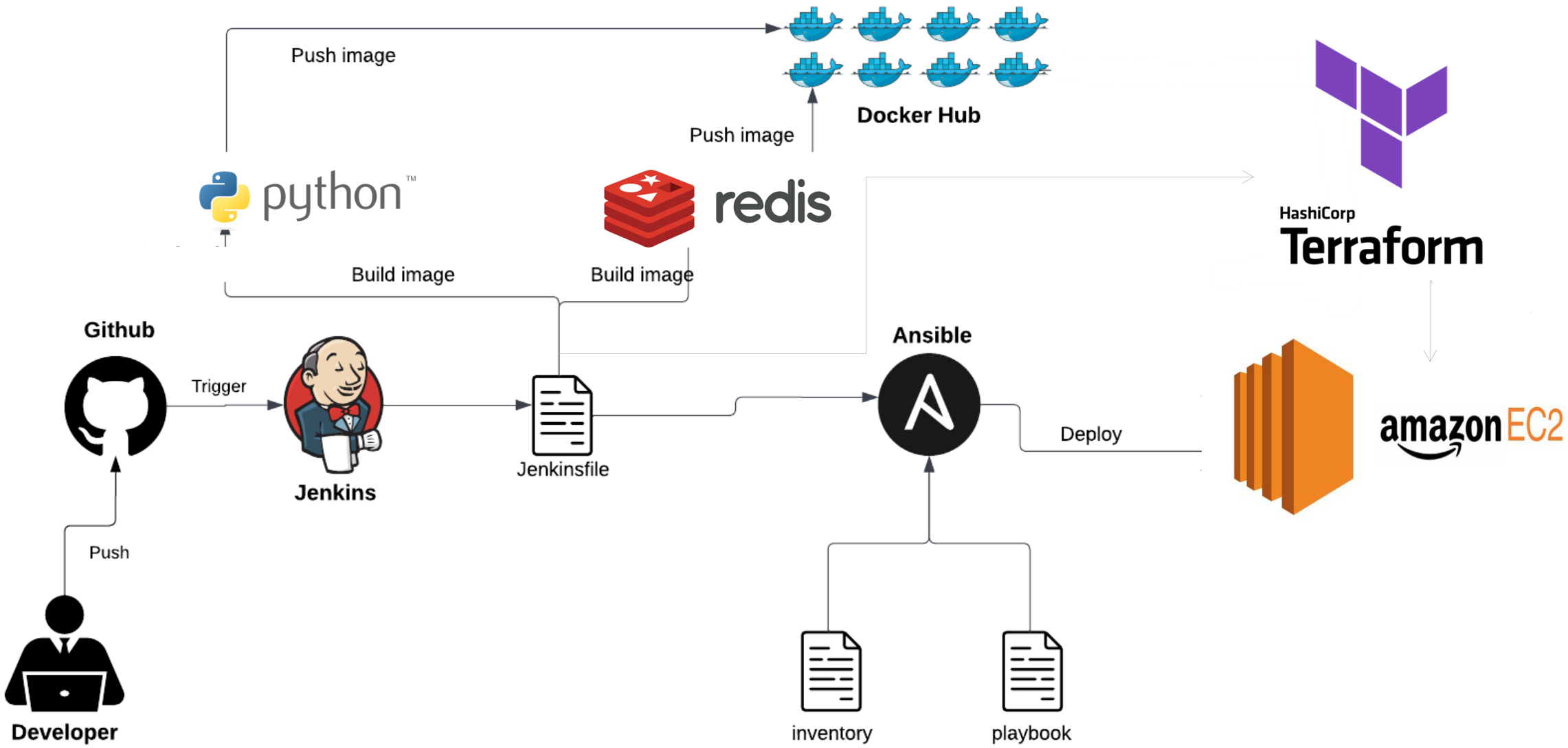
Digital Egypt Pioneers Initiative

AUTOMATED DEPLOYMENT PIPELINE WITH JENKINS AND DOCKER

Supervisor: Eng. Ahmed Nasr

TEAM 3

Project overview



Phase 1: Initial Setup & Dockerization

- This phase focused on setting up the local environment, building the Dockerized application, and testing locally.

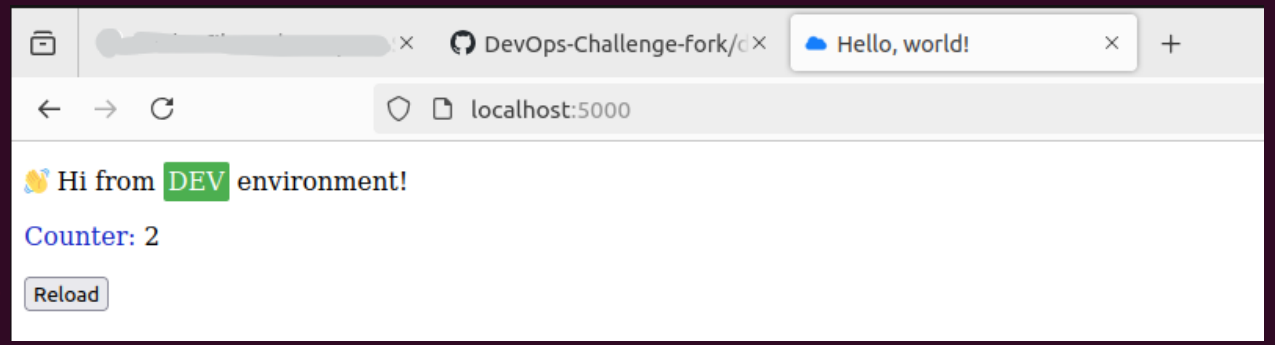
```yaml
version: '3'
services:
  app:
    build: .
    container_name: my_python_app
    ports:
      - "5000:5000"
    environment:
      - ENVIRONMENT=DEV
      - HOST=0.0.0.0
      - PORT=5000
      - REDIS_HOST=redis
      - REDIS_PORT=6379
      - REDIS_DB=0
    depends_on:
      - redis
    command: sh -c "sleep 5 && python hello.py" # Wait for Redis to start

  redis:
    image: "redis:5.0"
    container_name: redis
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
```

```dockerfile
FROM python:3.8-slim

WORKDIR /app

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "hello.py"]
```

