Team name: Sudoers

Ahmed EL-Nemr
Hossam Gamal
Mahmoud Khairy
Wageeh Saad

Group Name: YAT224_CAI1_SWD1_G6D
Instructor Name: Islam Reda

Project name:
Automated Deployment Pipeline with Jenkins and Docker

Project Documentation

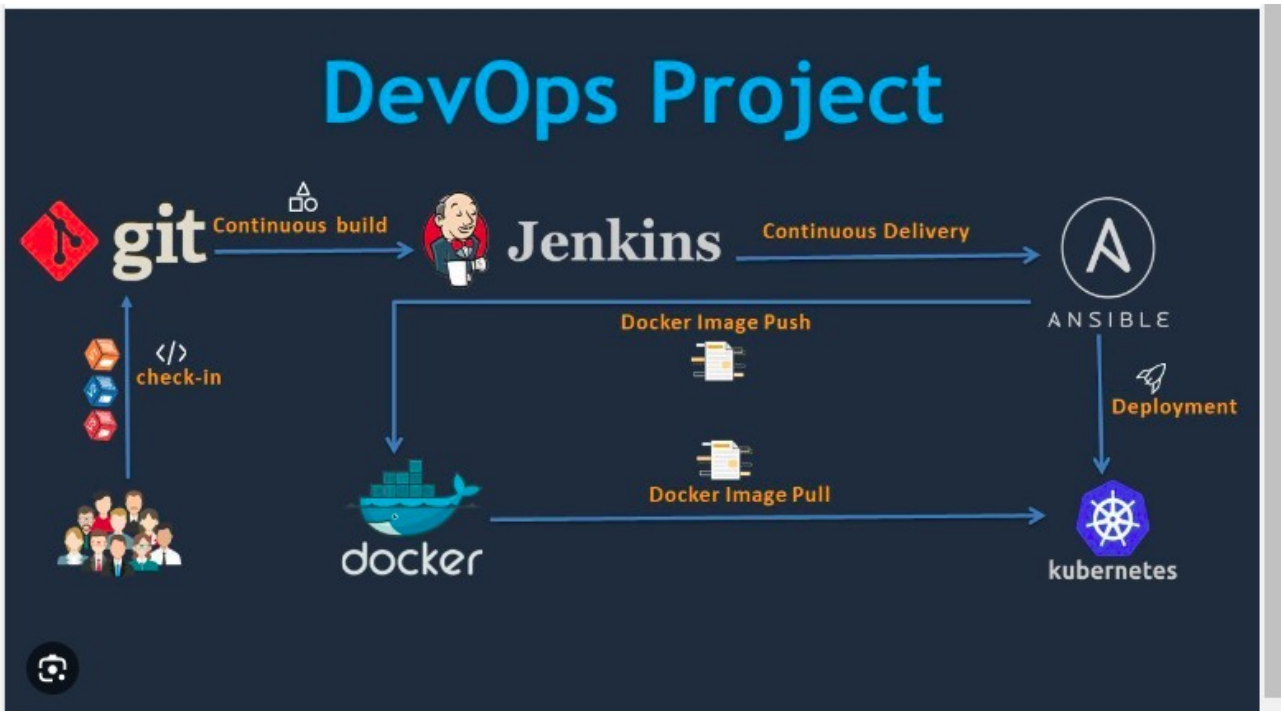


Table of Contents

Project Description....3
Prerequisites....4
Infrastructure....6
Network directory....6
Security directory....7
Compute directory....7
Main directory....8
Creating Repository at Git Hub....9
Create a basic Dockerized application:....10
Application:....12
Setup Ansible....12
Kubernetes Integration:....13
Kubernetes Deployment (YAML File)....14
Kubernetes Namespace (YAML File)....14
Kubernetes LoadBalancer (YAML File)....15
Jenkins CI/CD....16
Create Push/Deploy Pipeline Script....16
Key Features:....16
Notes on Specific Stages:....18
Error Handling:....18
#ACCESS REMOTE REPO....19
#TERRAFORM INIT & ACTION....19
#TEST INVENTORY....20
#TEST SSH....20
#RUN ANSIBLE PLAYBOOK....21
#COPY K8SMANIFEST to EC2....21
#Build and Tag Docker Image....21
#Configure Docker & Login....22
#Push Docker Image to Docker Hub....22
Deploy App on Minikube....23
#DESTROY COMMAND....23
References....25

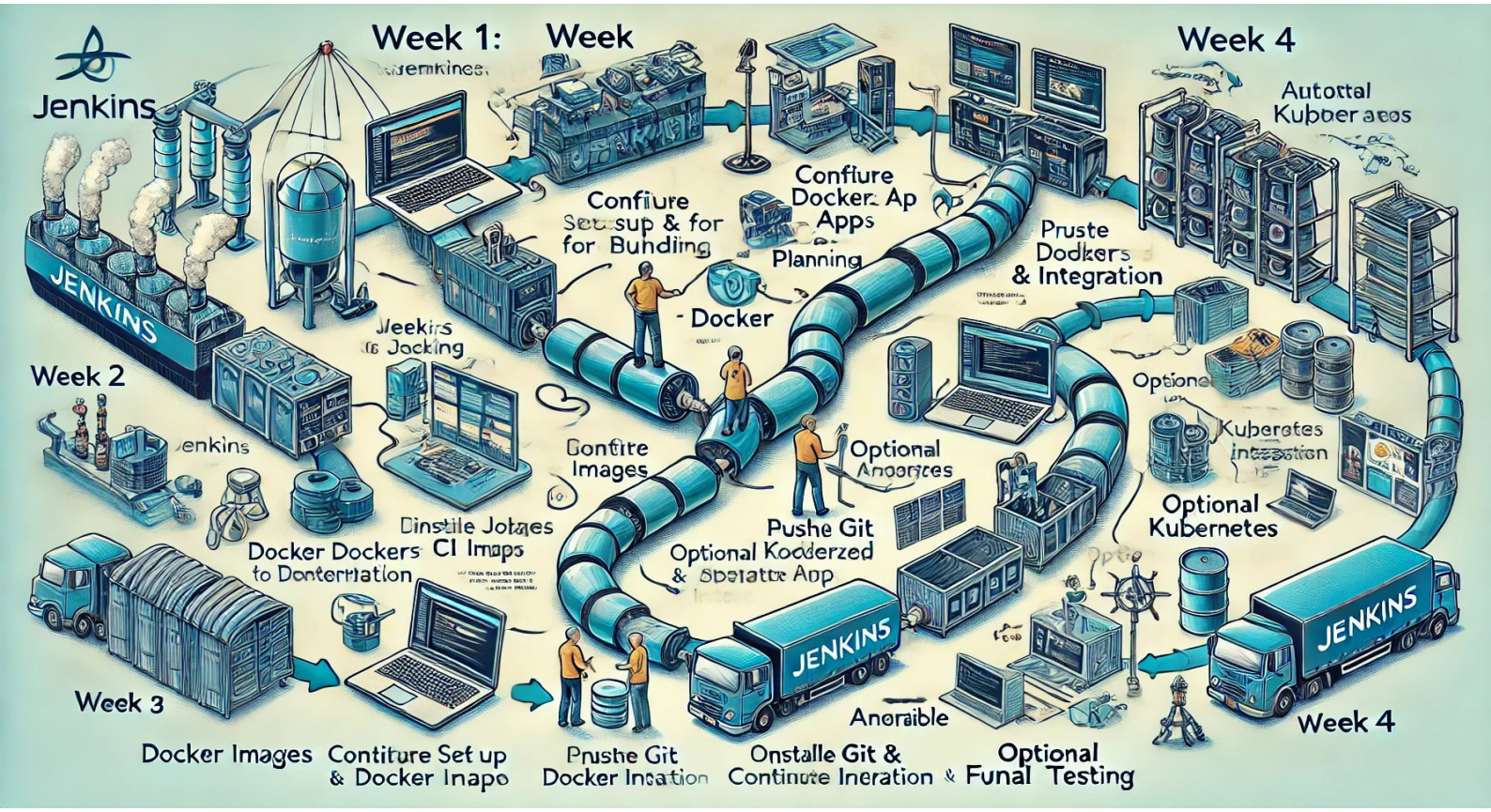
Project Description

Objective: Implement an automated CI/CD pipeline using Jenkins, Docker, and Ansible.

Description: Develop a pipeline to automate the build, testing, and deployment of a sample application. Utilize Jenkins for continuous integration, Docker for containerization, and Ansible for configuration management. Implement automated testing and deployment to a cloud []environment.

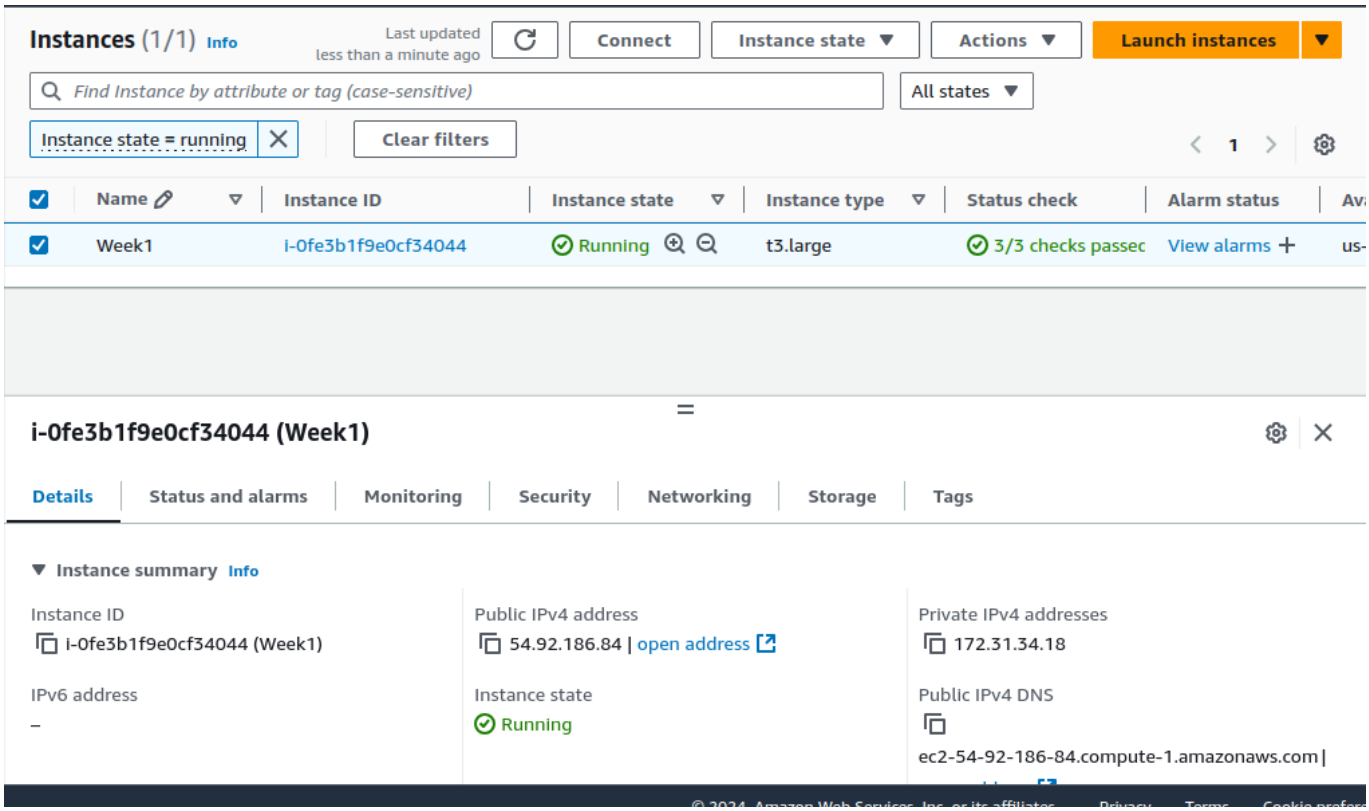
Technologies to use:

