Trend Store Application Documentation

Step 1: Cloning the repository.

- Git clone https://github.com/Vennilavan12/Trend.git

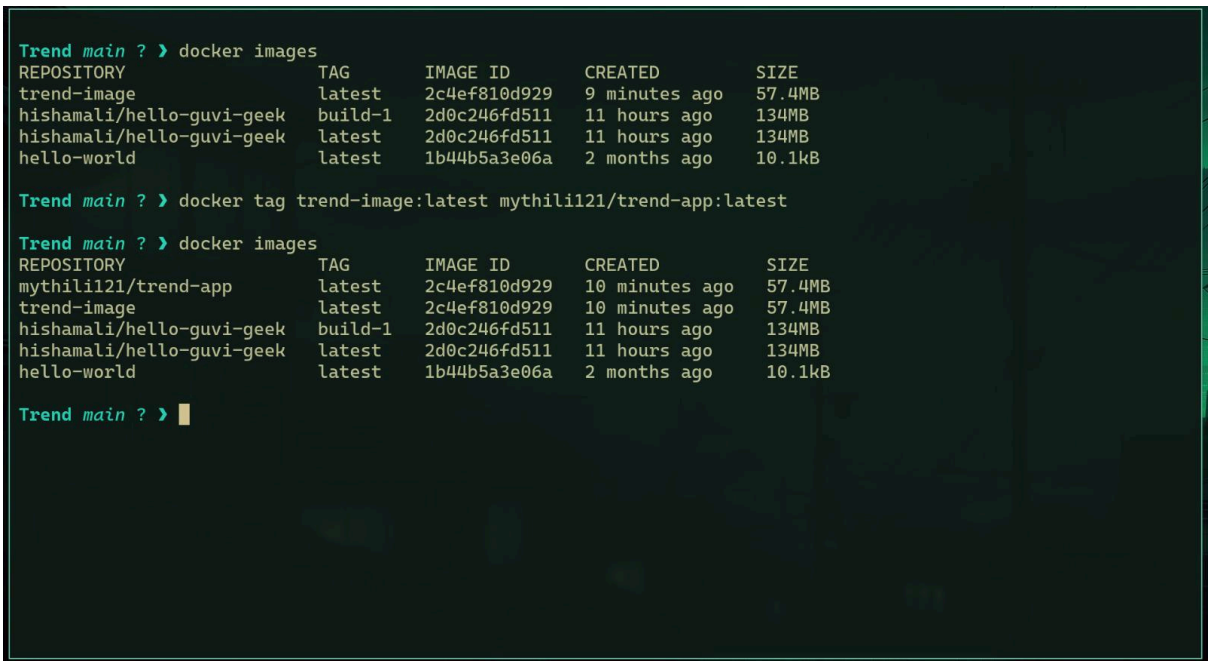
Step 2 : Dockerize the application using dockerfile.

Dockerfile

```
FROM nginx:alpine
COPY dist/ /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Step 3: Build and run locally

- docker build -t brain-tasks-app .
- docker run -p 80:3000 brain-tasks-app



Step 4 : Terraform -Infrastructure as a Code.

main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = "us-east-1"  # Or your preferred AWS region
}

resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "trend-vpc"
  }
}

resource "aws_subnet" "public" {
  count = 2
  vpc_id     = aws_vpc.main.id
  cidr_block = cidrsubnet(aws_vpc.main.cidr_block, 8, count.index)
  availability_zone = data.aws_availability_zones.available.names[count.index]
```

