Kubernetes Capstone project

Project Description: Image Deployment using Terraform, Ansible, Docker, and Kubernetes

Overview

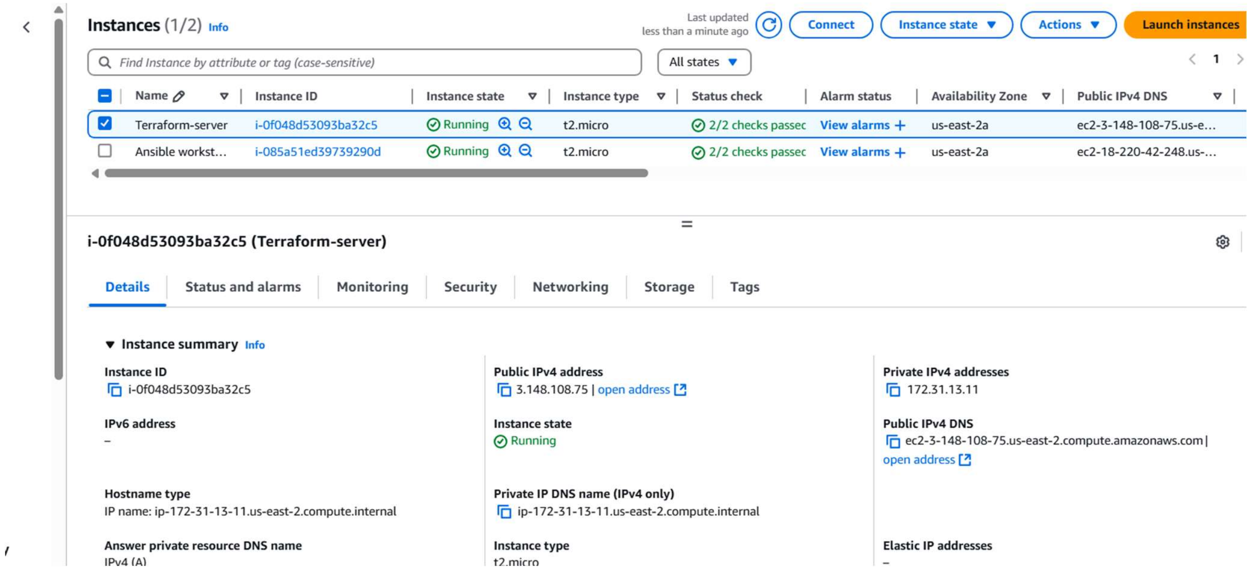
This project demonstrates the deployment of a web application using a combination of Terraform, Ansible, Docker, and Kubernetes. The primary goal is to launch and configure infrastructure and application components efficiently, showcasing the integration of these technologies.

Project Workflow

Step 1: Terraform Workstation Setup

- **Infrastructure Provisioning:**
    - Launched an Ubuntu EC2 instance (t2.micro) that serves as a Terraform workstation.
    - This instance is used to provision additional resources efficiently.

Installation of Terraform server:


Instances (1/2) Info
Last updated less than a minute ago
Connect
Instance state ▼
Actions ▼
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states ▼
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
☑	Terraform-server	i-0f048d53093ba32c5	Running	t2.micro	2/2 checks passec	View alarms +	us-east-2a	ec2-3-148-108-75.us-e...
□	Ansible workst...	i-085a51ed39739290d	Running	t2.micro	2/2 checks passec	View alarms +	us-east-2a	ec2-18-220-42-248.us-...

i-0f048d53093ba32c5 (Terraform-server)

Instance ID: i-0f048d53093ba32c5

Public IPv4 address: 3.148.108.75 | open address

Private IPv4 addresses: 172.31.13.11

IPv6 address: –

Instance state: Running

Public IPv4 DNS: ec2-3-148-108-75.us-east-2.compute.amazonaws.com | open address

Hostname type: IP name: ip-172-31-13-11.us-east-2.compute.internal

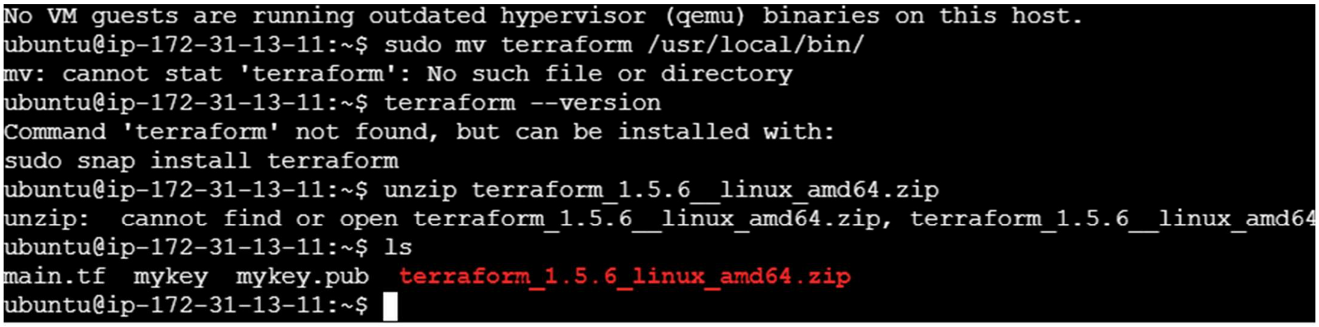
Private IP DNS name (IPv4 only): ip-172-31-13-11.us-east-2.compute.internal