1. **Jenkins:** Automation orchestration for CI/CD pipeline.
2. **Ansible:** Configuration Management Tool
3. **Docker:** Containerization for consistency and portability.
4. **Kubernetes:** Container orchestration for deployment.
5. **Gmail /Slack Integration:** Email / Slack notifications for pipeline status.
6. **Prometheus and Grafana:** Monitoring and visualization of system metrics.
7. **AWS :** Creating virtual machines



Prerequisites

1- create & Launch EC2 (Ubuntu) on AWS Cloud



2- SSH on EC2

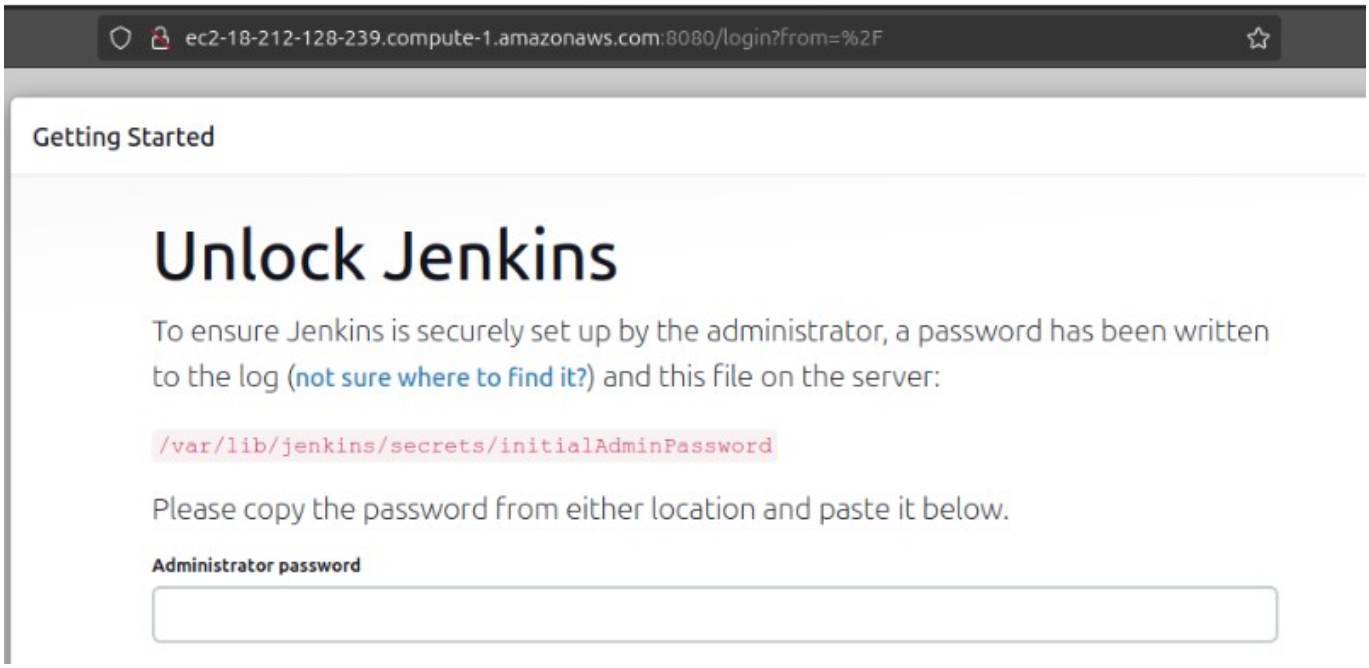
```
ahmed@ahmed-HP-ProBook-450-G6:~/Downloads$ ssh -i "ssh2.pem" ubuntu@ec2-18-212-128-239.compute-1.amazonaws.com
```

3-Using Jenkins Documentations to install Jenkins

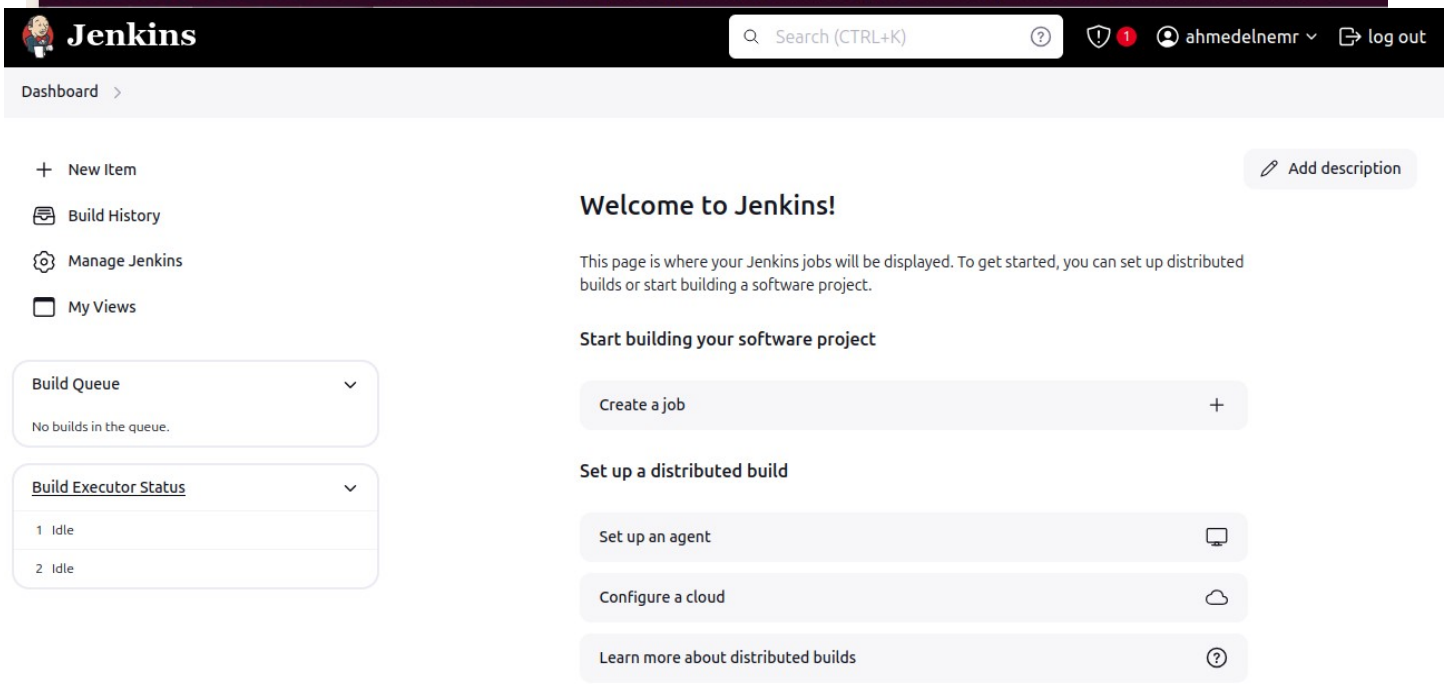
```
ubuntu@ip-172-31-34-18:~$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

4- Enable, start & check status of Jenkins

```
ubuntu@ip-172-31-34-18:~$ sudo snap install openjdk
openjdk 23+37 from John Neffenger (jgneff) installed
ubuntu@ip-172-31-34-18:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-34-18:~$ sudo systemctl start jenkins
ubuntu@ip-172-31-34-18:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-10-13 06:37:07 UTC; 10s ago
   Main PID: 4525 (java)
      Tasks: 45 (limit: 1130)
     Memory: 405.1M (peak: 419.1M)
        CPU: 11.567s
     CGroup: /system.slice/jenkins.service
             └─4525 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --
```




```
ubuntu@ip-172-31-34-18:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



Infrastructure

Start by creating the infrastructure using Terraform on AWS environment. we will run the terraform script

Network directory

create new VPC

```
resource "aws_vpc" "my-vpc" {
  cidr_block = var.cidr_block
  tags = {
    Name = "my-vpc"
  }
}
```

create internet Gateway

```
resource "aws_internet_gateway" "my_igw" {
  vpc_id = aws_vpc.my-vpc.id
  tags = {
    Name = "igw"
  }
}
```

create route table

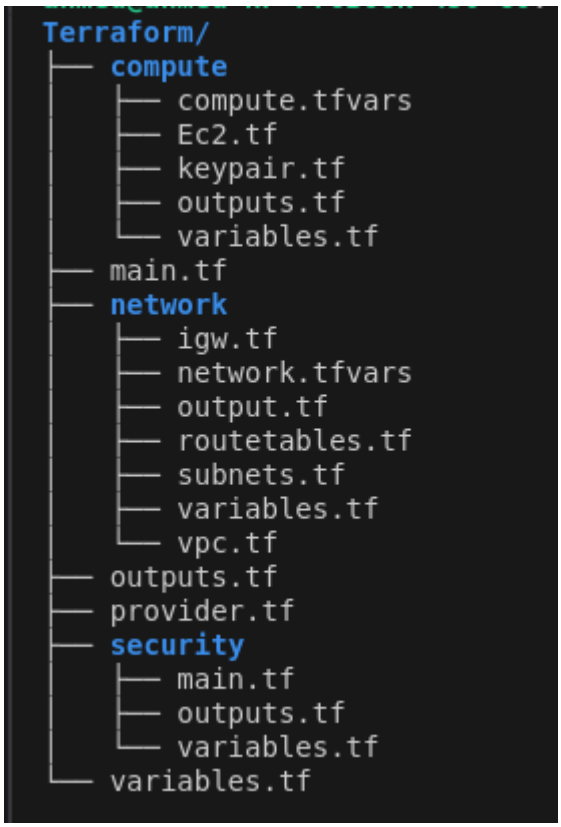
```
resource "aws_route_table" "my_route_public" {
  vpc_id = aws_vpc.my-vpc.id
}
resource "aws_route" "public_route" {
route_table_id = aws_route_table.my_route_public.id
destination_cidr_block = "0.0.0.0/0"
gateway_id = aws_internet_gateway.my_igw.id
}
resource "aws_route_table_association" "public_subnet_assoc" {
subnet_id = aws_subnet.public-subnet.id
route_table_id = aws_route_table.my_route_public.id
}
```

create subnets

```
resource "aws_subnet" "public-subnet" {
  vpc_id     = aws_vpc.my-vpc.id
  cidr_block = var.public_subnet_cidr
  map_public_ip_on_launch = true
  tags = {
    Name = "public-subnet"
  }
}
```

variables file

```
variable "cidr_block" {}
variable "public_subnet_cidr" {}
```



output file

```
output "public-subnet" {
  value = aws_subnet.public-subnet.id
}
output "vpc-id" {
  value = aws_vpc.my-vpc.id
}
```

Network variables

```
cidr_block = "10.0.0.0/16"
public_subnet_cidr = "10.0.1.0/24"
private_subnet_cidr = "10.0.2.0/24"
```

Security directory

main file

```
resource "aws_security_group" "public" {
  name        = "allow_ssh"
  description = "Allow ssh inbound traffic and all outbound traffic"
  vpc_id      = var.vpc_id
  ingress {
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port = 0
    to_port = 0
    protocol = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "allow_ssh"
  }
}
```

security variables file

```
variable "vpc_id" {}
variable "cidr_block" {}
```

security output file

```
output "public" {
value = aws_security_group.public.id
}
```