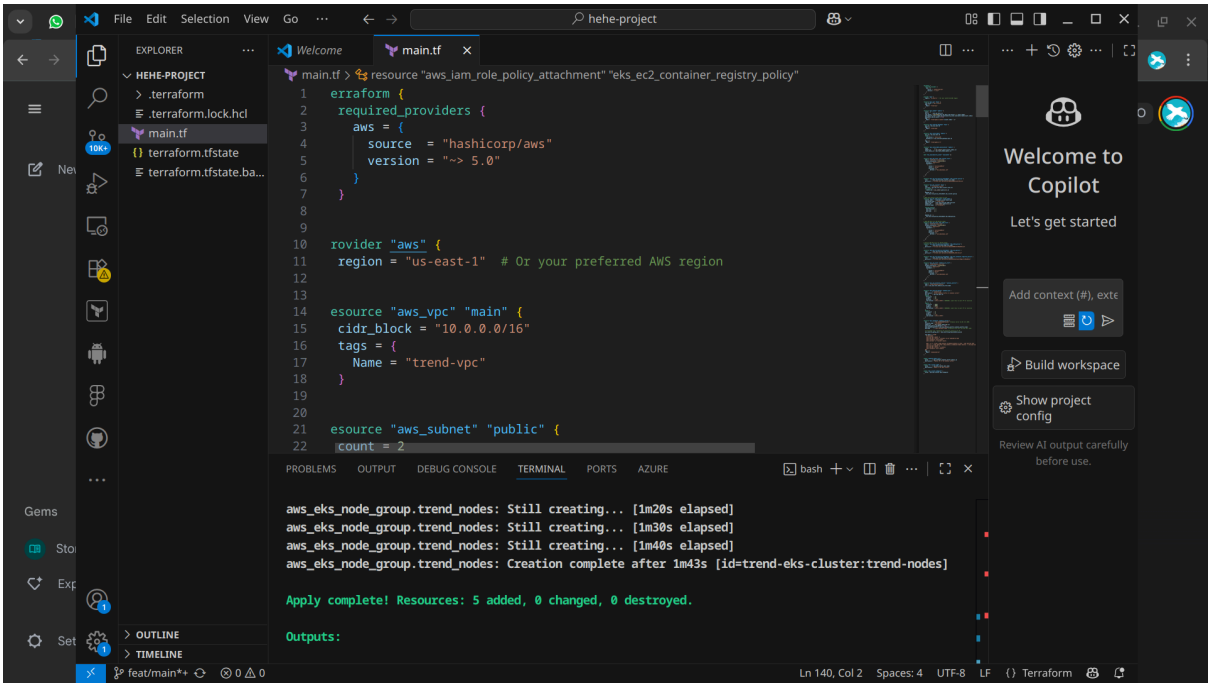
```
  map_public_ip_on_launch = true
  tags = {
    Name = "trend-public-subnet-${count.index + 1}"
  }
}

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id
  tags = {
    Name = "trend-igw"
  }
}

resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.main.id
  }
  tags = {
    Name = "trend-public-rt"
  }
}

resource "aws_route_table_association" "public" {
  count          = 2
  subnet_id      = aws_subnet.public[count.index].id
  route_table_id = aws_route_table.public.id
}

data "aws_availability_zones" "available" {}


resource "aws_iam_role" "eks_cluster_role" {
  name = "trend-eks-cluster-role"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "eks.amazonaws.com"
        }
```

```
    },
   ]
  })
}

resource "aws_iam_role_policy_attachment" "eks_cluster_policy" {
  role       = aws_iam_role.eks_cluster_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
}

resource "aws_eks_cluster" "main" {
  name     = "trend-eks-cluster"
  role_arn = aws_iam_role.eks_cluster_role.arn
  vpc_config {
    subnet_ids = aws_subnet.public[*].id
  }
  depends_on = [
    aws_iam_role_policy_attachment.eks_cluster_policy
  ]
}
# Add this block to your main.tf file
resource "aws_eks_node_group" "trend_nodes" {
  cluster_name    = aws_eks_cluster.main.name
  node_group_name = "trend-nodes"
  node_role_arn   = aws_iam_role.eks_node_role.arn
  subnet_ids      = aws_subnet.public[*].id
  instance_types  = ["t2.medium"]

  scaling_config {
    desired_size = 2
    max_size     = 3
    min_size     = 1
  }

  depends_on = [
    aws_iam_role_policy_attachment.eks_node_policy,
  ]
}

# Add IAM Role for EKS Worker Nodes
resource "aws_iam_role" "eks_node_role" {
  name = "trend-eks-node-role"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
```

```
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "ec2.amazonaws.com"
        }
      },
    ]
  })
}

# Attach IAM Policies for Worker Nodes
resource "aws_iam_role_policy_attachment" "eks_node_policy" {
  role       = aws_iam_role.eks_node_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
}

resource "aws_iam_role_policy_attachment" "eks_cni_policy" {
  role       = aws_iam_role.eks_node_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
}

resource "aws_iam_role_policy_attachment" "eks_ec2_container_registry_policy" {
  role       = aws_iam_role.eks_node_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
}

resource "aws_iam_role" "jenkins_ec2_role" {
  name = "trend-jenkins-ec2-role"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Principal = {
          Service = "ec2.amazonaws.com"
        }
      },
    ]
  })
}
```

