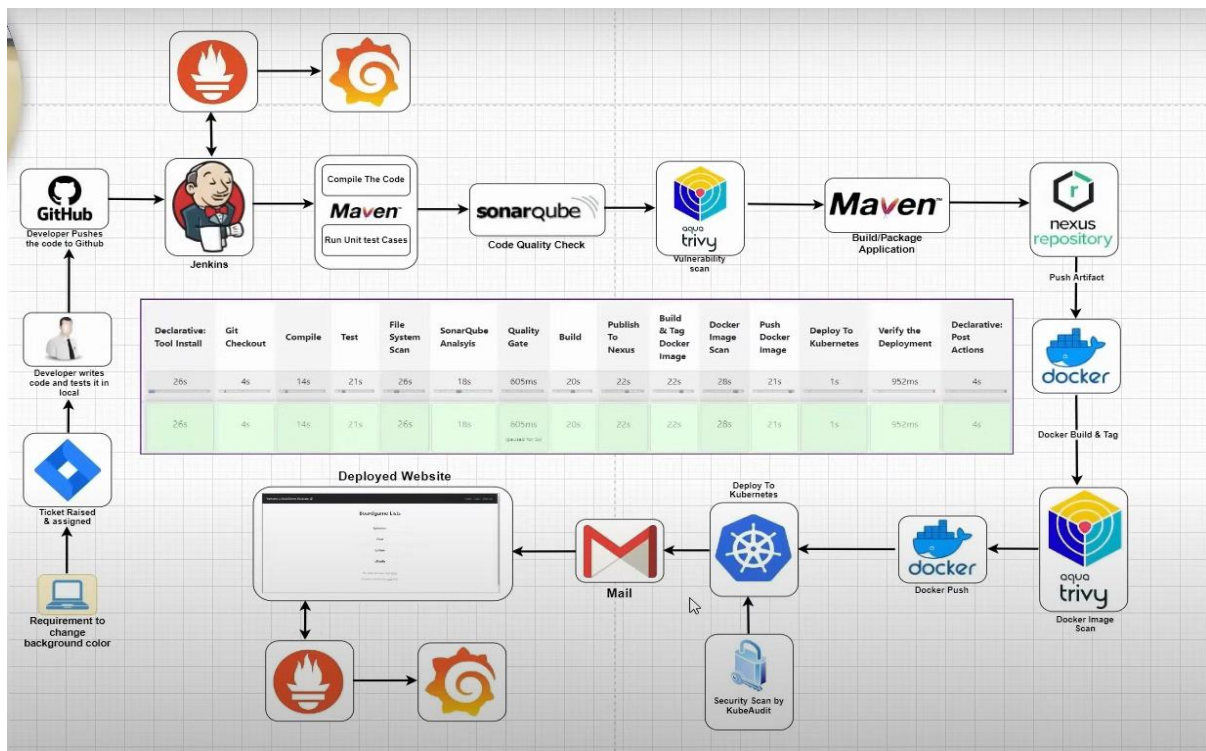


Ekart Ultimate CI/CD Pipeline:



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:

→ Take Jenkins Server and install java17 and Jenkins, trivy, docker, kubectl.

→ Install docker for future use

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 → Set Up all these



The screenshot shows the Jenkins 'Tools' configuration page. The breadcrumb trail at the top is 'Dashboard > Manage Jenkins > Tools'. Below this, there's a section for 'JDK installations' with an 'Add JDK' button and an 'Edited' status. The main configuration area is titled 'JDK' and contains a 'Name' field with the value 'jdk17'. Below the name field, there's a checkbox labeled 'Install automatically' which is checked. Underneath, there's a section titled 'Install from adoptium.net' with a 'Version' dropdown menu set to 'jdk-17.0.9+9'. At the bottom of the configuration area, there's an 'Add Installer' button.

Maven installations ^ Edited

Add Maven

Maven

Name

maven3

☒ Install automatically ?

Install from Apache

Version

3.6.3

Save

Apply

SonarQube Scanner installations ^ Edited

Add SonarQube Scanner

SonarQube Scanner

Name

sonar-scanner

☒ Install automatically ?

Install from Maven Central

Version

SonarQube Scanner 5.0.1.3006

Save

Apply

Dependency-Check installations ^ Edited

Add Dependency-Check

Dependency-Check

Name

Dp-Check

☒ Install automatically ?

Install from github.com

Version

dependency-check 6.5.1

Dashboard > Manage Jenkins > Tools

Docker installations

Docker installations ^ Edited

Add Docker

≡ Docker

Name

docker

☒ Install automatically ?

≡ Download from docker.com

Docker version ?

latest

Go To Jenkins → Manage Jenkins → Systems → Set Up all these

SonarQube installations

List of SonarQube installations

Name

sonar-scanner

Server URL

Default is http://localhost:9000

http://52.14.19.17:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

sonar-token

+ Add ▾

Save

Apply

Dashboard > Manage Jenkins > System >

Extended E-mail Notification

SMTP server

smtp.gmail.com

SMTP Port

465

Advanced ^ Edited

Credentials

singaredy91@gmail.com/***** (Email-cred)

+ Add ▾

☒ Use SSL

☐ Use TLS

☐ Use OAuth 2.0

Dashboard > Manage Jenkins > System >

E-mail Notification

SMTP server
smtp.gmail.com

Default user e-mail suffix ?

Advanced ^ Edited

☒ Use SMTP Authentication ?

User Name
singareddyv91@gmail.com

Password
Concealed [Change Password](#)

☒ Use SSL ?
☐ Use TLS

SMTP Port ?
465

Reply-To Address

[Save](#) [Apply](#)

Go To Jenkins → Manage Jenkins → Credentials → Set Up all these

← → ↺ Not secure 18.189.32.64:8080/manage/credentials/ ☆ 🔒 🔒 🔒 Venkatesh v log out