Compute directory

create EC2 on AWS Cloud_

```
resource "aws_instance" "appdep" {
  ami               = var.ami_id
  instance_type     = var.instance_type
  security_groups   = [var.public_SG]
  subnet_id         = var.public-subnet
  associate_public_ip_address = true
  key_name          = aws_key_pair.tf-key-pairz.id

  user_data = <<-EOF
    #!/bin/bash
    echo '${tls_private_key.rsa-key.private_key_pem}' > /home/ec2-user/private-key.pem
    chmod 400 /home/ec2-user/private-key.pem
  EOF

  provisioner "local-exec" {
    command = <<-EOC
      echo "[all]" > inventory.txt
      echo "${self.public_ip} ansible_user=ec2-user ansible_ssh_private_key_file=/home/ec2-user/private-key.pem"
>> inventory.txt
    EOC
  }

  tags = {
    Name = "App EC2"
  }
}
```

keypair file

```
#Generate public private key pair
resource "tls_private_key" "rsa-key" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "aws_key_pair" "tf-key-pairz" {
  key_name   = "tf-key-pairz"
  public_key = tls_private_key.rsa-key.public_key_openssh
}

resource "local_file" "tf-key" {
  content  = tls_private_key.rsa-key.private_key_pem
  filename = "tf-key-pairz.pem"
}
```

variables file

```
variable "ami_id" {}
variable "instance_type" {}
variable "public_SG" {}
variable "public-subnet" {}
```

Main directory

main file

```
module "network" {
  source             = "./network"
  cidr_block         = var.cidr_block
  public_subnet_cidr  = var.public_subnet_cidr
}
module "security" {
  source = "./security"
  vpc_id = module.network.vpc-id
  cidr_block = var.cidr_block
}
module "compute" {
  source = "./compute"
  ami_id = var.ami_id
  instance_type = var.instance_type
  public_SG = module.security.public
  public-subnet = module.network.public-subnet
}
```

main Provider file

```
provider "aws" {
  region = var.region
}
```

main output file

```
output "AppEc2" {
value = module.compute.AppEc2}
```

compute output file

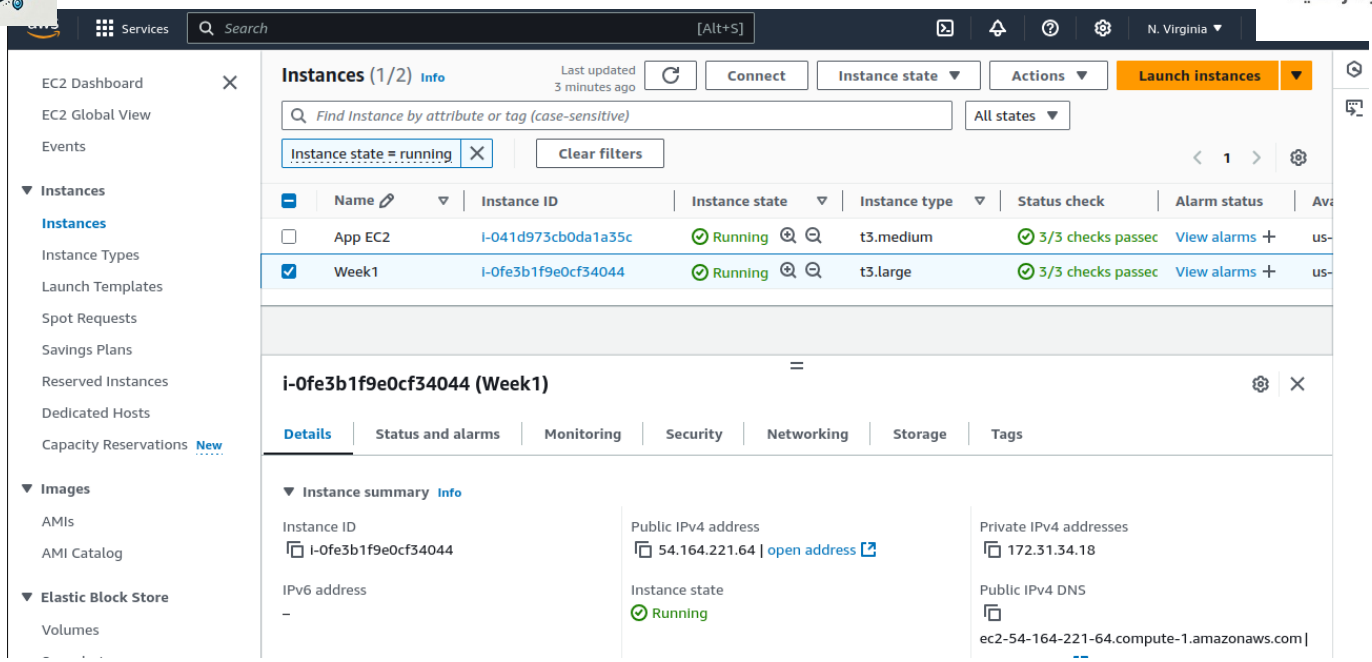
```
output "AppEc2" {
value = aws_instance.appdep.id
}
```

compute tfvars file

```
ami_id = "ami-0fff1b9a61dec8a5f"
instance_type = "t3.micro"
```

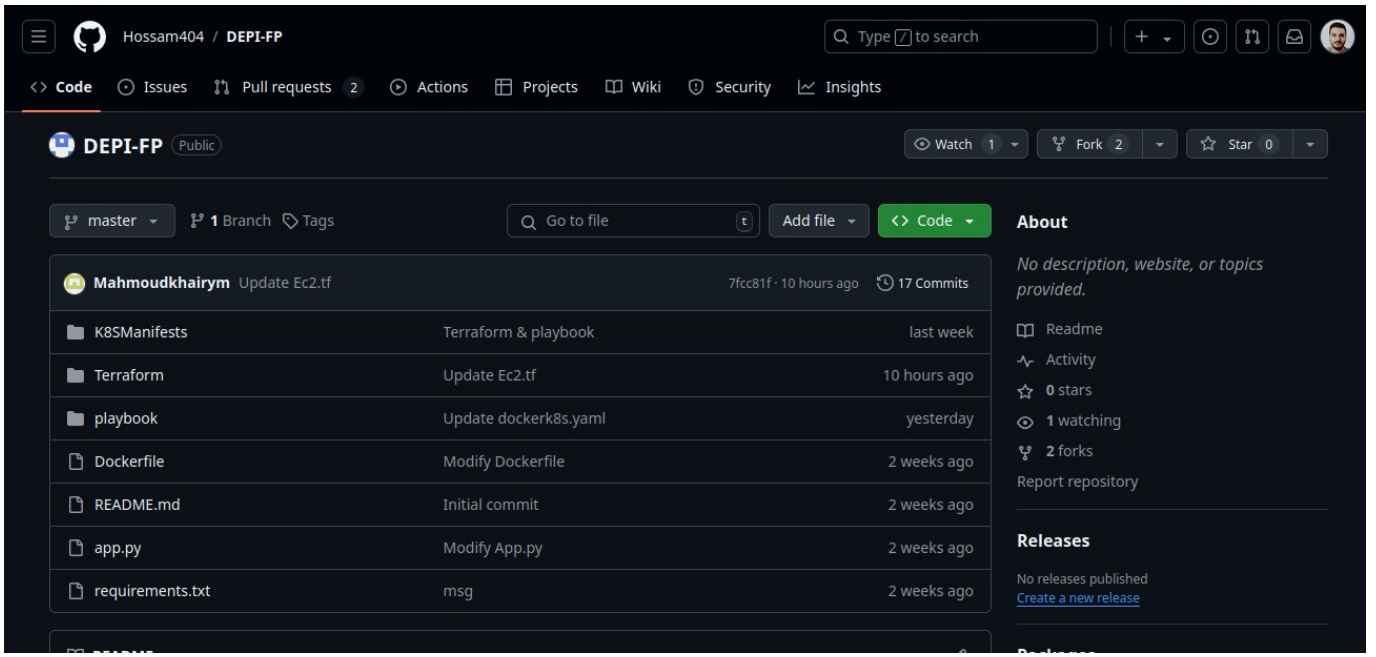
main Variables file

```
variable "region" {
type = string
default = "us-east-1"
}

variable "cidr_block" {
type = string
default = "10.0.0.0/16"

}

variable "public_subnet_cidr" {
  type = string
  description = "cidr range of public subnet"
}

variable "ami_id" {
  type = string
  description = "ami id for ec2 amazon linux"

}

variable "instance_type" {

  type = string
  description = "es2 instance type"
}
```



Creating Repository at Git Hub

Integrate Git: Set up GitHub repository and integrate with Jenkins for continuous integration



Name	Last commit message	Last commit da...
..		
compute	Update Ec2.tf	10 hours ago
network	Terraform & playbook	last week
security	Terraform & playbook	last week
main.tf	Terraform & playbook	last week
outputs.tf	Terraform & playbook	last week
provider.tf	Terraform & playbook	last week
variables.tf	Terraform & playbook	last week

-Devops planning Engineer pull #Infrastructure as a code (Terraform) using Git Code

Create a basic Dockerized application:

Develop a simple application (e.g., a webapp) and Dockerize it.