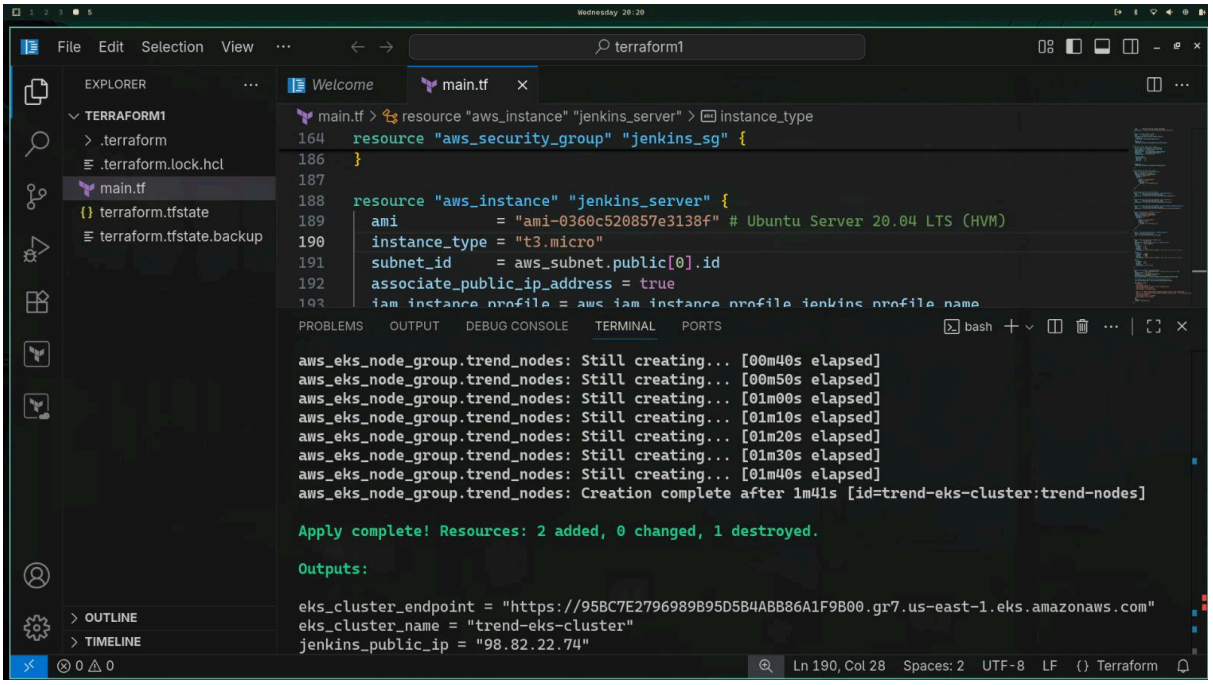
```
resource "aws_iam_instance_profile" "jenkins_profile" {
  name = "trend-jenkins-profile"
  role = aws_iam_role.jenkins_ec2_role.name
}


resource "aws_security_group" "jenkins_sg" {
  name        = "jenkins-sg"
  description = "Allow inbound traffic to Jenkins server"
  vpc_id      = aws_vpc.main.id
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"] # WARNING: Limit this to your IP for security
  }
  ingress {
    from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"] # WARNING: Limit this to your IP for security
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "jenkins_server" {
  ami           = "ami-0360c520857e3138f" # Ubuntu Server 20.04 LTS (HVM)
  instance_type = "t2.medium"
  subnet_id     = aws_subnet.public[0].id
  associate_public_ip_address = true
  iam_instance_profile = aws_iam_instance_profile.jenkins_profile.name
  key_name      = "your-key-pair-name" # Replace with your EC2 key pair name

  # Corrected line: reference the security group by its ID
  vpc_security_group_ids = [aws_security_group.jenkins_sg.id]

  user_data = <<-EOT
    #!/bin/bash
    sudo apt-get update -y
```

