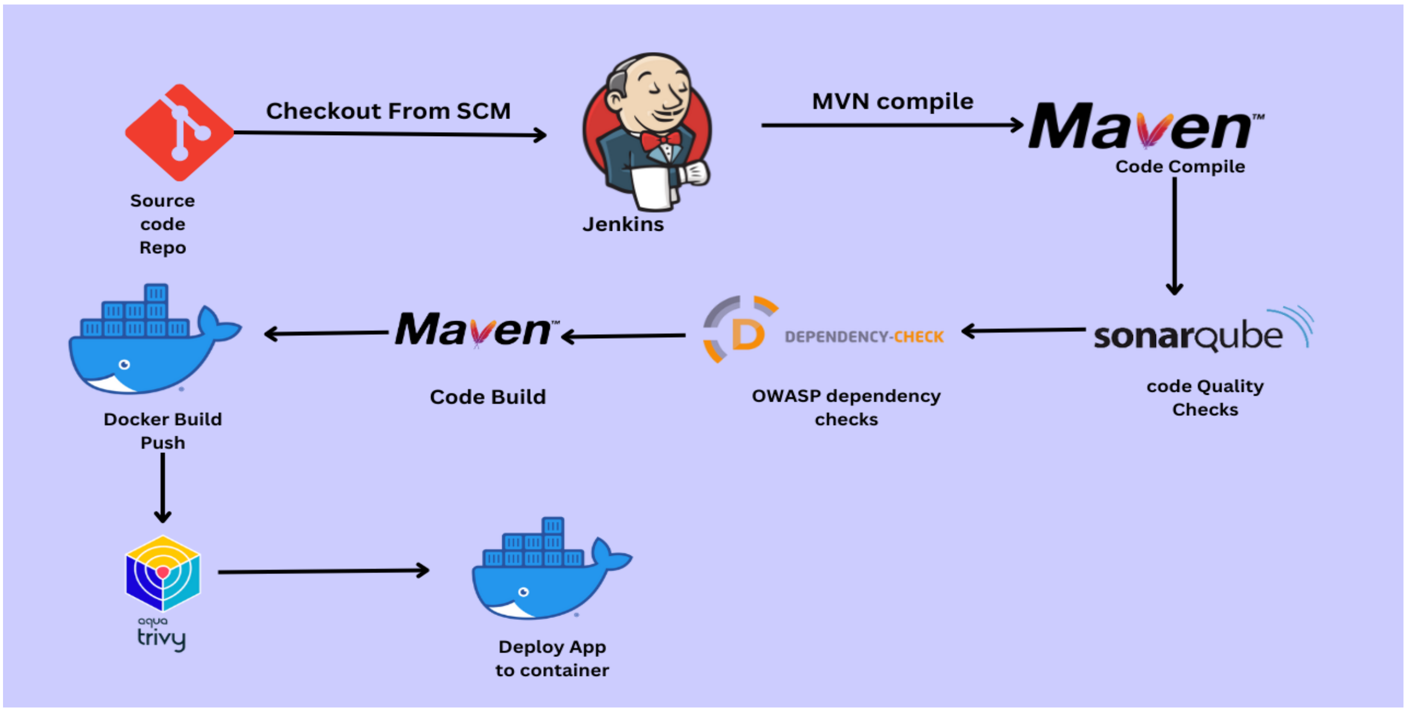
We will be using Jenkins as a CICD tool and deploying our application on Docker container.



We will be deploying our application in two ways, using Docker Container and other is using Tomcat Server.

Steps:-

Step 1 — Create an **Ubuntu T2 Large** Instance

Step 2 — Install **Jenkins, Docker and Trivy**. Create a **Sonarqube Container using Docker**.

Step 3 — Install Plugins like **JDK, Sonarqube Scanner, Maven, OWASP Dependency Check**,

Step 4 — Create a Pipeline Project in Jenkins using **Declarative Pipeline**

Step 5 — Install **OWASP Dependency Check Plugins**

Step 6 — Docker Image Build and Push

Step 7 — Deploy image using Docker

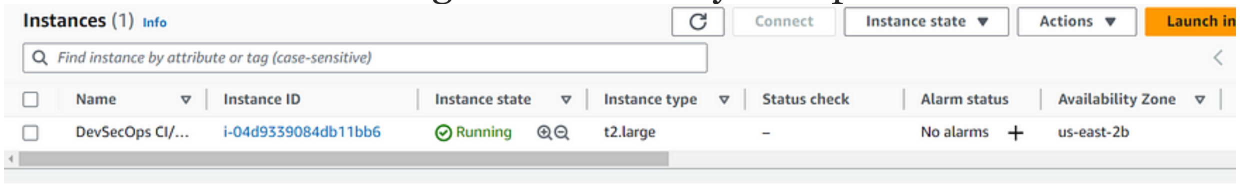
Step 8 — Access the Real World Application

Step 9 — Terminate the AWS EC2 Instance

References

Now, lets get started and dig deeper into each of these steps :-

Step 1 — Launch an AWS T2 Large Instance. Use the image as Ubuntu. You can create a new key pair or use an existing one. Enable HTTP and HTTPS settings in the Security Group.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
DevSecOps CI/...	i-04d9339084db11bb6	Running	t2.large	-	No alarms	us-east-2b

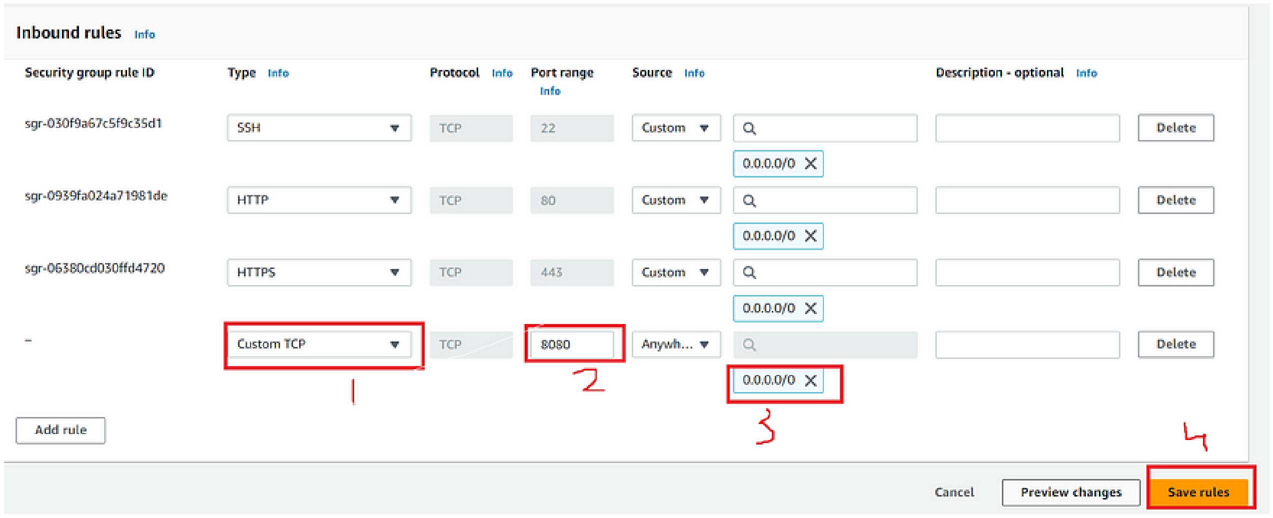
Step 2 — Install Jenkins, Docker and Trivy

2A — To Install Jenkins

Connect to your console, and enter these commands to Install Jenkins

```
sudo apt-get update

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo
tee \
/usr/share/keyrings/jenkins-keyring.asc >/dev/null
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 update
sudo apt install openjdk-17-jdk
sudo apt install openjdk-17-jre

sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins

sudo cat  /var/lib/jenkins/secrets/initialAdminPassword
```

Once Jenkins is installed, you will need to go to your AWS EC2 Security Group and open Inbound Port 8080, since Jenkins works on Port 8080.

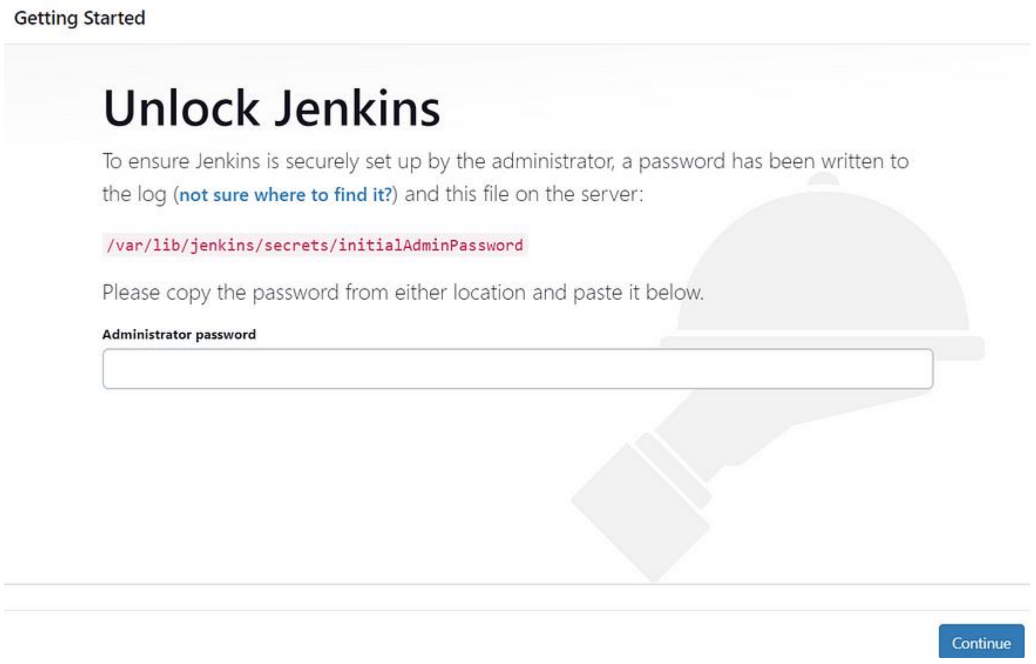

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-030f9a67c5f9c35d1	SSH	TCP	22	Custom	0.0.0.0/0	
sgr-0939fa024a71981de	HTTP	TCP	80	Custom	0.0.0.0/0	
sgr-06380cd030ffd4720	HTTPS	TCP	443	Custom	0.0.0.0/0	
-	Custom TCP	TCP	8080	Anywh...	0.0.0.0/0	

