

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

```
Trend main > nano Dockerfile
Trend main ? > docker build -t trend-image .
[*] Building 19.3s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 106B
=> [internal] load metadata for docker.io/nginx:stable-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 9.29MB
=> [1/2] FROM docker.io/nginx:stable-alpine@sha256:8f2bcf97c473dfe311e79a510ee540ee02e28c1e6a64e1ef89bfad32574ef18
=> => resolve docker.io/nginx:stable-alpine@sha256:8f2bcf97c473dfe311e79a510ee540ee02e28c1e6a64e1ef89bfad32574ef18
=> => sha256:87273e8eb118b64e6d2c1e7e9f74b456bdc3991c22871f42b68622c317994c7 2.49kB / 2.49kB
=> => sha256:a97d82f789e2e0e735e48a697aec360e22cb42a0f4b405d47781976e814e478ed 18.77kB / 18.77kB
=> => sha256:95be8f4718a48e45cabc4f6e19f7aed08101465aca88a577b084a57ee42eb1b 628B / 628B
=> => sha256:8f2bcf97c473dfe311e79a510ee540ee02e28c1e6a64e1ef89bfad32574ef18 10.32kB / 10.32kB
=> => sha256:8568f40e3c6d227d81398f408c93aee216df5cfa814041a91451f8e921a12 3.64MB / 3.64MB
=> => sha256:6d998b1fd4a4752ed1b278e90c363cad99b0394fc34c73a2e4724084221d1 1.79MB / 1.79MB
=> => sha256:62c31b3012d33aa8a2fe15793525a9d6dbf9c57f1f9ec22b88a2f869b1c3e 954B / 954B
=> => sha256:85a9985bea3a96919f33aaec7cec165a06ffc948c8de0410026c0d516c7016 404B / 404B
=> => sha256:8485f215401f7b3ec3cb9a4c66eb912c076fad17aa985155b6c57e132c2042 1.21kB / 1.21kB
=> => sha256:f2d0d37ae31d4ab6192eeaa9aaf8c1c8366ddb1db36cb22c98fd2fd259c1542c 1.40kB / 1.40kB
=> => sha256:e585282f4286292f32208111c879a44e86b5848afce1246fdd32170864108 15.51MB / 15.51MB
=> => extracting sha256:8368f06e3c6d237d81398f408c93aee216df5cfa814041a91451f8e921a12 0.24s
=> => extracting sha256:6d998b1fd4a4752ed1b278e90c363cad99b0394fc34c73a2e4724084221d1 0.3s
=> => extracting sha256:95be8f4718a48e45cabc4f6e19f7aed08101465aca88a577b084a57ee42eb1b 0.0s
=> => extracting sha256:62c31b3012d33aa8a2fe15793525a9d6dbf9c57f1f9ec22b88a2f869b1c3e 0.0s
=> => extracting sha256:85a9985bea3a96919f33aaec7cec165a06ffc948c8de0410026c0d516c7016 0.0s
=> => extracting sha256:8485f215401f7b3ec3cb9a4c66eb912c076fad17aa985155b6c57e132c2042 0.0s
=> => extracting sha256:f2d0d37ae31d4ab6192eeaa9aaf8c1c8366ddb1db36cb22c98fd2fd259c1542c 0.0s
=> => extracting sha256:e585282f4286292f32208111c879a44e86b5848afce1246fdd32170864108 1.1s
=> [2/2] COPY dist /usr/share/nginx/html
=> => exporting image
=> => exporting layers
=> => writing image sha256:2c4ef818d92939d14750b485d69efe98a88548de1268a0affae7fc9383982e5
=> => naming to docker.io/library/trend-image 0.0s
```

```
Trend main ? > docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
trend-image         latest     2c4ef810d929  9 minutes ago 57.4MB
hishamali/hello-guvi-geek build-1    2d0c246fd511 11 hours ago 134MB
hishamali/hello-guvi-geek latest     2d0c246fd511 11 hours ago 134MB
hello-world         latest     1b44b5a3e06a 2 months ago 10.1kB

Trend main ? > docker tag trend-image:latest mythili121/trend-app:latest

Trend main ? > docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
mythili121/trend-app latest     2c4ef810d929 10 minutes ago 57.4MB
trend-image         latest     2c4ef810d929 10 minutes ago 57.4MB
hishamali/hello-guvi-geek build-1    2d0c246fd511 11 hours ago 134MB
hishamali/hello-guvi-geek latest     2d0c246fd511 11 hours ago 134MB
hello-world         latest     1b44b5a3e06a 2 months ago 10.1kB

Trend main ? > █
```

