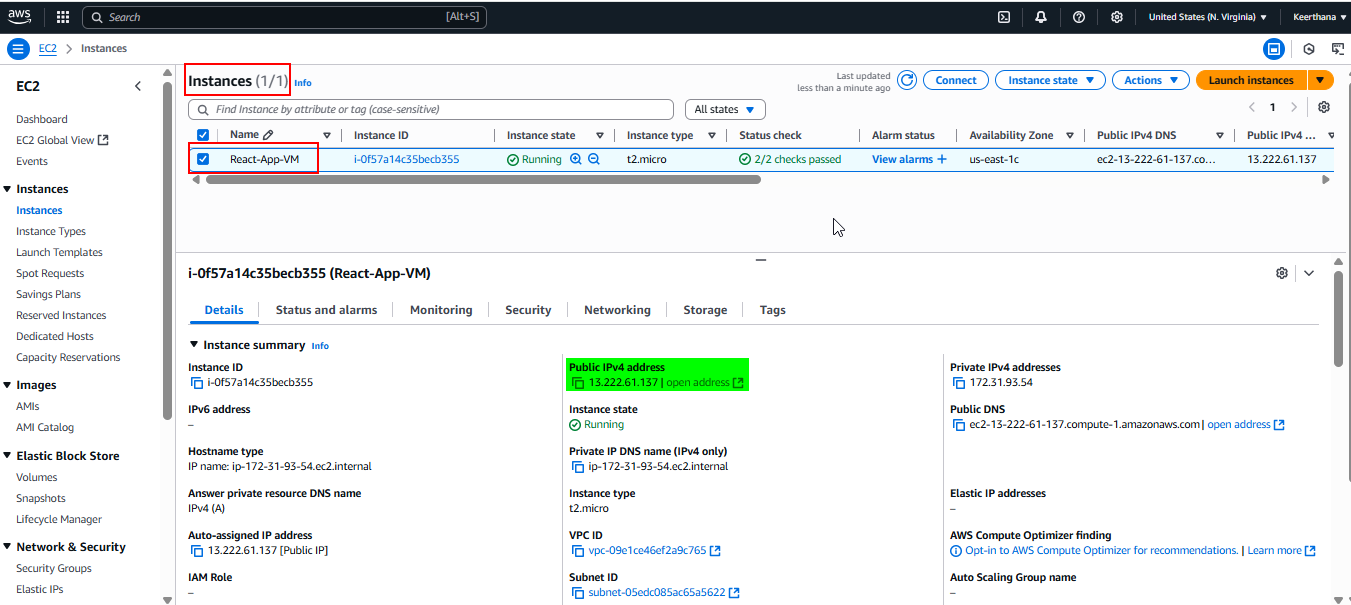
**React Application Deployment**

**Launch an EC2 instance**

****

**Install Docker**

sudo apt update && sudo apt upgrade -y

sudo apt install -y docker.io

sudo systemctl enable docker

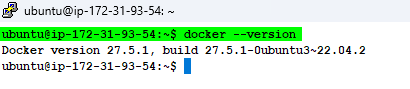
sudo systemctl start docker

sudo usermod -aG docker ubuntu

sudo usermod -aG docker $USER

sudo systemctl restart docker

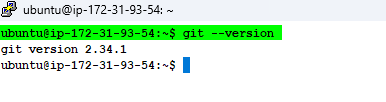
docker --version



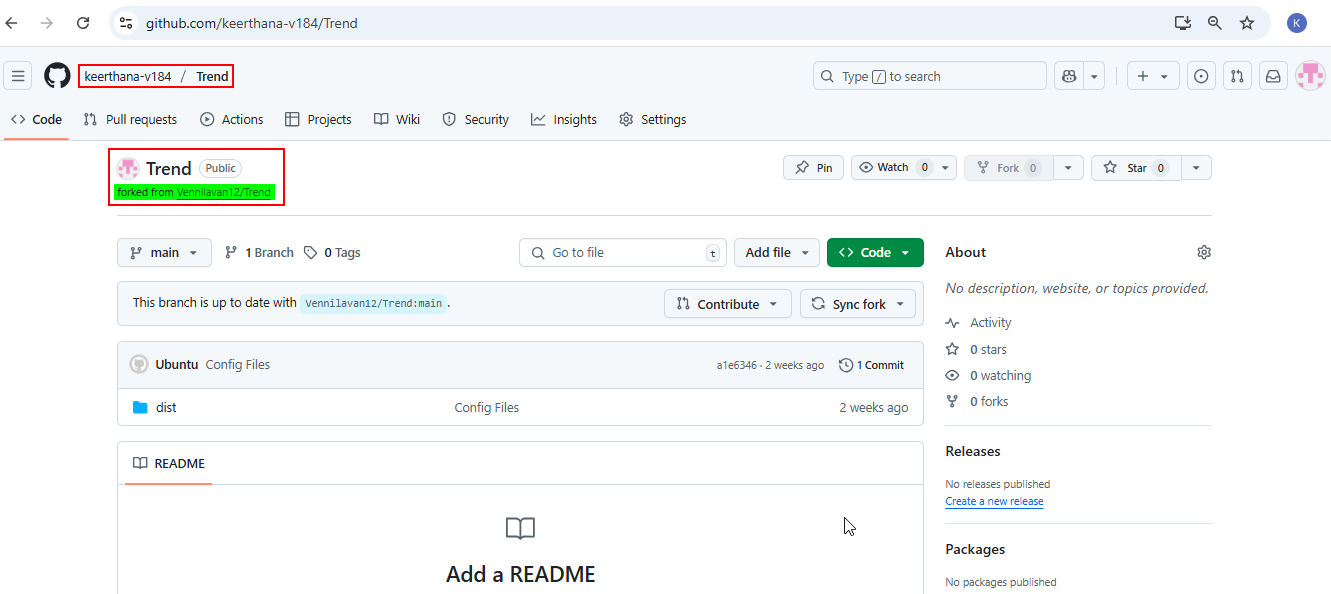
**Install Git**

sudo apt update -y

sudo apt install -y git



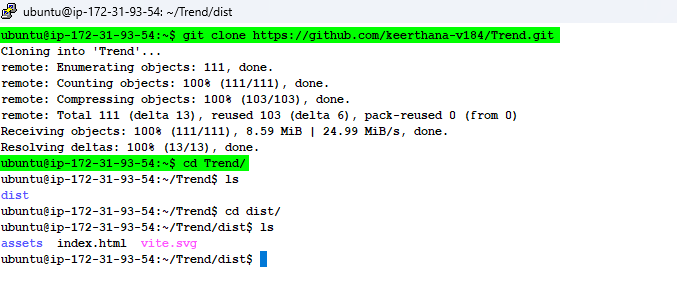
**Forked the given Repo to my GitHub:**



**Clone Your Forked Repo**

git clone https://github.com/keerthana-v184/Trend.git

cd Trend



**Create Dockerfile**

FROM nginx:alpine

RUN rm -rf /usr/share/nginx/html/\*

COPY dist/ /usr/share/nginx/html/

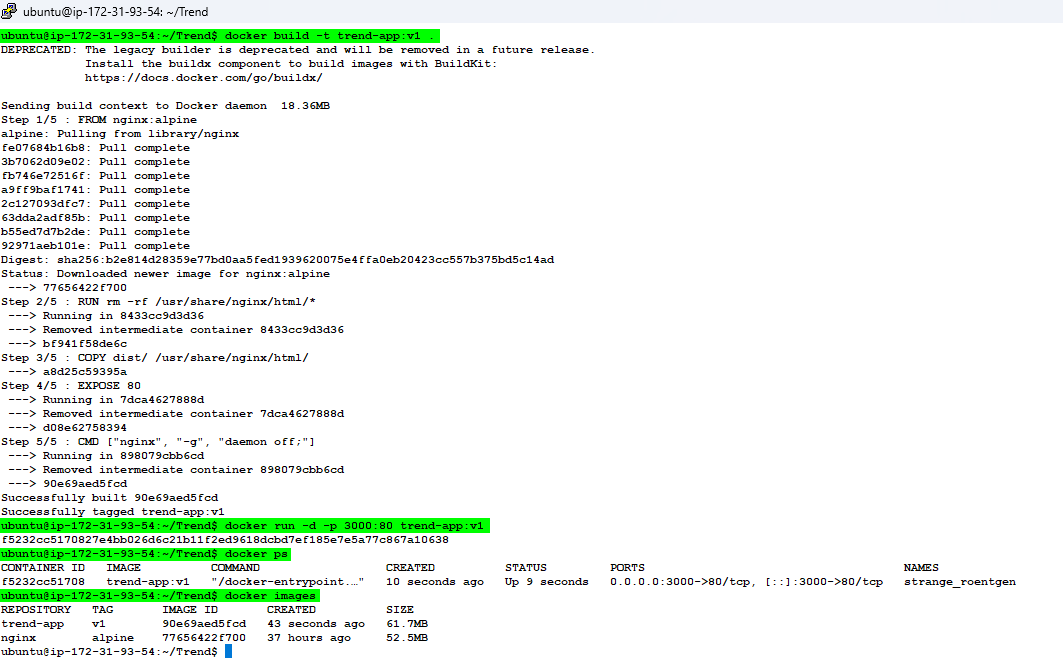
EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

**Build and Run Docker Image**

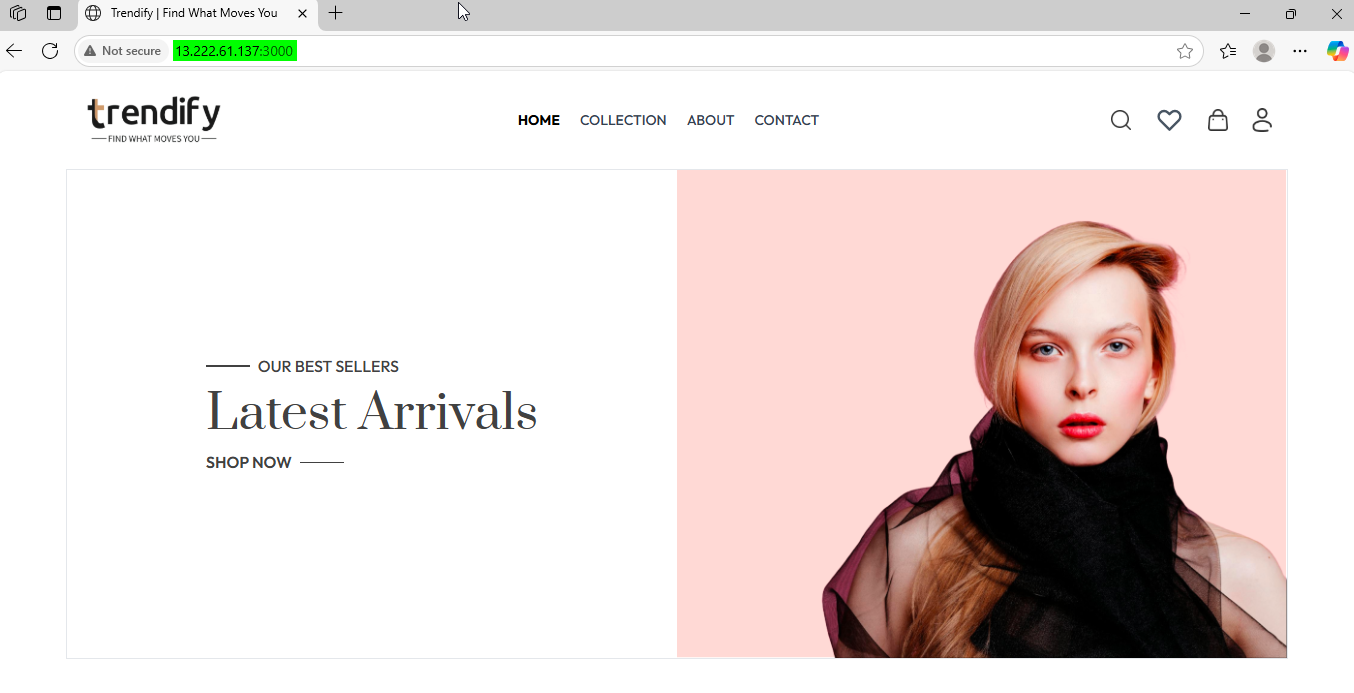
docker build -t trend-app:v1 .

docker run -d -p 3000:80 trend-app:v1

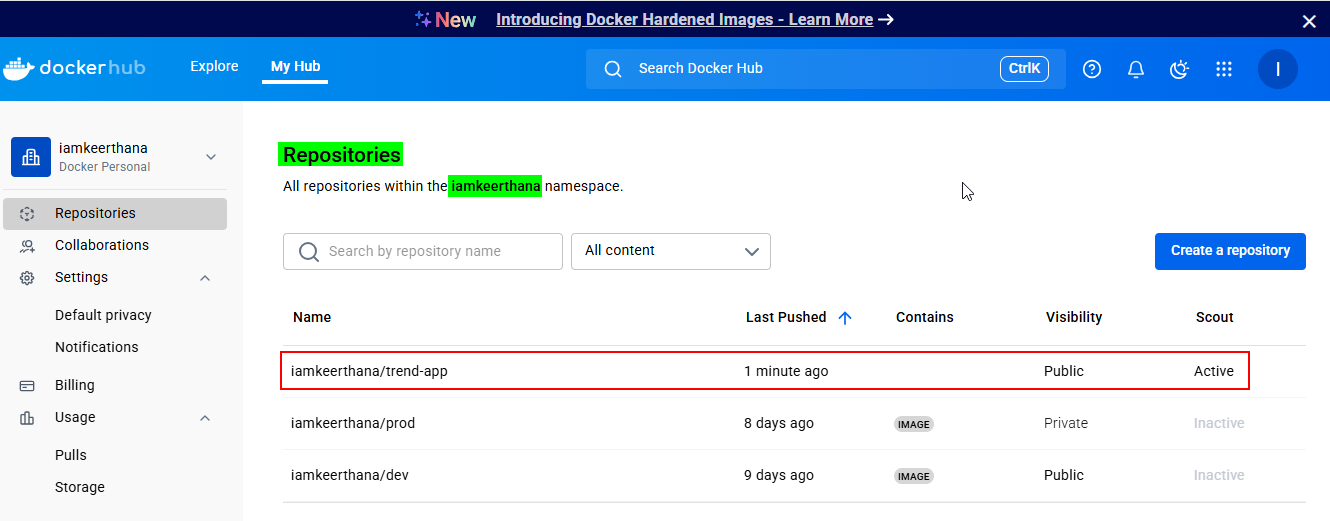
****

**Search Public IP in Browser**

* <http://13.222.61.137:3000/>

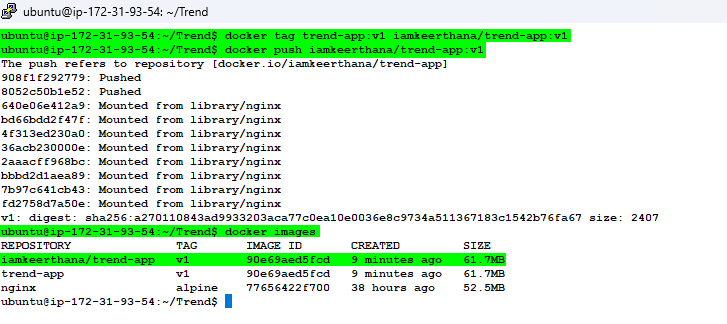
****

**Create Docker hub repo:**

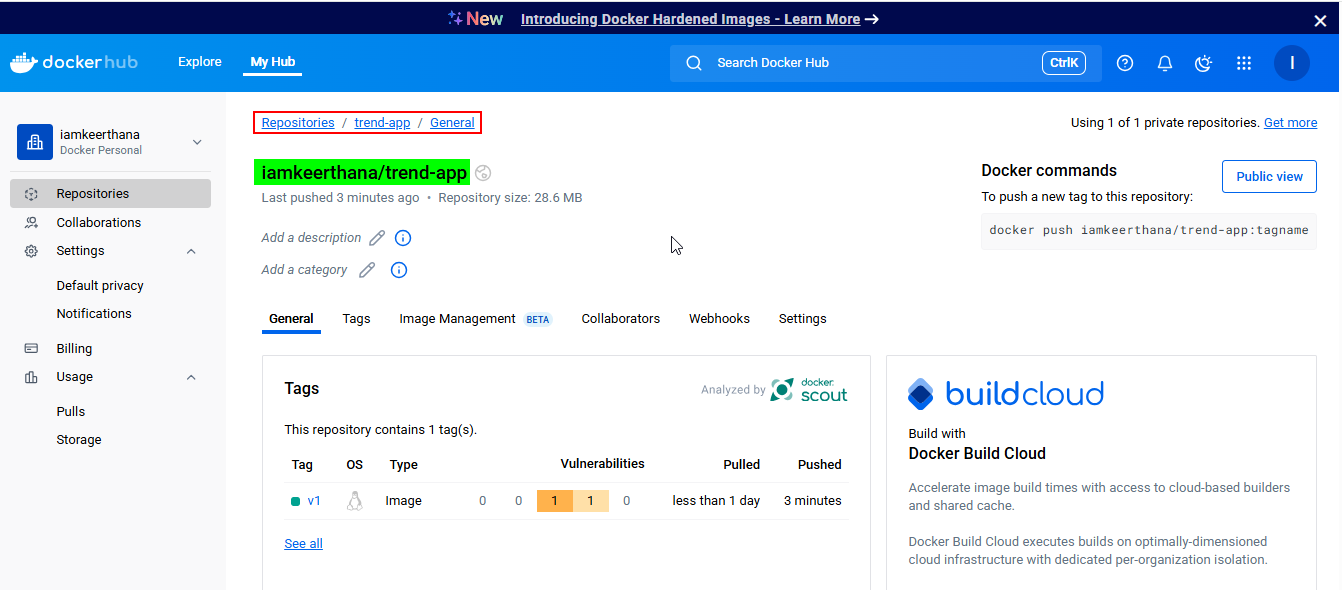


**Tag image and Push to DockerHub:**

* docker tag trend-app:v1 iamkeerthana/trend-app:v1
* docker push iamkeerthana/trend-app:v1