Dashboard > Manage Jenkins > Credentials

Credentials

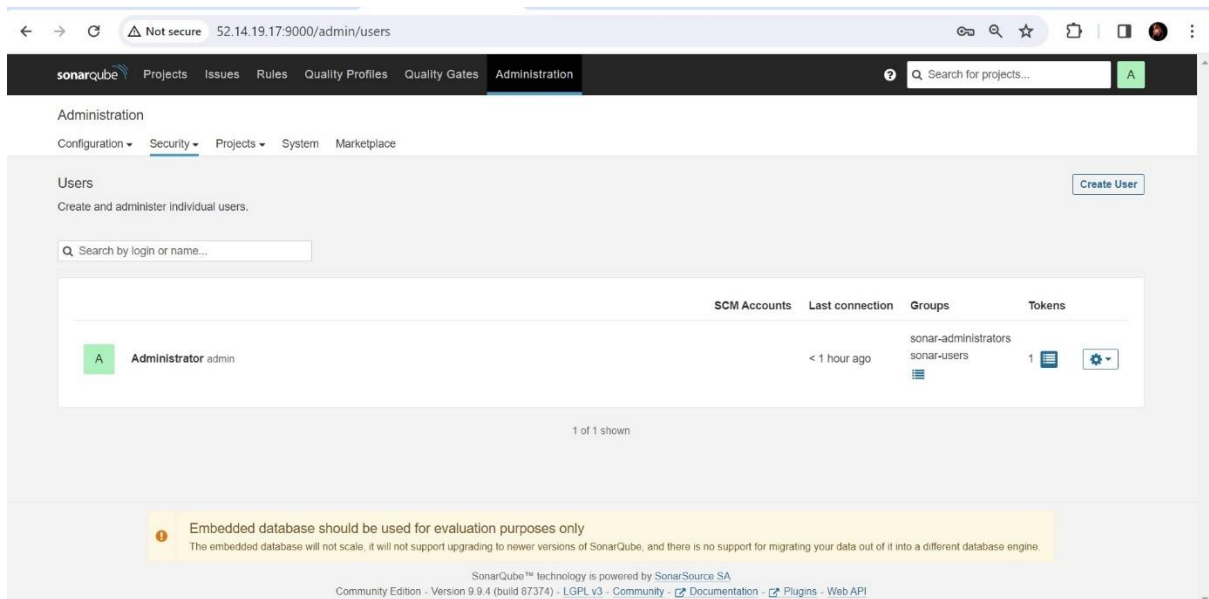
T	P	Store ↓	Domain	ID	Name
		System	(global)	sonar-token	sonar-token
		System	(global)	docker-cred	venkatesh09/***** (docker-cred)
		System	(global)	k8-cred	k8-cred
		System	(global)	Email-cred	singareddyv91@gmail.com/***** (Email-cred)

Stores scoped to Jenkins

P	Store ↓	Domains
---	---------	---------

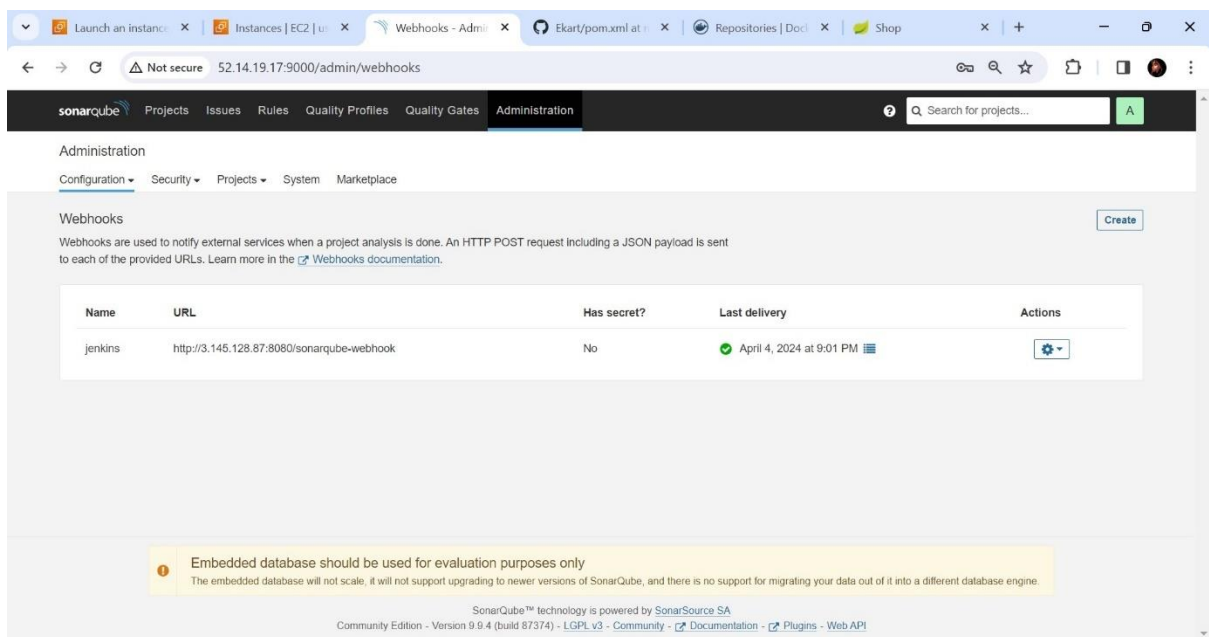
For SonarQube Integrate to Jenkins:

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



The screenshot shows the SonarQube Administration interface, specifically the 'Users' section. The breadcrumb trail is 'Administration > Security > Users'. A 'Create User' button is in the top right. Below the header, there's a search bar 'Search by login or name...'. A table lists users with columns: 'Name', 'Last connection', 'Groups', and 'Tokens'. One user is listed: 'Administrator admin' with a last connection of '< 1 hour ago', groups 'sonar-administrators' and 'sonar-users', and 1 token. A yellow warning box at the bottom states: 'Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.' The footer mentions 'SonarQube™ technology is powered by SonarSource SA' and 'Community Edition - Version 9.9.4 (build 87374) - LGPL v3 - Community - Documentation - Plugins - Web API'.

Name	Last connection	Groups	Tokens
Administrator admin	< 1 hour ago	sonar-administrators sonar-users	1



The screenshot shows the SonarQube Administration interface, specifically the 'Webhooks' section. The breadcrumb trail is 'Administration > Security > Webhooks'. A 'Create' button is in the top right. Below the header, there's a description: 'Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the [Webhooks documentation](#).' A table lists webhooks with columns: 'Name', 'URL', 'Has secret?', 'Last delivery', and 'Actions'. One webhook is listed: 'jenkins' with URL 'http://3.146.128.87:8080/sonarqube-webhook', 'Has secret?' set to 'No', and 'Last delivery' on 'April 4, 2024 at 9:01 PM'. A yellow warning box at the bottom is identical to the one in the previous screenshot. The footer is also identical.

Name	URL	Has secret?	Last delivery	Actions
jenkins	http://3.146.128.87:8080/sonarqube-webhook	No	April 4, 2024 at 9:01 PM	

For Nexus Integrate to Jenkins:

Go To Jenkins → Manage Jenkins → Managed Files → Set Up all these



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

Pipeline:

The Ultimate CI/CD Corporate DevOps Pipeline Project

pipeline {

agent any

tools {

jdk 'jdk17'

maven 'maven3'

}

environment {

SCANNER_HOME= tool 'sonar-scanner'

