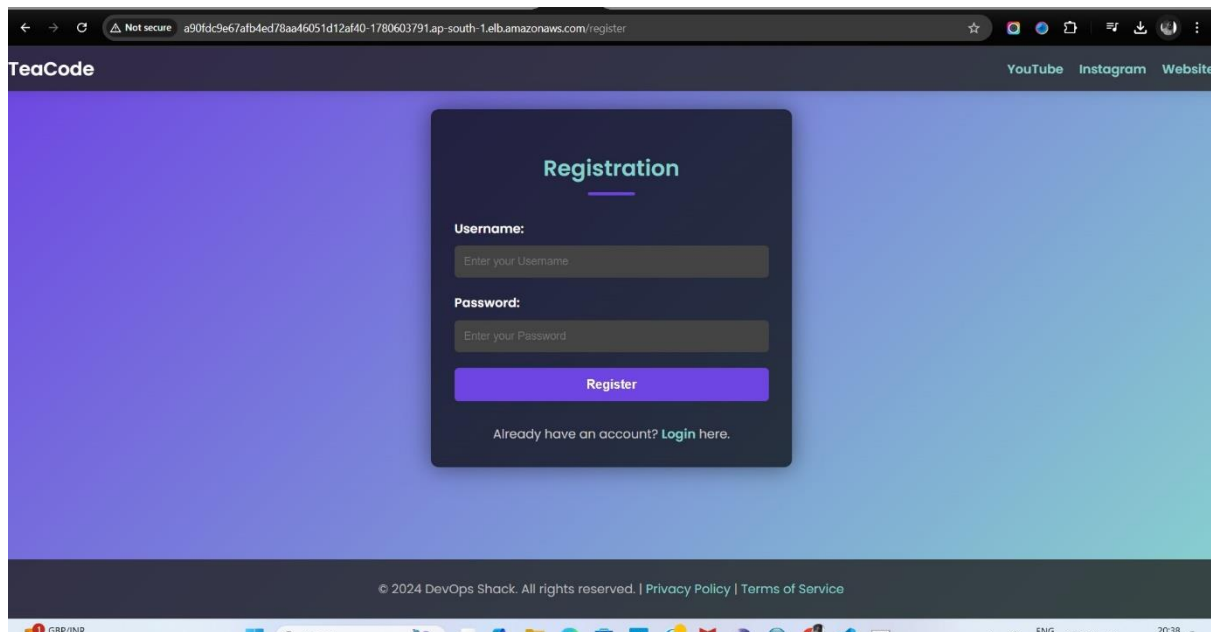


Production Level CI/CD Pipeline Project | CI/CD DevOps Project



Divya Satpute

@TeaCode1122

Divya satpute

What we are doing ????

1. Setup Repo
2. Set-Up Required Servers[Jenkins, SonarQube, Nexus, Monitoring Tools]
3. Configure Tools
4. Create The Pipelines & Create EKS Clusters
5. Trigger The Pipeline To Deploy the Application
6. Assign a Custom domain to the deployed application
7. Monitor The Application

Prerequisites

Step 1

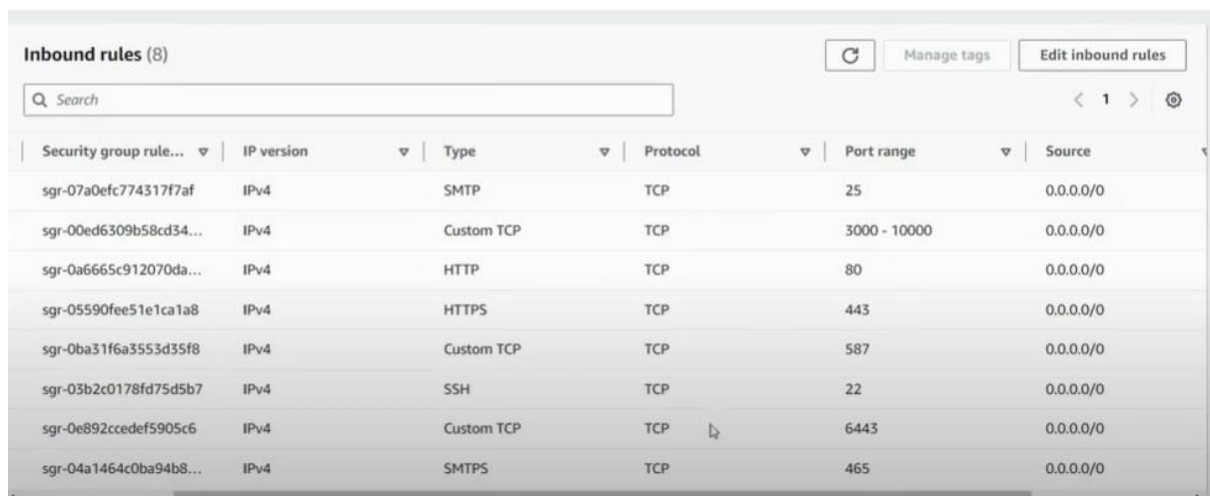
Setting up EKS Cluster Using Terraform

AWS Console launch server for terraform

t2 medium

40 storage

open this Ports inbound rule on security group

A screenshot of the AWS Management Console showing the 'Inbound rules (8)' for a security group. The table lists eight rules with columns for Security group rule ID, IP version, Type, Protocol, Port range, and Source. The rules include SMTP, Custom TCP, HTTP, HTTPS, Custom TCP, SSH, Custom TCP, and SMTPS.

Security group rule...	IP version	Type	Protocol	Port range	Source
sgr-07a0efc774317f7af	IPv4	SMTP	TCP	25	0.0.0.0/0
sgr-00ed6309b58cd34...	IPv4	Custom TCP	TCP	3000 - 10000	0.0.0.0/0
sgr-0a6665c912070da...	IPv4	HTTP	TCP	80	0.0.0.0/0
sgr-05590fee51e1ca1a8	IPv4	HTTPS	TCP	443	0.0.0.0/0
sgr-0ba31f6a3553d35f8	IPv4	Custom TCP	TCP	587	0.0.0.0/0
sgr-03b2c0178fd75d5b7	IPv4	SSH	TCP	22	0.0.0.0/0
sgr-0e892ccedef5905c6	IPv4	Custom TCP	TCP	6443	0.0.0.0/0
sgr-04a1464c0ba94b8...	IPv4	SMTPS	TCP	465	0.0.0.0/0

update repo

```
$sudo apt update -y
```

Install AWS CLI

```
$curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
sudo apt install unzip
```

```
unzip awscliv2.zip
```

```
sudo ./aws/install
```

@TeaCode1122

Divya satpute

AWS Configure Provide Access key and Secret key on Aws Console

aws configure

```
ubuntu@ip-172-31-29-133:~$ aws configure
AWS Access Key ID [None]: AKIA2UC3AX6WYCNQAUGX
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]:
```

Install Kubectl

```
$curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
```

```
$chmod +x ./kubectl
```

```
$sudo mv ./kubectl /usr/local/bin
```

```
$kubectl version --short --client
```

