  config file = None
  configured module search path = ['/home/clone/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/clone/.local/lib/python3.9/site-packages/ansible
  ansible collection location = /home/clone/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/clone/.local/bin/ansible
  python version = 3.9.25 (main, May 25 2026, 00:00:00) [GCC 11.5.0 20240719 (Red Hat 11.5.0-5)] (/usr/bin/python3)
  jinja version = 3.1.6
  libyaml = True
[clone@ip-172-31-6-129 ansible]$ ansible-navigator --version
ansible-navigator 24.2.0
[clone@ip-172-31-6-129 ansible]$
```

Step 3: Clone the Automation Project

Cloned the Git repository into the `/home/clone/ansible` project directory.

```
[clone@ip-172-31-6-129 ansible]$ git clone https://github.com/nandusivadas/Ansible-Automation-Project.git .
Cloning into '.'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 14 (delta 0), reused 14 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (14/14), done.
[clone@ip-172-31-6-129 ansible]$ ls
ansible.cfg  custom.yml  inventory  issue.yml  myrole.yml  packages.yml  roles
[clone@ip-172-31-6-129 ansible]$
```

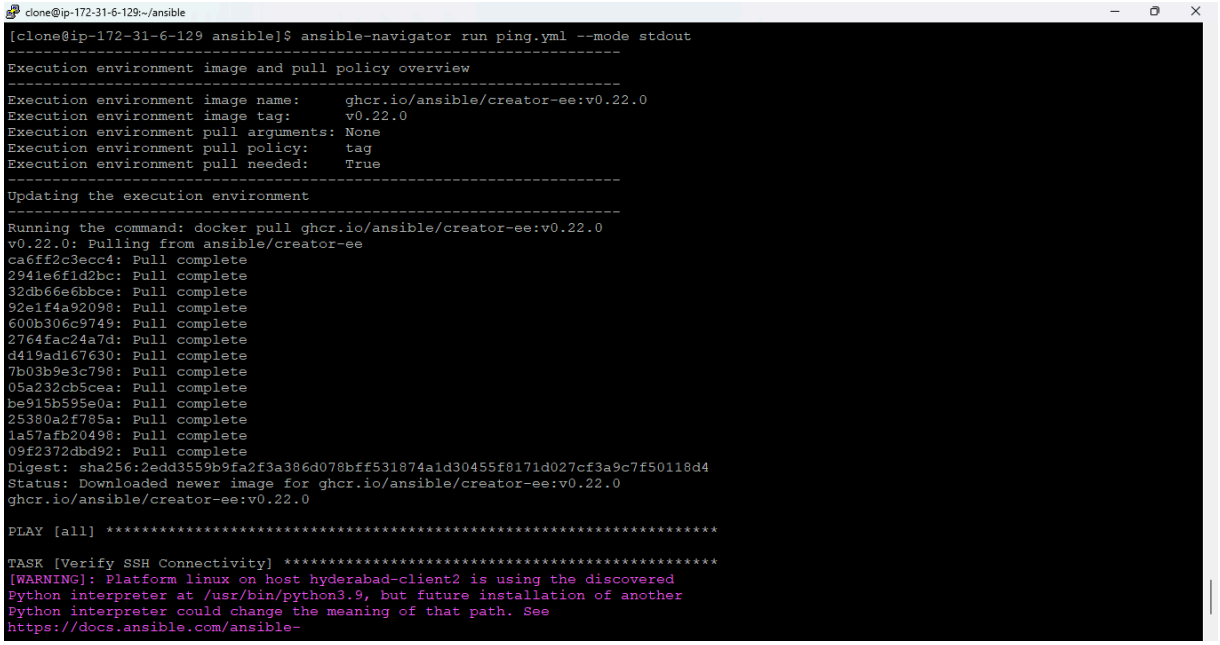
Step 4: Configure the Inventory File

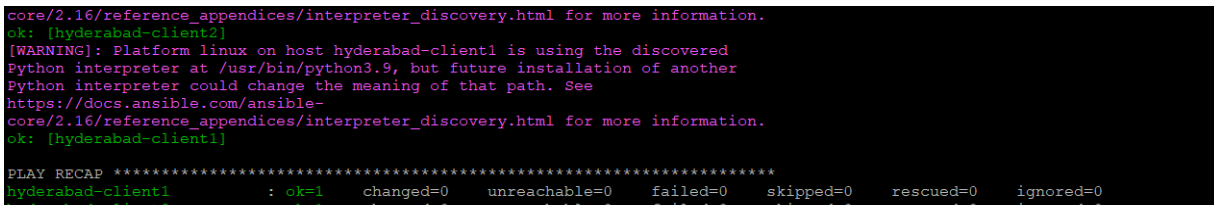
Updated the Ansible inventory file with the hostnames and IP addresses of the client nodes.

