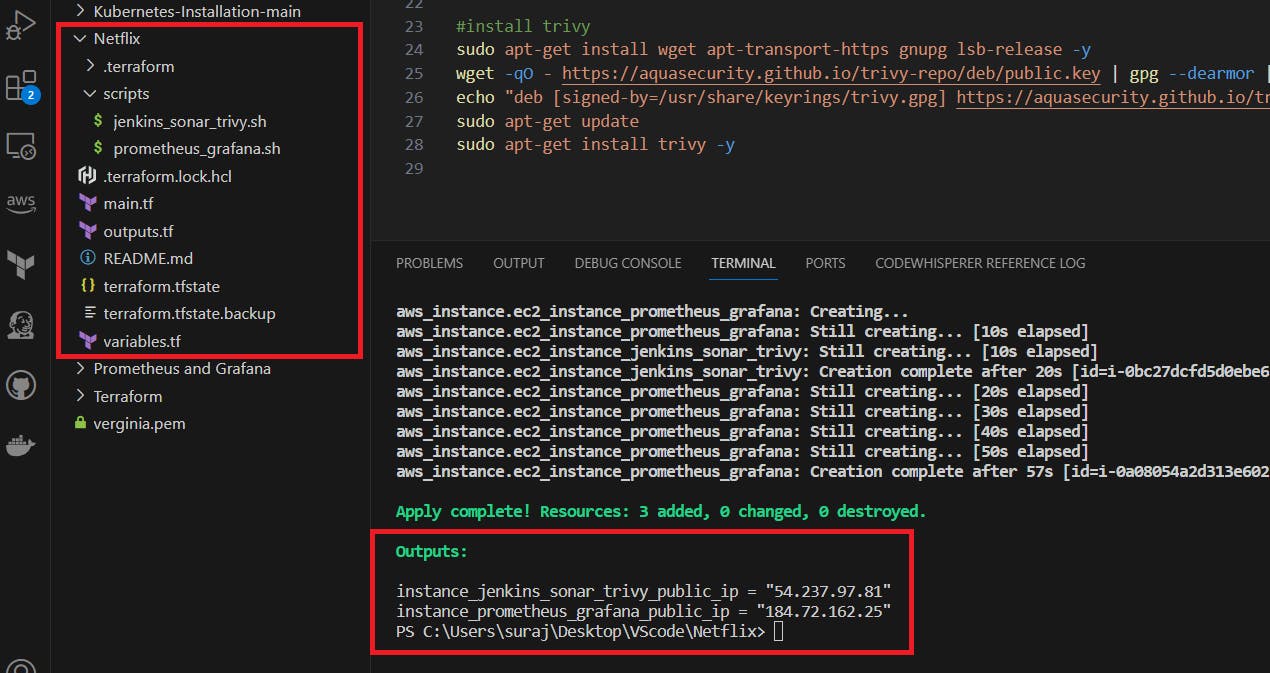
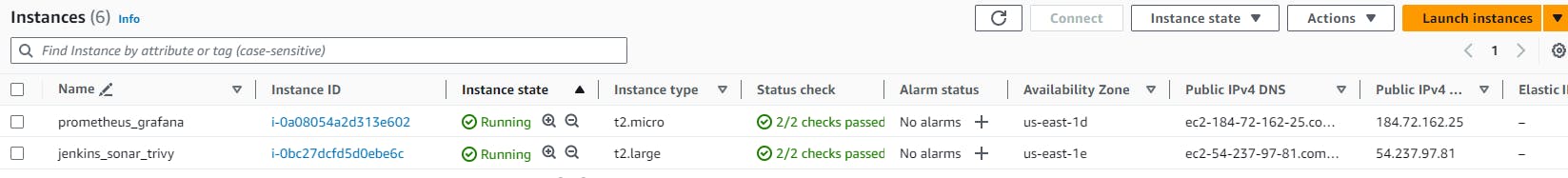
**Netflix Clone CI-CD with Terraform |Kubernetes| Monitoring | Email**