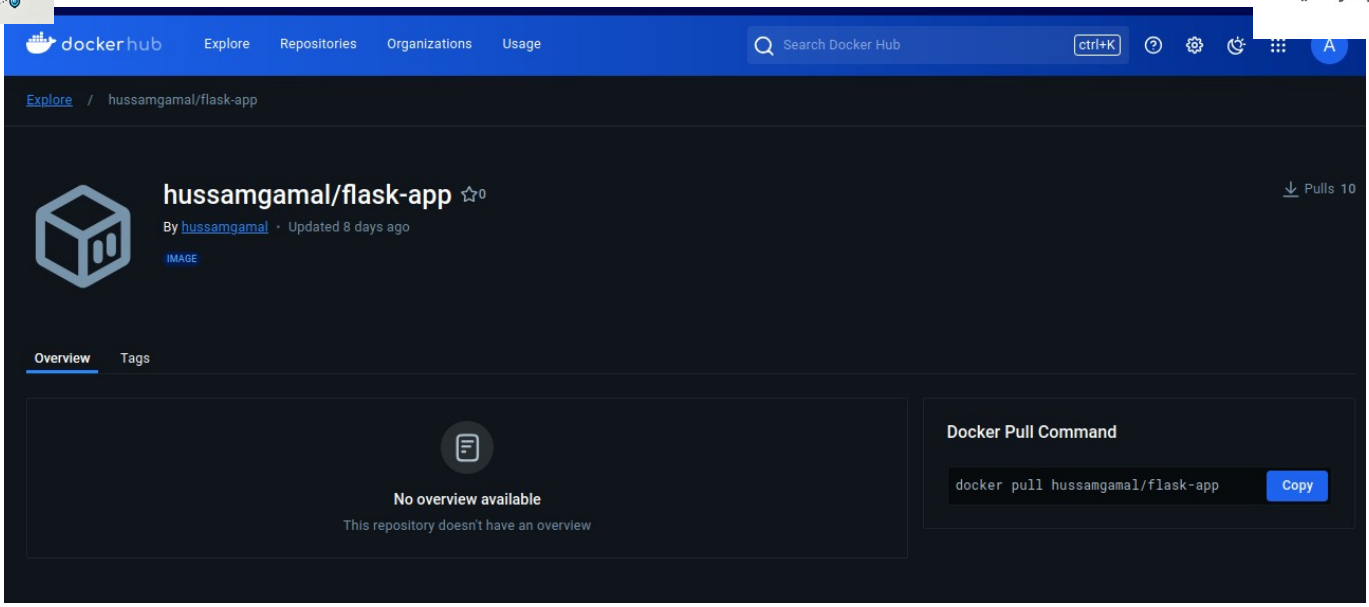
1- Clone the Python Web App from GitHub

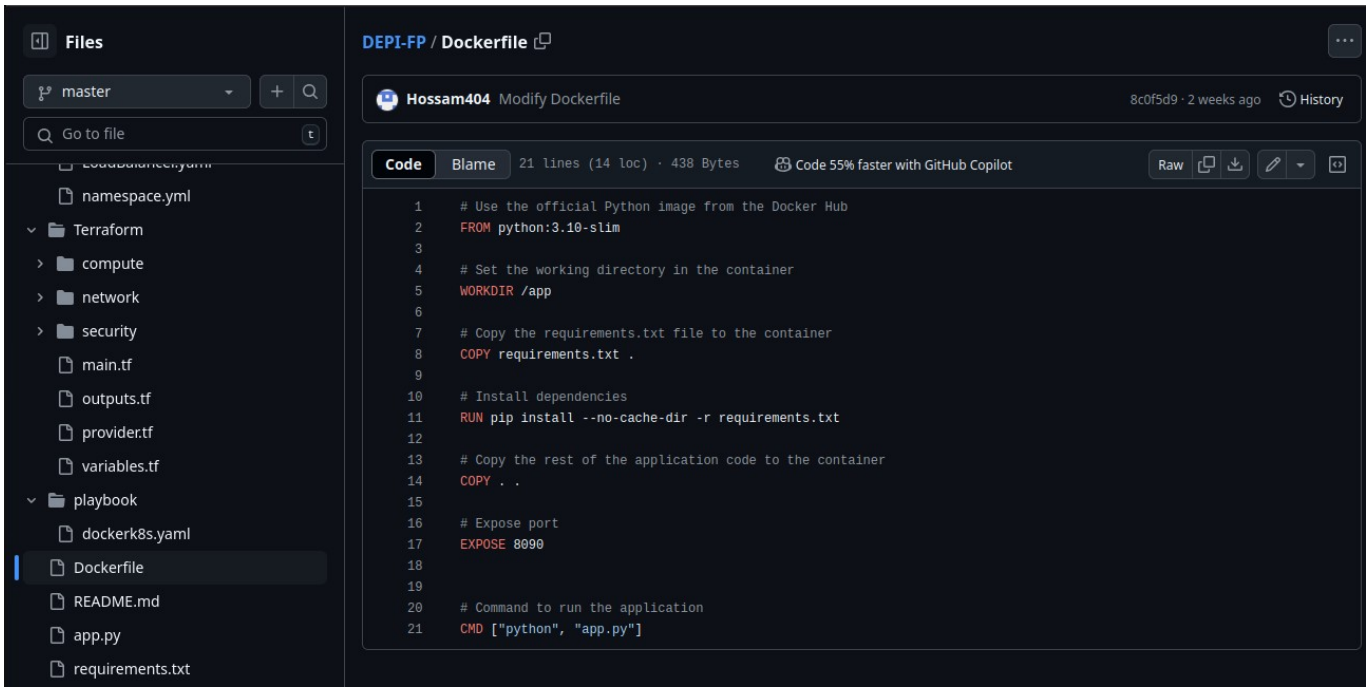
https://github.com/Mahmoudkhairym/depi-proj.git

```
ubuntu@ip-172-31-34-18:~$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: ****************************************
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as Ahmedelnemr35
ubuntu@ip-172-31-34-18:~$ gh repo clone Mahmoudkhairym/depi-proj
Cloning into 'depi-proj'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 12 (delta 0), reused 8 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (12/12), done.
```

```
ubuntu@ip-172-31-34-18:~/depi-proj$ sudo docker build -t depi-proj/app .
[+] Building 10.9s (10/10) FINISHED                                    docker:default
 => [internal] load build definition from Dockerfile                           0.1s
 => => transferring dockerfile: 185B                                           0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim            0.6s
 => [internal] load .dockerignore                                              0.0s
 => => transferring context: 2B                                                0.0s
 => [1/5] FROM docker.io/library/python:3.10-slim@sha256:1eb5d76bf3e9e612176e  5.5s
 => => resolve docker.io/library/python:3.10-slim@sha256:1eb5d76bf3e9e612176e  0.0s
 => => sha256:1eb5d76bf3e9e612176ebf5eadf8f27ec300b7b4b9a99f5856f8232fd33aa16e 9.13kB / 9.13kB   0.0s
 => => sha256:d4f909c76cb3a81469556d25f2baa3a419fe86ef94ef90d5063cf225ab3cb6ac 1.75kB / 1.75kB   0.0s
 => => sha256:467b0be1afa53d27ebd128ed6177ab16921dfbc01da7312e143cfe4de9f57f88 5.22kB / 5.22kB   0.0s
 => => sha256:302e3ee498053a7b5332ac79e8efebec16e900289fc1ecd1c754ce8fa047fcab 29.13MB / 29.13MB 1.6s
 => => sha256:eeec0daa4c1f7c8a5ea1f02ac1c8813e867ebea7df1dfc0db19d59c69cdc217e 3.51MB / 3.51MB   0.3s
```

```
^[[B^[[A^Cubuntu@ip-172-31-34-18:~/depi-proj$ sudo docker run -p 5000:5000 depi-proj/app .
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
41.234.46.80 - - [13/Oct/2024 08:05:52] "GET / HTTP/1.1" 200 -
41.234.46.80 - - [13/Oct/2024 08:06:02] "GET /how%20are%20you HTTP/1.1" 200 -
```


dockerhub
Explore
Repositories
Organizations
Usage
Search Docker Hub
ctrl+K
Explore / hussamgamal/flask-app
hussamgamal/flask-app
By hussamgamal · Updated 8 days ago
IMAGE
Pulls 10
Overview
Tags
No overview available
This repository doesn't have an overview
Docker Pull Command
docker pull hussamgamal/flask-app
Copy


Files
master
Go to file
namespace.yml
Terraform
compute
network
security
main.tf
outputs.tf
provider.tf
variables.tf
playbook
dockerk8s.yaml
Dockerfile
README.md
app.py
requirements.txt
DEPI-FP / Dockerfile
Hossam404 Modify Dockerfile
8c0f5d9 · 2 weeks ago
History
Code
Blame
21 lines (14 loc) · 438 Bytes
Code 55% faster with GitHub Copilot
Raw
Use the official Python image from the Docker Hub
FROM python:3.10-slim
Set the working directory in the container
WORKDIR /app
Copy the requirements.txt file to the container
COPY requirements.txt .
Install dependencies
RUN pip install --no-cache-dir -r requirements.txt
Copy the rest of the application code to the container
COPY . .
Expose port
EXPOSE 8090
Command to run the application
CMD ["python", "app.py"]

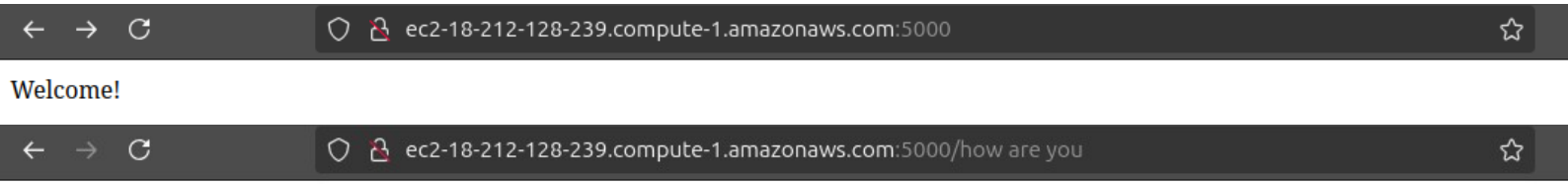
Application:

Simple Web Application platform using Flask/ Phython

- Open static web page on localhost:5000/ with “Welcome” message

-localhost:5000/how are you >> “ I am good, how about you? “

```python
import os
from flask import Flask
app = Flask(__name__)

@app.route("/")
def main():
    return "Welcome!"

@app.route('/how are you')
def hello():
    return 'I am good, how about you?'

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```



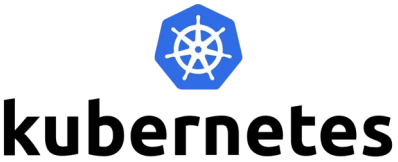
Setup Ansible

Install Ansible for configuration management.

```
ubuntu@ip-172-31-34-18:~/depi-proj$ $ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Kubernetes Integration:

configure Kubernetes to orchestrate the deployment of Docker containers if desired



by using Ansible Create Yaml . Playbook to install Docker & Kubernetes as below

```
---
- name: Setup Docker, Kubernetes, and Minikube on EC2 instance
  hosts: all
  become: true  # Use 'become' to run tasks as sudo
  tasks:
    - name: Update package index
      command: dnf makecache

    - name: Install necessary dependencies
      dnf:
        name: conntrack
        state: present

    - name: Install Docker
      dnf:
        name: docker
        state: present

    - name: Start Docker service
      systemd:
        name: docker
        state: started
        enabled: true

    - name: Download kubectl binary
      shell: |
        curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

    - name: Make kubectl binary executable
      command: chmod +x kubectl

    - name: Move kubectl to /usr/local/bin
      command: mv kubectl /usr/local/bin/

    - name: Add Kubernetes repository
      shell: |
        cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
        [kubernetes]
        name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-
x86_64
        enabled=1
```

```
          gpgcheck=1
        repo_gpgcheck=1
        gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
        EOF

    - name: Install Minikube
      shell: |
        curl -Lo minikube
https://storage.googleapis.com/minikube/releases/latest/minikube-
linux-amd64 &&
        chmod +x minikube &&
        sudo mv minikube /usr/local/bin/
```

Kubernetes Deployment (YAML File)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-k8s
  namespace: flask-app
  labels:
    app: my-app-k8s
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app-k8s
  template:
    metadata:
      labels:
        app: my-app-k8s
    spec:
      containers:
      - name: my-flask-app
        image: hussamgamal/flask-app:v1
        ports:
     - containerPort: 5000
```

Kubernetes Namespace (YAML File)

```
apiVersion: v1
kind: Namespace
metadata:
  name: flask-app
```

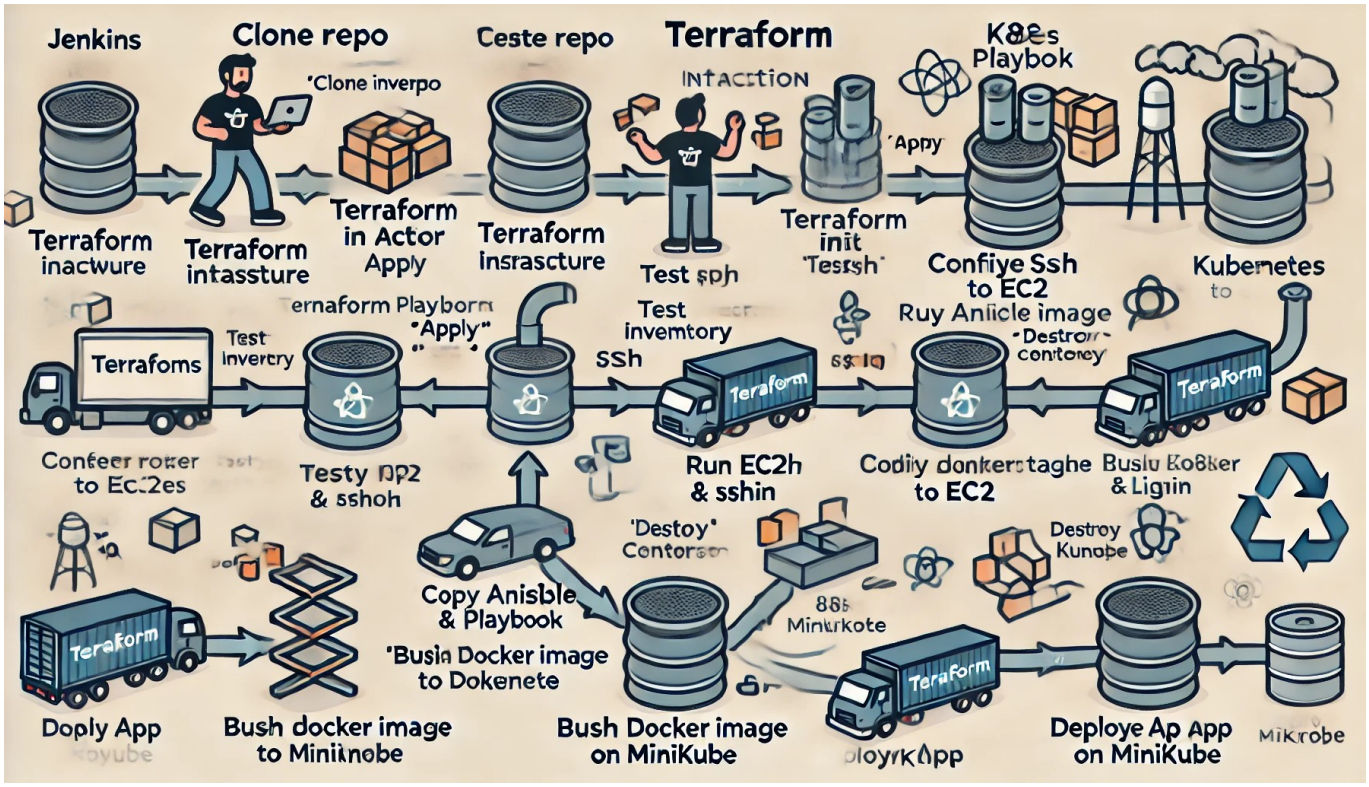
Kubernetes LoadBalancer (YAML File)