Now, grab your Public IP Address

```
<EC2 Public IP Address:8080>
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Unlock Jenkins using an administrative password and install the required plugins.


Getting Started

Unlock Jenkins

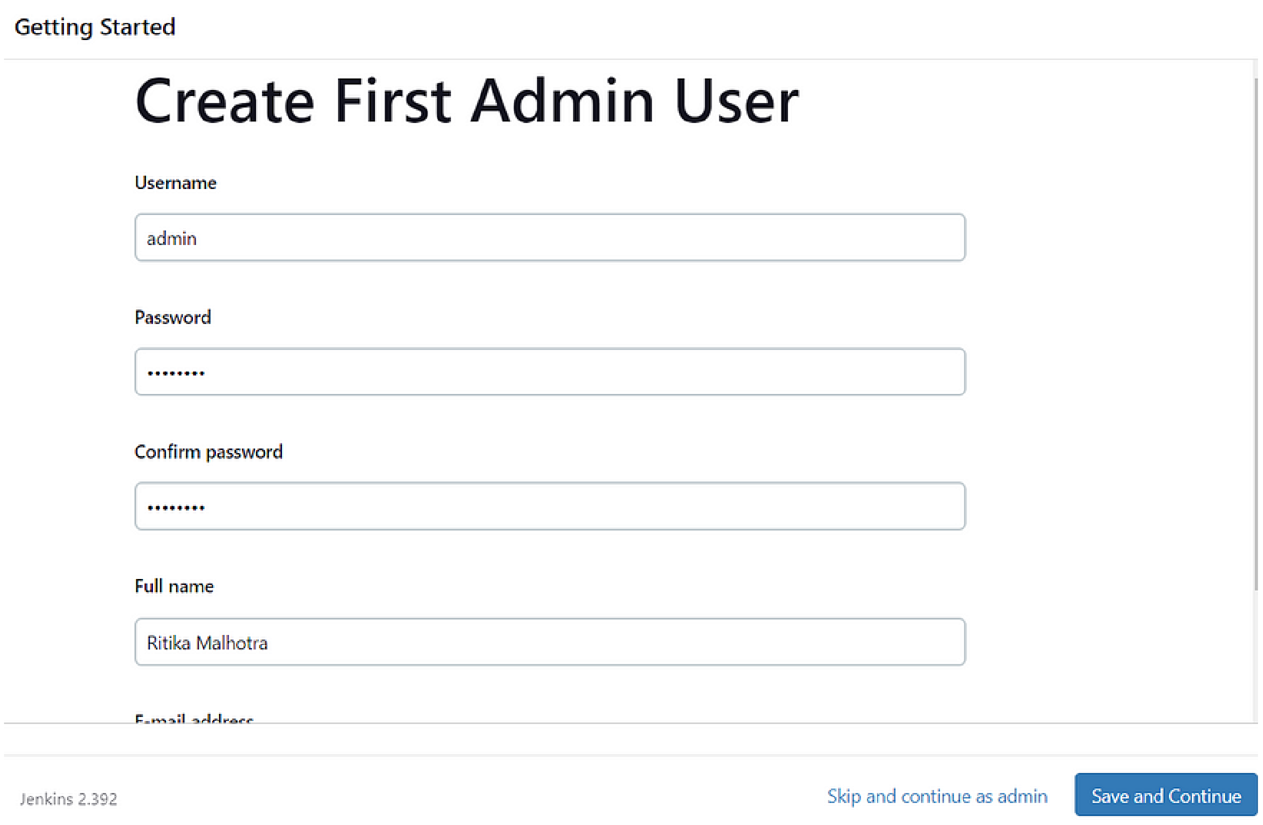
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Jenkins will now get installed and install all the libraries.


Getting Started

Create First Admin User

Username: admin

Password: ••••••••

Confirm password: ••••••••

Full name: Ritika Malhotra

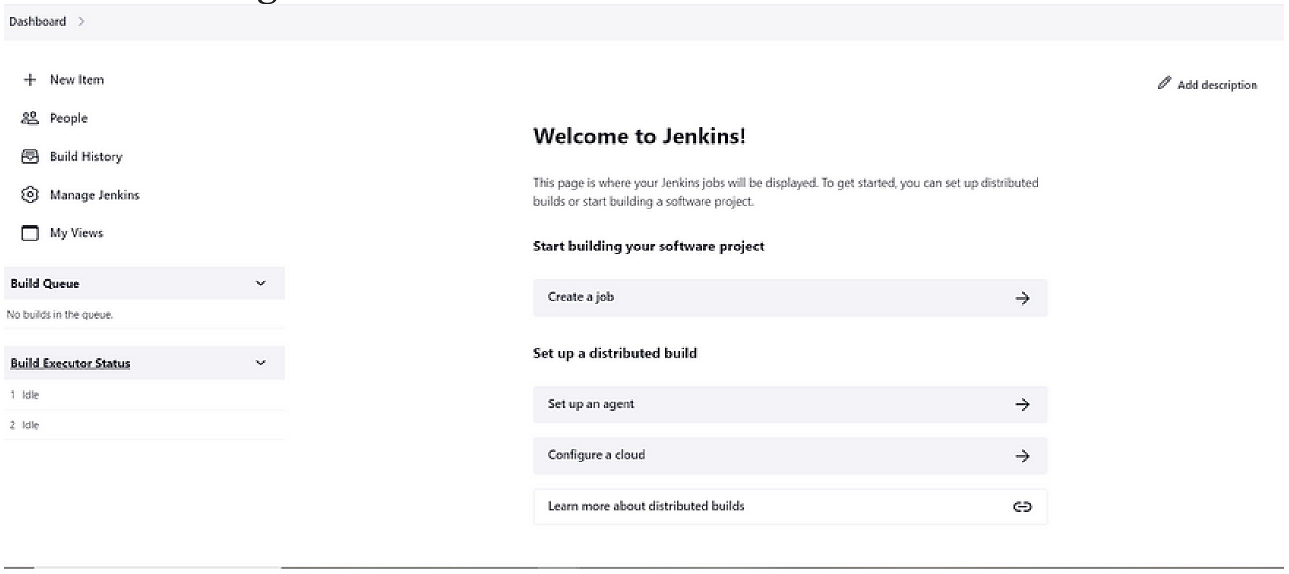
E-mail address

Jenkins 2.392

Skip and continue as admin

Save and Continue

Jenkins Getting Started Screen


Dashboard >

+ New Item
People
Build History
Manage Jenkins
My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Add description

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

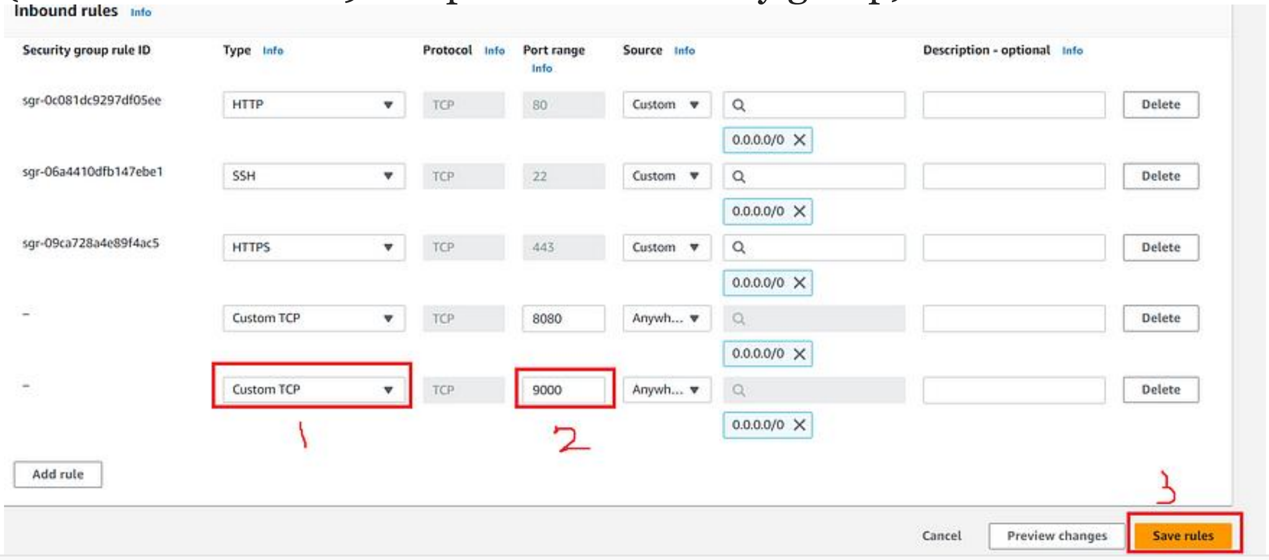
Configure a cloud

Learn more about distributed builds

2B — Install Docker