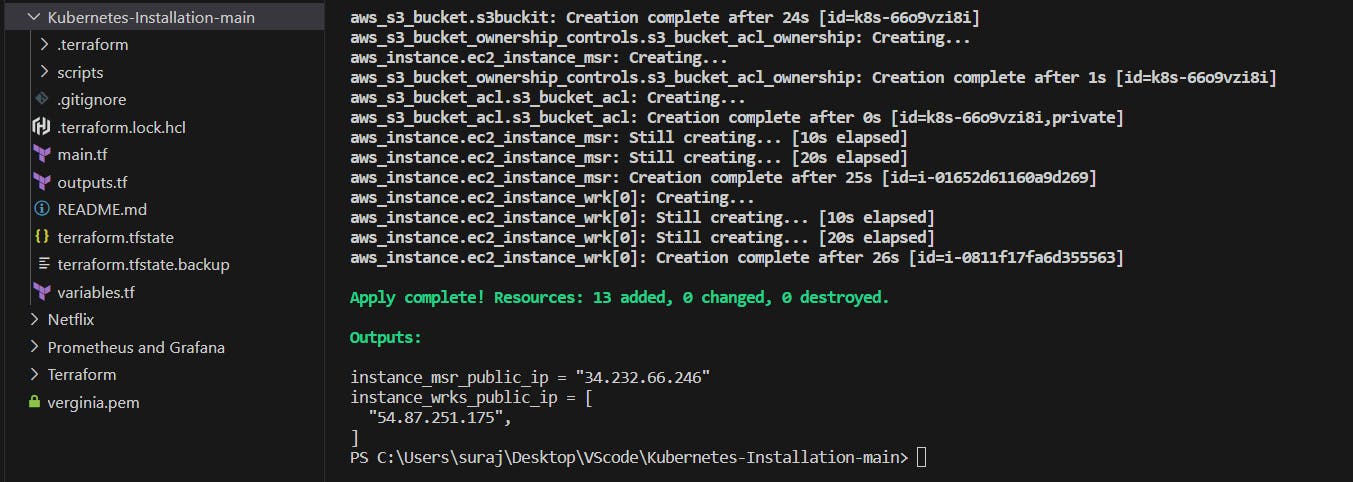
Hello there, buddies. We will set up a Netflix clone. We will be use Terraform to provision an EC2 instance with Kubernetes, Jenkins, Docker, SonarQube Container, Trivy, Prometheus & Grafana. Using Jenkins as a CICD tool, we deploy our application on a Docker container and a Kubernetes cluster, and we monitor Jenkins and Kubernetes metrics with Grafana, Prometheus, and Node exporter. I hope you found this informative blog useful.

Terraform to provision : Jenkins, Docker, SonarQube Container, Trivy, Prometheus & Grafana  
<https://github.com/surajpjoshi/Terraform_Netflix.git>

