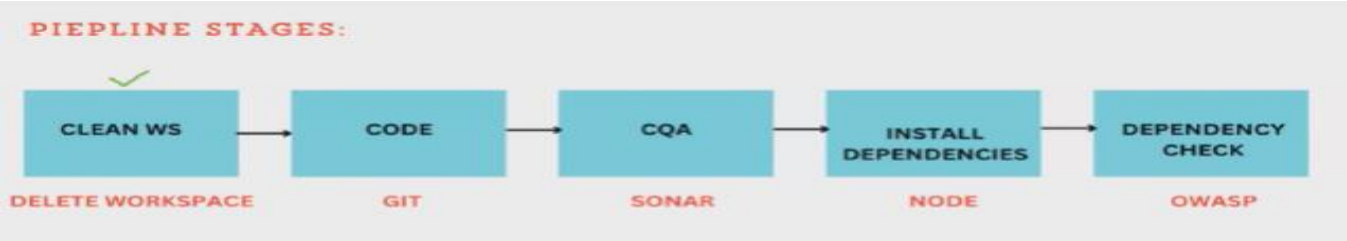
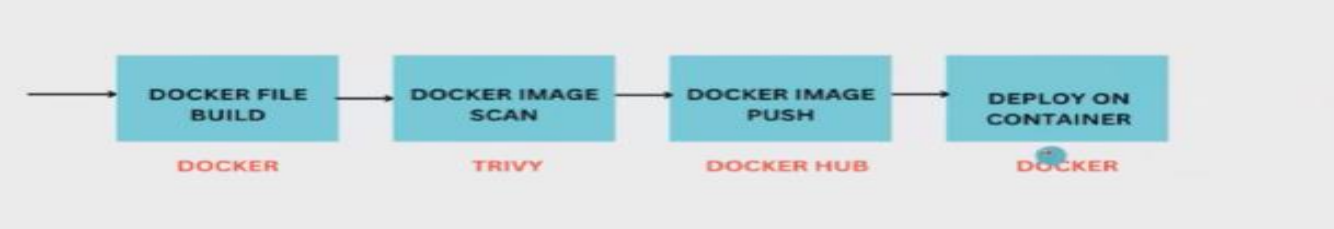
Here is a step-by-step list of the pipeline stages for deploying a React application:





Java name:jdk17 jdk-17.0.8.1 +1

Sonar name: mysonar Sonarqube Scanner 6.2.1.4610 sonar-password

nodeJS name: node16 16.2.0 version

dependency check owasp: dp-check Install from github.com dependency-check 6.5.1

In this project, we leverage Jenkins to create a robust CI/CD pipeline that integrates tools like **Docker**, **Trivy**, **SonarQube**, and **OWASP Dependency Check** to deliver secure and high-quality software.

Tools used:

1. GitHub
2. Jenkins
3. Docker
4. OWASP
5. Trivy
6. Sonarqube
7. NodeJS

GitHub

GitHub is a platform for hosting and sharing code using Git. It helps developers collaborate by tracking changes, managing versions, and reviewing code. You can create repositories (repos) to store and organize projects. It also offers features like issue tracking and CI/CD workflows.

Jenkins

Jenkins is an automation tool for building and deploying software. It uses pipelines to automate tasks like testing, building, and deploying code. It supports plugins to integrate with various tools. Jenkins makes Continuous Integration and Continuous Delivery (CI/CD) easy.

Docker

Docker is a tool to create, share, and run lightweight virtualized environments called containers. Containers package code and dependencies together for consistent behavior across systems. It simplifies app deployment and avoids "it works on my machine" issues. Docker images are reusable blueprints for containers.

OWASP

OWASP (Open Web Application Security Project) focuses on improving web application security. It provides resources like the **OWASP Top 10**, a list of the most critical security risks in web apps. Developers use it as a guide to build secure applications. OWASP also creates free tools for security testing.

Trivy

Trivy is a security tool that scans containers, filesystems, and code for vulnerabilities. It detects issues like outdated dependencies and misconfigurations. Easy to use, it integrates with CI/CD pipelines to ensure secure builds. Trivy helps identify security risks early in the development process.

SonarQube

SonarQube analyzes code to find bugs, vulnerabilities, and code smells (poor practices). It gives a detailed report on code quality and suggests improvements. It supports multiple languages and integrates with CI/CD tools. SonarQube helps teams maintain clean, reliable, and secure code.