Installation of Terraform

```
$sudo snap install terraform --classic
```

terraform --version

```
ubuntu@ip-172-31-29-133:~$ terraform --version
Terraform v1.9.5
on linux_amd64
```

clone the Repo for EKS Terraform Script

```
$git clone https://github.com/divyasatpute/FullStack-Blogging-App.git
```

change directory

```
$cd FullStack-Blogging-App/
```

change directory

```
$cd EKS_Terraform/
```

In Variables.tf file you just need to change Your key name

AND in main.tf file you just need to change region and availability zone as per your requirement

@TeaCode1122

Divya satpute

FullStack-Blogging-App / EKS_Terraform / variables.tf

diviyasatpute Update variables.tf

Code Blame 5 lines (5 loc) · 144 Bytes Code 55% faster with GitHub Copilot

```
1 variable "ssh_key_name" {
2     description = "The name of the SSH key pair to use for instances"
3     type        = string
4     default     = "DevOps"
5 }
```

Now terraform initialization

\$terraform init

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

\$terraform plan

\$terraform apply --auto-approve

Apply complete! Resources: 17 added, 0 changed, 0 destroyed.

Outputs:

```
cluster_id = "devopsshack-cluster"
node_group_id = "devopsshack-cluster:devopsshack-node-group"
subnet_ids = [
    "subnet-0fe06ccf41121492c",
    "subnet-0f57af72dfddb1f0e",
]
vpc_id = "vpc-0e79f529bc0c8518e"
```

In Order to communicate with aws eks cluster we need to update our kubeconfig file

\$aws eks --region ap-south-1 update-kubeconfig --name devopsshack-cluster

```
Added new context arn:aws:eks:ap-south-1:73055530221:cluster/devopsshack-cluster to /home/ubuntu/.kube/config
ubuntu@ip-172-31-41-223:~/FullStack-Blogging-App/EKS_Terraform$ kubectl get no
NAME                                STATUS    ROLES    AGE    VERSION
ip-10-0-0-60.ap-south-1.compute.internal Ready    <none>   73s    v1.30.4-eks-a737599
ip-10-0-0-91.ap-south-1.compute.internal Ready    <none>   75s    v1.30.4-eks-a737599
ip-10-0-1-144.ap-south-1.compute.internal Ready    <none>   76s    v1.30.4-eks-a737599
```

Step 2

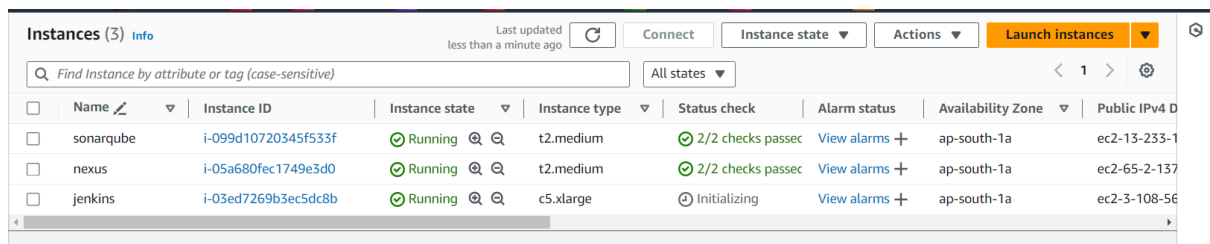
40 GB Storage

Launch 1 EC2 Machine one for Jenkins

t2.large

40 GB storage

@TeaCode1122



The screenshot shows the AWS Management Console 'Instances' page. At the top, there are buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. Below these is a search bar and a table of instances. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 D. Three instances are listed: 'sonarqube' (Running, t2.medium), 'nexus' (Running, t2.medium), and 'jenkins' (Initializing, c5.xlarge).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
sonarqube	i-099d10720345f533f	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1a	ec2-13-233-1
nexus	i-05a680fec1749e3d0	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1a	ec2-65-2-137
jenkins	i-03ed7269b3ec5dc8b	Running	c5.xlarge	Initializing	View alarms	ap-south-1a	ec2-3-108-56

Connect them with using gitbash

Installation Jenkins

step 1

Install java (latest stable version)

```
$sudo apt install openjdk-17-jre-headless -y
```

Install Jenkins

```
$vi 1.sh
```

Paste the all command in 1.sh file

```
$sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
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-get update
```

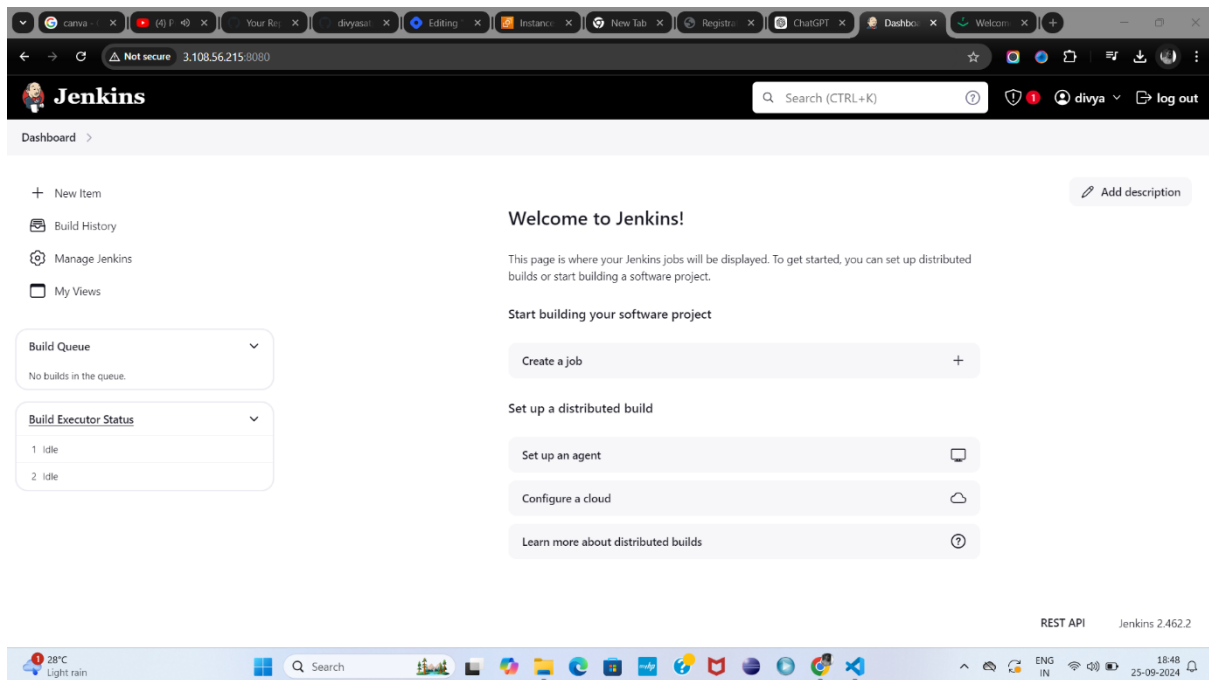
```
sudo apt-get install jenkins -y
```

Change the permission

```
$sudo chmod +x 1.sh
```

Run the file

```
$/1.sh
```



Installation docker on Jenkins machine

Install docker

```
$sudo apt install docker.io -y
```

change permission

```
$sudo chmod 666 /var/run/docker.sock
```

Installation Trivy on Jenkins machine

```
$sudo apt-get install wget apt-transport-https gnupg lsb-release
```

```
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -
```

```
echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | sudo tee -a /etc/apt/sources.list.d/trivy.list
```

```
sudo apt-get update
```

```
sudo apt-get install trivy -y
```