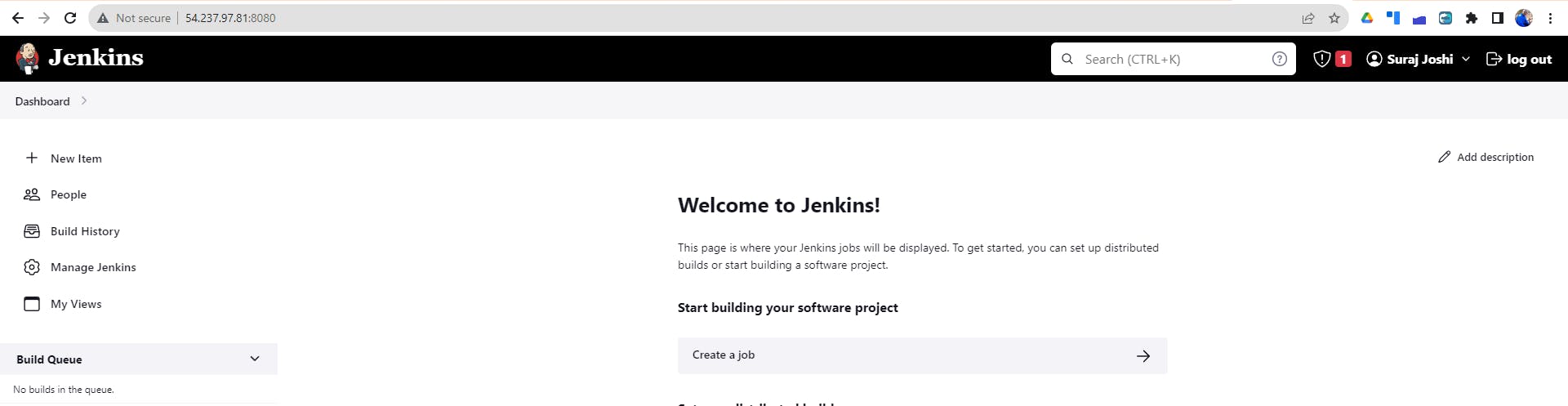




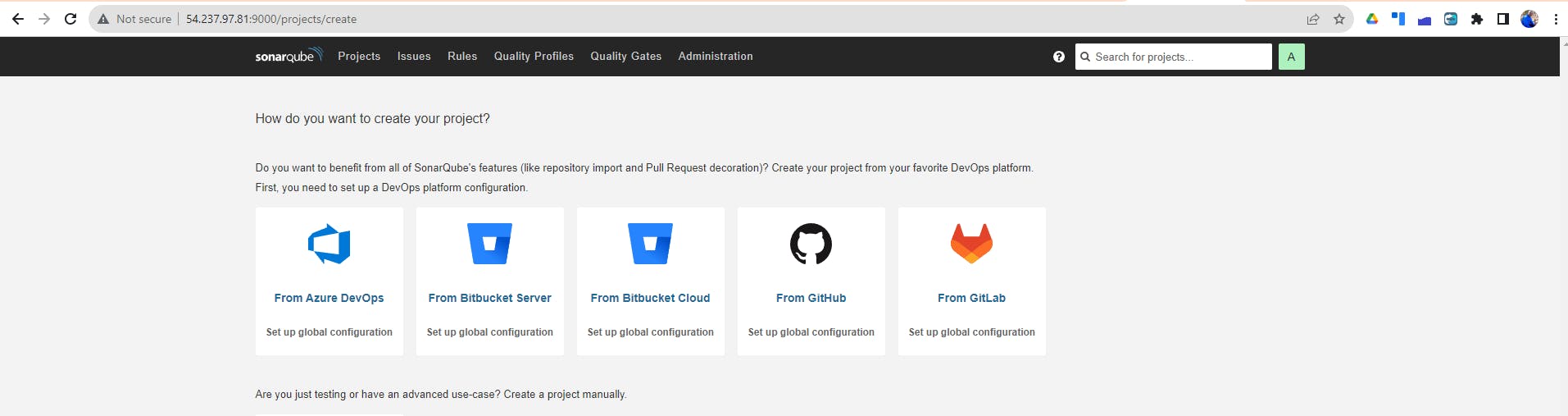
Terraform to provision : kubernetes



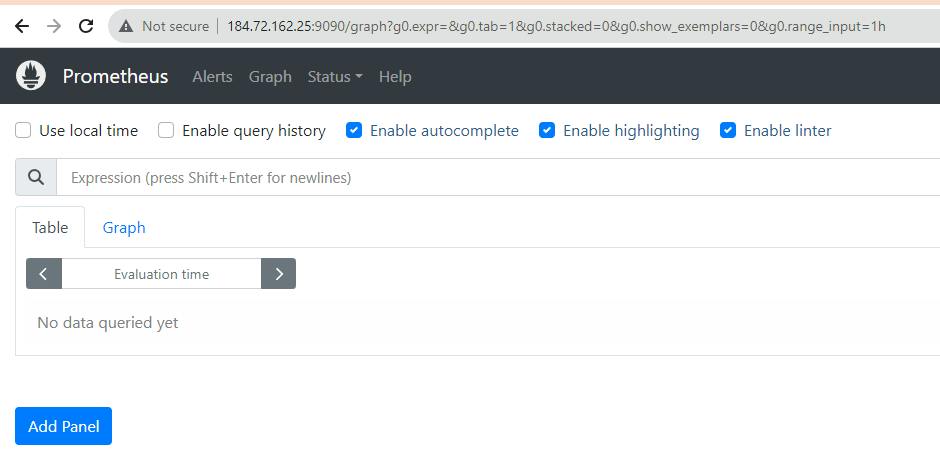