Node.js

Node.js is a runtime that lets you run JavaScript outside the browser. It's great for building fast and scalable server-side applications. Node.js uses non-blocking, event-driven programming for high performance. It's commonly used for web servers, APIs, and real-time apps.

Key Highlights:

1. **Code Quality Assurance**:
    - **SonarQube** is used to analyze source code, ensuring it adheres to coding standards and detecting bugs, code smells, and vulnerabilities.
2. **Vulnerability Scanning**:

- **Trivy** scans container images for known vulnerabilities, helping to secure the application at the image level.
- **OWASP Dependency Check** analyzes dependencies for vulnerabilities, ensuring no insecure libraries or frameworks are introduced.

3. **Dockerized Environments**:
    - All stages of the pipeline run in Dockerized containers, ensuring consistency and simplifying environment management.
4. **CI/CD Workflow**:
    - The pipeline automates **build**, **test**, and **deploy** processes, making it efficient and reliable.
    - Security and quality checks are seamlessly integrated into the pipeline, ensuring every release is both secure and robust.

GitHub Repo:

GitHub - devops0014/Zomato-Project

Contribute to devops0014/Zomato-Project development by creating an account on GitHub.

github.com

Let's implement it step-by-step,

STEP-1: Launch EC2 Instance with below Configurations

1. AMI: Amazon Linux Kernel 5.10
2. Instance Type: T2. Large
3. EBS Volume: 20+ GB
4. Security Groups: All Traffic

STEP-2: Install Jenkins

Copy

Copy

#! /bin/bash

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

yum install java-17-amazon-corretto -y

yum install jenkins -y

systemctl start jenkins.service

systemctl enable jenkins.service

systemctl status jenkins.service

STEP-3: Install Docker & GIT

Copy

Copy

yum install docker git -y

systemctl start docker

systemctl status docker

chmod 777 ///var/run/docker.sock

STEP-4: Install Sonarqube

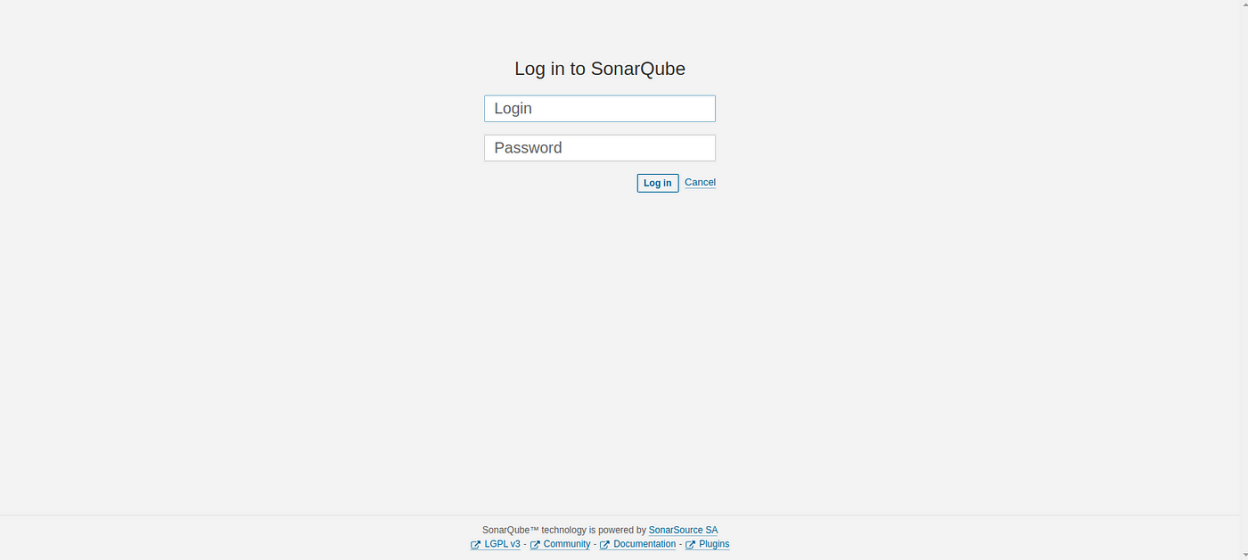
Copy

Copy

docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

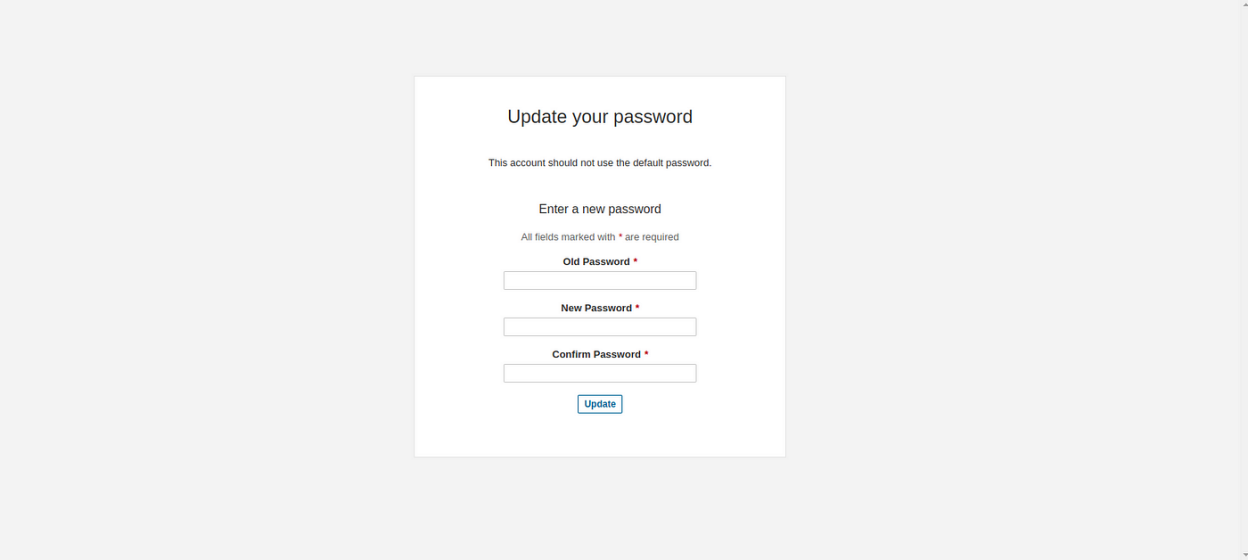
Now our sonarqube is up and running

Now our sonarqube is up and running on Port :9000

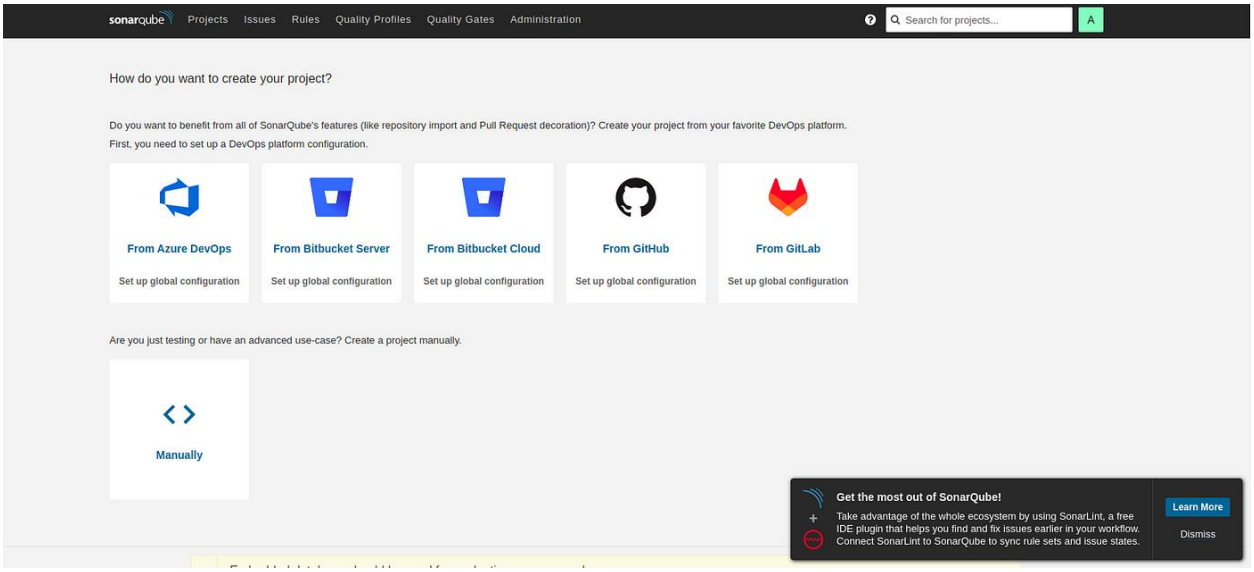


username - admin
password - admin

Now our sonarqube is up and running



Update New password, This is Sonar Dashboard.