}

```
stages {  
    stage('Git Checkout') {  
        steps {  
            checkout scmGit(branches: [[name: '*/master']],  
extensions: [], userRemoteConfigs: [[url:  
'https://github.com/Singareddy-Venkatesh/Ekart.git']])  
        }  
    }  
    stage('Compile') {  
        steps {  
            sh 'mvn compile'  
        }  
    }  
    stage('Test') {  
        steps {  
            sh 'mvn test -DskipTests=true'  
        }  
    }  
    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 -DskipTests=true'  
    }  
}
```

```
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 -DskipTests=true"  
        }  
    }  
}  
  
stage('Build & Docker Image') {  
    steps {  
        script {
```

```
        withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {

            sh "docker build -t venkatesh09/ekart:latest -f
docker/Dockerfile ."

        }

    }

}

stage('Trivy') {

    steps {

        sh "trivy image venkatesh09/ekart:latest > trivy-
report.txt"

    }

}

stage('Docker Push Image') {

    steps {

        script {

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

                sh "docker push venkatesh09/ekart: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.yml"
      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>

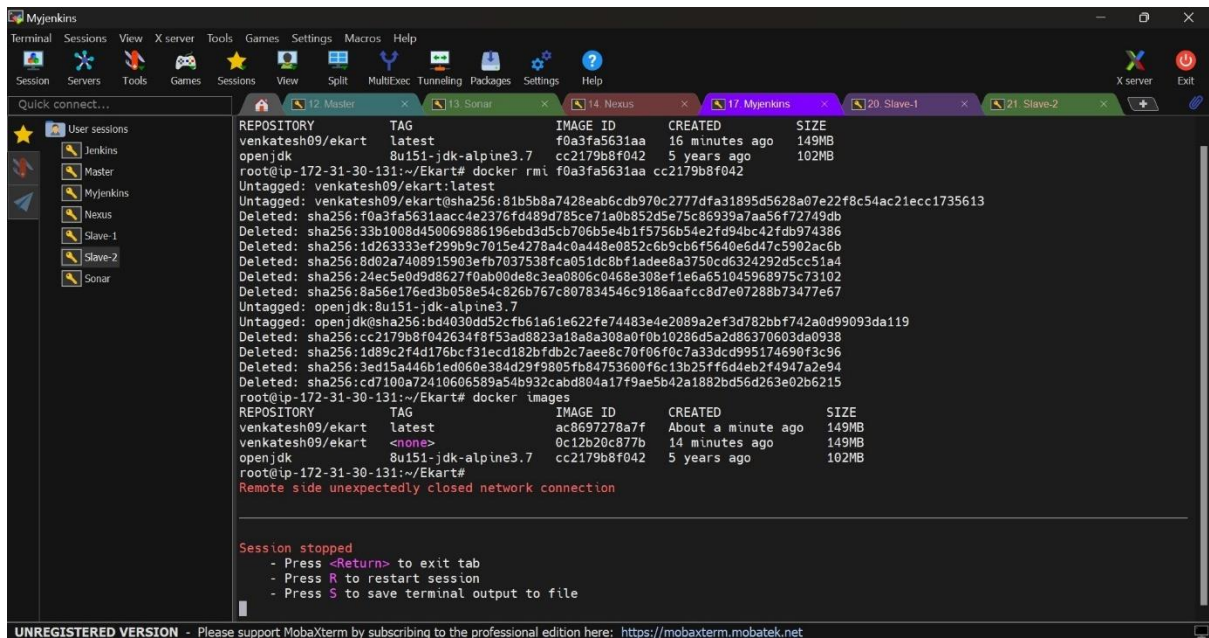
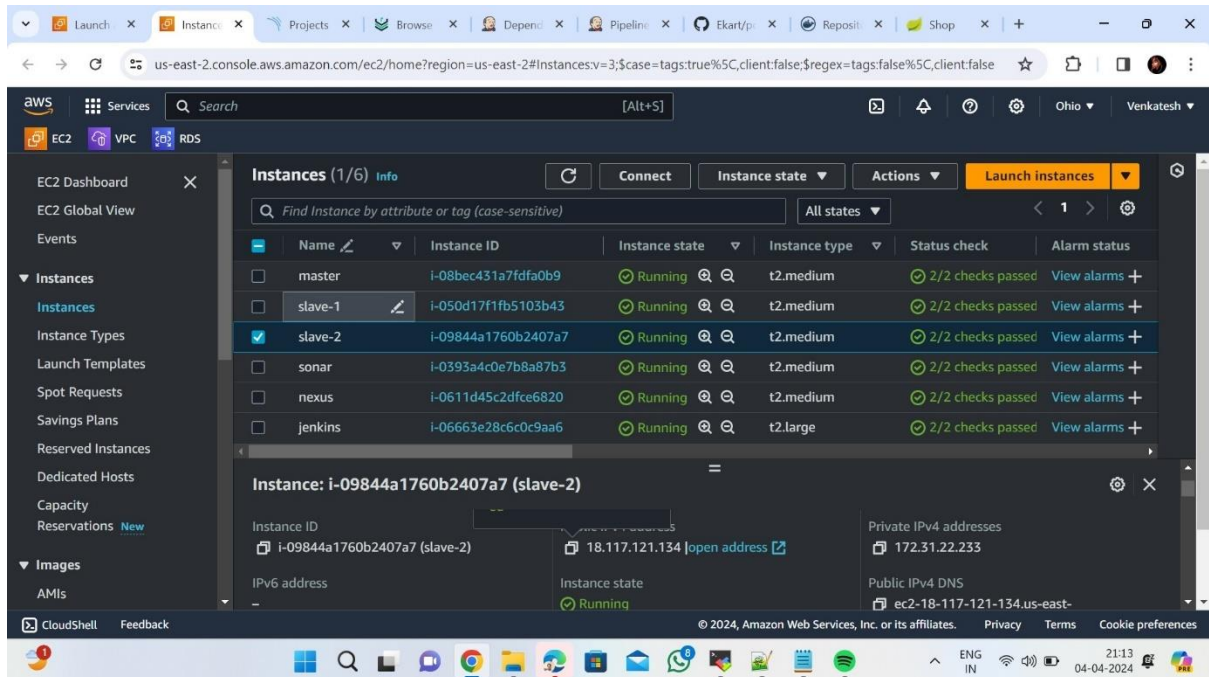
</html>

""

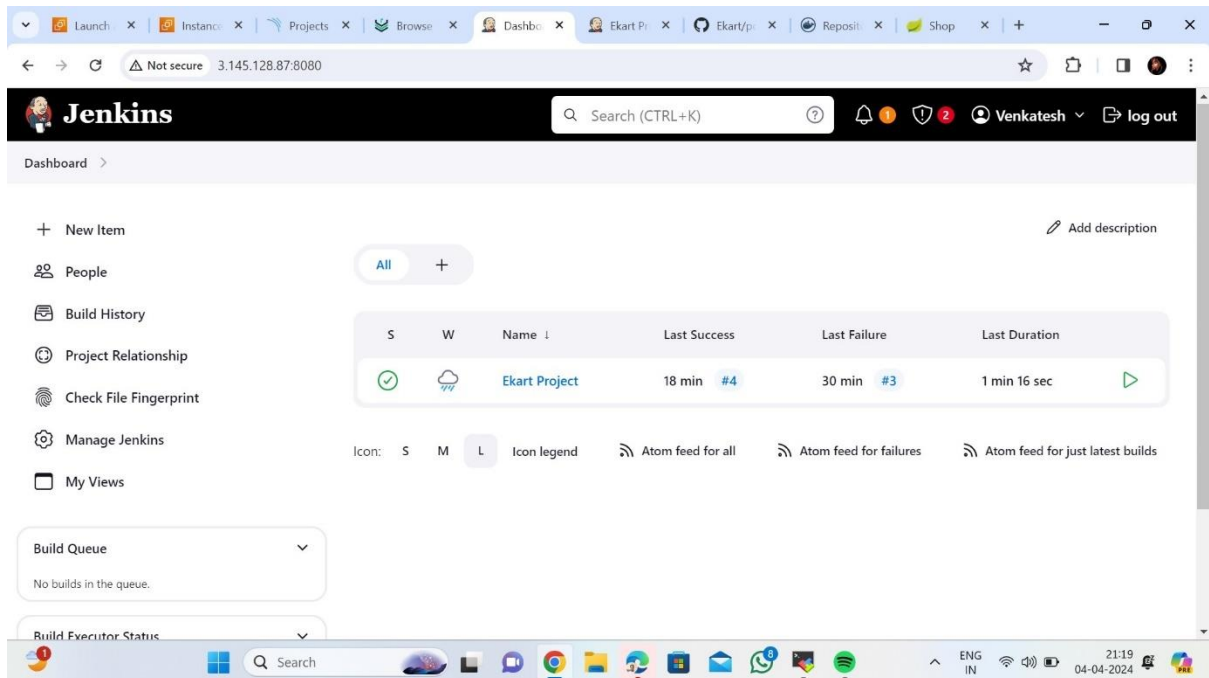
```

```
emailtext (  
    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-report.txt'  
)  
}  
}  
}  
}
```