Phase 1: Initial Setup & Dockerization

- This phase focused on setting up the local environment, building the Dockerized application, and testing locally.

```
hp@Ubuntu-Khaleel:~/devops-project/DevOps-Challenge-fork$ docker-compose up -d
Creating network "devops-challenge-fork_default" with the default driver
Creating volume "devops-challenge-fork_redis_data" with default driver
Pulling redis (redis:5.0)...
5.0: Pulling from library/redis
a603fa5e3b41: Pull complete
77631c3ef092: Pull complete
ed3847cf62b8: Pull complete
295236254fbb: Pull complete
1d25d6f70191: Pull complete
23acdbbf5eef: Pull complete
Digest: sha256:fc5ecd863862f89f04334b7cbb57e93c9790478ea8188a49f6e57b0967d38c75
Status: Downloaded newer image for redis:5.0
Creating redis ... done
Creating my_python_app ... done
hp@Ubuntu-Khaleel:~/devops-project/DevOps-Challenge-fork$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS
                                  NAMES
d79d6db7c28f   devops-challenge-fork_app      "python hello.py"        9 seconds ago    Up 6 seconds    0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
                                  my_python_app
b4ded7136856   redis:5.0                      "docker-entrypoint.s…"   13 seconds ago   Up 10 seconds   0.0.0.0:6379->6379/tcp, :::6379->6379/tcp
                                  redis
d3b4cc686bc8   myjenkins-blueocean:2.414.2    "/usr/bin/tini -- /u…"   3 weeks ago      Up 16 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:50000->
50000/tcp, :::50000->50000/tcp   jenkins-blueocean
hp@Ubuntu-Khaleel:~/devops-project/DevOps-Challenge-fork$
```