STEP-5: Install Trivy

Copy

Copy

wget https://github.com/aquasecurity/trivy/releases/download/v0.18.3/trivy_0.18.3_Linux-64bit.tar.gz

tar zxvf trivy_0.18.3_Linux-64bit.tar.gz

sudo mv trivy /usr/local/bin/

echo 'export PATH=$PATH:/usr/local/bin/' >> ~/.bashrc

source .bashrc

trivy --version

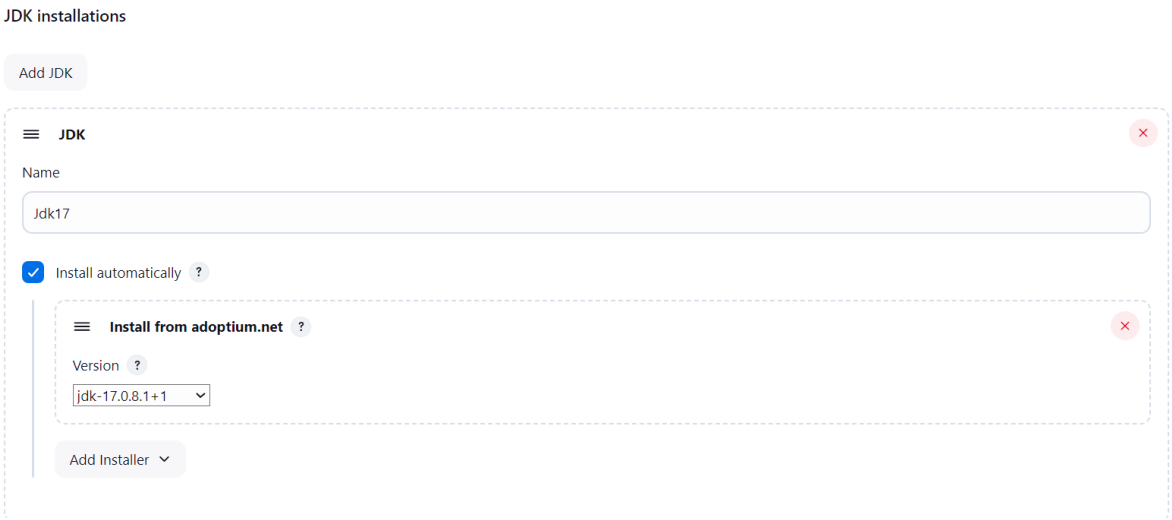
STEP-6: Install the following dependencies on Jenkins

1. Sonarqube Scanner
2. Eclipse Temurin Installer
3. NodeJS
4. OWASP Dependency-Check
5. Docker Pipeline
6. Blue Ocean
7. Pipeline Stage View

STEP-7: Integrate all the tools to Jenkins

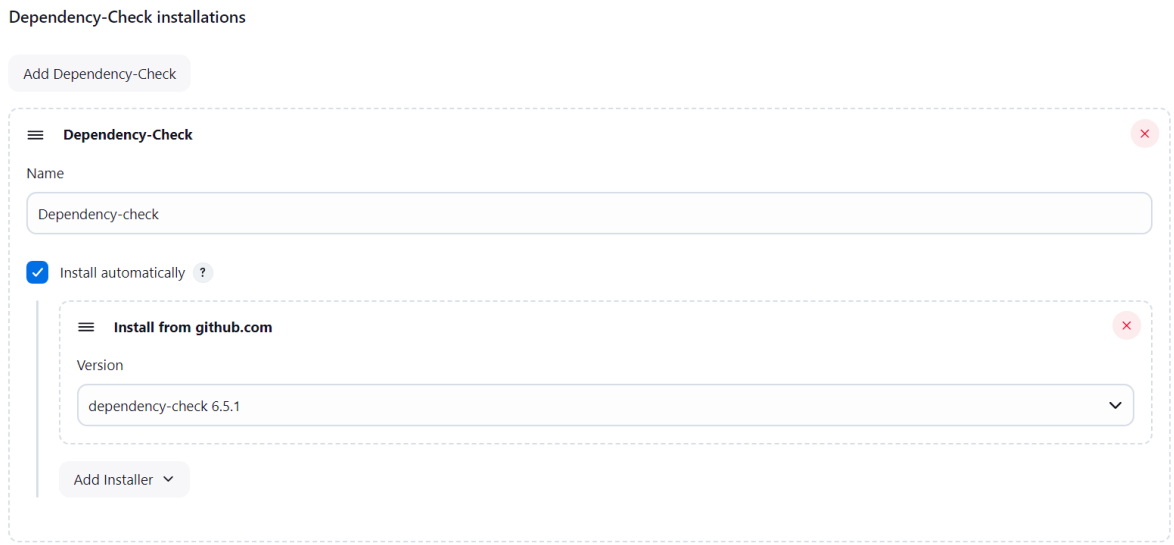
Go to Jenkins Dashboard → Manage Jenkins → Tools