```
sudo apt-getupdate
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER
sudo chmod 777 /var/run/docker.sock
sudo docker ps
```

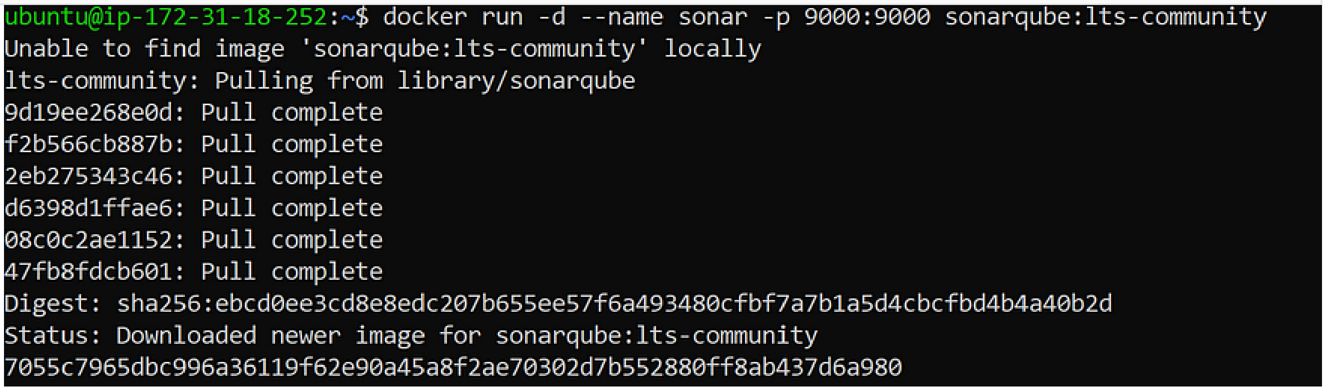
After the docker installation, we create a sonarqube container (Remember added 9000 port in the security group)



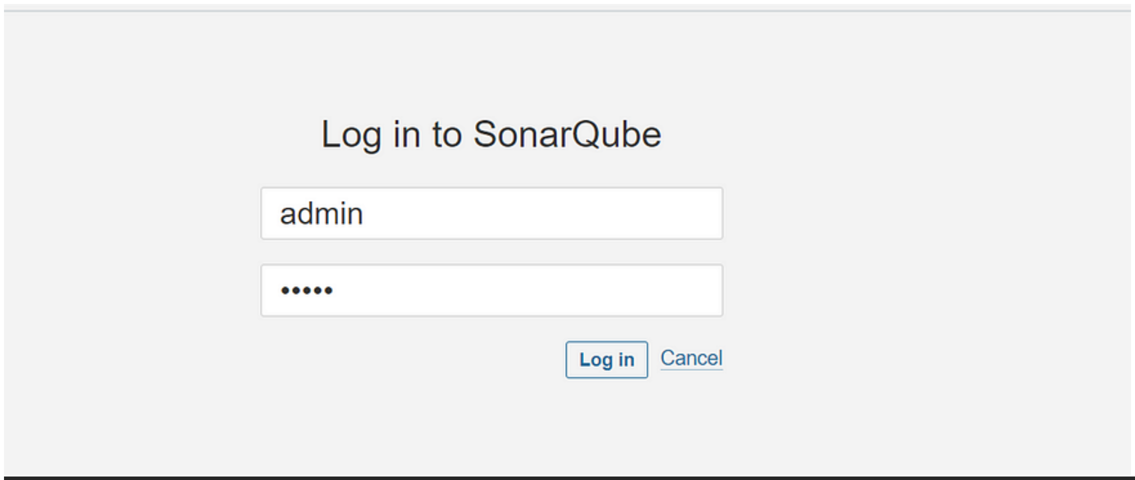
Inbound rules Info

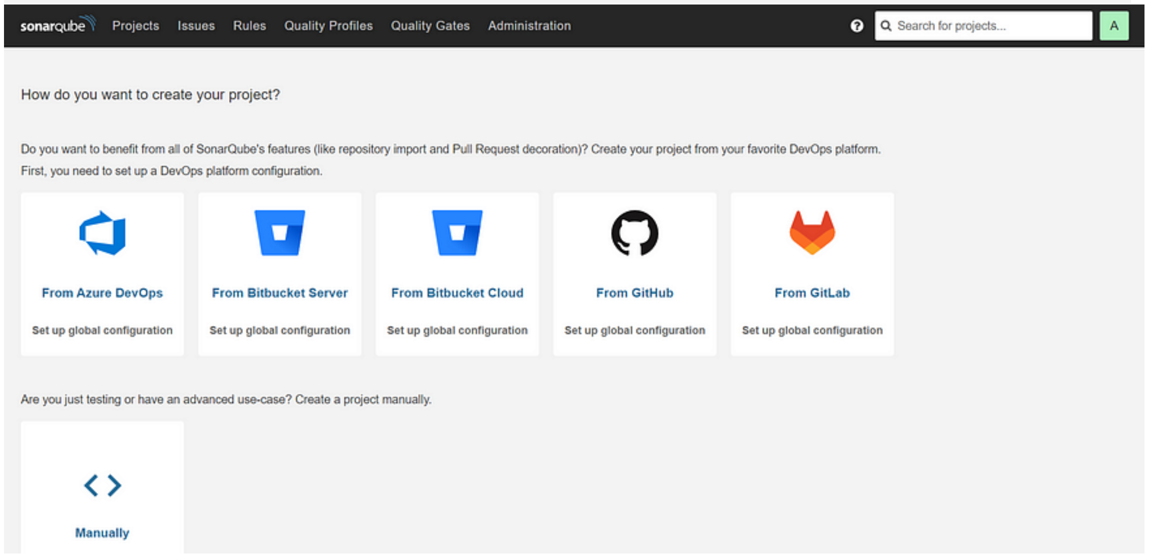
Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-0c081dc9297df05ee	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-06a4410dfb147ebe1	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-09ca728a4e89f4ac5	HTTPS	TCP	443	Custom	0.0.0.0/0		Delete
-	Custom TCP	TCP	8080	Anywh...	0.0.0.0/0		Delete
-	Custom TCP	TCP	9000	Anywh...	0.0.0.0/0		Delete

```
docker run -d --name sonar -p9000:9000 sonarqube:lts-community
```



```
ubuntu@ip-172-31-18-252:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
9d19ee268e0d: Pull complete
f2b566cb887b: Pull complete
2eb275343c46: Pull complete
d6398d1ffae6: Pull complete
08c0c2ae1152: Pull complete
47fb8fdcb601: Pull complete
Digest: sha256:ebcd0ee3cd8e8edc207b655ee57f6a493480cfbf7a7b1a5d4cbcfbd4b4a40b2d
Status: Downloaded newer image for sonarqube:lts-community
7055c7965dbc996a36119f62e90a45a8f2ae70302d7b552880ff8ab437d6a980
```





How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.
First, you need to set up a DevOps platform configuration.

Are you just testing or have an advanced use-case? Create a project manually.

2C — Install Trivy

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

wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --
dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null

echo"deb [signed-by=/usr/share/keyrings/trivy.gpg]
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
```

Next, we will login to Jenkins and start to configure our Pipeline in Jenkins

Step 3 — Install Plugins like JDK, Sonarqube Scanner, Maven, OWASP Dependency Check,

3A — Install Plugin

Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

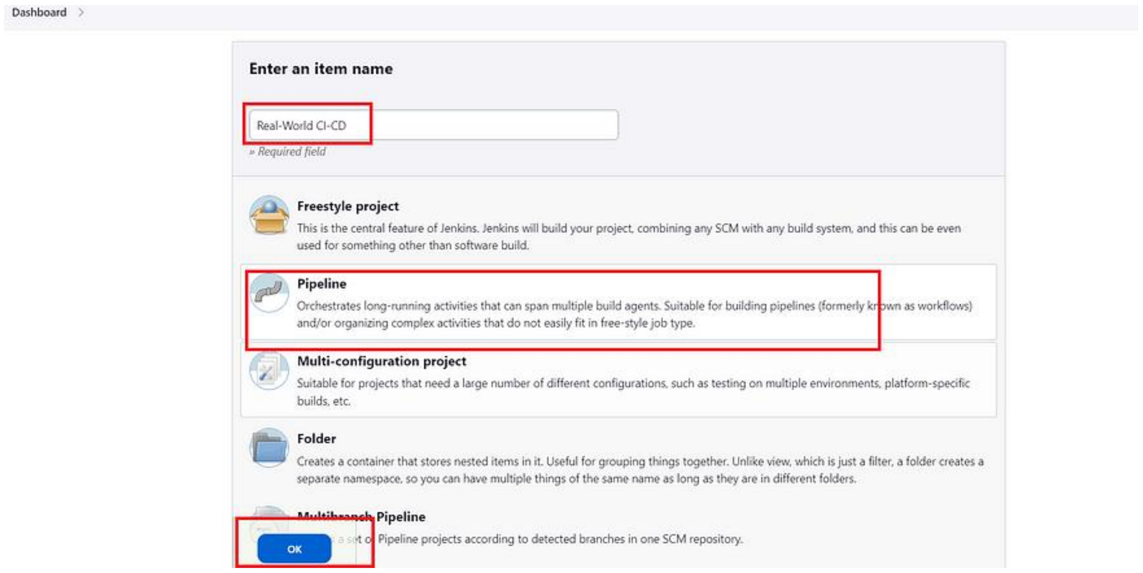
2 → SonarQube Scanner (Install without restart)

3B — Configure Java and Maven in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK and Maven3 → Click on Apply and Save

3C — Create a Job

Label it as Real-World CI-CD, click on Pipeline and Ok.


Dashboard >

Enter an item name

Real-World CI-CD

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

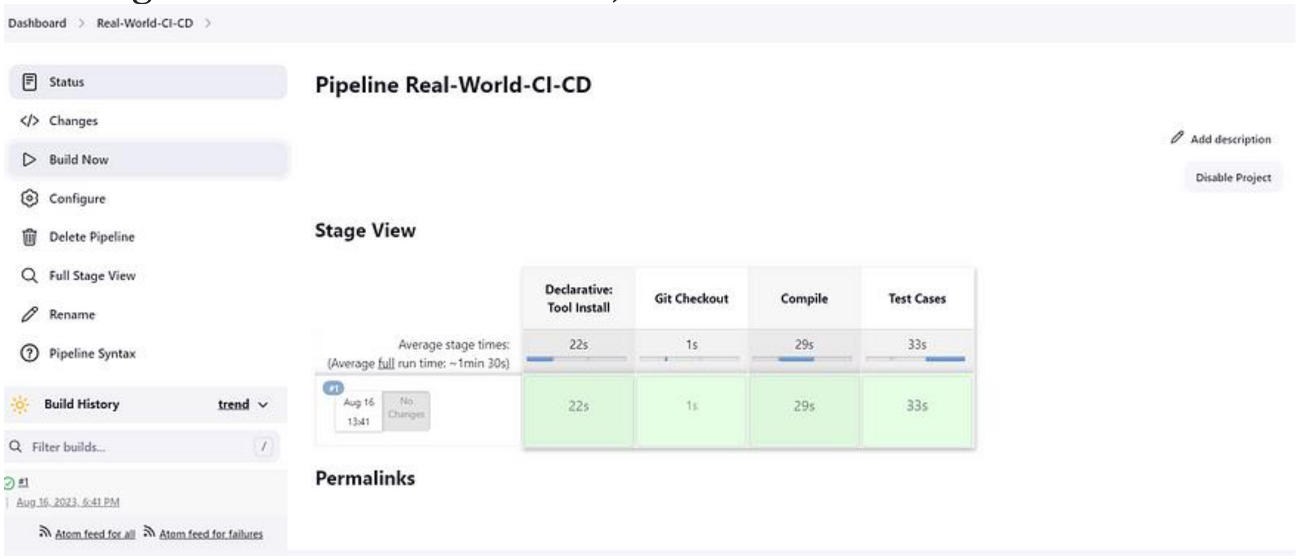
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
a set of Pipeline projects according to detected branches in one SCM repository.