```
apiVersion: v1
kind: Service
metadata:
  name: my-app-k8s-service
  namespace: flask-app
spec:
  type: NodePort
  ports:
  - port: 5000
    targetPort: 5000
    nodePort: 30007
  selector:
    app: my-app-k8s
```

Jenkins CI/CD

Create Push/Deploy Pipeline Script

This Jenkins pipeline is designed to manage Terraform infrastructure and deploy a Flask app on a Kubernetes cluster using Minikube, with an option to either apply or destroy the infrastructure. The pipeline also includes stages to interact with Docker, Ansible, and Kubernetes.



Key Features:

1. **Parameterized Action**:
    - The `parameters` block allows the user to select either "Apply" or "Destroy" as the Terraform action.
2. **Environment Variables**:
    - AWS credentials and Docker registry details are configured using Jenkins' credential management.
    - The Ansible private key and playbook are set as environment variables.

3. **Stages Breakdown**:
    - **Access Remote Repo**: Clones the repository from GitHub.
    - **Terraform Init and Action**:
        - Runs `terraform init` and `terraform plan` with variable files.
        - Executes either `terraform apply` or `terraform destroy` based on the chosen action.
        - Reads and validates the `inventory.txt` file after applying the infrastructure.
    - **Test Inventory**: Displays the contents of `inventory.txt` when applying.
    - **Wait for EC2**: Pauses the pipeline for 60 seconds to allow the EC2 instance to boot.
    - **Test SSH**: Attempts an SSH connection to the EC2 instance using the private key to validate the instance's availability.
    - **Run Ansible Playbook**: Runs the specified Ansible playbook if Terraform was applied successfully.
    - **Copy K8SManifests to EC2**: Copies Kubernetes manifests to the EC2 instance for deployment.
    - **Build and Tag Docker Image**: Builds a Docker image for the Flask app and tags it.
    - **Configure Docker & Login**: Logs into Docker Hub using stored credentials.
    - **Push Docker Image to Docker Hub**: Pushes the tagged image to Docker Hub.
    - **Deploy App on Minikube**: Deploys the Flask app on a Minikube Kubernetes cluster running on the EC2 instance.
    - **Destroy Confirmation**: Outputs a message confirming successful destruction of resources when "Destroy" is selected.

Notes on Specific Stages:

1. **Terraform Init and Action**:
    - This stage handles both the initialization and execution of Terraform. If "Apply" is chosen, it also ensures that `inventory.txt` exists after applying.
2. **Ansible and Kubernetes Deployment**:
    - After confirming the EC2 instance is running, the pipeline uses Ansible to configure the instance and deploy the Flask app via Kubernetes manifests.
3. **Docker Image Handling**:
    - The pipeline builds the Docker image for the Flask app and pushes it to Docker Hub, ensuring the image is available for deployment in Minikube.
4. **Minikube Deployment**:
    - The Minikube Kubernetes cluster is started on the EC2 instance, and the app is deployed using Kubernetes manifests.

Error Handling:

- If `inventory.txt` is not found after applying the infrastructure, an error is raised.
- The pipeline gracefully handles both "Apply" and "Destroy" actions, providing clear output messages on success or failure.

```
pipeline {
    agent any

    parameters {
        choice(name: 'action', choices: ["Apply", "Destroy"],
description: 'Select Terraform Action')
    }

    environment {
        AWS_ACCESS_KEY_ID = credentials('aws_access_key_id')
        AWS_SECRET_ACCESS_KEY = credentials('aws_secret_access_key')
        DOCKER_REGISTRY = 'hussamgamal/flask-app'
        ANSIBLE_PRIVATE_KEY = 'Terraform/tf-key-pairz.pem'
        PLAYBOOK_FILE = 'playbook/dockerk8s.yaml'
        DOCKER_HUB_CREDENTIALS = credentials('docker-hub-credentials')
    }
```

#ACCESS REMOTE REPO

```
    stages {
        stage('Access Remote Repo') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name:
'*/master']], userRemoteConfigs: [[url:
'https://github.com/Hossam404/DEPI-FP']]])
            }
        }
```

#TERRAFORM INIT & ACTION

```
        stage('Terraform Init and Action') {
            steps {
                script {
                    dir('Terraform') {
                        sh 'terraform init -upgrade'
                        sh 'terraform plan
-var-file="network/network.tfvars" -var-file="compute/compute.tfvars"'

                        def tfaction = params.action
                        if (tfaction == 'Apply') {
                            sh 'terraform apply
-var-file="network/network.tfvars" -var-file="compute/compute.tfvars"
-auto-approve'

                            def inventoryPath = 'inventory.txt'
                            if (fileExists(inventoryPath)) {
                                env.INVENTORY_CONTENT =
readFile(inventoryPath)
                            } else {
                                error("inventory.txt not found after
apply!")
                            }
                        } else if (tfaction == 'Destroy') {
                            sh 'terraform destroy
-var-file="network/network.tfvars" -var-file="compute/compute.tfvars"
-auto-approve'
                        } else {
                            error("Invalid choice for 'action'
parameter")
                        }
                    }
                }
            }
        }
```

#TEST INVENTORY

```
        stage('Test Inventory') {
            when {
                expression { params.action == 'Apply' &&
env.INVENTORY_CONTENT }
            }
            steps {
                script {
                    sh "cat Terraform/inventory.txt"
                }
            }
        }

        stage('Wait for EC2') {
            when {
                expression { params.action == 'Apply' }
            }
            steps {
                script {
                    echo 'Waiting for EC2 instance to boot up...'
                    sleep(time: 60, unit: 'SECONDS')
                }
            }
        }
```

#TEST SSH

```
stage('Test SSH') {
    when {
        expression { params.action == 'Apply' && env.INVENTORY_CONTENT
}
    }
    steps {
        script {
            def publicIp = sh(script: "grep ansible_user=ec2-user
Terraform/inventory.txt | awk '{print \$1}'", returnStdout:
true).trim()
            echo "Attempting SSH to EC2 instance at IP: ${publicIp}"
            sh "chmod 400 ${ANSIBLE_PRIVATE_KEY}"
            sh "ls -l"
            sh "ssh -i ${ANSIBLE_PRIVATE_KEY} -o
StrictHostKeyChecking=no ec2-user@${publicIp} exit"
        }
    }
}
```

#RUN ANSIBLE PLAYBOOK

```
        stage('Run Ansible Playbook') {
            when {
                expression { params.action == 'Apply' &&
env.INVENTORY_CONTENT }
            }
            steps {
                sh 'ansible-playbook -i Terraform/inventory.txt $
{PLAYBOOK_FILE}'
            }
        }
```

#COPY K8SMANIFEST to EC2

```
        stage('Copy K8SManifests to EC2') {
            when {
                expression { params.action == 'Apply' &&
env.INVENTORY_CONTENT }
            }
            steps {
                script {
                    def publicIp = sh(script: "grep ansible_user=ec2-
user Terraform/inventory.txt | awk '{print \$1}'", returnStdout:
true).trim()

                    sh """
                        scp -i ${ANSIBLE_PRIVATE_KEY} -r K8SManifests
ec2-user@${publicIp}:/home/ec2-user/K8SManifests
                    """
                }
            }
        }
```

#Build and Tag Docker Image

```
        stage('Build and Tag Docker Image') {
            when {
                expression { params.action == 'Apply' }
            }
            steps {
                sh 'docker build -t flask-app:v1 .'
                sh "docker tag flask-app:v1 ${DOCKER_REGISTRY}:v1"
            }
        }
```

#Configure Docker & Login

```
        stage('Configure Docker & Login') {
            when {
                expression { params.action == 'Apply' }
            }
            steps {
                script {
                    sh """
                        echo "${DOCKER_HUB_CREDENTIALS_PSW}" | docker
login -u "${DOCKER_HUB_CREDENTIALS_USR}" --password-stdin
                    """
                }
            }
        }
```

#Push Docker Image to Docker Hub

```
        stage('Push Docker Image to Docker Hub') {
            when {
                expression { params.action == 'Apply' }
            }
            steps {
                sh 'docker push ${DOCKER_REGISTRY}:v1'
            }
        }
```

Deploy App on Minikube

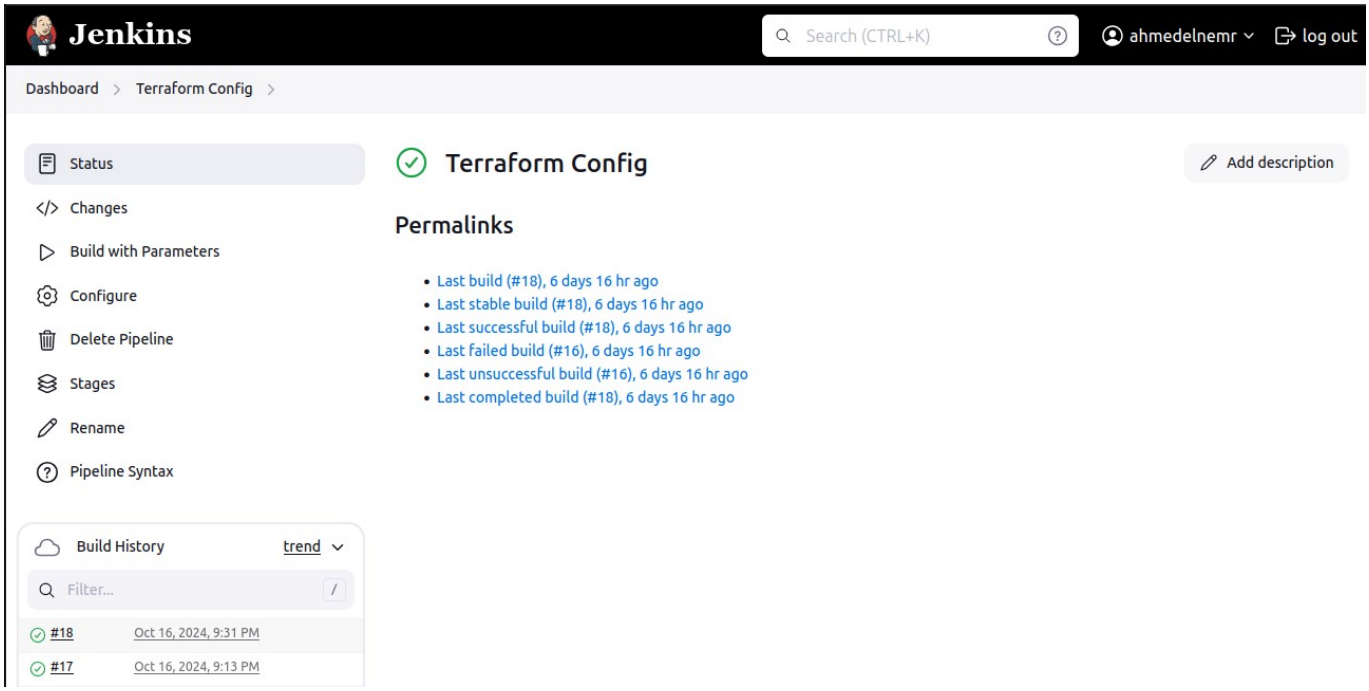
```
      stage('Deploy App on Minikube') {
    when {
        expression { params.action == 'Apply' }
    }
    steps {
        script {
            def publicIp = sh(script: "grep ansible_user=ec2-user
Terraform/inventory.txt | awk '{print \$1}'", returnStdout:
true).trim()
            sh """
            ssh -i ${ANSIBLE_PRIVATE_KEY} ec2-user@${publicIp} <<EOF
            sudo usermod -aG docker ec2-user
            newgrp docker

            minikube start --driver=docker

            sleep 90

            kubectl apply -f K8SManifests/namespace.yml
            kubectl apply -f K8SManifests/Deploymnet.Yaml
            kubectl apply -f K8SManifests/LoadBalancer.yaml

            sleep 30
            kubectl get svc -o wide -n flask-app
            EOF
            """
        }
    }
}
```

#DESTROY COMMAND