Jenkins : <http://54.237.97.81:8080/>



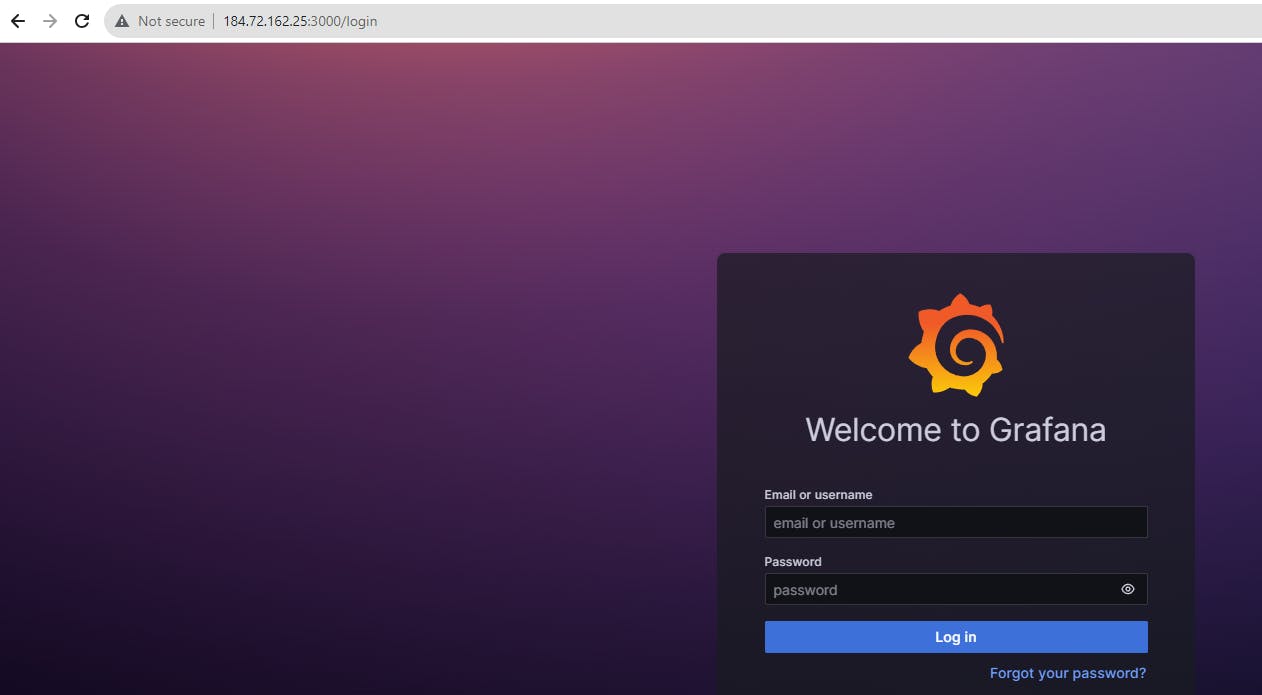
Sonarqube : [http://54.237.97.81:9000/](http://54.237.97.81:9000/projects)



