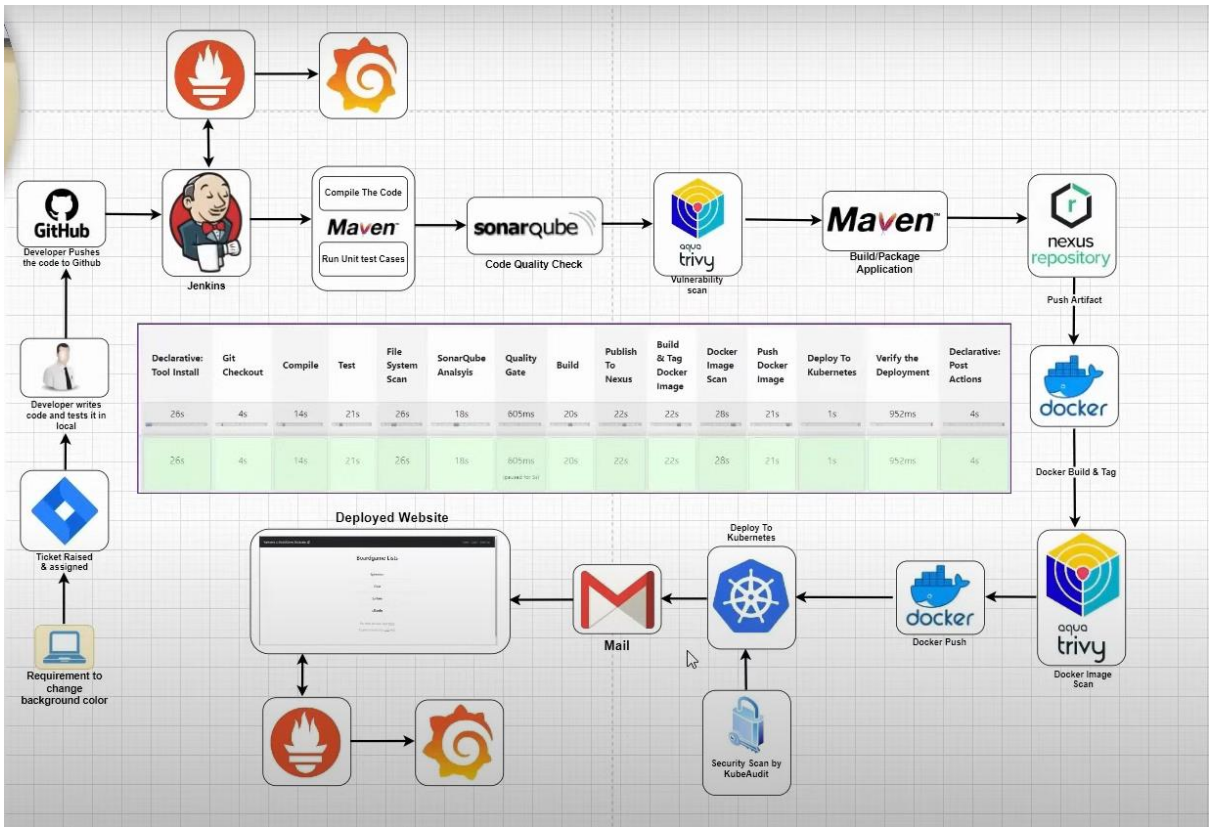
Boardgame Ultimate CI/CD Pipeline:

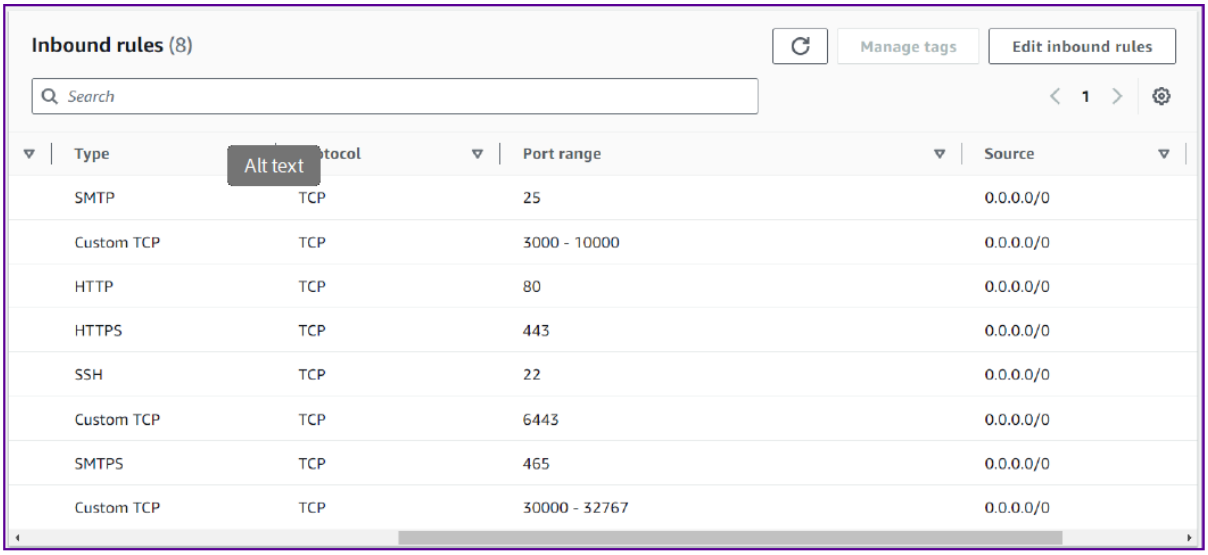
NAME: S. Venkateswarareddy
EMAIL: Singareddyv91@gmail.com
PHONE NO: 9491029062
BATCH NO: 3PM, 113
COURSE: AWS AND DEVOPS



Step1: Take EC2 Instance for Cluster Creation One for Master and two or more for Worker nodes with Minimum t2.medium (2 CPU, 4 GB Memory)

→Launch another Instances for Jenkins, SonarQube, Nexus.

→Enable these Ports on Security Group in INBOUND Rules:



Inbound rules (8)

Manage tags | Edit inbound rules

Search

1

Type	Protocol	Port range	Source
SMTP	TCP	25	0.0.0.0/0
Custom TCP	TCP	3000 - 10000	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0
Custom TCP	TCP	6443	0.0.0.0/0
SMTPS	TCP	465	0.0.0.0/0
Custom TCP	TCP	30000 - 32767	0.0.0.0/0

→Setup K8-Cluster using kubeadm [K8 Version-->1.28.1]

→Installing Jenkins on Ubuntu:

Install Plugins:

Eclipse temurin installer

Maven Integration plugin

SonarQube Scanner

Nexus Artifact uploader

Config file provider

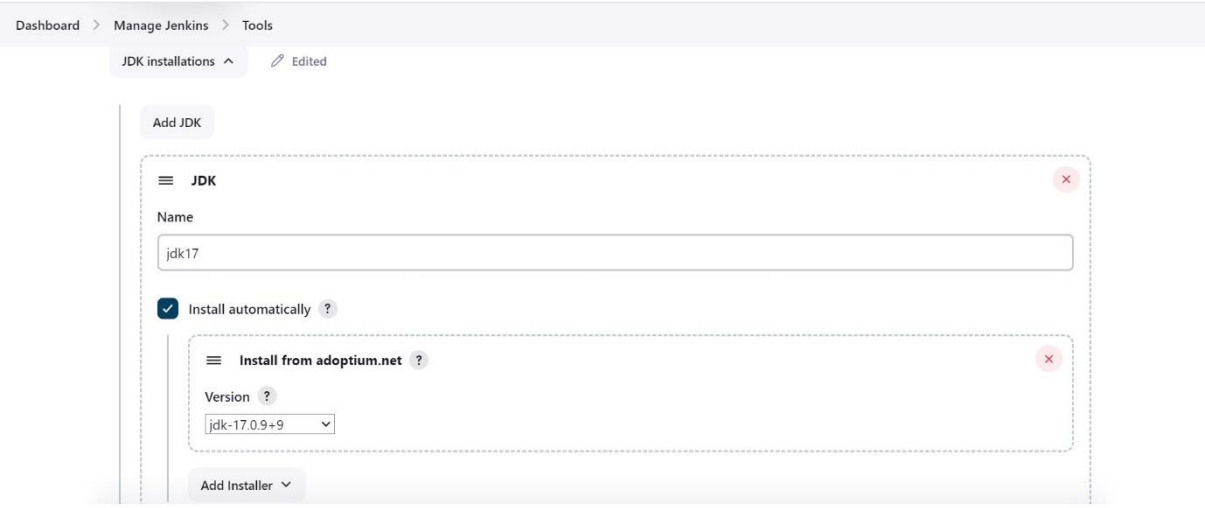
OWASP dependency check

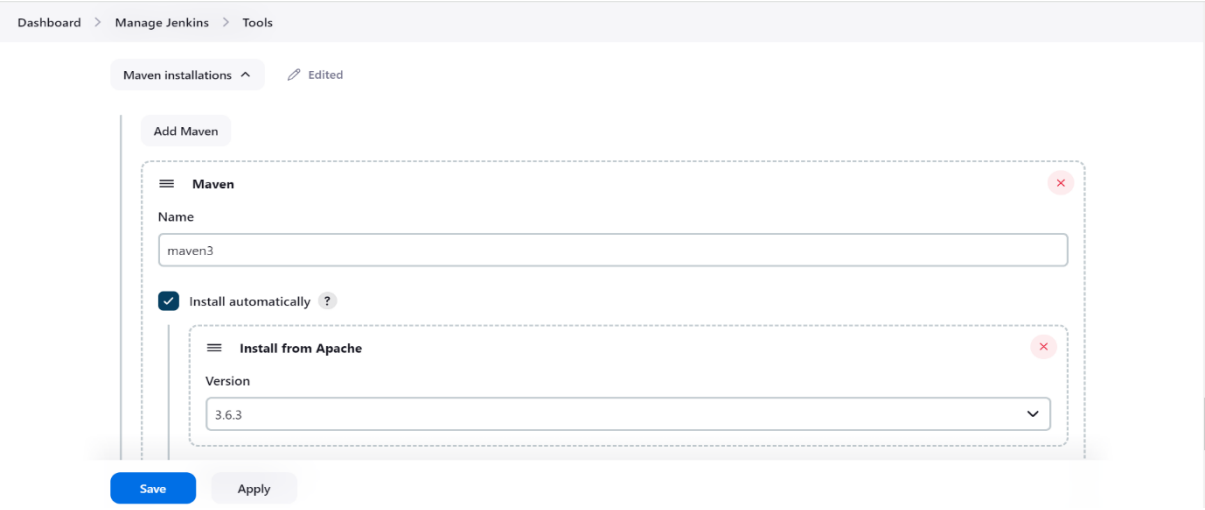
Docker, docker pipeline, Cloud Bees, docker build