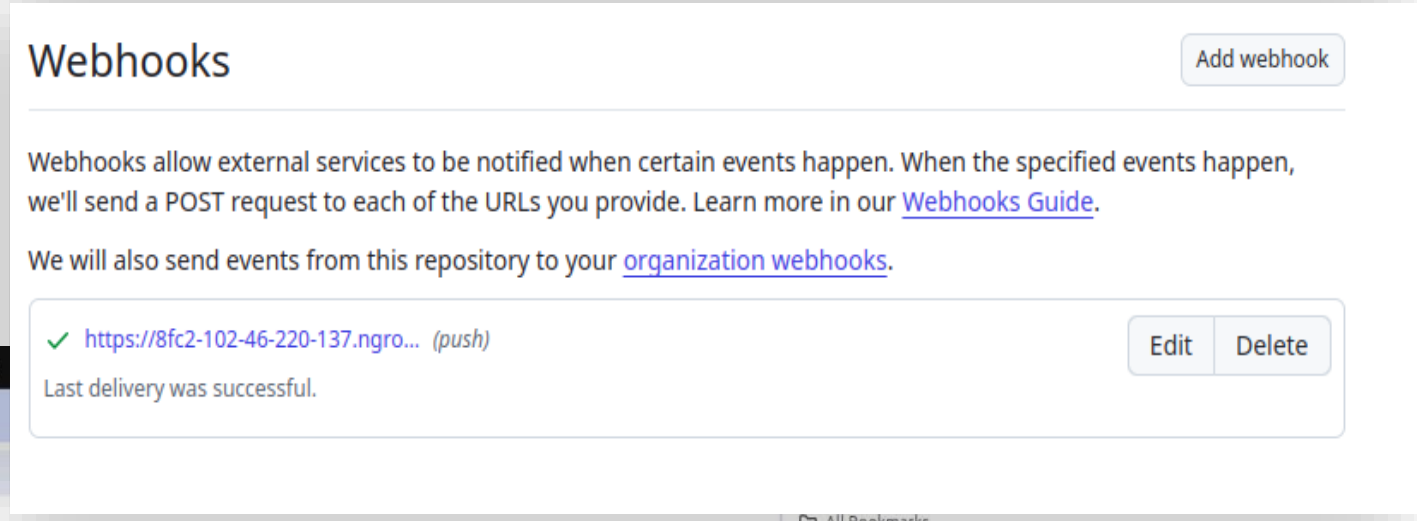
Phase 2: Jenkins & CI Integration

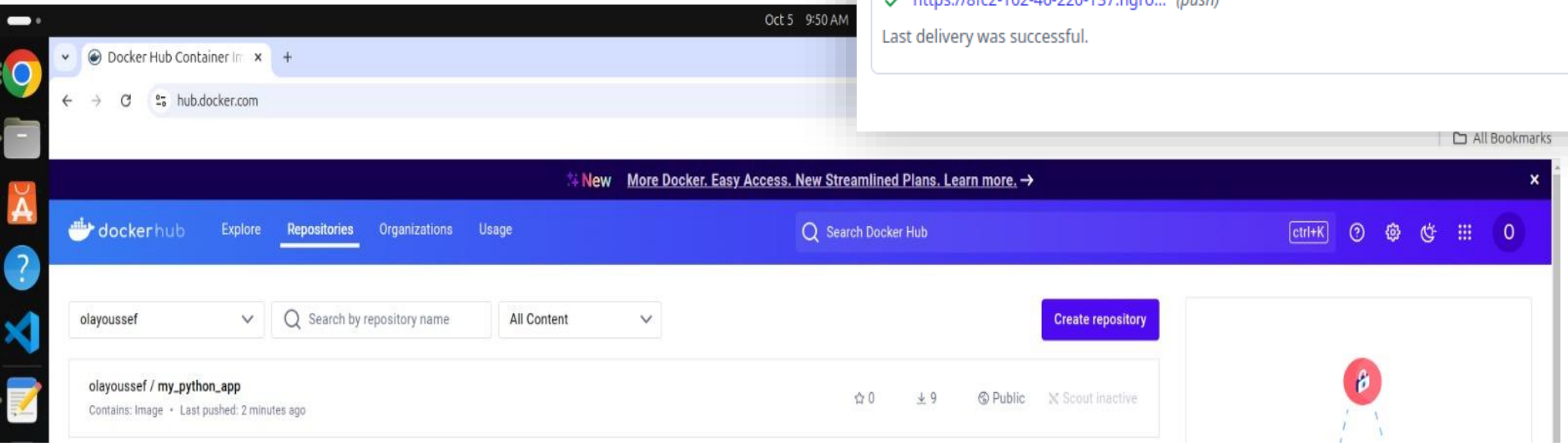
- This phase we have to:
- Set up and configure Jenkins for CI/CD
- Integrate with Docker/GitHub/Dockerhub for automated builds

```
stages {

    stage('Build and Push Docker Image') {
        steps {
            script {

                sh 'docker build -t my_python_app .'


                withCredentials([usernamePassword(credentialsId: 'dockerhub-credentials', passwordVariable: 'pass', usernameVariable: 'dockerhubuser')]) {
                    sh 'docker login -u $dockerhubuser -p $pass'
                    sh 'docker tag my_python_app:latest $DOCKER_IMAGE_NAME:latest'
                    sh 'docker push $DOCKER_IMAGE_NAME:latest'
                }
            }
        }
    }

    stage('Run App') {
        steps {
            script {
                sh '''
                # Clean up any existing containers
                docker rm -f my_python_app || true
                docker rm -f redis || true
                # Run the application
                docker-compose up -d
                '''
            }
        }
    }
```