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/AmazonEKSEKSWorkerNodePolicy"
}

```

```

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
```

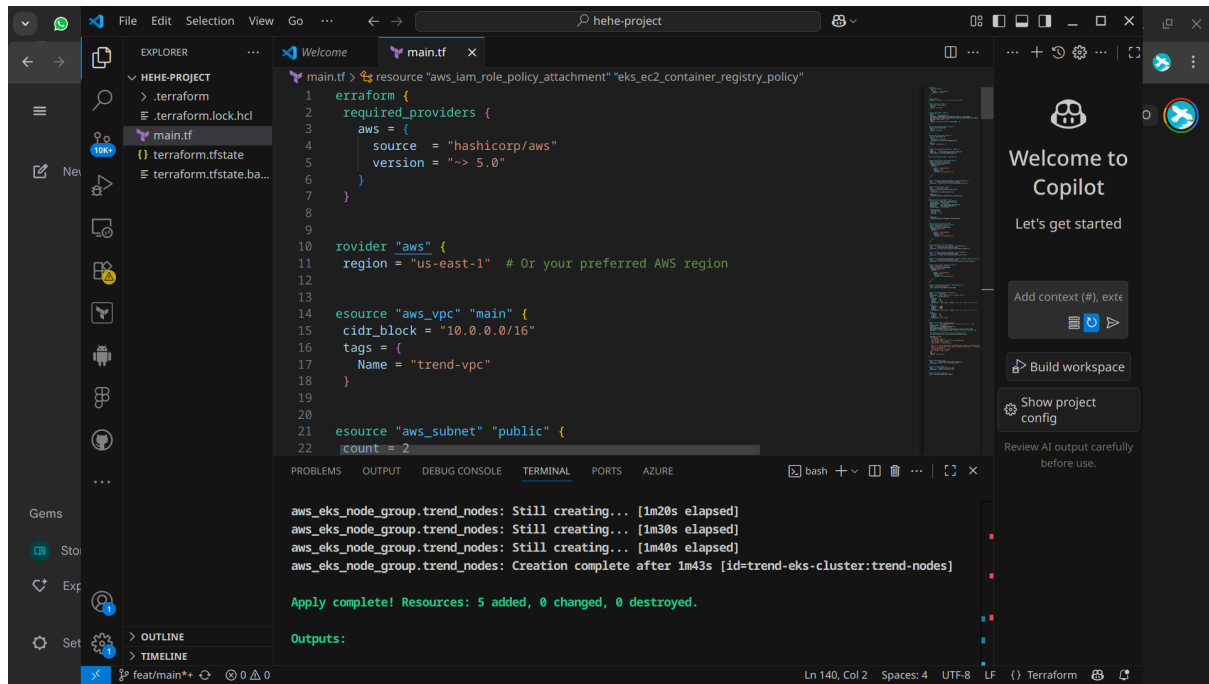
```
sudo apt-get install -y docker.io git openjdk-11-jdk
sudo systemctl start docker
sudo usermod -a -G docker ubuntu
```