Answer private resource DNS name: IPv4 (A)

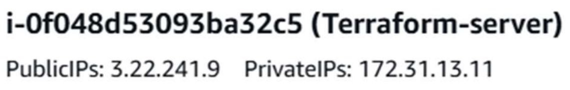
Instance type: t2.micro

Elastic IP addresses: –

Configuration of Terraform:



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-13-11:~$ sudo mv terraform /usr/local/bin/
mv: cannot stat 'terraform': No such file or directory
ubuntu@ip-172-31-13-11:~$ terraform --version
Command 'terraform' not found, but can be installed with:
sudo snap install terraform
ubuntu@ip-172-31-13-11:~$ unzip terraform_1.5.6__linux_amd64.zip
unzip:  cannot find or open terraform_1.5.6__linux_amd64.zip, terraform_1.5.6__linux_amd64
ubuntu@ip-172-31-13-11:~$ ls
main.tf  mykey  mykey.pub  terraform_1.5.6_linux_amd64.zip
ubuntu@ip-172-31-13-11:~$
```



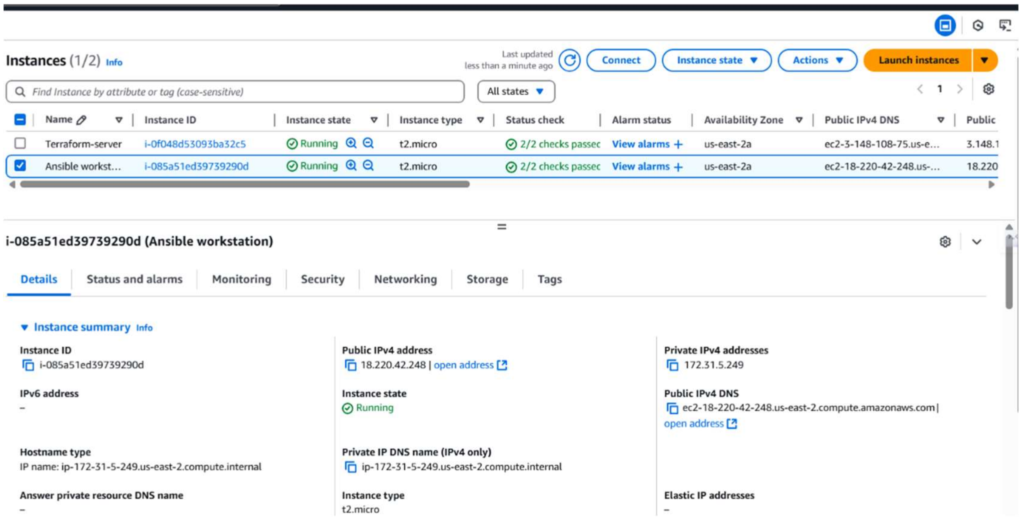
i-0f048d53093ba32c5 (Terraform-server)

PublicIPs: 3.22.241.9   PrivateIPs: 172.31.13.11

Step 2: Ansible Setup

- **Ansible Workstation Creation:**
    - From the Terraform workstation, provisioned a second EC2 instance (t2.micro, Ubuntu) to serve as an Ansible workstation.
    - Created an SSH key (using ssh-keygen) to enable secure access to the Ansible workstation.
- **Ansible Playbook Development:**
    - Developed an Ansible playbook (install_httpd.yaml) to automate the installation of the Apache web server.
    - Configured the playbook to verify functionality using a curl command to retrieve the web page from the public IP of the Apache server.

Launching of Ansible server using Terraform:



▬	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
□	Terraform-server	i-0f048d53093ba32c5	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-2a	ec2-3-148-108-75.us-e...	3.148.1
☑	Ansible workst...	i-085a51ed39739290d	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-2a	ec2-18-220-42-248.us-...	18.220

i-085a51ed39739290d (Ansible workstation)

Instance ID
i-085a51ed39739290d

Public IPv4 address
18.220.42.248 | open address

Private IPv4 addresses
172.31.5.249

IPv6 address
–

Instance state
✓ Running

Public IPv4 DNS
ec2-18-220-42-248.us-east-2.compute.amazonaws.com | open address

Hostname type
IP name: ip-172-31-5-249.us-east-2.compute.internal

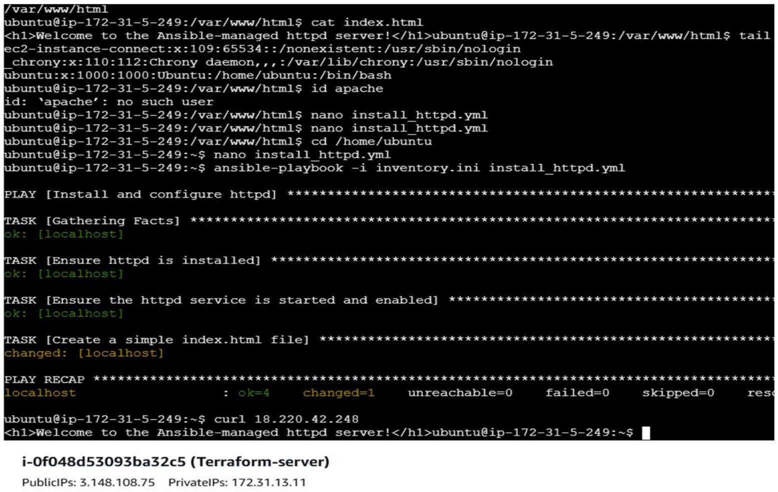
Private IP DNS name (IPv4 only)
ip-172-31-5-249.us-east-2.compute.internal

Answer private resource DNS name
–

Instance type
t2.micro

Elastic IP addresses
–

Configuration of Ansible server:

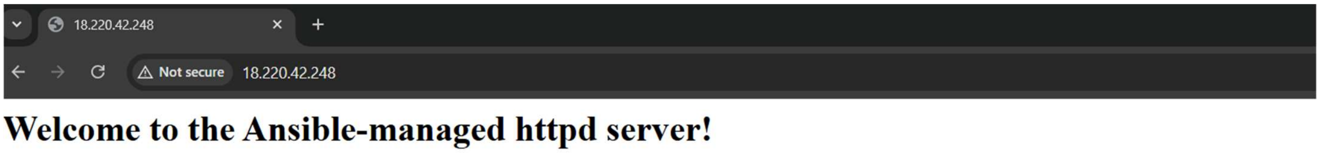


