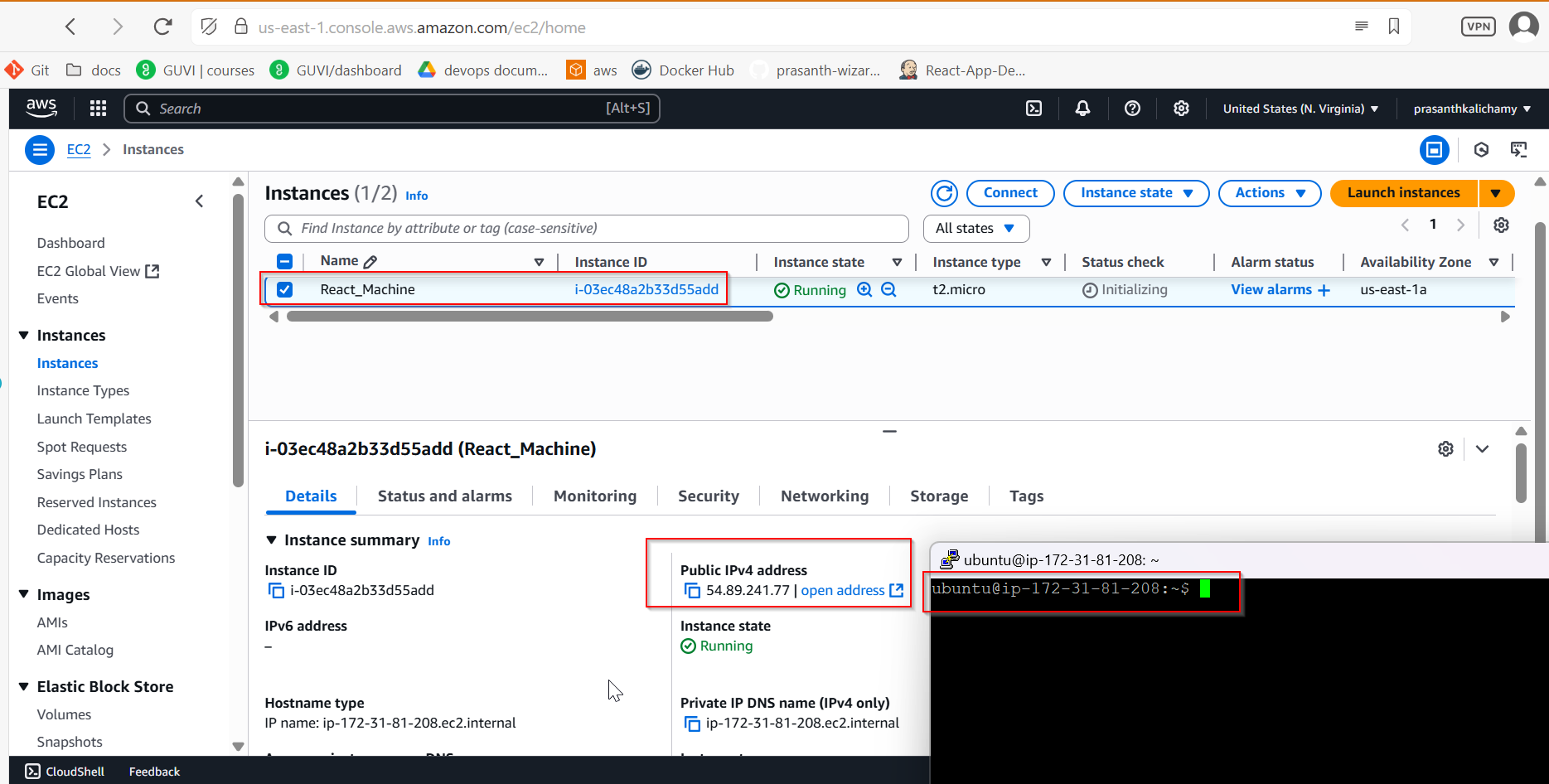
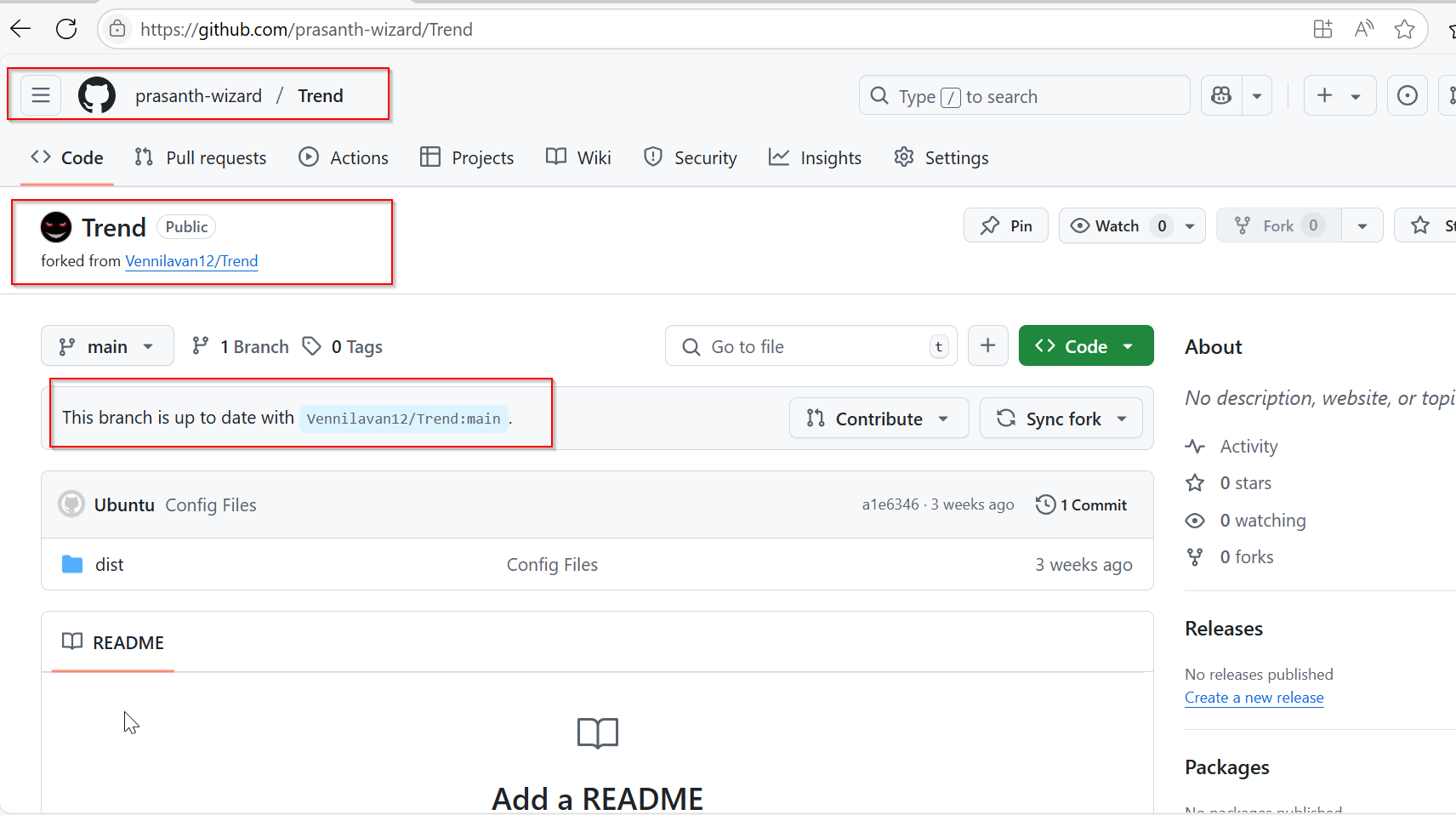
React-Application Deployment

**EC2 Instance has been launched and connected with putty:**



**Project repo forked from Vennilavan12/Trend:**

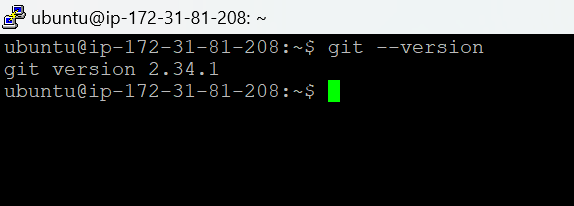


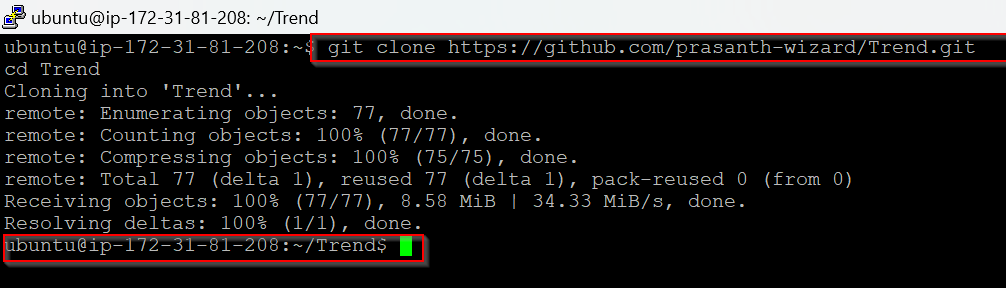
**Install Git on Instance:**

sudo apt update

sudo apt install git -y

git –version

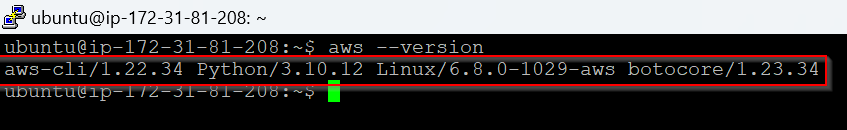




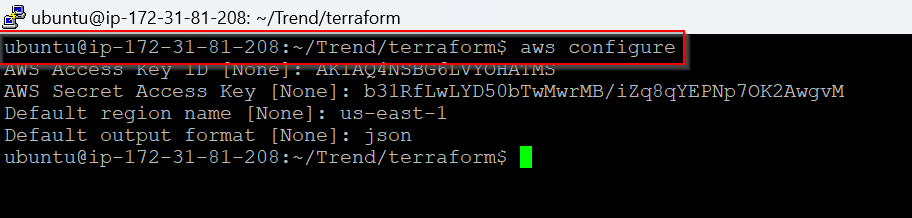
**AWSCLI:**

sudo apt install awscli -y

aws –version



* **Create new IAM User** (name it: terraform-user)
* Programmatic access: enable
* Attach **AdministratorAccess** policy
* aws configure



**Install Docker:**

sudo apt update

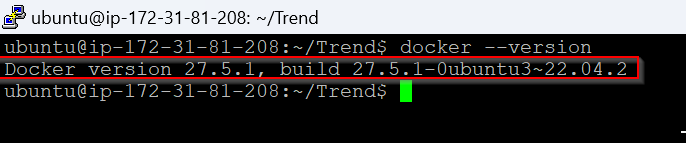
sudo apt install docker.io -y

sudo systemctl start docker

sudo systemctl enable docker

sudo usermod -aG docker $USER

docker –version



**Dockerize the Application:**

**Dockerfile:**

FROM nginx:alpine

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

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

EXPOSE 3000

RUN sed -i 's/80;/3000;/' /etc/nginx/conf.d/default.conf

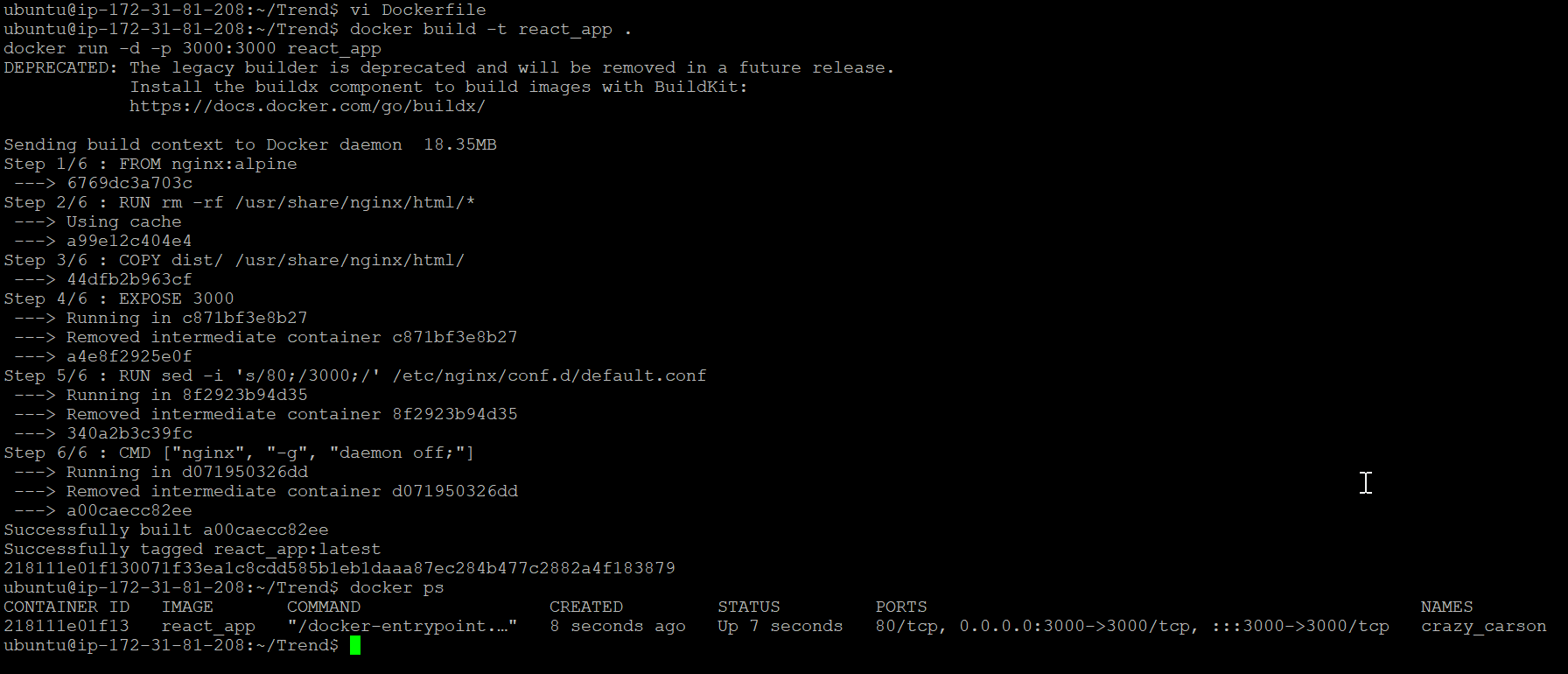
CMD ["nginx", "-g", "daemon off;"]

sudo usermod -aG docker $USER

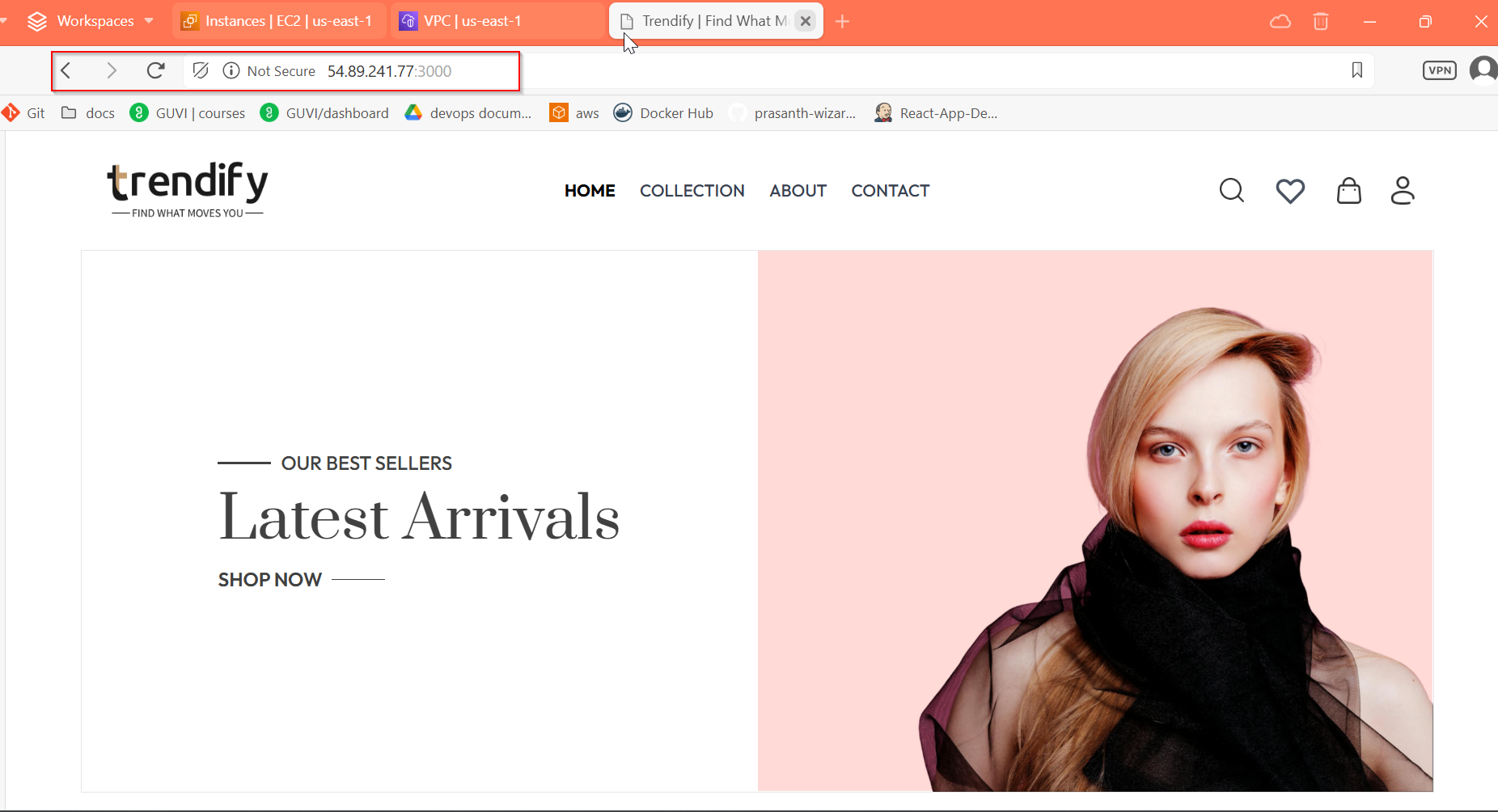
**Build and Test Docker Image:**

docker build -t React\_App .