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@jenkins:~$ trivy version
Version: 0.55.2
ubuntu@jenkins:~$
```

Installation kubectl on Jenkins machine

```
$curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
```

```
chmod +x ./kubectl
```

@TeaCode1122

Divya satpute

```
sudo mv ./kubectl /usr/local/bin
```

```
kubectl version --short --client
```

Installation Nexus as a docker container

update machine

```
$sudo apt update -y
```

Install docker

```
$sudo apt install docker.io -y
```

Create container

```
$sudo docker run -d -p 8081:8081 sonatype/nexus3
```

Access your Nexus On Browser http://PUBLIC_IP:8081/

our Nexus up and running but password is stored inside the container so for that we need to go inside the container

```
$sudo docker exec -it 629f2dda1a74 /bin/bash
```

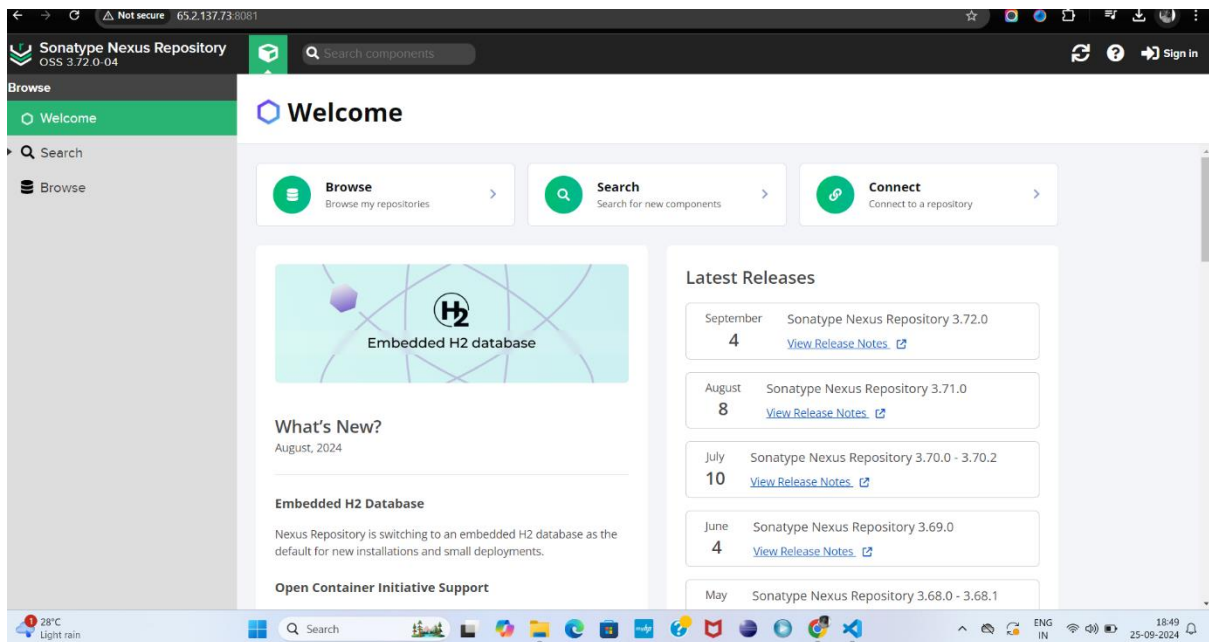
```
$cd sonatype-work/nexus3/
```