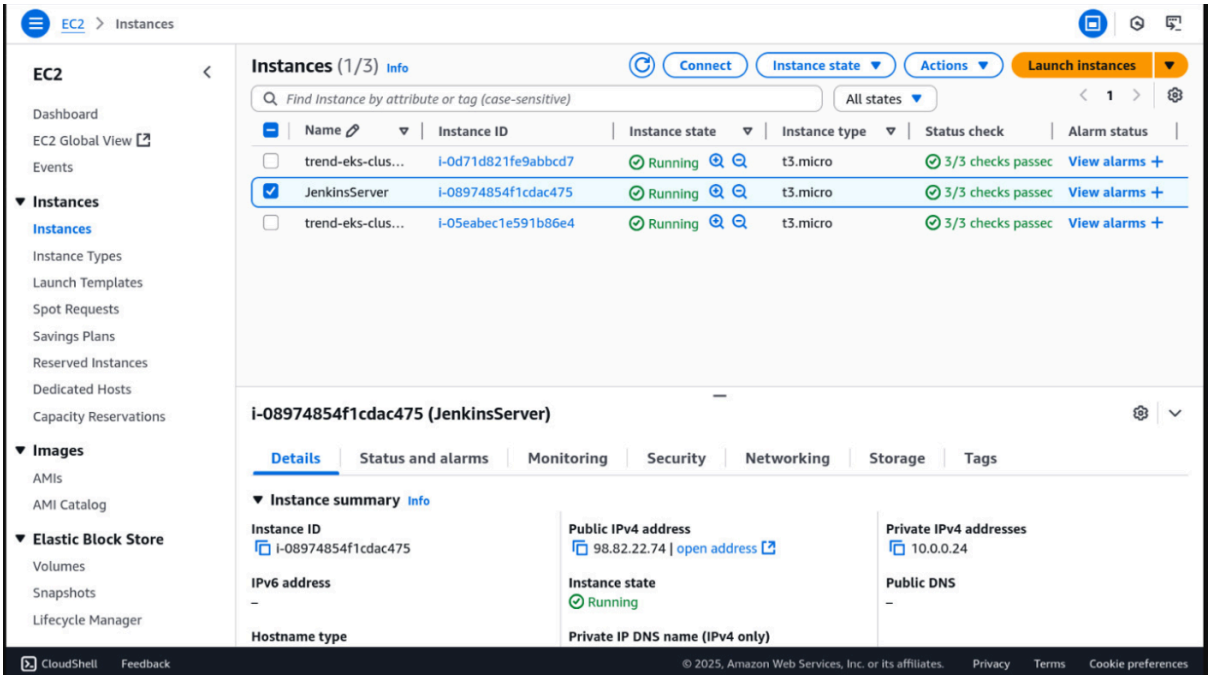
```
    sudo apt-get install -y docker.io git openjdk-11-jdk
    sudo systemctl start docker
    sudo usermod -a -G docker ubuntu

    wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
    sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
    sudo apt-get update -y
    sudo apt-get install -y jenkins
    sudo systemctl start jenkins
  EOT
  tags = {
    Name = "JenkinsServer"
  }
}


output "jenkins_public_ip" {
  value       = aws_instance.jenkins_server.public_ip
  description = "Public IP of the Jenkins server"
}

output "eks_cluster_name" {
  value       = aws_eks_cluster.main.name
  description = "Name of the EKS cluster"
}

output "eks_cluster_endpoint" {
  value = aws_eks_cluster.main.endpoint
}
```