docker run -d -p 3000:3000 React\_App



**Application Running on ip port :3000**

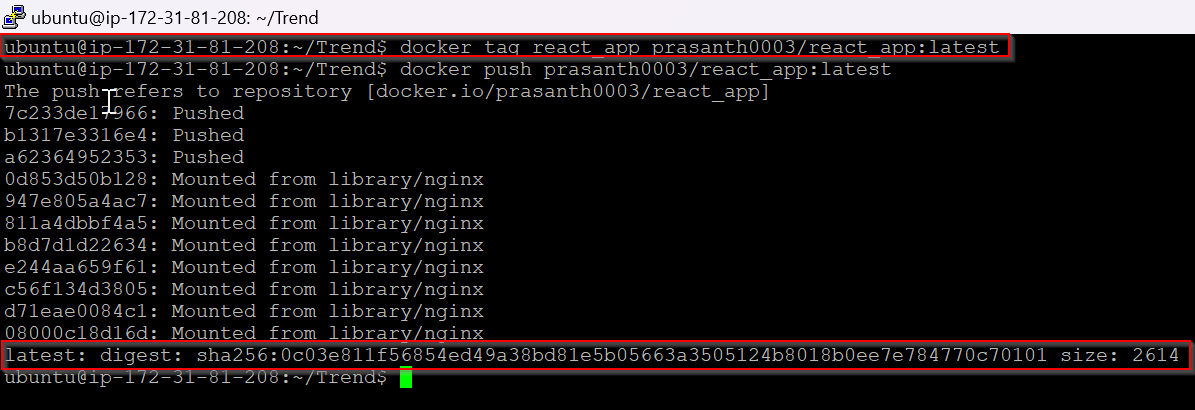


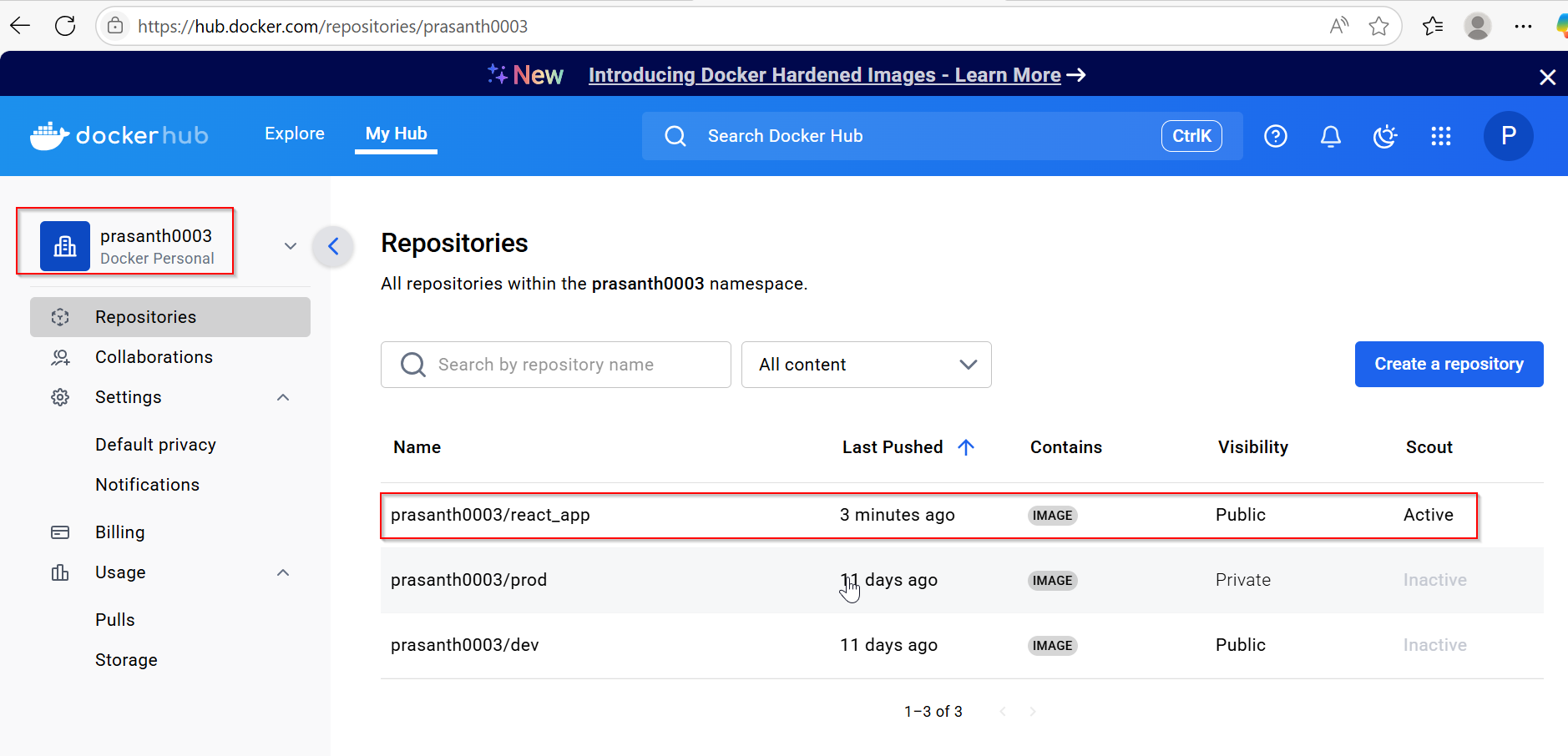
**Create DockerHub Repo:**

docker tag react\_app prasanth0003/react\_app:latest

docker login

docker push prasanth0003/react\_app:latest





**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] \

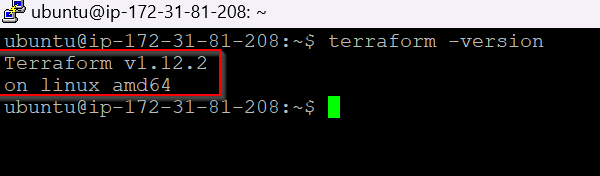
https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | \

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

sudo apt update -y

sudo apt install terraform -y

terraform -version



**Creating infrastructure for Jenkins:**

**Creating main.tf**

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.45.0"

}

}

required\_version = ">= 1.5.0"

}

provider "aws" {

region = var.aws\_region

}

# VPC

resource "aws\_vpc" "main" {

cidr\_block = "10.0.0.0/16"

enable\_dns\_support = true

enable\_dns\_hostnames = true

tags = {

Name = "trend-vpc"

}

}

# Internet Gateway

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

vpc\_id = aws\_vpc.main.id

tags = {

Name = "trend-igw"

}

}

# Public Subnets

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

vpc\_id = aws\_vpc.main.id

cidr\_block = "10.0.1.0/24"

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

map\_public\_ip\_on\_launch = true

tags = {

Name = "trend-public-subnet-1"

}

}

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

vpc\_id = aws\_vpc.main.id

cidr\_block = "10.0.2.0/24"

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

map\_public\_ip\_on\_launch = true

tags = {

Name = "trend-public-subnet-2"

}

}

# Route Table

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

vpc\_id = aws\_vpc.main.id

route {

cidr\_block = "0.0.0.0/0"

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

}

tags = {

Name = "trend-public-rt"

}

}

# Route Table Association

resource "aws\_route\_table\_association" "a1" {

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

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

}

resource "aws\_route\_table\_association" "a2" {

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

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

}

# Security Group

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

name = "jenkins-sg"

description = "Allow ports for Jenkins and app"

vpc\_id = aws\_vpc.main.id

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 80

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"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "jenkins-sg"

}

}

# IAM Role for EC2

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

name = "trend-ec2-role"

assume\_role\_policy = jsonencode({

Version = "2012-10-17"

Statement = [{

Action = "sts:AssumeRole"

Effect = "Allow"

Principal = {

Service = "ec2.amazonaws.com"

}

}]

})

}

# Attach policies to the IAM Role

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

}

# IAM Instance Profile

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

name = "trend-ec2-profile"

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

}

# EC2 Instance

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

ami = "ami-053b0d53c279acc90" # Ubuntu 22.04 in us-east-1

instance\_type = "t2.medium"

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

key\_name = "keypair"

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 openjdk-17-jdk -y

sudo apt install docker.io -y

sudo systemctl enable docker

sudo usermod -aG docker ubuntu

sudo apt install unzip -y

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.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-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update -y

sudo apt install jenkins -y

sudo systemctl start jenkins

sudo systemctl enable jenkins

sudo apt install -y curl

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 = "trend-jenkins-ec2"

}

}

**Outputs.tf**

output "ec2\_instance\_public\_ip" {

description = "Public IP of the Jenkins EC2 instance"

value = aws\_instance.jenkins\_ec2.public\_ip

}

output "vpc\_id" {

description = "ID of the created VPC"

value = aws\_vpc.main.id

}

output "iam\_instance\_profile\_name" {

description = "Name of the IAM instance profile"

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

}

**variables.tf**

variable "aws\_region" {

default = "us-east-1"