Enter this in Pipeline Script,

```
pipeline {
    agent any

    tools{
        jdk 'jdk17'
        maven 'maven3'
    }

     stages{

        stage("Git Checkout"){
            steps{
                git branch: 'main', changelog: false, poll: false, url:
'https://github.com/Milky19/Petclinic.git'
            }
        }

        stage("Compile"){
            steps{
                sh "mvn clean compile"
            }
        }

     }
}
```

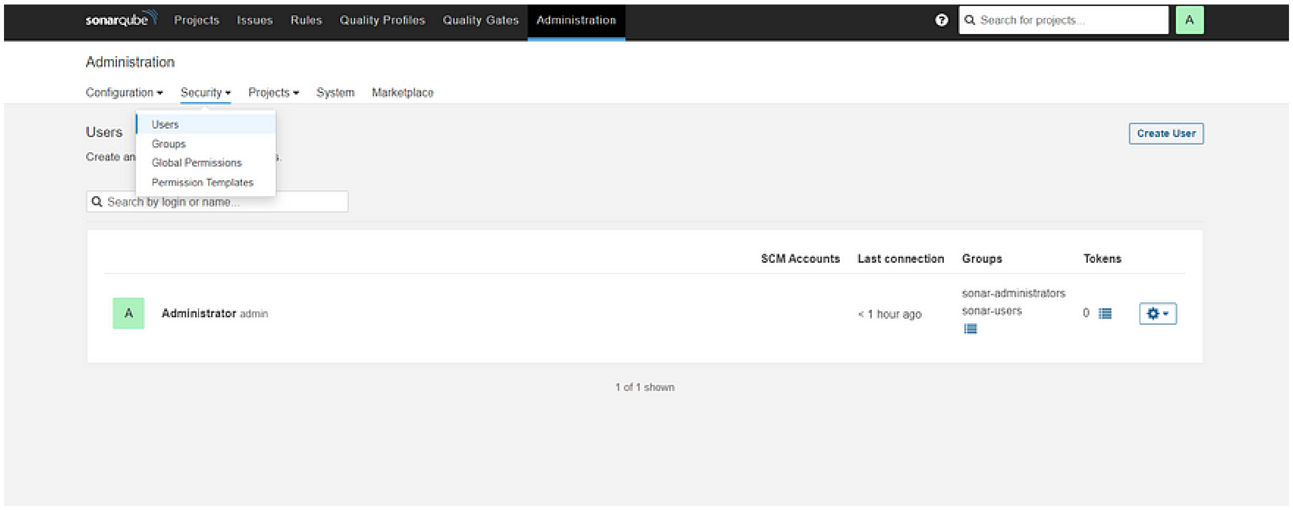
The stage view would look like this,



	Declarative: Tool Install	Git Checkout	Compile	Test Cases
Average stage times: (Average full run time: ~1min 30s)	22s	1s	29s	33s
#1 Aug 16 13:41 No Changes	22s	1s	29s	33s

Step 4 — Configure Sonar Server in Manage Jenkins

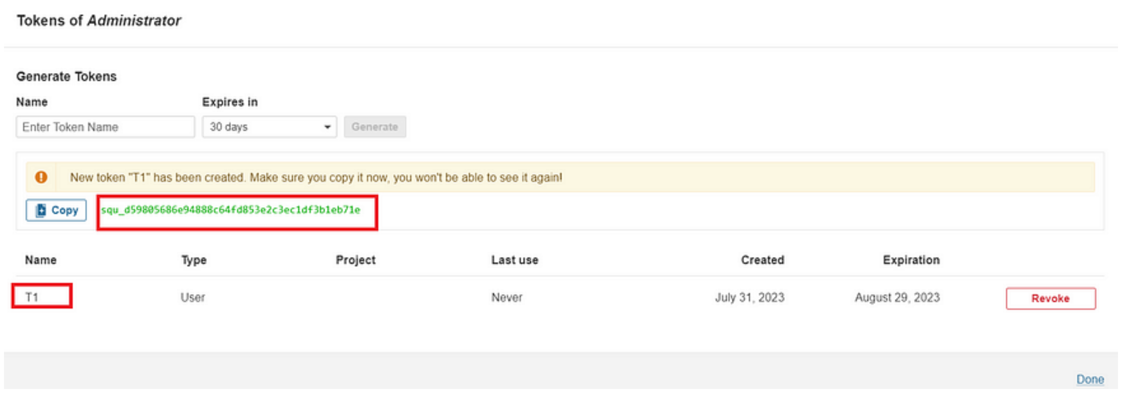
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000 , sp <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



	SCM Accounts	Last connection	Groups	Tokens
A Administrator admin		< 1 hour ago	sonar-administrators sonar-users	0

1 of 1 shown

Click on Update Token


Tokens of *Administrator*

Generate Tokens

Name: Enter Token Name
Expires in: 30 days

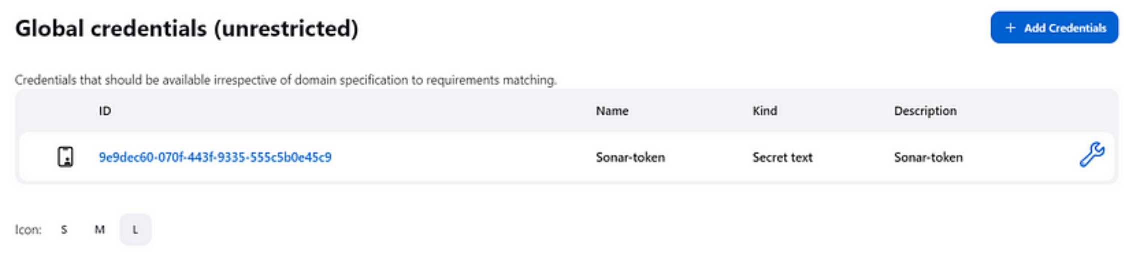
New token "T1" has been created. Make sure you copy it now, you won't be able to see it again!

```
squ_d59805686e94888c64fd853e2c3ec1df3b1eb71e
```

Name	Type	Project	Last use	Created	Expiration	
T1	User		Never	July 31, 2023	August 29, 2023	Revoke

Copy this Tokensqu_1fe3f7207ffb6a4860398475013b1c37a3177b53

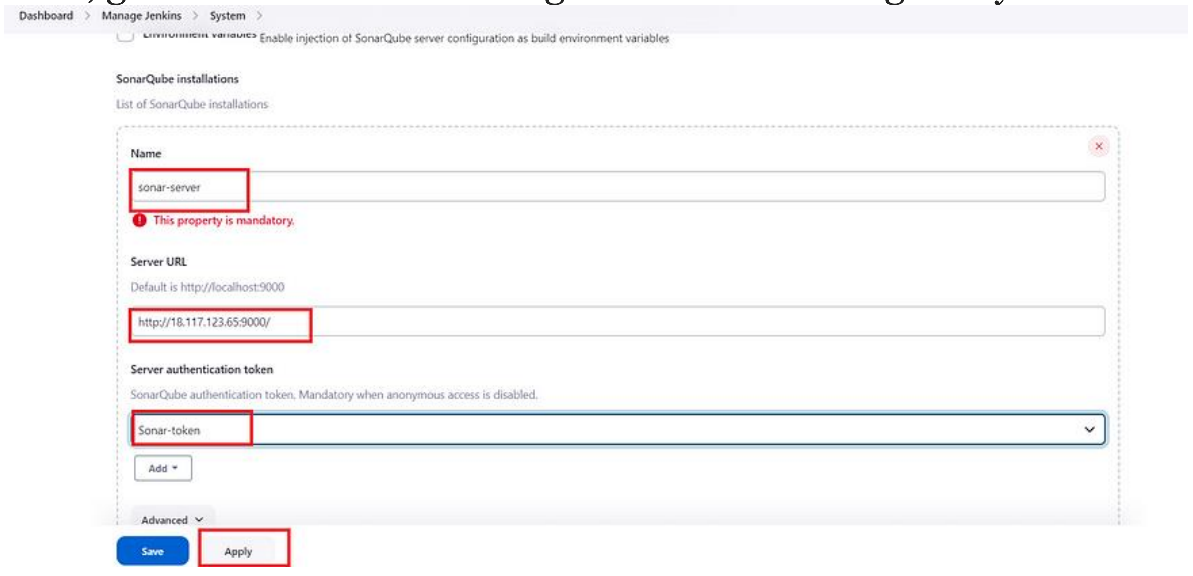
Goto Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this


Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
9e9dec60-070f-443f-9335-555c5b0e45c9	Sonar-token	Secret text	Sonar-token

Now, goto Dashboard → Manage Jenkins → Configure System



Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

Name: sonar-server

This property is mandatory.