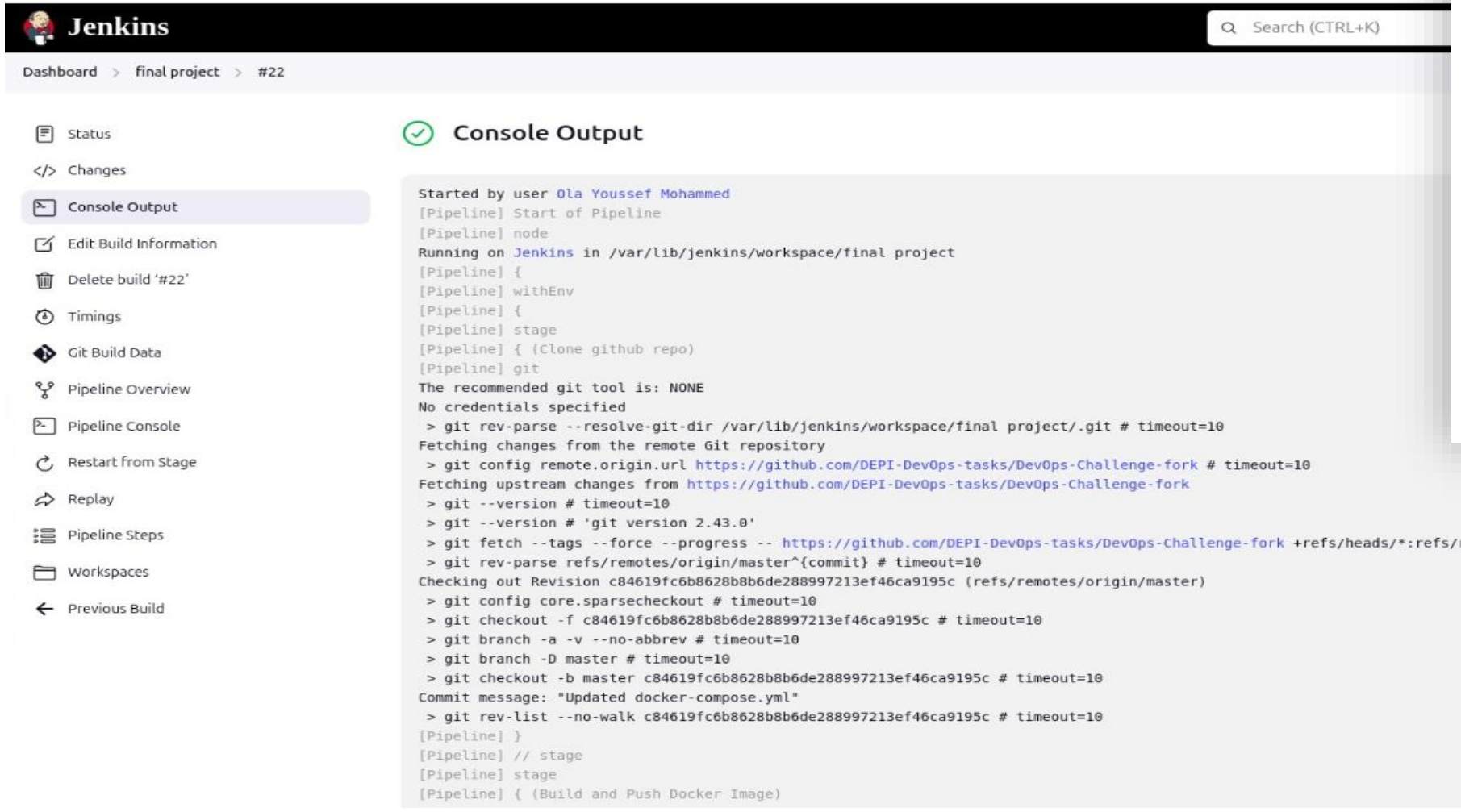
Phase 2: Jenkins & CI Integration

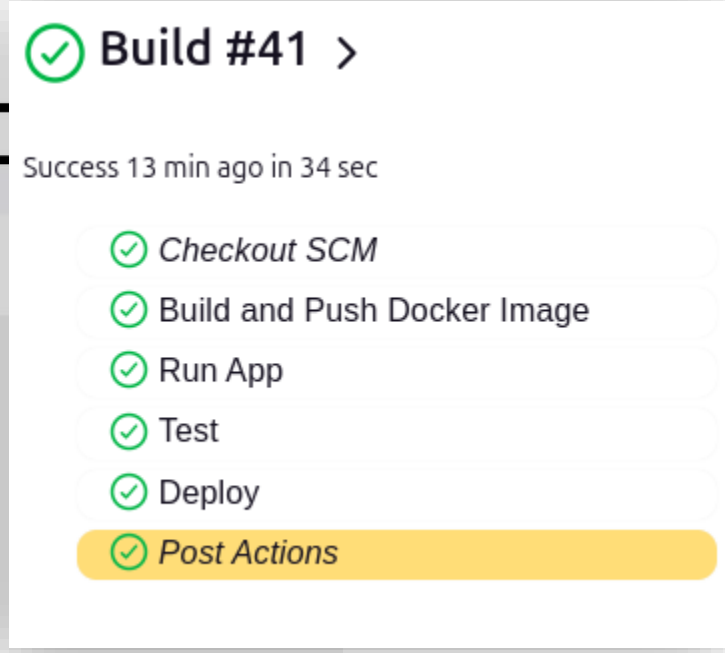
- **Build Docker images and run automated tests, configure notifications for build results**
- **Push Docker images to Docker Hub or a private registry**
- **Automate application deployment.**