}

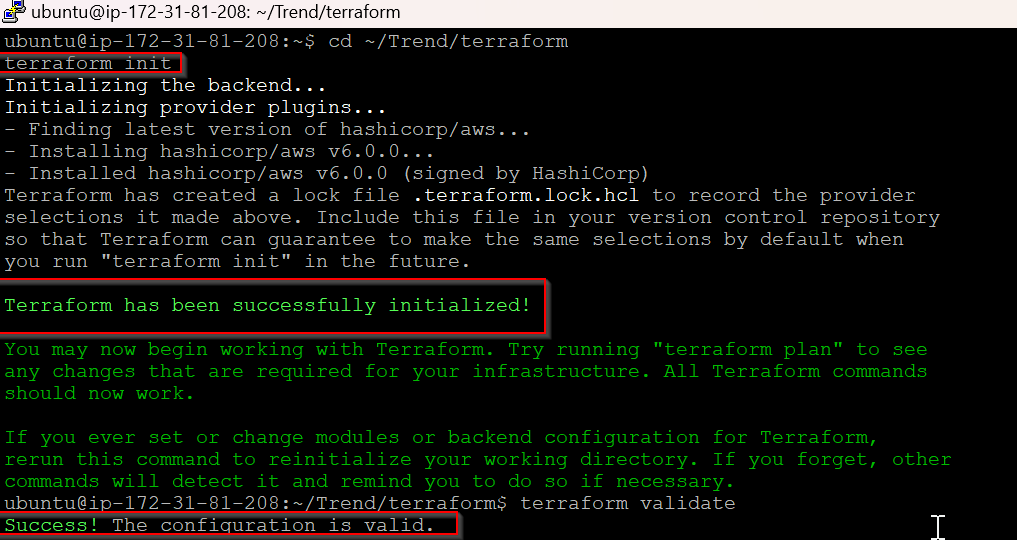
**Terraform Execution:**

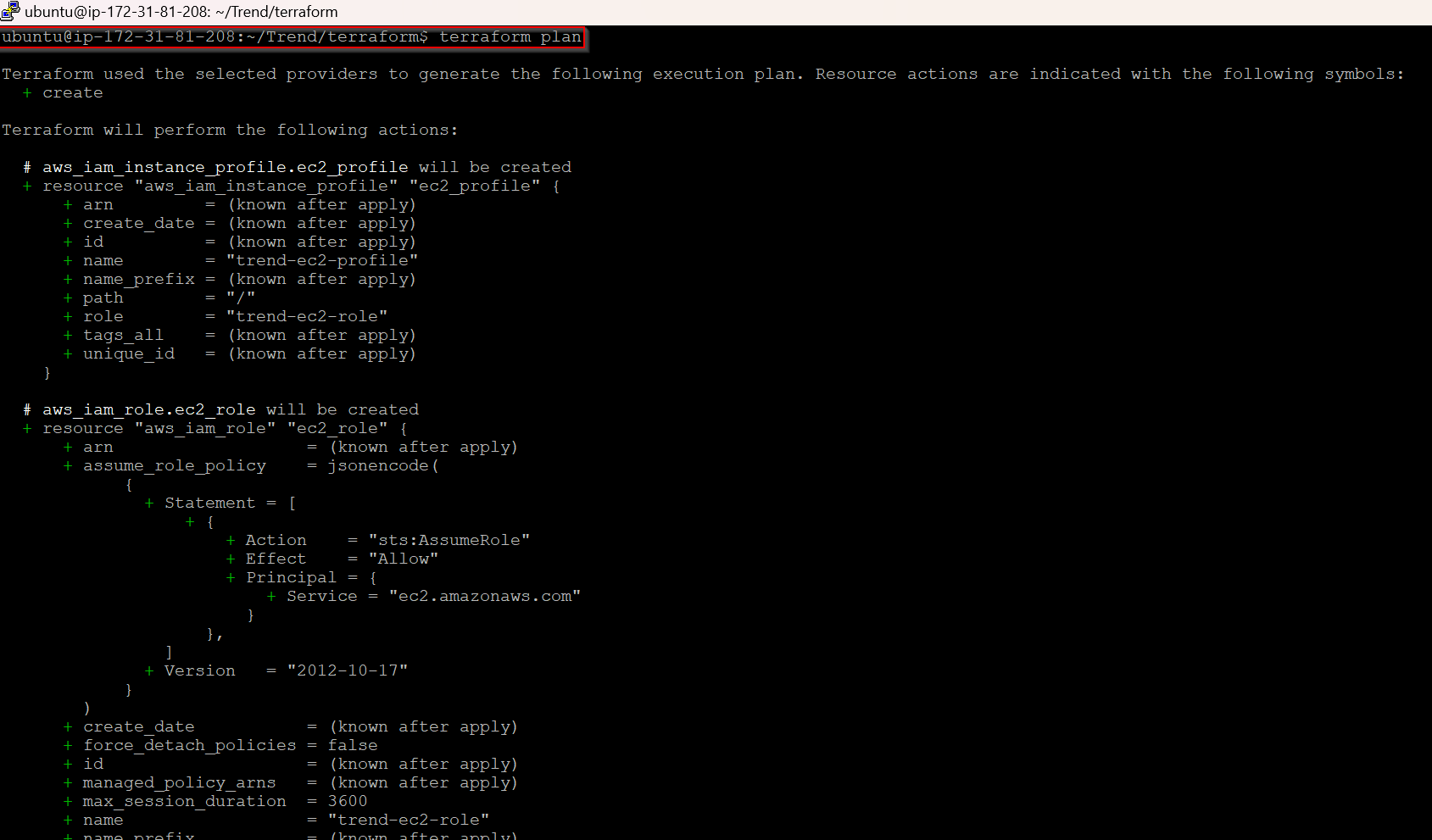
terraform init

terraform validate

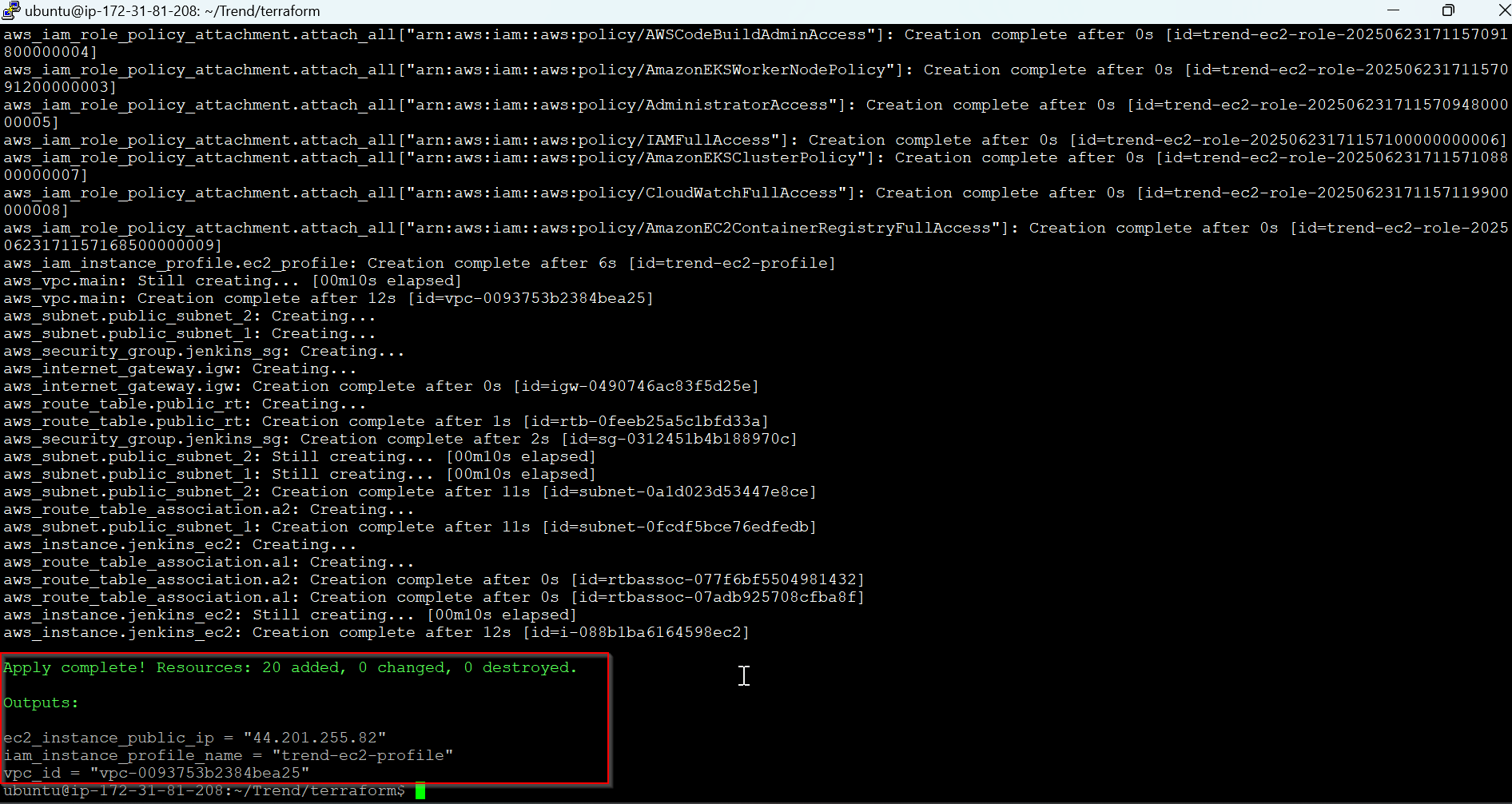
terraform plan

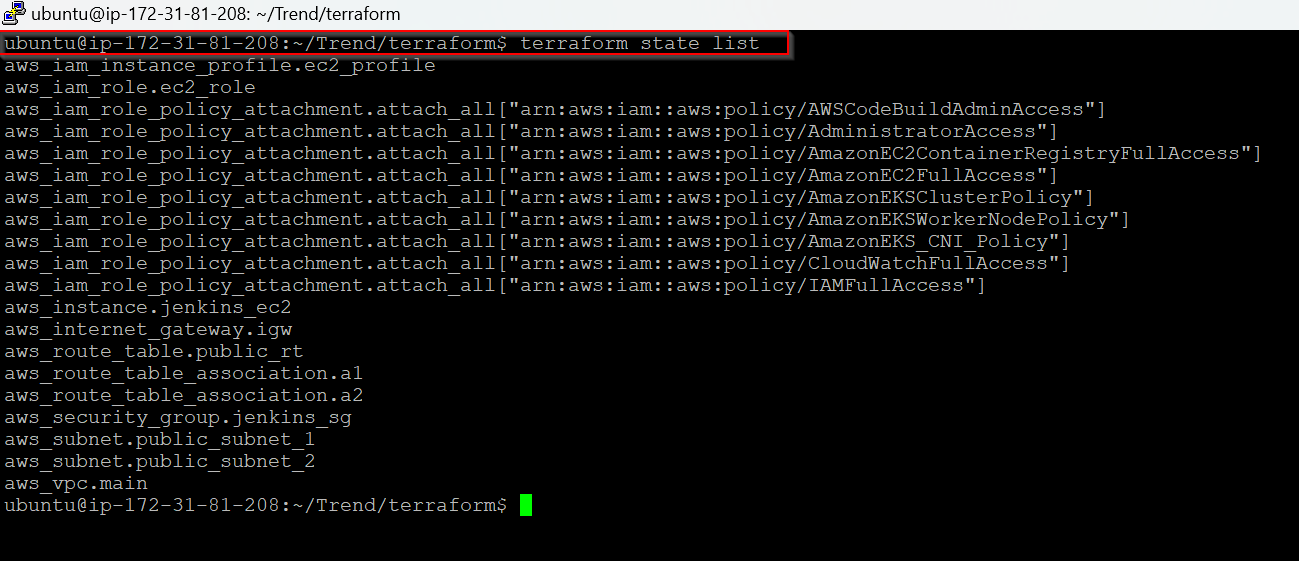
terraform apply





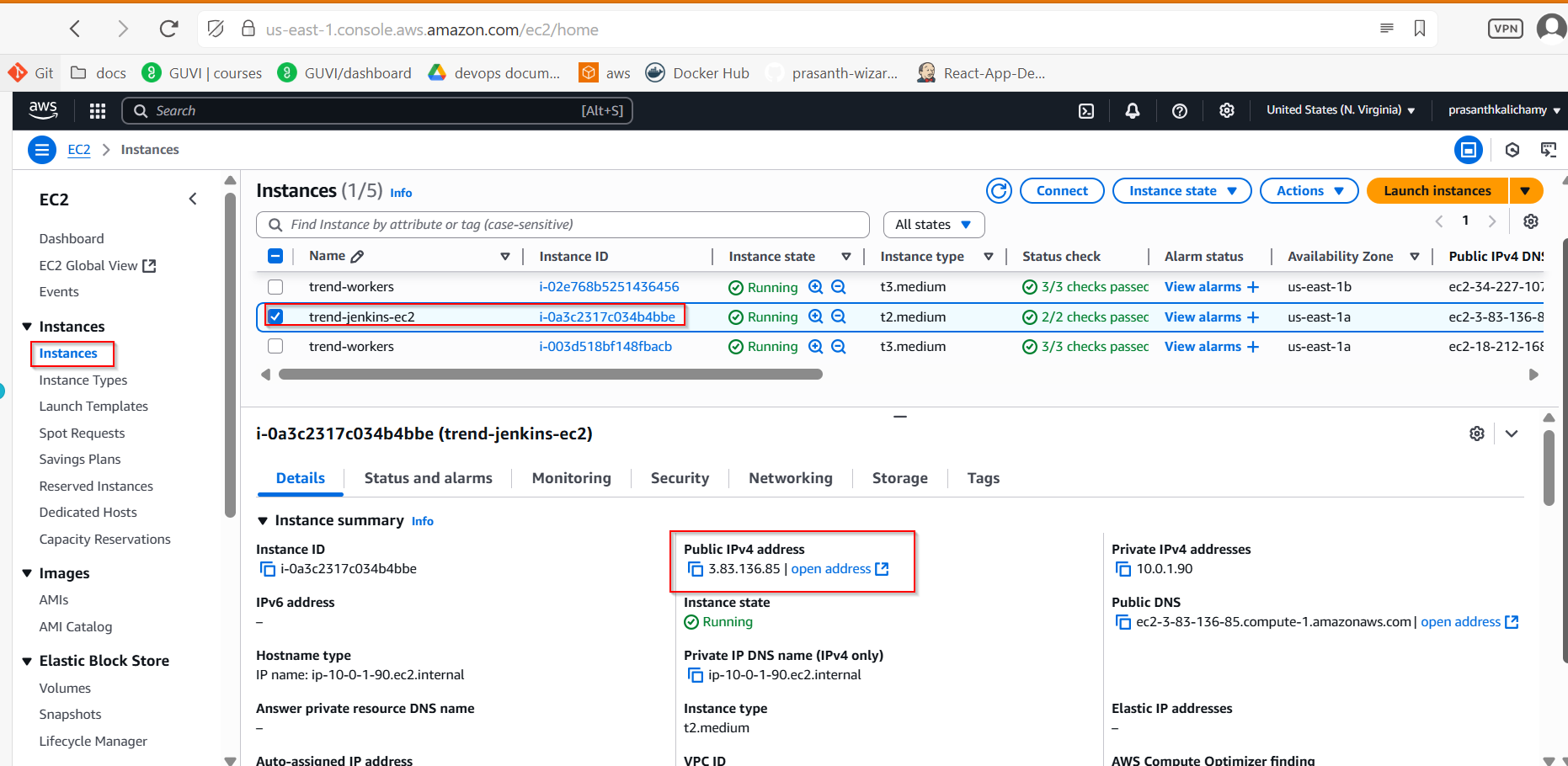
**Terraform apply:**



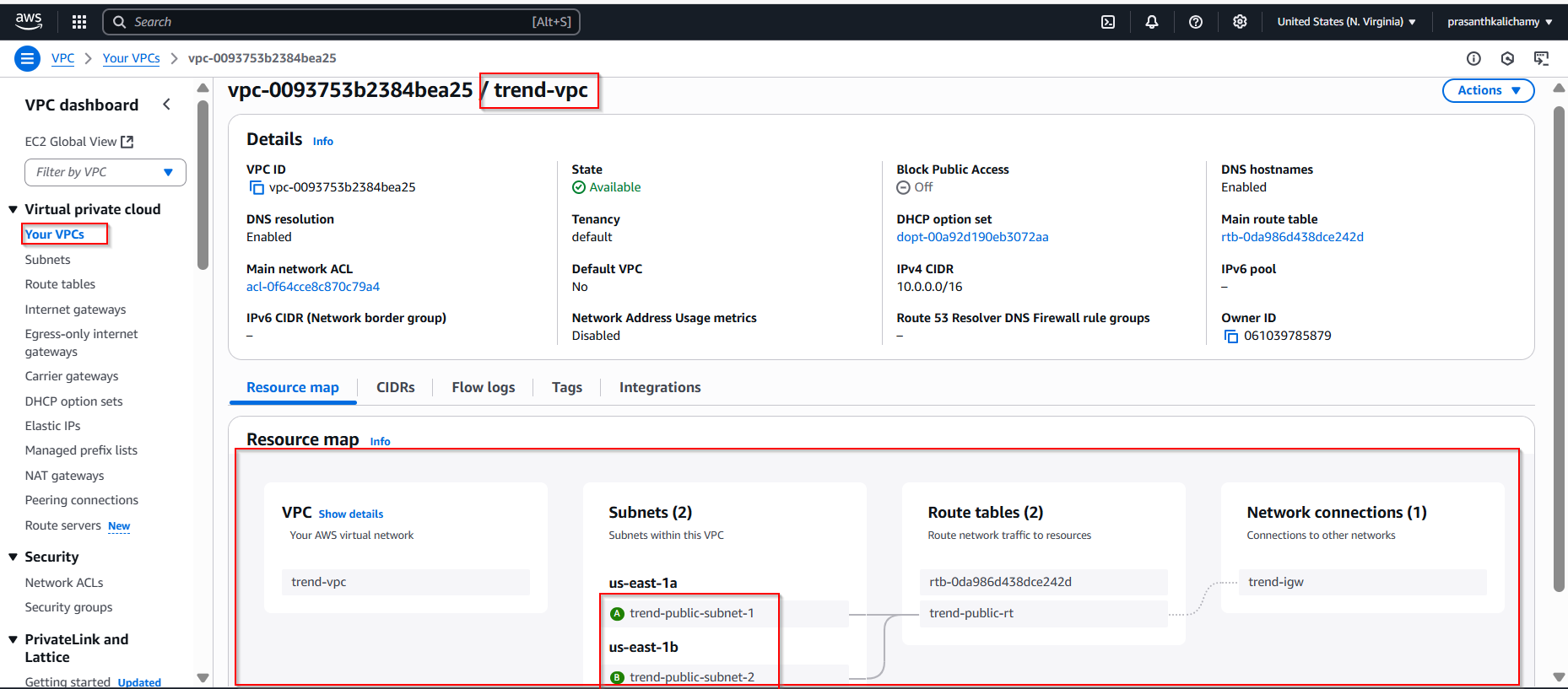


**Terraform infrastructure created:**

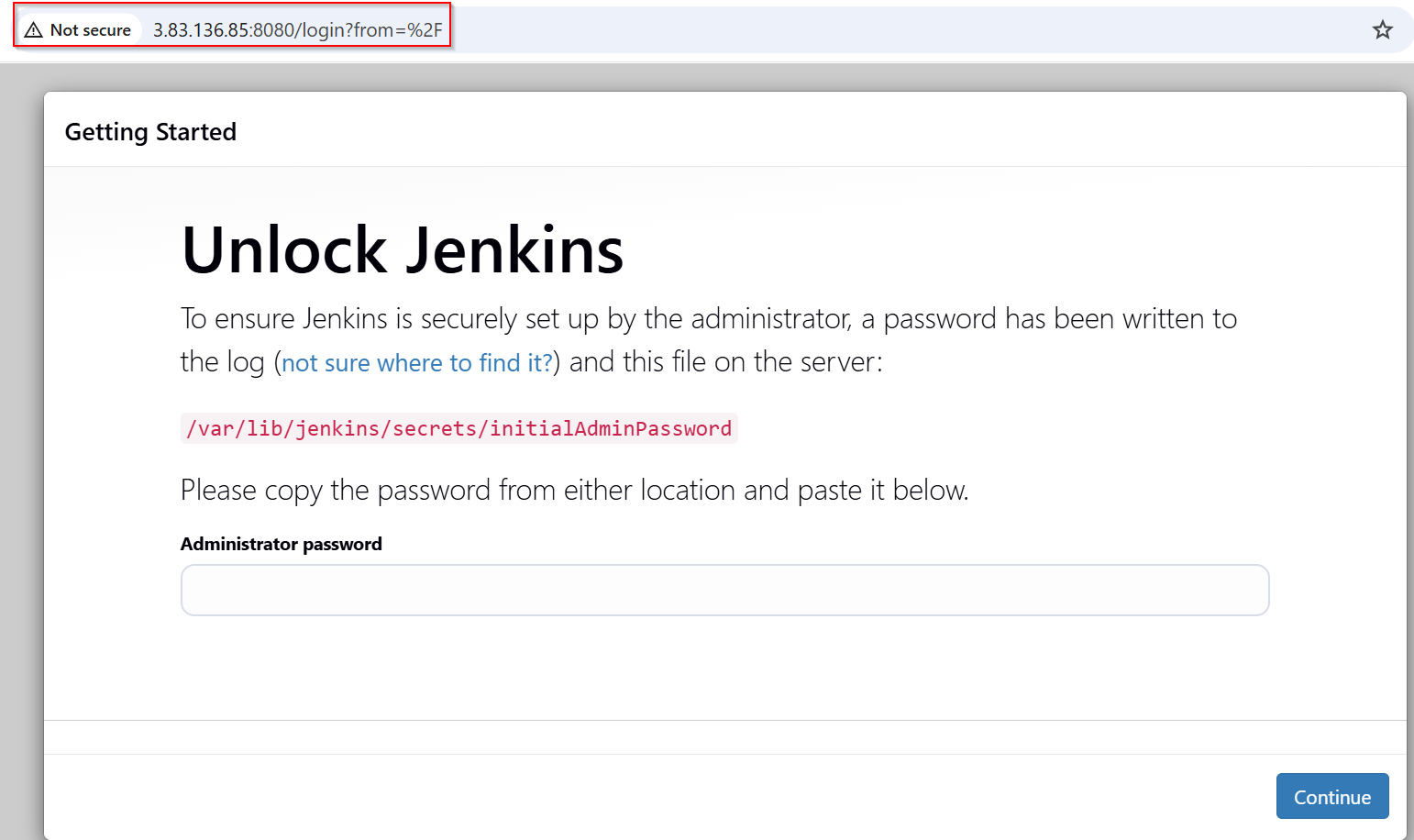
**EC2:**

****

**VPC and Subnet:**

****

**Jenkins:**

****

**Tool Installation 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

**Tool Installation eksctl:**

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

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

eksctl version



**Creating a eks-cluster.tf file**

module "eks" {

source = "terraform-aws-modules/eks/aws"

version = "19.21.0"

cluster\_name = "trend-cluster"

cluster\_version = "1.29"

subnet\_ids = [aws\_subnet.public\_subnet\_1.id, aws\_subnet.public\_subnet\_2.id]

vpc\_id = aws\_vpc.main.id

cluster\_endpoint\_public\_access = true

eks\_managed\_node\_group\_defaults = {

ami\_type = "AL2\_x86\_64"

instance\_types = ["t3.medium"]

}

eks\_managed\_node\_groups = {

trend-workers = {

min\_size = 1

max\_size = 2

desired\_size = 2

tags = {

Name = "trend-eks-node"

}

iam\_role\_additional\_policies = {

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

AmazonSSMManagedInstanceCore = "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"

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

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

}

}

}

tags = {

Environment = "dev"

Project = "trend"

}

}

**Creating a deployment.yaml file**

apiVersion: apps/v1

kind: Deployment

metadata:

name: trend-app

labels:

app: trend-app

spec:

replicas: 2

selector:

matchLabels:

app: trend-app

template:

metadata:

labels:

app: trend-app

spec:

containers:

- name: trend-app

image: prasanth0003/react\_app:latest

ports:

- containerPort: 3000

**Creating a service.yaml file**

apiVersion: v1

kind: Service

metadata:

name: trend-app-service

spec:

selector:

app: trend-app

ports:

- protocol: TCP

port: 3000

targetPort: 3000

type: LoadBalancer

**Creating a .dockerignore**

node\_modules

.git

.gitignore

.env

\*.log

**Creating a . gitignore**

node\_modules/

dist/

.env

\*.log

\*.pem

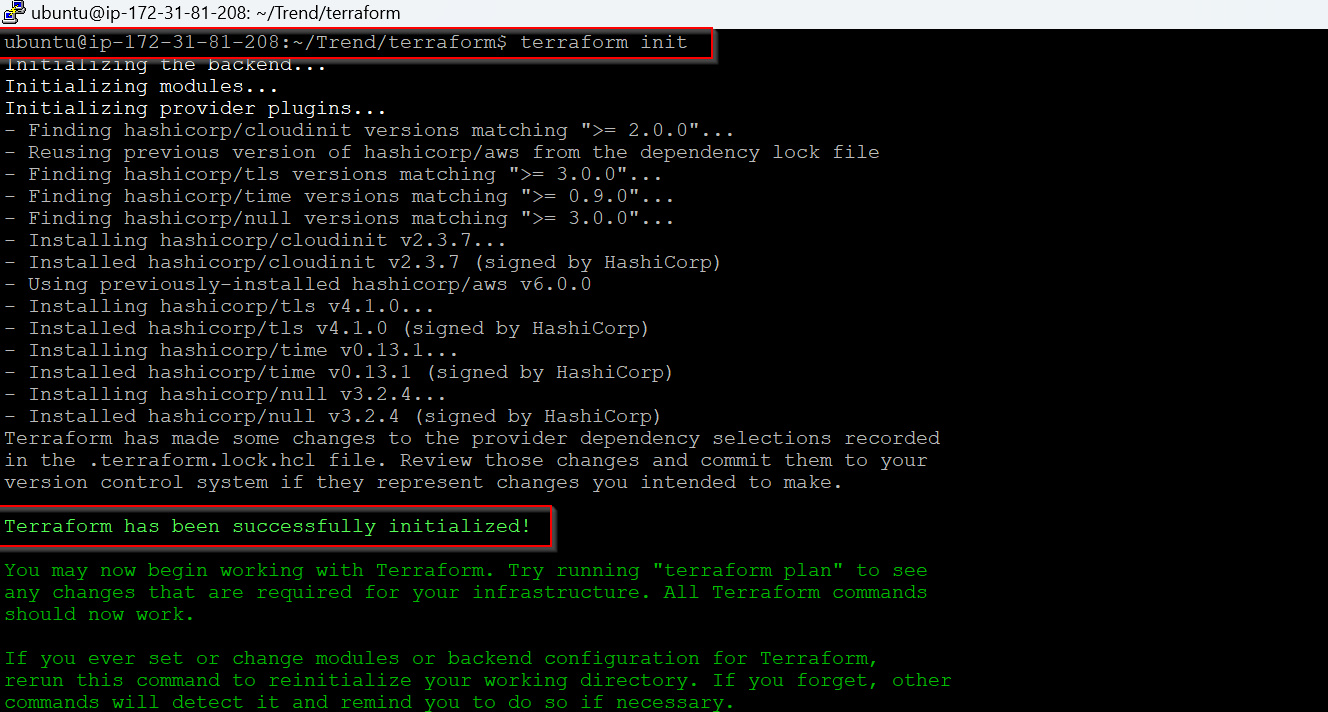
\*.tfstate

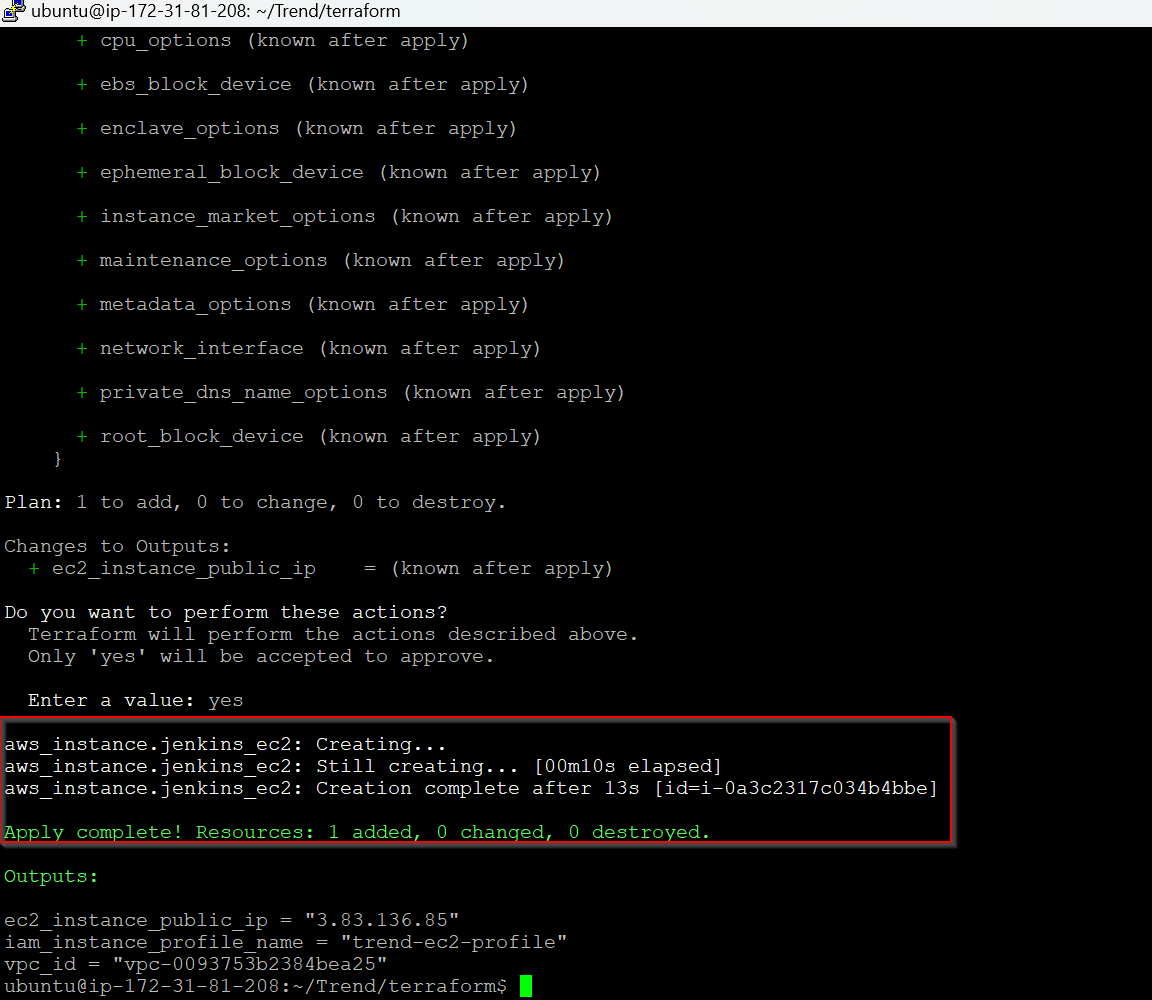
.terraform/

**Run Terraform to create EKS:**

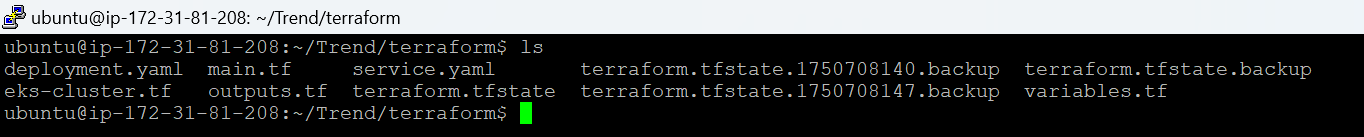
terraform init

Terraform apply

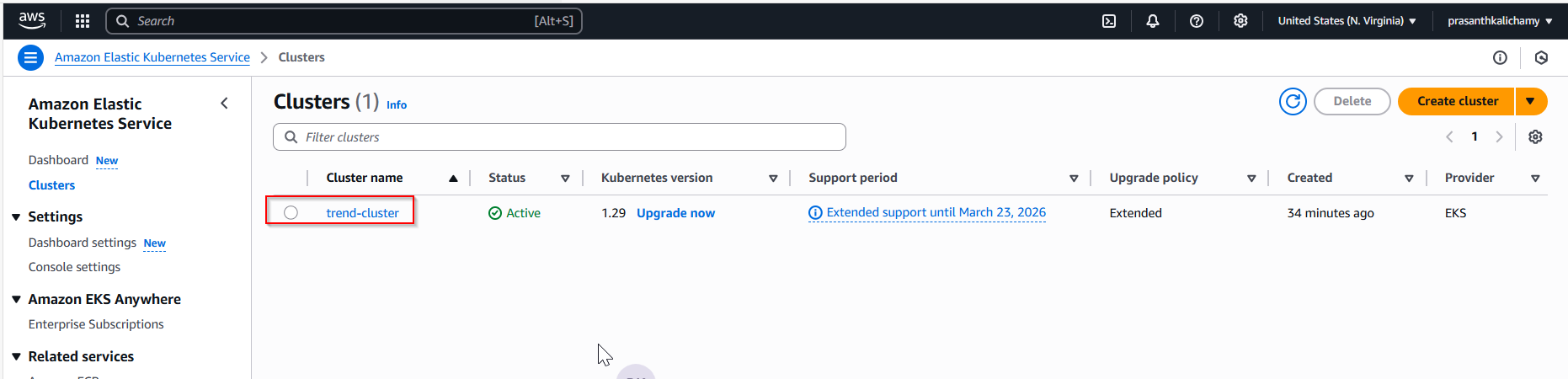


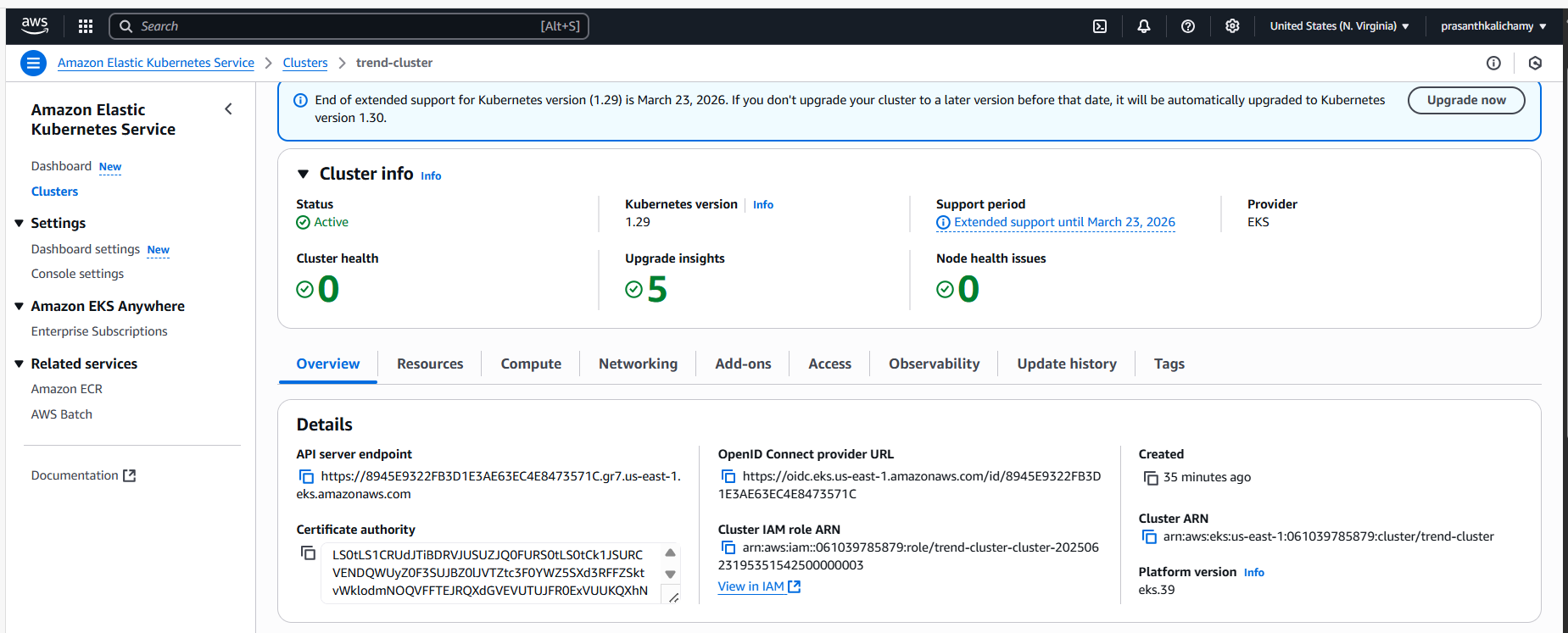
****

**State file has been created successfully:**

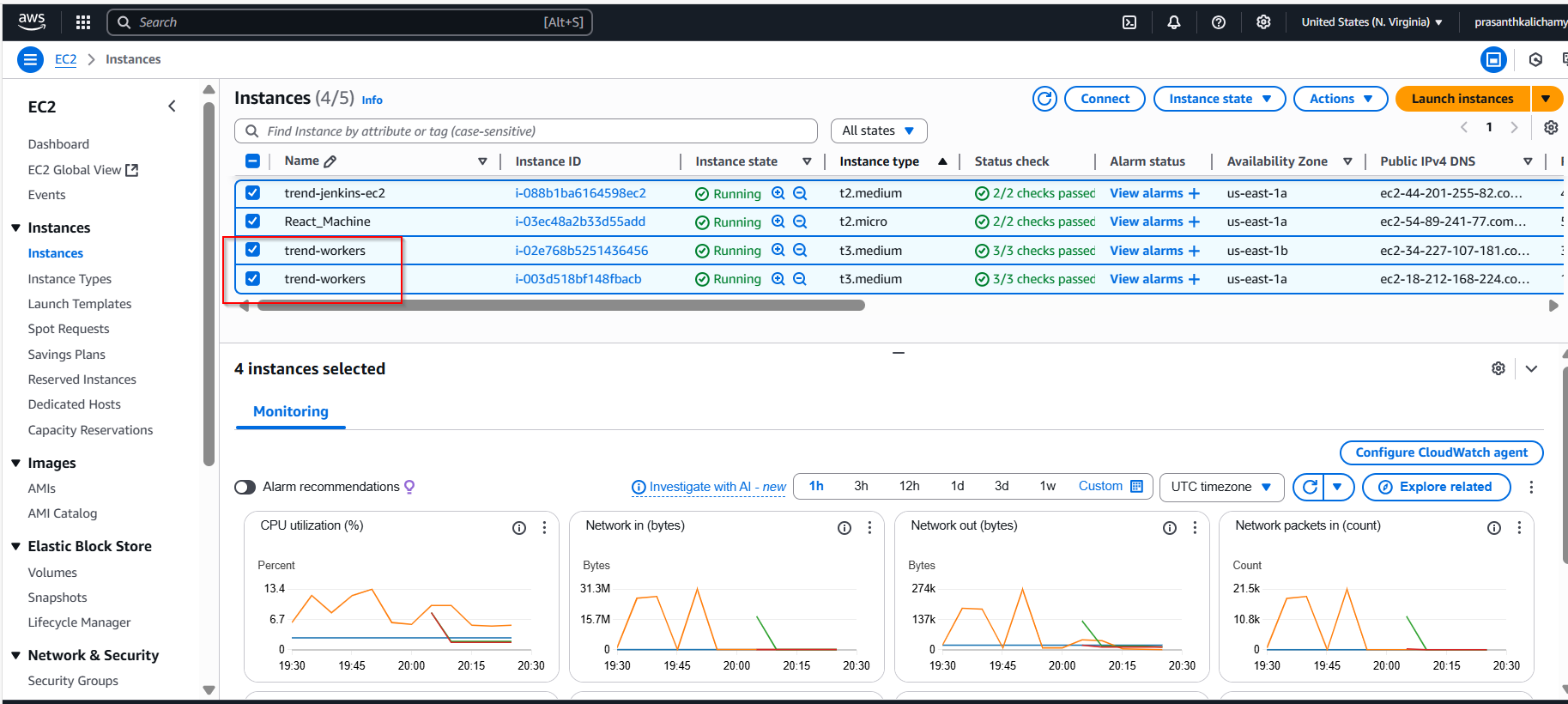


**Cluster created successfully:**

****

****

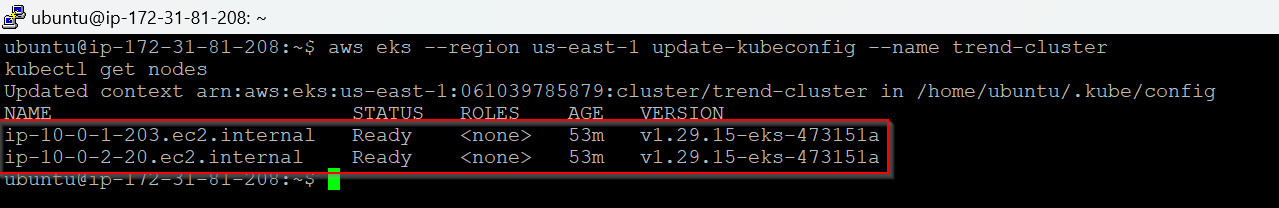
**Worker nodes created successfully:**

****

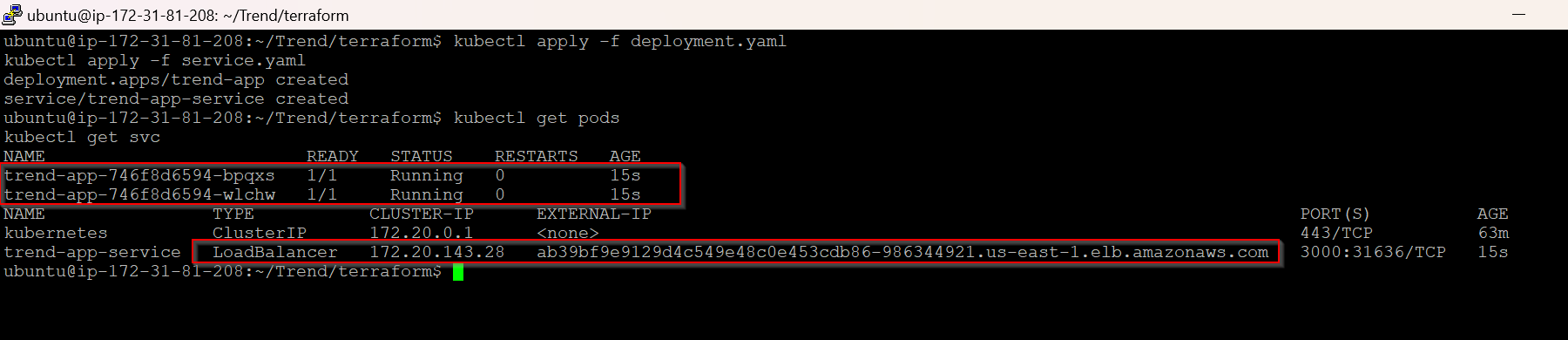
**Update** kubeconfig **on your EC2 (Jenkins) instance:**

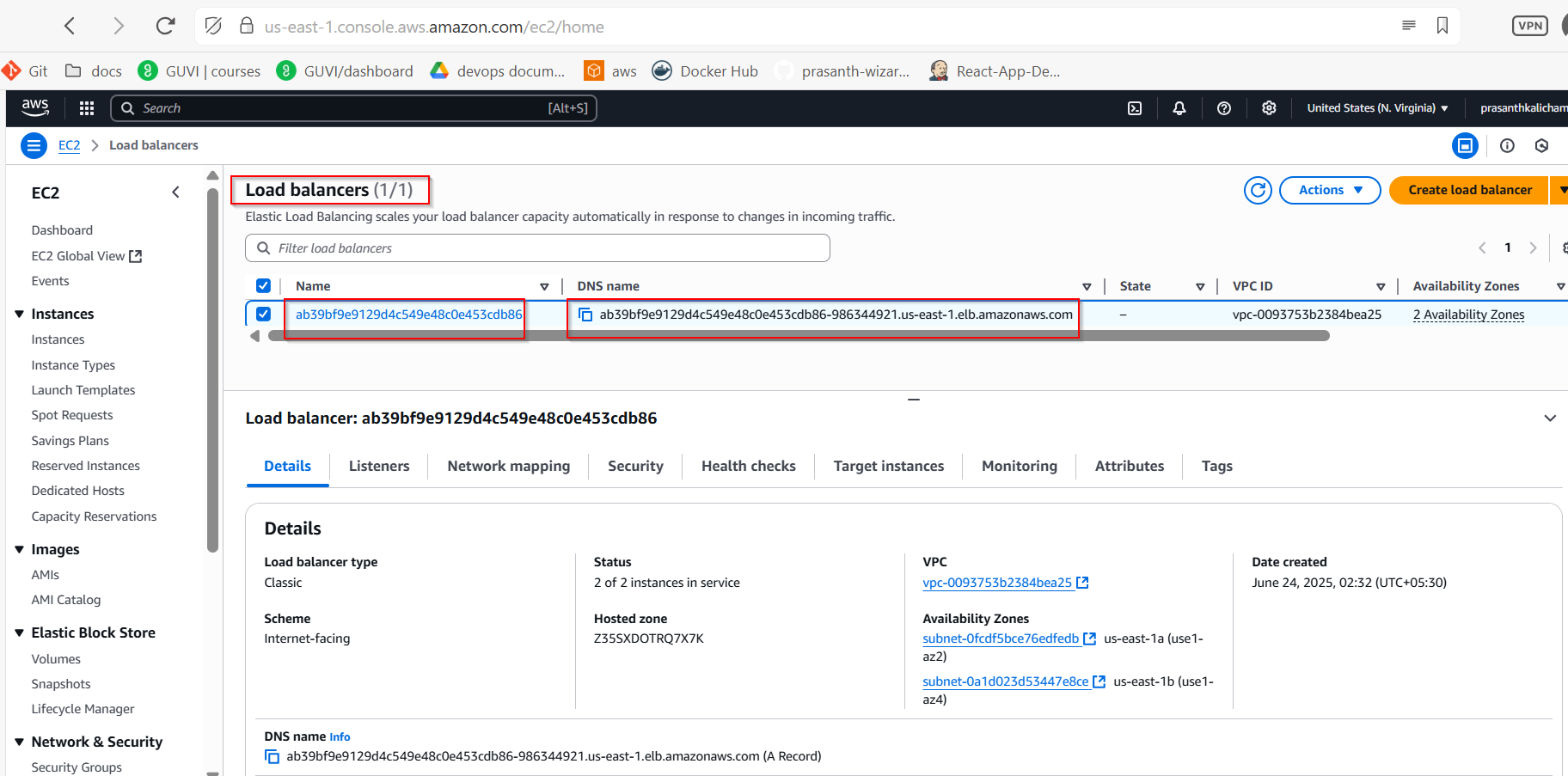
aws eks --region us-east-1 update-kubeconfig --name trend-cluster