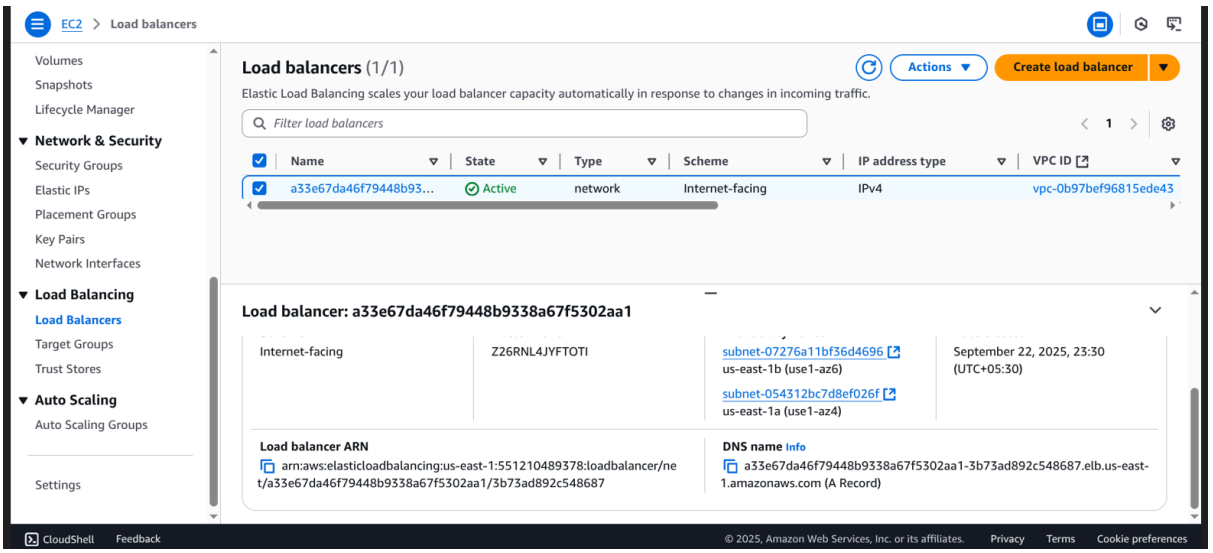


Step 5 : Terraform Initialize & Apply

- The terraform/main.tf file defines all the necessary AWS infrastructure.
- Run the following commands to create the VPC, EKS cluster, and Jenkins EC2 instance
    i. terraform init
    ii. terraform plan
    iii. terraform apply
- The terraform apply command will output the public IP of the Jenkins server.







Step 6 : Create a dockhub.

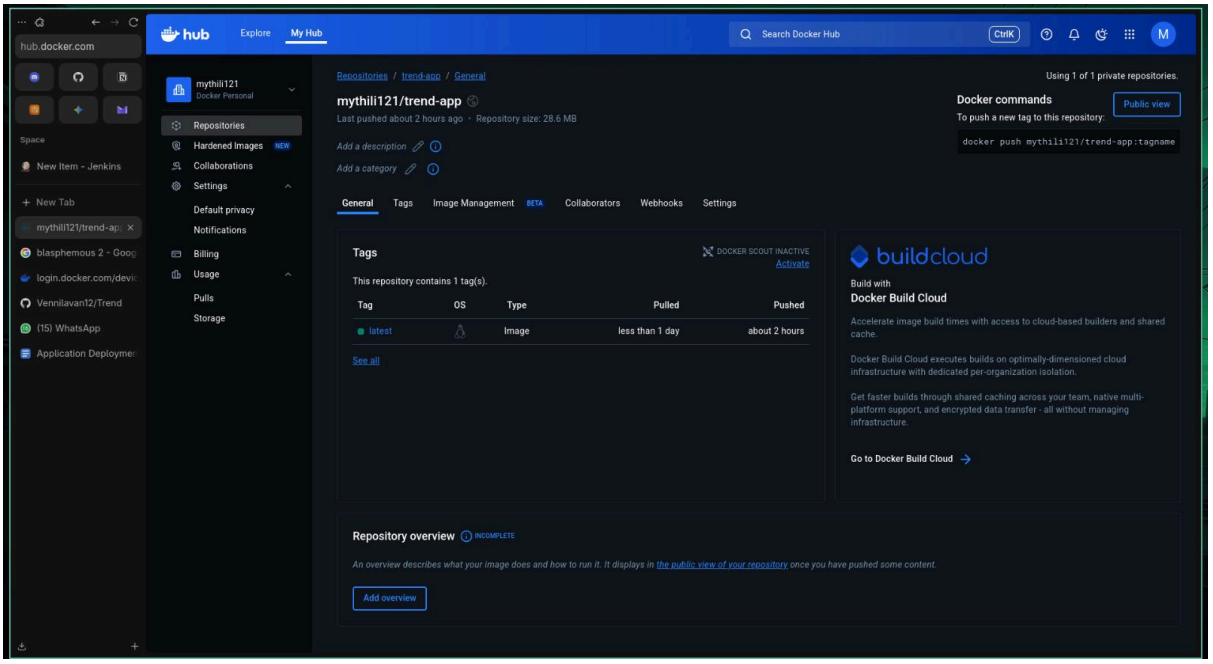
- Log in to https://hub.docker.com/
- Create repo: trend app

```
Trend main ? ❯ docker login

USING WEB-BASED LOGIN

i Info → To sign in with credentials on the command line, use 'docker login -u <username>'


Your one-time device confirmation code is: TBXN-RHJN
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser…


WARNING! Your credentials are stored unencrypted in '/home/bubu/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded

Trend main ? ❯
```

Step 7 : Tag and push the image.