```
[clone@ip-172-31-6-129 ansible]$ vim inventory
[clone@ip-172-31-6-129 ansible]$ cat inventory
[dev]
hyderabad-client1 ansible_host=172.31.15.188

[test]
hyderabad-client2 ansible_host=172.31.4.20

[all:vars]
ansible_user=clone
[clone@ip-172-31-6-129 ansible]$
```

Step 5: Verify SSH Connectivity

Verified passwordless SSH connectivity between the control node and the client nodes using the Ansible ping playbook.





Step 6: Execute the Required Playbooks

Executed the required Ansible playbooks to configure the client nodes and install the necessary packages.

```
[clone@ip-172-31-6-129 ansible]$ ansible-navigator run packages.yml --mode stdout

PLAY [dev] *********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host hyderabad-client1 is using the discovered
Python interpreter at /usr/bin/python3.9, but future installation of another
Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [hyderabad-client1]

TASK [Update all packages] *****************************************************
ok: [hyderabad-client1]

TASK [Install mariadb] *********************************************************
changed: [hyderabad-client1]

TASK [Install php] *************************************************************
changed: [hyderabad-client1]

TASK [Install Development Tools] ***********************************************
changed: [hyderabad-client1]

PLAY [test] ********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host hyderabad-client2 is using the discovered
Python interpreter at /usr/bin/python3.9, but future installation of another
Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [hyderabad-client2]

TASK [Install mariadb] *********************************************************
changed: [hyderabad-client2]
```

```
TASK [Install php] *************************************************************
changed: [hyderabad-client2]

PLAY RECAP *********************************************************************
hyderabad-client1          : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
hyderabad-client2          : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[clone@ip-172-31-6-129 ansible]$
```

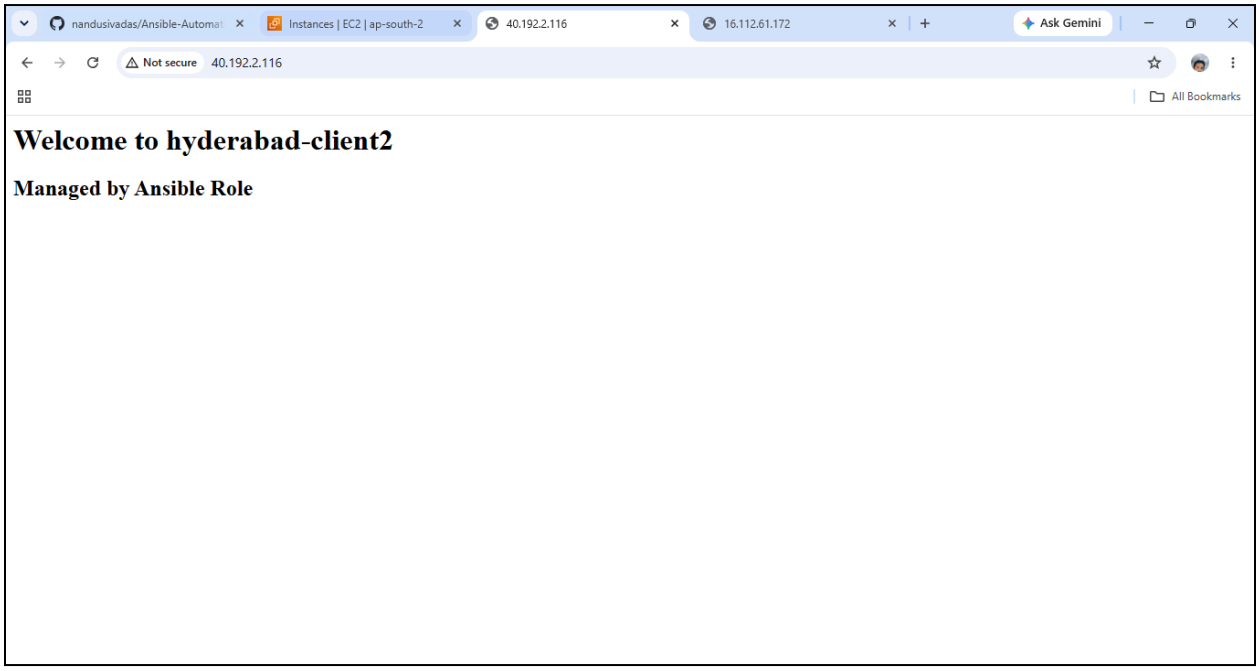
Step 7: Deploy the Apache Web Server

Executed the Ansible role to install and configure the Apache web server on the target node.

