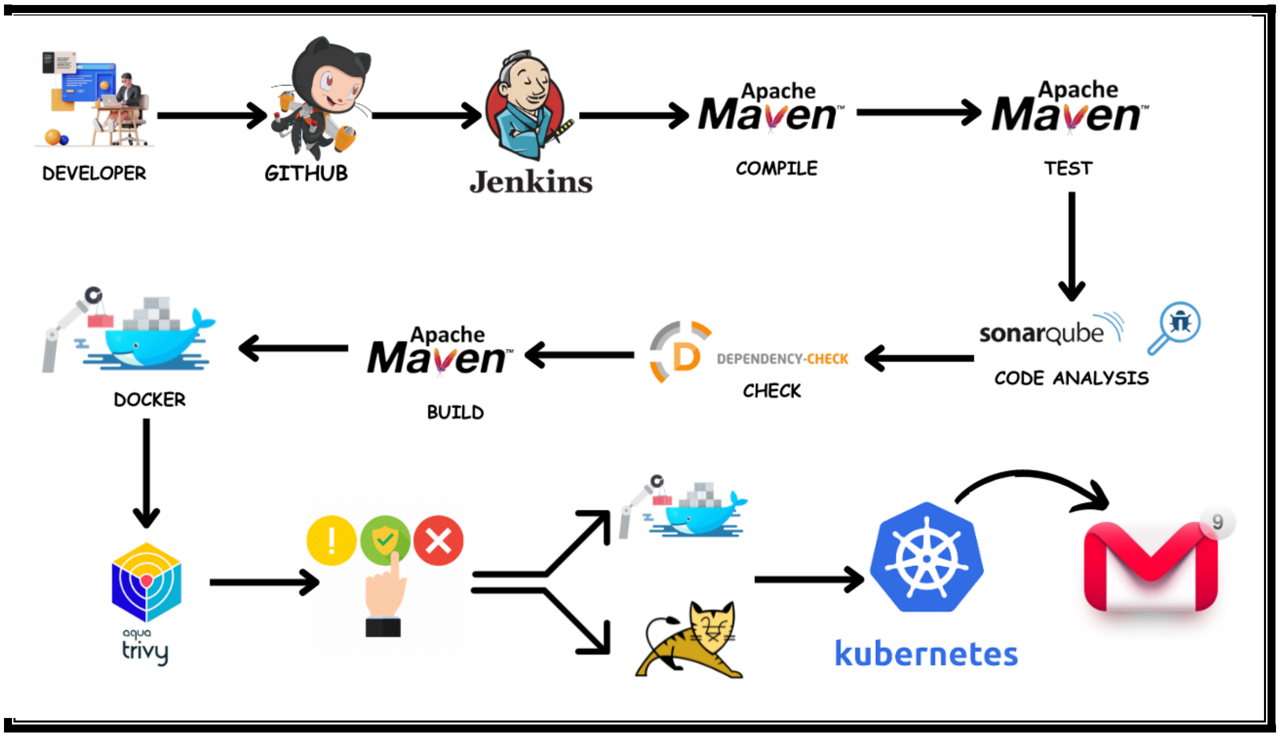
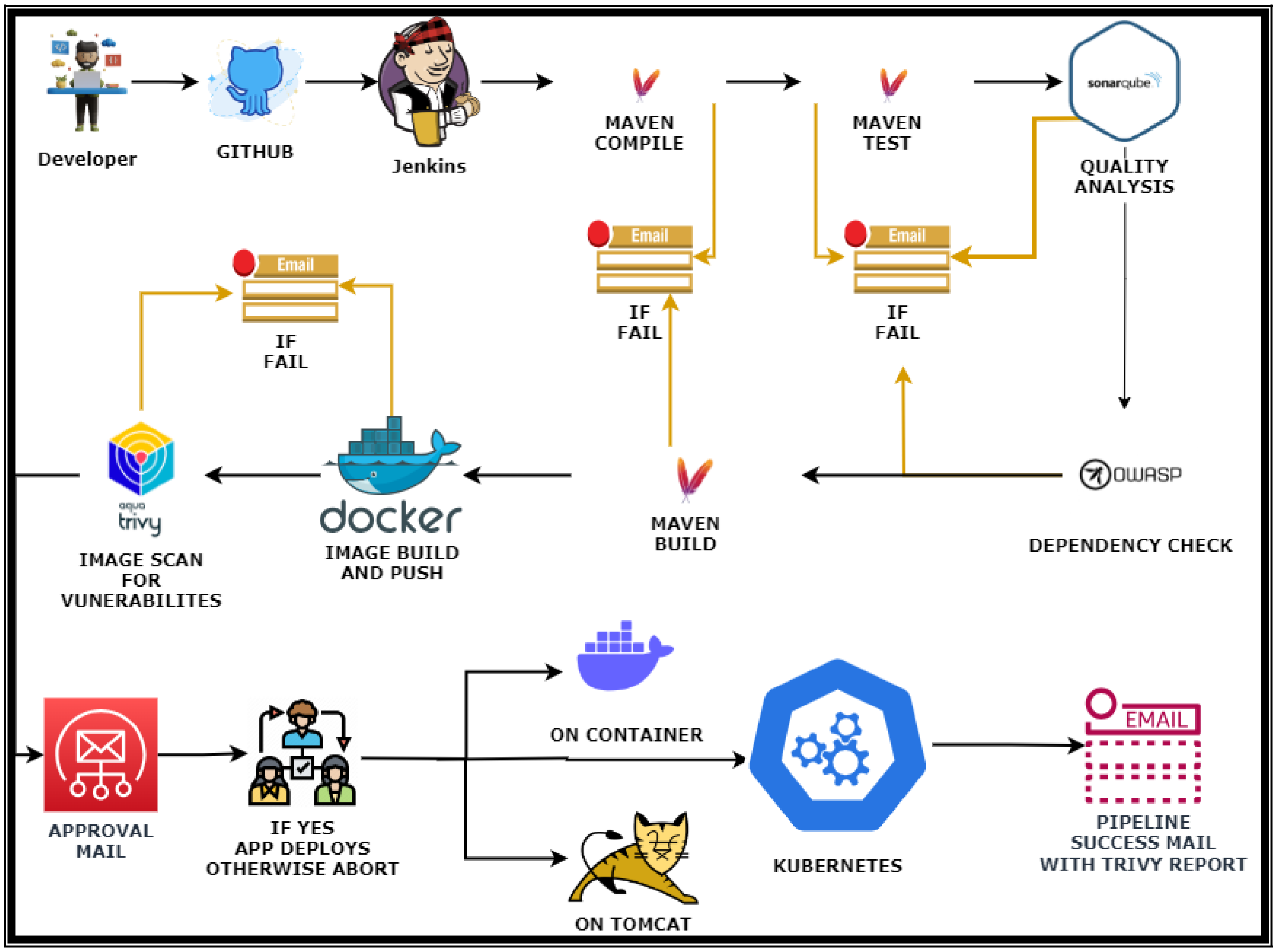
COMPLETE CI/CD PROJECT