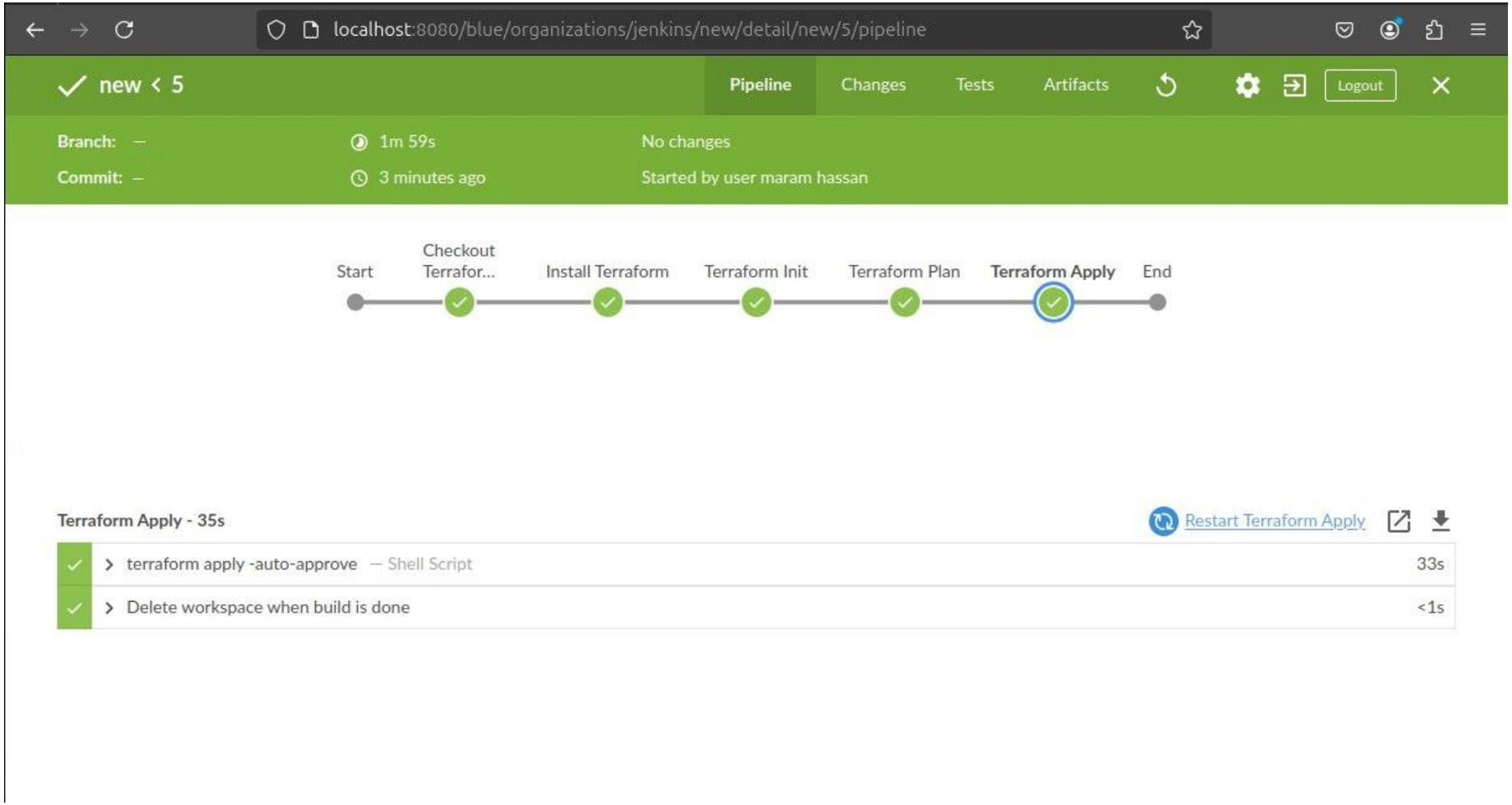
Phase 2: Jenkins & CI Integration



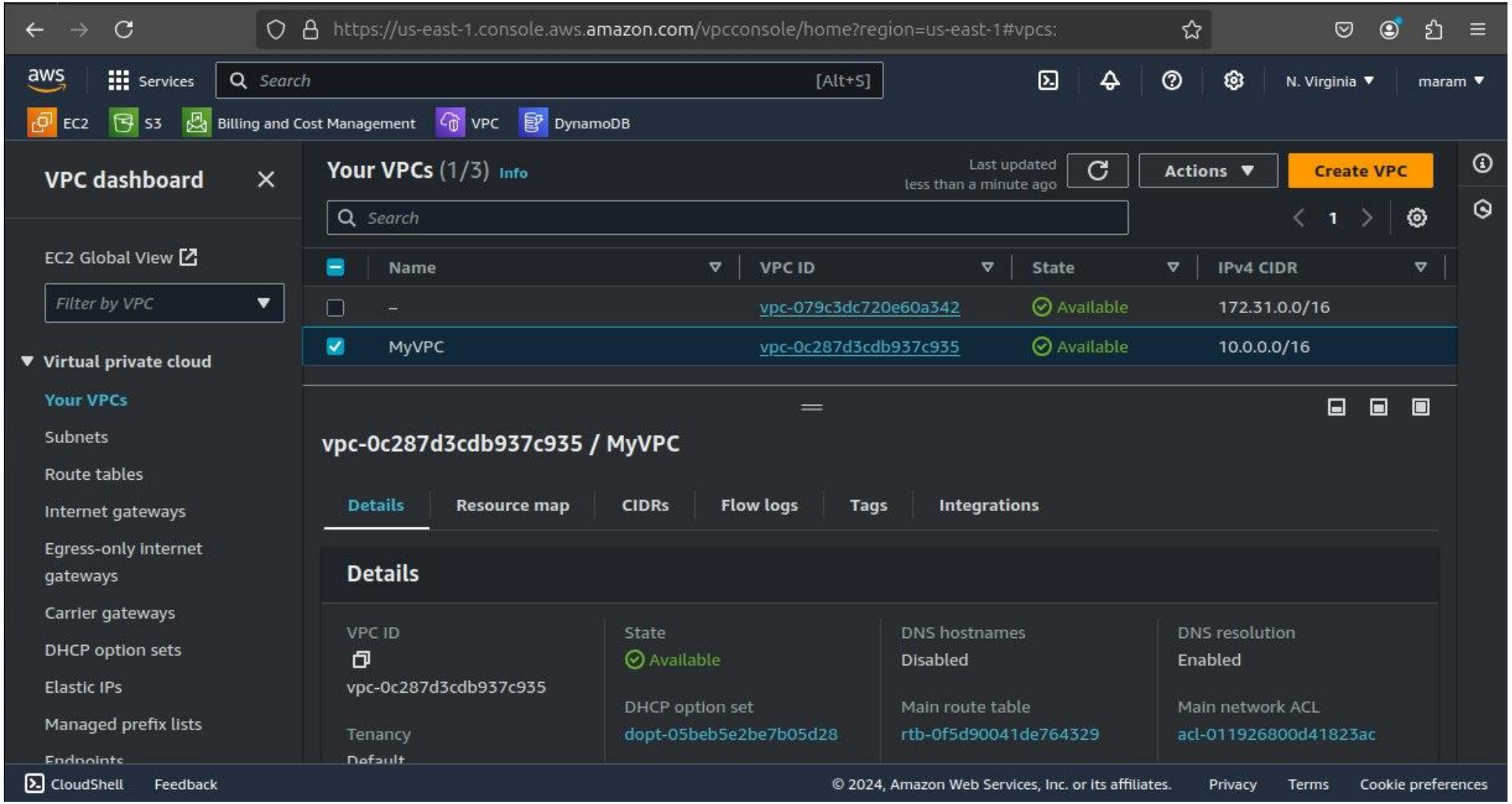


Phase 3: Infrastructure Provisioning with Terraform