- docker tag trend-app your-dockerhub-username/trend-app:latest
- docker login
- docker push your-dockerhub-username/trend-app:latest

```
Trend main ? ❯ docker push mythili121/trend-app:latest
The push refers to repository [docker.io/mythili121/trend-app]
bf482c750969: Pushed
3662d61b1197: Mounted from library/nginx
19c101c8a6e8: Mounted from library/nginx
d3c82e18bdb8: Mounted from library/nginx
72997926d63c: Mounted from library/nginx
c79f431f2893: Mounted from library/nginx
60e1771cb327: Mounted from library/nginx
f76dbf6a56fe: Mounted from library/nginx
7003d23cc217: Mounted from library/nginx
latest: digest: sha256:8c586eefb09c0393ce71c720be8429f6140b51993dd94310042904eb1cdc2330 size: 2200

Trend main ? ❯
```



Step 8 : Creation of AWS EKS cluster and verifying Cluster

- Set up an EKS cluster. Ensure the IAM role for your worker nodes has permissions to create AWS Load Balancers.
- eksctl create cluster --name brain-tasks-cluster --region <your-region> --nodegroup-name brain-tasks-nodes --node-type t3.medium --nodes 2

Step 9 : Kubernetes files.

Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: trend-app-deployment
  labels:
    app: trend-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: trend-app
  template:
    metadata:
      labels:
        app: trend-app
    spec:
      containers:
```

```
      - name: trend-app-container
        image: mythili121/trend-app:latest
        ports:
        - containerPort: 80
```

Service.yaml :

```
apiVersion: v1
kind: Service
metadata:
  name: trend-app-service
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "nlb"
spec:
  type: LoadBalancer
  selector:
    app: trend-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

Step 10 : Version Control - Git hub

Gitignore:

```
/node_modules
.env
.DS_Store
```

npm-debug.log*
yarn-debug.log*
yarn-error.log*
.terraform
.terraform.lock.hcl
*.tfstate
.tfstate.
.vscode/
.idea/

Dockerignore:
.git
.gitignore
.dockerignore
.vscode/
node_modules/

Step 11 : Push to github.

- git init
- git remote add origin https://github.com/yourusername/trend-devops.git
- git add .
- git commit -m "Initial commit"
- git push -u origin main
-

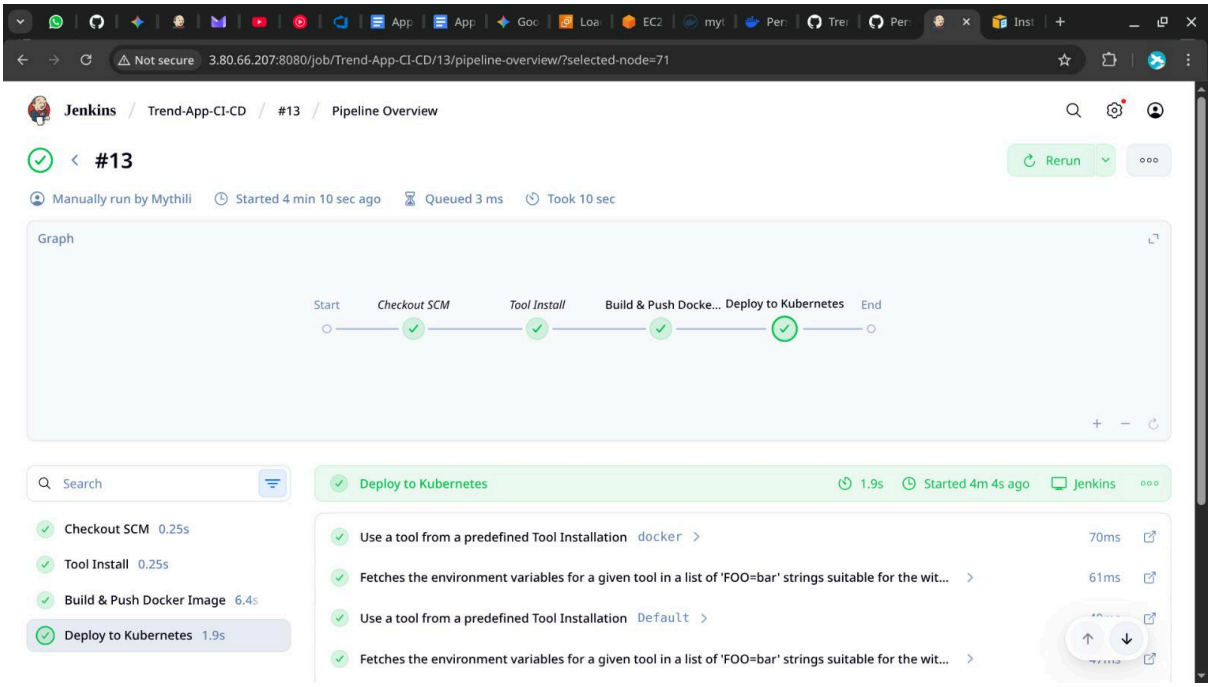
Step 12 : Jenkins CI/CD set up.