kubectl get nodes

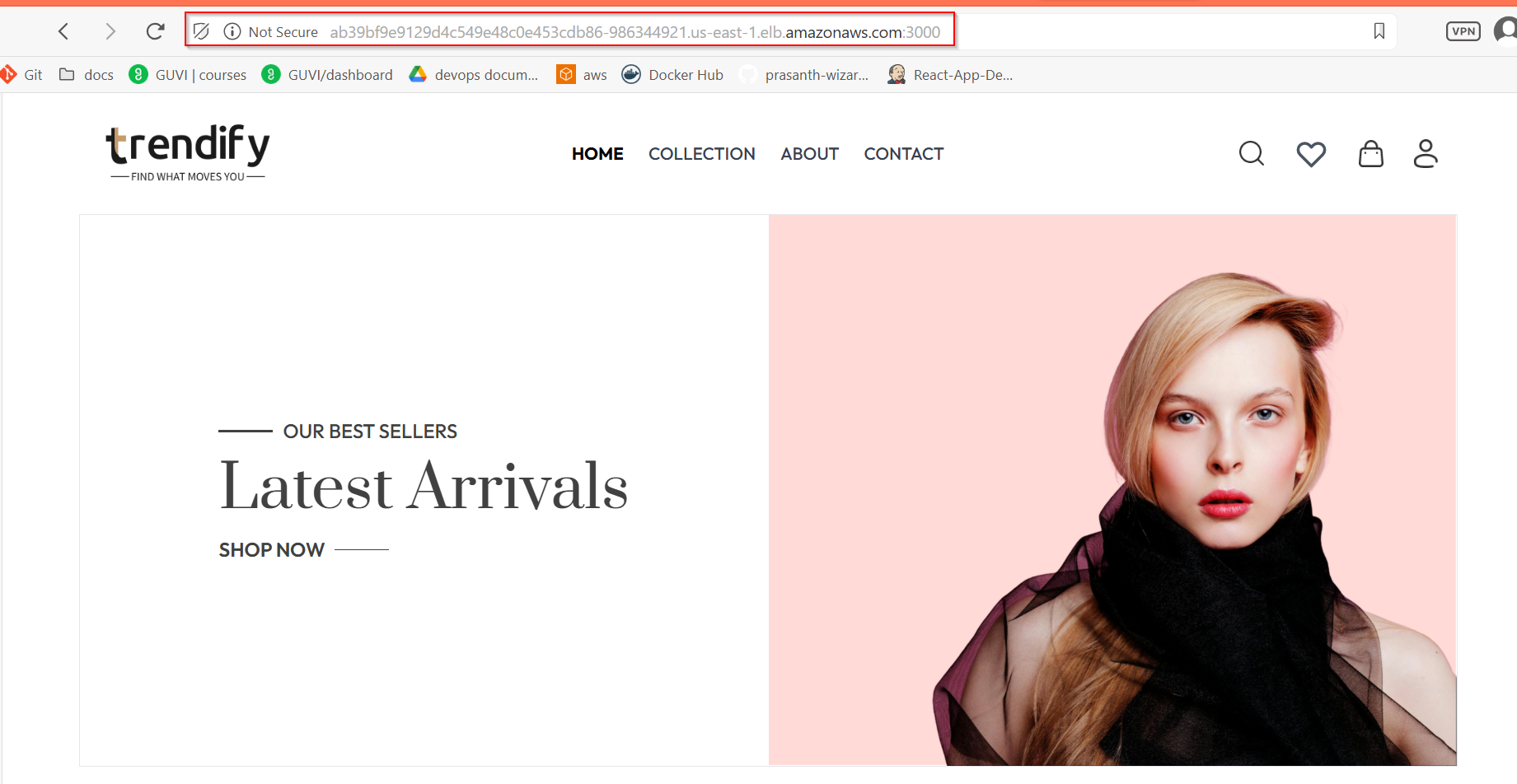


**Deploying the application:**

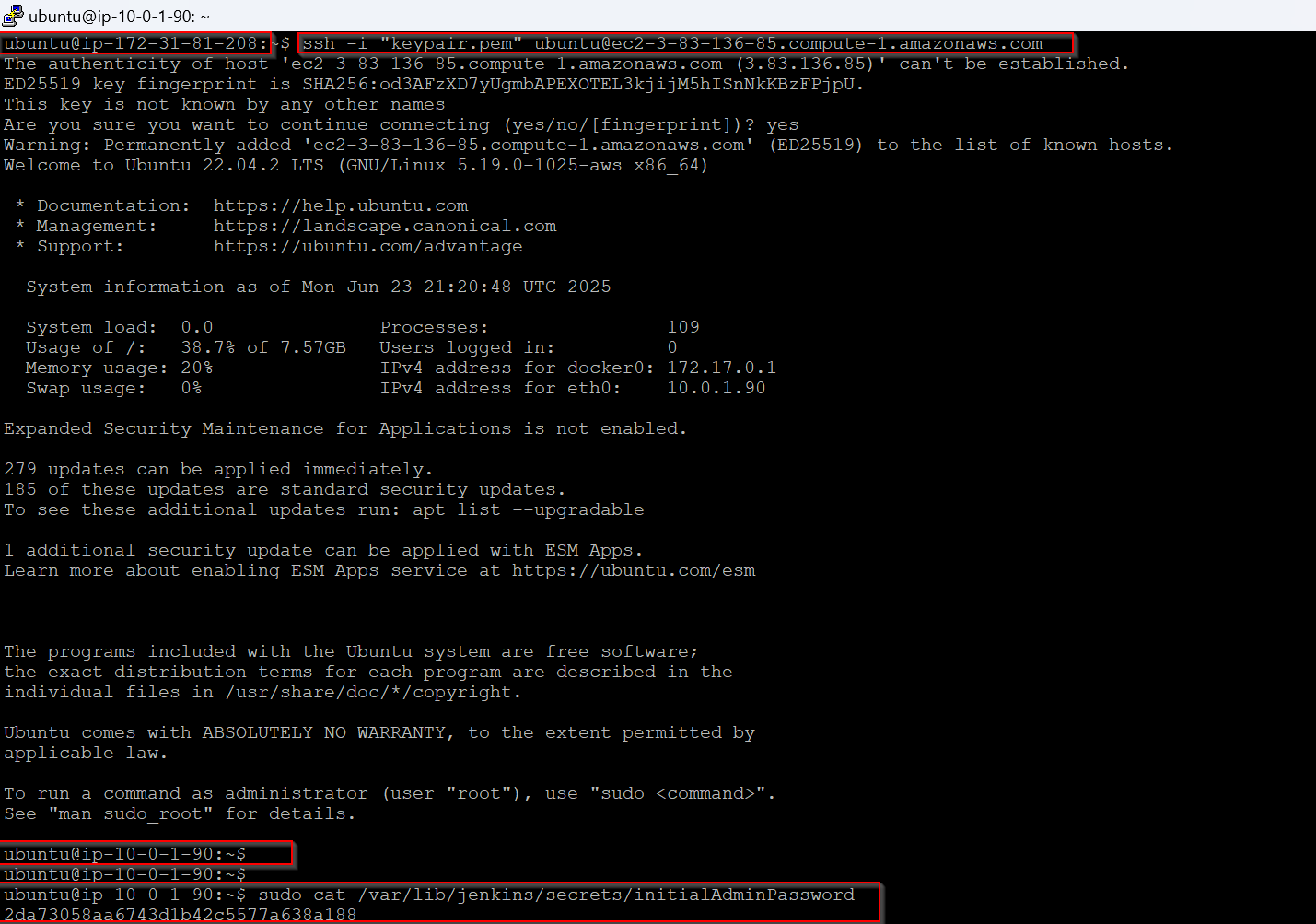
****

****

DNS: ab39bf9e9129d4c549e48c0e453cdb86-986344921.us-east-1.elb.amazonaws.com

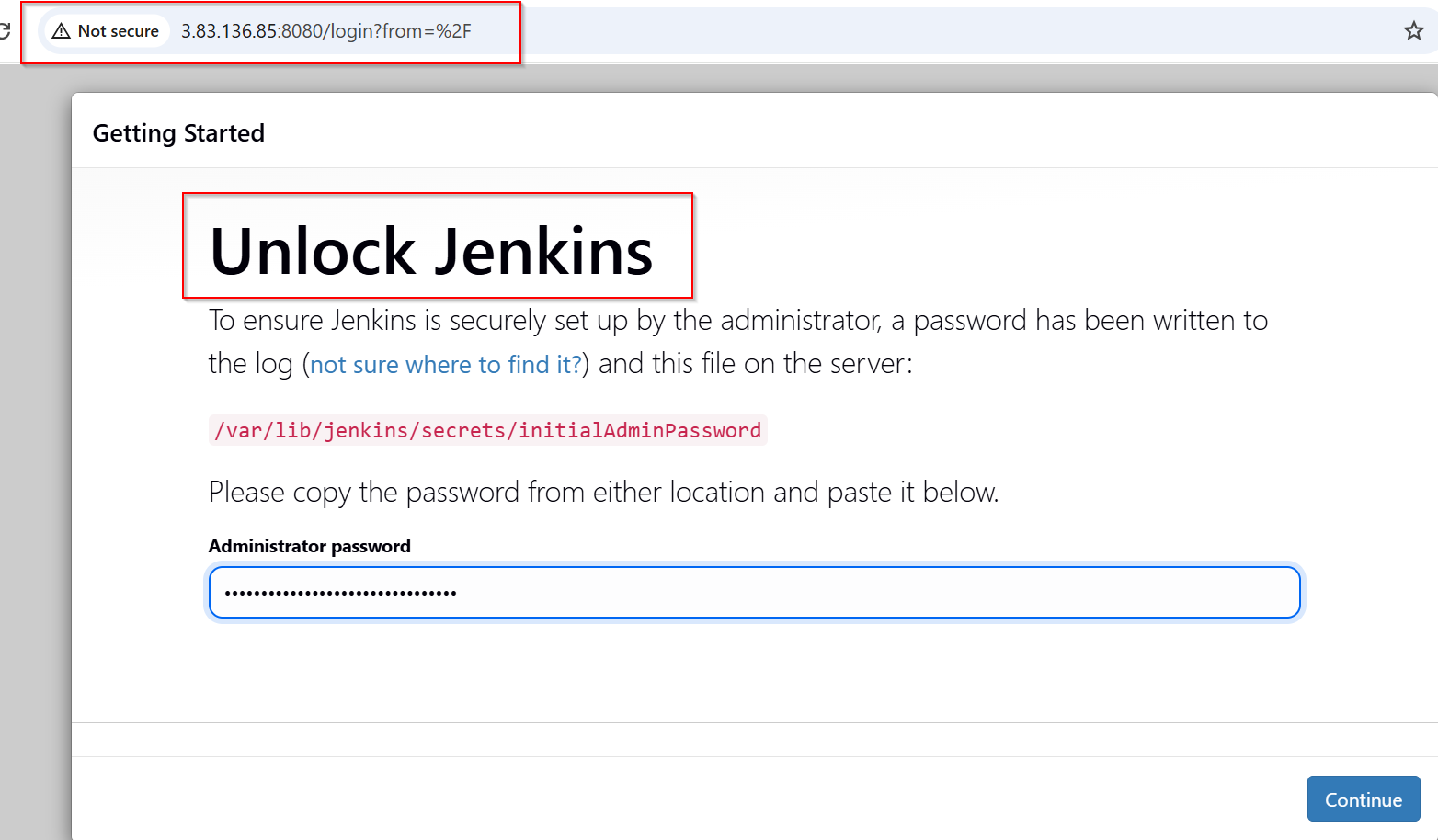
****

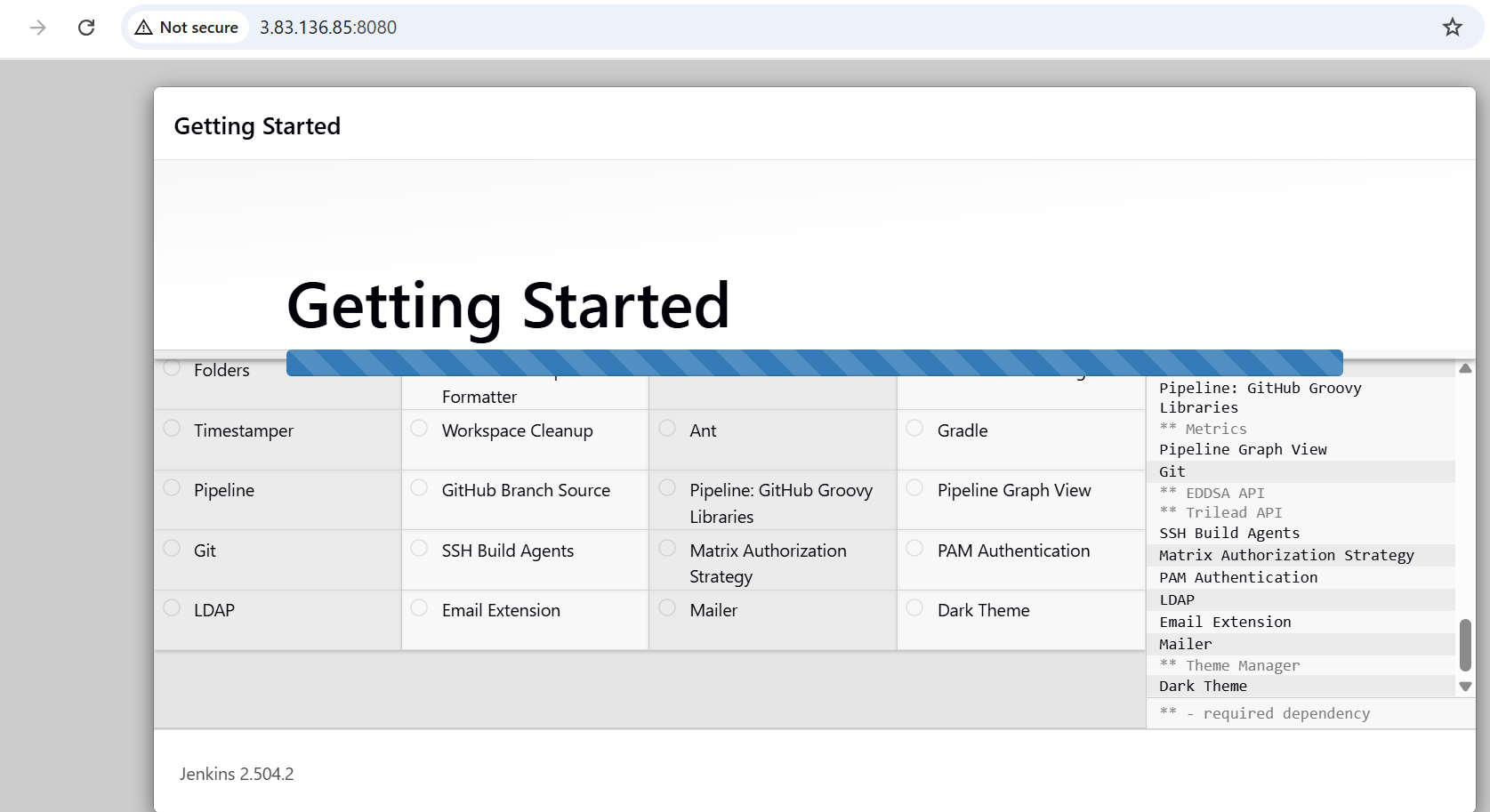
**Connected to the Jenkins server:**

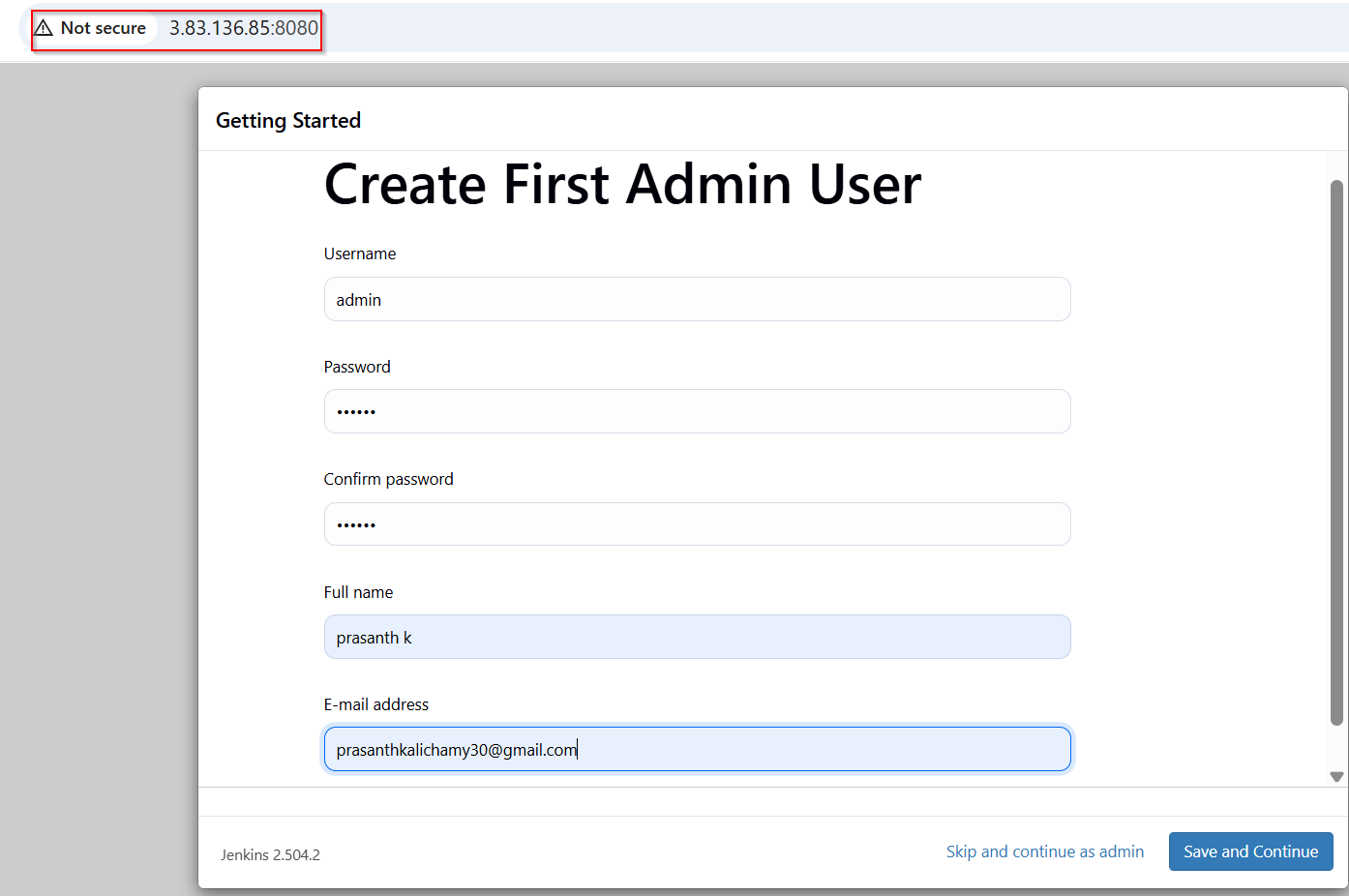
****

**Jenkins Automation Configration:**

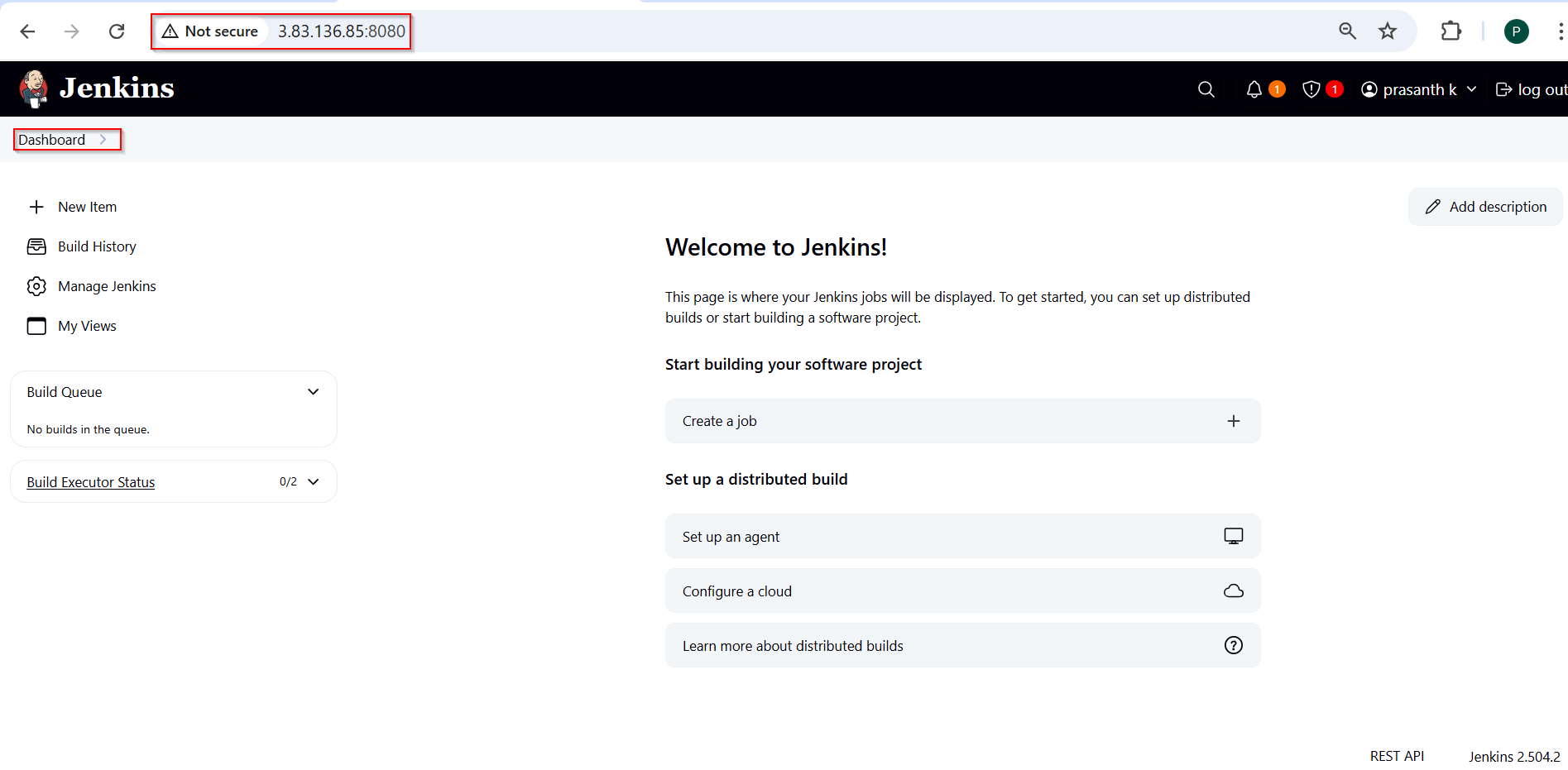
**Jenkins login:**

****

****

****

**Dashboard:**

****

**Jenkins Plugin Installation:**

* Docker Pipeline
* Kubernetes CLI Plugin
* GitHub Integration Plugin
* Pipeline
* Blue Ocean
* Docker Commons Plugin

sudo systemctl restart Jenkins

**Docker & GitHub Credentials Setup:**

**For Docker:**

Kind - Username with password

Username - prasanth0003

Password - dckr\_pat\_JzuwE76Mhzexn\_niQUenqTp6VZA

ID - dockerhub

Description - DockerHub PAT for pushing images

**For Github:**

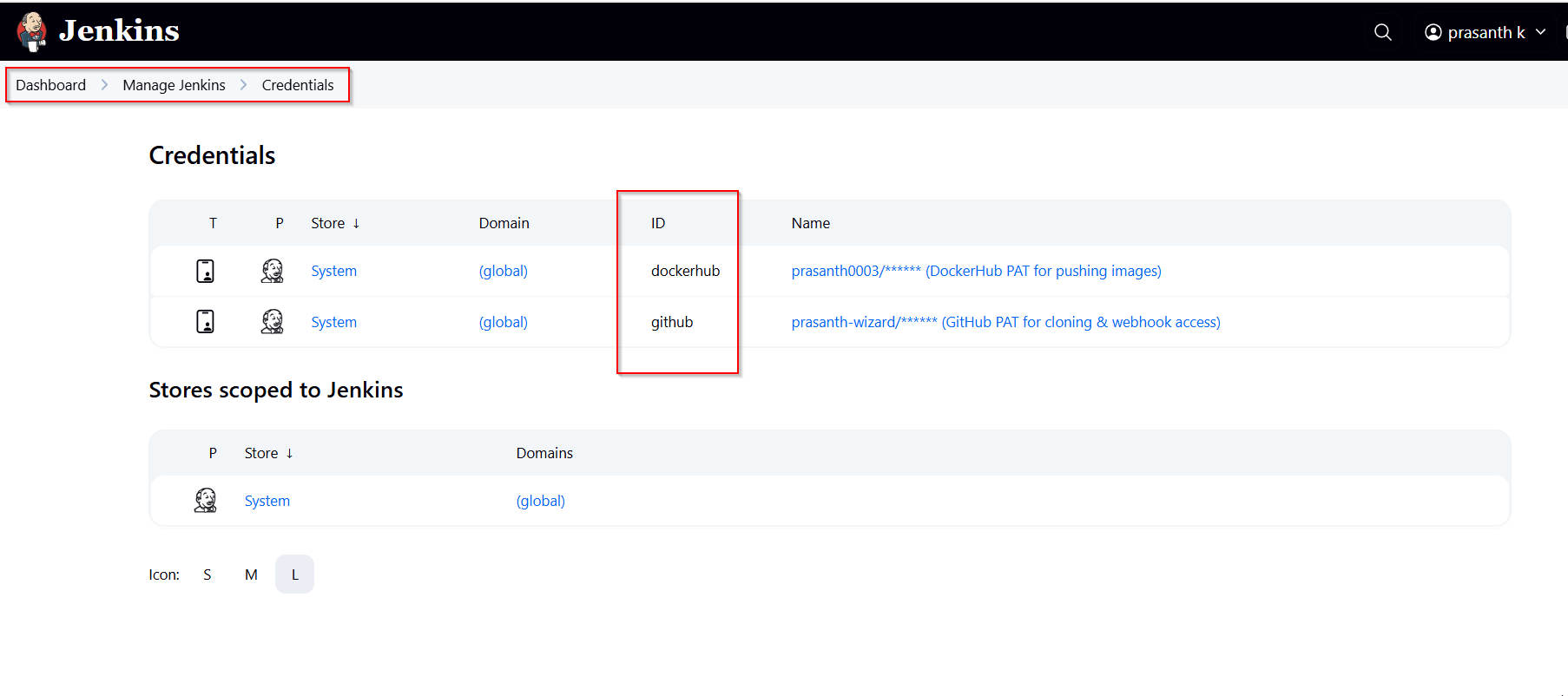
Kind - Username with password

Username - prasanth-wizard

Password - github\_pat\_11BROM3OA0QcMxJ2KuOWaZ\_x4znHMT3Xz76dPIT1p08hJpR9778110Niljs1plcrqVFEOUZAFObSs6FekM