```
$cat admin.password
```

here you can got password

```
ubuntu@nexus:~$ sudo docker exec -it 629f2dda1a74 /bin/bash
bash-4.4$ cd sonatype-work/nexus3/
bash-4.4$ ls
admin.password  cache  elasticsearch  generated-bundles  javaprefs  keystores  log  restore-from-backup
blobs           db     etc            instances          karaf.pid  lock       port  tmp
bash-4.4$ cat admin.password
15197a44-d60e-430a-9737-4869241cd053bash-4.4$
```

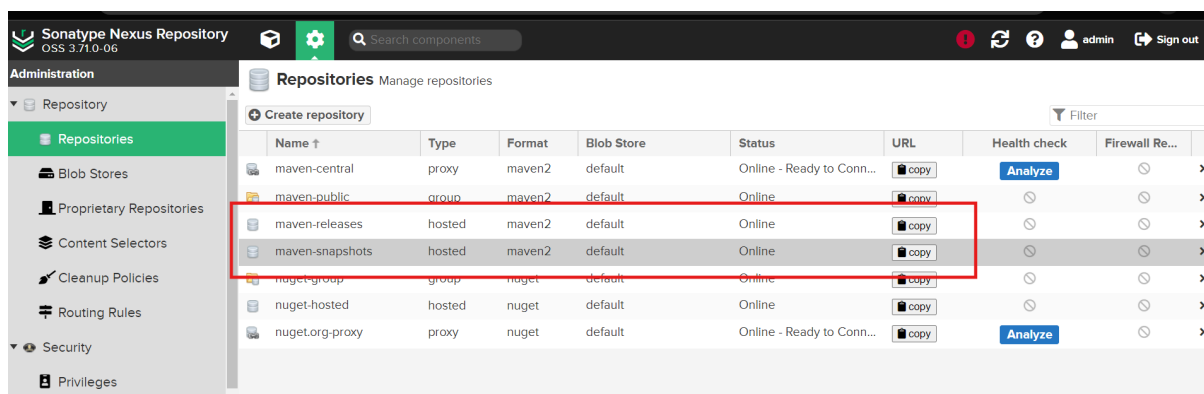
Now You Can See Our Nexus also working fine and able to sign in



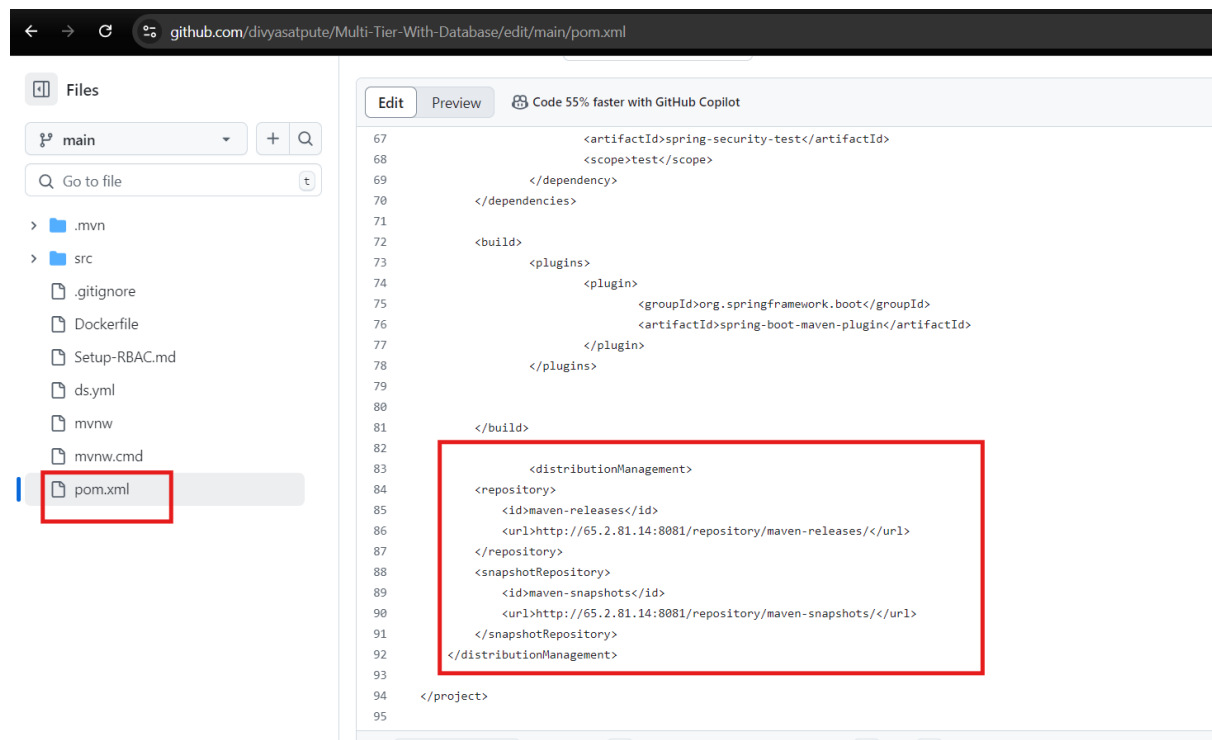
Nexus Configuration

Go to nexus dashboard --> click on settings ----> click on repositories

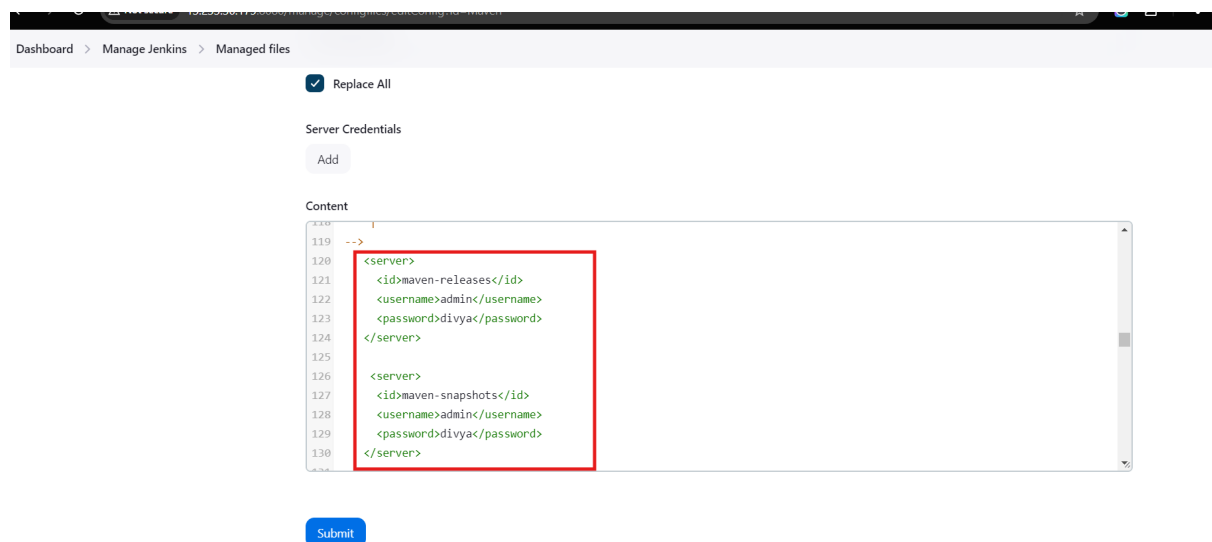
copy the Maven-releases URL and Maven snapshot URL and paste it on POX.XML file



Divya satpute



for credentials go to Jenkins Dashboard --->click on manage Jenkins---> Managed files---> click on Add new Config--->Global Maven settings.xml--->provide id "anything"---> click on next



Installation SonarQube as a docker container

update machine

\$sudo apt update -y

Install docker

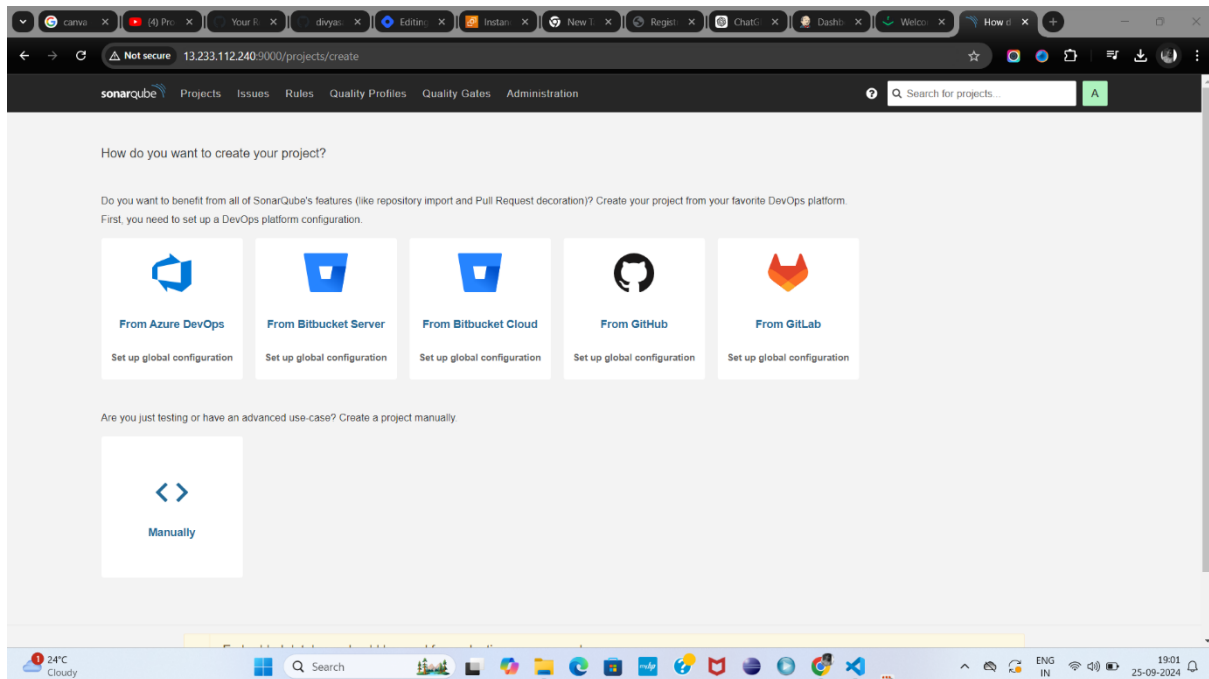
\$sudo apt install docker.io -y

Create container

@TeaCode1122

Divya satpute

\$sudo docker run -it -p 9000:9000 sonarqube:lts-community



Configuration on Jenkins

Installation Plugins

SonarQube Scanner

Config File Provider

Maven Integration

Pipeline Maven Integration

Kubernetes

Kubernetes Client API

Kubernetes Credentials

Kubernetes CLI

Kubernetes Credentials Provider

Docker Pipeline

Docker Commons

Docker

Eclipse Temurin installer

Pipeline: Stage View

@TeaCode1122

Configuration System

Sonar Scanner

Configuration tools

Go to Manage Jenkins ----> tools

add SonarQube Scanner

add Maven

Add Docker

Dashboard > Manage Jenkins > Tools

Docker installations ^ / Edited

Add Docker

≡ Docker

Name

docker

☒ Install automatically ?

≡ Download from docker.com

Docker version ?

latest


Add Installer ▾

Add Docker

Save

Apply

Dashboard > Manage Jenkins > Tools

Maven installations ^  Edited

Add Maven

≡

Maven

Name

maven3

☒ Install automatically ?

≡

Install from Apache

Version

3.9.9

Add Installer ▾

Add Maven


Save

Apply

@TeaCode1122

Dashboard > Manage Jenkins > Tools

SonarQube Scanner installations

SonarQube Scanner installations ^  Edited

Add SonarQube Scanner

≡ SonarQube Scanner

Name

sonar-scanner

☒ Install automatically ?

≡ Install from Maven Central

Version

SonarQube Scanner 6.2.0.4584

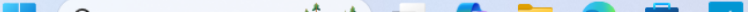
Add Installer ▾

Add SonarQube Scanner

Save

Apply

24°C



Dashboard > Manage Jenkins > Tools

Name

jdk17

☒ Install automatically ?

Install Oracle Java SE Development Kit from the website ?

Version

Java SE Development Kit 9.0.4

☐ I agree to the Java SE Development Kit Licence Agreement

! Installing JDK requires Oracle account. Please enter your username/password

! Oracle Java SE 11+ is not available for business, commercial or production use without a commercial license.
Public updates for Oracle Java SE 8 released after January 2019 will not be available for business, commercial or production use without a
[Oracle Java SE Licensing FAQ](#)

Install from adoptium.net ?

Version ?

jdk-17.0.11+9

Save

Apply

Deployment

Create Service Account, Role & Assign that role, And create a secret for Service Account and generate a Token

Create namespace