JDK installations — v17

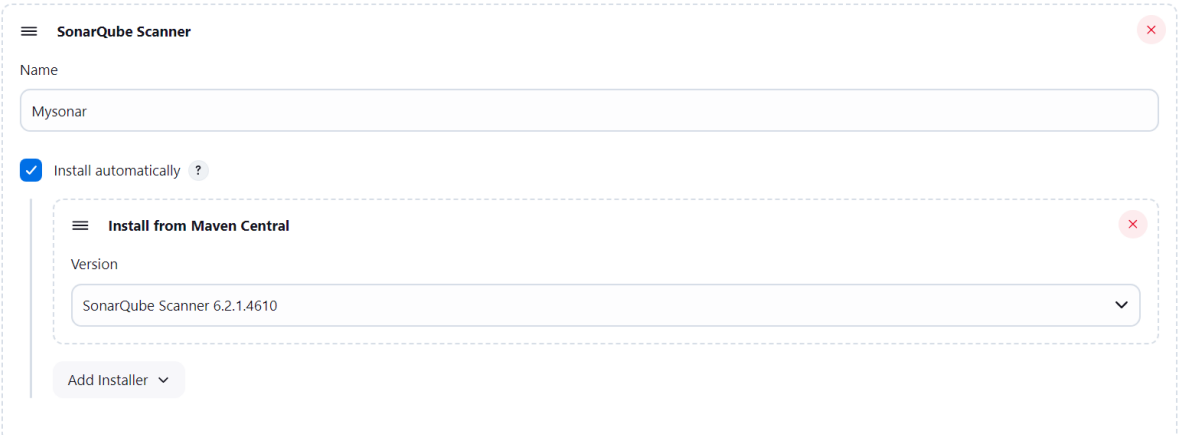


NodeJS installations — v 16

Dependency-Check installations



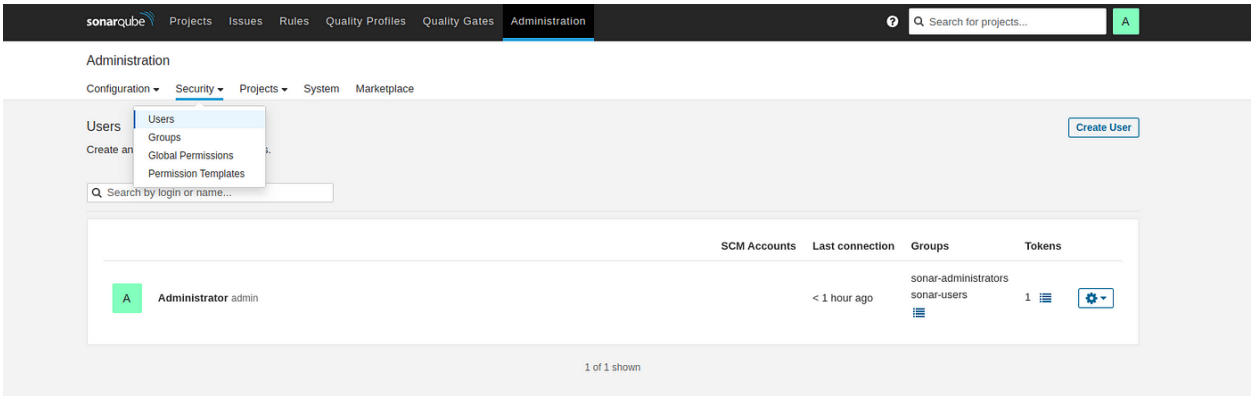
SonarQube Scanner installations



After that click on apply

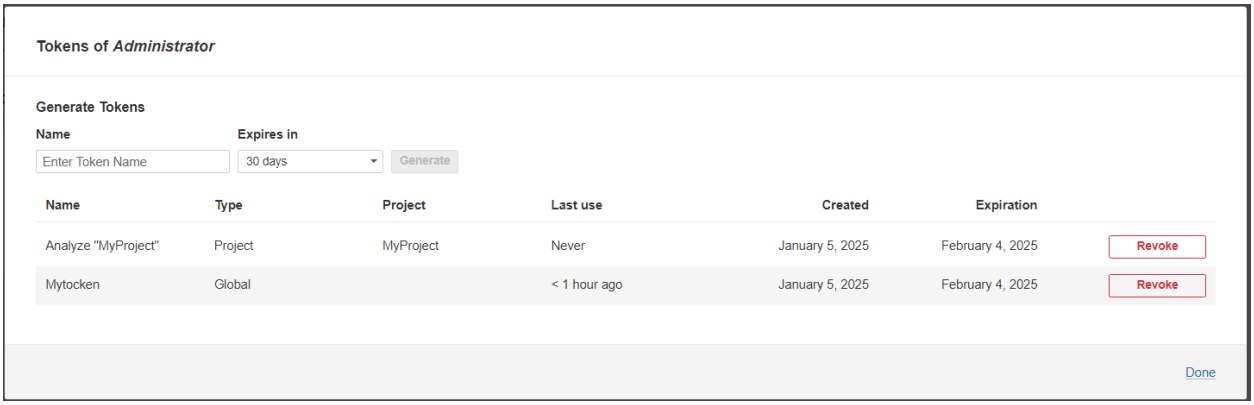
STEP-8: Configure Sonar with Jenkins

Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

Create a token with a name and generate and copy the tocken



Goto Jenkins Dashboard → Manage Jenkins → Credentials → System → Global credentials (unrestricted) → Add Secret Text

Enter detail and click on create

Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

Click on Apply and Save

STEP-9: Add Dockerhub Credentials

Goto Jenkins Dashboard → Manage Jenkins → Credentials → System → Global credentials (unrestricted) → Add Username and password