**Image Pushed to Dockerhub:**



**Install AWS CLI**

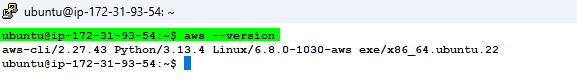
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

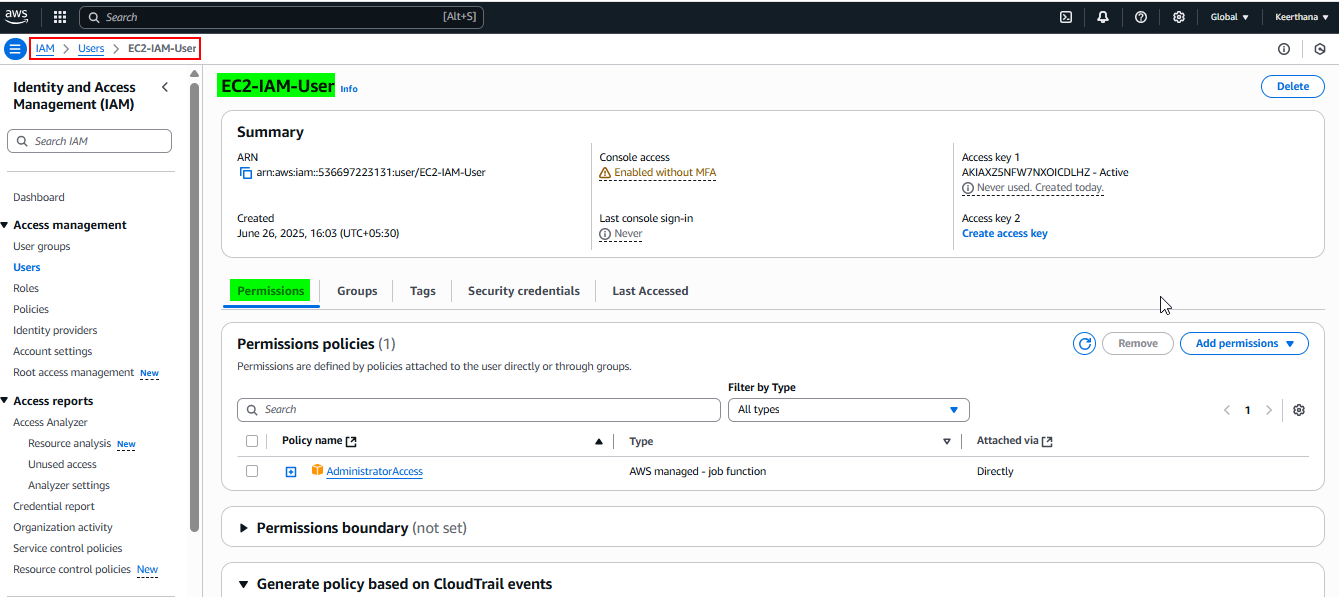
sudo ./aws/install

sudo ./aws/install –update

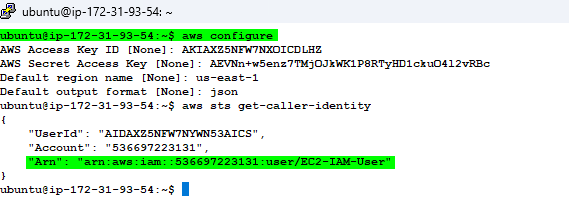
aws --version



**Create IAM user and Configure:**



**Aws configure**



**Install Terraform**

sudo apt update -y

sudo apt install -y wget unzip gnupg software-properties-common

wget -O- https://apt.releases.hashicorp.com/gpg | \

gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \

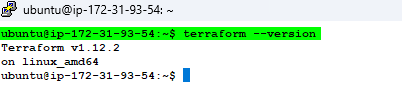
https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | \

sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update -y

sudo apt install terraform -y

terraform --version

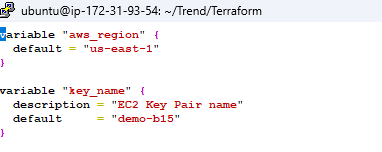


**Terraform**

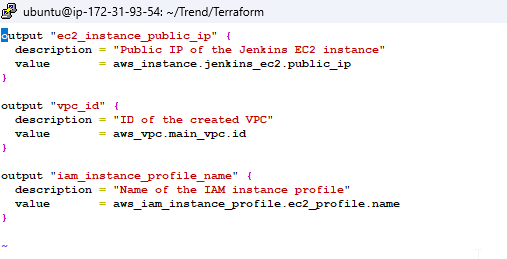
**Create Infrastructure with Jenkins:**

mkdir tf-infra && cd tf-infra

**Create variables.tf**

****

**Create outputs.tf**

****

**Create main.tf**

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.45.0"

}

}

required\_version = ">= 1.5.0"

}

provider "aws" {

region = var.aws\_region

}

resource "aws\_vpc" "main\_vpc" {

cidr\_block = "10.0.0.0/16"

enable\_dns\_support = true

enable\_dns\_hostnames = true

tags = {

Name = "main-vpc"

}

}

resource "aws\_internet\_gateway" "igw" {

vpc\_id = aws\_vpc.main\_vpc.id

tags = {

Name = "main-igw"

}

}

resource "aws\_subnet" "public\_subnet\_1" {

vpc\_id = aws\_vpc.main\_vpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

map\_public\_ip\_on\_launch = true

tags = {

Name = "main-public-subnet-1"

}

}

resource "aws\_subnet" "public\_subnet\_2" {

vpc\_id = aws\_vpc.main\_vpc.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-east-1b"

map\_public\_ip\_on\_launch = true

tags = {

Name = "main-public-subnet-2"

}

}

resource "aws\_route\_table" "public\_rt" {

vpc\_id = aws\_vpc.main\_vpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.igw.id

}

tags = {

Name = "main-public-rt"

}

}

resource "aws\_route\_table\_association" "public\_assoc\_1" {

subnet\_id = aws\_subnet.public\_subnet\_1.id

route\_table\_id = aws\_route\_table.public\_rt.id

}

resource "aws\_route\_table\_association" "public\_assoc\_2" {

subnet\_id = aws\_subnet.public\_subnet\_2.id

route\_table\_id = aws\_route\_table.public\_rt.id

}

resource "aws\_security\_group" "jenkins\_sg" {

name = "jenkins-sg"

description = "Allow SSH, HTTP, Jenkins"

vpc\_id = aws\_vpc.main\_vpc.id

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 8080

to\_port = 8080

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 3000

to\_port = 3000

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 80

to\_port = 80

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 = "jenkins-sg"

}

}

resource "aws\_iam\_role" "ec2\_role" {

name = "main-ec2-role"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [{

Action = "sts:AssumeRole"

Effect = "Allow"

Principal = {

Service = "ec2.amazonaws.com"

}

}]

})

}

resource "aws\_iam\_role\_policy\_attachment" "attach\_all" {

for\_each = toset([

"arn:aws:iam::aws:policy/AdministratorAccess",

"arn:aws:iam::aws:policy/AmazonEC2FullAccess",

"arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy",

"arn:aws:iam::aws:policy/AmazonEKSClusterPolicy",

"arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy",

"arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryFullAccess",

"arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess",

"arn:aws:iam::aws:policy/CloudWatchFullAccess",

"arn:aws:iam::aws:policy/IAMFullAccess"

])

role = aws\_iam\_role.ec2\_role.name

policy\_arn = each.value

}

resource "aws\_iam\_instance\_profile" "ec2\_profile" {

name = "main-ec2-profile"

role = aws\_iam\_role.ec2\_role.name

}

resource "aws\_instance" "jenkins\_ec2" {

ami = "ami-0a7d80731ae1b2435"

instance\_type = "t2.medium"

subnet\_id = aws\_subnet.public\_subnet\_1.id

key\_name = var.key\_name

vpc\_security\_group\_ids = [aws\_security\_group.jenkins\_sg.id]

iam\_instance\_profile = aws\_iam\_instance\_profile.ec2\_profile.name

user\_data = <<-EOF

#!/bin/bash

sudo apt update -y

sudo apt install -y docker.io openjdk-17-jdk git curl

curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list

sudo apt update -y

sudo apt install -y jenkins

sudo systemctl enable docker

sudo systemctl start docker

sudo systemctl enable jenkins

sudo systemctl start jenkins

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

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

sudo chmod +x kubectl

sudo mv kubectl /usr/local/bin/

EOF

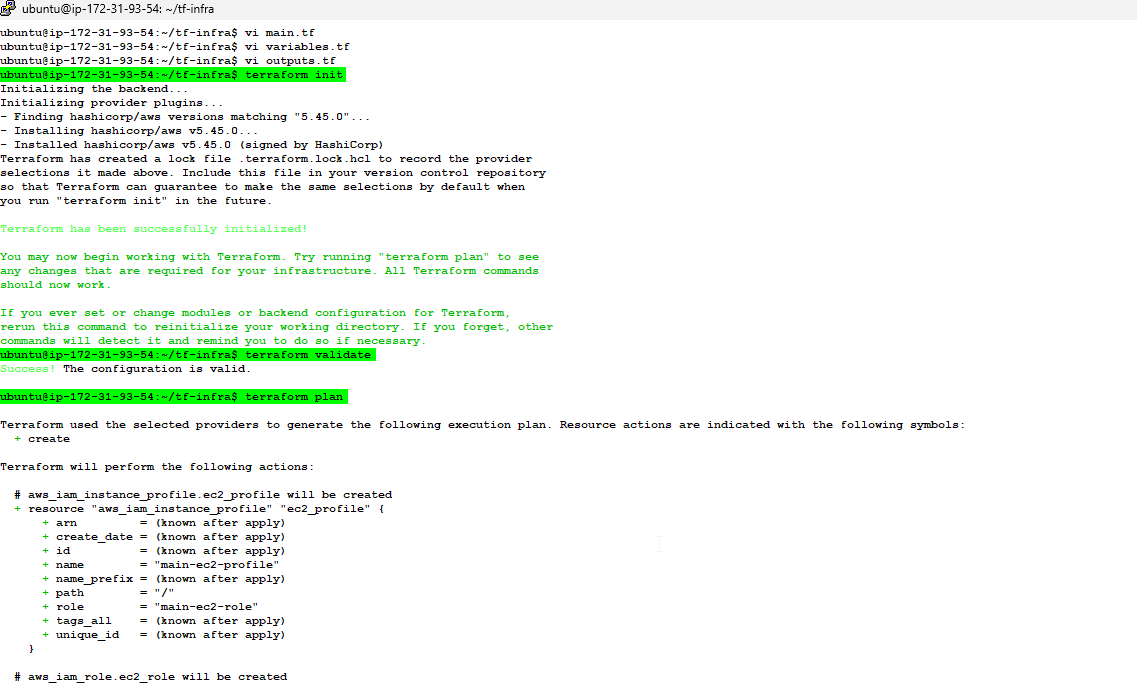
tags = {

Name = "jenkins-instance"