```
$kubectl create ns webapps
```

Creating Service Account

```
$vi svc.yml
```

```
apiVersion: v1
```

```
kind: ServiceAccount
```

```
metadata:
```

```
  name: jenkins
```

```
  namespace: webapps
```

```
kubectl apply -f svc.yml
```

Create Role

```
$vi role.yml
```

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: Role
```

@TeaCode1122

Divya satpute

metadata:

name: app-role

namespace: webapps

rules:

- apiGroups:

- ""

- apps

- autoscaling

- batch

- extensions

- policy

- rbac.authorization.k8s.io

resources:

- pods

- componentstatuses

- configmaps

- daemonsets

- deployments

- events

- endpoints

- horizontalpodautoscalers

- ingress

- jobs

- limitranges

- namespaces

- nodes

- secrets

- pods

- persistentvolumes

- persistentvolumeclaims

- resourcequotas

@TeaCode1122

Divya satpute

- replicaset
- replicationcontrollers
- serviceaccounts
- services

verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]

\$kubectI apply -f role.yml

Bind the role to service account

\$vi bind.yml

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: app-rolebinding

namespace: webapps

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: app-role

subjects:

- namespace: webapps

kind: ServiceAccount

name: jenkins

kubectI apply -f bind.yml

for token

vi jen.secret.yml

apiVersion: v1

kind: Secret

type: kubernetes.io/service-account-token

metadata:

name: mysecretname

annotations:

kubernetes.io/service-account.name: jenkins

@TeaCode1122

@TeaCode1122

Divya satpute

```
SCANNER_HOME= tool 'sonar-scanner'

}

stages {
    stage('Git Checkout') {
        steps {
            git branch: 'main', credentialsId: 'git-cred', url: 'https://github.com/divyasatpute/full-stack-app-project.git'
        }
    }
    stage('Compile') {
        steps {
            sh 'mvn compile'
        }
    }
    stage('Test') {
        steps {
            sh 'mvn test'
        }
    }
    stage('Trivy fs scan') {
        steps {
            sh 'trivy fs --format table -o fs.html .'
        }
    }
    stage('SonarQube Analysis') {
        steps {
            withSonarQubeEnv('sonar-server') {
                sh "'$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Blogging-app -Dsonar.projectKey=Blogging-app \-Dsonar.java.binaries=target'"
            }
        }
    }
}
```

@TeaCode1122

Divya satpute

```
}  
}  
stage('Build') {  
    steps {  
        sh 'mvn clean package'  
    }  
}  
  
stage('Publish Artifacts') {  
    steps {  
        withMaven(globalMavenSettingsConfig: 'maven-settings', jdk: 'jdk17', maven: 'maven3',  
mavenSettingsConfig: '', traceability: true) {  
            sh 'mvn deploy'  
        }  
    }  
}  
  
stage('Docker Build & Tag ') {  
    steps {  
        script{  
            withDockerRegistry(credentialsId: 'docker-cred', toolName: 'docker') {  
  
                sh 'docker build -t divyasatpute/bloggingapp:latest . --no-cache '  
            }  
        }  
    }  
}  
  
stage('Trivy image scan') {  
    steps {  
        sh 'trivy image --format table -o image.html divyasatpute/bloggingapp:latest'  
    }  
}  
  
stage('Docker Push') {
```