username : dockerhub-username

password: dockerhub-password

credentials id : docker-password

Click on Apply and Save

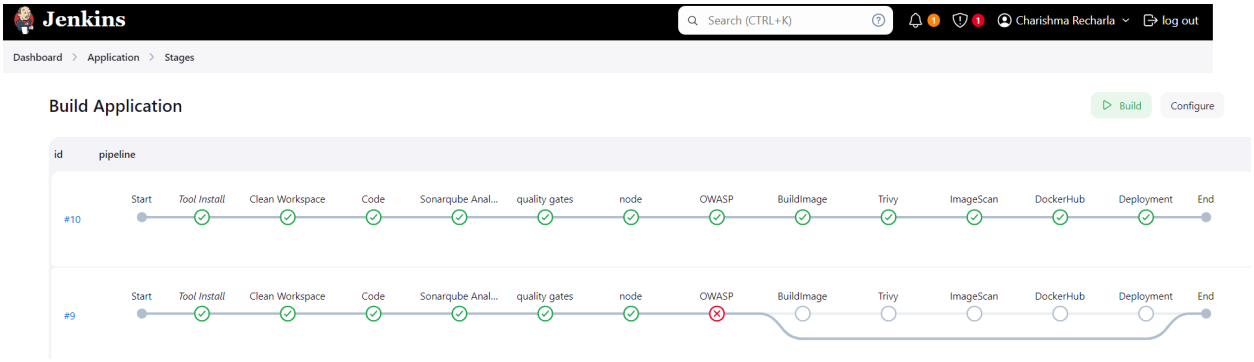
STEP-10: Create a Jenkins Job

Goto Jenkins Dashboard create a job as My-Deployment Name, select pipeline and click on ok. pipeline {

```
pipeline {
agent any
tools {
   jdk ' Jdk17'
   nodejs 'Node16'
}
environment {
   SCANNER_HOME = tool 'Mysonar'
}
stages {
   stage("Clean Workspace") {
      steps {
         cleanWs()
      }
   }
   stage("Code") {
      steps {
         git "https://github.com/rohit23106/Zomato-Project.git"
      }
   }
   stage("Sonarqube Analysis") {
      steps {
         withSonarQubeEnv('Mysonar') {
            sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=zomato \
            -Dsonar.projectKey=zomato '''
         }
```