Server URL

Default is http://localhost:9000

http://18.117.123.65:9000/

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

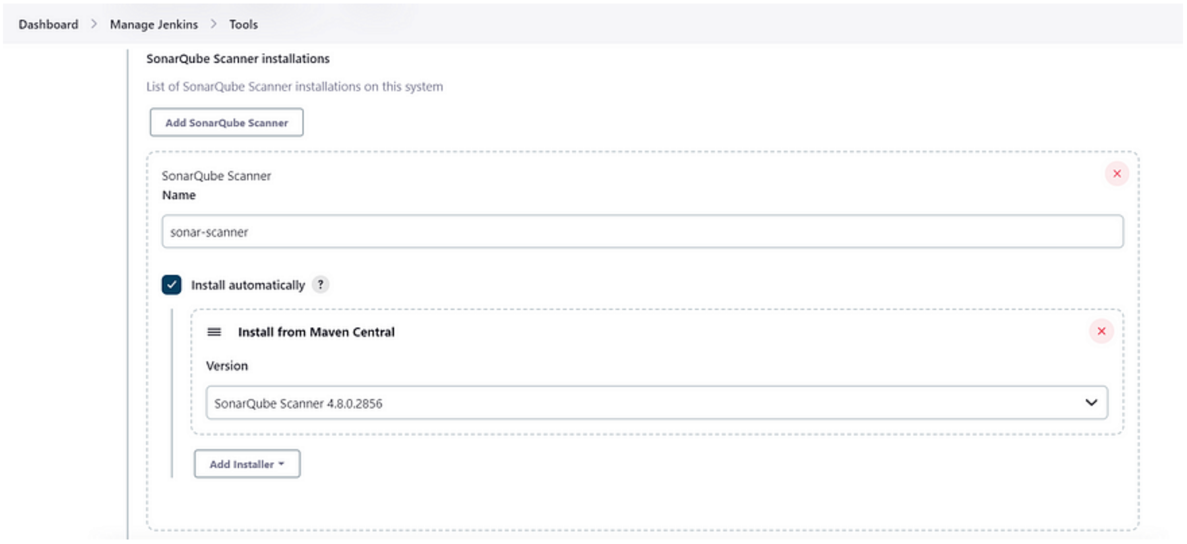
Sonar-token

Click on Apply and Save

Configure System option is used in Jenkins to configure different server

Global Tool Configuration is used to configure different tools that we install using Plugins

We will install sonar-scanner in tools.



SonarQube Scanner installations

List of SonarQube Scanner installations on this system

SonarQube Scanner

Name

sonar-scanner

Install automatically ✓

Install from Maven Central

Version

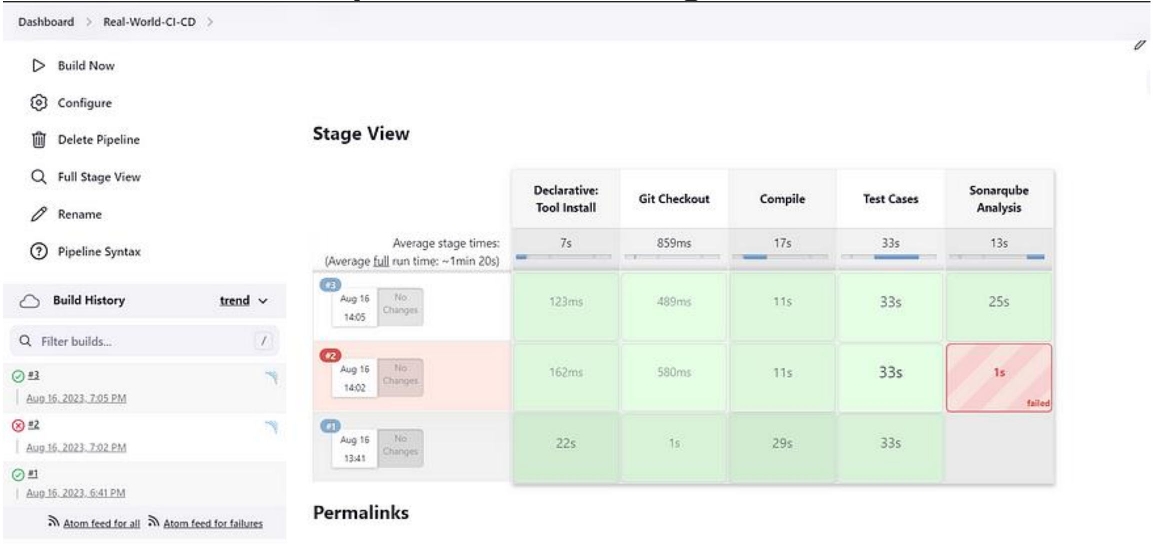
SonarQube Scanner 4.8.0.2856

Lets goto our Pipeline and add Sonar-qube Stage in our Pipeline Script

```
pipeline {
    agent any

    tools{
        jdk 'jdk17'
        maven 'maven3'
    }
    stages{

        stage("Git Checkout"){
            steps{
                git branch: 'main', changelog: false, poll: false, url:
'https://github.com/Milky19/Petclinic.git'
```

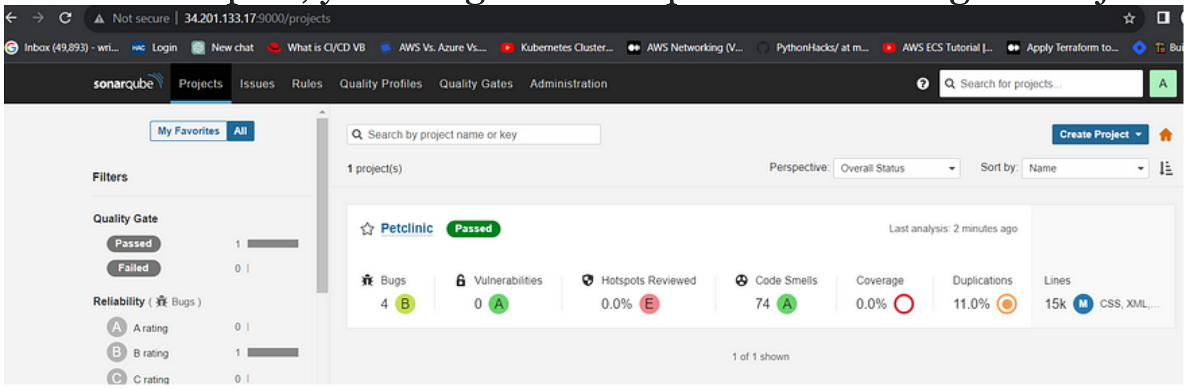
```
                }
            }

            stage("Compile"){
                steps{
                    sh "mvn clean compile"
      }
           }
    stage("Sonarqube Analysis "){
                steps{
                    script {
                    withSonarQubeEnv(credentialsId: 'Sonar-token') {
                    sh 'mvn sonar:sonar'
                    }
                  }
                }
            }


stage("quality gate"){
        steps {
          script {
           waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
          }
        }
      }

            }
        }
}
```

Click on Build now, you will see the stage view like this



	Declarative: Tool Install	Git Checkout	Compile	Test Cases	Sonarqube Analysis
Average stage times: (Average full run time: ~1min 20s)	7s	859ms	17s	33s	13s
#3 Aug 16 14:05 No Changes	123ms	489ms	11s	33s	25s
#2 Aug 16 14:02 No Changes	162ms	580ms	11s	33s	1s failed
#1 Aug 16 13:41 No Changes	22s	1s	29s	33s	

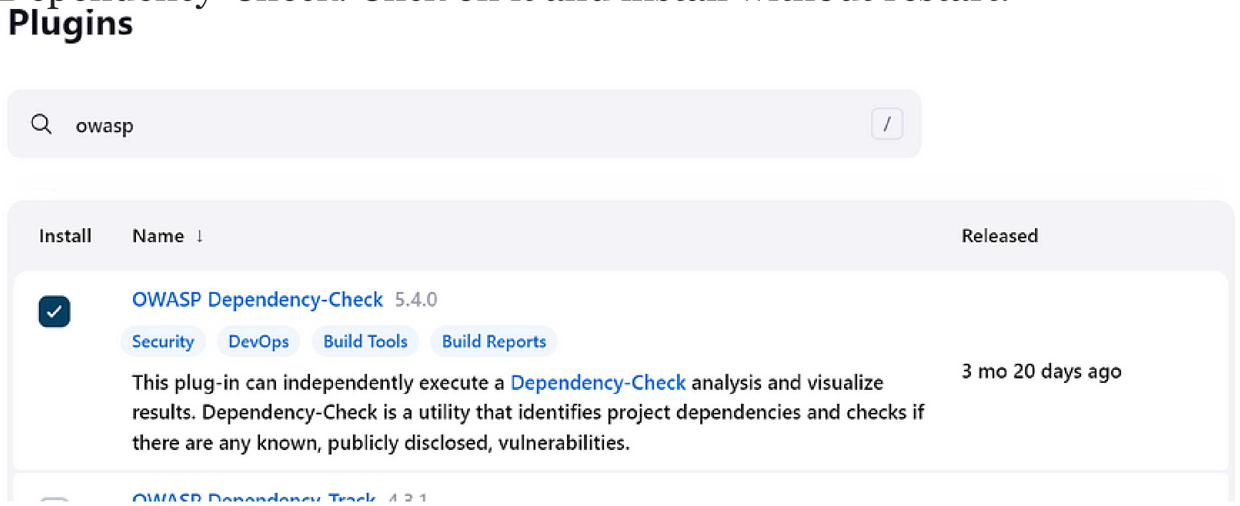
To see the report, you can goto Sonarqube Server and goto Projects.



You can see the report has been generated and the status shows as passed. You can see that there are 15K lines. To see detailed report, you can go to issues.