```
        stage('Destroy Confirmation') {
            when {
                expression { params.action == 'Destroy' }
            }
            steps {
                echo "SUCCESS: All Resources Destroyed"
            }
        }
    }
    post {
        success {
            echo 'Pipeline completed successfully!'
        }
        failure {
            echo 'Pipeline failed!'
        }
    }
}
```

Pipeline Table Design

Stage	Description	Dependencies
Access Remote Repo	Checks out the code from a remote Git repository.	None
Terraform Init and Action	Initializes Terraform and executes the specified Terraform command (apply or destroy).	Access Remote Repo
Test Inventory (Conditional)	Tests the generated inventory file.	Terraform Init and Action (if action is "Apply")
Wait for EC2 (Conditional)	Waits for the EC2 instance to boot up.	Terraform Init and Action (if action is "Apply")
Test SSH (Conditional)	Tests the SSH connection to the EC2 instance.	Terraform Init and Action (if action is "Apply")
Run Ansible Playbook (Conditional)	Executes the Ansible playbook.	Terraform Init and Action (if action is "Apply")
Copy K8SManifests to EC2 (Conditional)	Copies Kubernetes manifests to the EC2 instance.	Terraform Init and Action (if action is "Apply")
Build and Tag Docker Image	Builds and tags the Docker image.	Terraform Init and Action (if action is "Apply")
Configure Docker & Login (Conditional)	Configures Docker and logs in to Docker Hub.	Terraform Init and Action (if action is "Apply")
Push Docker Image to Docker Hub (Conditional)	Pushes the Docker image to Docker Hub.	Terraform Init and Action (if action is "Apply")
Deploy App on Minikube (Conditional)	Deploys the application on Minikube.	Terraform Init and Action (if action is "Apply")
Destroy Confirmation (Conditional)	Displays a confirmation message after destroying resources.	Terraform Init and Action (if action is "Destroy")



OUTPUT CONSOLE

Started by user ahmedelnemr
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/CICD Pipeline
[Pipeline] {
[Pipeline] withCredentials
Masking supported pattern matches of $AWS_ACCESS_KEY_ID or $DOCKER_HUB_CREDENTIALS or $DOCKER_HUB_CREDENTIALS_USR or $DOCKER_HUB_CREDENTIALS_PSW or $AWS_SECRET_ACCESS_KEY
[Pipeline] {
[Pipeline] withEnv
Warning: A secret was passed to "withEnv" using Groovy String interpolation, which is insecure.
Affected argument(s) used the following variable(s):
[DOCKER_HUB_CREDENTIALS_USR]
See https://jenkins.io/redirect/groovy-string-interpolation for details.
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Access Remote Repo)
[Pipeline] checkout
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/CICD Pipeline/.git # timeout=10

Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/Hossam404/DEPI-FP # timeout=10
Fetching upstream changes from https://github.com/Hossam404/DEPI-FP
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git fetch --tags --force --progress -- https://github.com/Hossam404/DEPI-FP
+refs/heads/:refs/remotes/origin/ # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 31d8cc722e9fc45d464035e4def0a2aac3d83bac (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 31d8cc722e9fc45d464035e4def0a2aac3d83bac # timeout=10
Commit message: "update"
 > git rev-list --no-walk 5e668a57d91f6f00970045d2b589cb49e7cebff7 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Terraform Init and Action)
[Pipeline] script
[Pipeline] {
[Pipeline] dir
Running in /var/lib/jenkins/workspace/CICD Pipeline/Terraform
[Pipeline] {
[Pipeline] sh
+ terraform init -upgrade
#[0m#[1mInitializing the backend...#[0m
#[0m#[1mUpgrading modules...#[0m
- compute in compute
- network in network
- security in security
#[0m#[1mInitializing provider plugins...#[0m
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/tls...
- Finding latest version of hashicorp/local...
- Using previously-installed hashicorp/aws v5.72.1
- Using previously-installed hashicorp/tls v4.0.6
- Using previously-installed hashicorp/local v2.5.2

#[0m#[1m#[32mTerraform has been successfully initialized!#[0m#[32m#[0m
#[0m#[32m
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other

commands will detect it and remind you to do so if necessary.#[0m
[Pipeline] sh
+ terraform plan -var-file=network/network.tfvars -var-file=compute/compute.tfvars
#[0m#[1mmodule.compute.tls_private_key.rsa-key: Refreshing state...
[id=43748a6d5a51affe120bfac2a070033b6de9c7ba]#[0m
#[0m#[1mmodule.compute.local_file.tf-key: Refreshing state...
[id=4e1f4385878683bca552c33d5893762180cd6478]#[0m
#[0m#[1mmodule.compute.aws_key_pair.tf-key-pairz: Refreshing state... [id=tf-key-pairz]#[0m
#[0m#[1mmodule.network.aws_vpc.my-vpc: Refreshing state... [id=vpc-0d0c6b935708ff444]#[0m
#[0m#[1mmodule.network.aws_internet_gateway.my_igw: Refreshing state... [id=igw-0aad3d484d3bc294e]#[0m
#[0m#[1mmodule.network.aws_subnet.public-subnet: Refreshing state... [id=subnet-07d33e390dea8cfba]#[0m
#[0m#[1mmodule.network.aws_route_table.my_route_public: Refreshing state... [id=rtb-0cbbe43b8ab0824a7]#[0m
#[0m#[1mmodule.security.aws_security_group.public: Refreshing state... [id=sg-09abc05576844d751]#[0m
#[0m#[1mmodule.network.aws_route.public_route: Refreshing state... [id=r-rtb-0cbbe43b8ab0824a71080289494]#[0m
#[0m#[1mmodule.network.aws_route_table_association.public_subnet_assoc: Refreshing state... [id=rtbassoc-0da42a76547335eca]#[0m

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  #[32m+#[0m create#[0m

Terraform will perform the following actions:

#[1m # module.compute.aws_instance.appdep#[0m will be created
#[0m #[32m+#[0m#[0m resource "aws_instance" "appdep" {
      #[32m+#[0m#[0m ami   = "ami-0fff1b9a61dec8a5f"
      #[32m+#[0m#[0m arn   = (known after apply)
      #[32m+#[0m#[0m associate_public_ip_address   = true
      #[32m+#[0m#[0m availability_zone   = "us-east-1f"
      #[32m+#[0m#[0m cpu_core_count   = (known after apply)
      #[32m+#[0m#[0m cpu_threads_per_core   = (known after apply)
      #[32m+#[0m#[0m disable_api_stop   = (known after apply)
      #[32m+#[0m#[0m disable_api_termination   = (known after apply)
      #[32m+#[0m#[0m ebs_optimized   = (known after apply)
      #[32m+#[0m#[0m get_password_data   = false
      #[32m+#[0m#[0m host_id   = (known after apply)
      #[32m+#[0m#[0m host_resource_group_arn   = (known after apply)
      #[32m+#[0m#[0m iam_instance_profile   = (known after apply)
      #[32m+#[0m#[0m id   = (known after apply)
      #[32m+#[0m#[0m instance_initiated_shutdown_behavior = (known after apply)

```
    #[32m+#[0m#[0m instance_lifecycle               = (known after apply)
#[32m+#[0m#[0m instance_state                  = (known after apply)
#[32m+#[0m#[0m instance_type                   = "t2.large"
#[32m+#[0m#[0m ipv6_address_count              = (known after apply)
#[32m+#[0m#[0m ipv6_addresses                  = (known after apply)
#[32m+#[0m#[0m key_name                        = "tf-key-pairz"
#[32m+#[0m#[0m monitoring                      = (known after apply)
#[32m+#[0m#[0m outpost_arn                     = (known after apply)
#[32m+#[0m#[0m password_data                   = (known after apply)
#[32m+#[0m#[0m placement_group                 = (known after apply)
#[32m+#[0m#[0m placement_partition_number      = (known after apply)
#[32m+#[0m#[0m primary_network_interface_id    = (known after apply)
#[32m+#[0m#[0m private_dns                     = (known after apply)
#[32m+#[0m#[0m private_ip                      = (known after apply)
#[32m+#[0m#[0m public_dns                      = (known after apply)
#[32m+#[0m#[0m public_ip                       = (known after apply)
#[32m+#[0m#[0m secondary_private_ips           = (known after apply)
#[32m+#[0m#[0m security_groups                 = [
    #[32m+#[0m#[0m "sg-09abc05576844d751",
  ]
#[32m+#[0m#[0m source_dest_check               = true
#[32m+#[0m#[0m spot_instance_request_id        = (known after apply)
#[32m+#[0m#[0m subnet_id                       = "subnet-07d33e390dea8cfba"
#[32m+#[0m#[0m tags                            = {
    #[32m+#[0m#[0m "Name" = "App EC2"
  }
#[32m+#[0m#[0m tags_all                        = {
    #[32m+#[0m#[0m "Name" = "App EC2"
  }
#[32m+#[0m#[0m tenancy                         = (known after apply)
#[32m+#[0m#[0m user_data                       = (sensitive value)
#[32m+#[0m#[0m user_data_base64                = (known after apply)
#[32m+#[0m#[0m user_data_replace_on_change     = false
#[32m+#[0m#[0m vpc_security_group_ids          = (known after apply)

#[32m+#[0m#[0m capacity_reservation_specification (known after apply)

#[32m+#[0m#[0m cpu_options (known after apply)

#[32m+#[0m#[0m ebs_block_device (known after apply)

#[32m+#[0m#[0m enclave_options (known after apply)