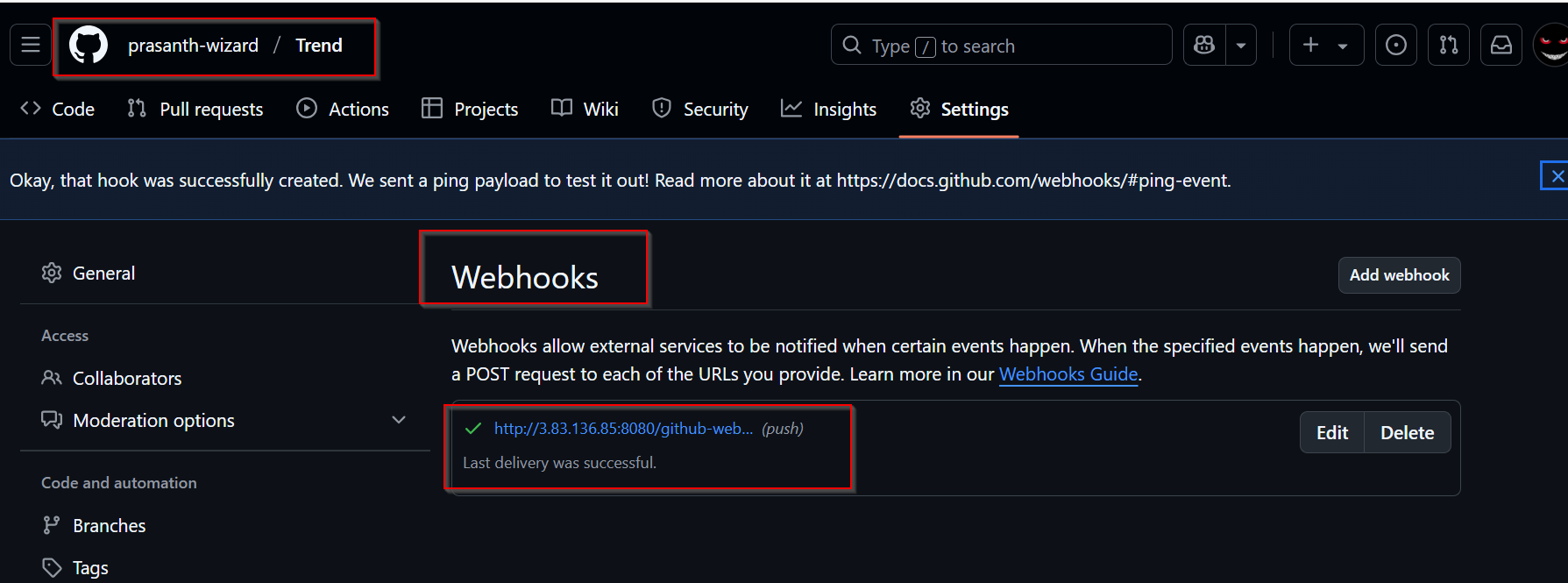
ID - github

Description - GitHub PAT for cloning & webhook access

****

**Add GitHub Webhook (For Auto CI/CD):**

* Payload URL: http://3.83.136.85:8080/github-webhook/
* Content type: application/json
* Secret: Leave blank
* Events: Choose Just the push event



**Creating Jenkinsfile:**

pipeline {

agent any

environment {

DOCKER\_IMAGE = "prasanth0003/react\_app"

DOCKER\_CREDENTIALS\_ID = 'dockerhub'

}

stages {

stage('Clone Repository') {

steps {

git branch: 'main', credentialsId: 'github', url: 'https://github.com/prasanth-wizard/Trend.git'

}

}

stage('Build Docker Image') {

steps {

script {

docker.build("${DOCKER\_IMAGE}")

}

}

}

stage('Login & Push to DockerHub') {

steps {

script {

docker.withRegistry('', DOCKER\_CREDENTIALS\_ID) {

docker.image("${DOCKER\_IMAGE}").push("latest")

}

}

}

}

stage('Deploy to EKS') {

steps {

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

sh 'kubectl apply -f deployment.yaml'

sh 'kubectl apply -f service.yaml'

sh 'kubectl get svc'

}

}

}

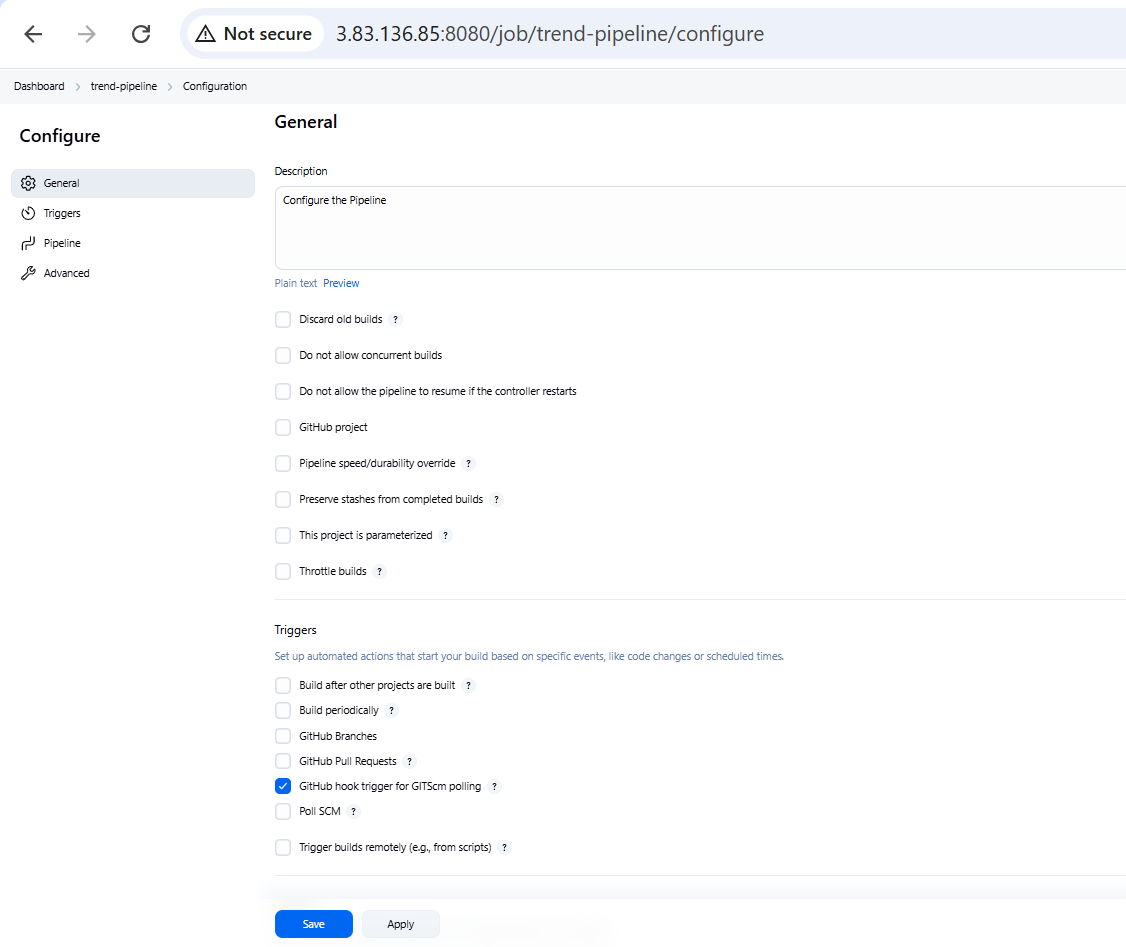
}

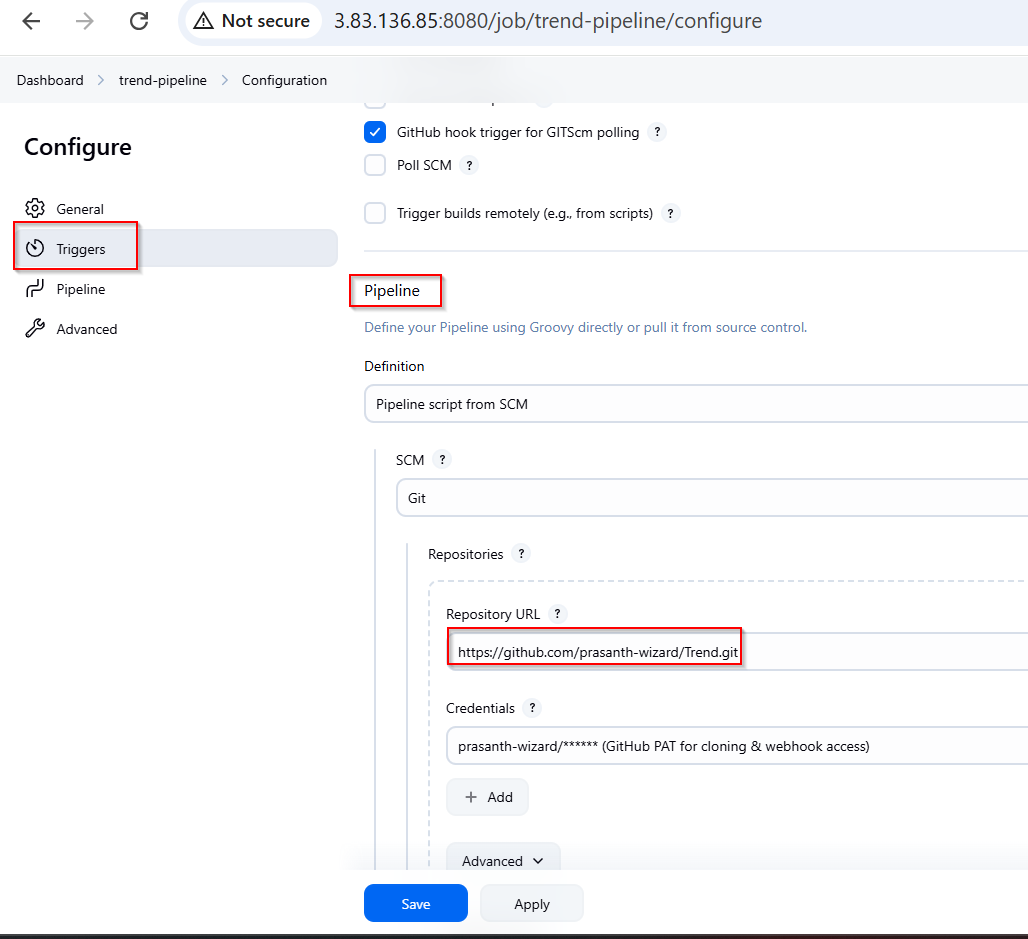
**Create Jenkins Pipeline:**

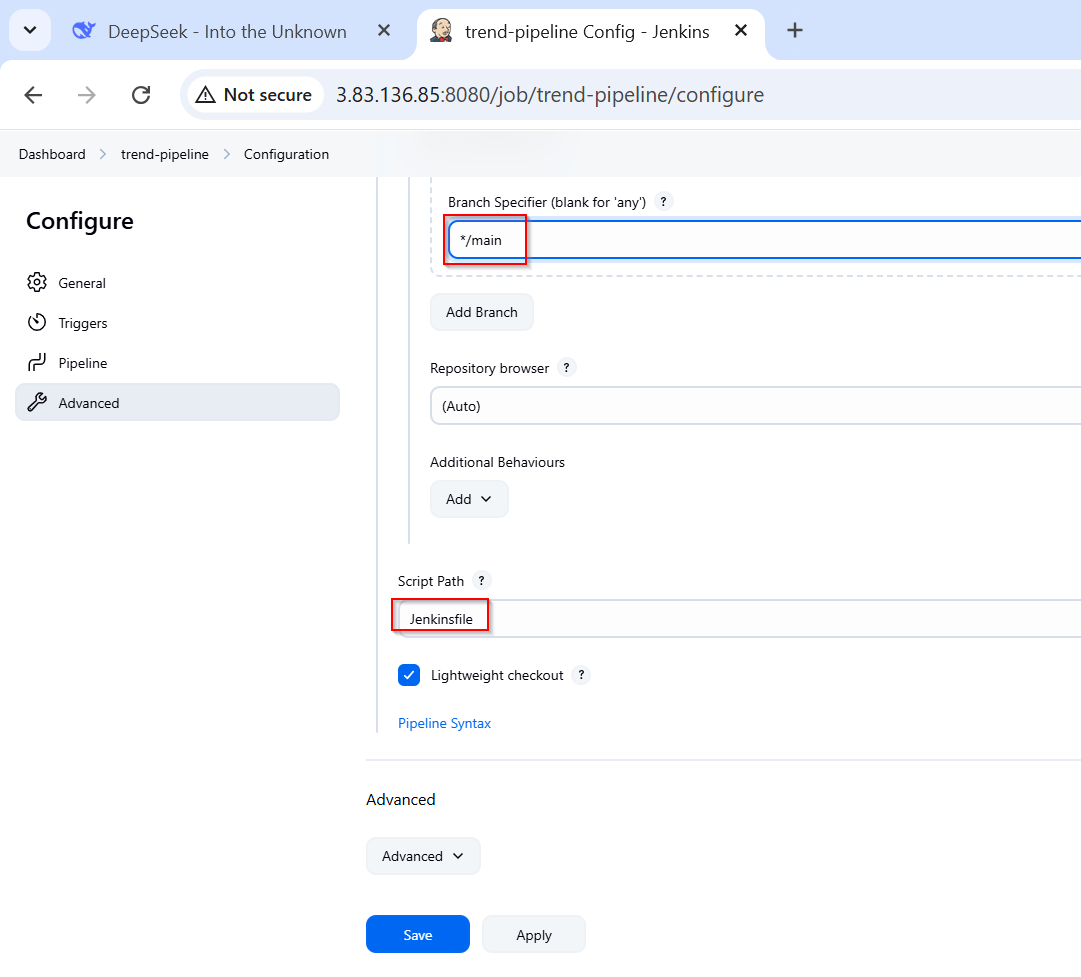
**Create New Pipeline Job:**

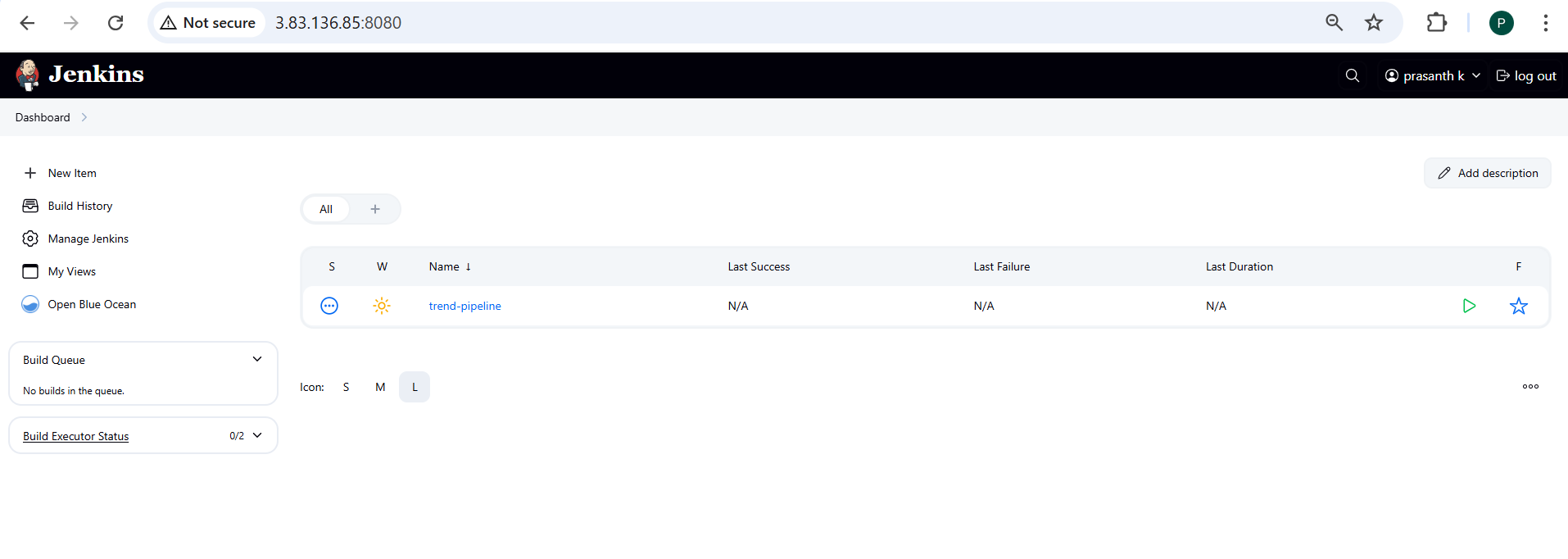
* Click **“New Item”**
* Enter **“trend-pipeline”** as the job name
* Select **“Pipeline”**
* Click **OK**

**Configure the Pipeline:**

****

****

****

****

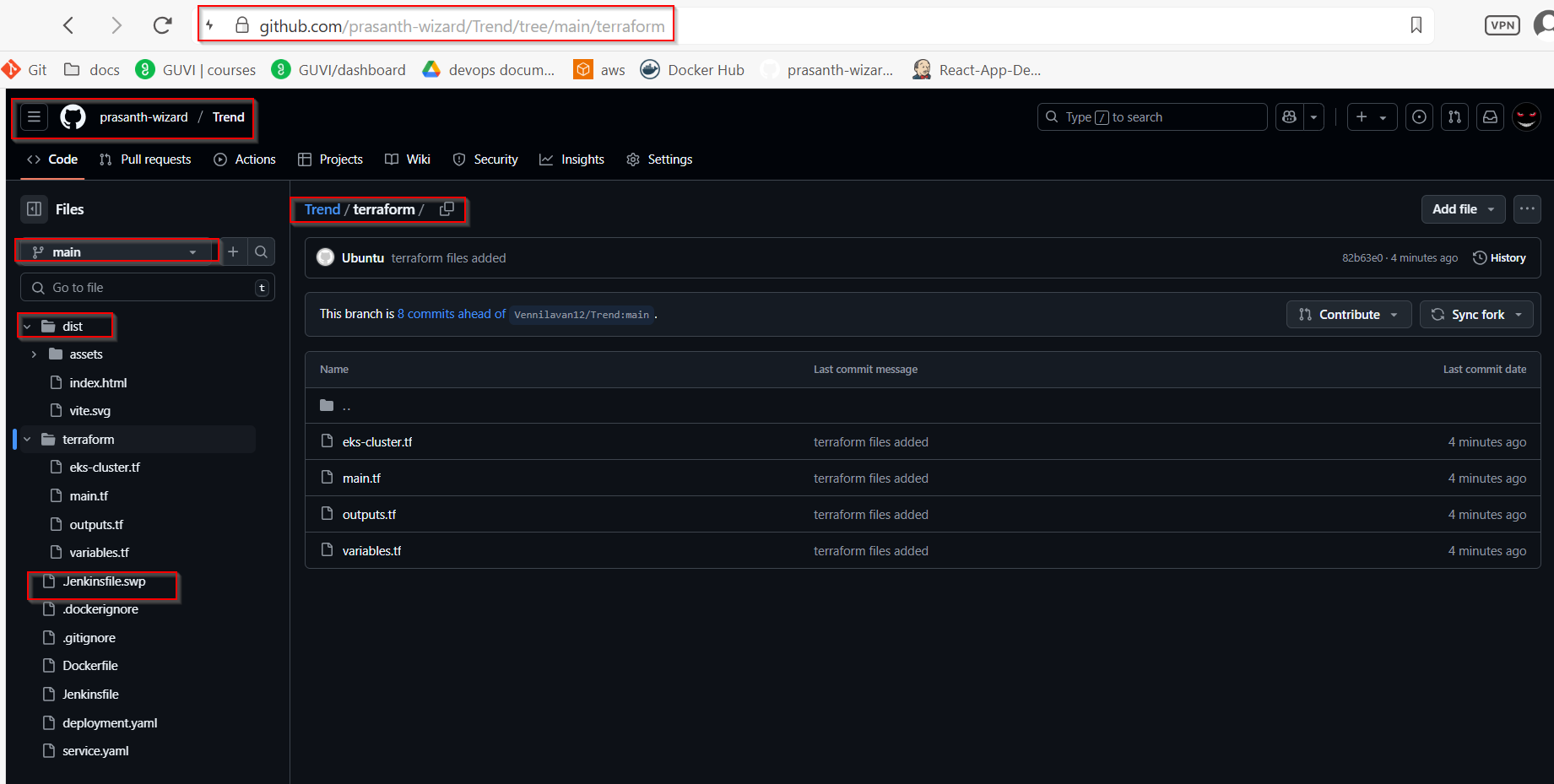
**File push to github repo:**

Git add .