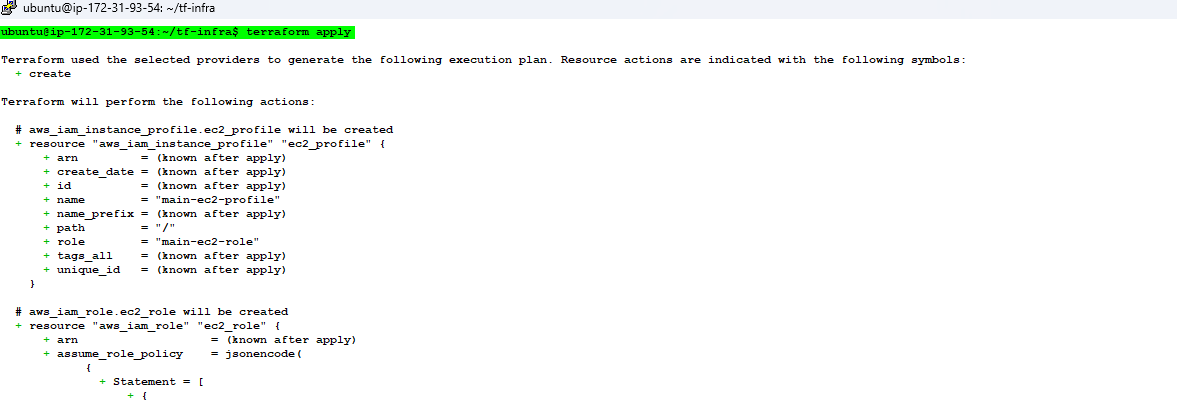
}

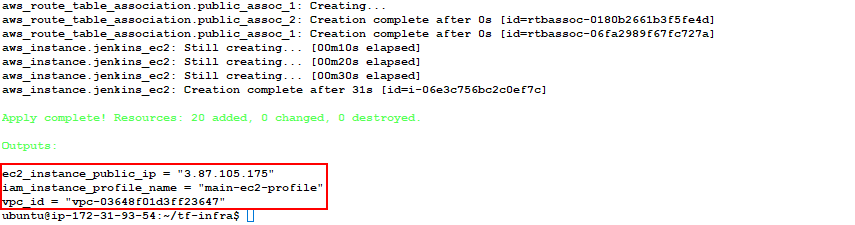
}

**Terraform – Initialization & Validation**

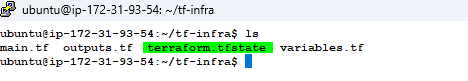
****

**Terraform – Apply**

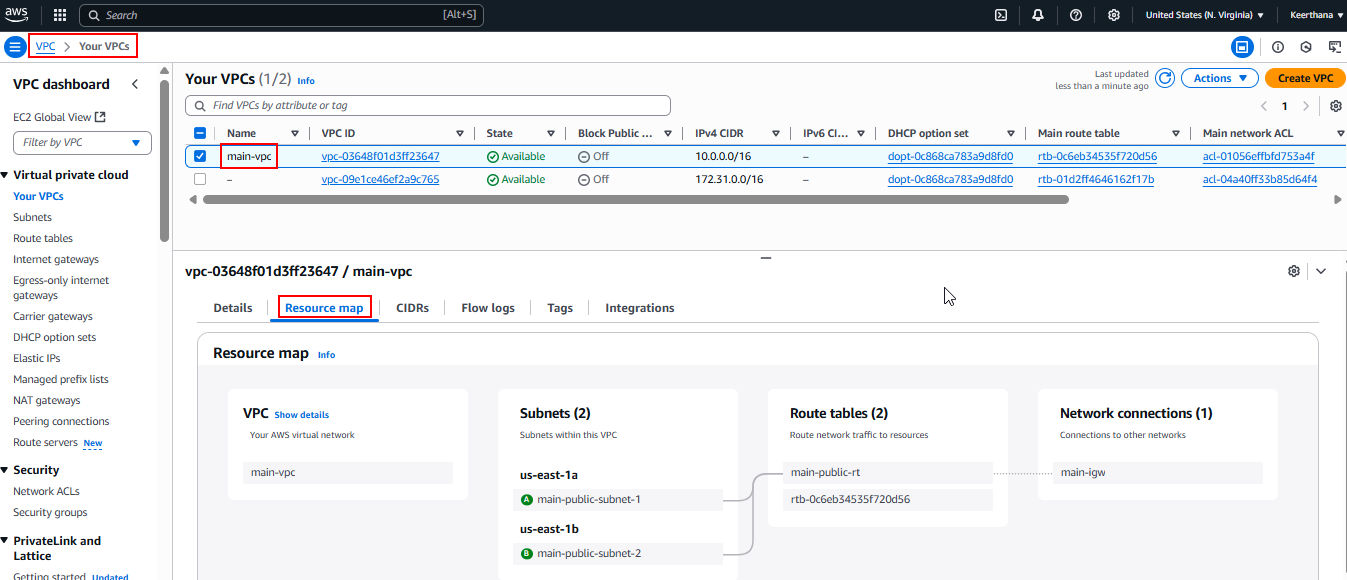
****

****

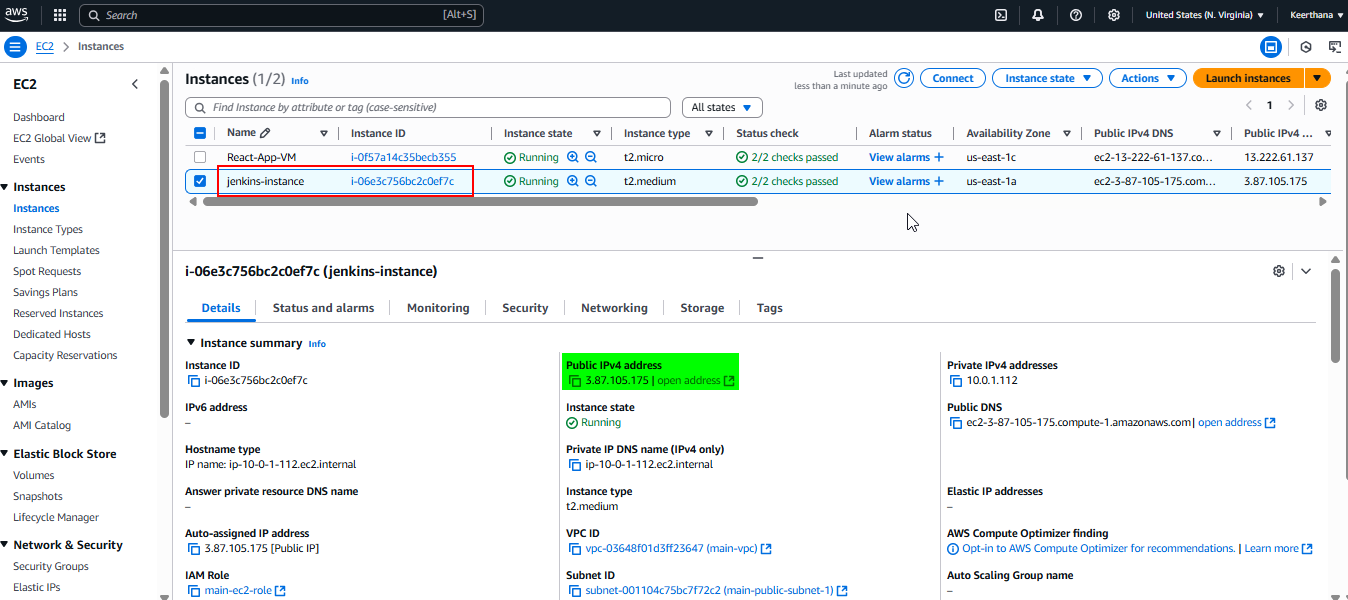
**Terraform.tfstate file got created**

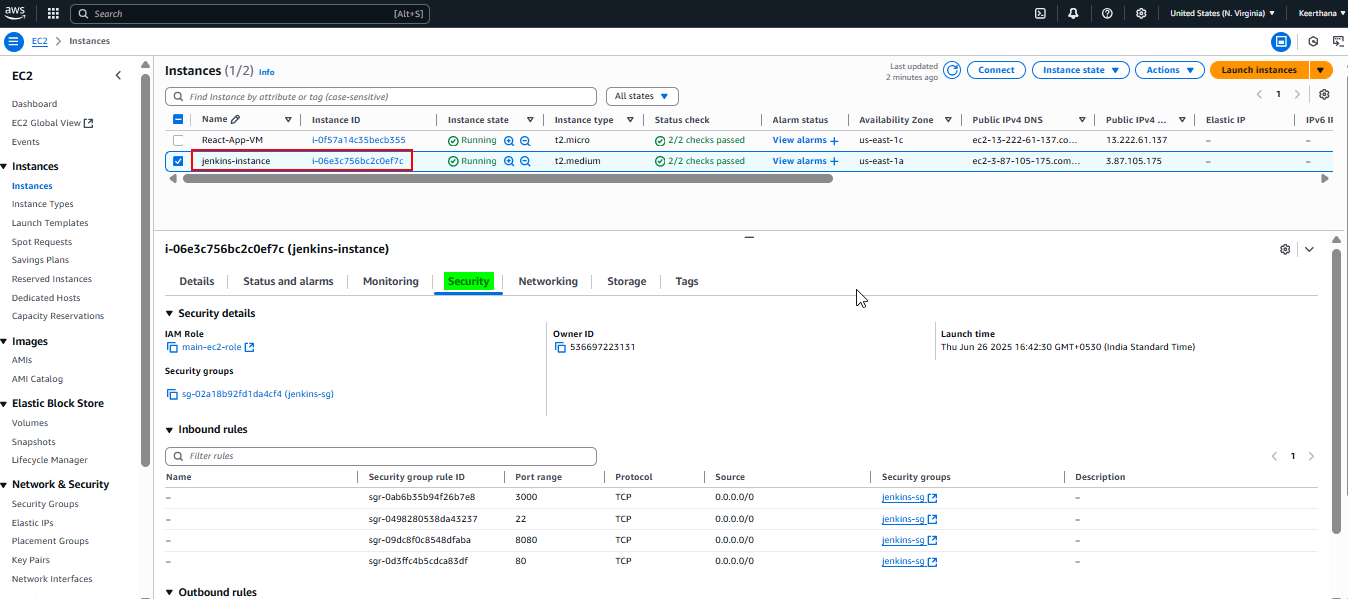
****

**VPC Created**

****

**Jenkins Machine Created**

****

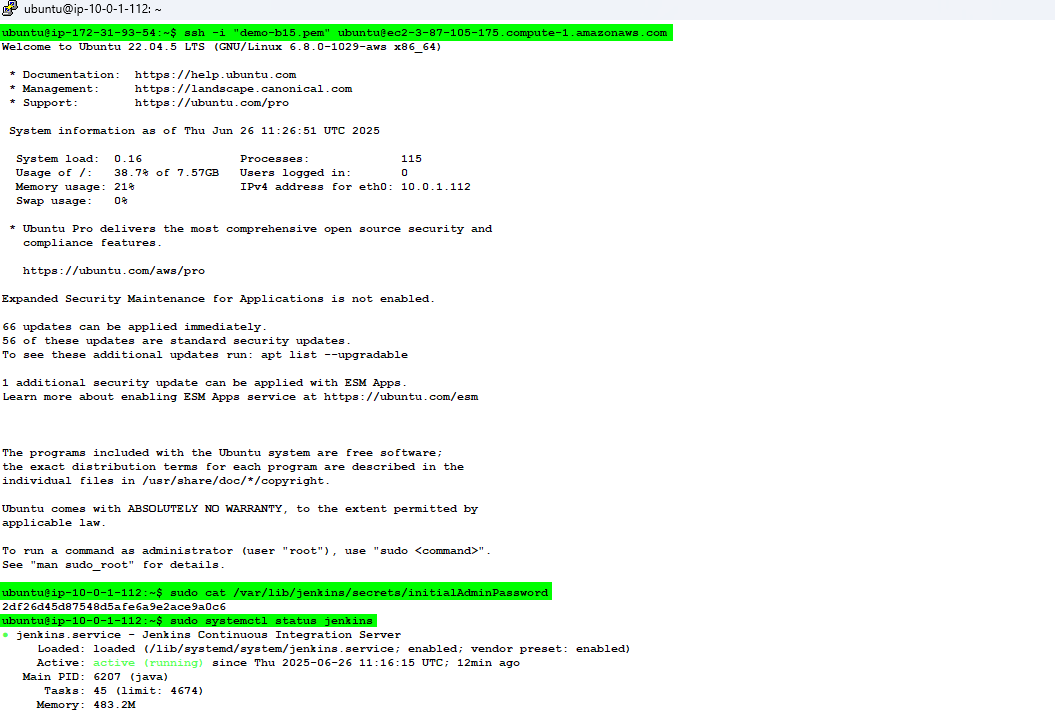
****

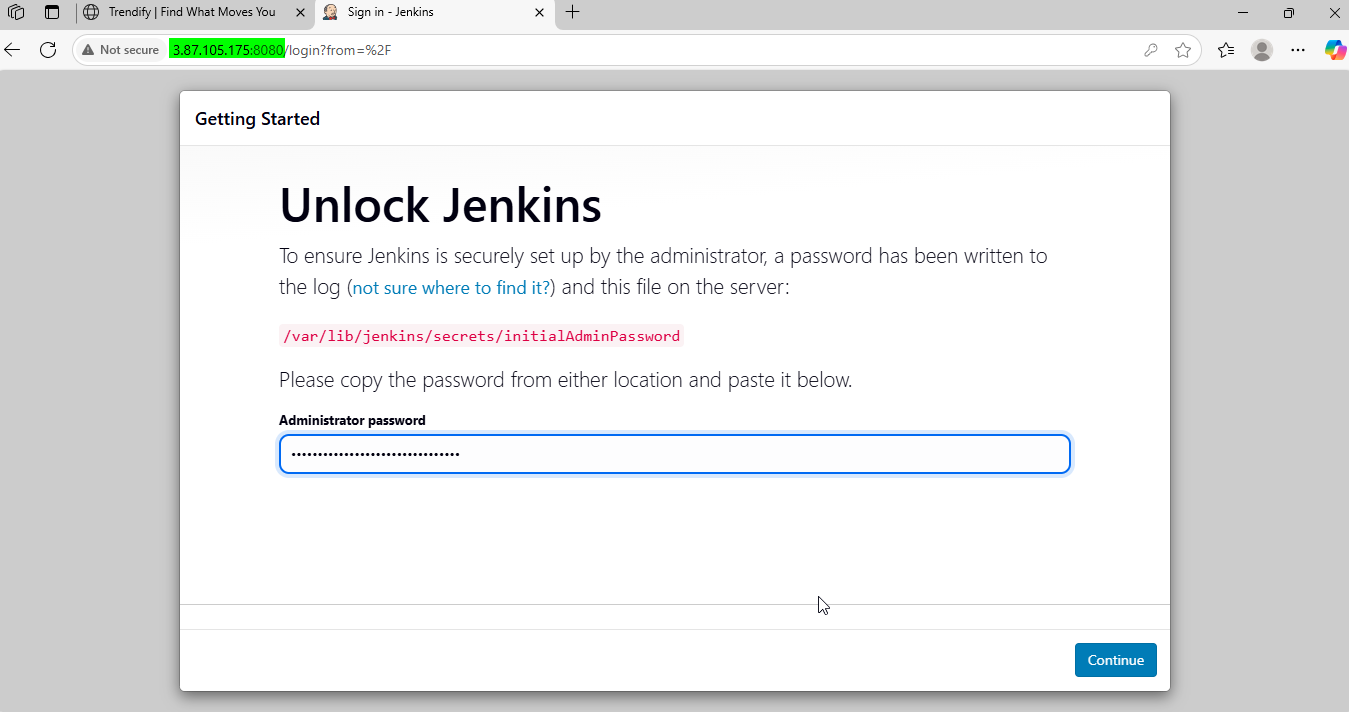
**SSH into Jenkins Instance from your local EC2**

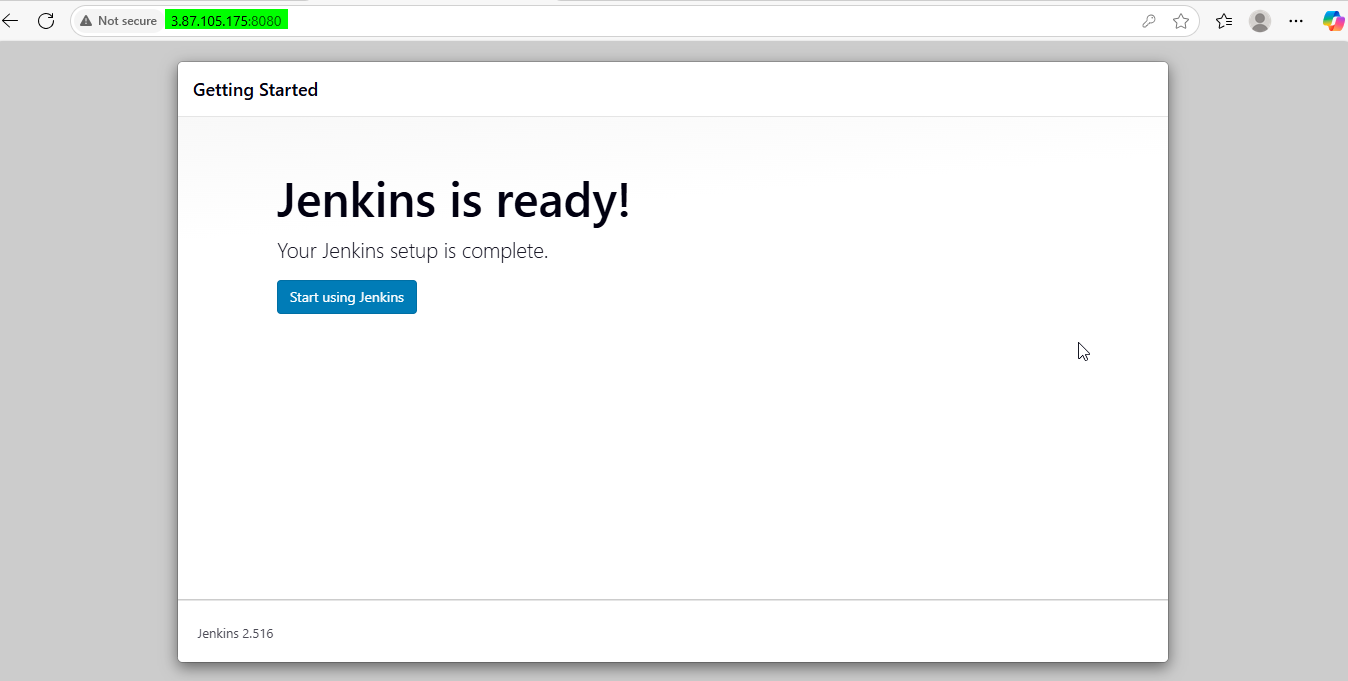
ssh -i "demo-b15.pem" [ubuntu@ec2-3-87-105-175.compute-1.amazonaws.com](mailto:ubuntu@ec2-3-87-105-175.compute-1.amazonaws.com)

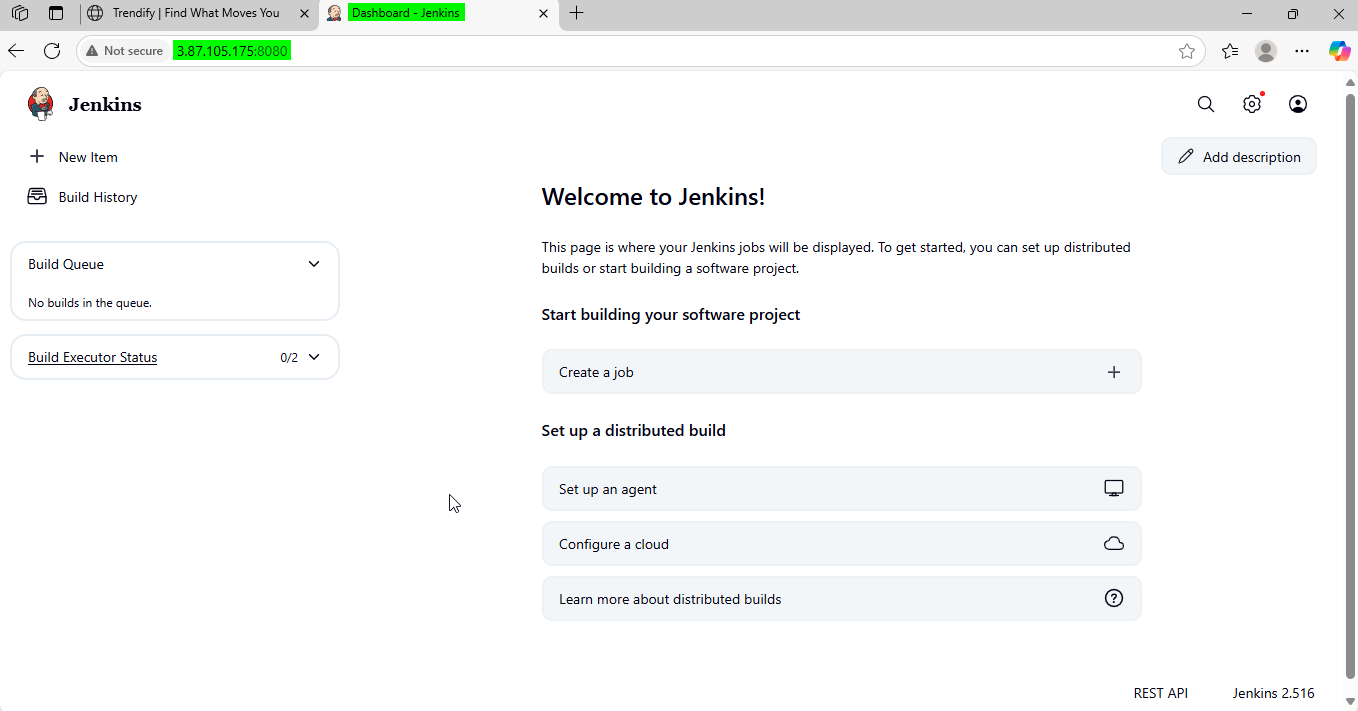
**Check Jenkins status and get Jenkins initial admin password**