Jenkins Configuration

- Access Jenkins: Open a browser and navigate to http://<jenkins_public_ip>:8080.
- Retrieve Admin Password: SSH into the Jenkins EC2 instance to get the initial admin password.

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

- Install Plugins: Install the required plugins: Docker Pipeline, Git, Kubernetes Pipeline, and Kubernetes CLI.
- Add Credentials: Go to Manage Jenkins -> Credentials and add your DockerHub credentials (dockerhub-credentials) and your AWS kubeconfig as a secret file (eks-cluster-credentials).
- Create Pipeline Job: Create a new Pipeline job, configure it to use Git, and point the script path to Jenkinsfile.



Step 13 : Set up Webhook

- In your GitHub repository settings, create a webhook that points to your Jenkins server's GitHub webhook endpoint to trigger builds automatically on pushes.
- Go to GitHub repo → Settings → Webhooks:
- Payload URL: http://<jenkins-url>/github-webhook/
  Content type: application/json
- Trigger: Push event

Step 14 : Declarative pipeline script

```
pipeline {
    agent any

    tools {
        dockerTool 'docker'
        git 'Default'
    }

    environment {
        DOCKERHUB_CREDENTIALS_ID = 'dockerhub-credentials'
        KUBECONFIG_CREDENTIALS_ID = 'eks-cluster-credentials'
    }
```

```
    stages {
        stage('Build & Push Docker Image') {
            steps {
                script {
                    def dockerImage =
docker.build("mythili121/trend-app:${env.BUILD_NUMBER}", ".")

                    withCredentials([usernamePassword(credentialsId:
DOCKERHUB_CREDENTIALS_ID, passwordVariable: 'DOCKERHUB_TOKEN',
usernameVariable: 'DOCKERHUB_USER')]) {
                        docker.withRegistry('https://registry.hub.docker.com',
DOCKERHUB_CREDENTIALS_ID) {
                            dockerImage.push()
                        }
                    }
                }
            }
        }

        stage('Deploy to Kubernetes') {
            steps {
                script {
                    sh "sed -i 's|image: .*|image: mythili121/trend-app:${env.BUILD_NUMBER}|g'
kubernetes/deployment.yaml"

                    // Use withCredentials to expose the kubeconfig file as an environment variable
                    withCredentials([file(credentialsId: KUBECONFIG_CREDENTIALS_ID,
variable: 'KUBE_CONFIG_FILE')]) {
                        // Set the KUBECONFIG environment variable for the shell session
                        sh 'KUBECONFIG=$KUBE_CONFIG_FILE kubectl apply -f kubernetes/'
                    }
                }
            }
        }
    }
}
```

Step 15 : Monitoring

Monitoring (Grafana)

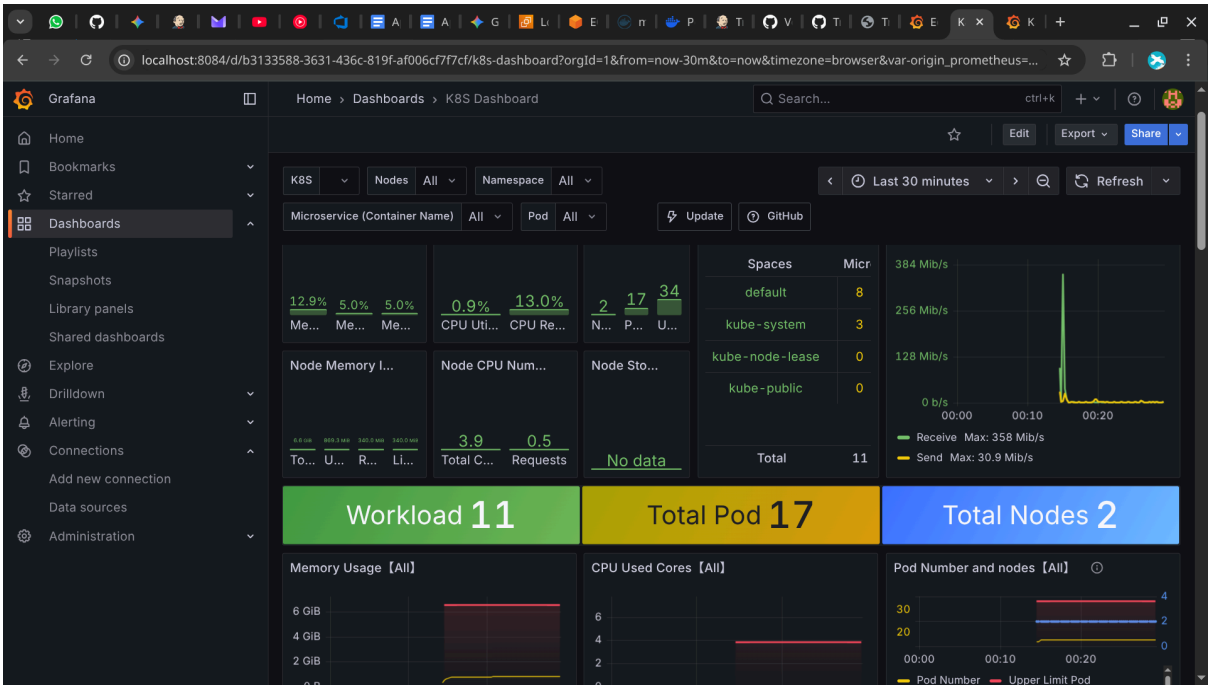
- Install Helm: Install Helm on the Jenkins server.