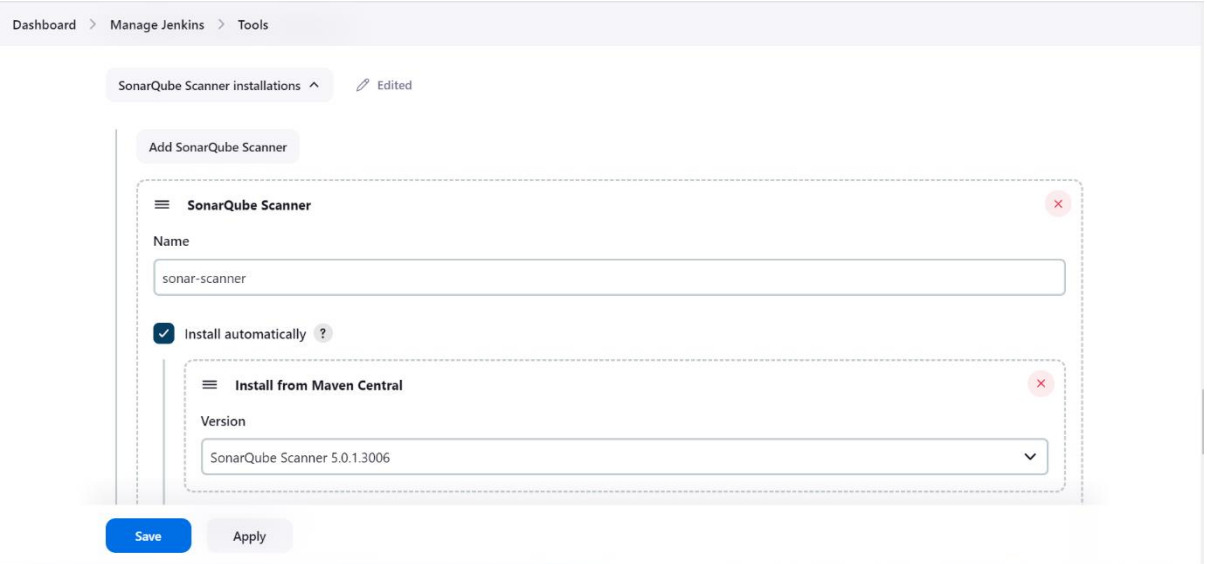
Kubernetes, Kubernetes CLI

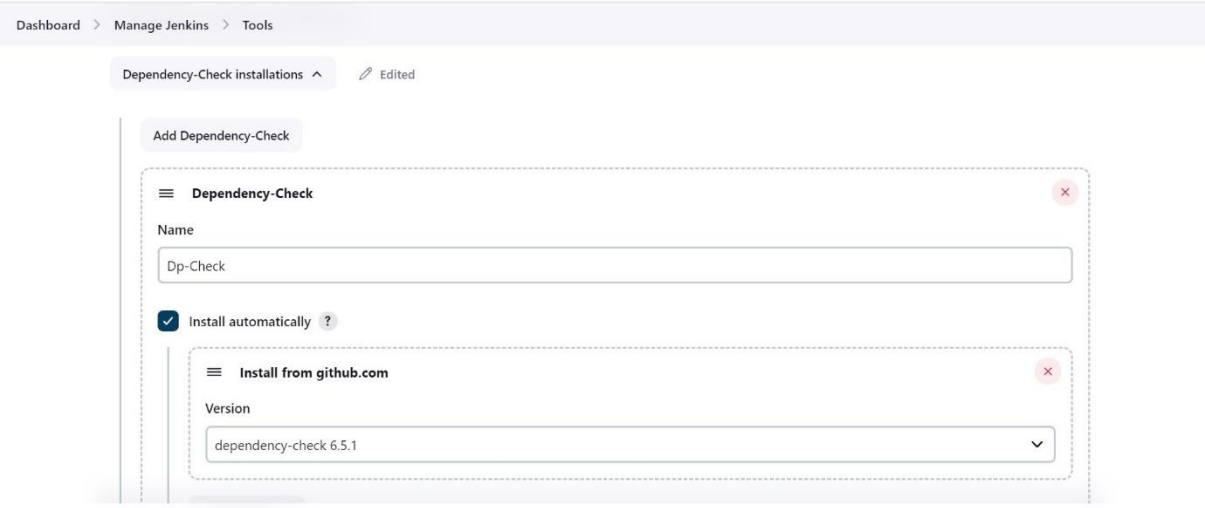
Prometheus

Go To Jenkins →Manage Jenkins → Tools→SetUp all these

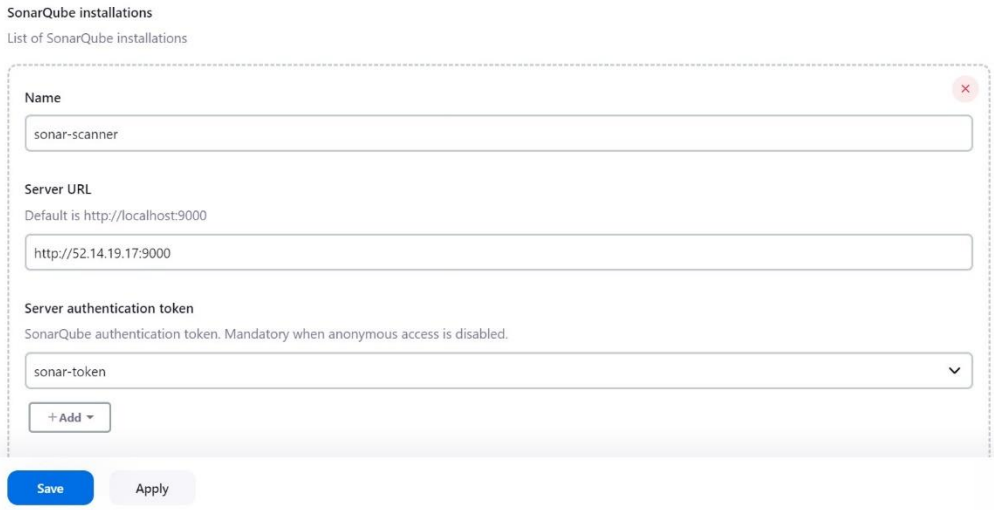


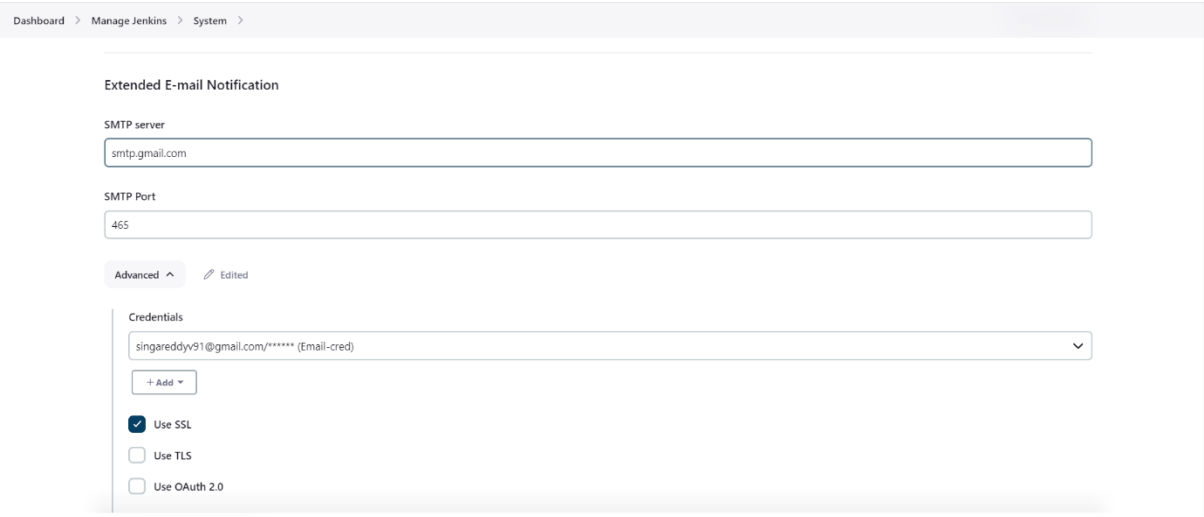


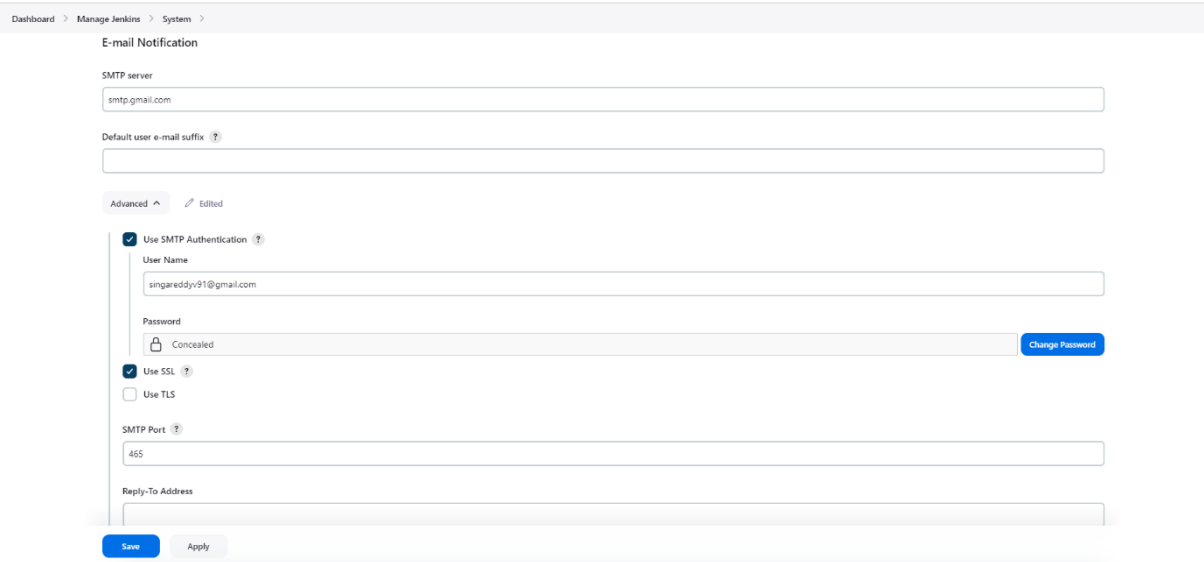




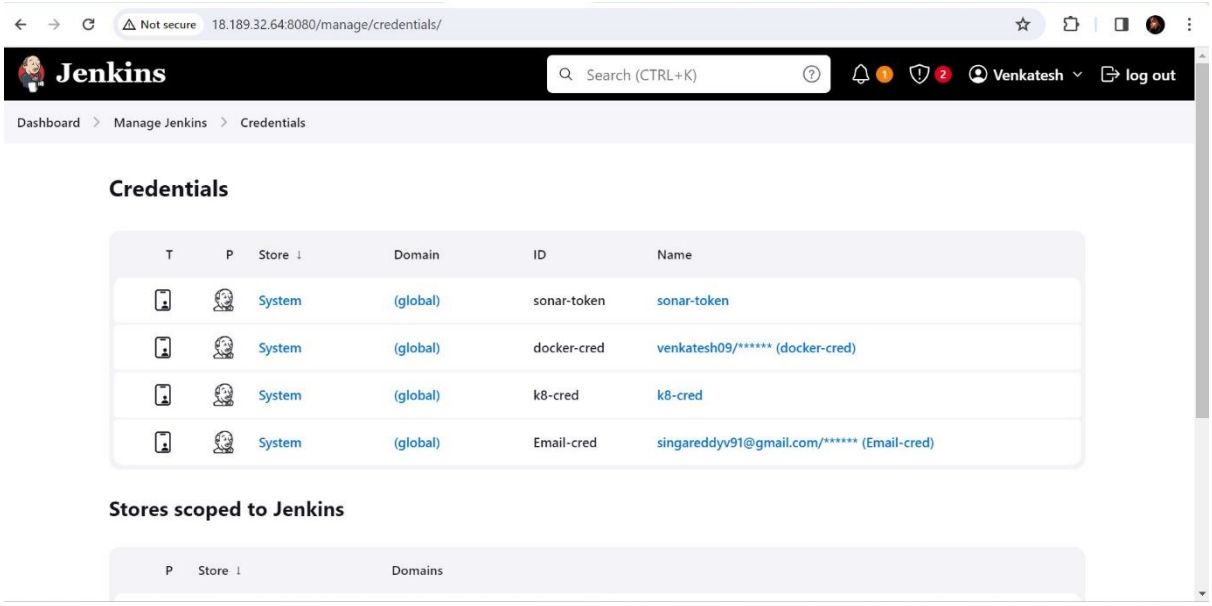
Go To Jenkins →Manage Jenkins →Systems→SetUp all these