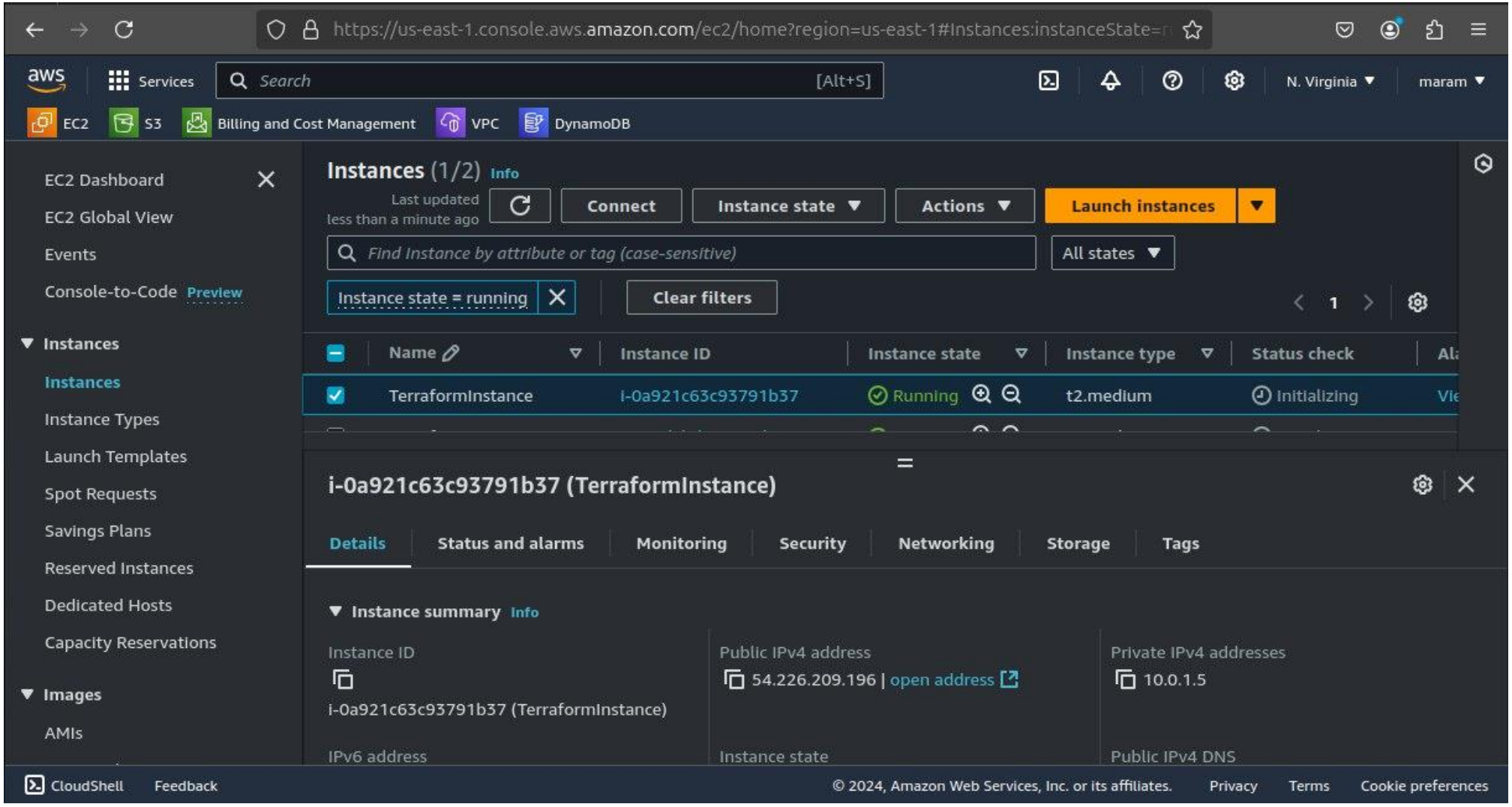
- This phase used Terraform to provision AWS resources for the application’s deployment.