#[32m+#[0m#[0m ephemeral_block_device (known after apply)
```

#[32m+#[0m#[0m instance_market_options (known after apply)

#[32m+#[0m#[0m maintenance_options (known after apply)

#[32m+#[0m#[0m metadata_options (known after apply)

#[32m+#[0m#[0m network_interface (known after apply)

#[32m+#[0m#[0m private_dns_name_options (known after apply)

#[32m+#[0m#[0m root_block_device (known after apply)
}

#[1mPlan:#[0m 1 to add, 0 to change, 0 to destroy.
#[0m
Changes to Outputs:
#[32m+#[0m#[0m AppEc2 = (known after apply)
#[33m╷#[0m#[0m
#[33m│#[0m #[0m#[1m#[33mWarning: #[0m#[0m#[1mValue for undeclared variable#[0m
#[33m│#[0m #[0m
#[33m│#[0m #[0m#[0mThe root module does not declare a variable named "private_subnet_cidr" but
#[33m│#[0m #[0ma value was found in file "network/network.tfvars". If you meant to use
#[33m│#[0m #[0mthis value, add a "variable" block to the configuration.
#[33m│#[0m #[0m
#[33m│#[0m #[0mTo silence these warnings, use TF_VAR_... environment variables to provide
#[33m│#[0m #[0mcertain "global" settings to all configurations in your organization. To
#[33m│#[0m #[0mreduce the verbosity of these warnings, use the -compact-warnings option.
#[33m╵#[0m#[0m
#[90m
——
————————————————————————#[0m

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if you run "terraform apply" now.
[Pipeline] sh
+ terraform apply -var-file=network/network.tfvars -var-file=compute/compute.tfvars -auto-approve
#[0m#[1mmodule.compute.tls_private_key.rsa-key: Refreshing state...
[id=43748a6d5a51affe120bfac2a070033b6de9c7ba]#[0m
#[0m#[1mmodule.compute.local_file.tf-key: Refreshing state...
[id=4e1f4385878683bca552c33d5893762180cd6478]#[0m
#[0m#[1mmodule.compute.aws_key_pair.tf-key-pairz: Refreshing state... [id=tf-key-pairz]#[0m
#[0m#[1mmodule.network.aws_vpc.my-vpc: Refreshing state... [id=vpc-0d0c6b935708ff444]#[0m
#[0m#[1mmodule.network.aws_internet_gateway.my_igw: Refreshing state... [id=igw-
0aad3d484d3bc294e]#[0m

#[0m#[1mmodule.network.aws_subnet.public-subnet: Refreshing state... [id=subnet-07d33e390dea8cfba]#[0m
#[0m#[1mmodule.network.aws_route_table.my_route_public: Refreshing state... [id=rtb-0cbbe43b8ab0824a7]#[0m
#[0m#[1mmodule.security.aws_security_group.public: Refreshing state... [id=sg-09abc05576844d751]#[0m
#[0m#[1mmodule.network.aws_route.public_route: Refreshing state... [id=r-rtb-0cbbe43b8ab0824a71080289494]#[0m
#[0m#[1mmodule.network.aws_route_table_association.public_subnet_assoc: Refreshing state... [id=rtbassoc-0da42a76547335eca]#[0m

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  #[32m+#[0m create#[0m

Terraform will perform the following actions:

#[1m # module.compute.aws_instance.appdep#[0m will be created
#[0m #[32m+#[0m#[0m resource "aws_instance" "appdep" {
    #[32m+#[0m#[0m ami   = "ami-0fff1b9a61dec8a5f"
    #[32m+#[0m#[0m arn   = (known after apply)
    #[32m+#[0m#[0m associate_public_ip_address   = true
    #[32m+#[0m#[0m availability_zone   = "us-east-1f"
    #[32m+#[0m#[0m cpu_core_count   = (known after apply)
    #[32m+#[0m#[0m cpu_threads_per_core   = (known after apply)
    #[32m+#[0m#[0m disable_api_stop   = (known after apply)
    #[32m+#[0m#[0m disable_api_termination   = (known after apply)
    #[32m+#[0m#[0m ebs_optimized   = (known after apply)
    #[32m+#[0m#[0m get_password_data   = false
    #[32m+#[0m#[0m host_id   = (known after apply)
    #[32m+#[0m#[0m host_resource_group_arn   = (known after apply)
    #[32m+#[0m#[0m iam_instance_profile   = (known after apply)
    #[32m+#[0m#[0m id   = (known after apply)
    #[32m+#[0m#[0m instance_initiated_shutdown_behavior = (known after apply)
    #[32m+#[0m#[0m instance_lifecycle   = (known after apply)
    #[32m+#[0m#[0m instance_state   = (known after apply)
    #[32m+#[0m#[0m instance_type   = "t2.large"
    #[32m+#[0m#[0m ipv6_address_count   = (known after apply)
    #[32m+#[0m#[0m ipv6_addresses   = (known after apply)
    #[32m+#[0m#[0m key_name   = "tf-key-pairz"
    #[32m+#[0m#[0m monitoring   = (known after apply)
    #[32m+#[0m#[0m outpost_arn   = (known after apply)
    #[32m+#[0m#[0m password_data   = (known after apply)
    #[32m+#[0m#[0m placement_group   = (known after apply)
    #[32m+#[0m#[0m placement_partition_number   = (known after apply)

```
    #[32m+#[0m#[0m primary_network_interface_id        = (known after apply)
#[32m+#[0m#[0m private_dns                          = (known after apply)
#[32m+#[0m#[0m private_ip                           = (known after apply)
#[32m+#[0m#[0m public_dns                           = (known after apply)
#[32m+#[0m#[0m public_ip                            = (known after apply)
#[32m+#[0m#[0m secondary_private_ips                = (known after apply)
#[32m+#[0m#[0m security_groups                      = [
   #[32m+#[0m#[0m "sg-09abc05576844d751",
  ]
#[32m+#[0m#[0m source_dest_check                    = true
#[32m+#[0m#[0m spot_instance_request_id             = (known after apply)
#[32m+#[0m#[0m subnet_id                            = "subnet-07d33e390dea8cfba"
#[32m+#[0m#[0m tags                                 = {
   #[32m+#[0m#[0m "Name" = "App EC2"
  }
#[32m+#[0m#[0m tags_all                             = {
   #[32m+#[0m#[0m "Name" = "App EC2"
  }
#[32m+#[0m#[0m tenancy                              = (known after apply)
#[32m+#[0m#[0m user_data                            = (sensitive value)
#[32m+#[0m#[0m user_data_base64                     = (known after apply)
#[32m+#[0m#[0m user_data_replace_on_change          = false
#[32m+#[0m#[0m vpc_security_group_ids               = (known after apply)

#[32m+#[0m#[0m capacity_reservation_specification (known after apply)

#[32m+#[0m#[0m cpu_options (known after apply)

#[32m+#[0m#[0m ebs_block_device (known after apply)

#[32m+#[0m#[0m enclave_options (known after apply)

#[32m+#[0m#[0m ephemeral_block_device (known after apply)

#[32m+#[0m#[0m instance_market_options (known after apply)

#[32m+#[0m#[0m maintenance_options (known after apply)

#[32m+#[0m#[0m metadata_options (known after apply)

#[32m+#[0m#[0m network_interface (known after apply)

#[32m+#[0m#[0m private_dns_name_options (known after apply)

#[32m+#[0m#[0m root_block_device (known after apply)
```

}

#[1mPlan:#[0m 1 to add, 0 to change, 0 to destroy.
#[0m
Changes to Outputs:
  #[32m+#[0m#[0m AppEc2 = (known after apply)
#[0m#[1mmodule.compute.aws_instance.appdep: Creating...#[0m#[0m
#[0m#[1mmodule.compute.aws_instance.appdep: Still creating... [10s elapsed]#[0m#[0m
#[0m#[1mmodule.compute.aws_instance.appdep: Provisioning with 'local-exec'...#[0m#[0m
#[0m#[1mmodule.compute.aws_instance.appdep (local-exec):#[0m #[0mExecuting: ["/bin/sh" "-c"
"echo \"[all]\" > inventory.txt\necho \"44.220.57.45 ansible_user=ec2-user
ansible_ssh_private_key_file=Terraform/tf-key-pairz.pem\" >> inventory.txt\n"]
#[0m#[1mmodule.compute.aws_instance.appdep: Creation complete after 12s [id=i-
05e8487f141ab4e04]#[0m
#[33m╷#[0m#[0m
#[33m│ #[0m #[0m#[1m#[33mWarning: #[0m#[0m#[1mValue for undeclared variable#[0m
#[33m│ #[0m #[0m
#[33m│ #[0m #[0m#[0mThe root module does not declare a variable named "private_subnet_cidr" but
#[33m│ #[0m #[0ma value was found in file "network/network.tfvars". If you meant to use
#[33m│ #[0m #[0mthis value, add a "variable" block to the configuration.
#[33m│ #[0m #[0m
#[33m│ #[0m #[0mTo silence these warnings, use TF_VAR_... environment variables to provide
#[33m│ #[0m #[0mcertain "global" settings to all configurations in your organization. To
#[33m│ #[0m #[0mreduce the verbosity of these warnings, use the -compact-warnings option.
#[33m╵#[0m#[0m
#[0m#[1m#[32m
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
#[0m#[0m#[1m#[32m
Outputs:

#[0mAppEc2 = "i-05e8487f141ab4e04"
[Pipeline] fileExists
[Pipeline] readFile
[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test Inventory)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ cat Terraform/inventory.txt

[all]
44.220.57.45 ansible_user=ec2-user ansible_ssh_private_key_file=Terraform/tf-key-pairz.pem
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Wait for EC2)
[Pipeline] script
[Pipeline] {
[Pipeline] echo
Waiting for EC2 instance to boot up...
[Pipeline] sleep
Sleeping for 1 min 0 sec
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test SSH)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ grep ansible_user=ec2-user Terraform/inventory.txt
+ awk {print $1}
[Pipeline] echo
Attempting SSH to EC2 instance at IP: 44.220.57.45
[Pipeline] sh
+ chmod 400 Terraform/tf-key-pairz.pem
[Pipeline] sh
+ ls -l
total 32
-rw-r--r-- 1 jenkins jenkins 438 Oct 16 21:13 Dockerfile
drwxr-xr-x 2 jenkins jenkins 4096 Oct 16 21:13 K8SManifests
-rw-r--r-- 1 jenkins jenkins 3037 Oct 23 13:22 README.md
drwxr-xr-x 6 jenkins jenkins 4096 Oct 23 14:55 Terraform
drwxr-xr-x 2 jenkins jenkins 4096 Oct 23 14:55 Terraform@tmp
-rw-r--r-- 1 jenkins jenkins 253 Oct 16 21:13 app.py
drwxr-xr-x 2 jenkins jenkins 4096 Oct 22 20:32 playbook
-rw-r--r-- 1 jenkins jenkins   29 Oct 16 21:13 requirements.txt
[Pipeline] sh
+ ssh -i Terraform/tf-key-pairz.pem -o StrictHostKeyChecking=no ec2-user@44.220.57.45 exit
Warning: Permanently added '44.220.57.45' (ED25519) to the list of known hosts.
[Pipeline] }
[Pipeline] // script

[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run Ansible Playbook)
[Pipeline] sh
+ ansible-playbook -i Terraform/inventory.txt playbook/dockerk8s.yaml

PLAY [Setup Docker, Kubernetes, and Minikube on EC2 instance] ******

TASK [Gathering Facts] ********************
[WARNING]: Platform linux on host 44.220.57.45 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/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [44.220.57.45]

TASK [Update package index] ********************
changed: [44.220.57.45]

TASK [Install necessary dependencies] ***************
changed: [44.220.57.45]

TASK [Install Docker] *********************
changed: [44.220.57.45]

TASK [Start Docker service] ********************
changed: [44.220.57.45]

TASK [Download kubectl binary] ******************
changed: [44.220.57.45]

TASK [Make kubectl binary executable] ***************
changed: [44.220.57.45]

TASK [Move kubectl to /usr/local/bin] ***************
changed: [44.220.57.45]

TASK [Add Kubernetes repository] ******************
changed: [44.220.57.45]

TASK [Install Minikube] *********************
changed: [44.220.57.45]

PLAY RECAP ************************

44.220.57.45 : ok=10 changed=9 unreachable=0 failed=0 skipped=0 rescuec
ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Copy K8SManifests to EC2)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ grep ansible_user=ec2-user Terraform/inventory.txt
+ awk {print $1}
[Pipeline] sh
+ scp -i Terraform/tf-key-pairz.pem -r K8SManifests
ec2-user@44.220.57.45:/home/ec2-user/K8SManifests
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build and Tag Docker Image)
[Pipeline] sh
+ docker build -t flask-app:v1 .
#0 building with "default" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 477B done
#1 DONE 0.0s

#2 [auth] library/python:pull token for registry-1.docker.io
#2 DONE 0.0s

#3 [internal] load metadata for docker.io/library/python:3.10-slim
#3 DONE 0.1s

#4 [internal] load .dockerignore
#4 transferring context: 2B done
#4 DONE 0.0s

#5 [1/5] FROM docker.io/library/python:3.10-
slim@sha256:eb9ca77b1a0ffbde84c1dc333beb3490a2638813cc25a339f8575668855b9ff1
#5 DONE 0.0s

#6 [internal] load build context
#6 transferring context: 96.63kB 0.0s done

#6 DONE 0.0s

#7 [2/5] WORKDIR /app
#7 CACHED

#8 [3/5] COPY requirements.txt .
#8 CACHED

#9 [4/5] RUN pip install --no-cache-dir -r requirements.txt
#9 CACHED

#10 [5/5] COPY . .
#10 DONE 4.7s

#11 exporting to image
#11 exporting layers
#11 exporting layers 3.1s done
#11 writing image sha256:6f360e406dfa6475d2606f144948a71026d50d247bb3b787ac60ee8b217204f3 done
#11 naming to docker.io/library/flask-app:v1 done
#11 DONE 3.2s
[Pipeline] sh
Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.
Affected argument(s) used the following variable(s):
[DOCKER_HUB_CREDENTIALS_USR]
See https://jenkins.io/redirect/groovy-string-interpolation for details.
+ docker tag flask-app:v1 **/flask-app:v1
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Configure Docker & Login)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.
Affected argument(s) used the following variable(s):
[DOCKER_HUB_CREDENTIALS_USR, DOCKER_HUB_CREDENTIALS_PSW]
See https://jenkins.io/redirect/groovy-string-interpolation for details.
+ docker login -u ** --password-stdin
+ echo **
WARNING! Your password will be stored unencrypted in /var/lib/jenkins/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded

[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Push Docker Image to Docker Hub)
[Pipeline] sh
+ docker push **/flask-app:v1
The push refers to repository [docker.io/****/flask-app]
34879d47671e: Preparing
a102865163b2: Preparing
1f9942caaad8: Preparing
d6c1d4a26596: Preparing
aa223cd851d4: Preparing
796a04bf70eb: Preparing
eb70195b3e7f: Preparing
98b5f35ea9d3: Preparing
796a04bf70eb: Waiting
eb70195b3e7f: Waiting
98b5f35ea9d3: Waiting
1f9942caaad8: Layer already exists
aa223cd851d4: Layer already exists
a102865163b2: Layer already exists
d6c1d4a26596: Layer already exists
98b5f35ea9d3: Layer already exists
eb70195b3e7f: Layer already exists
796a04bf70eb: Layer already exists
34879d47671e: Pushed
v1: digest: sha256:e61e8dfb5e4a6776598ff77f525ba152ce1100c00ebcb97e44650a1b4a732e7c size: 1996
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy App on Minikube)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ grep ansible_user=ec2-user Terraform/inventory.txt
+ awk {print $1}
[Pipeline] sh
+ ssh -i Terraform/tf-key-pairz.pem ec2-user@44.220.57.45
Pseudo-terminal will not be allocated because stdin is not a terminal.

A newer release of "Amazon Linux" is available.
  Version 2023.6.20241010:
Run "/usr/bin/dnf check-release-update" for full release and version update info