Step 5 — Install OWASP Dependency Check Plugins

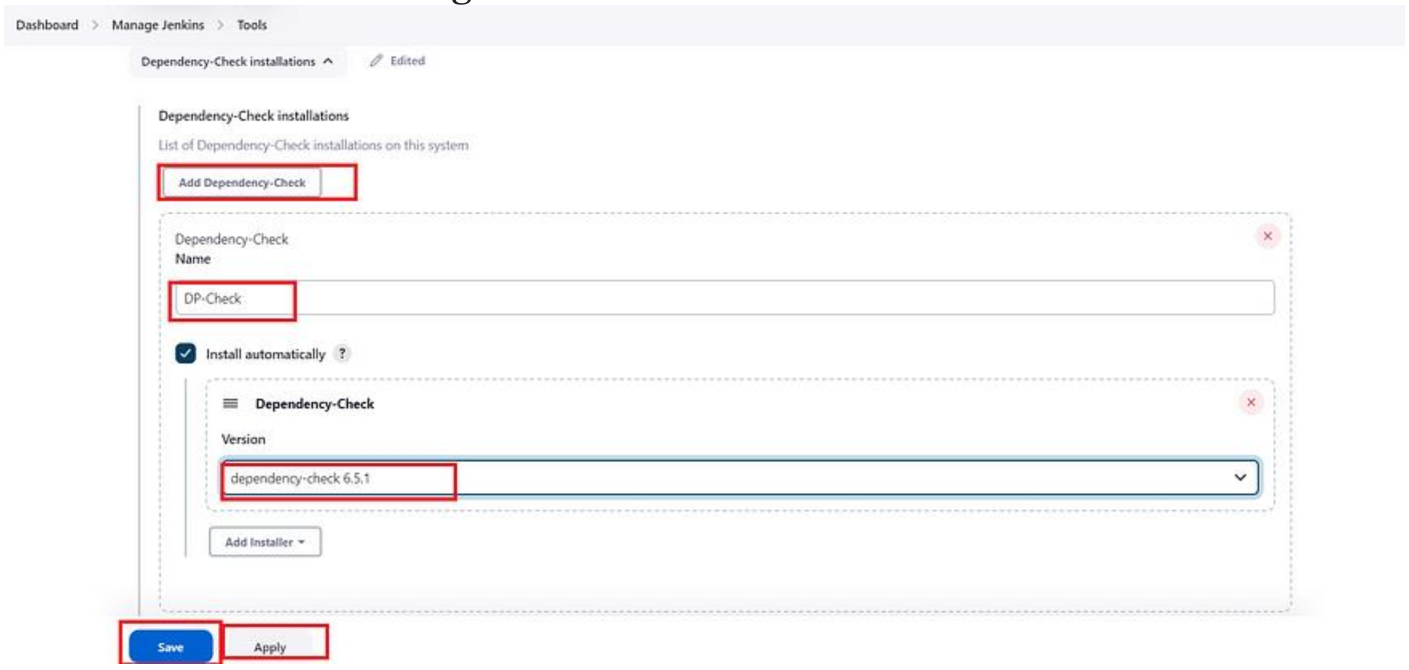
GotoDashboard → Manage Jenkins → Plugins → OWASP Dependency-Check. Click on it and install without restart.



This plug-in can independently execute a Dependency-Check analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.

First, we configured Plugin and next we have to configure Tool

Goto Dashboard → Manage Jenkins → Tools →


Dashboard > Manage Jenkins > Tools

Dependency-Check installations

Dependency-Check installations

List of Dependency-Check installations on this system

Add Dependency-Check

Dependency-Check

Name

DP-Check

✓ Install automatically

Dependency-Check

Version

dependency-check 6.5.1

Add Installer

Save Apply

Click on apply and Save here.

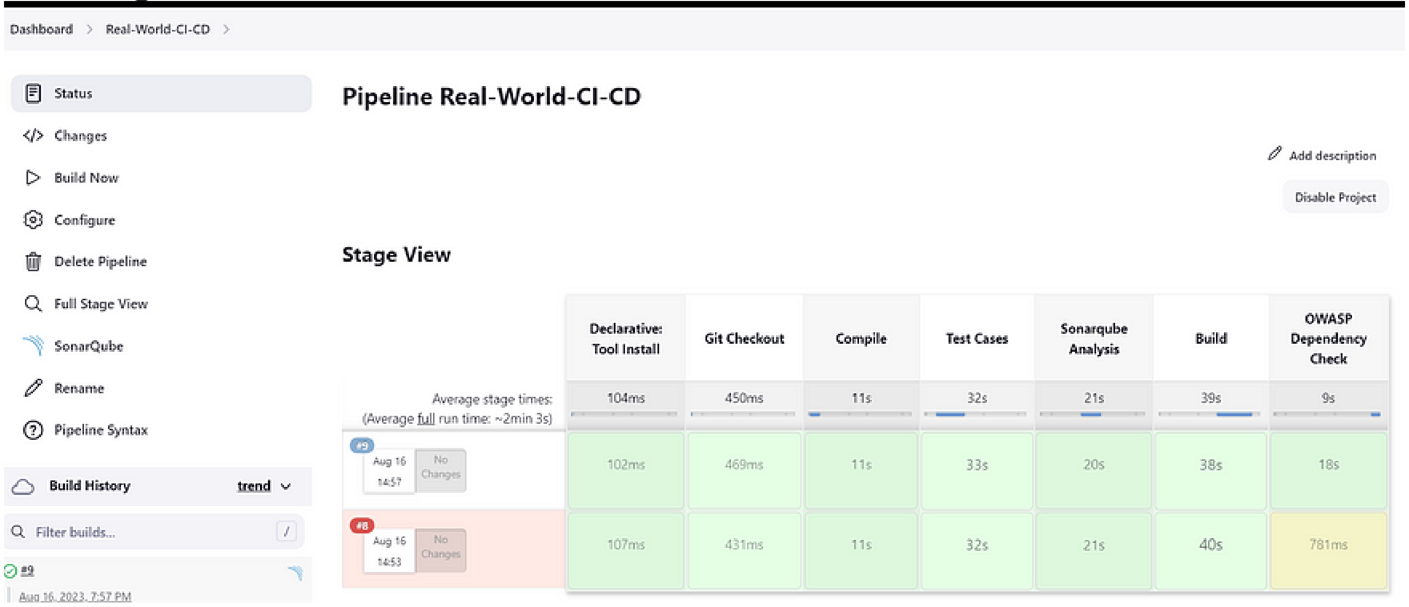
Now goto configure → Pipeline and add this stage to your pipeline

```
   stage("OWASP Dependency Check"){
            steps{
                dependencyCheck additionalArguments: '--scan ./ --format HTML
', odcInstallation: 'DP-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-
report.xml'
            }
        }
stage("Build"){
            steps{
                sh " mvn clean install"
            }
        }
```

The final pipeline would look like this,

```
pipeline {
    agent any

    tools{
        jdk 'jdk17'
        maven 'maven3'
    }

     environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }

    stages{

        stage("Git Checkout"){
            steps{
                git branch: 'main', changelog: false, poll: false, url:
'https://github.com/Milky19/Petclinic.git'
            }
        }

        stage("Compile"){
            steps{
                sh "mvn clean compile"
            }
        }

         stage("Test Cases"){
            steps{
                sh "mvn test"
   }
        }
stage("Sonarqube Analysis "){
            steps{
                script {
                withSonarQubeEnv(credentialsId: 'Sonar-token') {
                sh 'mvn sonar:sonar'
                }
              }
            }
        }

        stage("Build"){
            steps{
                sh " mvn clean install"
            }
        }

          stage("OWASP Dependency Check"){
            steps{
                dependencyCheck additionalArguments: '--scan ./ ' ,
odcInstallation: 'DP-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-
report.xml'
            }
        }

    }
}
```

The stage view would look like this,

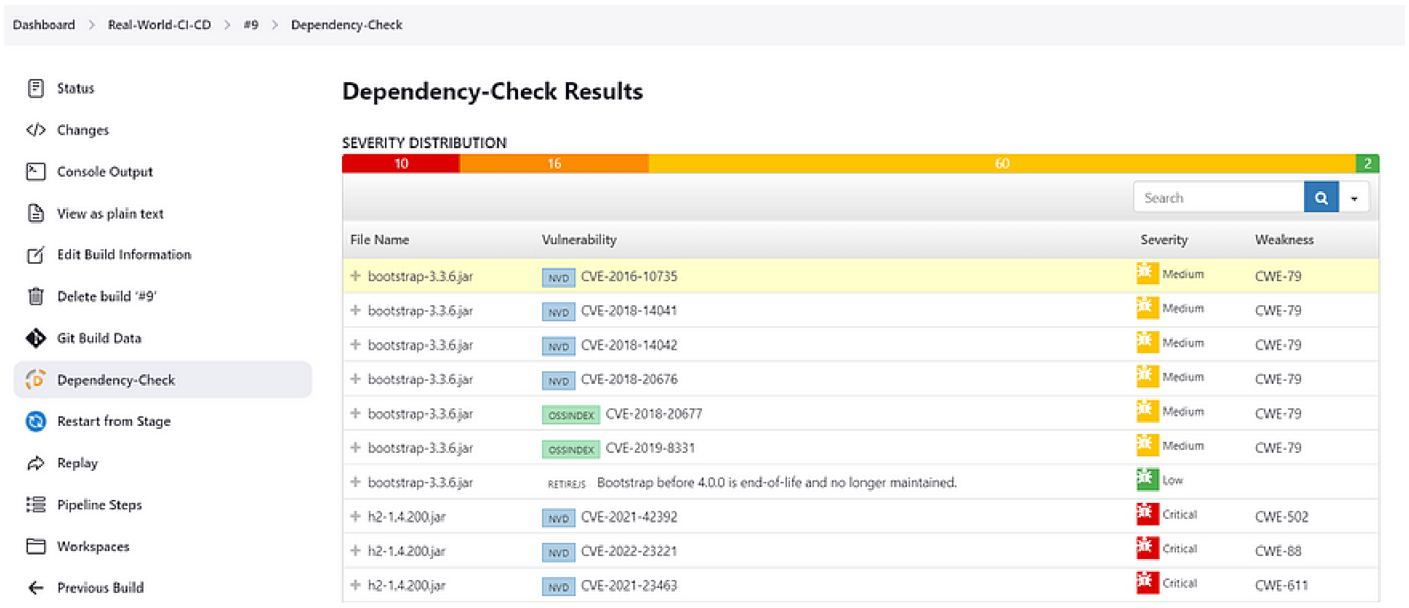


Pipeline Real-World-CI-CD

Stage View

	Declarative: Tool Install	Git Checkout	Compile	Test Cases	Sonarqube Analysis	Build	OWASP Dependency Check
Average stage times: (Average full run time: ~2min 3s)	104ms	450ms	11s	32s	21s	39s	9s
#9 Aug 16 14:57 No Changes	102ms	469ms	11s	33s	20s	38s	18s
#8 Aug 16 14:53 No Changes	107ms	431ms	11s	32s	21s	40s	781ms