```
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"
}
```

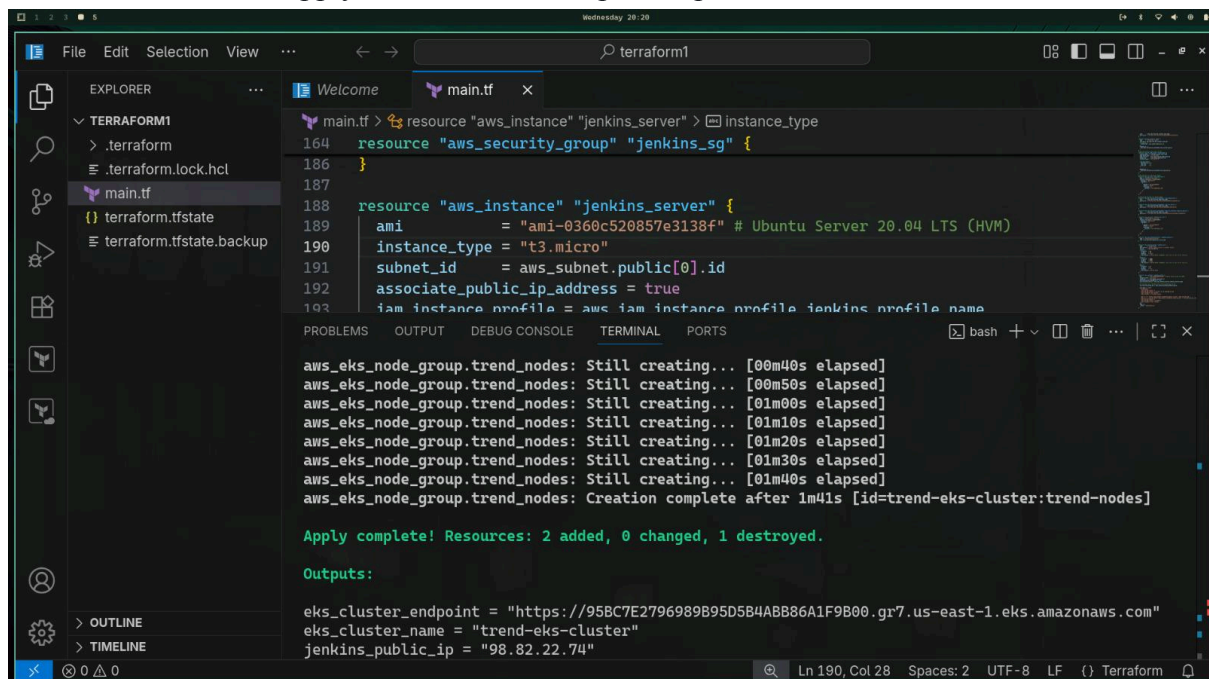
```
output "eks_cluster_name" {
  value      = aws_eks_cluster.main.name
  description = "Name of the EKS cluster"
}
```

```
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
 - terraform init
 - terraform plan
 - terraform apply
- The terraform apply command will output the public IP of the Jenkins server.



EC2 > Instances

Instances (1/3) Info [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
trend-eks-clus...	i-0d71d821fe9abbd7	Running	t3.micro	3/3 checks passed	View alarms
JenkinsServer	i-08974854f1cdac475	Running	t3.micro	3/3 checks passed	View alarms
trend-eks-clus...	i-05eabec1e591b86e4	Running	t3.micro	3/3 checks passed	View alarms

i-08974854f1cdac475 (JenkinsServer)

[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

Instance summary Info

Instance ID i-08974854f1cdac475	Public IPv4 address 98.82.22.74 open address	Private IPv4 addresses 10.0.0.24
IPv6 address -	Instance state Running	Public DNS -
Hostname type	Private IP DNS name (IPv4 only)	

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Load balancers

Load balancers (1/1) [Actions](#) [Create load balancer](#)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

Name	State	Type	Scheme	IP address type	VPC ID
a33e67da46f79448b93...	Active	network	Internet-facing	IPv4	vpc-0b97bef96815ede43

Load balancer: a33e67da46f79448b9338a67f5302aa1

Internet-facing	Z26RNL4JYFTOTI	subnet-07276a11b36d4696 us-east-1b (use1-az6) subnet-054312bc7d8ef026f us-east-1a (use1-az4)	September 22, 2025, 23:30 (UTC+05:30)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:551210489378:loadbalancer/net/a33e67da46f79448b9338a67f5302aa1/3b73ad892c548687		DNS name Info a33e67da46f79448b9338a67f5302aa1-3b73ad892c548687.elb.us-east-1.amazonaws.com (A Record)	

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 6 : Create a dockhub.