```
/var/www/html
ubuntu@ip-172-31-5-249:/var/www/html$ cat index.html
<h1>Welcome to the Ansible-managed httpd server!</h1>ubuntu@ip-172-31-5-249:/var/www/html$ tail
ec2-instance-connect:x:109:65534::/nonexistent:/usr/sbin/nologin
_chrony:x:110:112:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
ubuntu@ip-172-31-5-249:/var/www/html$ id apache
id: 'apache': no such user
ubuntu@ip-172-31-5-249:/var/www/html$ nano install_httpd.yml
ubuntu@ip-172-31-5-249:/var/www/html$ nano install_httpd.yml
ubuntu@ip-172-31-5-249:/var/www/html$ cd /home/ubuntu
ubuntu@ip-172-31-5-249:~$ nano install_httpd.yml
ubuntu@ip-172-31-5-249:~$ ansible-playbook -i inventory.ini install_httpd.yml

PLAY [Install and configure httpd] *************************************************************

TASK [Gathering Facts] *************************************************************************
ok: [localhost]

TASK [Ensure httpd is installed] ***************************************************************
ok: [localhost]

TASK [Ensure the httpd service is started and enabled] *****************************************
ok: [localhost]

TASK [Create a simple index.html file] *********************************************************
changed: [localhost]

PLAY RECAP *************************************************************************************
localhost                  : ok=4    changed=1    unreachable=0    failed=0    skipped=0    resc