Launches EC2 Instances:



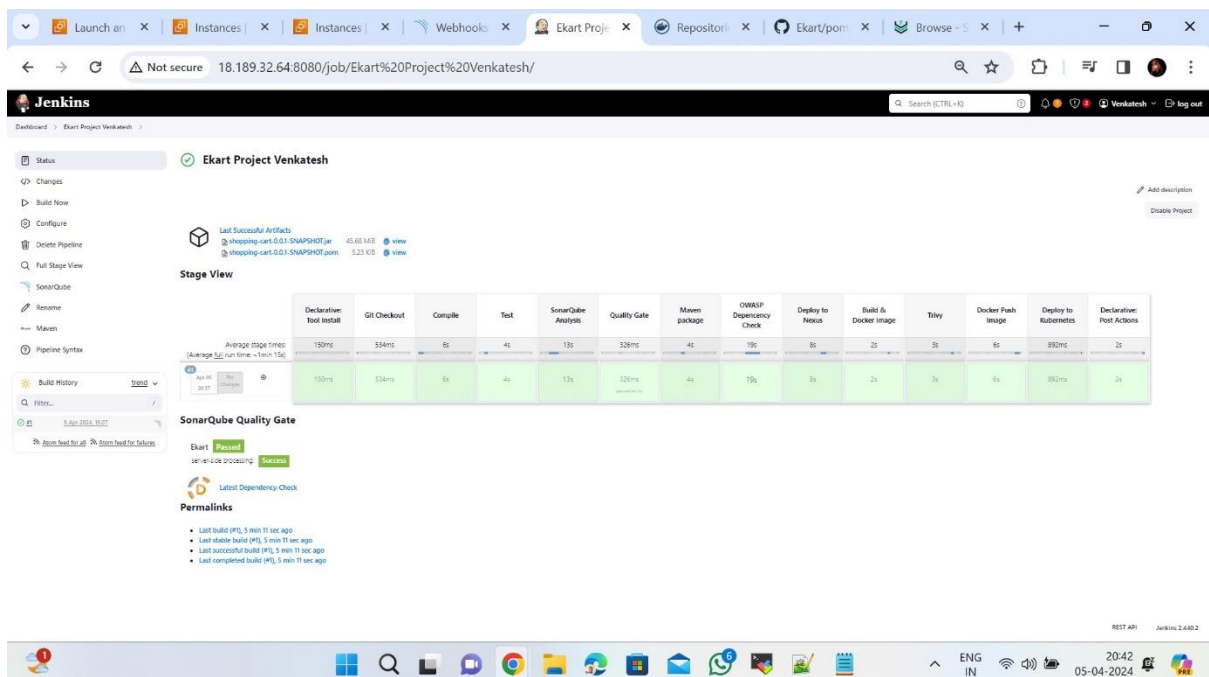
Created A NEW JOB:



The screenshot shows the Jenkins Dashboard. The top navigation bar includes a search bar, a user profile for 'Venkatesh', and a 'log out' button. The left sidebar contains links for 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. The main content area displays a table of builds for the 'Ekart Project'. The table has columns for 'S' (Status), 'W' (Webhook), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. The 'Ekart Project' build is shown with a status of 'S' (Success) and a duration of '1 min 16 sec'. Below the table, there are links for 'Icon: S M L', 'Icon legend', and 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'. A 'Build Queue' section shows 'No builds in the queue.' and a 'Build Executor Status' section is partially visible.

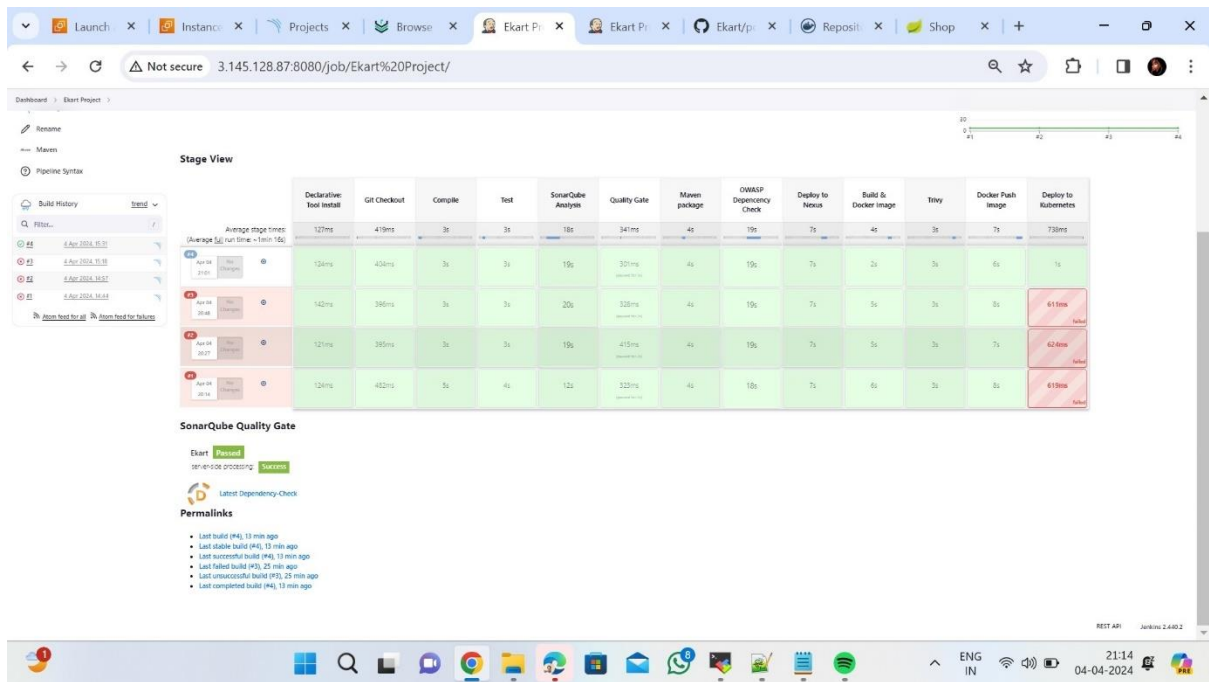
S	W	Name	Last Success	Last Failure	Last Duration
✓	☁	Ekart Project	18 min #4	30 min #3	1 min 16 sec