Divya satpute

```
    steps {  
      script{  
        withDockerRegistry(credentialsId: 'docker-cred', toolName: 'docker') {  
  
          sh 'docker push divyasatpute/bloggngapp:latest'  
        }  
      }  
    }  
  }  
  stage('k8-Deploy') {  
    steps {  
      withKubeConfig(caCertificate: '', clusterName: 'devopsshack-cluster', contextName: '',  
credentialsId: 'k8-cred', namespace: 'webapps', restrictKubeConfigAccess: false, serverUrl:  
'https://0D7DFCF662ECC24043497267C6A5BDEB.gr7.ap-south-1.eks.amazonaws.com') {  
        sh 'kubectl apply -f deployment-service.yml'  
        sleep 20  
      }  
    }  
  }  
  stage('verify the Deployment') {  
    steps {  
      withKubeConfig(caCertificate: '', clusterName: 'devopsshack-cluster', contextName: '',  
credentialsId: 'k8-cred', namespace: 'webapps', restrictKubeConfigAccess: false, serverUrl:  
'https://0D7DFCF662ECC24043497267C6A5BDEB.gr7.ap-south-1.eks.amazonaws.com') {  
        sh 'kubectl get pods'  
        sh 'kubectl get svc'  
      }  
    }  
  }  
}
```

Divya satpute

Installation Monitoring tool

```
$sudo apt update -y
```

```
$wget https://github.com/prometheus/prometheus/releases/download/v3.0.0-beta.0/prometheus-3.0.0-beta.0.linux-amd64.tar.gz
```

```
$tar -xvf prometheus-3.0.0-beta.0.linux-amd64.tar.gz
```

```
$wget
```

```
https://github.com/prometheus/blackbox_exporter/releases/download/v0.25.0/blackbox_exporter-0.25.0.linux-amd64.tar.gz
```

```
$tar -xvf blackbox_exporter-0.25.0.linux-amd64.tar.gz
```

```
$cd prometheus-3.0.0-beta.0.linux-amd64
```

```
$/prometheus &
```

```
$cd prometheus-3.0.0-beta.0.linux-amd64
```

```
$vi prometheus.yml
```

```
static_configs:
  - targets: ["localhost:9090"]
- job_name: 'blackbox'
  metrics_path: /probe
  params:
    module: [http_2xx] # Look for a HTTP 200 response.
  static_configs:
    - targets:
      - http://prometheus.io # Target to probe with http.
      - http://a90fdc9e67afb4ed78aa46051d12af40-1780603791.ap-south-1.elb.amazonaws.com/ # Target to probe with http on port 8080.
  relabel_configs:
    - source_labels: [__address__]
      target_label: __param_target
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: 13.232.13.30:9115
```

access prometheus <http://13.232.13.30:9090>

for blackbox exporter

```
$cd blackbox_exporter-0.25.0.linux-amd64
```

```
$/blackbox_exporter &
```

access blackbox <http://13.232.13.30:9090>



@TeaCode1122

Divya satpute

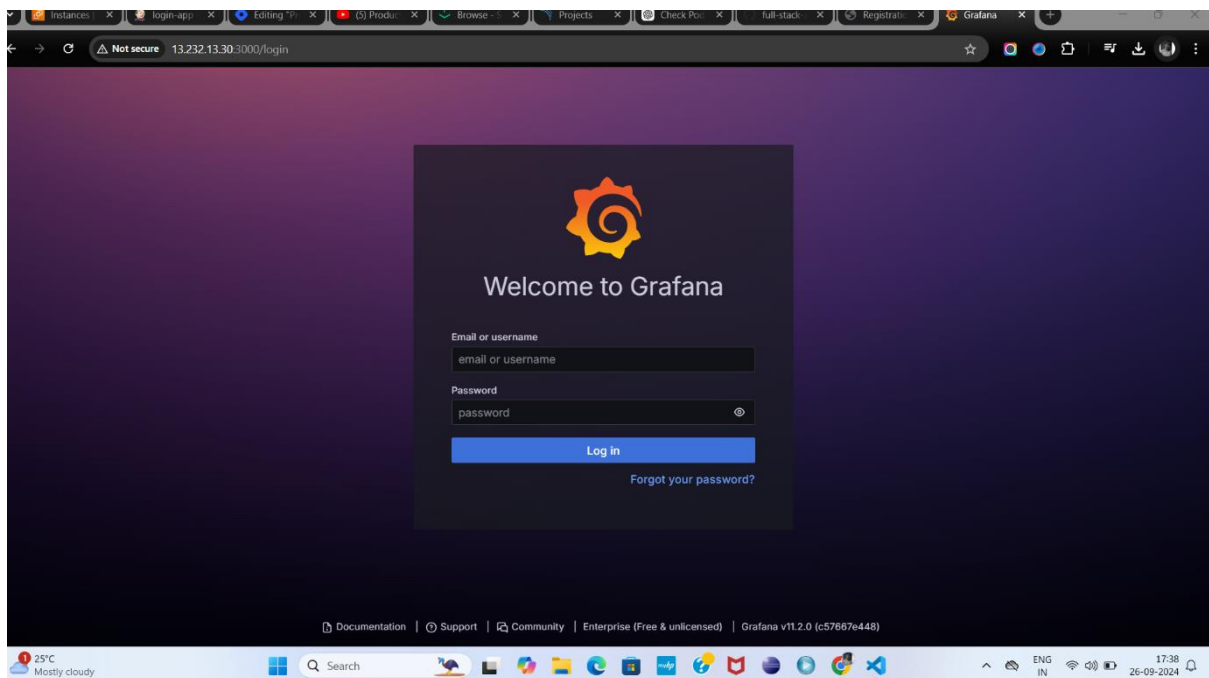
For Grafana

```
$sudo apt-get install -y adduser libfontconfig1 musl
```

```
$wget https://dl.grafana.com/enterprise/release/grafana-enterprise_11.2.0_amd64.deb
```