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

****

****

****

****

**Install kubectl**

curl -LO "https://dl.k8s.io/release/$(curl -Ls https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

sudo chmod +x kubectl

sudo mv kubectl /usr/local/bin/

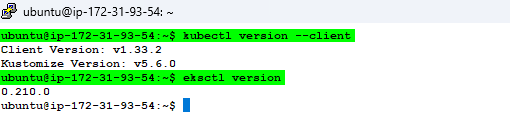
kubectl version --client

**Install eksctl**

curl --silent --location "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

****

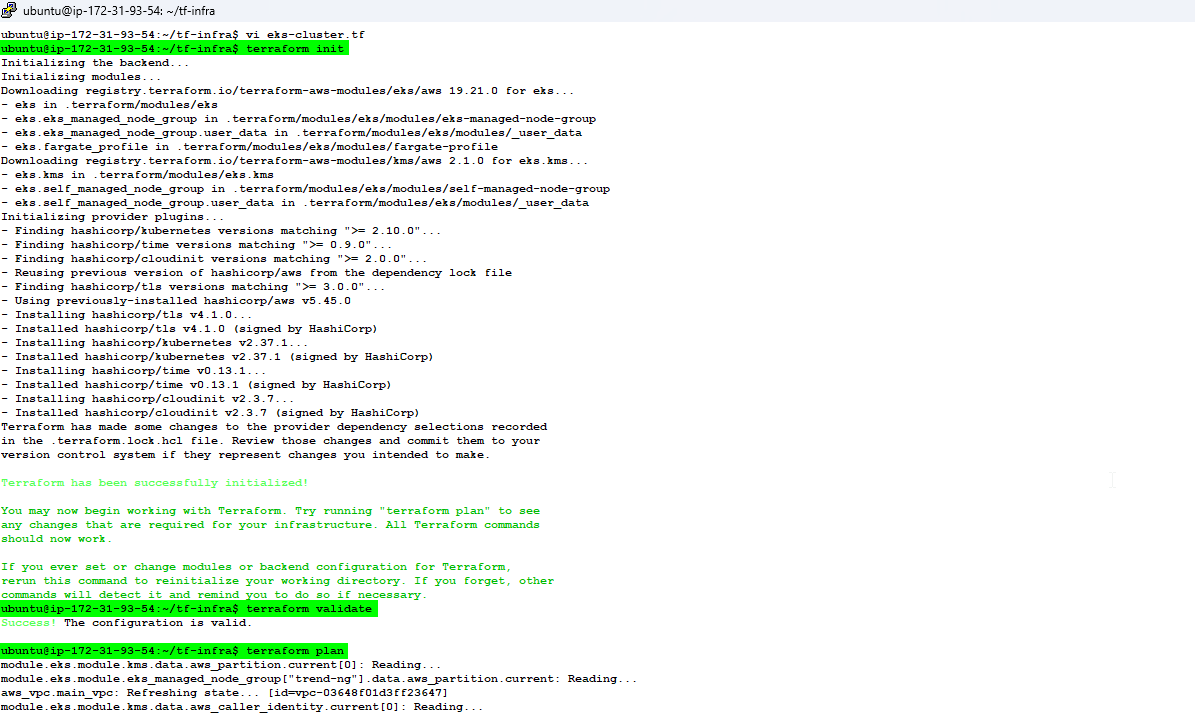
**EKS Cluster Setup**

**Create Cluster and Node Group**

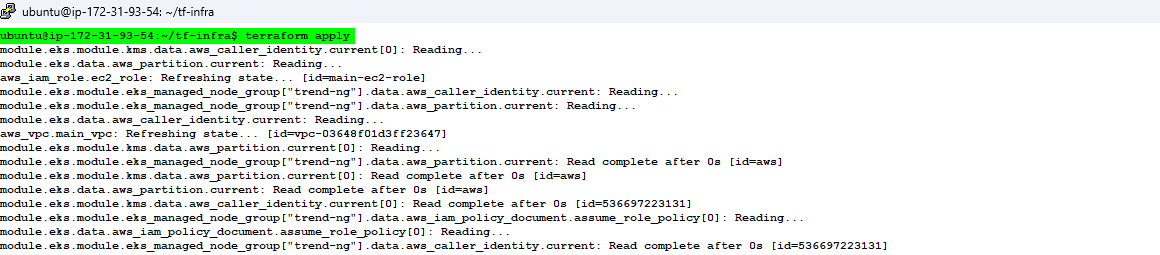
**eks-cluster.tf**

****

**Terraform – Initialization & Validation**

****

**Terraform – Apply**

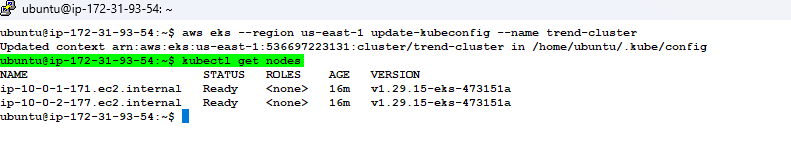
****

**Update your kubeconfig to access the cluster:**

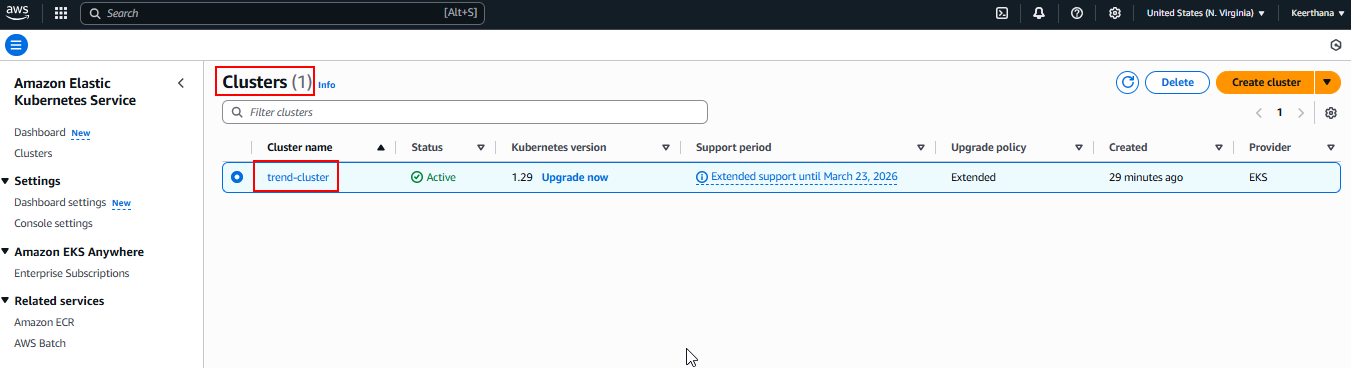
aws eks --region us-east-1 update-kubeconfig --name trend-cluster