Successfully Runs the Pipeline:

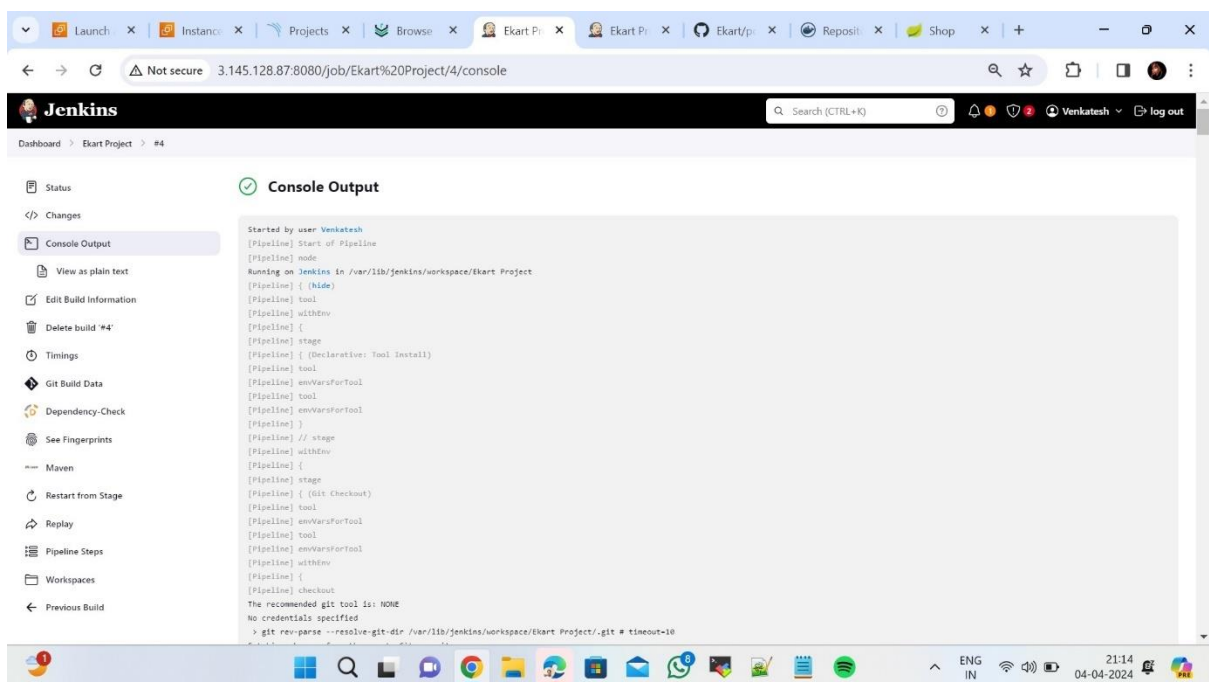


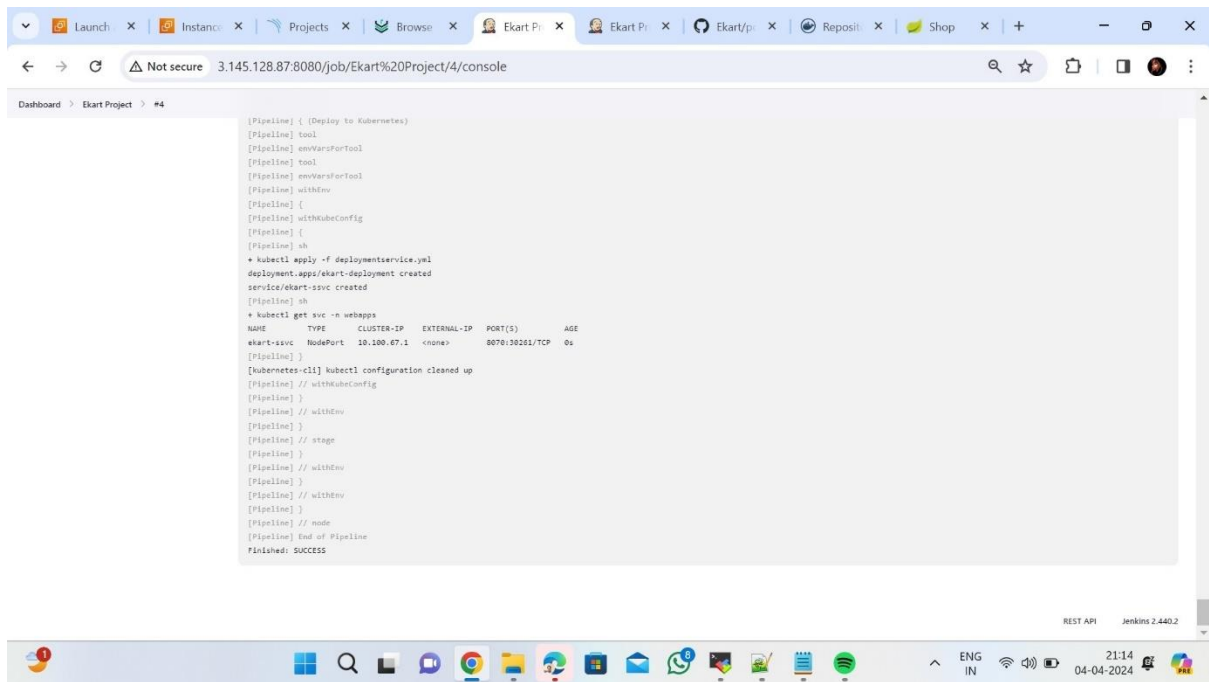
The screenshot shows the Jenkins Pipeline View for the 'Ekart Project Venkatesh'. The left sidebar contains links for 'Status', 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Full Stage View', 'SonarQube', 'Rename', 'Maven', and 'Pipeline Syntax'. The main content area displays the 'Stage View' of the pipeline. The stages are: 'Declarative: Tool install', 'Git Checkout', 'Compile', 'Test', 'SonarQube Analysis', 'Quality Gate', 'Maven package', 'OWASP Dependency Check', 'Deploy to Nexus', 'Build & Docker image', 'Tidy', 'Docker Push Image', 'Deploy to Kubernetes', and 'Declarative: Post Actions'. The 'Average stage times' are shown for each stage. Below the stage view, there is a 'SonarQube Quality Gate' section showing 'Ekart' as 'Passed' and 'Latest Dependency Check' as 'Success'. The 'Permalinks' section lists the last build, last stable build, last successful build, and last completed build, all with links to view the build details.