ubuntu@ip-172-31-5-249:~$ curl 18.220.42.248
<h1>Welcome to the Ansible-managed httpd server!</h1>ubuntu@ip-172-31-5-249:~$
```

i-0f048d53093ba32c5 (Terraform-server)

PublicIPs: 3.148.108.75   PrivateIPs: 172.31.13.11

Web page of Ansible:

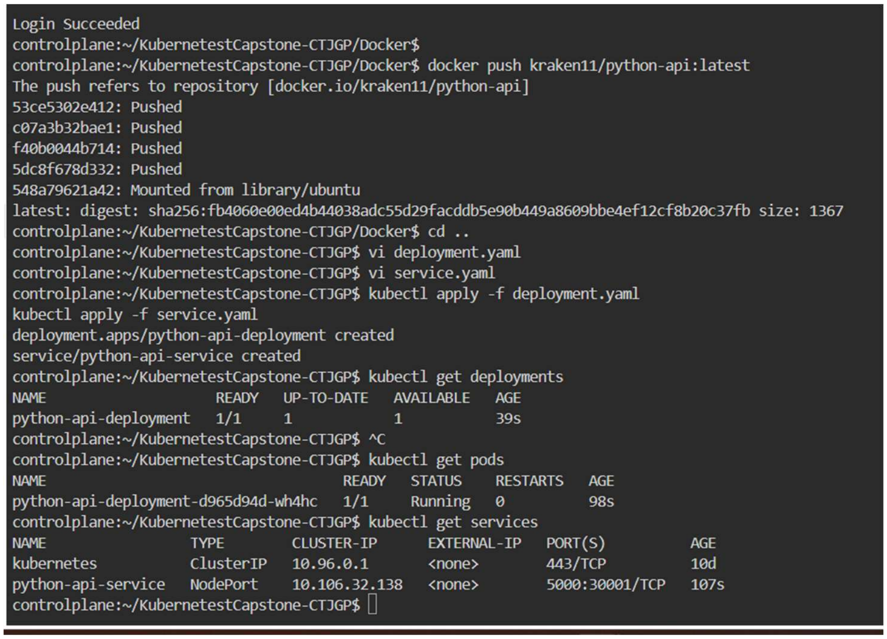


Welcome to the Ansible-managed httpd server!

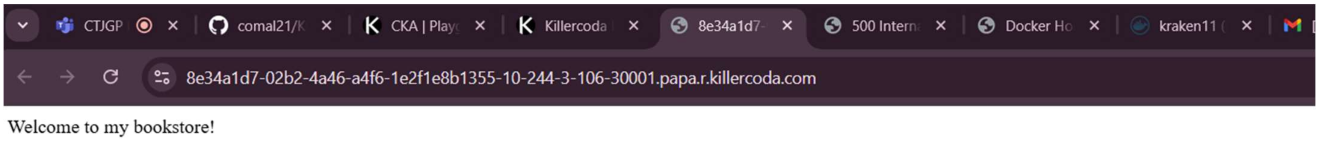
Step 3: Docker & Kubernetes Deployment

- **Docker Image Creation:**
  - Built a Docker image for a Python API using a provided Dockerfile.
  - Included necessary files: Dockerfile, requirements.txt, and the Python API code, all within the same directory.
  - Successfully pushed the Docker image to DockerHub, making it accessible for deployment.
- **Kubernetes Configuration:**
  - Created a Kubernetes pod using the Docker image from DockerHub, enabling the Python API to run in a scalable and resilient environment.
  - Configured a NodePort service to expose the application, allowing external access through specified ports.

Docker & Kubernetes Task:



```
Login Succeeded
controlplane:~/KubernetestCapstone-CTJGP/Docker$
controlplane:~/KubernetestCapstone-CTJGP/Docker$ docker push kraken11/python-api:latest
The push refers to repository [docker.io/kraken11/python-api]
53ce5302e412: Pushed
c07a3b32bae1: Pushed
f40b0044b714: Pushed
5dc8f678d332: Pushed
548a79621a42: Mounted from library/ubuntu
latest: digest: sha256:fb4060e00ed4b44038adc55d29facddb5e90b449a8609bbe4ef12cf8b20c37fb size: 1367
controlplane:~/KubernetestCapstone-CTJGP/Docker$ cd ..
controlplane:~/KubernetestCapstone-CTJGP$ vi deployment.yaml
controlplane:~/KubernetestCapstone-CTJGP$ vi service.yaml
controlplane:~/KubernetestCapstone-CTJGP$ kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
deployment.apps/python-api-deployment created
service/python-api-service created
controlplane:~/KubernetestCapstone-CTJGP$ kubectl get deployments
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
python-api-deployment   1/1     1            1           39s
controlplane:~/KubernetestCapstone-CTJGP$ ^C
controlplane:~/KubernetestCapstone-CTJGP$ kubectl get pods
NAME                                     READY   STATUS    RESTARTS   AGE
python-api-deployment-d965d94d-wh4hc    1/1     Running   0          98s
controlplane:~/KubernetestCapstone-CTJGP$ kubectl get services
NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes           ClusterIP   10.96.0.1       <none>        443/TCP          10d
python-api-service   NodePort    10.106.32.138   <none>        5000:30001/TCP   107s
controlplane:~/KubernetestCapstone-CTJGP$ 
```


CTJGP
comal21/K
CKA | Playg
Killercoda
8e34a1d7-
500 Intern
Docker Ho
kraken11 (
8e34a1d7-02b2-4a46-a4f6-1e2f1e8b1355-10-244-3-106-30001.papa.r.killercoda.com

Welcome to my bookstore!

Conclusion

This project effectively showcases the workflow for deploying a web application using modern DevOps tools. By leveraging Terraform for infrastructure automation, Ansible for configuration management, and Docker and Kubernetes for containerization and orchestration, we can achieve a robust and scalable deployment strategy.