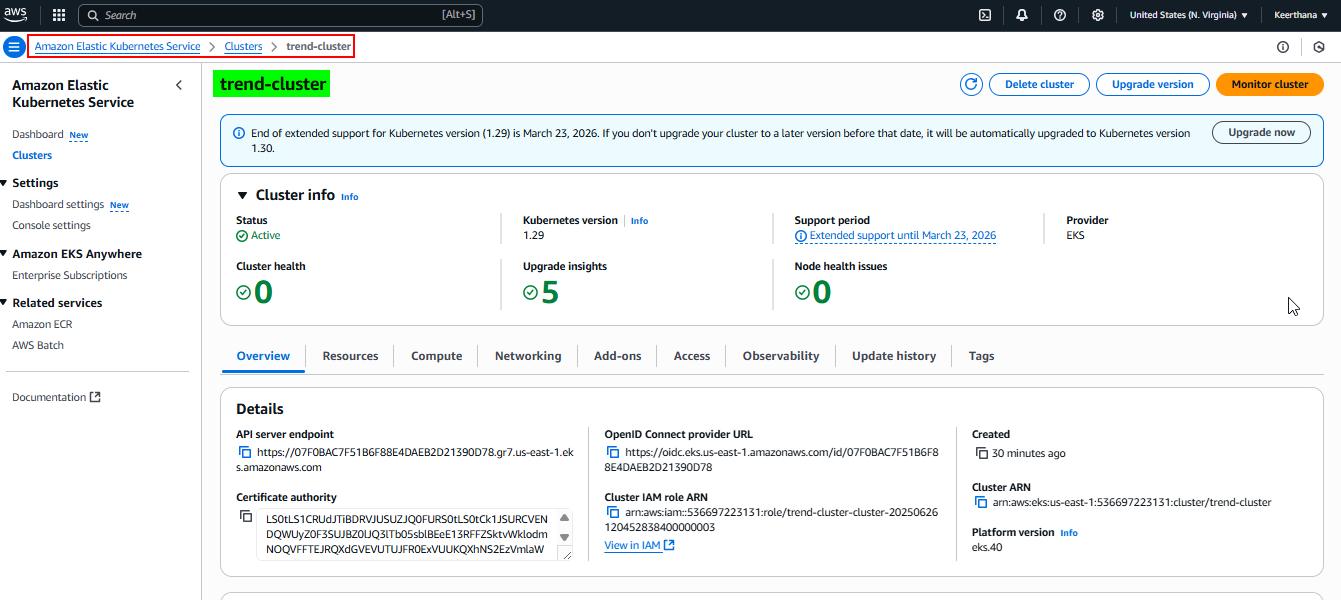
**Confirm connection to the cluster:**

kubectl get nodes

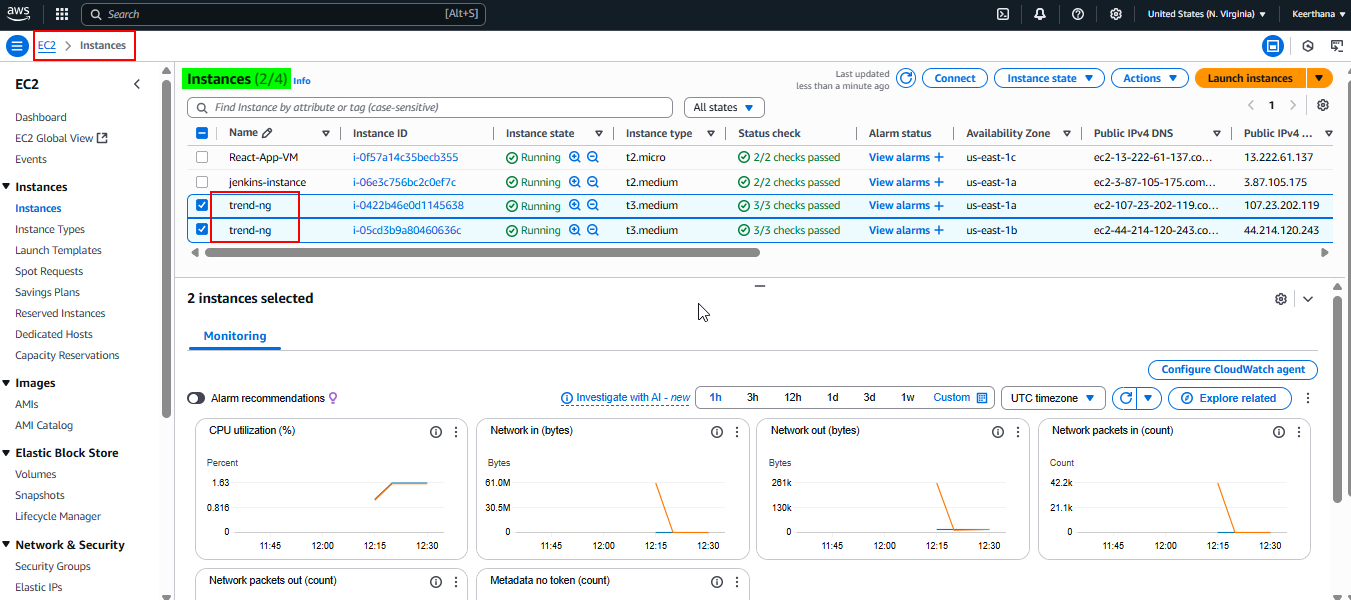


**AWS Console – EKS Cluster**

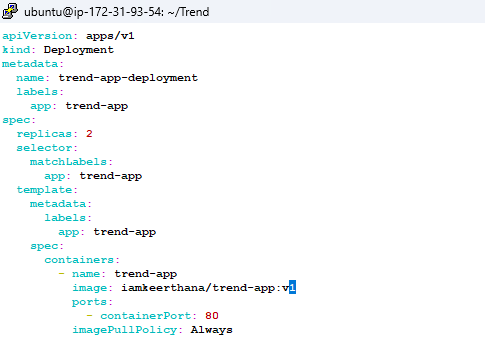




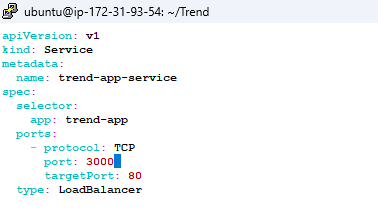
**AWS Console – Nodes**



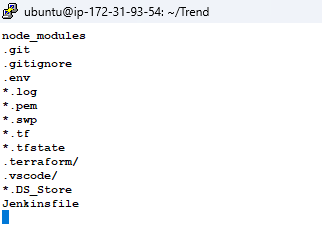
**Create deployment.yaml file**



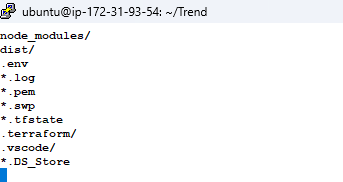
**Create service.yml file**



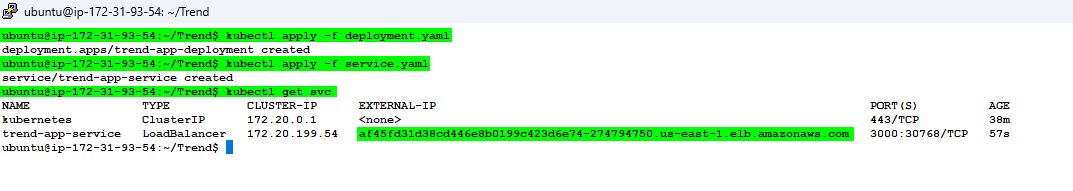
**Create .dockerignore**



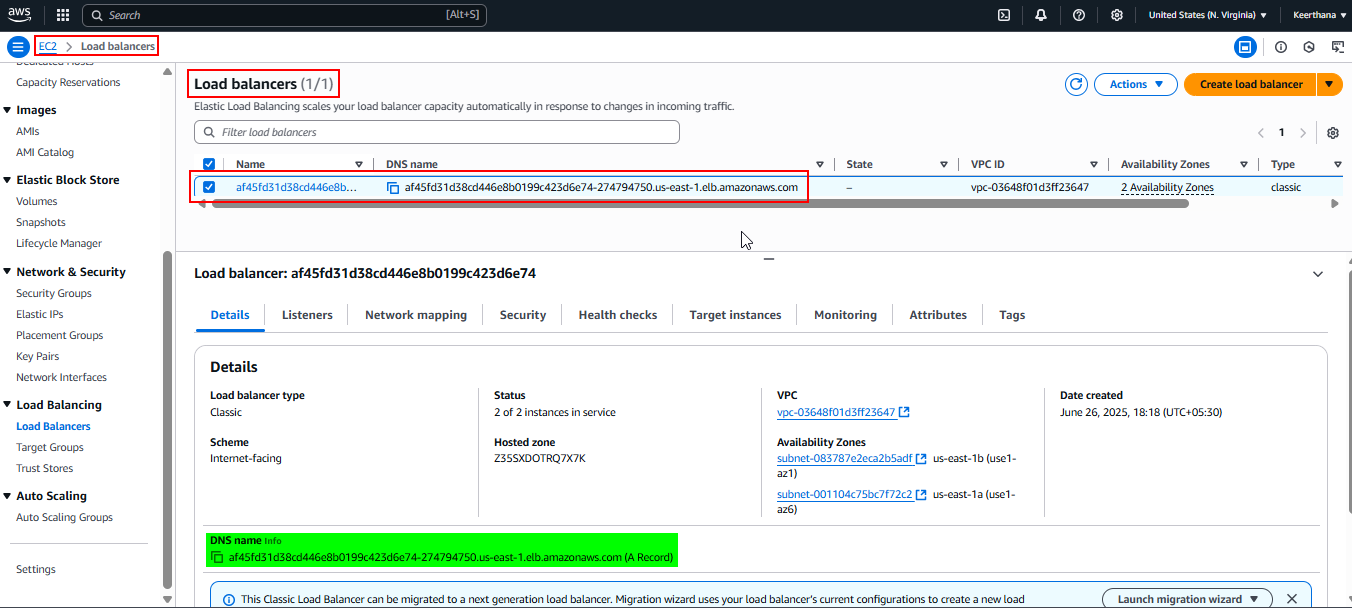
**Create .gitignore**



**Application Deployment:**

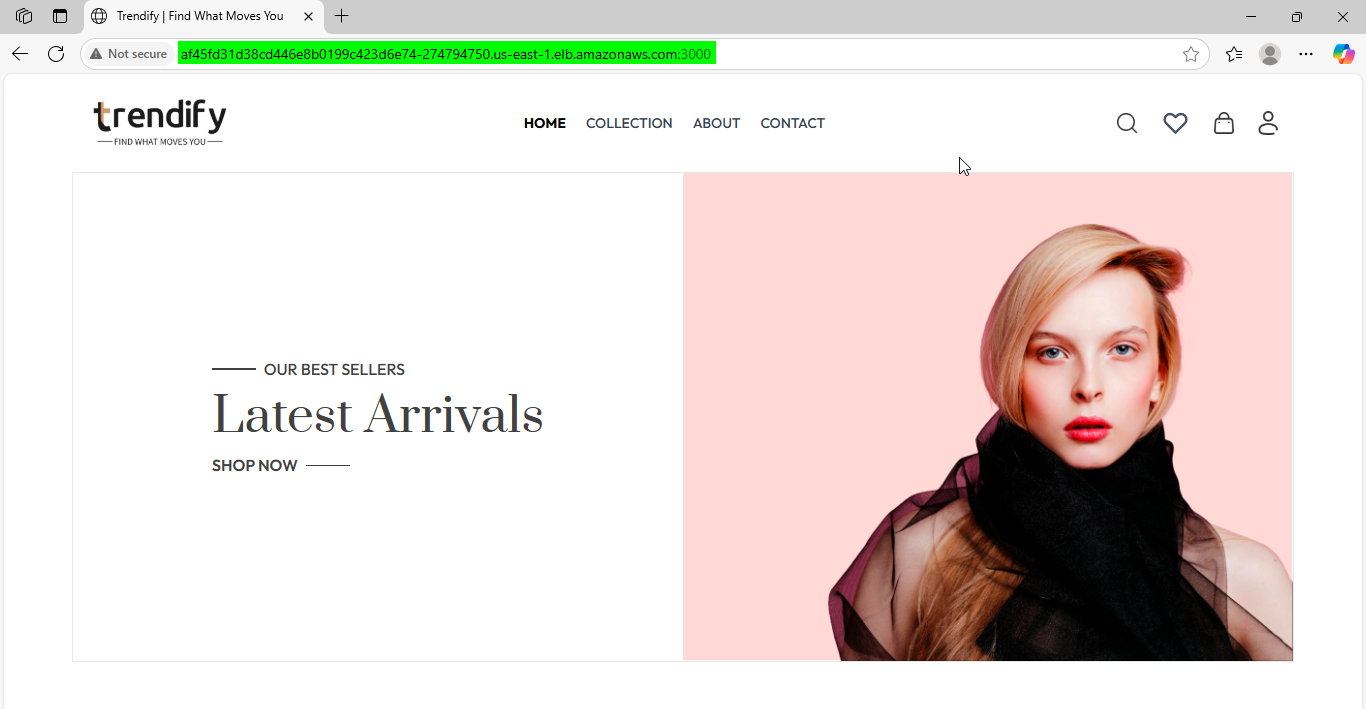
****

**Load balancer created**

****

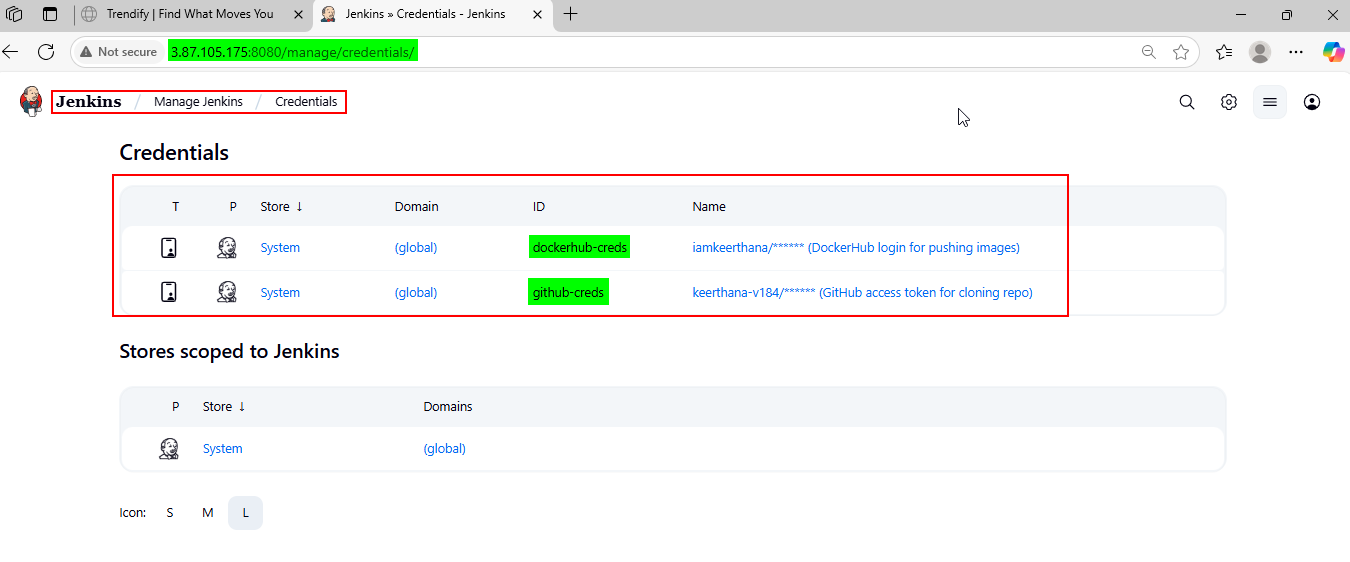
**Access Load balancer DNS in browser:**

* <http://af45fd31d38cd446e8b0199c423d6e74-274794750.us-east-1.elb.amazonaws.com:3000/>

****

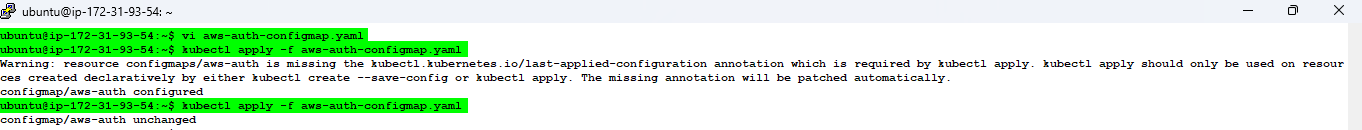
**Jenkins Automation:**

**Login into Jenkins console - Add Credentials for Docker and GitHub**

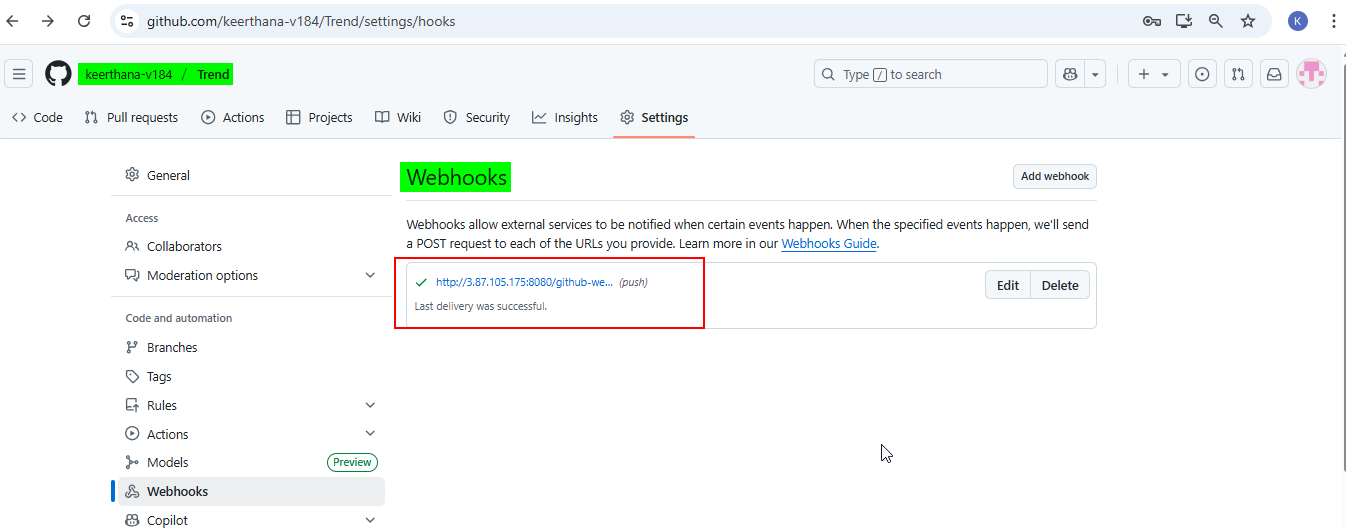
****

**Create aws-auth-configmap.yaml**

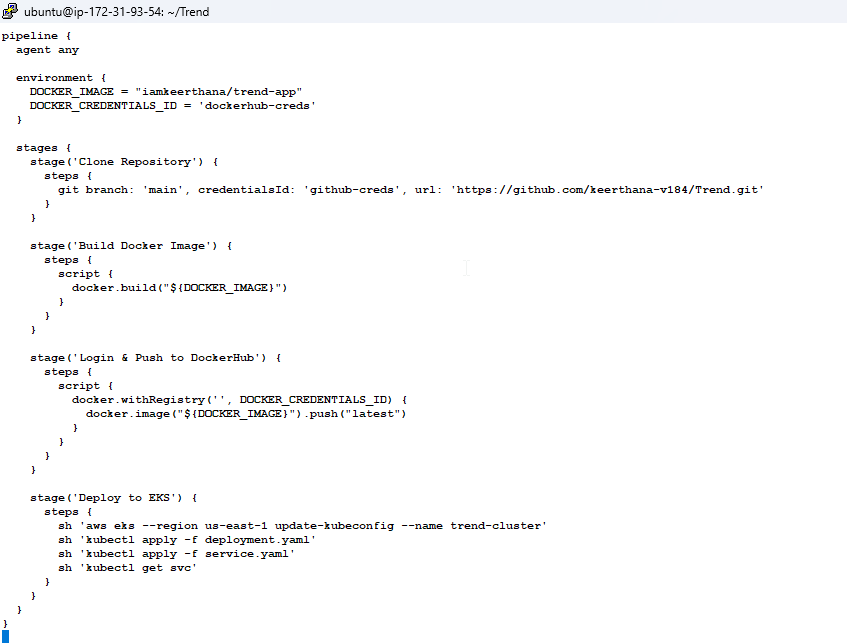
* Gives EKS nodes permission to join the cluster.