Stage	Time
Declarative: Tool install	150ms
Git Checkout	534ms
Compile	6s
Test	4s
SonarQube Analysis	13s
Quality Gate	326ms
Maven package	4s
OWASP Dependency Check	19s
Deploy to Nexus	6s
Build & Docker image	2s
Tidy	3s
Docker Push Image	6s
Deploy to Kubernetes	890ms
Declarative: Post Actions	2s

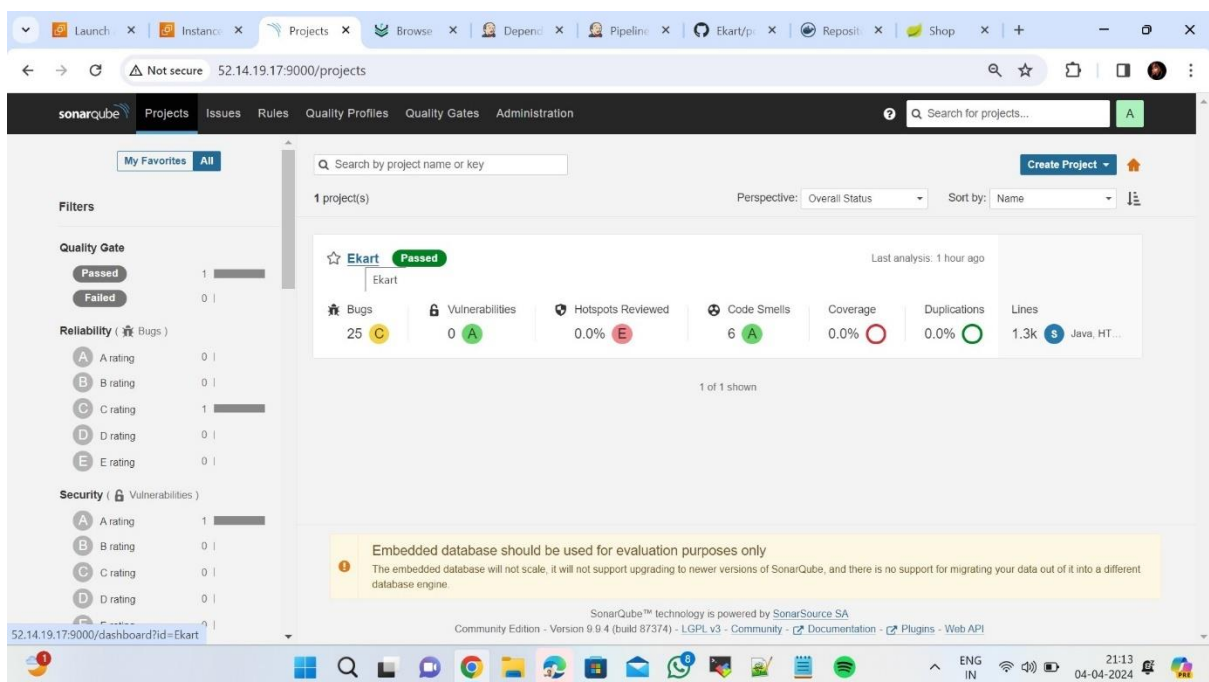


Console Output:

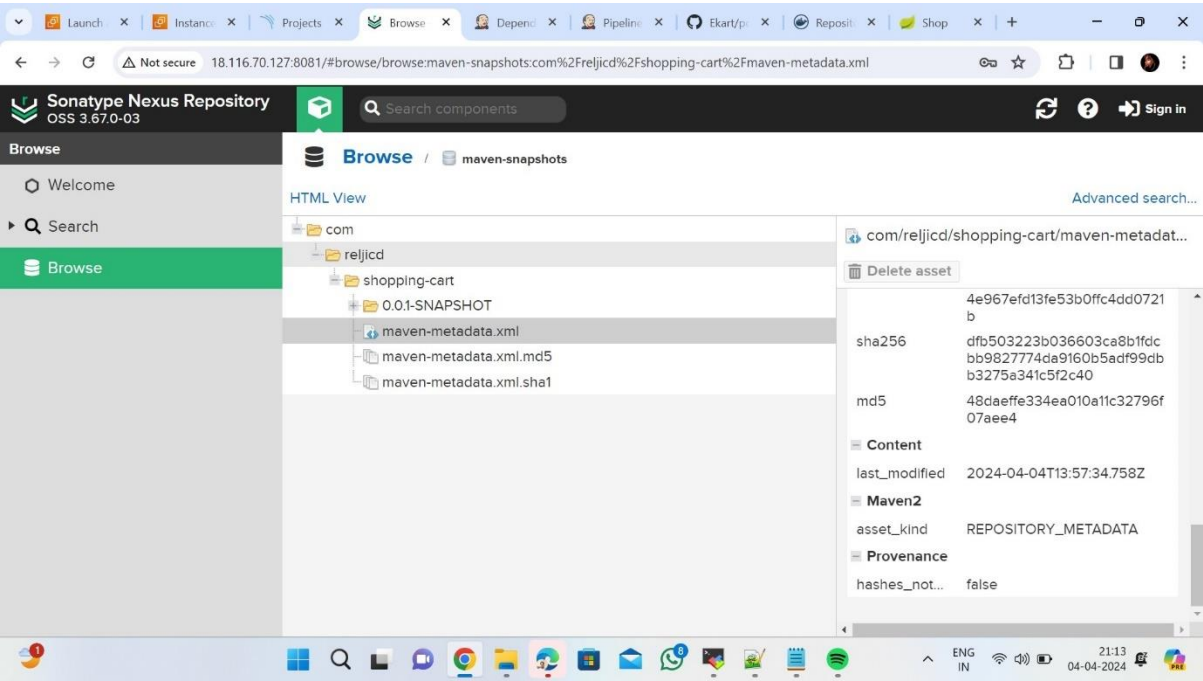




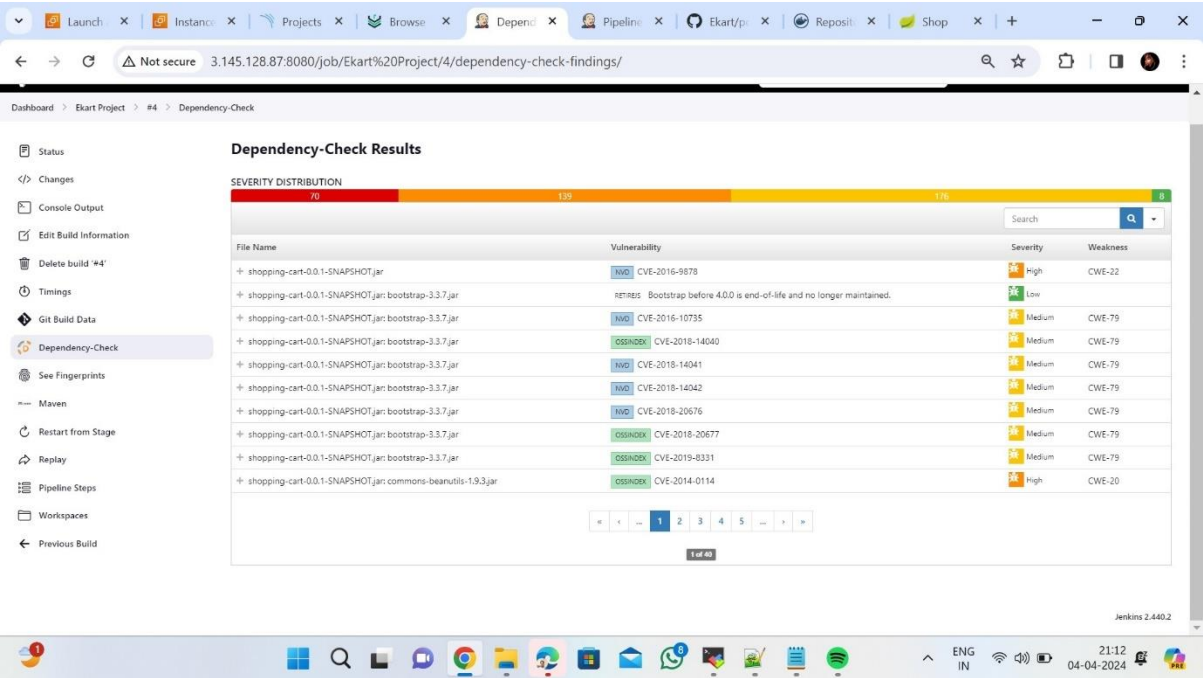
SonarQube Output:



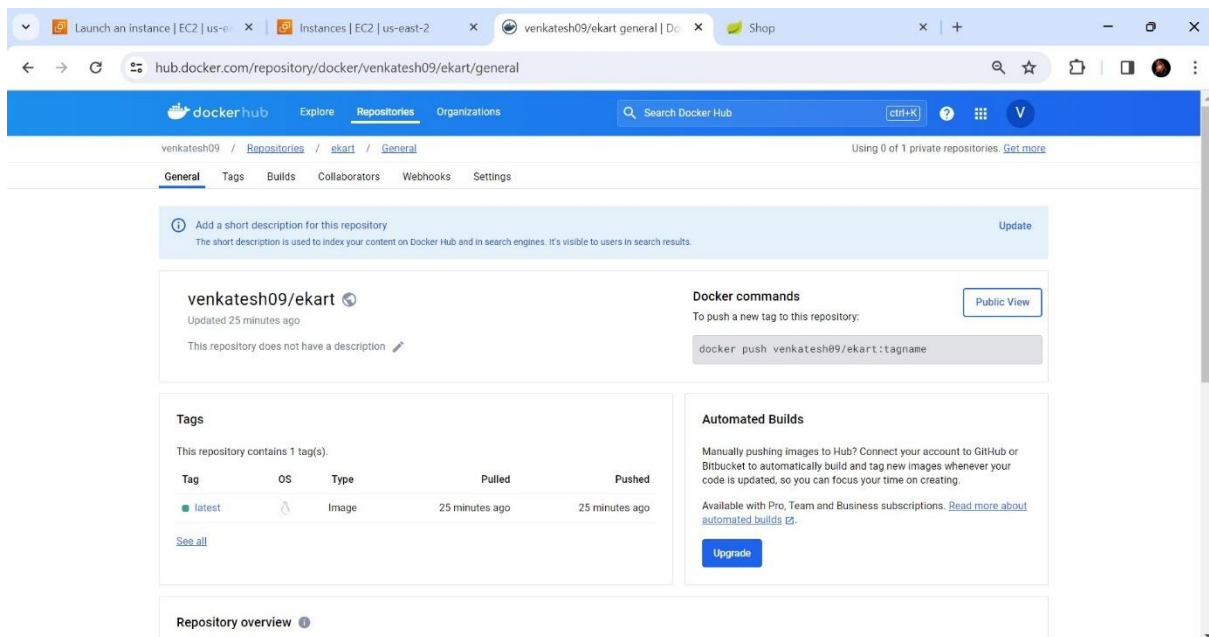
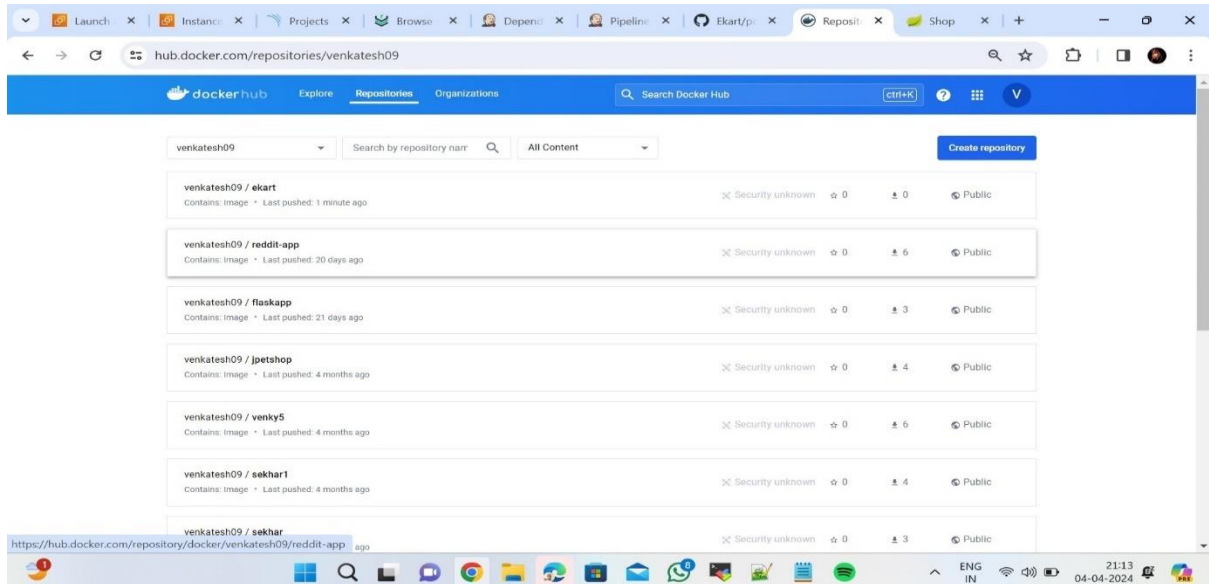
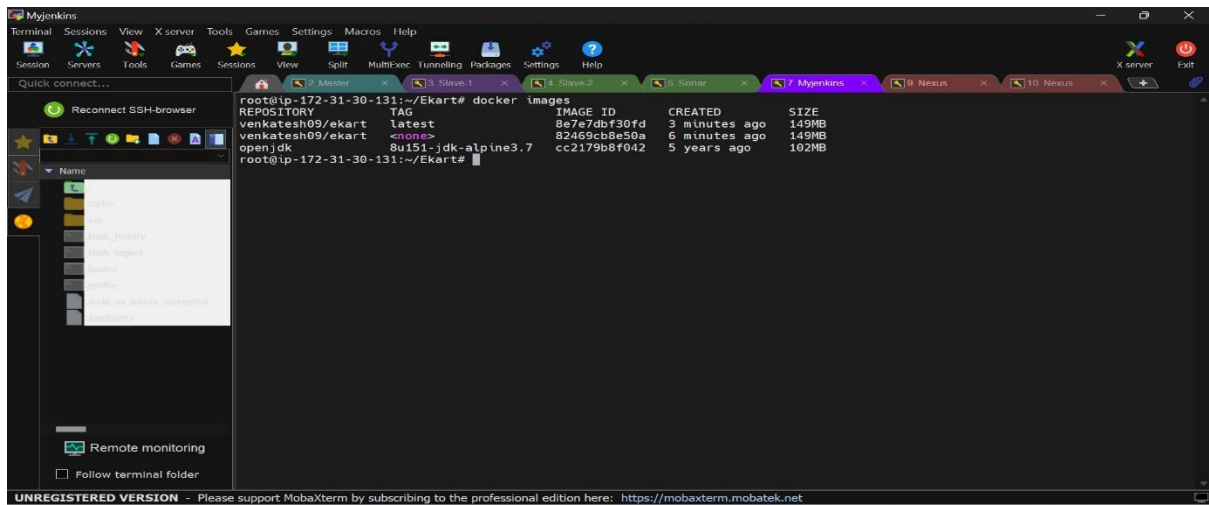
Nexus:



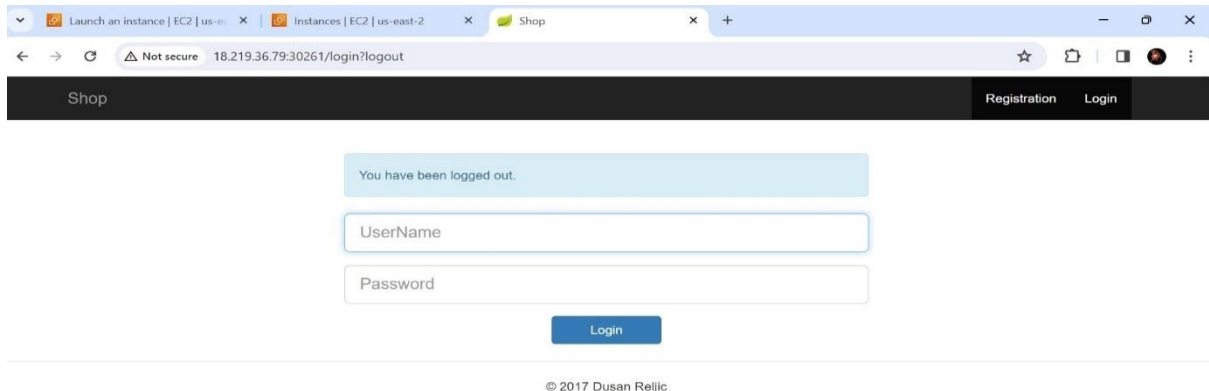
Dependency Check:



Docker:



Final Output:



Shop

Registration Login

You have been logged out.

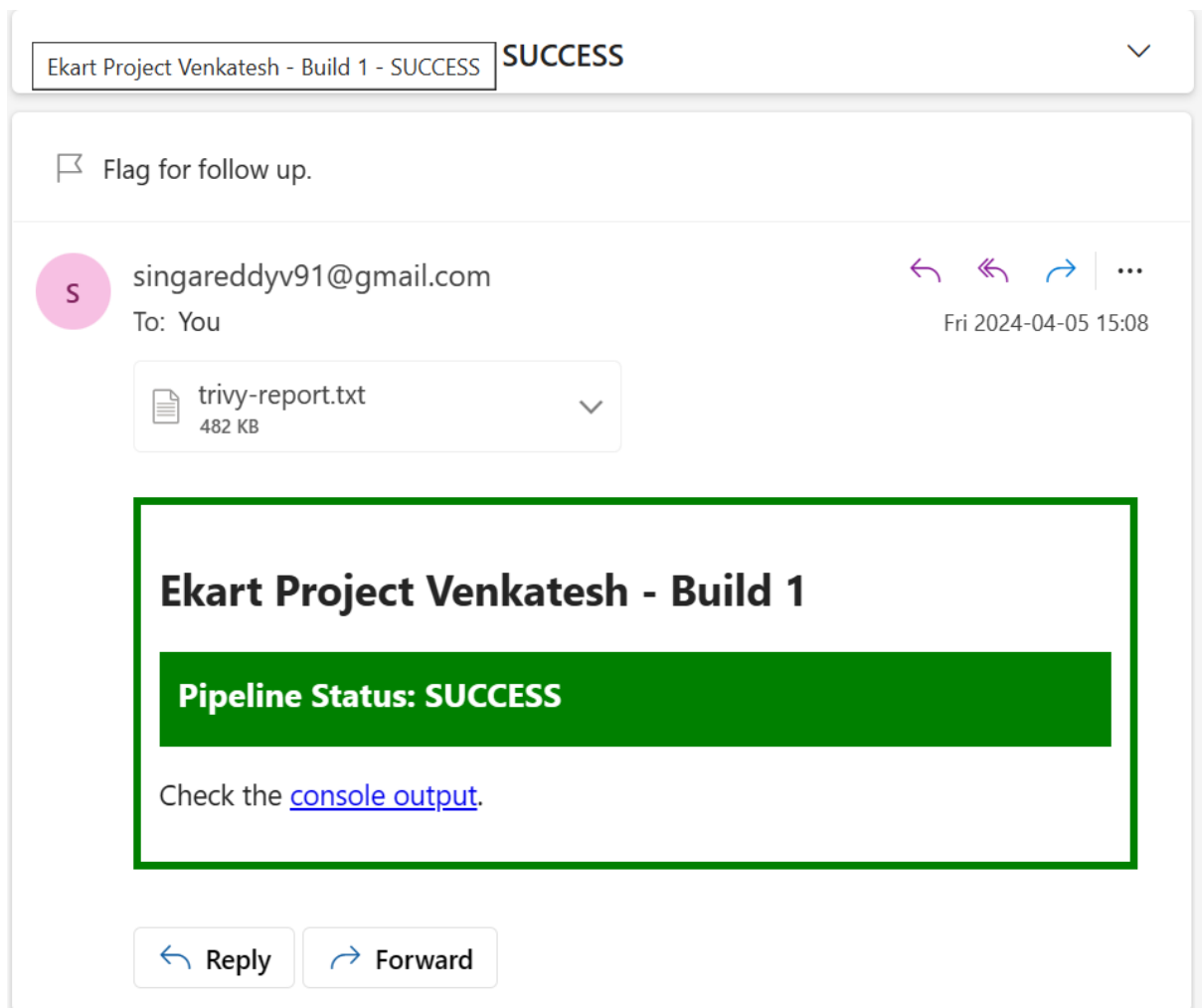
UserName

Password

Login

© 2017 Dusan Rejjic

Email Notification:



Ekart Project Venkatesh - Build 1 - SUCCESS SUCCESS

Flag for follow up.

singareddyv91@gmail.com
To: You

Fri 2024-04-05 15:08

trivy-report.txt
482 KB

Ekart Project Venkatesh - Build 1

Pipeline Status: SUCCESS

Check the [console output](#).

Reply Forward

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

Highly recommended 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