```
[clone@ip-172-31-6-129 ansible]$ ansible-navigator run myrole.yml --mode stdout

PLAY [test] ********************************************************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Platform linux on host hyderabad-client2 is using the discovered
Python interpreter at /usr/bin/python3.9, but future installation of another
Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [hyderabad-client2]

TASK [myrole : Install Apache] *************************************************
ok: [hyderabad-client2]

TASK [myrole : Start Apache] ***************************************************
changed: [hyderabad-client2]

TASK [myrole : Deploy index page] **********************************************
changed: [hyderabad-client2]

PLAY RECAP *********************************************************************
hyderabad-client2          : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[clone@ip-172-31-6-129 ansible]$
```

Step 8: Verify the Web Server Output

Verified the successful deployment by accessing the web application through a web browser.



Mission 2 Completion

Mission 2 was successfully completed by cloning the automation project, configuring the inventory, verifying SSH connectivity, and executing all required Ansible playbooks using Ansible Navigator. The Apache web server was deployed successfully, and the web application outputs were verified on the client nodes, confirming that all project requirements were achieved.

Project Outcome

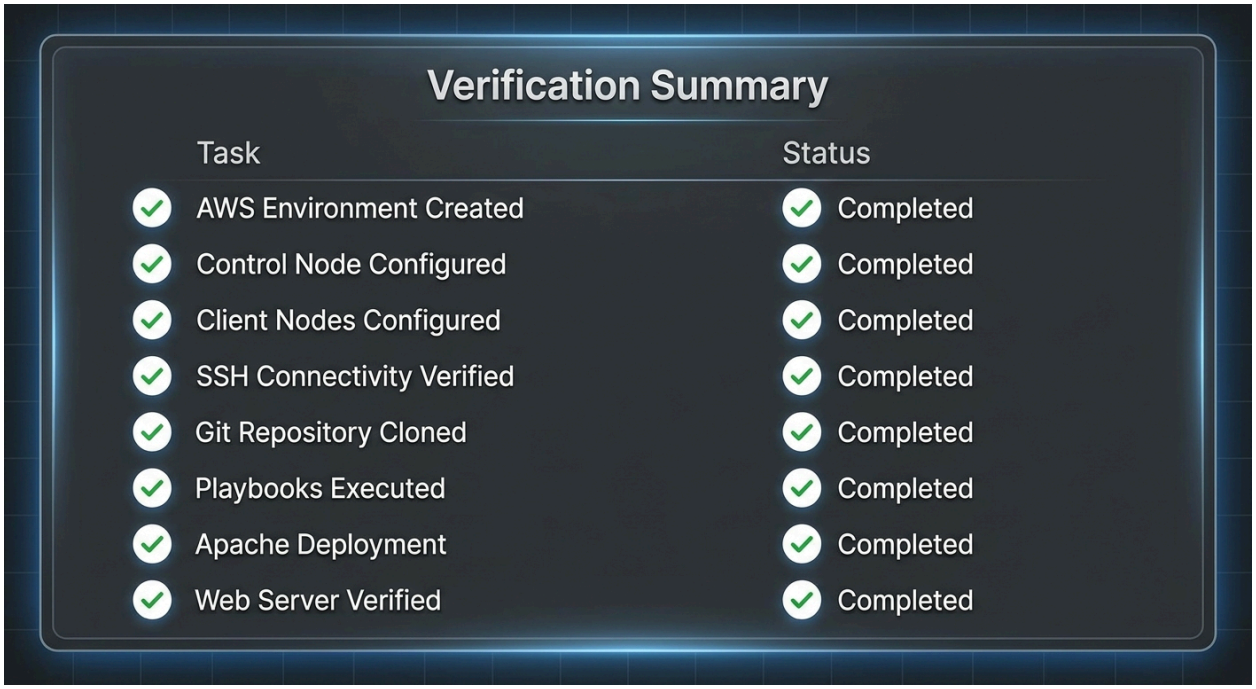
The project successfully automated the configuration and management of multiple Linux servers using Ansible. All required playbooks and roles were executed successfully, and the expected outputs were verified on the target systems.

Conclusion

The Ansible Automation Project was completed successfully by setting up the control node, configuring passwordless SSH communication, creating and executing playbooks, and deploying web server configurations using Ansible roles. The project demonstrated how automation simplifies system administration by reducing manual effort and ensuring consistent configuration across multiple servers.

Skills Gained

- AWS EC2 Instance Management
- Linux User Administration
- SSH Key-Based Authentication
- Git and GitHub
- Docker
- Ansible
- Ansible Navigator
- Ansible Playbooks
- Ansible Roles
- Apache Web Server Deployment



GitHub Repository

Repository URL:

https://github.com/nandusivadas/AWS-Infrastructure-Automation-using-Ansible

The complete source code, configuration files, documentation, and project resources are available in the GitHub repository.

References

- AWS Documentation
- Ansible Documentation
- Docker Documentation
- Git Documentation