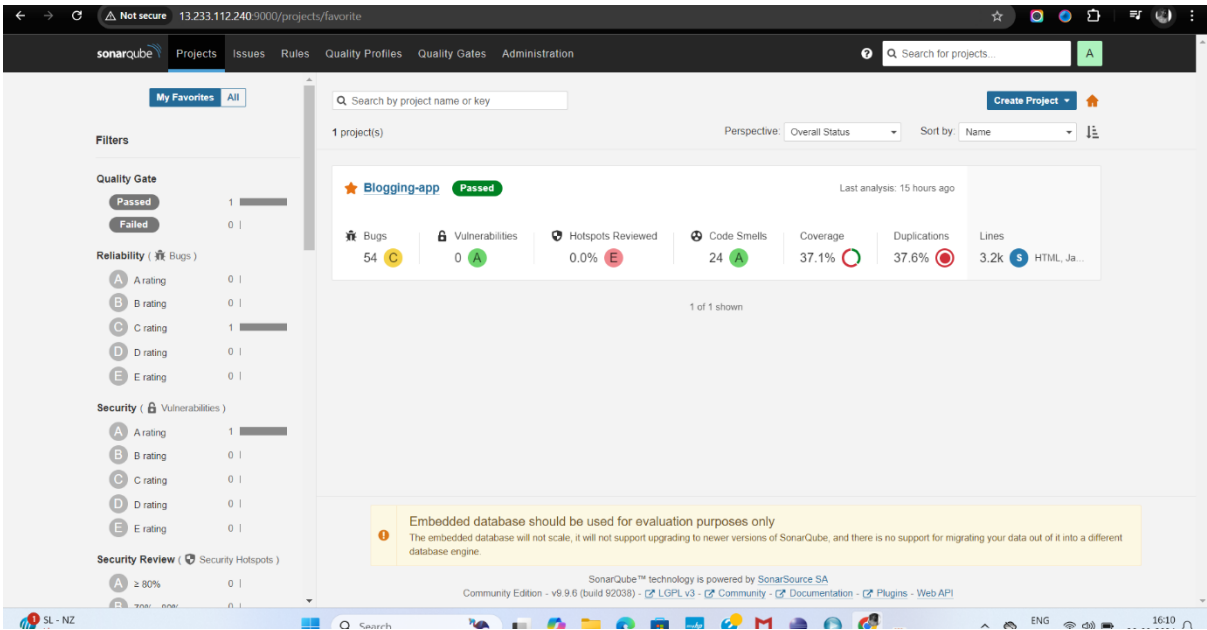
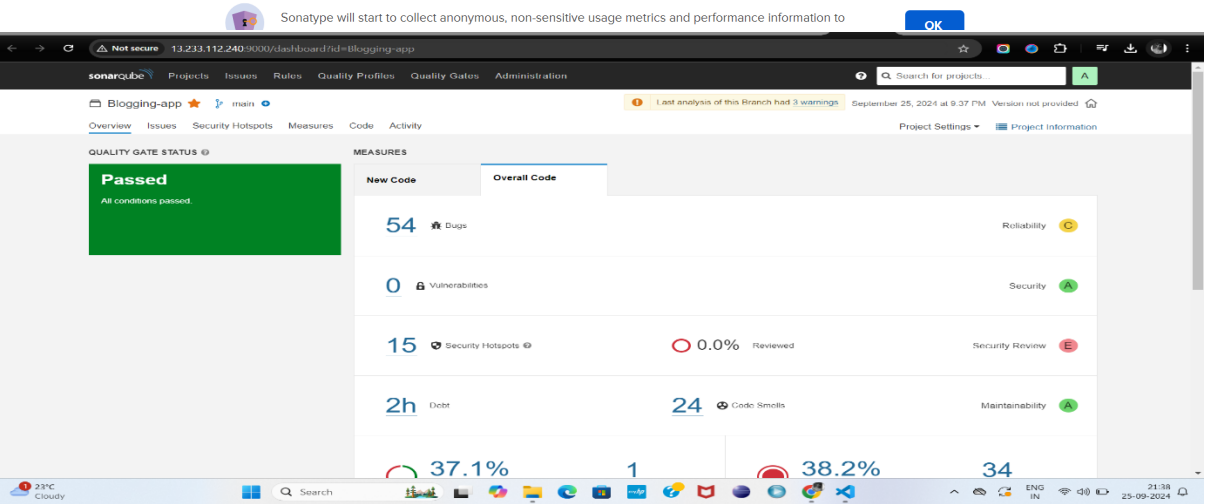
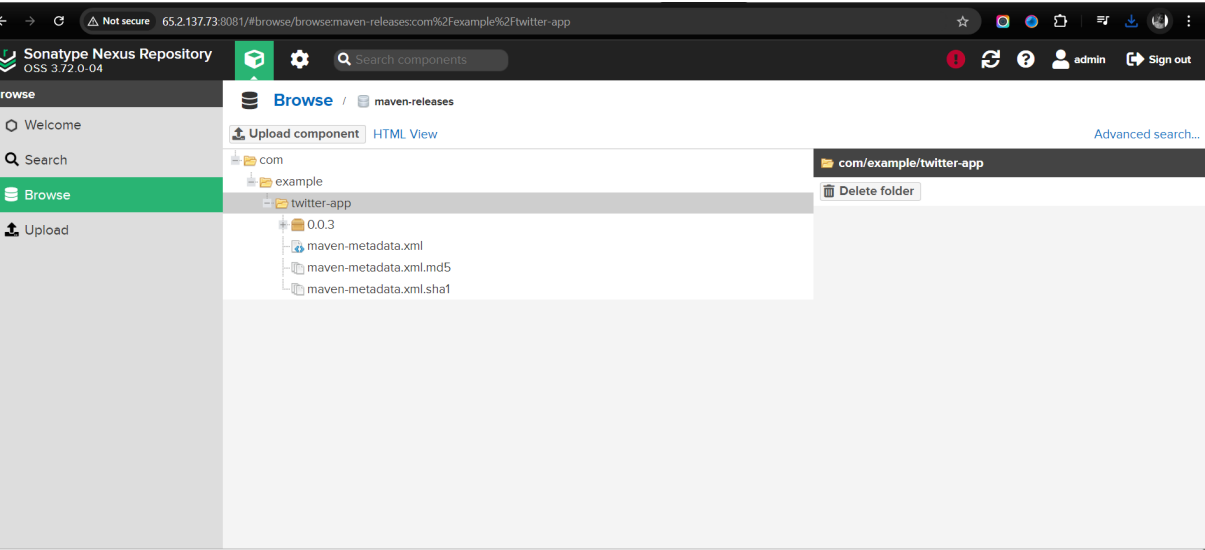
```
$sudo dpkg -i grafana-enterprise_11.2.0_amd64.deb
```