****

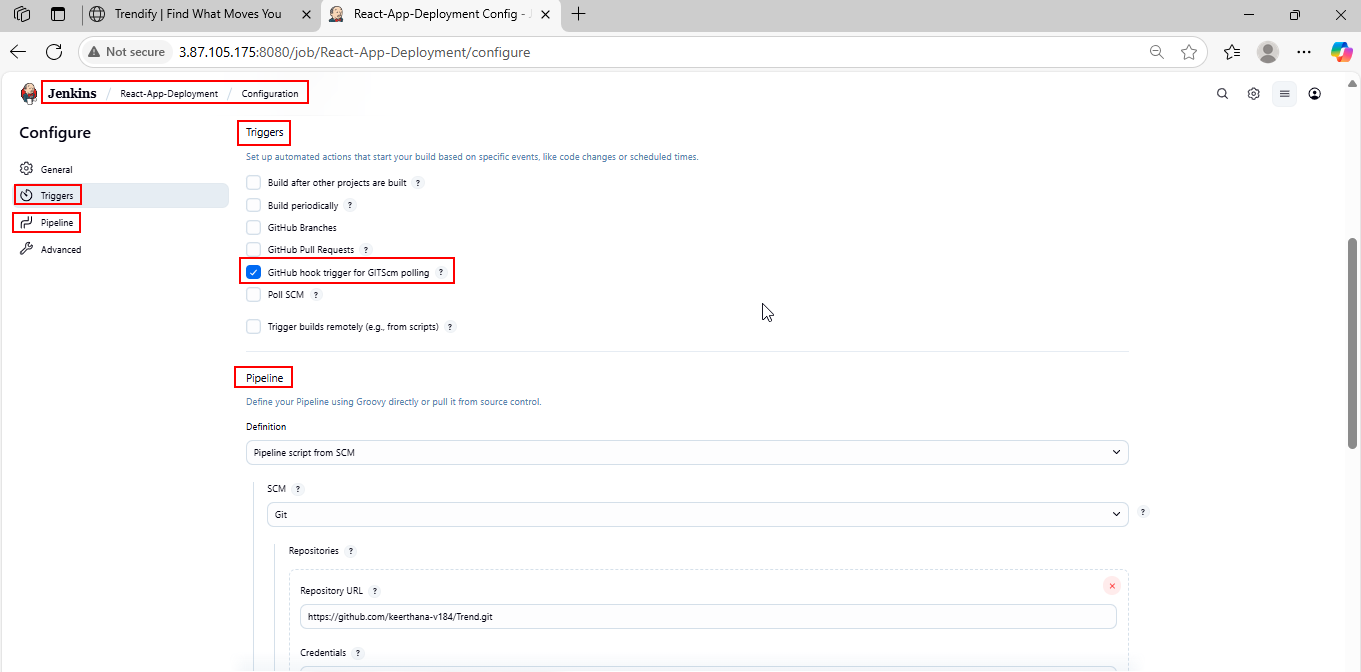
**Create webhook in GitHub:**

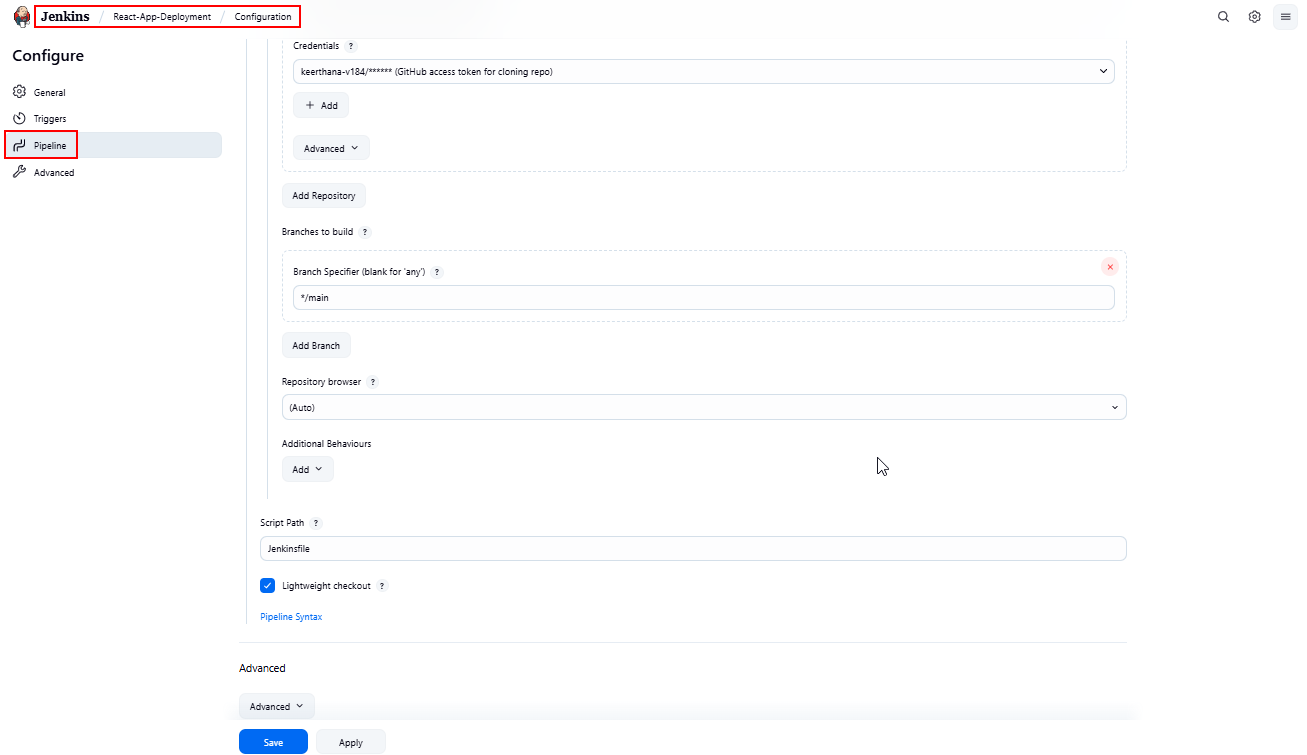
****

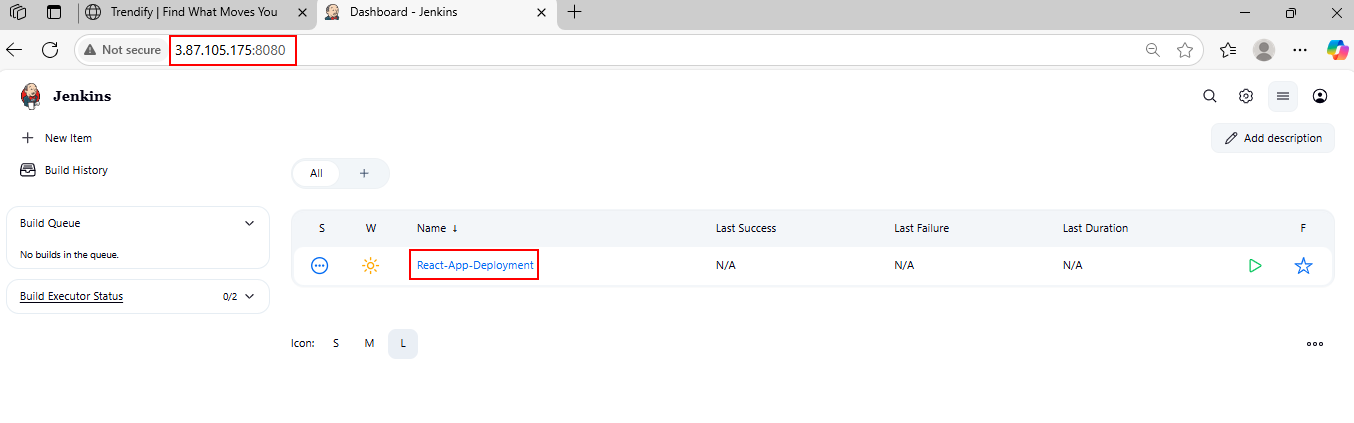
**Create Jenkins file:**

****

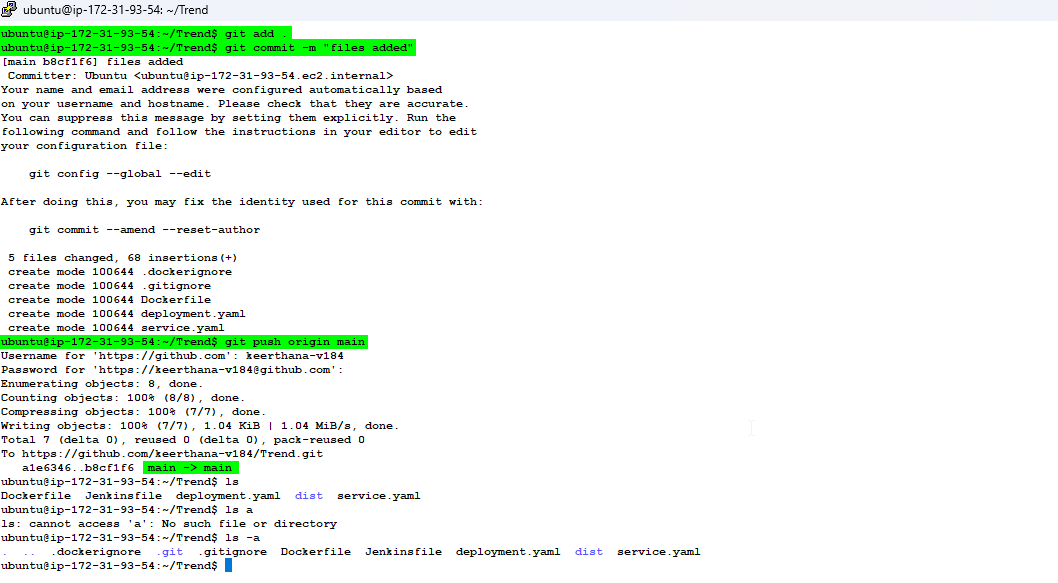
**Jenkins Pipeline Creation:**

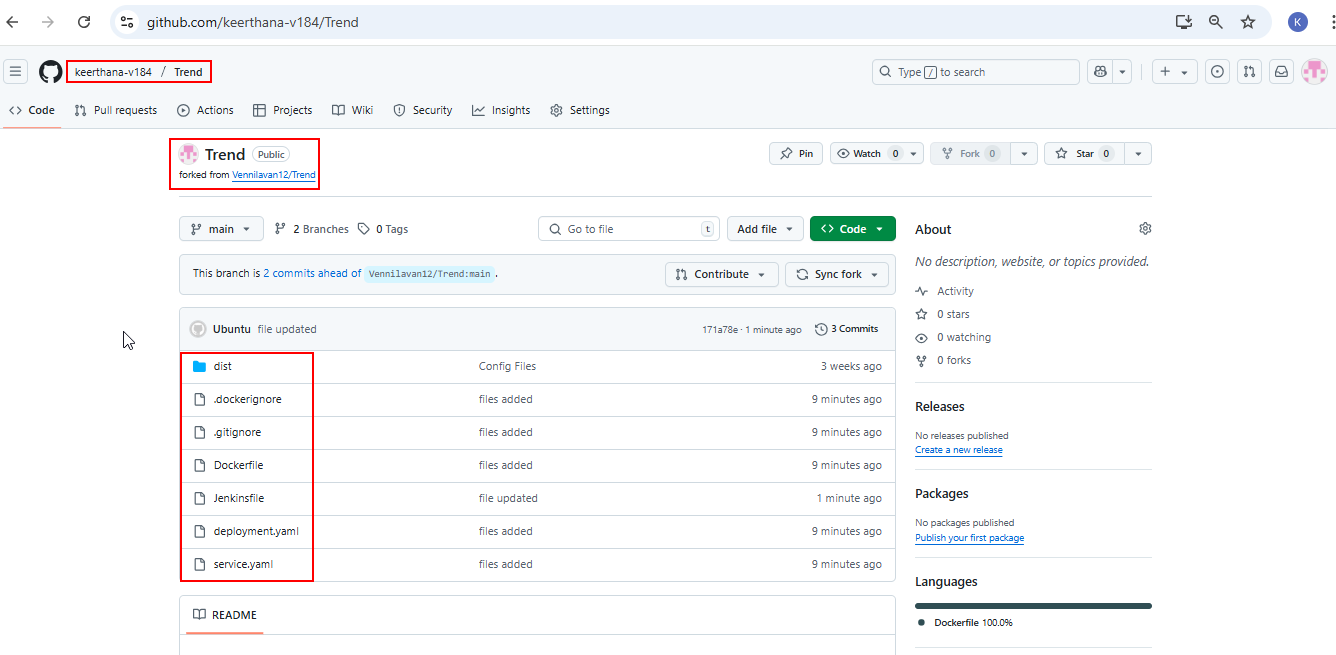
****

****

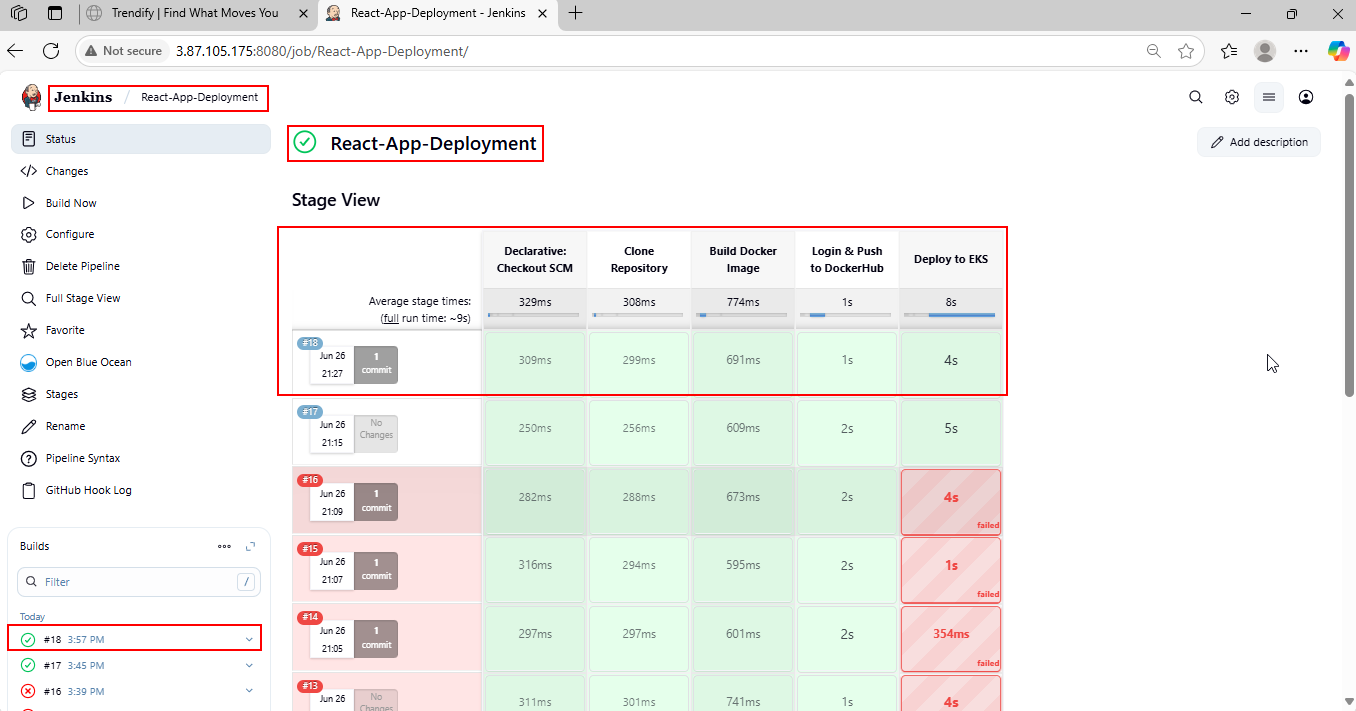
****

**Push code files to GitHub repo:**

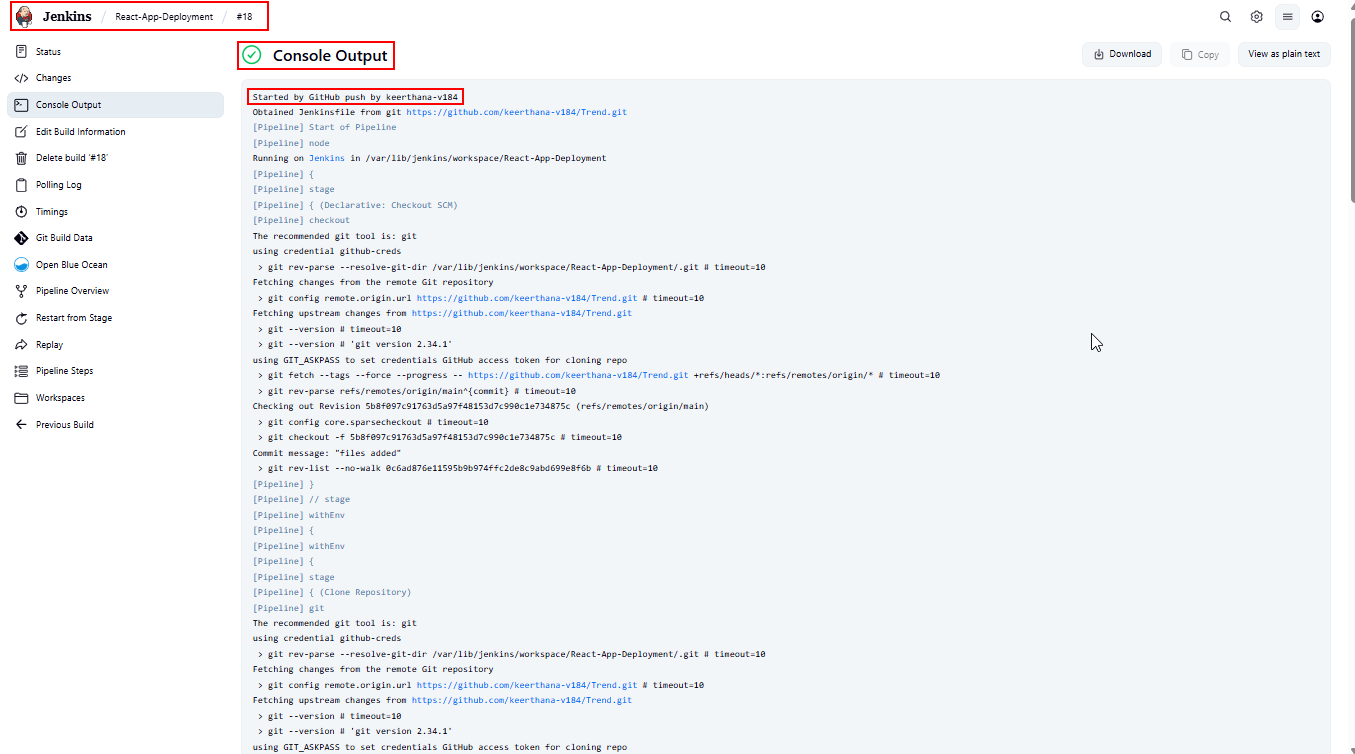
****

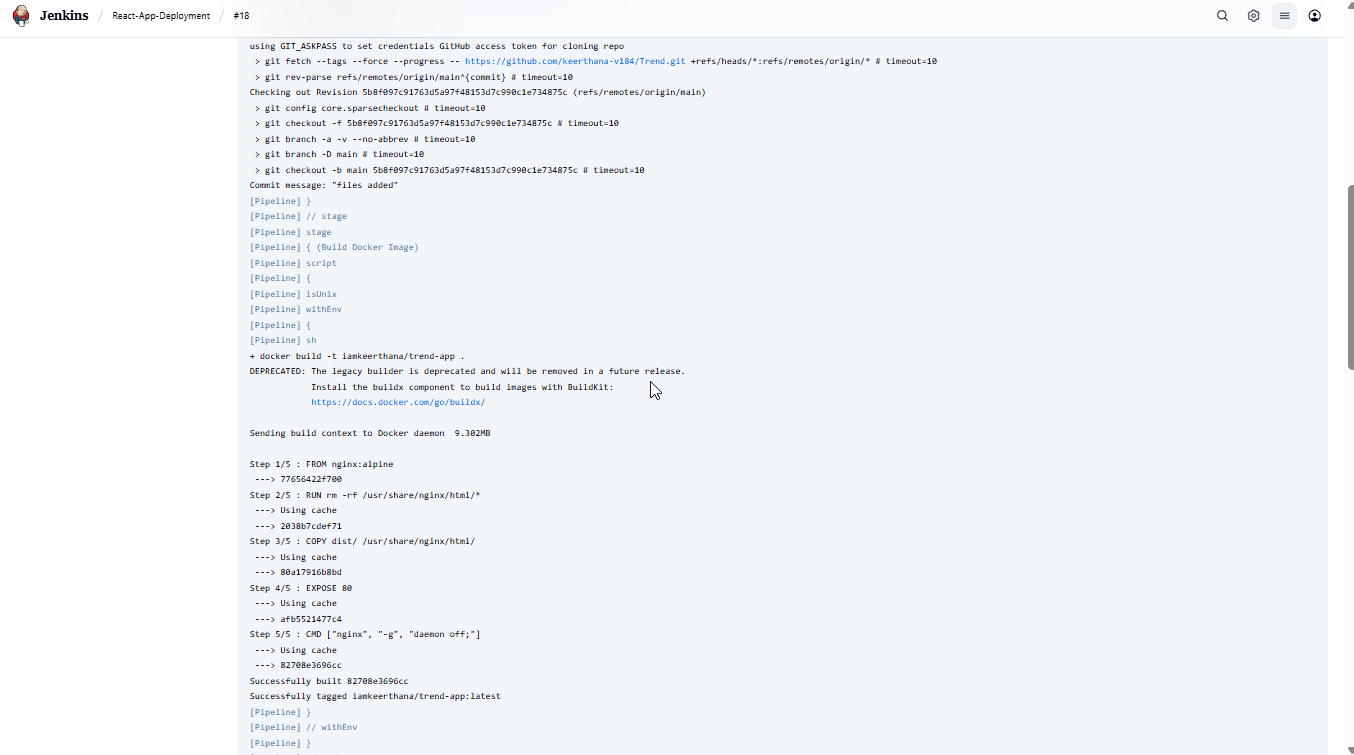
****

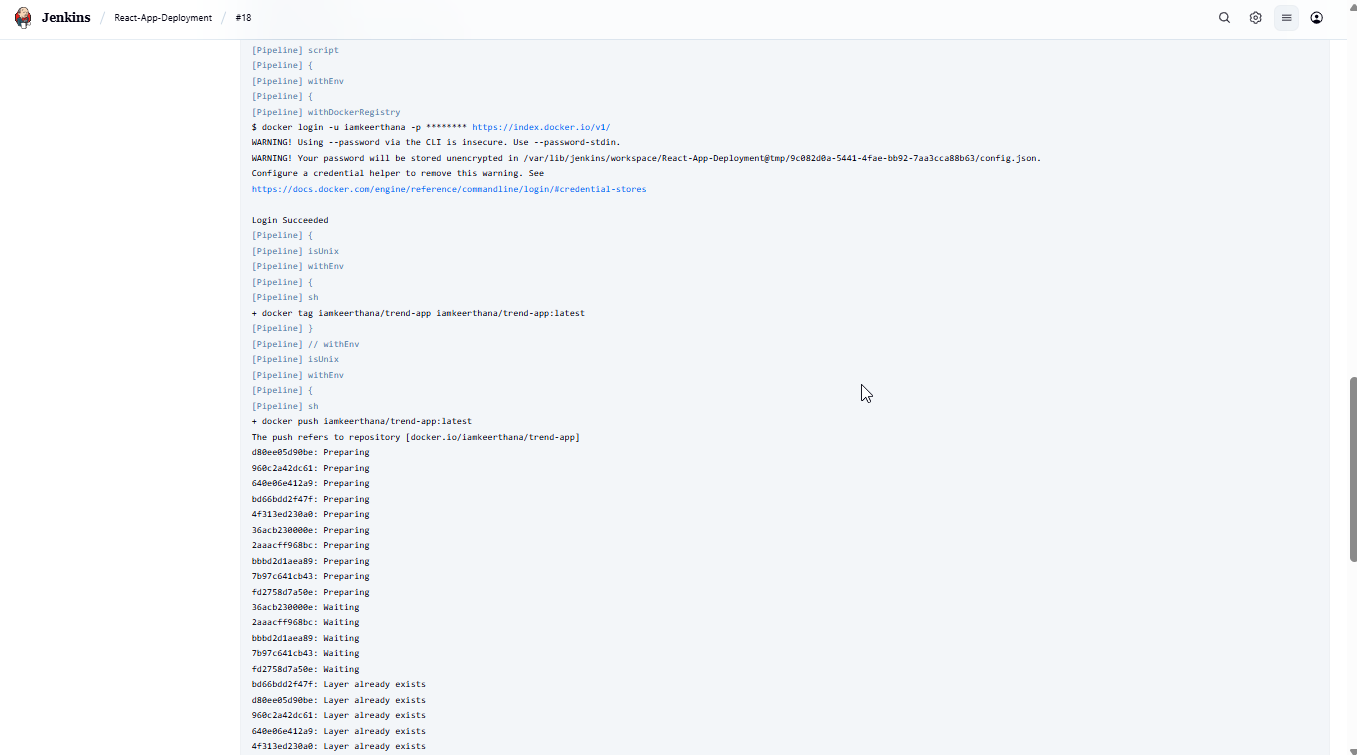
**Pipeline gets triggered and success:**