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

```
Trend main ? > docker login

USING WEB-BASED LOGIN

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

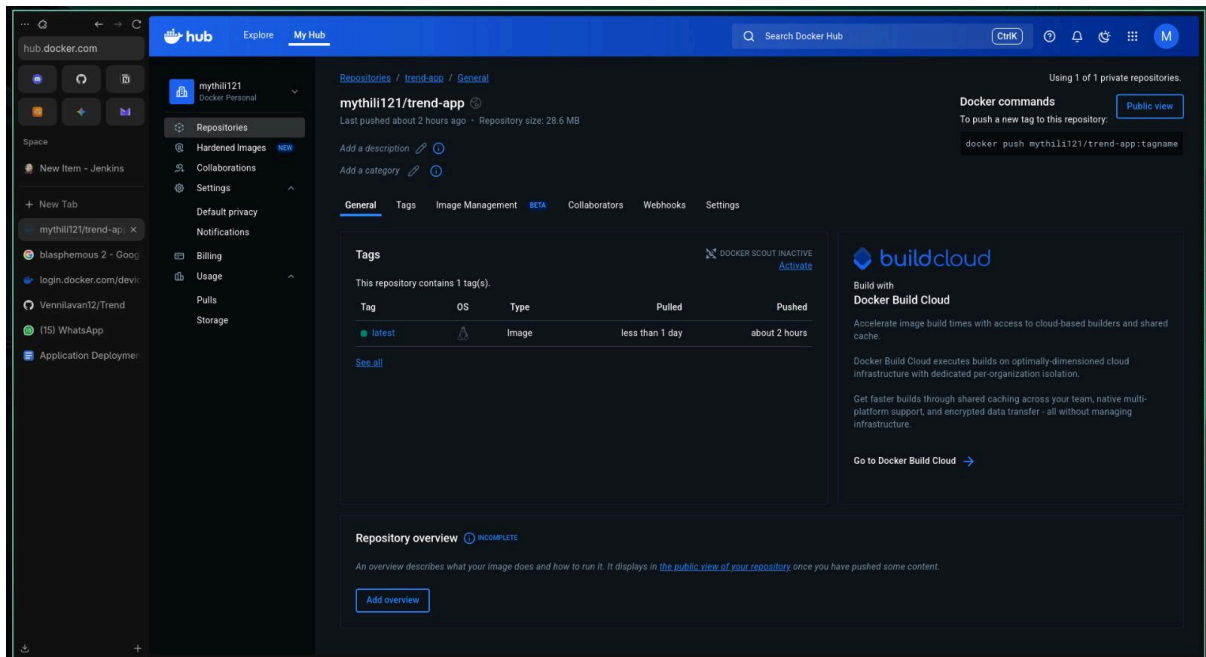
Trend main ? > █
```

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

```

~ > kubectl get svc
NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes           ClusterIP           172.20.0.1      <none>            443/TCP          61m
trend-app-service    LoadBalancer       172.20.164.215  a45e369911f5e4a10a5010a3c0fecc98-1a28e3e7b423b1ac.elb.us-east-1.amazonaws.com  80:31569/TCP    116s
~ > █