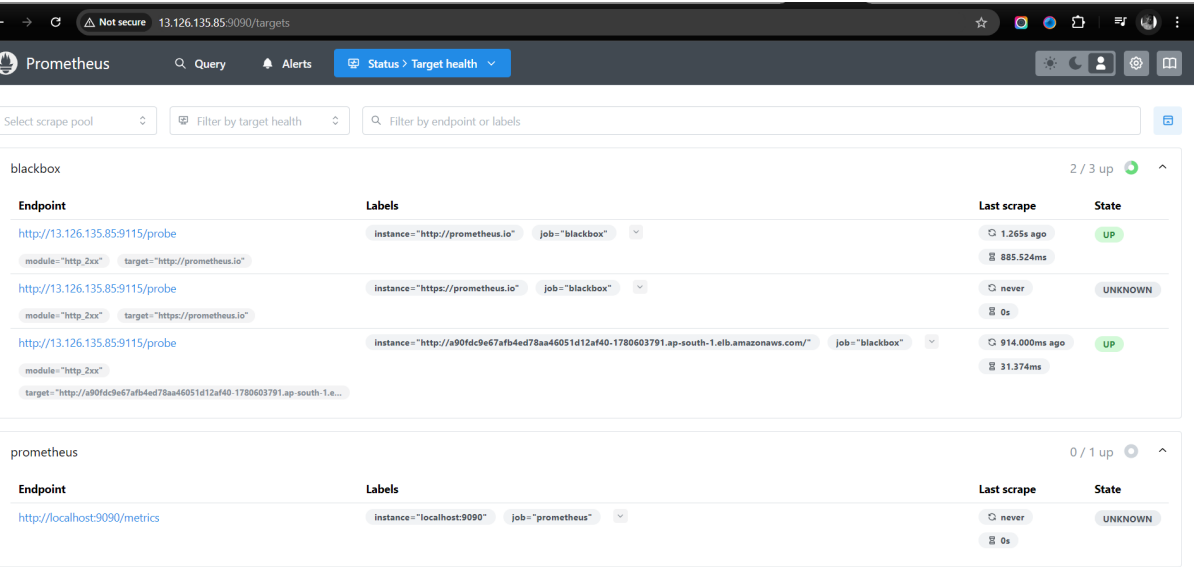
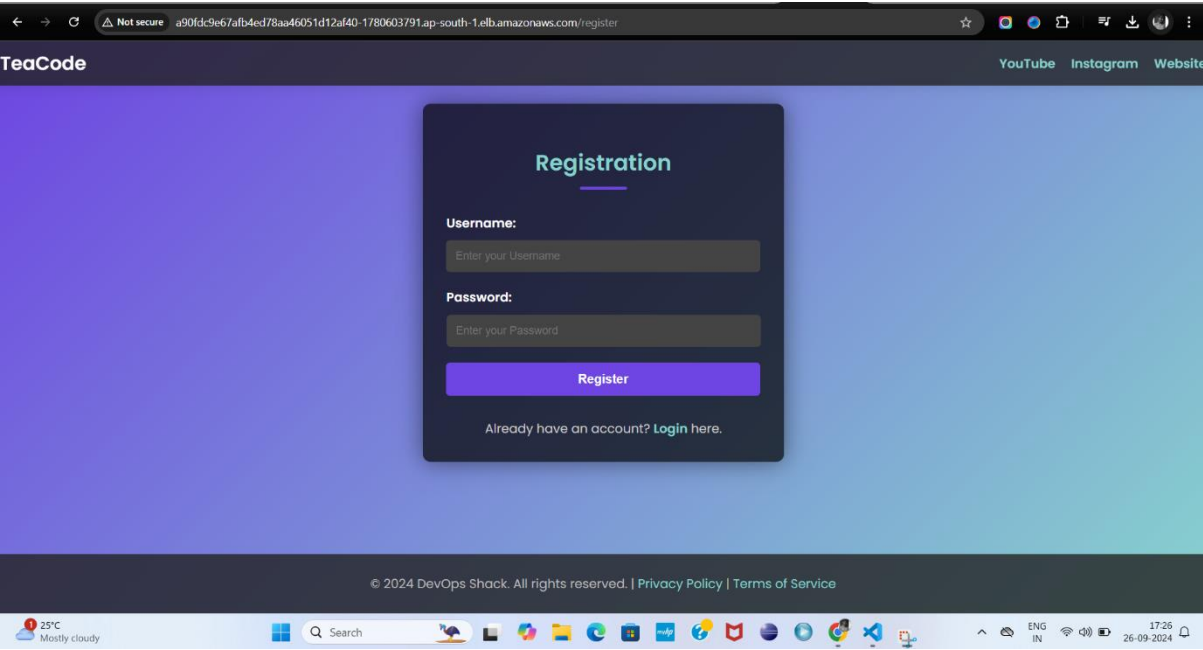
```
$sudo /bin/systemctl start grafana-server
```



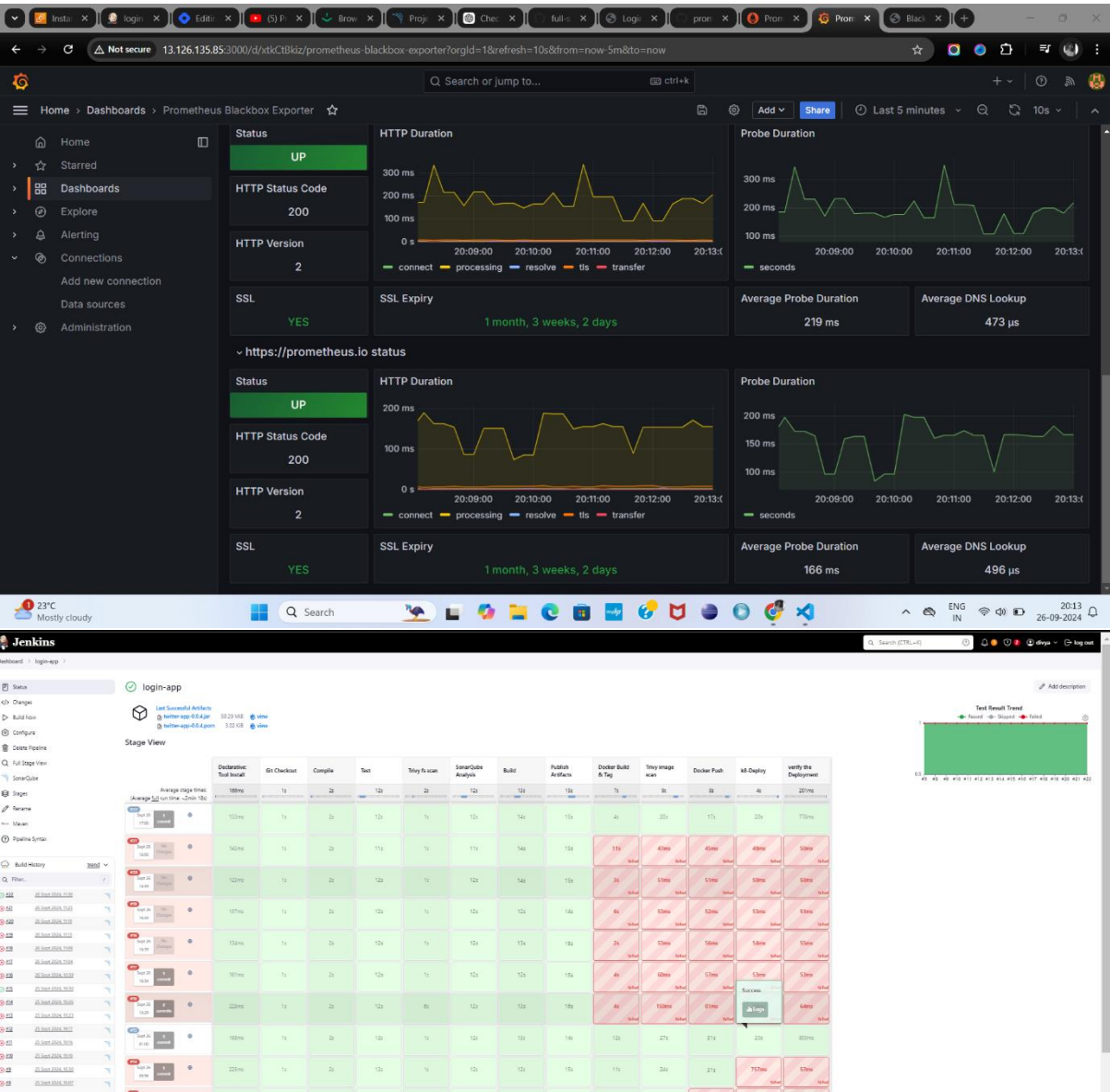
@TeaCode1122

Test Results





Divya satpute



Thanks youuu did it hurrayyyyy

@TeaCode1122