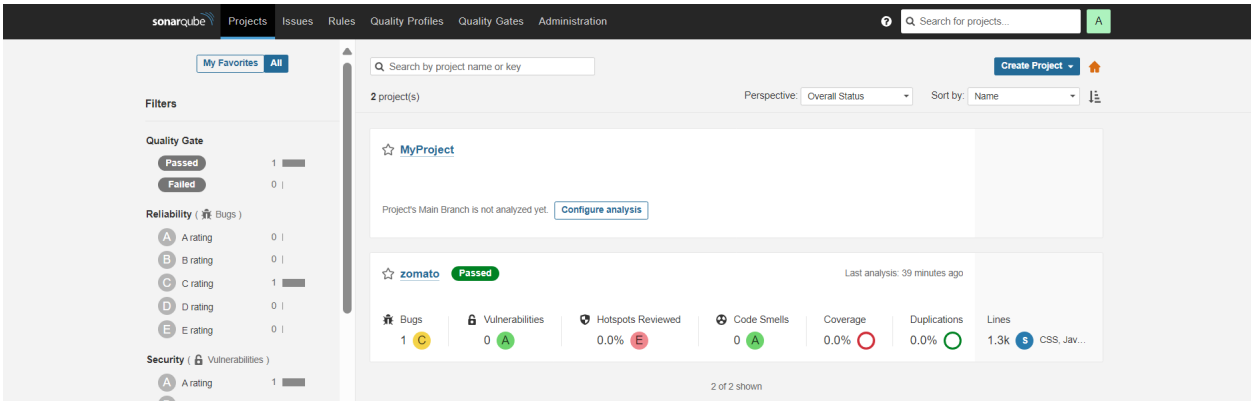
```
    }
}
stage("quality gates") {
    steps {
        script {
            waitForQualityGate abortPipeline: false, credentialsId: ' Sonar-Password'
        }
    }
}
stage("node") {
    steps {
        sh 'npm install'
    }
}
stage("OWASP") {
    steps {
         dependencyCheck additionalArguments: '--scan ./', odcInstallation: ' Dependency-check'
         dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
    }
}
stage("BuildImage") {
    steps {
        sh 'docker build -t image1 .'
    }
}
stage("Trivy") {
    steps {
          sh 'trivy fs . >> trivyfs.txt'
    }
```

```
        }
        stage("ImageScan") {
           steps {
            sh "trivy image image1"
           }
        }
        stage("DockerHub") {
           steps {
             script {
                  withDockerRegistry(credentialsId: 'docker-password')
                  {
                  sh 'docker tag image1 charishma179/dockerdeployment:v1'
                  sh 'docker push charishma179/dockerdeployment:v1'
                  }
               }
           }
        }
        stage("Deployment") {
           steps {
              sh 'docker run -itd --name cont1 -p 9999:3000 charishma179/dockerdeployment:v1'
           }
        }
     }
}
```

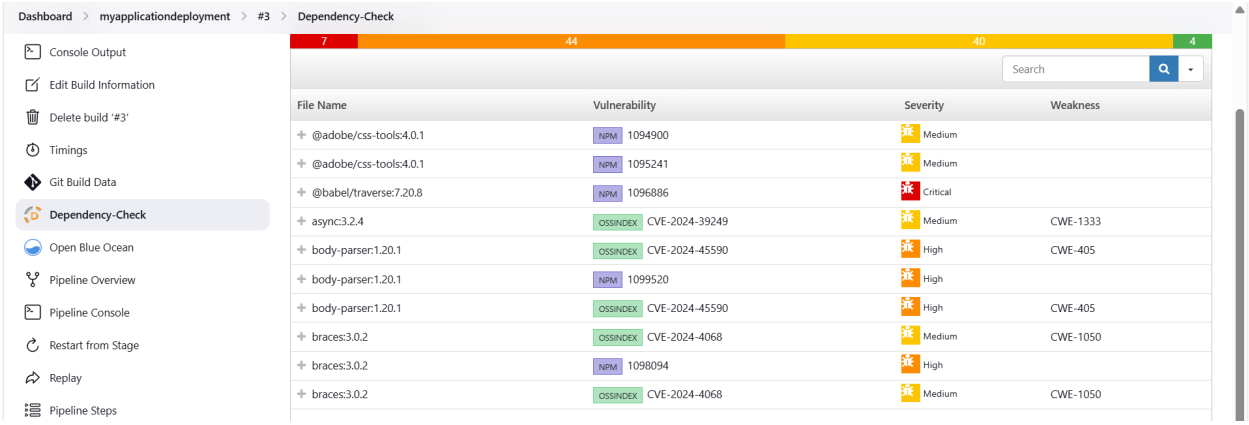


Declarative: Tool Install	Clean Workspace	Code	Sonarqube Analysis	quality gates	node	OWASP	BuildImage	Trivy	ImageScan	DockerHub	Deployment
11s	785ms	2s	24s	535ms	30s	5min 20s	56s	1s	34s	14s	520ms
200ms	361ms	996ms	22s	315ms	19s	10min 40s	1min 52s	2s	1min 7s	27s	747ms
23s	1s	3s	26s	755ms	40s	1s failed	318ms failed	320ms failed	256ms failed	279ms failed	294ms failed