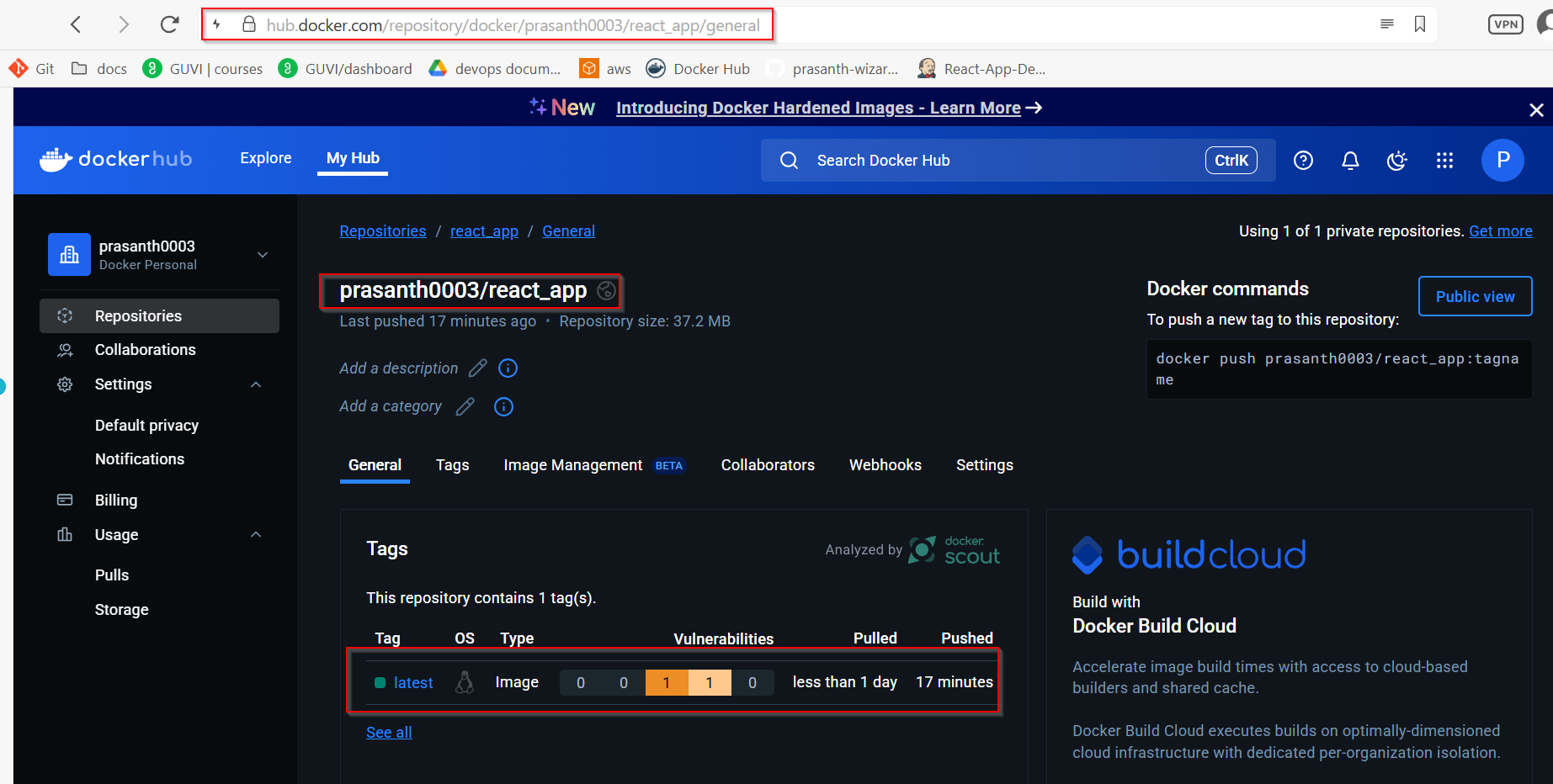
Git commit -m “ files has been added”

Git push origin main

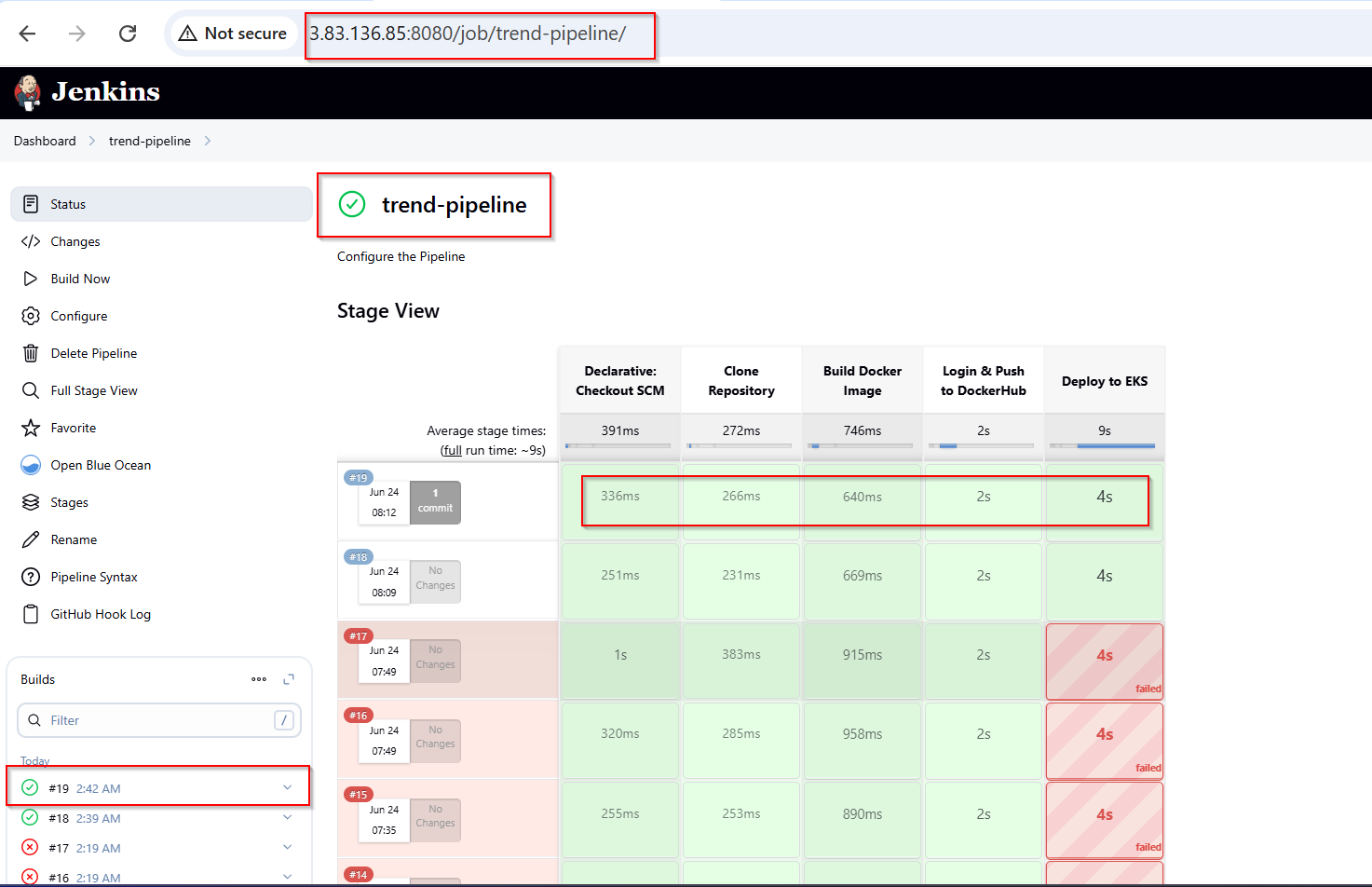
**GitHub:**

****

**DockerHub:**

****

**Jenkins pipeline trigged correctly:**

****

**Console Output:**

Started by GitHub push by prasanth-wizard

Obtained Jenkinsfile from git <https://github.com/prasanth-wizard/Trend.git>

[Pipeline] Start of Pipeline

[Pipeline] node

Running on [Jenkins](http://3.83.136.85:8080/computer/(built-in)/) in /var/lib/jenkins/workspace/trend-pipeline

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

The recommended git tool is: git

using credential github

> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/trend-pipeline/.git # timeout=10

Fetching changes from the remote Git repository

> git config remote.origin.url <https://github.com/prasanth-wizard/Trend.git> # timeout=10

Fetching upstream changes from <https://github.com/prasanth-wizard/Trend.git>

> git --version # timeout=10

> git --version # 'git version 2.34.1'

using GIT\_ASKPASS to set credentials GitHub PAT for cloning & webhook access

> git fetch --tags --force --progress -- <https://github.com/prasanth-wizard/Trend.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git rev-parse refs/remotes/origin/main^{commit} # timeout=10

Checking out Revision 739a8107cc0fb7bc08191e843521499fbbeb10bc (refs/remotes/origin/main)

> git config core.sparsecheckout # timeout=10

> git checkout -f 739a8107cc0fb7bc08191e843521499fbbeb10bc # timeout=10

Commit message: "Jenkins file updated"

> git rev-list --no-walk f4ae2ef4c591725a6a0cdc95b2058e84d824e10b # timeout=10

[Pipeline] }

[Pipeline] // stage

[Pipeline] withEnv

[Pipeline] {

[Pipeline] withEnv

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Clone Repository)

[Pipeline] git

The recommended git tool is: git

using credential github

> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/trend-pipeline/.git # timeout=10

Fetching changes from the remote Git repository

> git config remote.origin.url <https://github.com/prasanth-wizard/Trend.git> # timeout=10

Fetching upstream changes from <https://github.com/prasanth-wizard/Trend.git>

> git --version # timeout=10

> git --version # 'git version 2.34.1'

using GIT\_ASKPASS to set credentials GitHub PAT for cloning & webhook access

> git fetch --tags --force --progress -- <https://github.com/prasanth-wizard/Trend.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git rev-parse refs/remotes/origin/main^{commit} # timeout=10

Checking out Revision 739a8107cc0fb7bc08191e843521499fbbeb10bc (refs/remotes/origin/main)

> git config core.sparsecheckout # timeout=10

> git checkout -f 739a8107cc0fb7bc08191e843521499fbbeb10bc # timeout=10

> git branch -a -v --no-abbrev # timeout=10

> git branch -D main # timeout=10

> git checkout -b main 739a8107cc0fb7bc08191e843521499fbbeb10bc # timeout=10

Commit message: "Jenkins file updated"

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Build Docker Image)

[Pipeline] script

[Pipeline] {

[Pipeline] isUnix

[Pipeline] withEnv

[Pipeline] {

[Pipeline] sh

+ docker build -t prasanth0003/react\_app .

DEPRECATED: The legacy builder is deprecated and will be removed in a future release.

Install the buildx component to build images with BuildKit:

<https://docs.docker.com/go/buildx/>

Sending build context to Docker daemon 18.51MB

Step 1/6 : FROM nginx:alpine

---> 6769dc3a703c

Step 2/6 : RUN rm -rf /usr/share/nginx/html/\*

---> Using cache

---> 608f0aaaffbf

Step 3/6 : COPY dist/ /usr/share/nginx/html/

---> Using cache

---> c753cacbc8aa

Step 4/6 : EXPOSE 3000

---> Using cache

---> afb2fa994e7f

Step 5/6 : RUN sed -i 's/80;/3000;/' /etc/nginx/conf.d/default.conf

---> Using cache

---> 1c2cdfe7523f

Step 6/6 : CMD ["nginx", "-g", "daemon off;"]

---> Using cache

---> c73a21c031da

Successfully built c73a21c031da

Successfully tagged prasanth0003/react\_app:latest

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // script

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Login & Push to DockerHub)

[Pipeline] script

[Pipeline] {

[Pipeline] withEnv

[Pipeline] {

[Pipeline] withDockerRegistry

$ docker login -u prasanth0003 -p \*\*\*\*\*\*\*\* <https://index.docker.io/v1/>

WARNING! Using --password via the CLI is insecure. Use --password-stdin.

WARNING! Your password will be stored unencrypted in /var/lib/jenkins/workspace/trend-pipeline@tmp/93a784ef-7109-40d6-a1ef-3300e8c0284d/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] isUnix

[Pipeline] withEnv

[Pipeline] {

[Pipeline] sh

+ docker tag prasanth0003/react\_app prasanth0003/react\_app:latest

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] isUnix

[Pipeline] withEnv

[Pipeline] {

[Pipeline] sh

+ docker push prasanth0003/react\_app:latest

The push refers to repository [docker.io/prasanth0003/react\_app]

9edebf803b9a: Preparing

ed332502632e: Preparing

4221e53bbd7e: Preparing

0d853d50b128: Preparing

947e805a4ac7: Preparing

811a4dbbf4a5: Preparing

b8d7d1d22634: Preparing

e244aa659f61: Preparing

c56f134d3805: Preparing

d71eae0084c1: Preparing

08000c18d16d: Preparing

811a4dbbf4a5: Waiting

b8d7d1d22634: Waiting

e244aa659f61: Waiting

c56f134d3805: Waiting

08000c18d16d: Waiting

d71eae0084c1: Waiting

4221e53bbd7e: Layer already exists

ed332502632e: Layer already exists

0d853d50b128: Layer already exists

947e805a4ac7: Layer already exists

9edebf803b9a: Layer already exists

811a4dbbf4a5: Layer already exists

b8d7d1d22634: Layer already exists

d71eae0084c1: Layer already exists

e244aa659f61: Layer already exists

c56f134d3805: Layer already exists

08000c18d16d: Layer already exists

latest: digest: sha256:88f5f11392fa7eb48207b7ad8e596ee4bf201f5653868eccfff6e9f703aa72b9 size: 2614

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // withDockerRegistry

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // script

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Deploy to EKS)

[Pipeline] sh

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

Updated context arn:aws:eks:us-east-1:061039785879:cluster/trend-cluster in /var/lib/jenkins/.kube/config

[Pipeline] sh

+ kubectl apply -f deployment.yaml

deployment.apps/trend-app unchanged

[Pipeline] sh

+ kubectl apply -f service.yaml

service/trend-app-service unchanged

[Pipeline] sh

+ kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 172.20.0.1 <none> 443/TCP 6h42m

trend-app-service LoadBalancer 172.20.143.28 ab39bf9e9129d4c549e48c0e453cdb86-986344921.us-east-1.elb.amazonaws.com 3000:31636/TCP 5h39m

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

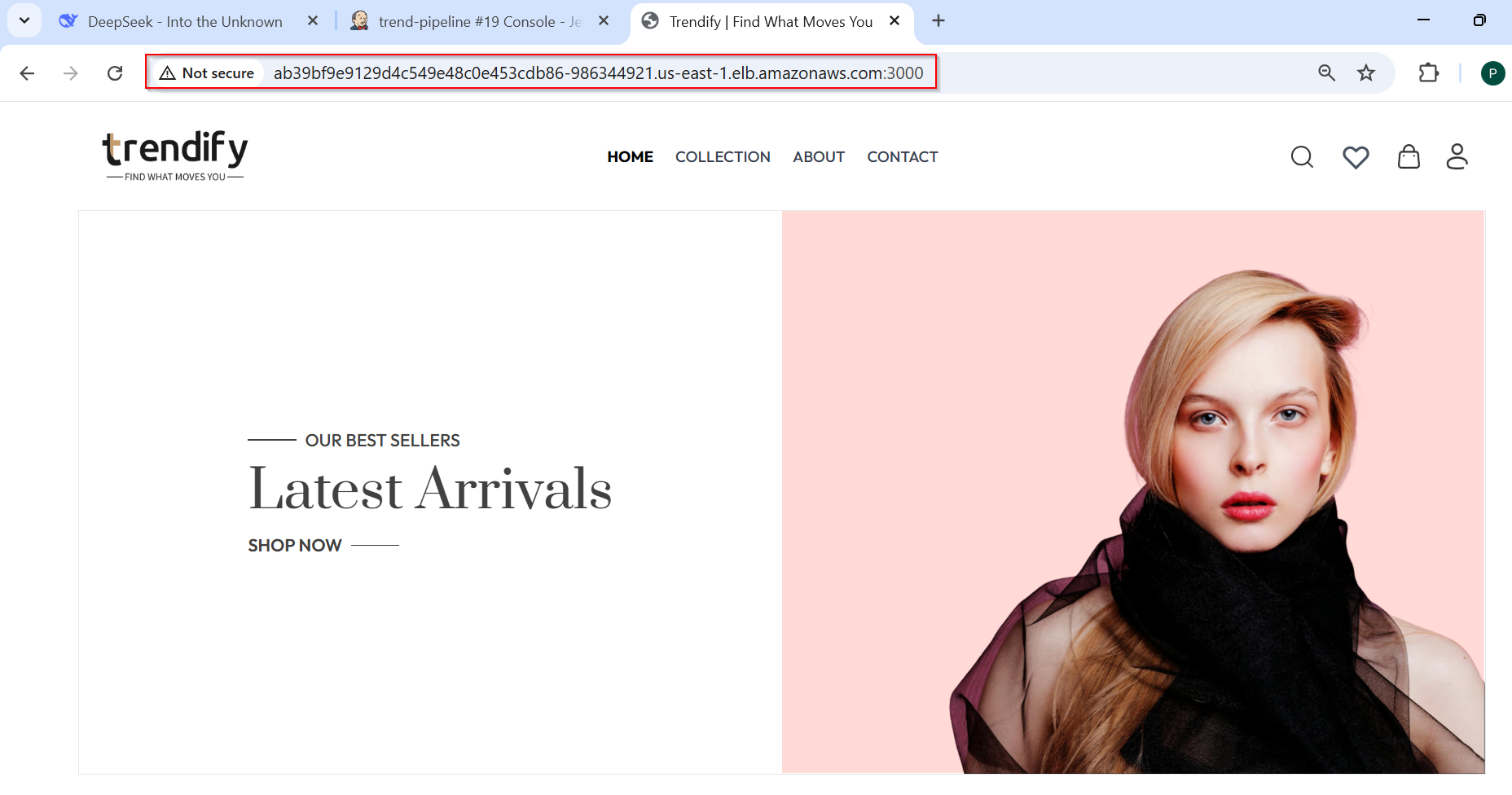
[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // node

[Pipeline] End of Pipeline

Finished: SUCCESS

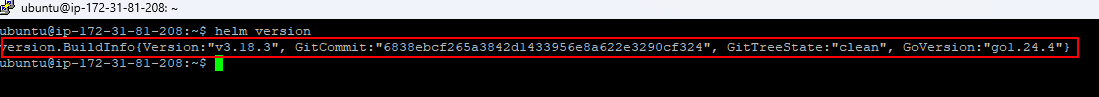
****

**Monitoring Setup on EKS (Prometheus + Grafana):**

**Install Helm on EC2**

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

helm version



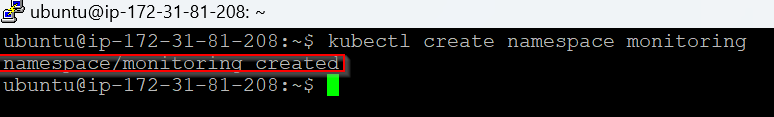
**Add Helm Repos for Prometheus and Grafana**

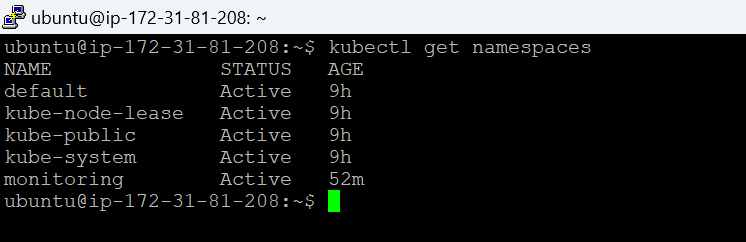
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

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

helm repo update

**kubectl create namespace monitoring**





**Helm Install for Prometheus and Grafana**

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

--namespace monitoring \

--create-namespace \

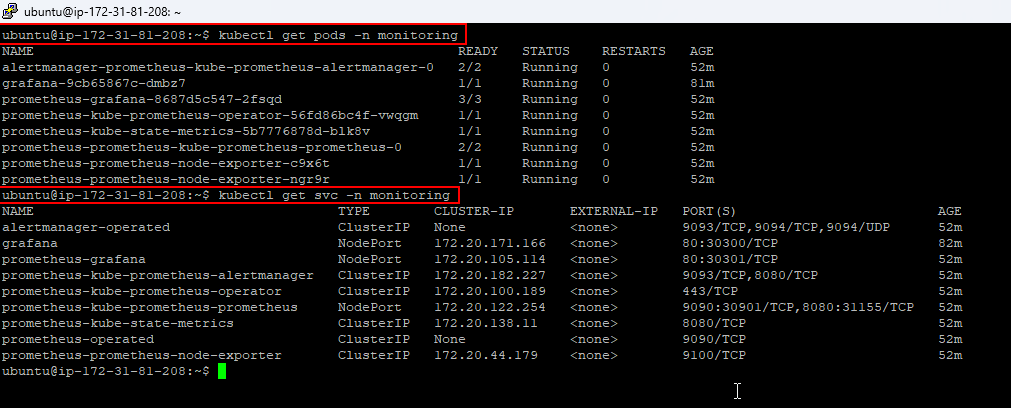
--set prometheus.service.type=NodePort \

--set prometheus.service.nodePort=30901 \

--set grafana.service.type=NodePort \

--set grafana.service.nodePort=30301 \

--set grafana.adminPassword='admin123'