Prometheus :



Grafana:



CICD Pipe Line:

COPY

COPY

pipeline{

agent any

tools{

jdk 'jdk17'

nodejs 'node16'

}

environment {

SCANNER\_HOME=tool 'sonar-scanner'

}

stages {

stage('clean workspace'){

steps{

cleanWs()

}

}

stage('Checkout from Git'){

steps{

git branch: 'main', url: 'https://github.com/surajpjoshi/Netflix-clone.git'

}

}

stage("Sonarqube Analysis "){

steps{

withSonarQubeEnv('sonar-server') {

sh ''' $SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=Netflix \

-Dsonar.projectKey=Netflix '''

}

}

}

stage("quality gate"){

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

}

}

}

stage('Install Dependencies') {

steps {

sh "npm install"

}

}

stage('OWASP FS SCAN') {

steps {

dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'

dependencyCheckPublisher pattern: '\*\*/dependency-check-report.xml'

}

}

stage('TRIVY FS SCAN') {

steps {

sh "trivy fs . > trivyfs.txt"

}

}

stage("Docker Build & Push"){

steps{

script{

withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){

sh "docker build --build-arg TMDB\_V3\_API\_KEY=2cffc1b7f9a2451e1c357eeb211c45d2 -t netflix ."

sh "docker tag netflix surajpjoshi/netflix:latest "

sh "docker push surajpjoshi/netflix:latest "

}

}

}

}

stage("TRIVY"){

steps{

sh "trivy image surajpjoshi/netflix:latest > trivyimage.txt"

}

}

stage('Deploy to kubernets'){

steps{

script{

dir('Kubernetes') {

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

sh 'kubectl apply -f deployment.yml'

sh 'kubectl apply -f service.yml'

}

}

}

}

}

}

post {

always {

emailext attachLog: true,

subject: "'${currentBuild.result}'",

body: "Project: ${env.JOB\_NAME}<br/>" +

"Build Number: ${env.BUILD\_NUMBER}<br/>" +

"URL: ${env.BUILD\_URL}<br/>",

to: 'surajpjoshi@gmail.com',

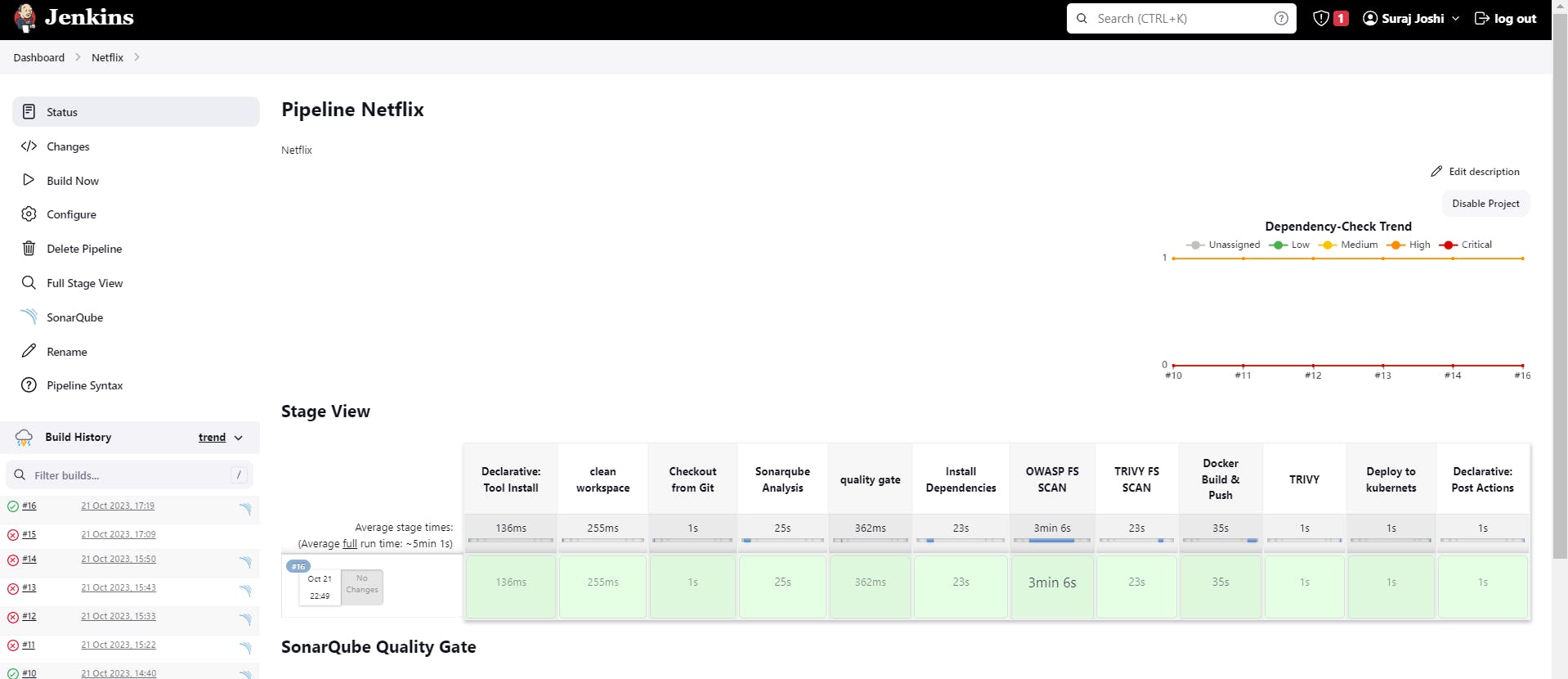
attachmentsPattern: 'trivyfs.txt,trivyimage.txt'