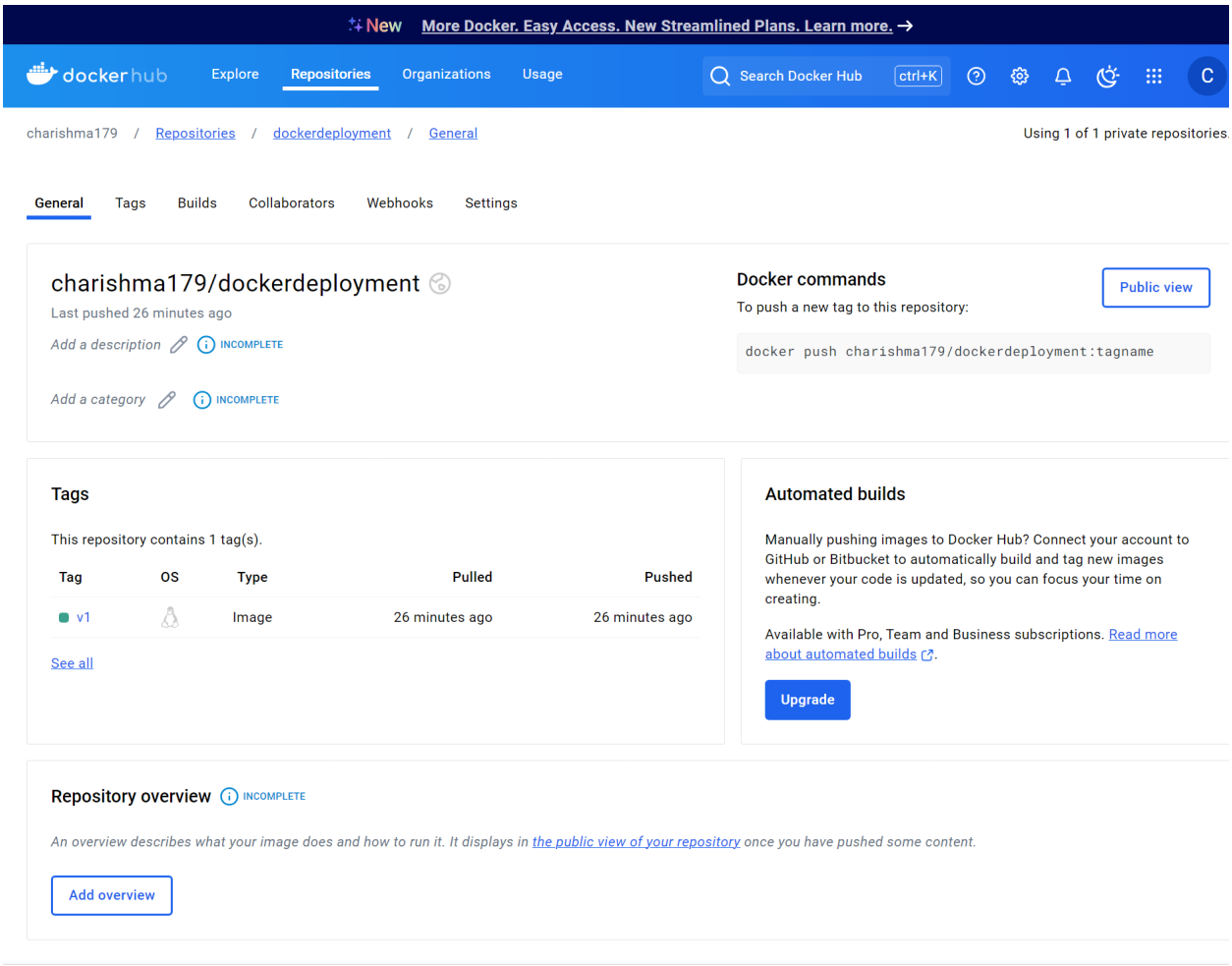
Sonarqube report



Dependency-Check Results



Docker hub registry image



Deployed website

React App