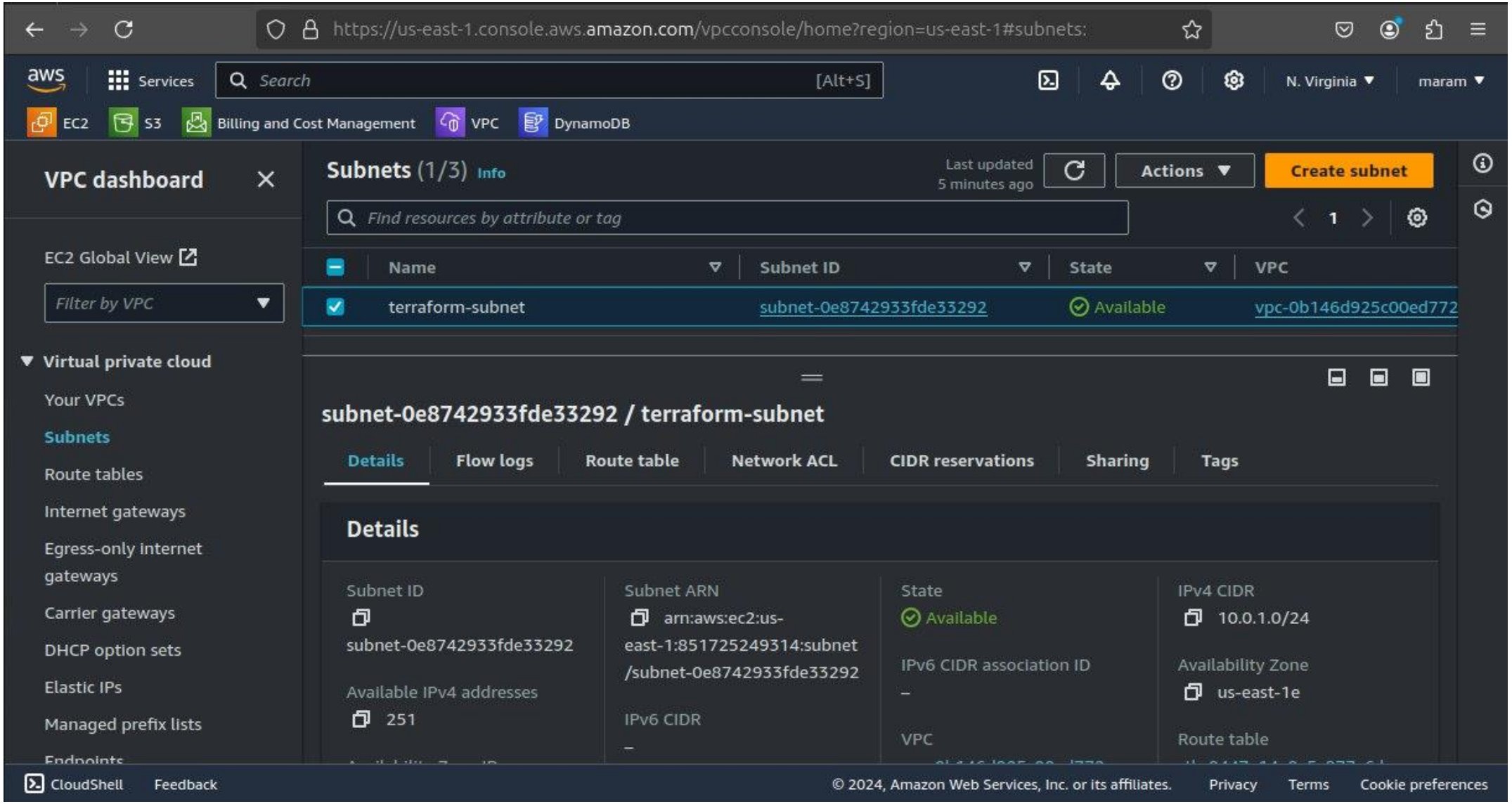
Phase 3: Infrastructure Provisioning with Terraform



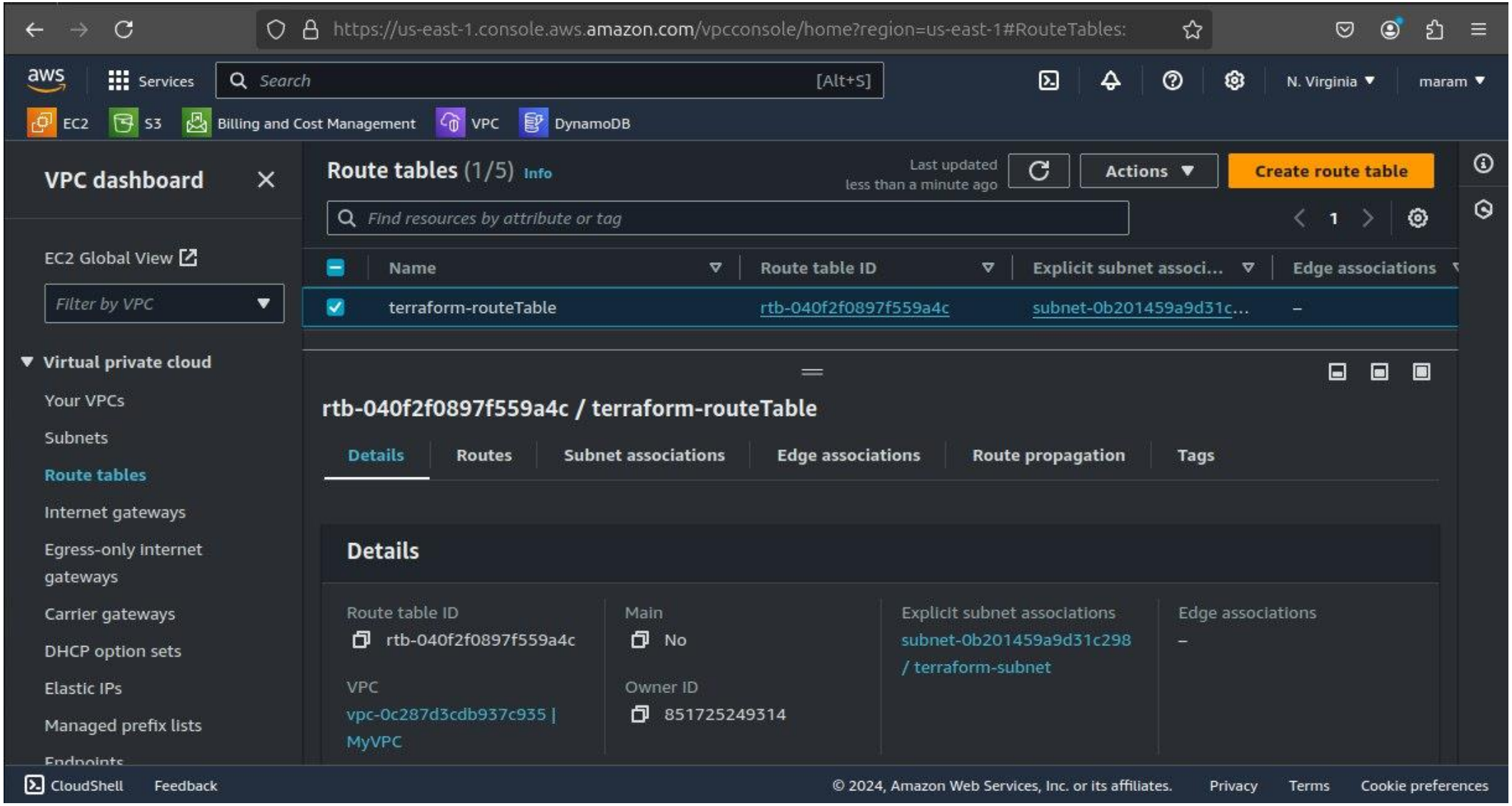
Phase 3: Infrastructure Provisioning with Terraform