```

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
-

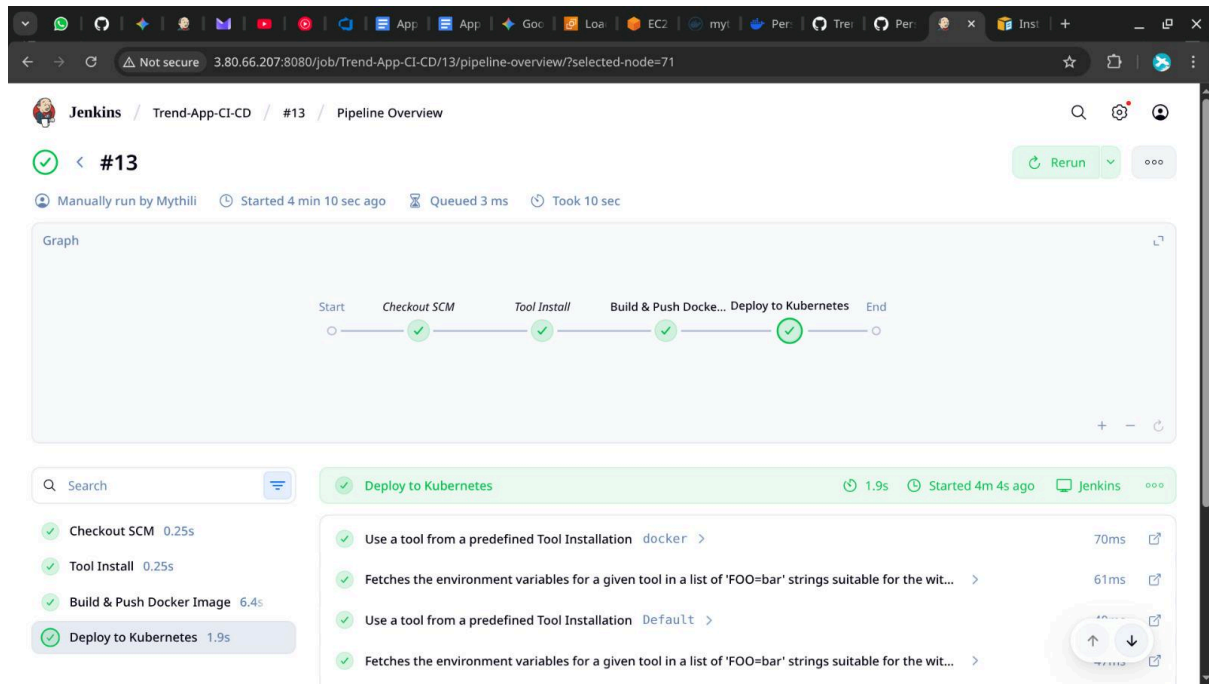
Step 12 : Jenkins CI/CD set up.

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)