You will see that in status, a graph will also be generated


Dashboard > Real-World-CI-CD > #9 > Dependency-Check

- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete build '#9'
- Git Build Data
- Dependency-Check
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces
- Previous Build

Dependency-Check Results

SEVERITY DISTRIBUTION

10 16 60 2

File Name	Vulnerability	Severity	Weakness
bootstrap-3.3.6.jar	NVD CVE-2016-10735	Medium	CWE-79
bootstrap-3.3.6.jar	NVD CVE-2018-14041	Medium	CWE-79
bootstrap-3.3.6.jar	NVD CVE-2018-14042	Medium	CWE-79
bootstrap-3.3.6.jar	NVD CVE-2018-20676	Medium	CWE-79
bootstrap-3.3.6.jar	OSSINDEX CVE-2018-20677	Medium	CWE-79
bootstrap-3.3.6.jar	OSSINDEX CVE-2019-8331	Medium	CWE-79
bootstrap-3.3.6.jar	RETIREJS Bootstrap before 4.0.0 is end-of-life and no longer maintained.	Low	
h2-1.4.200.jar	NVD CVE-2021-42392	Critical	CWE-502
h2-1.4.200.jar	NVD CVE-2022-23221	Critical	CWE-88
h2-1.4.200.jar	NVD CVE-2021-23463	Critical	CWE-611

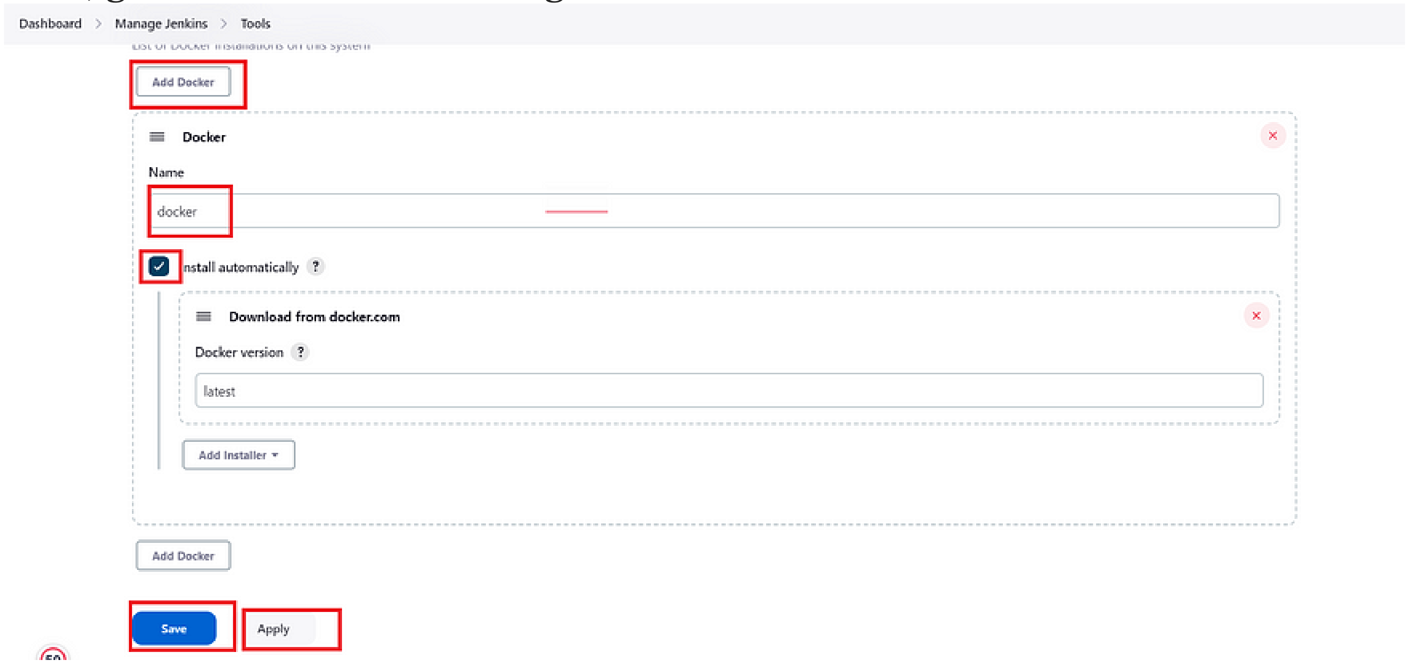
Step 6 — Docker Image Build and Push

We need to install Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

- Docker
- Docker Commons
- Docker Pipeline
- Docker API
- docker-build-step

and click on install without restart

Now, goto Dashboard → Manage Jenkins → Tools →



Dashboard > Manage Jenkins > Tools

List of Docker installations on this system

Docker

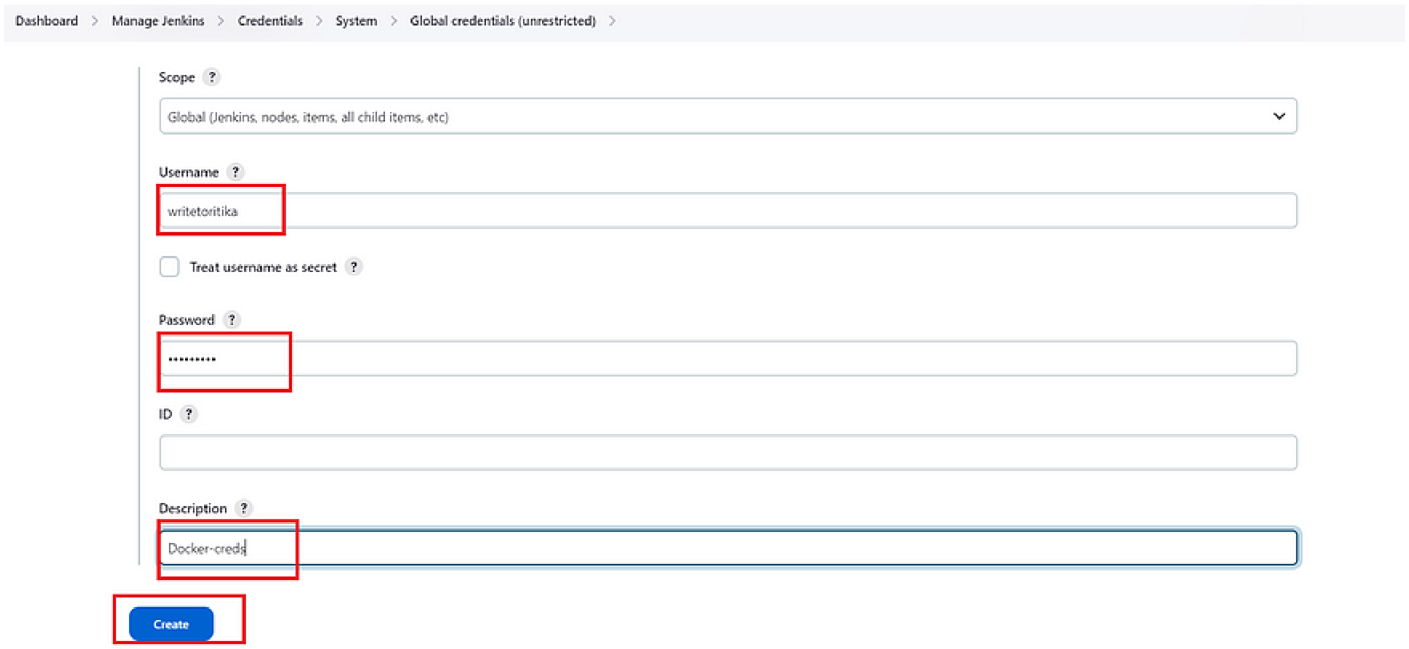
Name: docker

[✓] Install automatically

Download from docker.com

Docker version: latest

Add DockerHub Username and Password under Global Credentials



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: writetoritika

[ ] Treat username as secret

Password: •••••••••

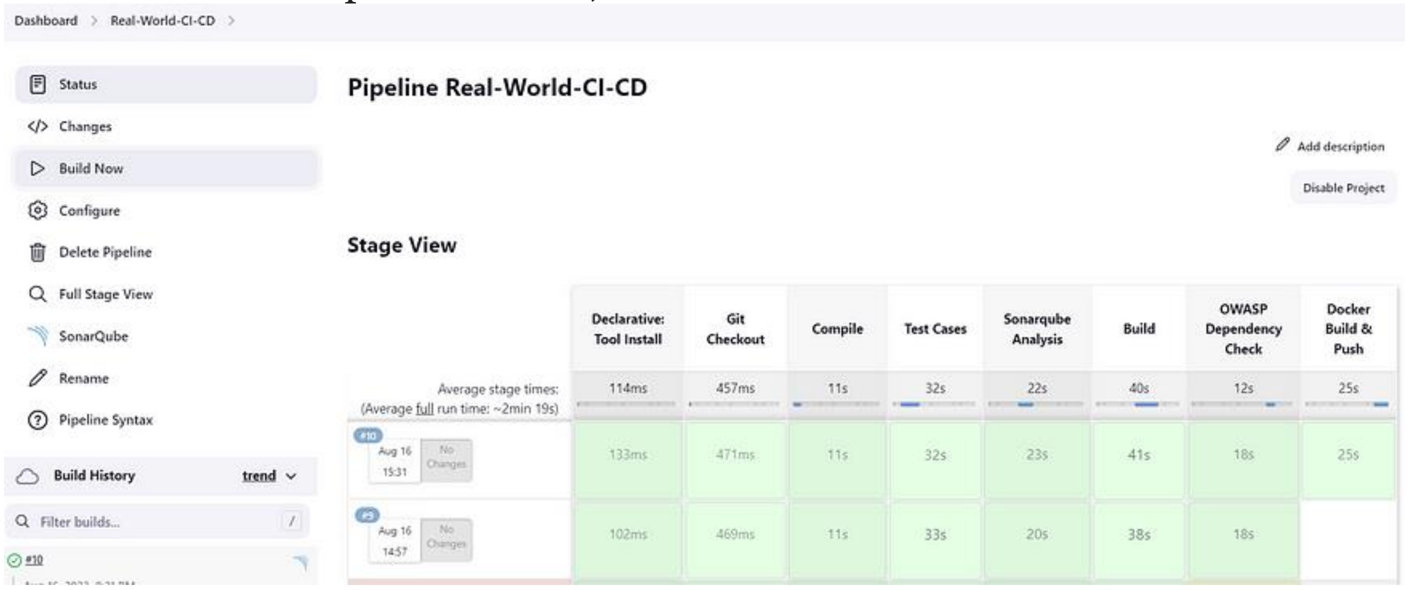
ID:

Description: Docker-creds

Add this stage in Pipeline Script

```
        stage("Docker Build & Push"){
                steps{
                    script{
                       withDockerRegistry(credentialsId: ''bc86df08-bacf-4695-
99cb-8cefb3406235', toolName: 'docker') {

                            sh "docker build -t petclinic1 ."
                            sh "docker tag petclinic1 hanvitha/pet-
clinic123:latest "
                            sh "docker push hanvitha/pet-clinic123:latest "

                        }
                    }
                }
            }
```

You will see the output like below,



Dashboard > Real-World-CI-CD >

Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
SonarQube
Rename
Pipeline Syntax

Build History trend

Filter builds...