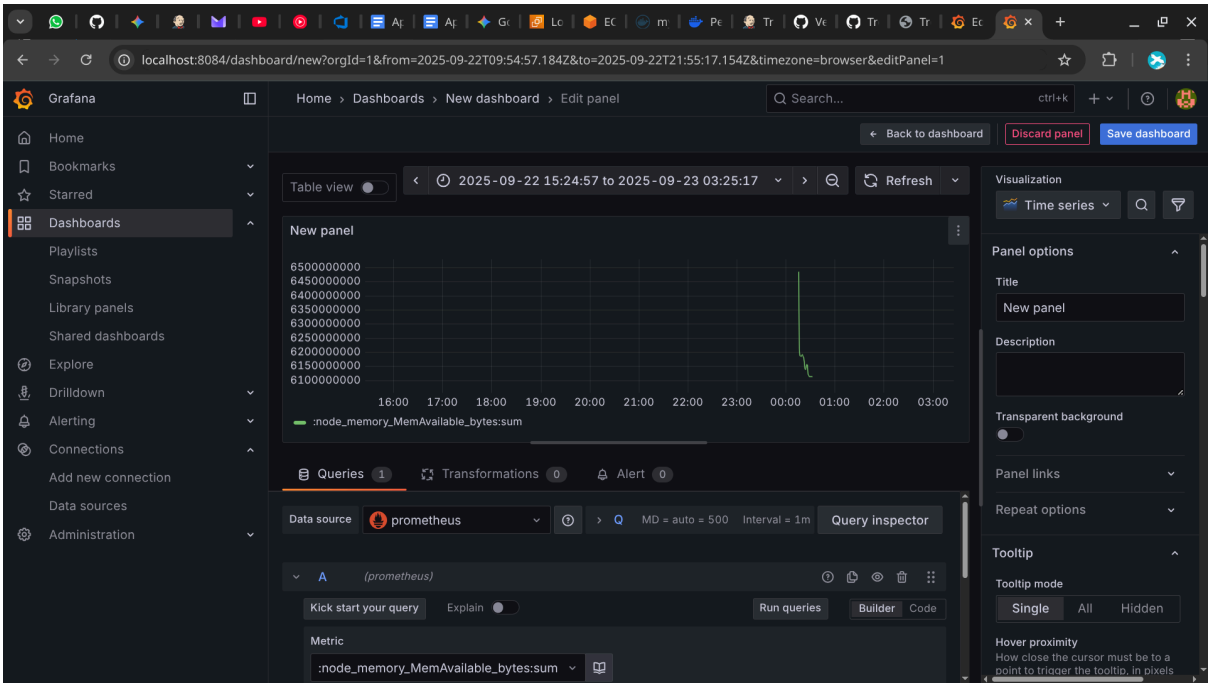
- Deploy Monitoring Stack: Use Helm to deploy Prometheus and Grafana to your EKS cluster.

    1. helm install prometheus prometheus-community/kube-prometheus-stack

    2. helm install grafana grafana/grafana

- Access Grafana: Port-forward the Grafana service to your local machine and use the admin password from the Kubernetes secret to log in.





Step 16 : Output