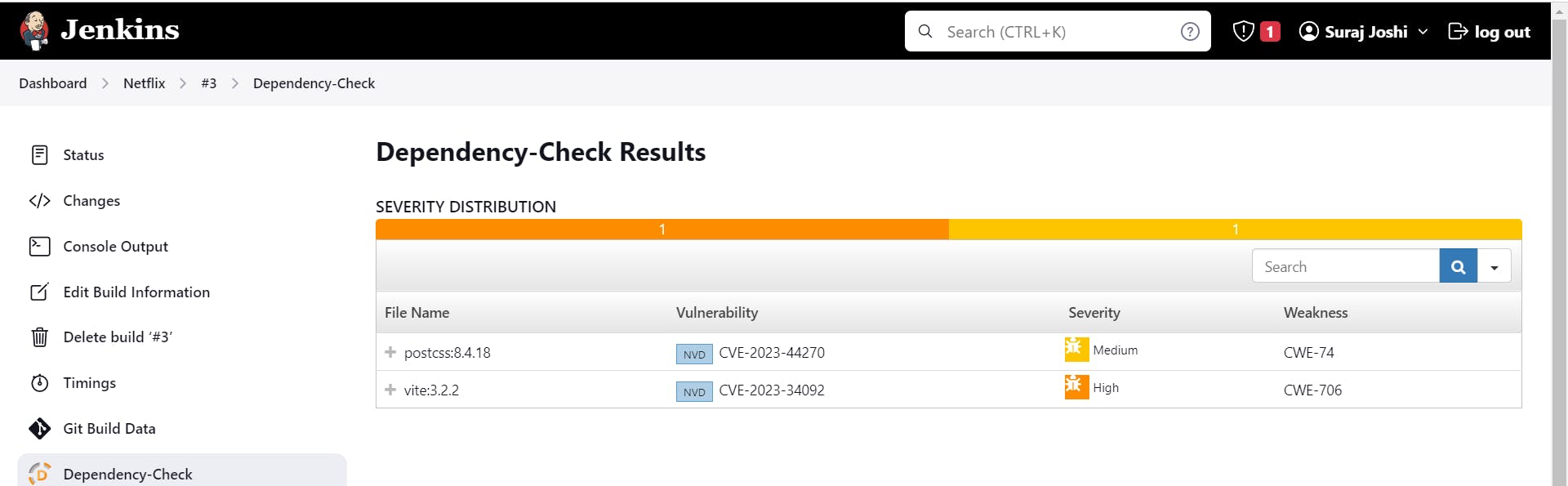
}

}

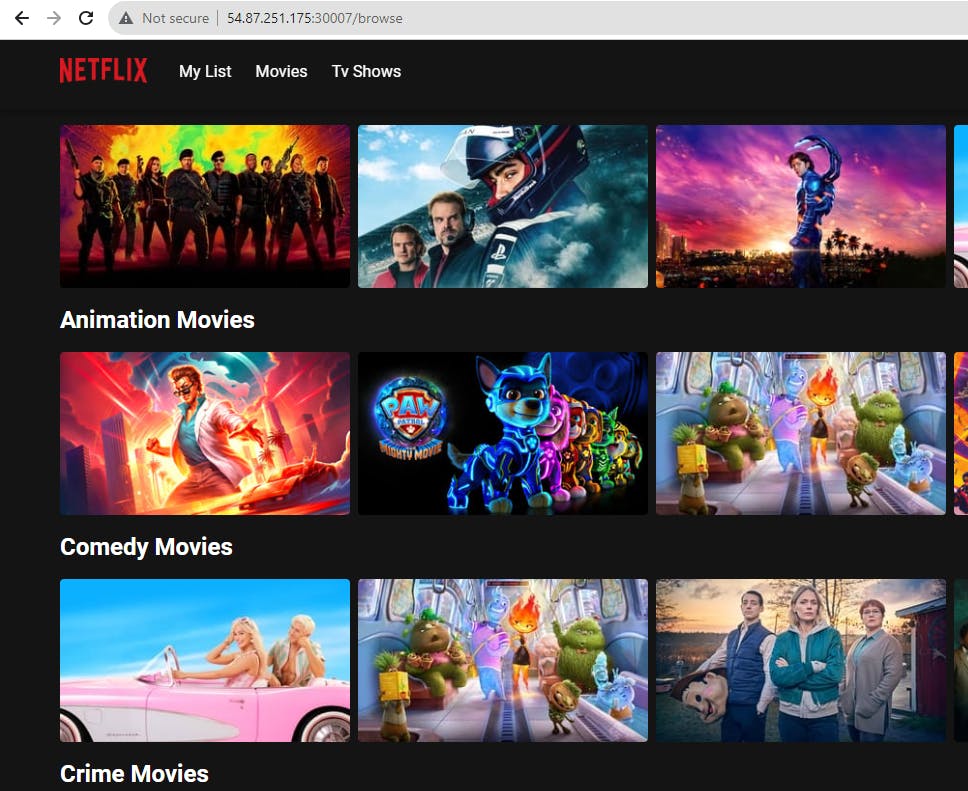
}

Build Pipe Line:

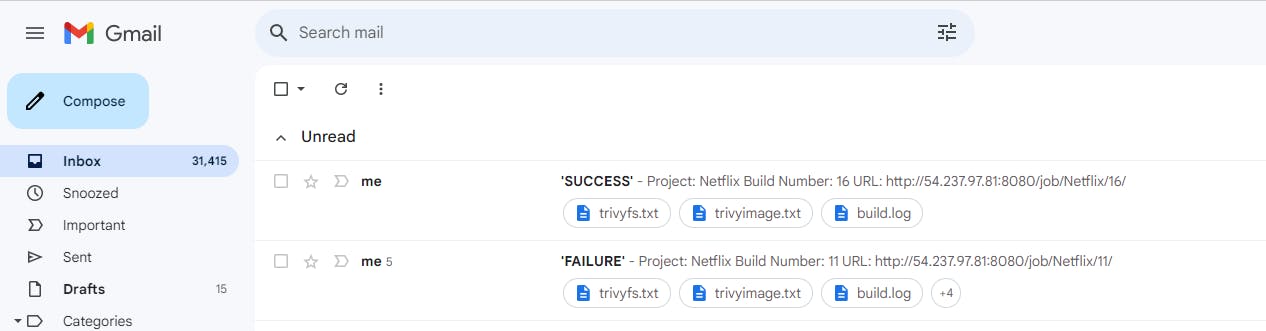


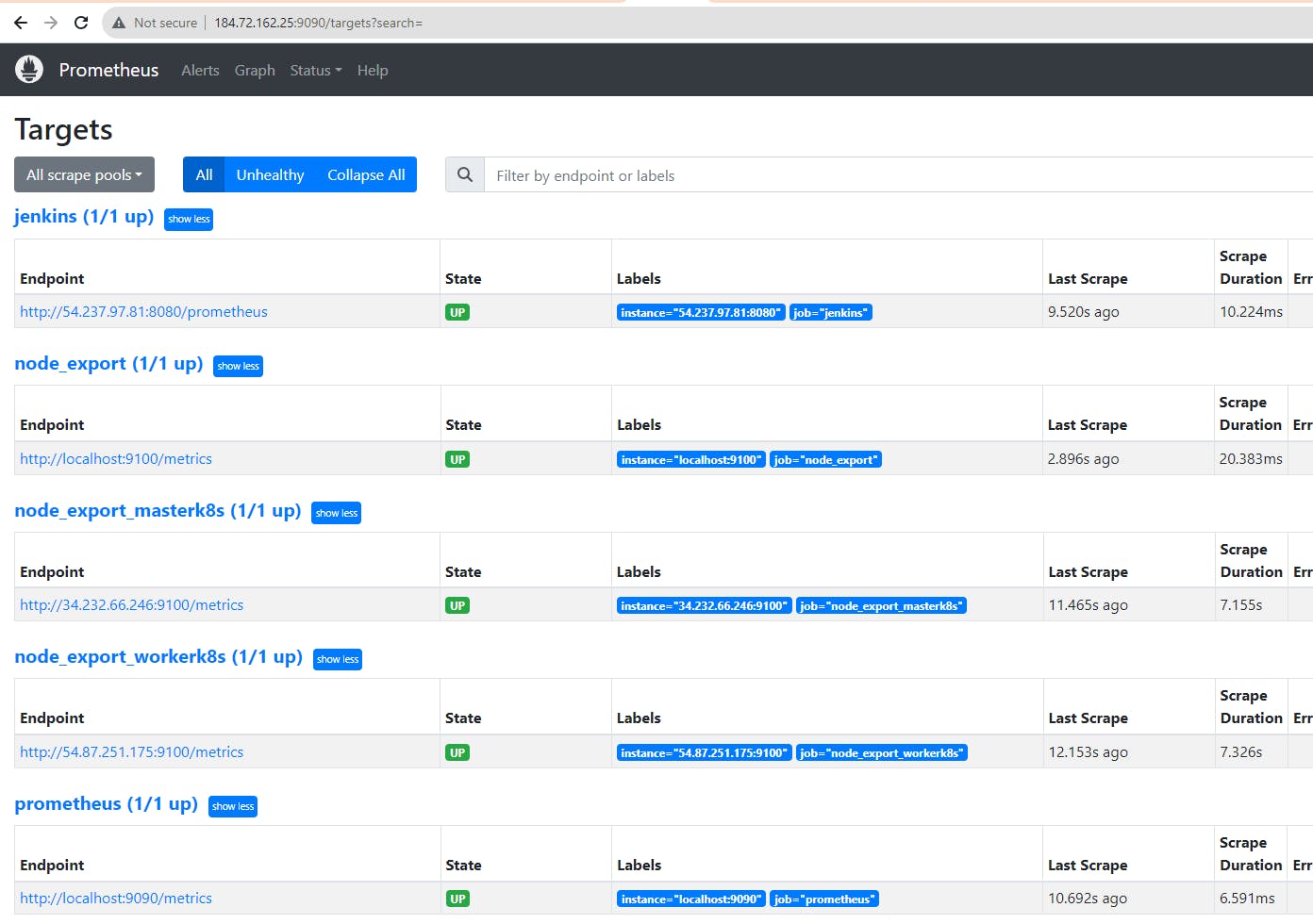


Finally : [http://54.87.251.175:30007](http://54.87.251.175:30007/browse)