#10

Pipeline Real-World-CI-CD

Add description
Disable Project

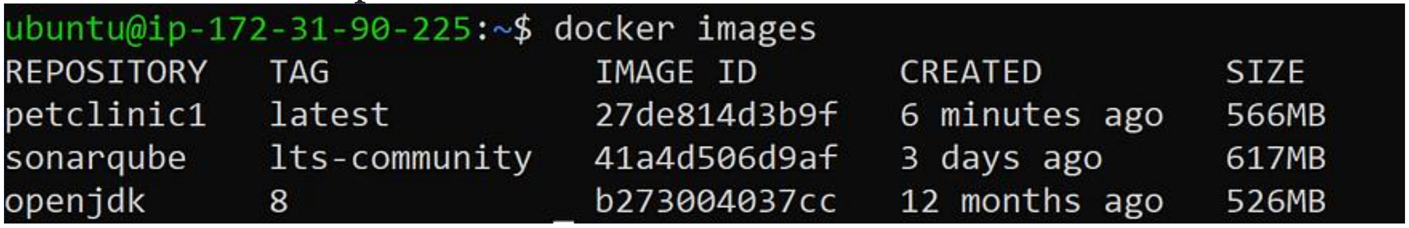
Stage View

	Declarative: Tool Install	Git Checkout	Compile	Test Cases	Sonarqube Analysis	Build	OWASP Dependency Check	Docker Build & Push
Average stage times: (Average full run time: ~2min 19s)	114ms	457ms	11s	32s	22s	40s	12s	25s
#10 Aug 16 15:31 No Changes	133ms	471ms	11s	32s	23s	41s	18s	25s
#9 Aug 16 14:57 No Changes	102ms	469ms	11s	33s	20s	38s	18s	

Now, when you do

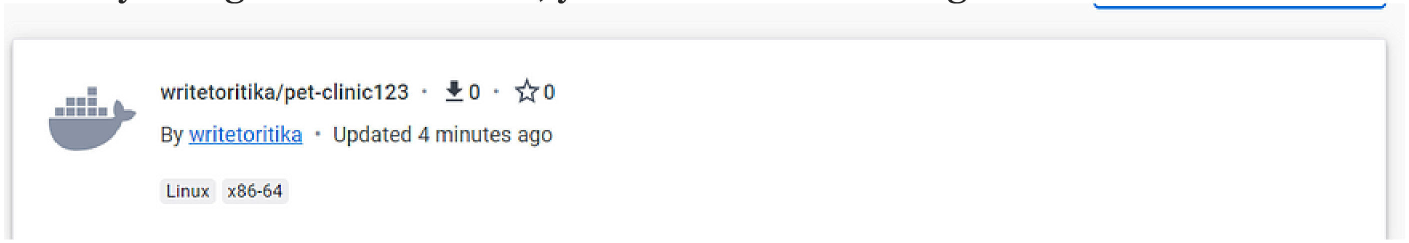
```
docker images
```

You will see this output



```
ubuntu@ip-172-31-90-225:~$ docker images
REPOSITORY    TAG             IMAGE ID       CREATED          SIZE
petclinic1    latest          27de814d3b9f   6 minutes ago    566MB
sonarqube     lts-community   41a4d506d9af   3 days ago       617MB
openjdk       8               b273004037cc   12 months ago    526MB
```

When you log in to Dockerhub, you will see a new image is created



writetoritika/pet-clinic123 · 0 · 0

By writetoritika · Updated 4 minutes ago

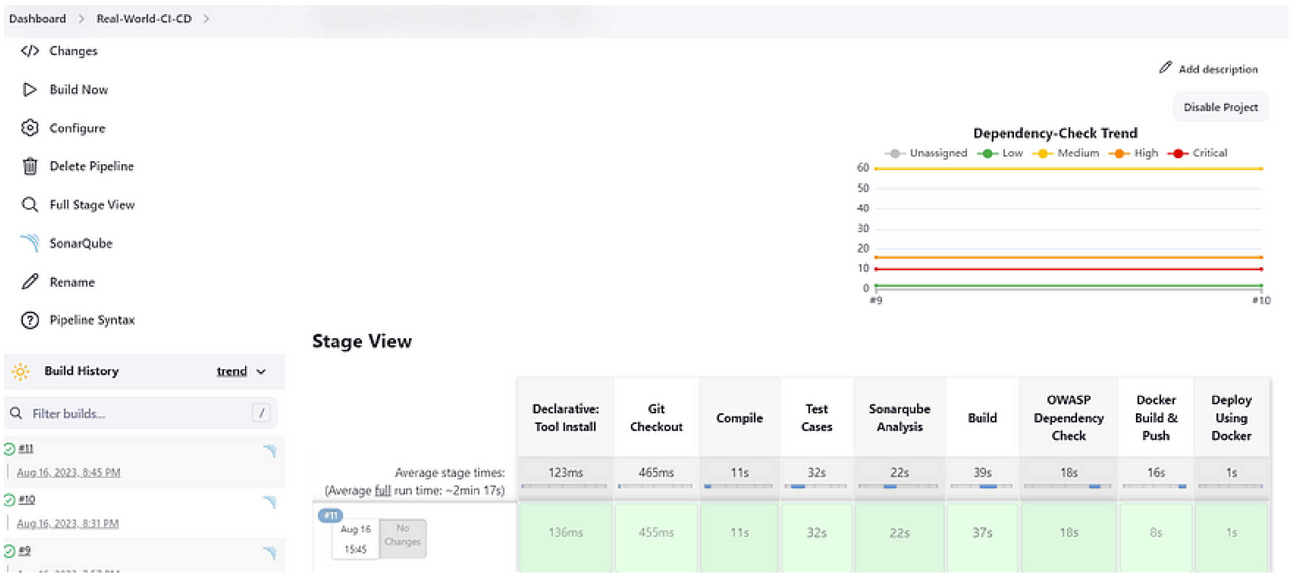
Linux x86-64

Step 7 — Deploy image using Docker

Add this stage to your pipeline syntax

```
stage("TRIVY"){
            steps{
                sh "trivy image hanvitha/pet-clinic123:latest"
            }
        }

stage("Deploy Using Docker"){
            steps{
                sh " docker run -d --name pet1 -p 8082:8080hanvitha/pet-
clinic123:latest "
            }
        }
```

You will see the Stage View like this,



Stage View

	Declarative: Tool Install	Git Checkout	Compile	Test Cases	Sonarqube Analysis	Build	OWASP Dependency Check	Docker Build & Push	Deploy Using Docker
Average stage times: (Average full run time: ~2min 17s)	123ms	465ms	11s	32s	22s	39s	18s	16s	1s
#11 Aug 16 15:45 No Changes	136ms	455ms	11s	32s	22s	37s	18s	8s	1s

Step 8 — Terminate the AWS EC2 Instance

Lastly, do not forget to terminate the AWS EC2 Instance.

Complete pipeline

```
pipeline{
    agent any
    tools {
        jdk 'jdk11'
        maven 'maven3'
    }
    stages{
        stage ('clean Workspace'){
            steps{
                cleanWs()
            }
        }
        stage ('checkout scm') {
            steps {
                checkout scmGit(branches: [[name: '*/master']], extensions: [],
userRemoteConfigs: [[url: 'https://github.com/Milky19/amazon-eks-jenkins-
terraform-aj7.git']])
            }
        }
        stage ('maven compile') {
            steps {
                sh 'mvn clean compile'
            }
        }
        stage ('sonarqube Analysis'){
            steps{
                script{
                    withSonarQubeEnv(credentialsId: 'Sonar-token') {
                      sh 'mvn sonar:sonar'
                    }
                }
            }
        }
        stage("quality gate"){
            steps {
                script {
                  waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-
token'
                }
           }
        }
        stage("OWASP Dependency Check"){
            steps{
                dependencyCheck additionalArguments: '--scan ./ --format HTML ',
odcInstallation: 'DP-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-
report.xml'
            }
        }
        stage ('Build war file'){
            steps{
                sh 'mvn clean install package'
            }
        }
        stage ('Build and push to docker hub'){
            steps{
                script{
                    withDockerRegistry(credentialsId: 'docker', toolName:
'docker') {
                        sh "docker build -t petclinic1 ."
                        sh "docker tag petclinic1 sevenajay/pet-clinic123:latest"
                        sh "docker push sevenajay/pet-clinic123:latest"
                    }
                }
            }
        }
        stage("TRIVY"){
            steps{
                sh "trivy image hanvitha/pet-clinic123:latest"
            }
        }
        stage ('Deploy to container'){
            steps{
                sh 'docker run -d --name pet1 -p 8082:8080 hanvitha/pet-
clinic123:latest'
            }
        }
    }
}
```