Phase 3: Infrastructure Provisioning with Terraform



Phase 3: Infrastructure Provisioning with Terraform



Phase 4: Deployment with Ansible

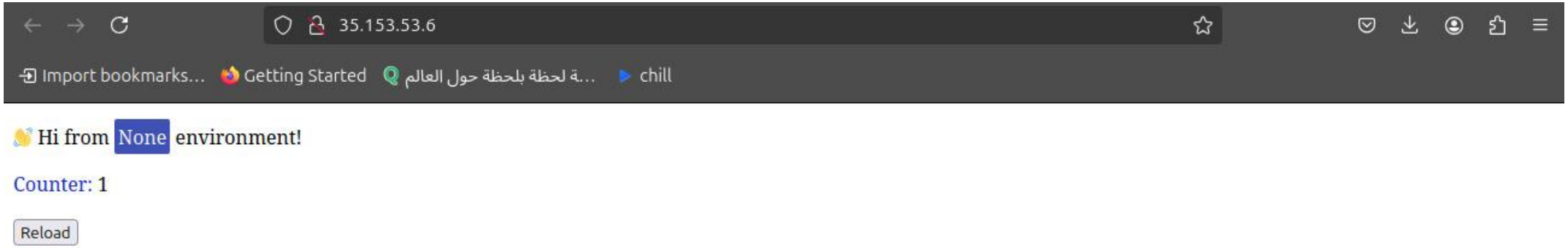
```yaml
- name: Deploy Docker Application on AWS EC2
  hosts: all
  become: true

  tasks:
    - name: Gather facts
      setup:

    - name: Update apt and install required packages
      apt:
        name: "{{ item }}"
        state: present
      with_items:
        - docker.io
        - python3-pip
        - python3-venv
        - python3-apt
        - curl  # Ensure curl is installed
        - git  # Optional: Install git if you need versi...
```

```yaml
    - name: Start Docker service
      service:
        name: docker
        state: started
        enabled: true

    - name: Create Python virtual environment
      command: python3 -m venv /home/ubuntu/venv
      args:
        creates: /home/ubuntu/venv

    - name: Install Docker Python module in virtual environment
      command: /home/ubuntu/venv/bin/python -m pip install docker

    - name: Create a directory for the application
      file:
        path: /home/ubuntu/app
        state: directory
```

Phase 4: Deployment with Ansible

```
    - name: Copy docker-compose.yml to EC2 instance
      copy:
        src: ./docker-compose.yml
        dest: /home/ubuntu/app/docker-compose.yml

    - name: Install Docker Compose
      shell: >
        curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)"
-o /usr/local/bin/docker-compose
      args:
        creates: /usr/local/bin/docker-compose

    - name: Set permissions for Docker Compose
      command: chmod +x /usr/local/bin/docker-compose

    - name: Verify Docker Compose installation
      command: docker-compose --version
      register: docker_compose_version

    - debug:
        var: docker_compose_version.stdout

    - name: Pull Docker images
      command: docker-compose -f /home/ubuntu/app/docker-compose.yml pull
      args:
        chdir: /home/ubuntu/app

    - name: Run Docker containers
      command: docker-compose -f /home/ubuntu/app/docker-compose.yml up -d
      args:
        chdir: /home/ubuntu
```

Phase 4: Deployment with Ansible



Digital Egypt Pioneers Initiative

THANK YOU

TEAM 3