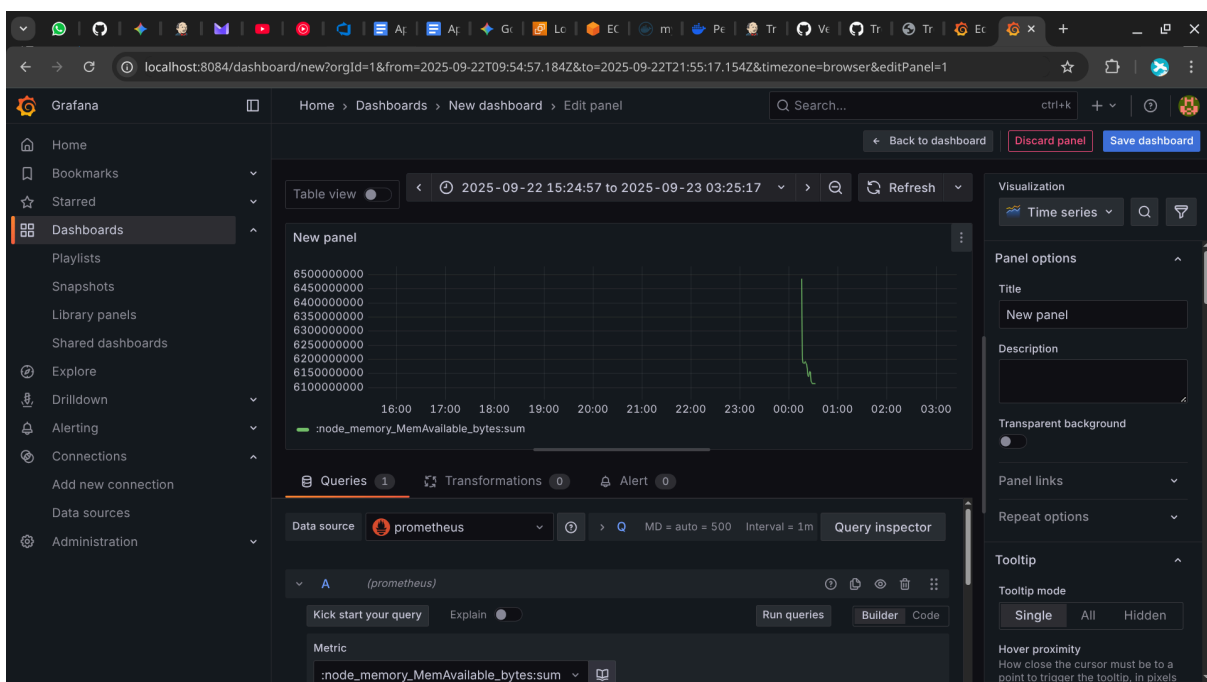
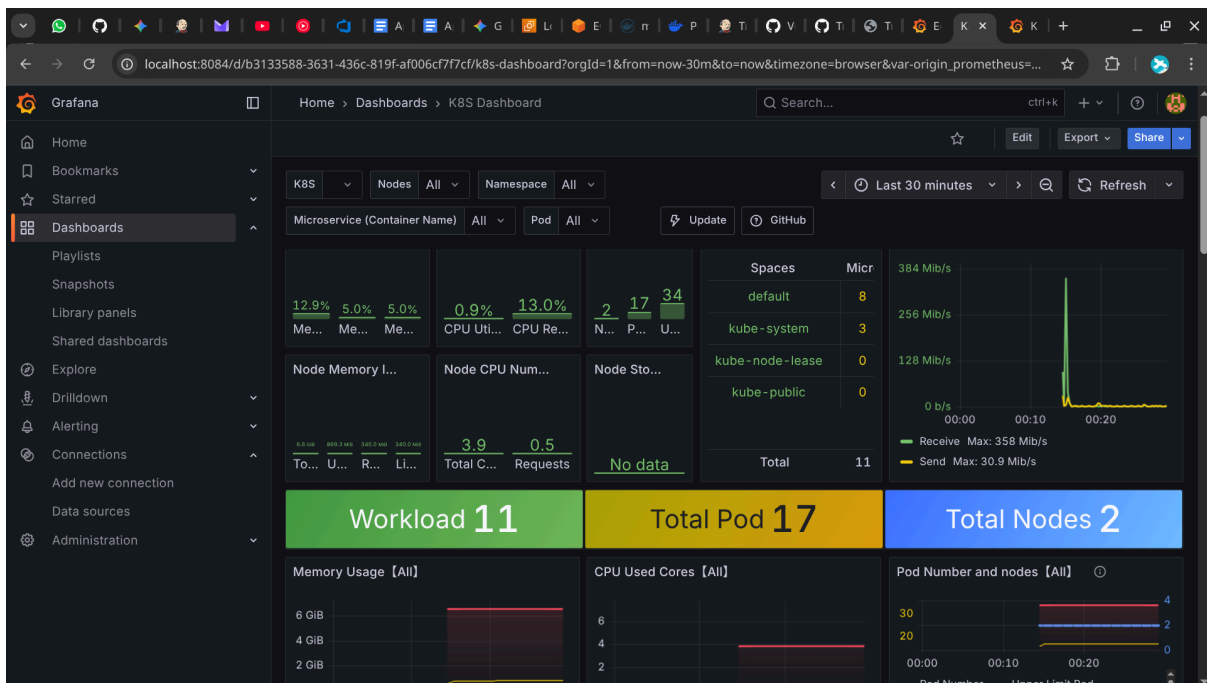
- Install Helm: Install Helm on the Jenkins server.

- 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