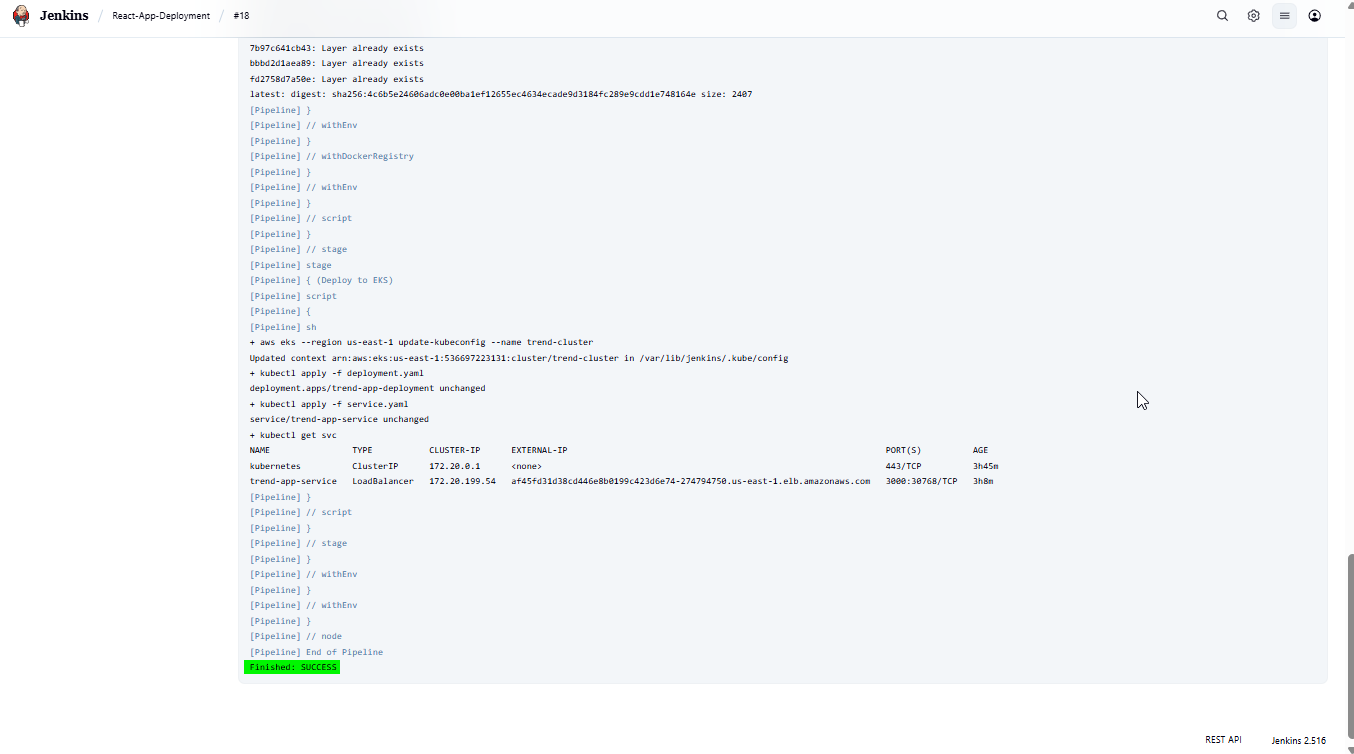
****

**Console Output:**

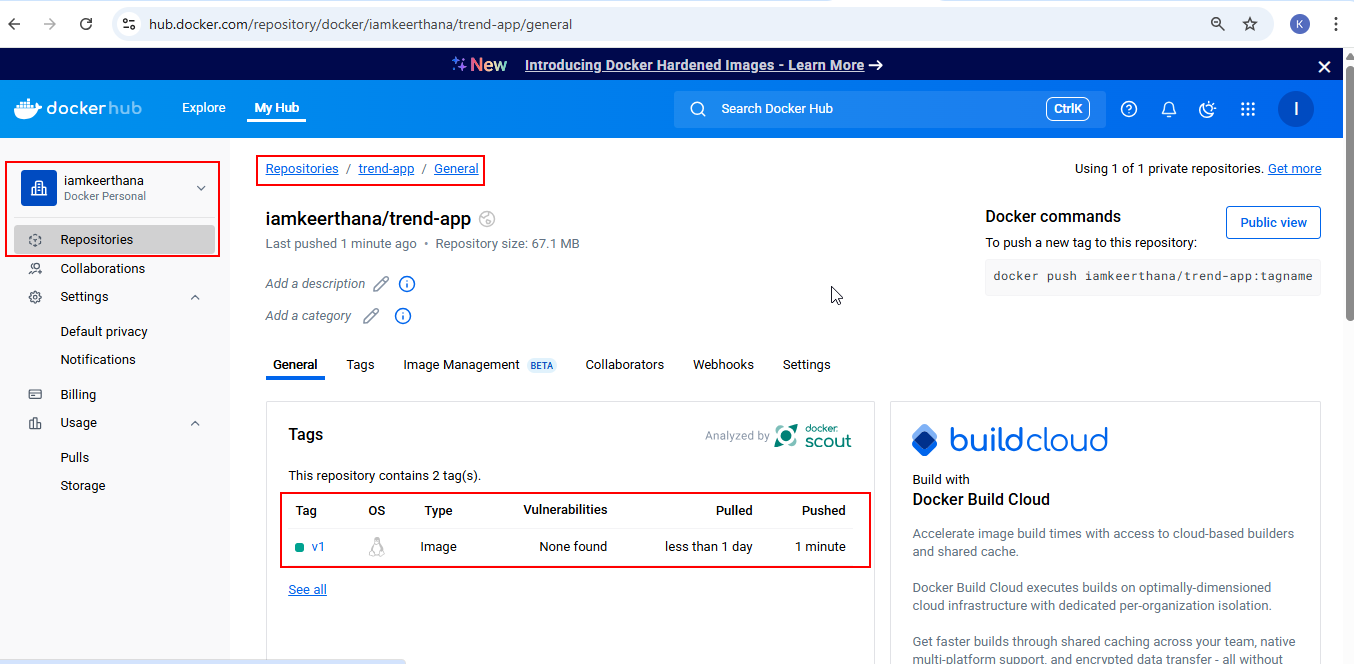
****

****

****

****

**Image Pushed to Dockerhub:**

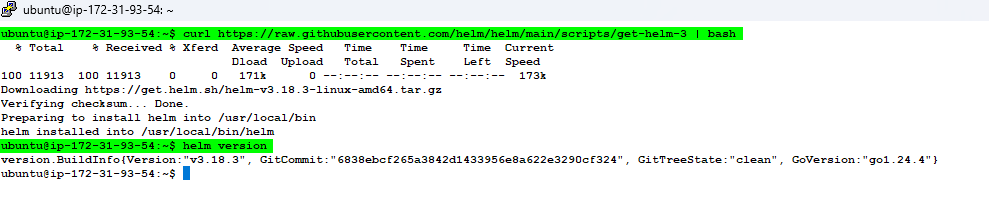
****

**Monitoring Setup on EKS:**

**Install Helm on EC2**

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

helm version

****

**Add Prometheus & Grafana Helm Repositories**

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

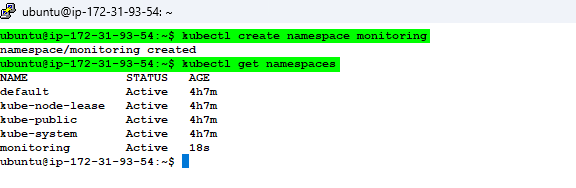
helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

****

**Create a Namespace for Monitoring**

kubectl create namespace monitoring

****

**Install Prometheus & Grafana using Helm**

helm install prometheus prometheus-community/kube-prometheus-stack \

--namespace monitoring \

--create-namespace \

--set prometheus.service.type=NodePort \

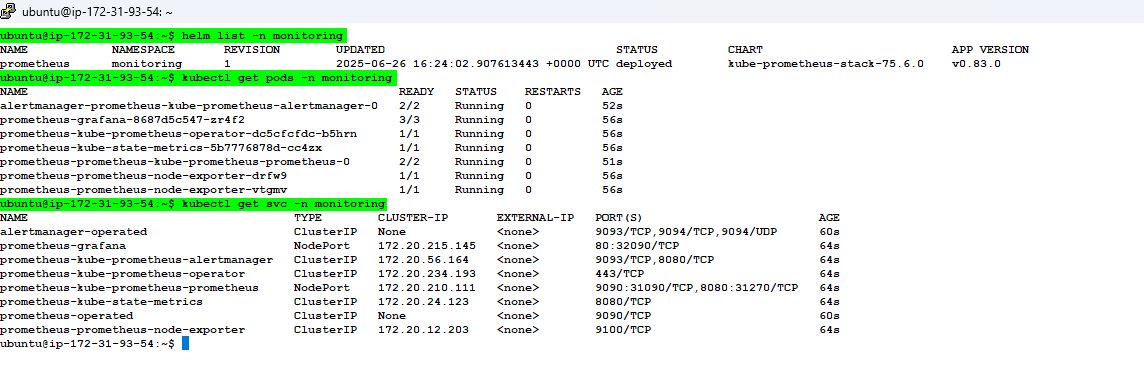
--set prometheus.service.nodePort=31090 \

--set grafana.service.type=NodePort \

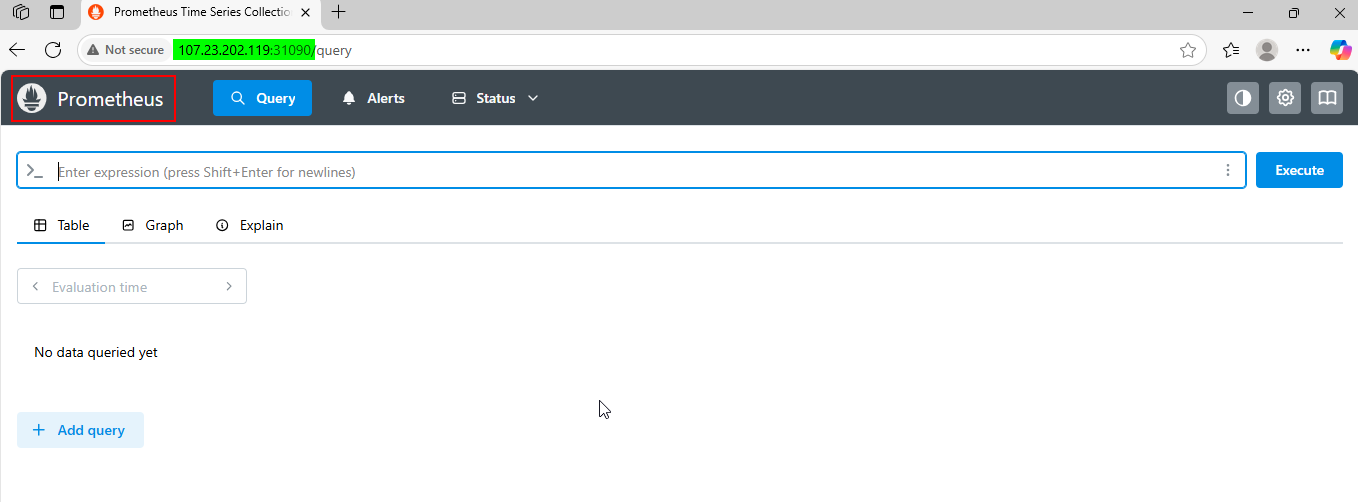
--set grafana.service.nodePort=32090 \

--set grafana.adminPassword='admin123'

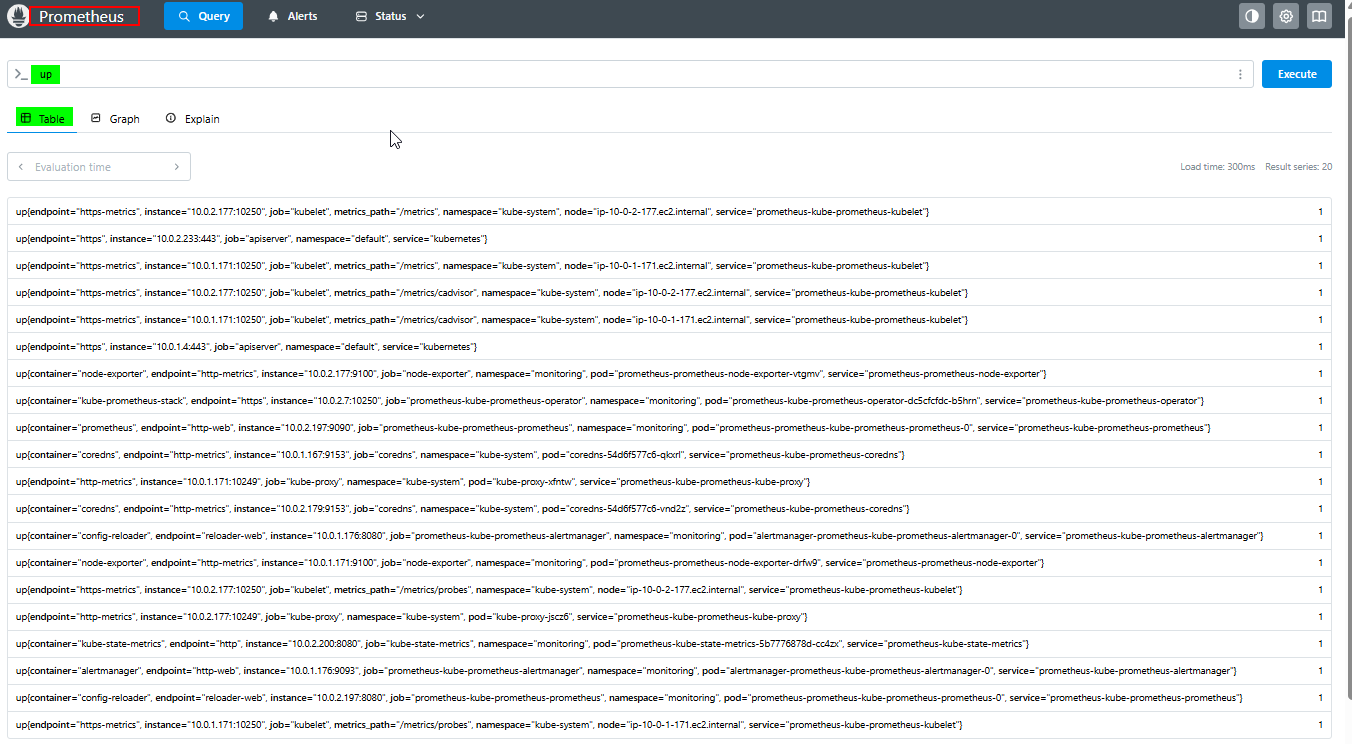
**Verify Prometheus & Grafana Installation**