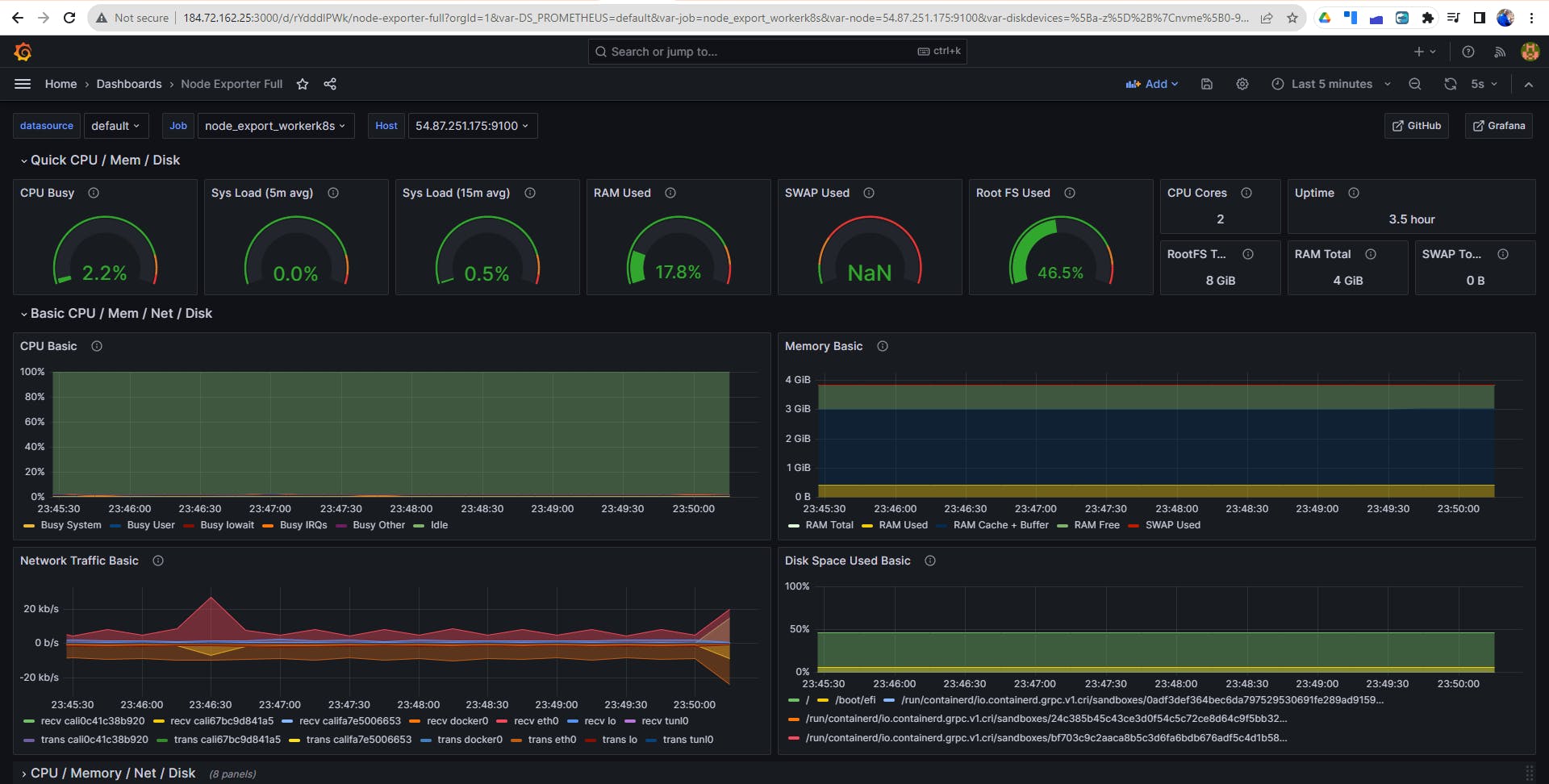


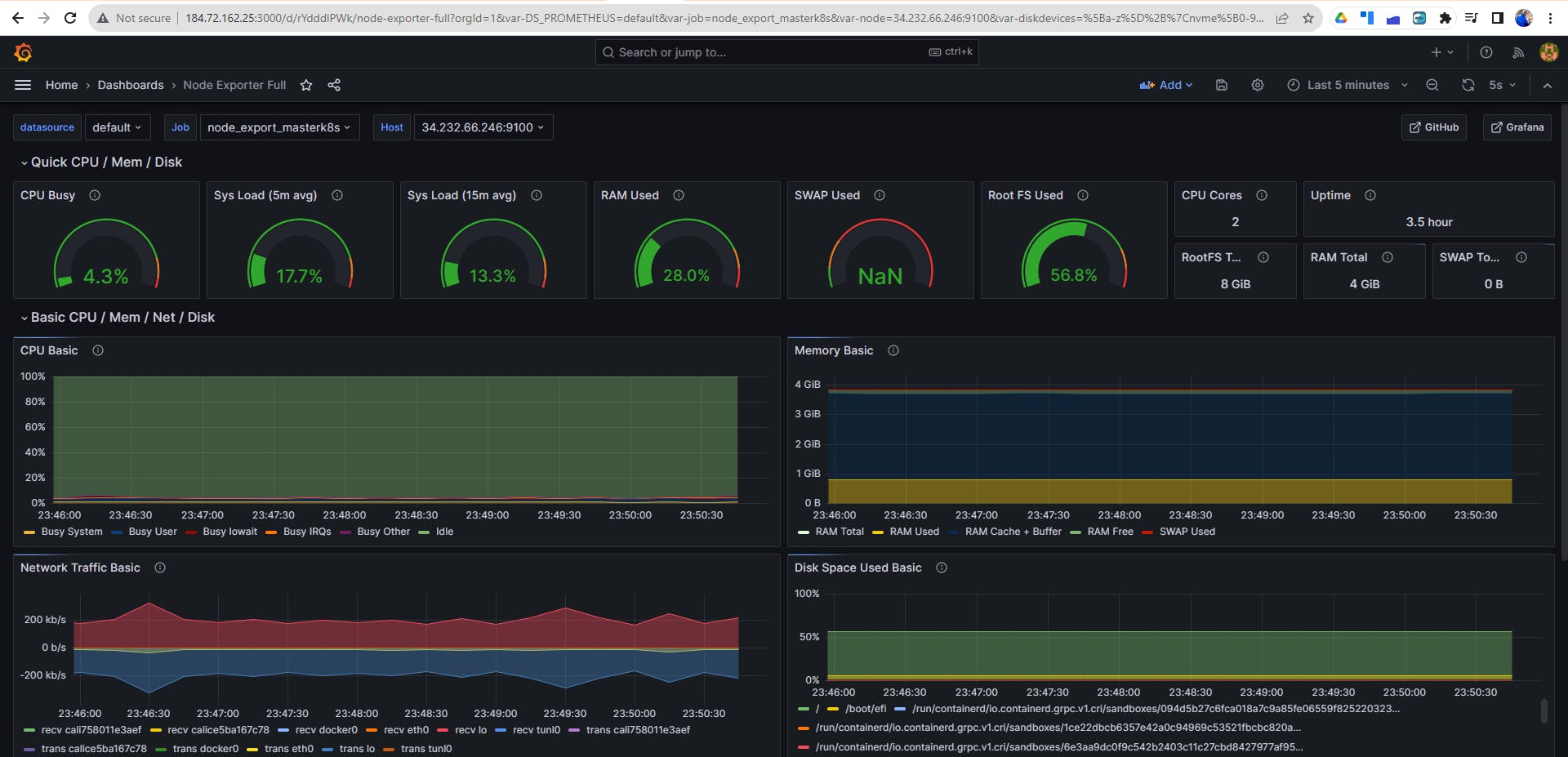
Email Notification :





Monitoring master & node on Grafana dashboard :





Finally, after taking a deep dive into the realm of devops and learning a lot..

COPY

COPY

terraform destroy