```
       ,     #_
    ~\_  ####_        Amazon Linux 2023
   ~~  \_#####\
   ~~     \###|
   ~~       \#/ _   https://aws.amazon.com/linux/amazon-linux-2023
    ~       V' '->
     ~         /
       ~~._.   _/
          _/ _/
        _/m/'
* minikube v1.34.0 on Amazon 2023.5.20241001 (xen/amd64)
* Using the docker driver based on user configuration
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Downloading Kubernetes v1.31.0 preload ...
    > gcr.io/k8s-minikube/kicbase...:  1.61 KiB / 487.90 MiB [>] 0.00% ? p/s ?    >
gcr.io/k8s-minikube/kicbase...:  177.75 KiB / 487.90 MiB [] 0.04% ? p/s ?    >
gcr.io/k8s-minikube/kicbase...:  3.91 MiB / 487.90 MiB [>] 0.80% ? p/s ?    >
gcr.io/k8s-minikube/kicbase...:  6.71 MiB / 487.90 MiB  1.37% 11.17 MiB p    >
gcr.io/k8s-minikube/kicbase...:  9.69 MiB / 487.90 MiB  1.99% 11.17 MiB p    >
gcr.io/k8s-minikube/kicbase...:  12.81 MiB / 487.90 MiB  2.63% 11.17 MiB    >
gcr.io/k8s-minikube/kicbase...:  15.17 MiB / 487.90 MiB  3.11% 11.36 MiB    >
gcr.io/k8s-minikube/kicbase...:  20.14 MiB / 487.90 MiB  4.13% 11.36 MiB    >
gcr.io/k8s-minikube/kicbase...:  23.63 MiB / 487.90 MiB  4.84% 11.36 MiB    >
gcr.io/k8s-minikube/kicbase...:  27.44 MiB / 487.90 MiB  5.62% 11.95 MiB    >
gcr.io/k8s-minikube/kicbase...:  32.38 MiB / 487.90 MiB  6.64% 11.95 MiB    >
gcr.io/k8s-minikube/kicbase...:  35.69 MiB / 487.90 MiB  7.32% 11.95 MiB    >
gcr.io/k8s-minikube/kicbase...:  41.75 MiB / 487.90 MiB  8.56% 12.71 MiB    >
gcr.io/k8s-minikube/kicbase...:  46.80 MiB / 487.90 MiB  9.59% 12.71 MiB    >
gcr.io/k8s-minikube/kicbase...:  53.75 MiB / 487.90 MiB  11.02% 12.71 MiB    >
gcr.io/k8s-minikube/kicbase...:  62.55 MiB / 487.90 MiB  12.82% 14.13 MiB    >
gcr.io/k8s-minikube/kicbase...:  69.00 MiB / 487.90 MiB  14.14% 14.13 MiB    >
gcr.io/k8s-minikube/kicbase...:  77.16 MiB / 487.90 MiB  15.81% 14.13 MiB    >
gcr.io/k8s-minikube/kicbase...:  87.00 MiB / 487.90 MiB  17.83% 15.85 MiB    >
gcr.io/k8s-minikube/kicbase...:  90.95 MiB / 487.90 MiB  18.64% 15.85 MiB    >
gcr.io/k8s-minikube/kicbase...:  95.47 MiB / 487.90 MiB  19.57% 15.85 MiB    >
gcr.io/k8s-minikube/kicbase...:  100.42 MiB / 487.90 MiB  20.58% 16.27 Mi    >
gcr.io/k8s-minikube/kicbase...:  104.66 MiB / 487.90 MiB  21.45% 16.27 Mi    >
gcr.io/k8s-minikube/kicbase...:  114.21 MiB / 487.90 MiB  23.41% 16.27 Mi    >
gcr.io/k8s-minikube/kicbase...:  120.04 MiB / 487.90 MiB  24.60% 17.33 Mi    >
gcr.io/k8s-minikube/kicbase...:  125.66 MiB / 487.90 MiB  25.76% 17.33 Mi    >
gcr.io/k8s-minikube/kicbase...:  133.44 MiB / 487.90 MiB  27.35% 17.33 Mi    >
gcr.io/k8s-minikube/kicbase...:  141.71 MiB / 487.90 MiB  29.04% 18.54 Mi    >
gcr.io/k8s-minikube/kicbase...:  150.75 MiB / 487.90 MiB  30.90% 18.54 Mi    >
```

gcr.io/k8s-minikube/kicbase...: 158.10 MiB / 487.90 MiB 32.40% 18.54 Mi > gcr.io/k8s-minikube/kicbase...: 163.08 MiB / 487.90 MiB 33.42% 19.65 Mi > gcr.io/k8s-minikube/kicbase...: 169.33 MiB / 487.90 MiB 34.71% 19.65 Mi > gcr.io/k8s-minikube/kicbase...: 174.16 MiB / 487.90 MiB 35.70% 19.65 Mi > gcr.io/k8s-minikube/kicbase...: 179.01 MiB / 487.90 MiB 36.69% 20.09 Mi > gcr.io/k8s-minikube/kicbase...: 184.63 MiB / 487.90 MiB 37.84% 20.09 Mi >
gcr.io/k8s-minikube/kicbase...: 193.24 MiB / 487.90 MiB 39.61% 20.09 Mi >
gcr.io/k8s-minikube/kicbase...: 199.52 MiB / 487.90 MiB 40.89% 21.00 Mi >
gcr.io/k8s-minikube/kicbase...: 204.71 MiB / 487.90 MiB 41.96% 21.00 Mi >
gcr.io/k8s-minikube/kicbase...: 213.43 MiB / 487.90 MiB 43.74% 21.00 Mi >
gcr.io/k8s-minikube/kicbase...: 220.15 MiB / 487.90 MiB 45.12% 21.86 Mi >
gcr.io/k8s-minikube/kicbase...: 233.28 MiB / 487.90 MiB 47.81% 21.86 Mi >
gcr.io/k8s-minikube/kicbase...: 253.23 MiB / 487.90 MiB 51.90% 21.86 Mi >
gcr.io/k8s-minikube/kicbase...: 272.00 MiB / 487.90 MiB 55.75% 26.03 Mi >
gcr.io/k8s-minikube/kicbase...: 291.38 MiB / 487.90 MiB 59.72% 26.03 Mi >
gcr.io/k8s-minikube/kicbase...: 311.71 MiB / 487.90 MiB 63.89% 26.03 Mi >
gcr.io/k8s-minikube/kicbase...: 324.46 MiB / 487.90 MiB 66.50% 29.98 Mi >
gcr.io/k8s-minikube/kicbase...: 339.63 MiB / 487.90 MiB 69.61% 29.98 Mi >
gcr.io/k8s-minikube/kicbase...: 357.77 MiB / 487.90 MiB 73.33% 29.98 Mi >
gcr.io/k8s-minikube/kicbase...: 374.97 MiB / 487.90 MiB 76.85% 33.49 Mi >
gcr.io/k8s-minikube/kicbase...: 392.46 MiB / 487.90 MiB 80.44% 33.49 Mi >
gcr.io/k8s-minikube/kicbase...: 408.44 MiB / 487.90 MiB 83.71% 33.49 Mi >
gcr.io/k8s-minikube/kicbase...: 421.88 MiB / 487.90 MiB 86.47% 36.37 Mi >
gcr.io/k8s-minikube/kicbase...: 437.58 MiB / 487.90 MiB 89.69% 36.37 Mi >
gcr.io/k8s-minikube/kicbase...: 457.53 MiB / 487.90 MiB 93.78% 36.37 Mi >
gcr.io/k8s-minikube/kicbase...: 477.25 MiB / 487.90 MiB 97.82% 39.97 Mi >
gcr.io/k8s-minikube/kicbase...: 487.90 MiB / 487.90 MiB 100.00% 44.56 M* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
namespace/flask-app created
deployment.apps/my-app-k8s created
service/my-app-k8s-service created
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR
my-app-k8s-service NodePort 10.103.217.61 <none> 5000:30007/TCP 30s app=my-app-k8s
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage

[Pipeline] stage
[Pipeline] { (Destroy Confirmation)
Stage "Destroy Confirmation" skipped due to when conditional
[Pipeline] getContext
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Pipeline completed successfully!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

References

1. Jenkins Documentation:
https://www.jenkins.io/doc/
2. Docker Documentation: https://docs.docker.com/
3. Kubernetes Documentation:
https://kubernetes.io/docs/

These resources provided valuable insights, guidance, and support throughout the project lifecycle, enabling us to achieve our goals effectively.