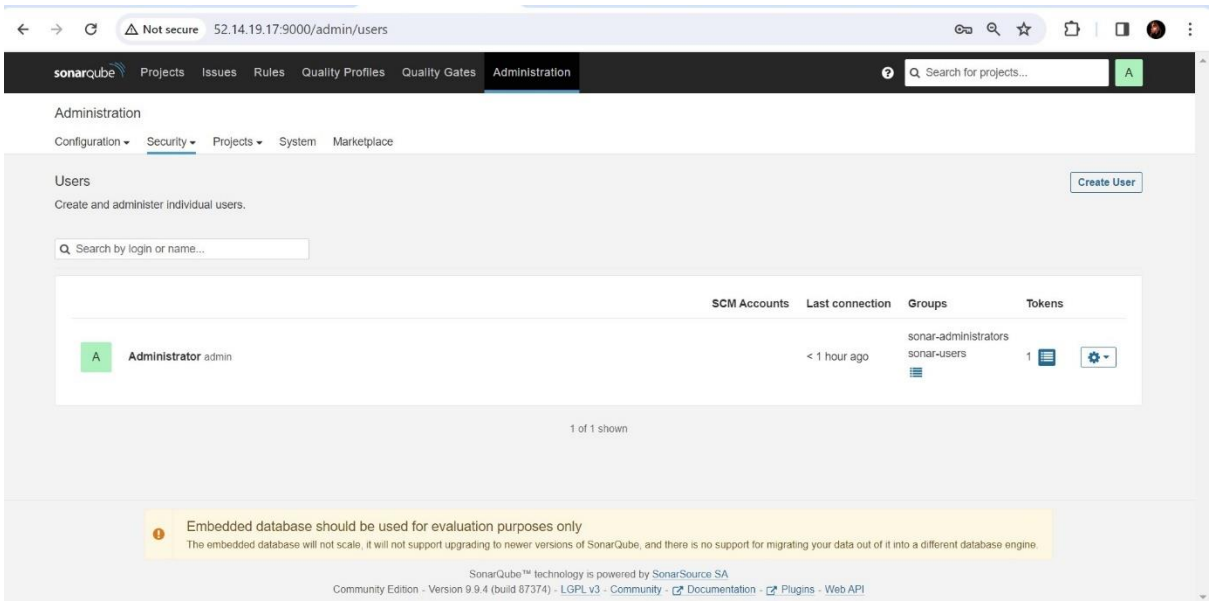


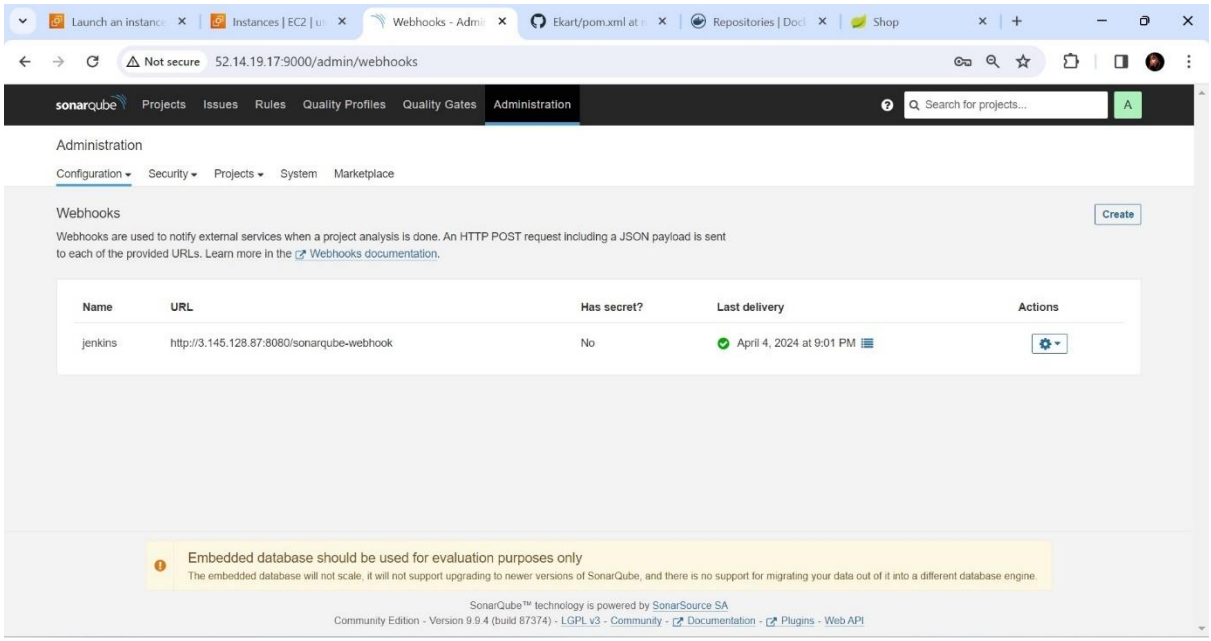
Go To Jenkins →Manage Jenkins →Credentials→SetUp all these



For SonarQube Integrate to Jenkins:

In SonarQube → Go to Administration→Security→Create User with Token for Credentials.





For Nexus Integrate to Jenkins:

Go To Jenkins →Manage Jenkins →Managed Files→SetUp all these

→Go to Kubernetes Master node Create RBAC (RoleBased Access Control)

Pipeline:

Boardgame Ultimate CI/CD Pipeline

```
pipeline {
    agent any
    tools {
        jdk 'jdk17'
        maven 'maven3'
    }
    environment {
```

```
        SCANNER_HOME= tool 'sonar-scanner'
    }
    stages {
        stage('Git Checkout') {
            steps {
                checkout scmGit(branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/Singareddy-Venkatesh/Boardgame.git']])
            }
        }
        stage('Compile') {
            steps {
                sh 'mvn compile'
            }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
        }
```

```
        stage('File System Scan') {
            steps {
                sh "trivy fs --format table -o trivy-fs-report.html ."
            }
        }
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('sonar-scanner') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Ekart -Dsonar.projectKey=Ekart -
Dsonar.java.binaries=. '''
                }
            }
        }
        stage('Quality Gate') {
            steps {
                script {
                    waitForQualityGate abortPipeline: false,
credentialsId: 'sonar-token'
                }
```

```
        }
    }
    stage('Maven package') {
        steps {
            sh 'mvn package'
        }
    }
    stage('OWASP Depencency Check') {
        steps {
            dependencyCheck additionalArguments: '--scan ./ --
format XML', odcInstallation: 'Dp-Check'
            dependencyCheckPublisher pattern:
'**/dependency-check-report.xml'
        }
    }
    stage('Deploy to Nexus') {
        steps {
            withMaven(globalMavenSettingsConfig: 'Global-
Settings', jdk: 'jdk17', maven: 'maven3',
mavenSettingsConfig: '', traceability: true) {
                sh "mvn deploy"
```

```
            }
        }
    }
    stage('Build & Docker Image') {
        steps {
            script {
                withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {
                    sh "docker build -t
venkatesh09/boardgame:latest ."
                }
            }
        }
    }
    stage('Trivy') {
        steps {
            sh "trivy image --format table -o trivy-image-
report.html venkatesh09/boardgame:latest"
        }
    }
```

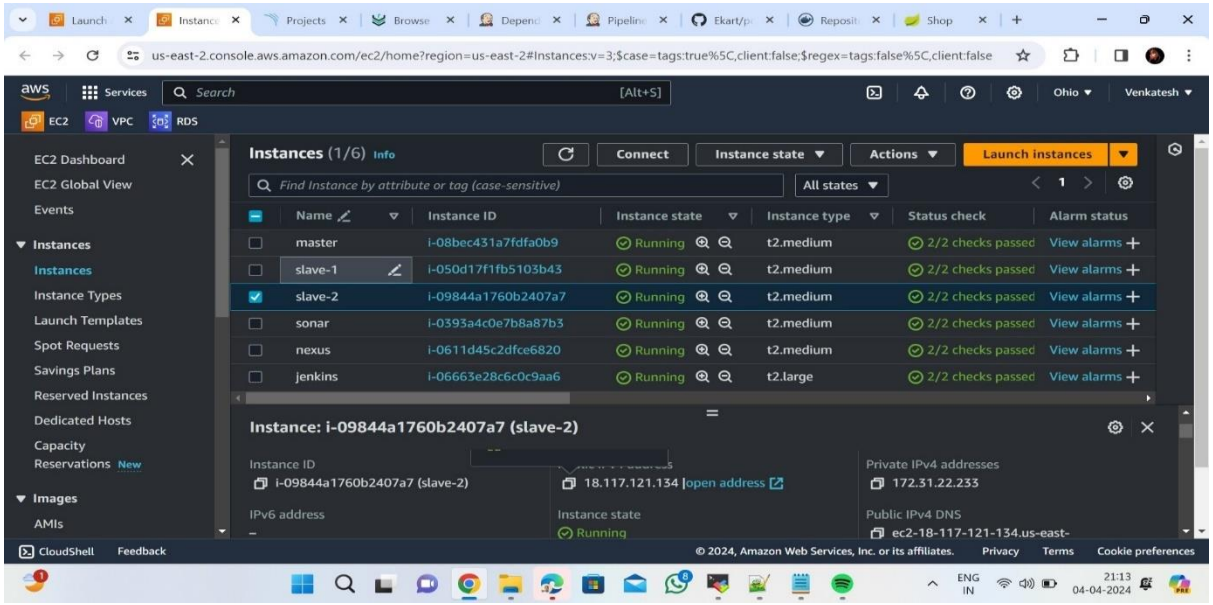
```
        stage('Docker Push Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {
                        sh "docker push
venkatesh09/boardgame:latest"
                    }
                }
            }
        }
        stage('Deploy to Kubernetes') {
            steps {
                withKubeConfig(caCertificate: '', clusterName:
'kubernetes', contextName: '', credentialsId: 'k8-cred',
namespace: 'webapps', restrictKubeConfigAccess: false,
serverUrl: 'https://172.31.29.58:6443') {
                    sh "kubectl apply -f deployment-service.yaml -n
webapps"
                    sh "kubectl get pods -n webapps"
                    sh "kubectl get svc -n webapps"
                }
```