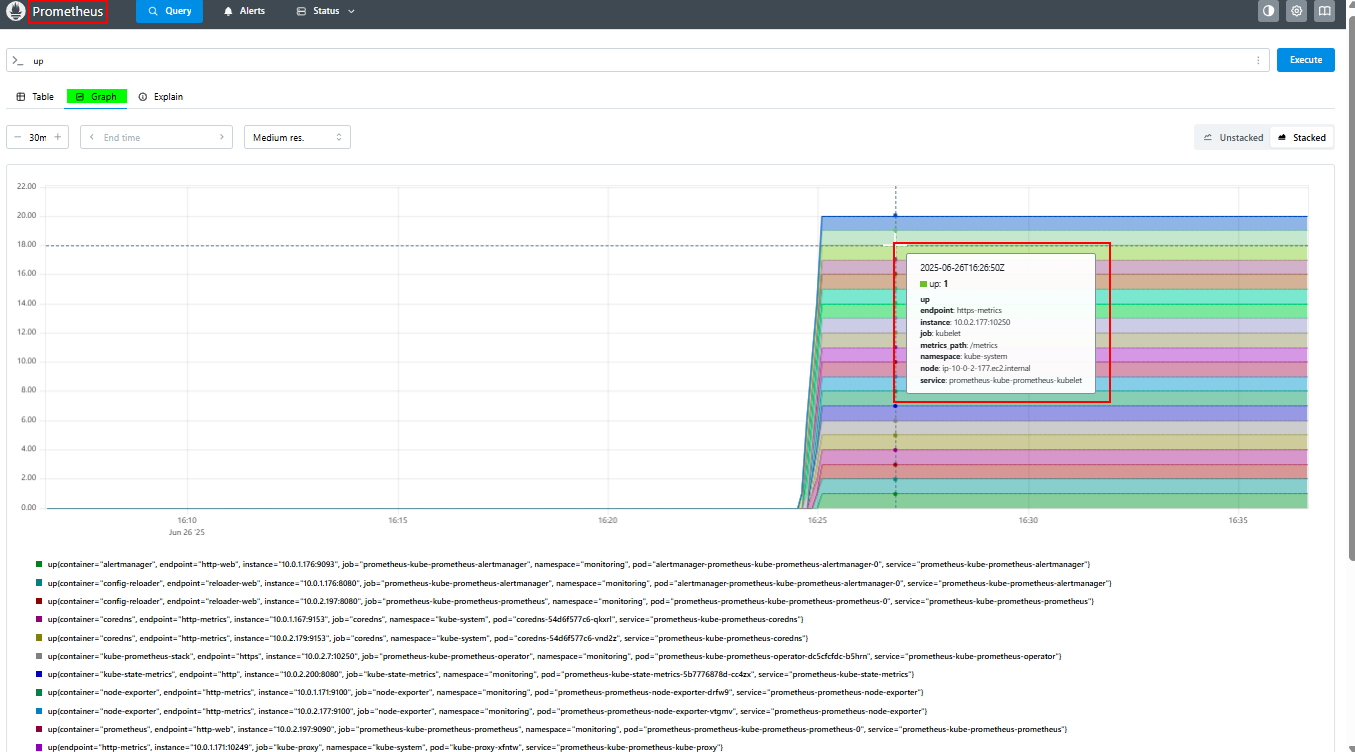
****

**Prometheus Home page:**

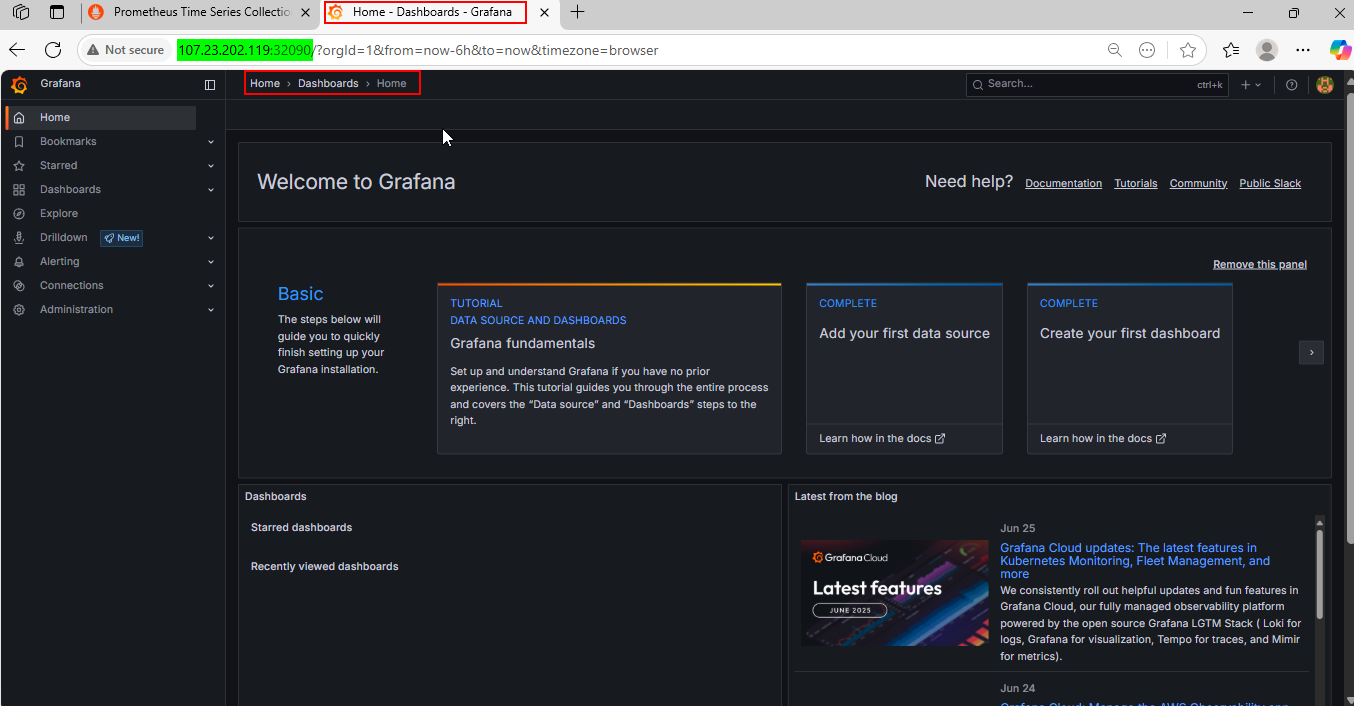
****

**Prometheus up status:**

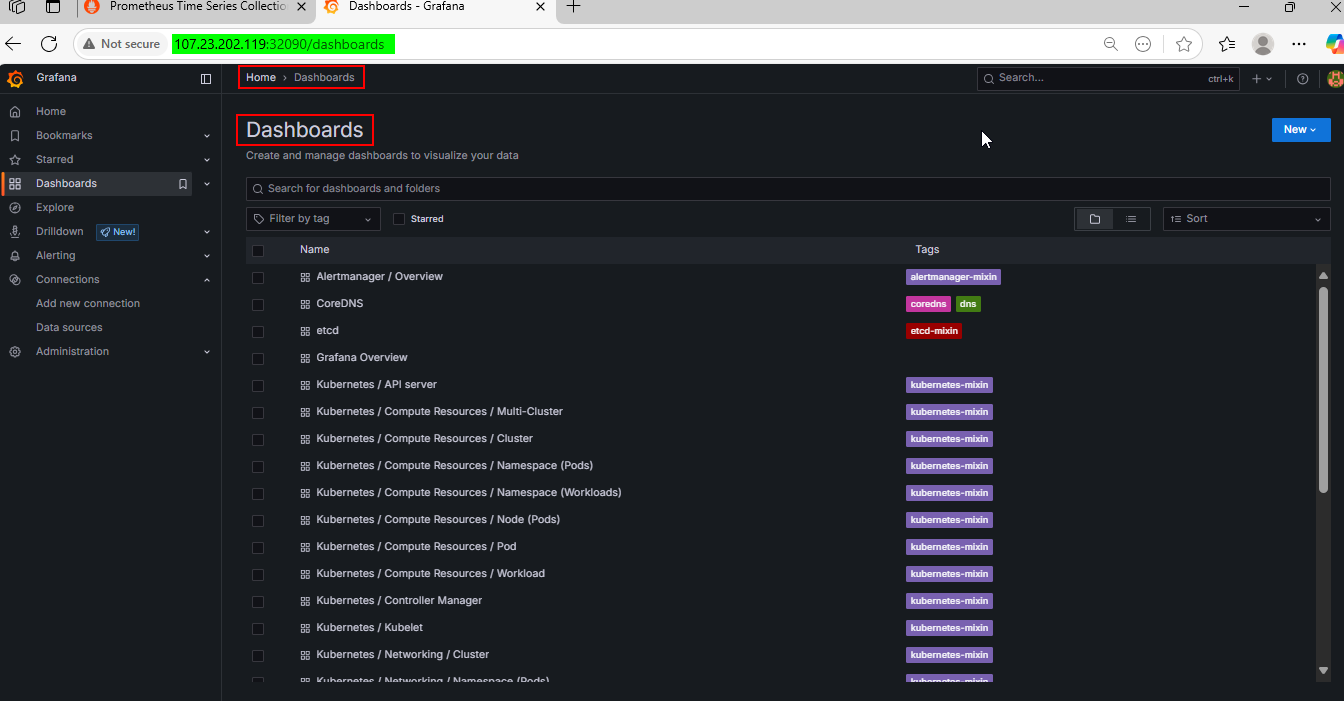
**  
  
Prometheus up status - Graph:**

****

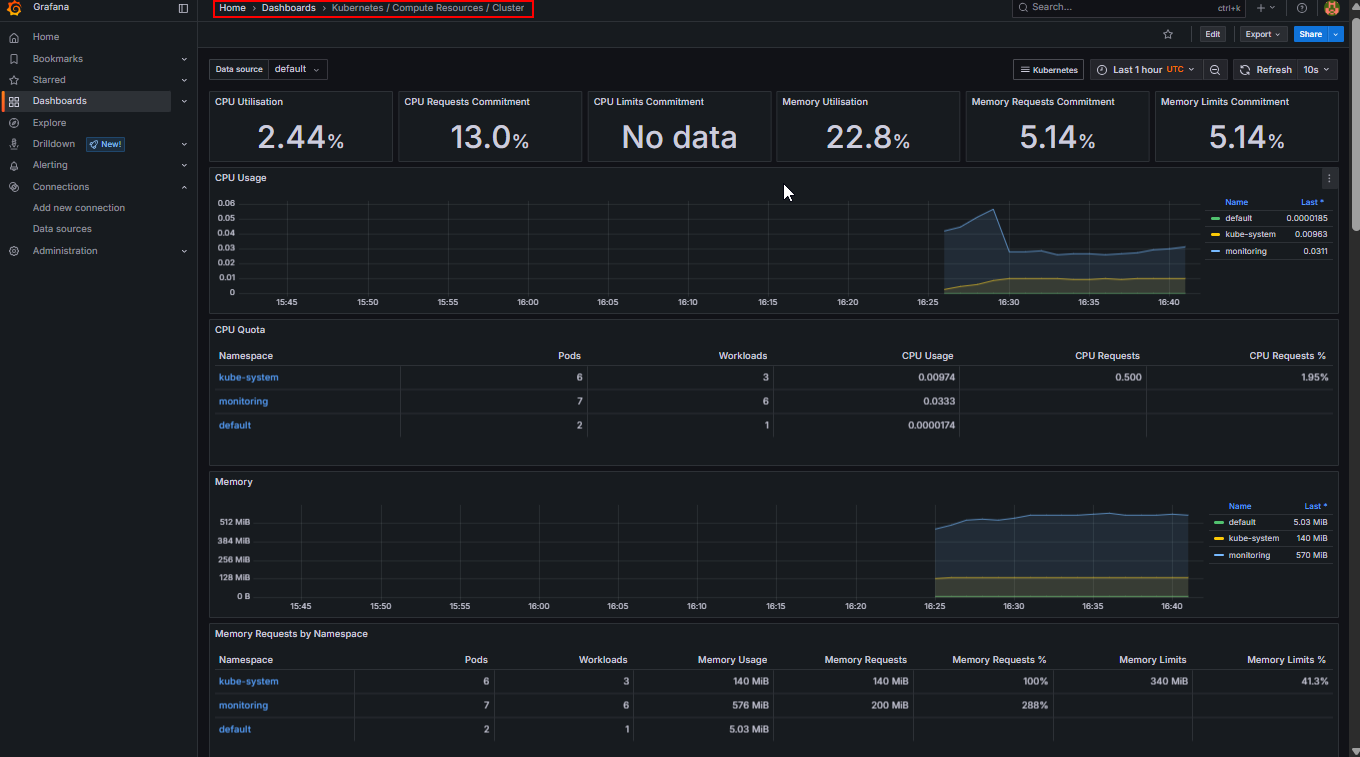
**Grafana:**

****

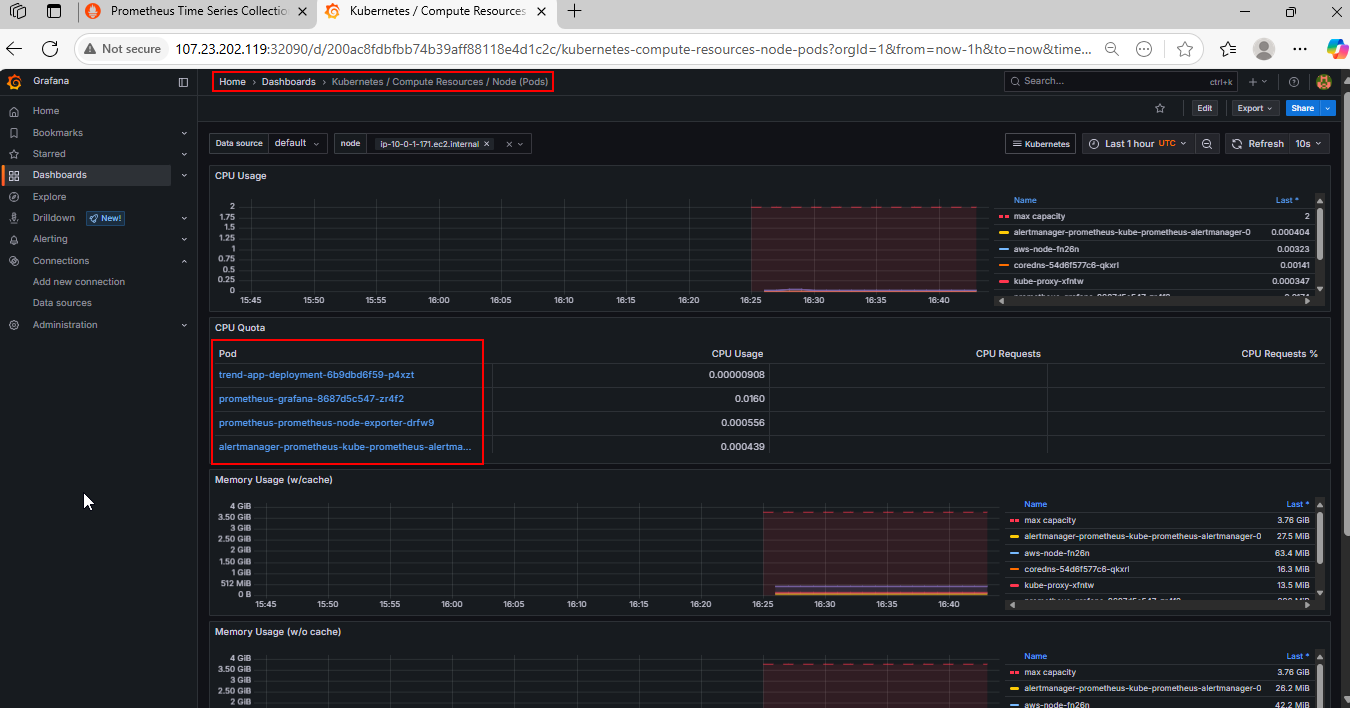
**Grafana Dashboard:**

****

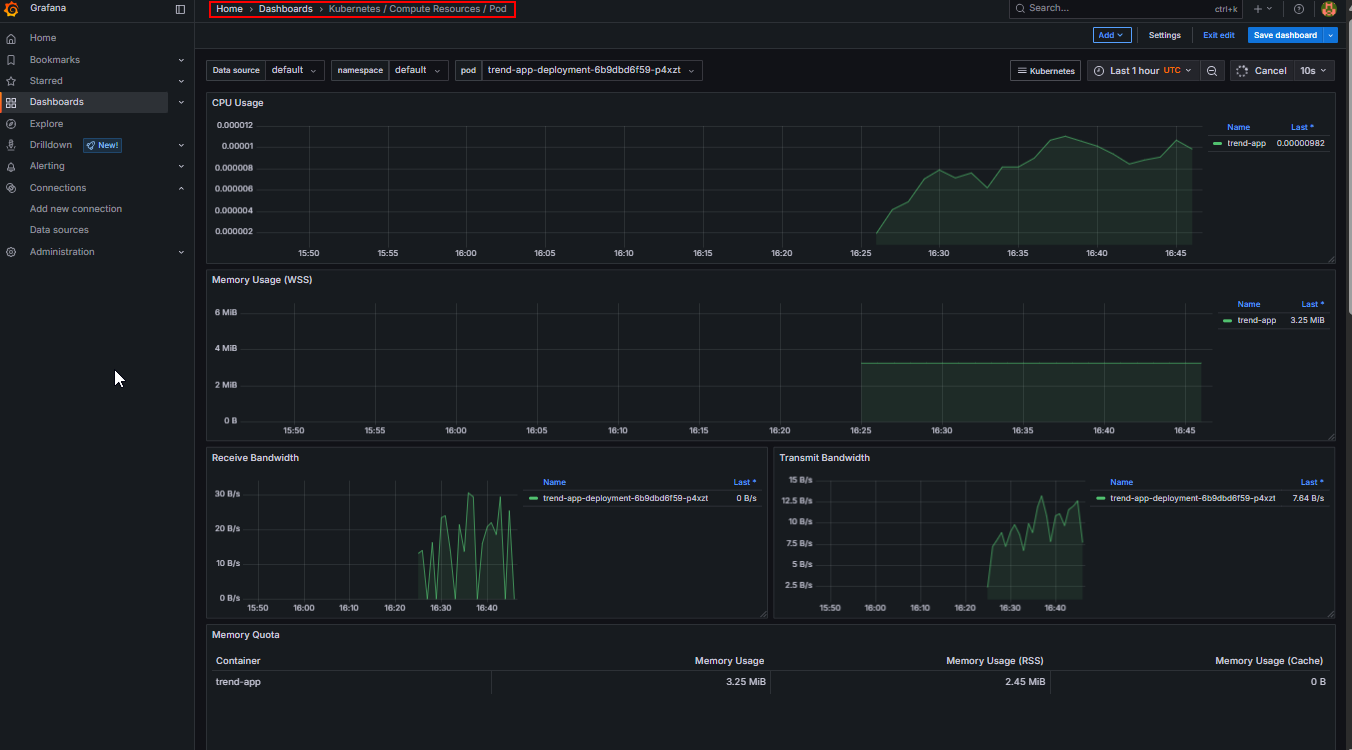
**Cluster Dashboard:**

****

**Nodes Dashboard:**

****

**Pods Dashboard:**

****