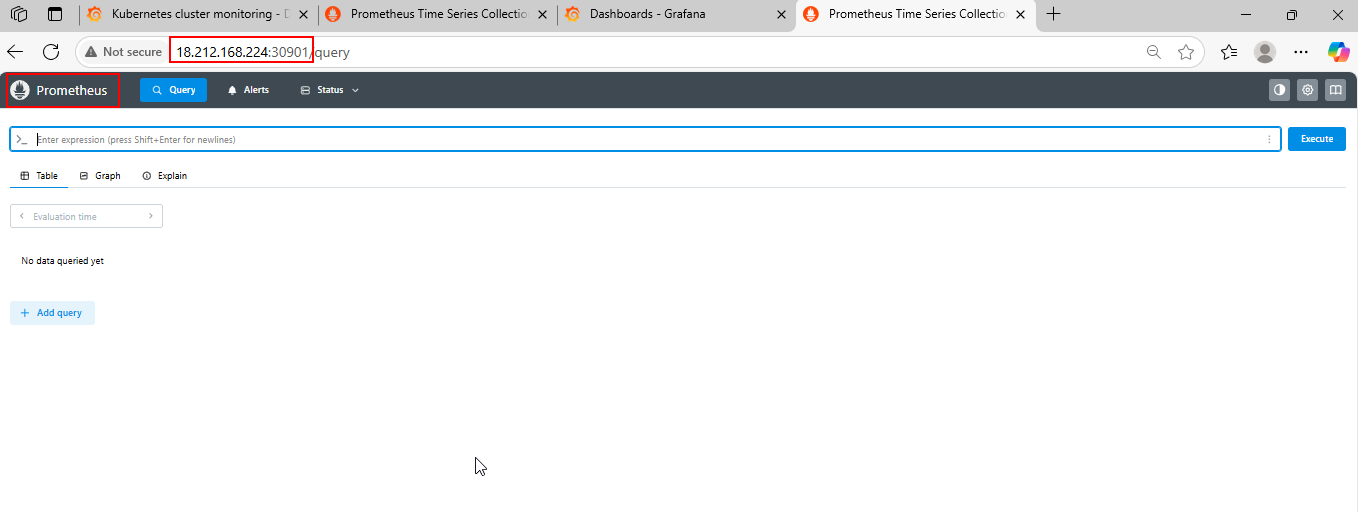


**URLs**

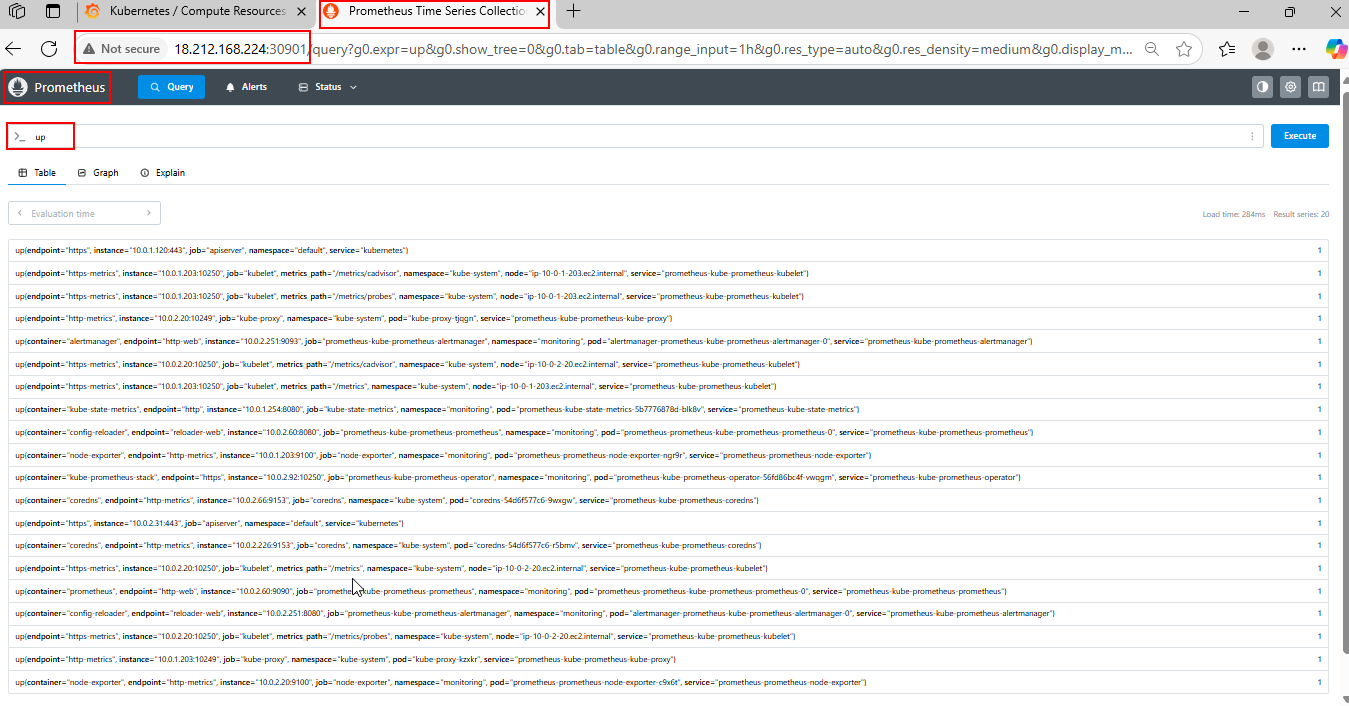
Prometheus - <http://18.212.168.224:30901>

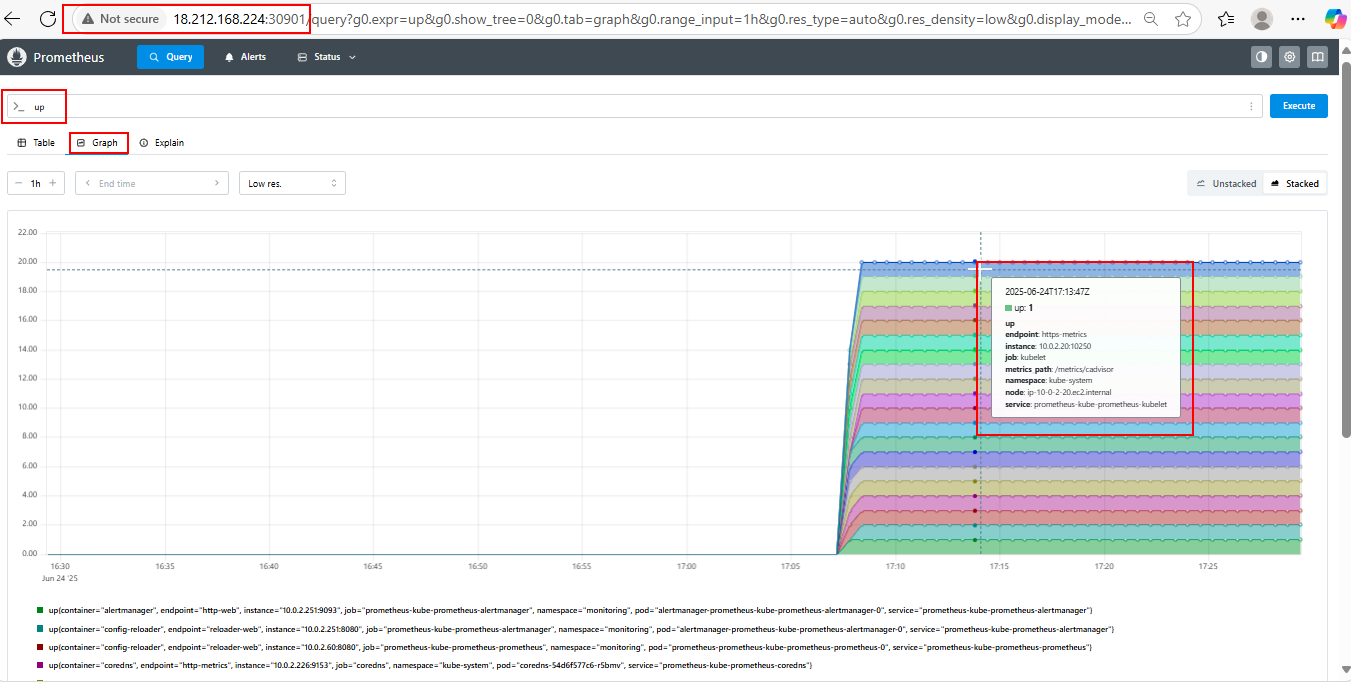
Grafana - <http://18.212.168.224:30301>

**Prometheus**

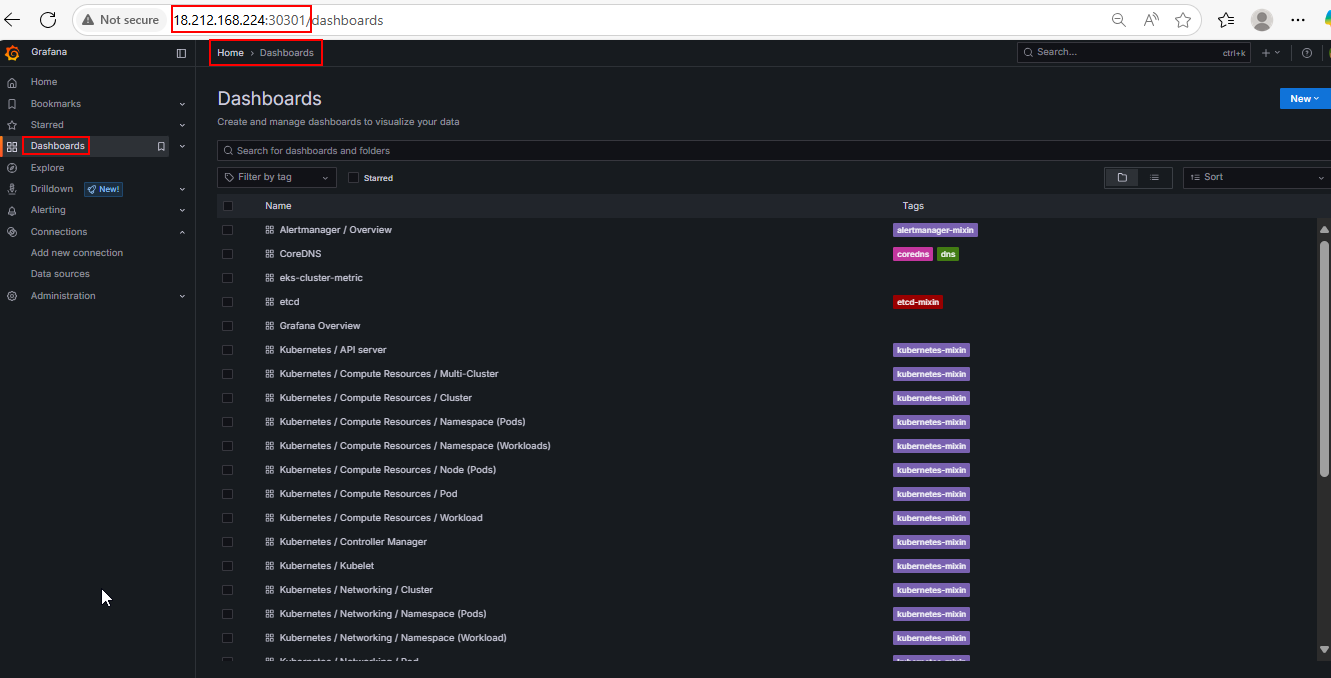


**Up status**

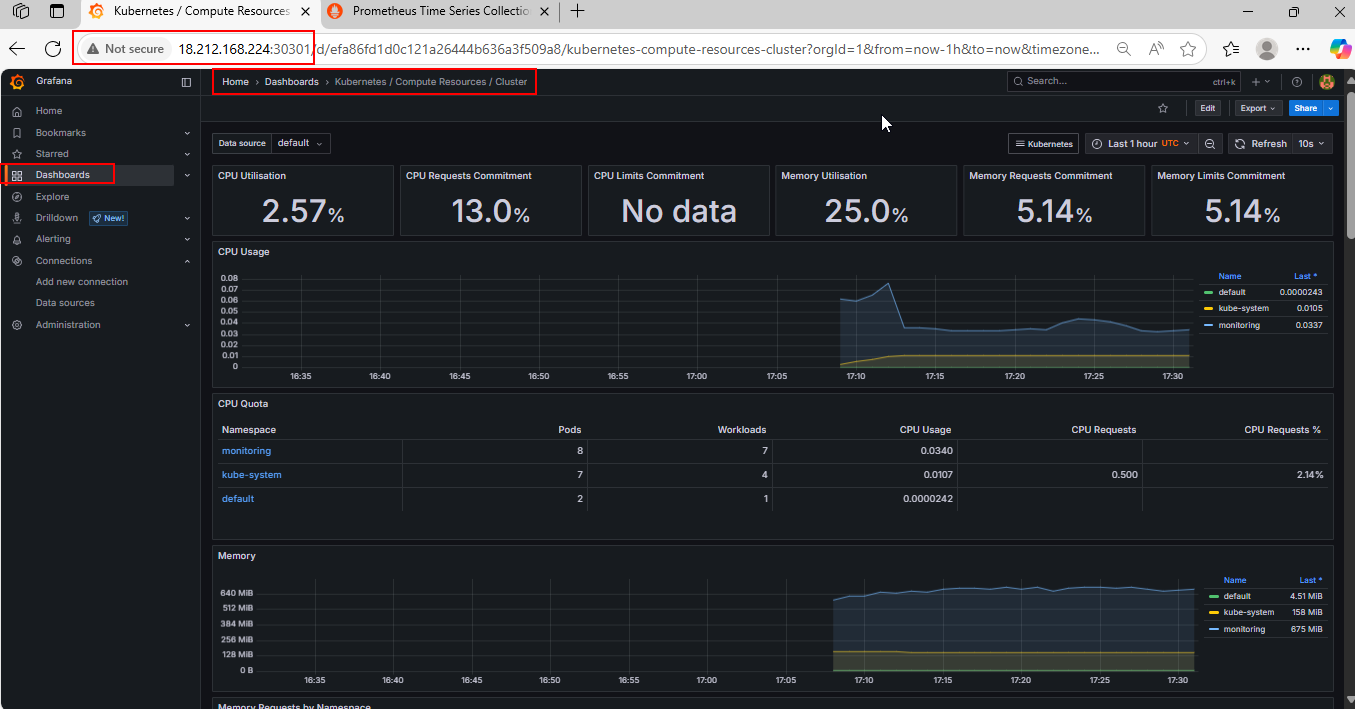




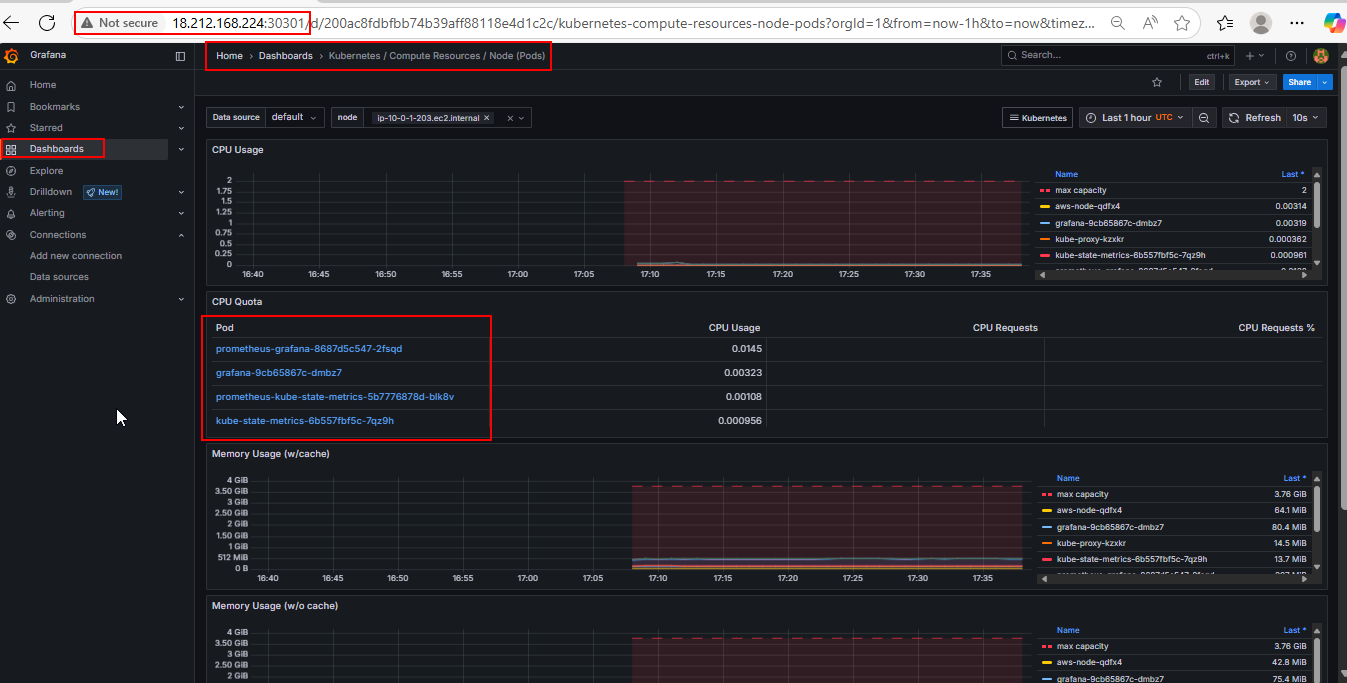
**Grafana:  
  
Dashboards**



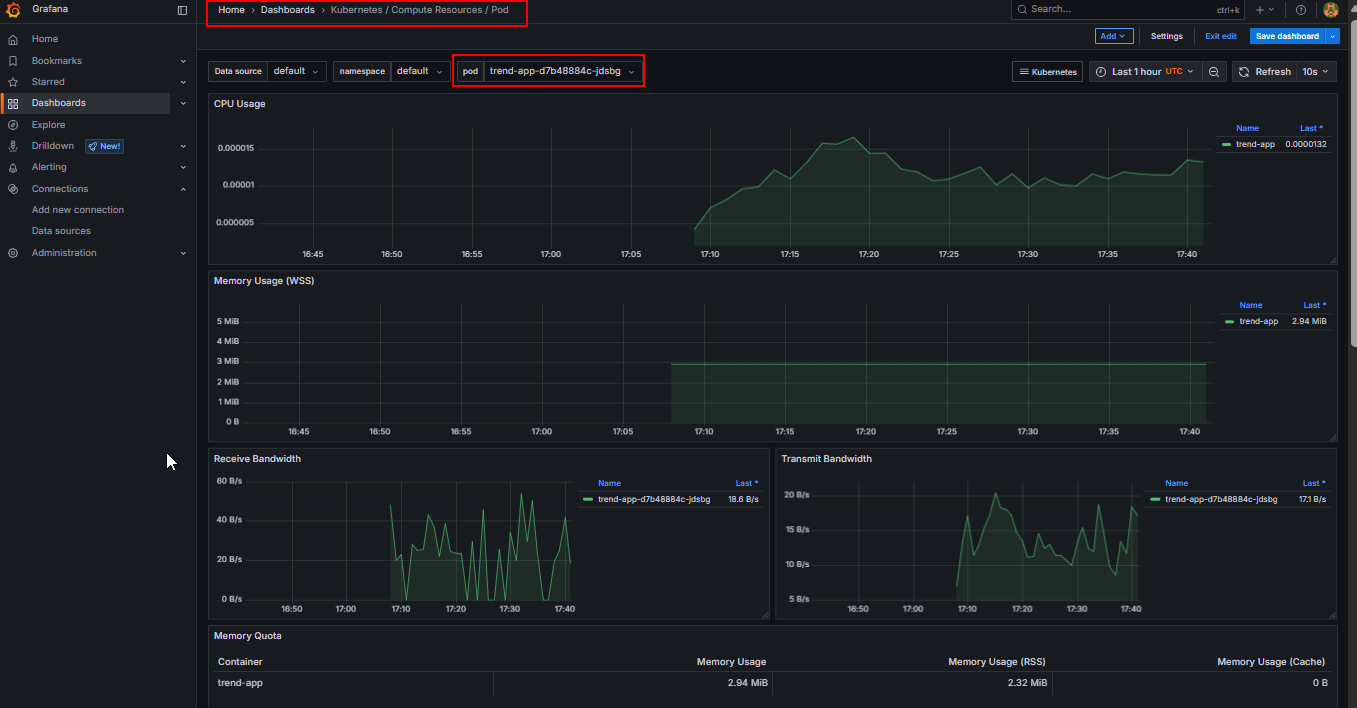
**Cluster:**



**Node (pods)**



**Pod**



**Cluster Networking:**