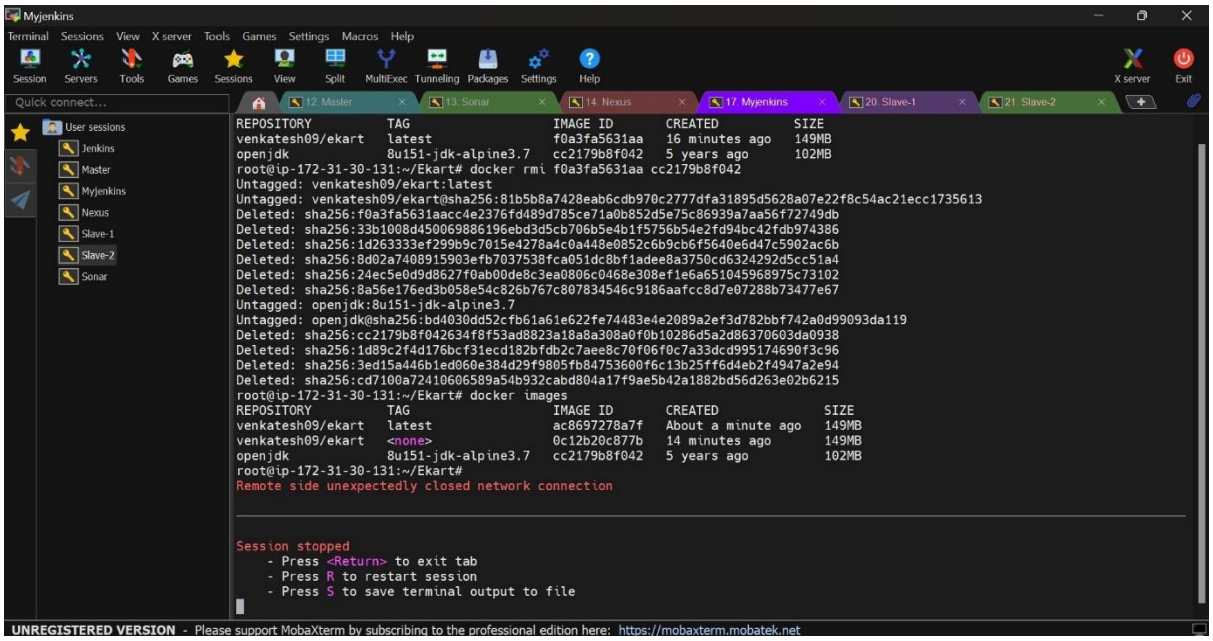
```
            }
        }
    }
    post {
        always {
        script {
        def jobName = env.JOB_NAME
        def buildNumber = env.BUILD_NUMBER
        def pipelineStatus = currentBuild.result ?: 'UNKNOWN'
        def bannerColor = pipelineStatus.toUpperCase() ==
'SUCCESS' ? 'green' : 'red'


        def body = """
        <html>
        <body>
        <div style="border: 4px solid ${bannerColor}; padding:
10px;">
        <h2>${jobName} - Build ${buildNumber}</h2>
        <div style="background-color: ${bannerColor}; padding:
10px;">
```

```
        <h3 style="color: white;">Pipeline Status: ${pipelineStatus.toUpperCase()}</h3>

        </div>

        <p>Check the <a href="${BUILD_URL}">console output</a>.</p>

        </div>

        </body>

        </html>

    """


        emailext (

            subject: "${jobName} - Build ${buildNumber} - ${pipelineStatus.toUpperCase()}",

            body: body,

            to: 'singareddyv91@gmail.com',

            from: 'jenkins@example.com',

            replyTo: 'jenkins@example.com',

            mimeType: 'text/html',

            attachmentsPattern: 'trivy-image-report.txt'

            )
```

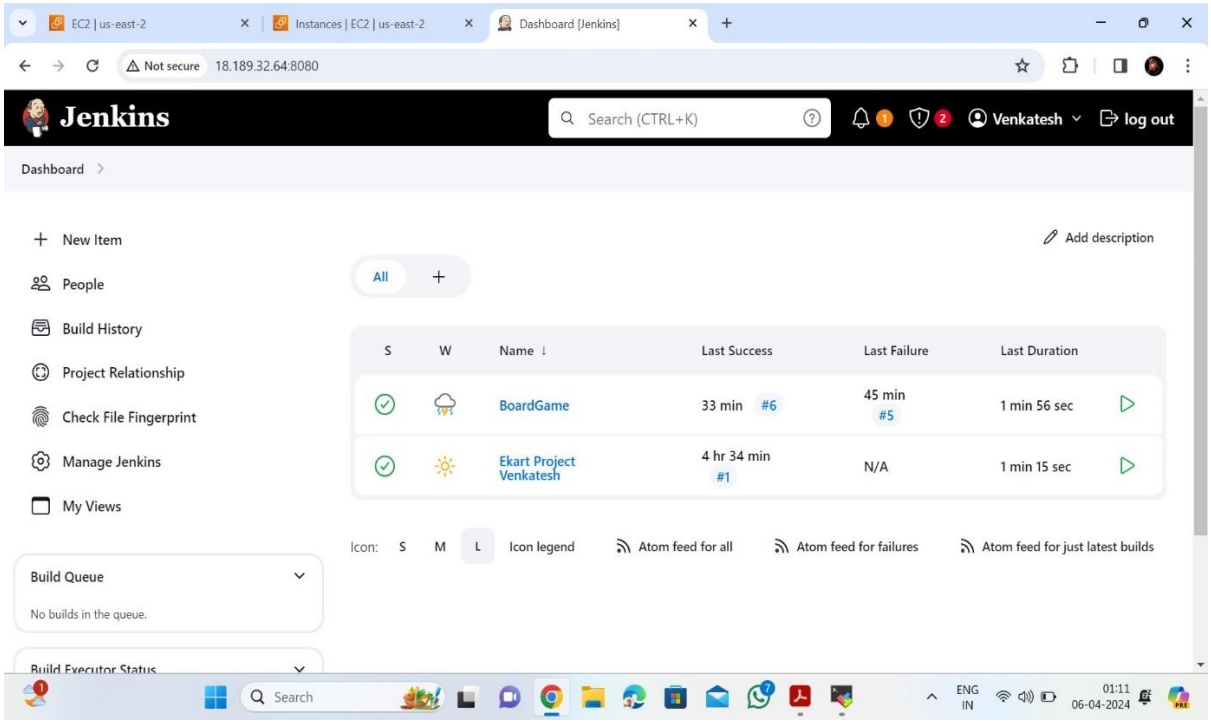
```
        }

        }

    }

}
```

Launch EC2 Instances:

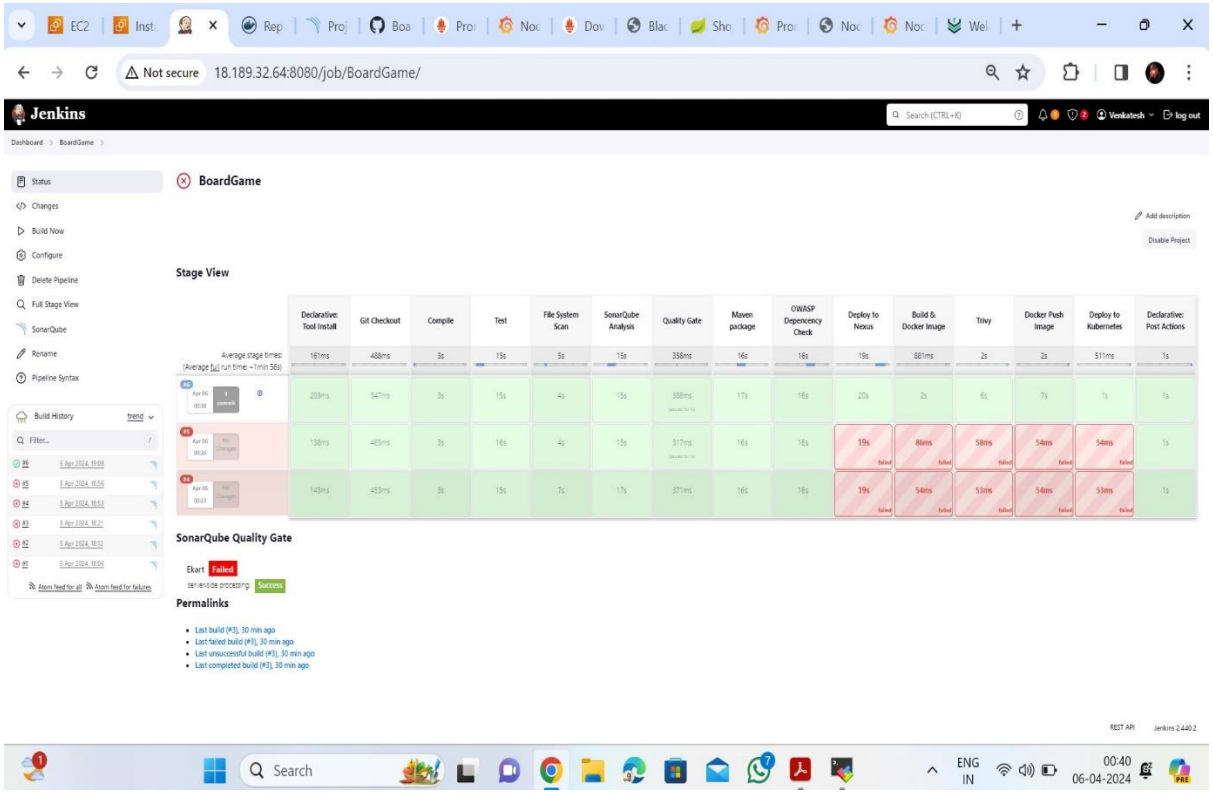




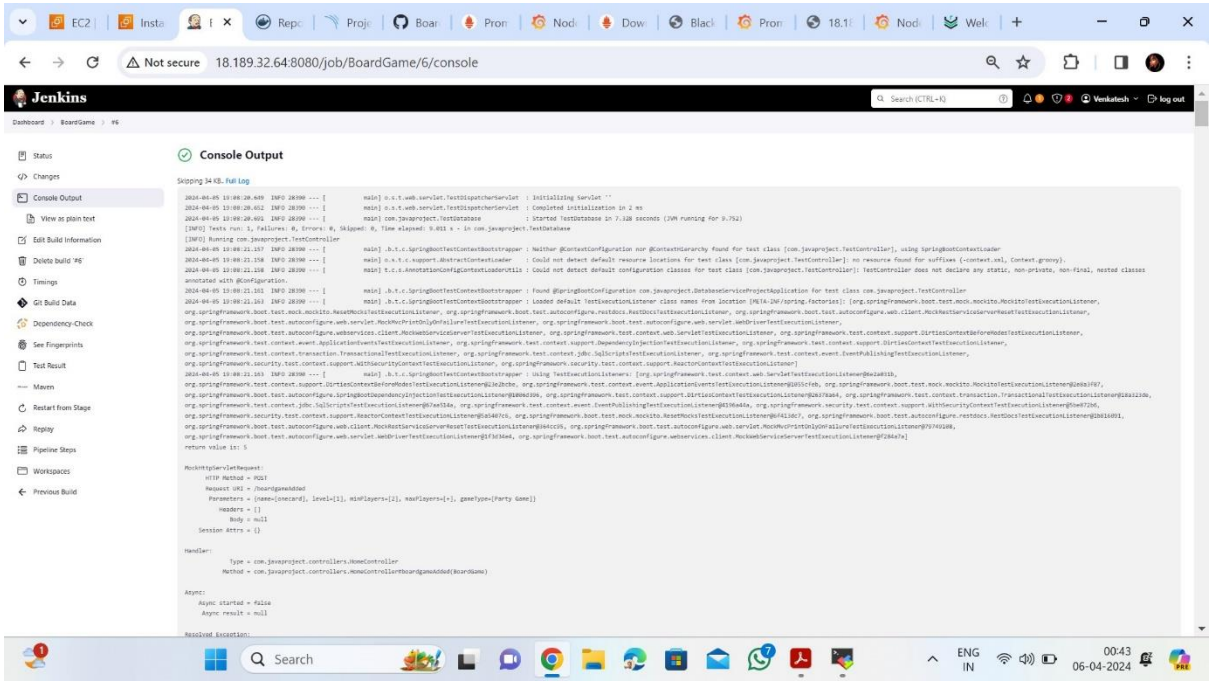
Created A NEW JOB:

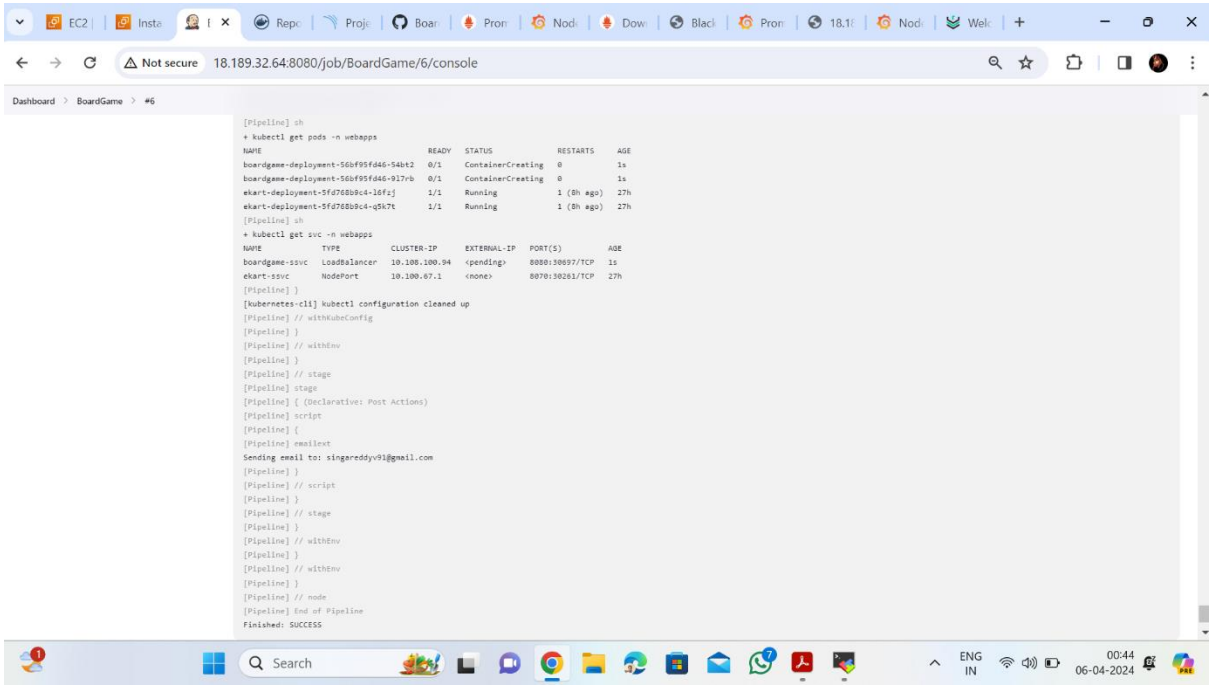


Successfully Runs the Pipeline:

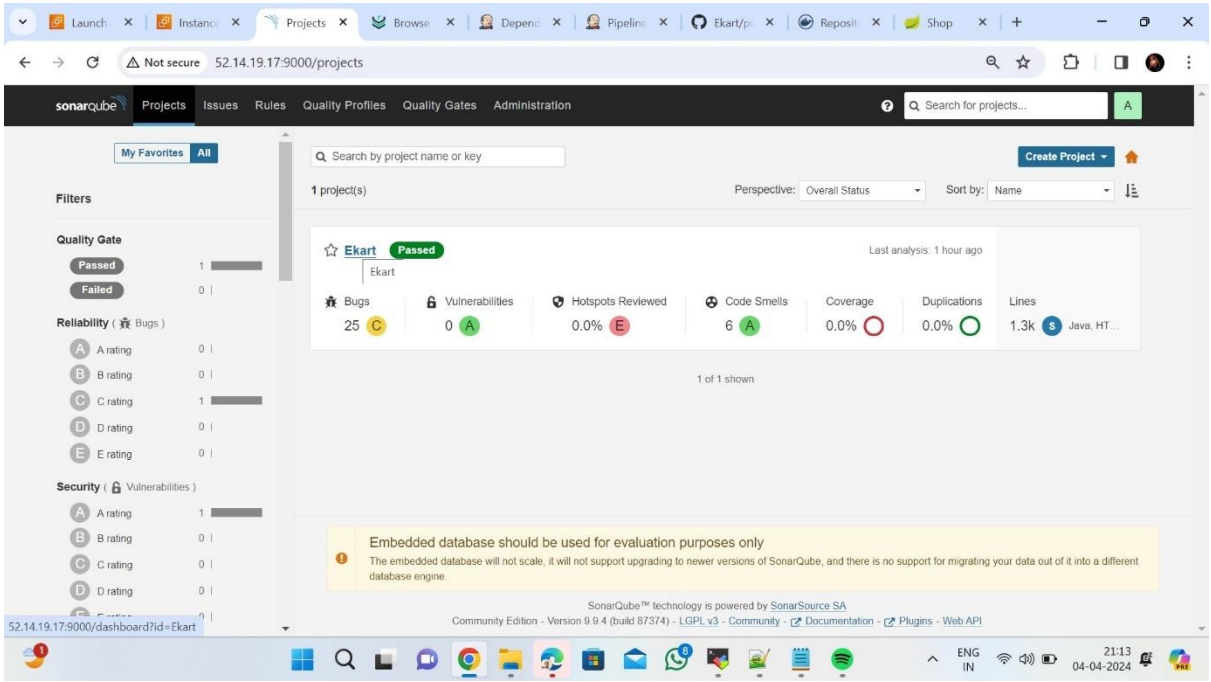


Console Output:

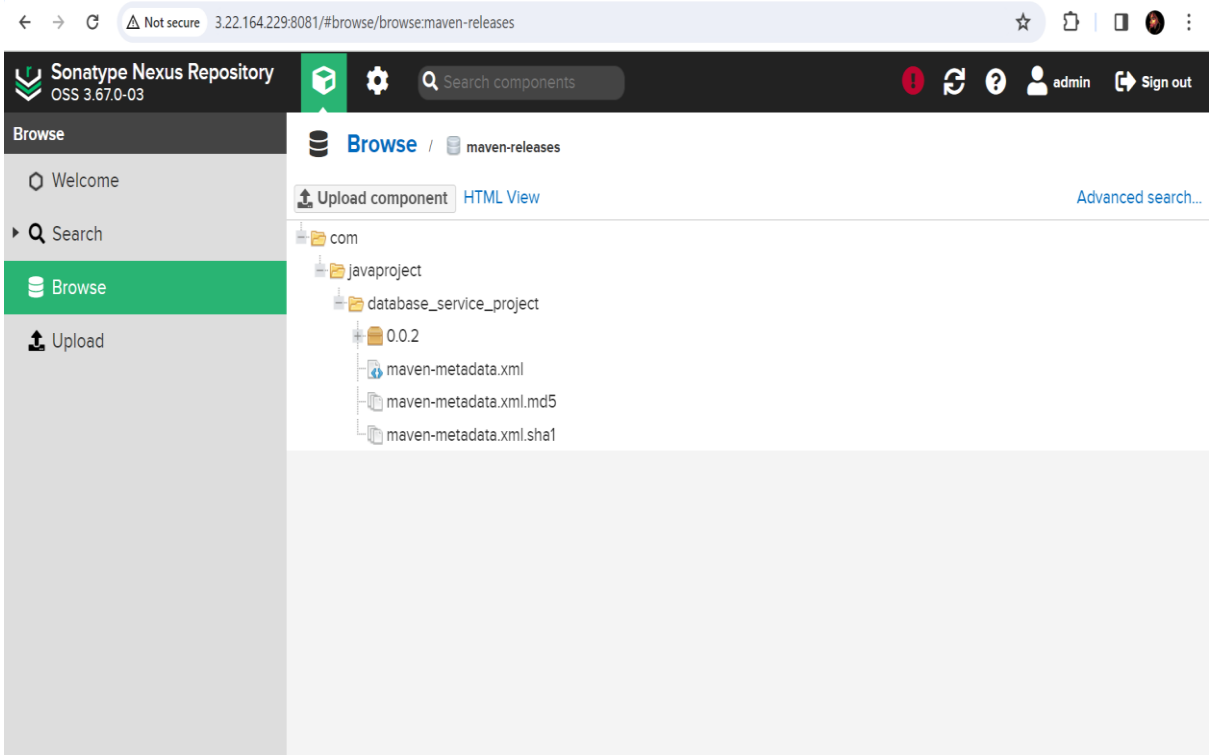




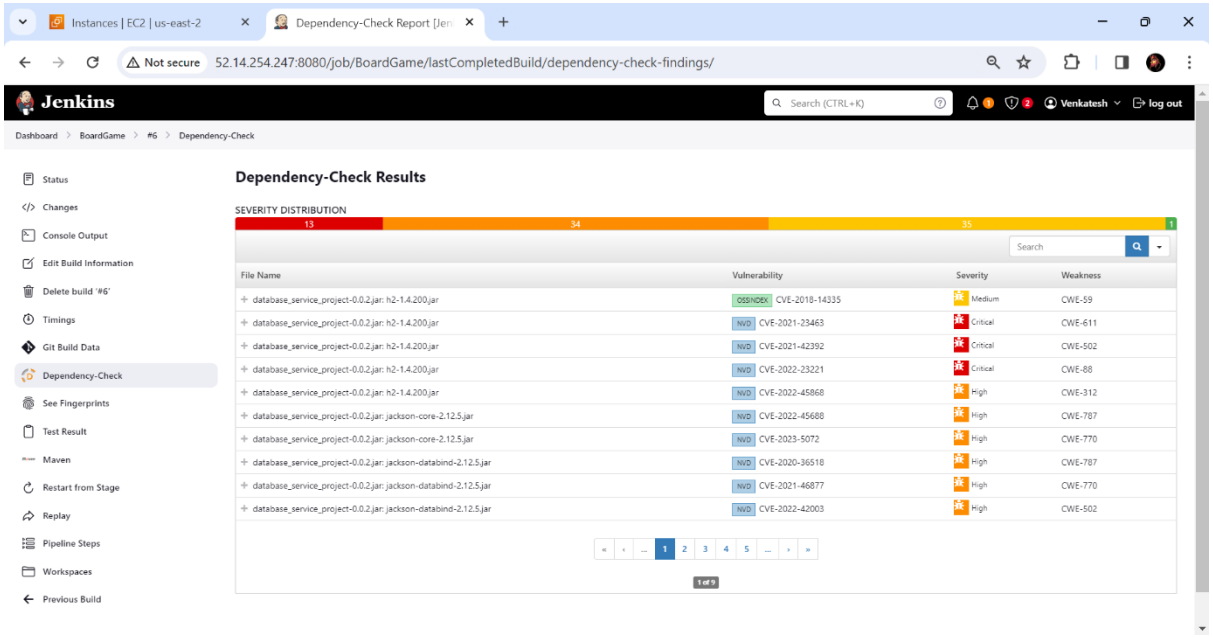
SonarQube Output:



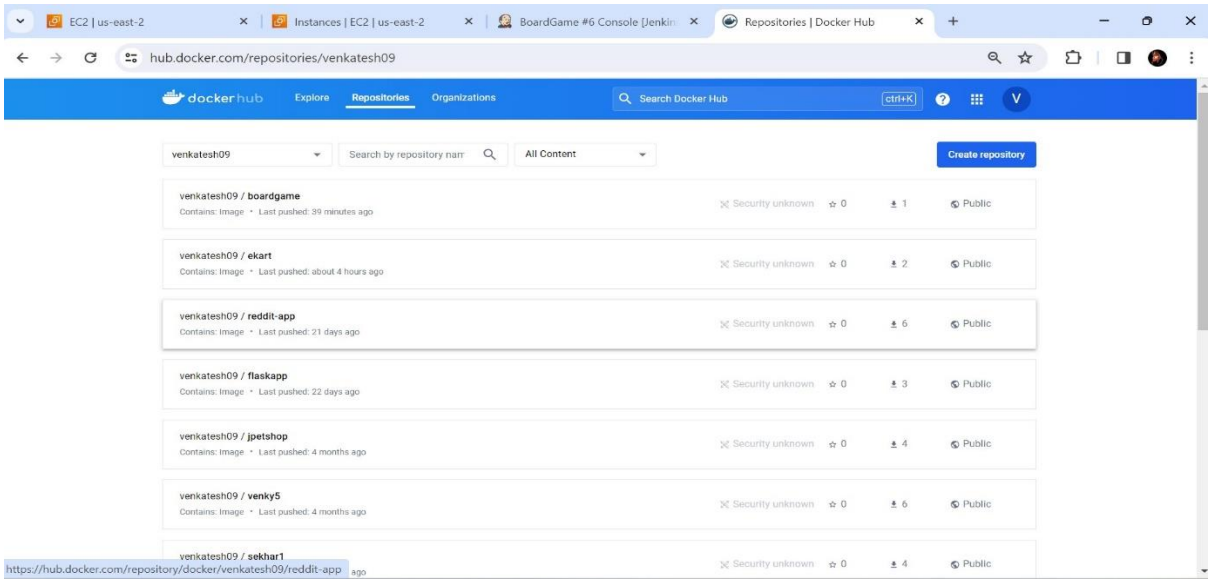
Nexus:



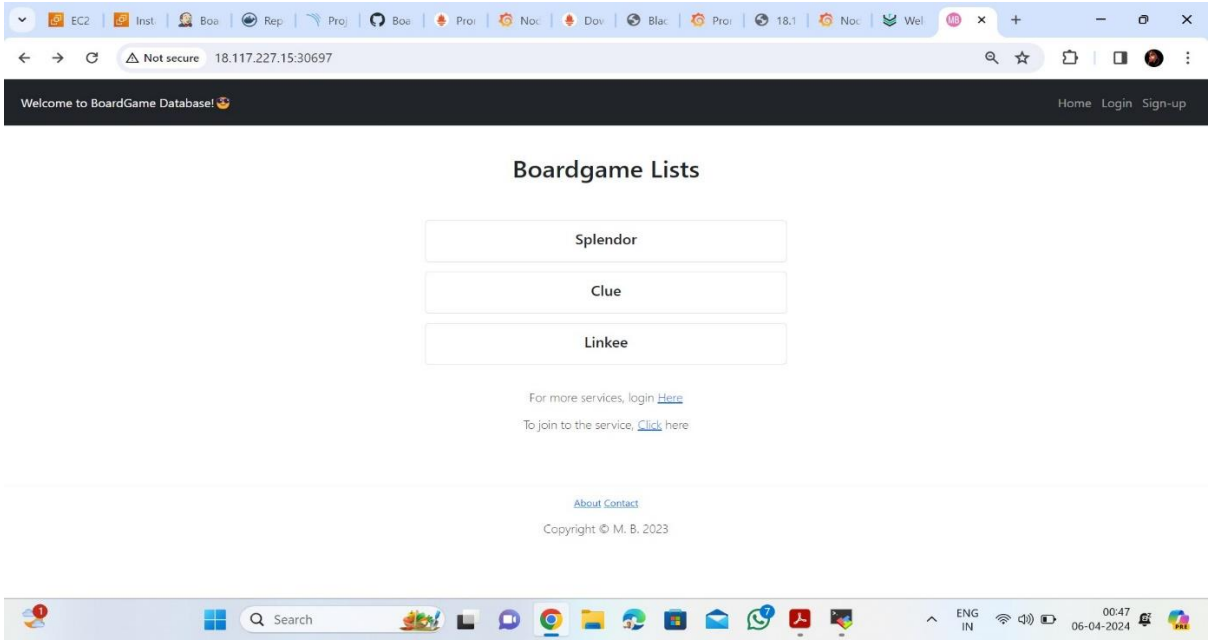
Dependency Check:



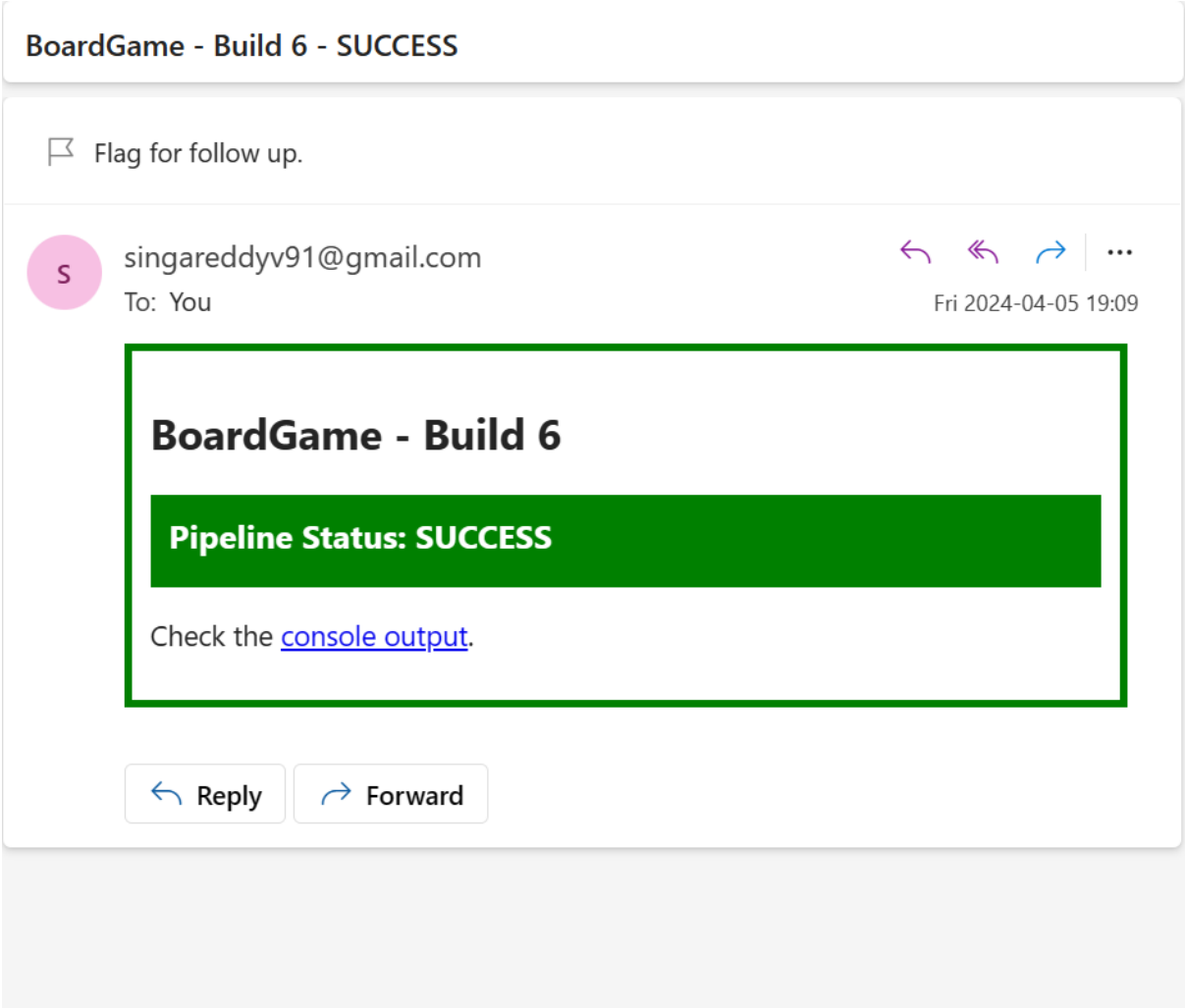
Docker:



Final Output:



Email Notification



Monitoring our websites by using Prometheus, Grafana, Blackbox,NodeExporter

Highly recomended to follow the steps

Links to download Prometheus, Node_Exporter & black Box exporter https://prometheus.io/download/
Links to download Grafana
https://grafana.com/grafana/download

Other link from video
https://github.com/prometheus/blackbox_exporter