I will be deploying a Pet Clinic Java Based Application. This is an everyday use case scenario used by several organizations. i will be using Jenkins as a CICD tool and deploying our application on Tomcat Server.

I will be deploying our application in two ways, using Docker Container and other is using Tomcat Server.And finally we will deploy it kubernetes Also.



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

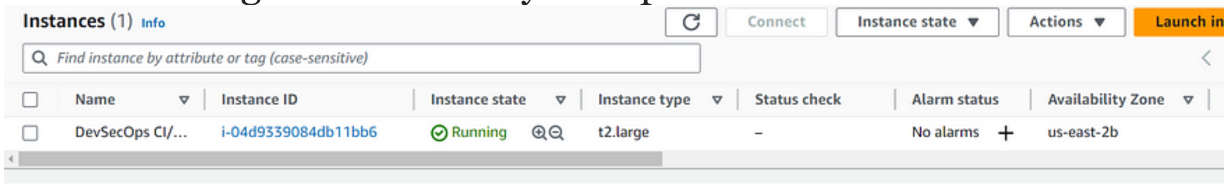
Step 7: Deploy image using Docker

Step 8: Install Tomcat on Port 8083 and finally deploy on Apache Tomcat using groovy pipeline script mentioned

Step 9: Access the Real World Application

Step 10: Terminate the AWS EC2 Instance

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

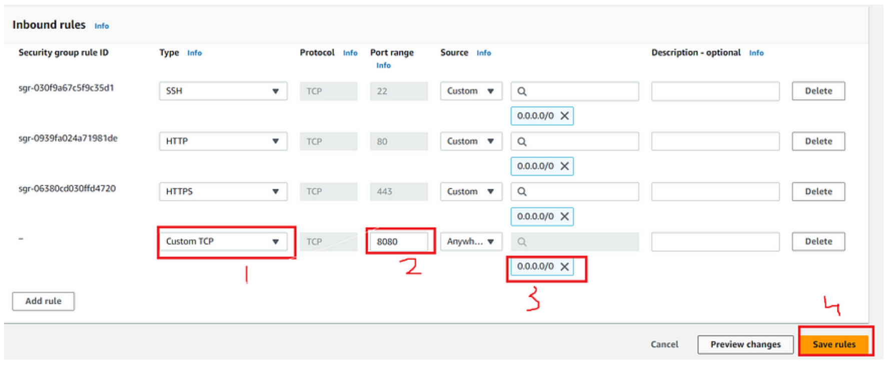
Step 2: Install Jenkins, Docker and Trivy

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.

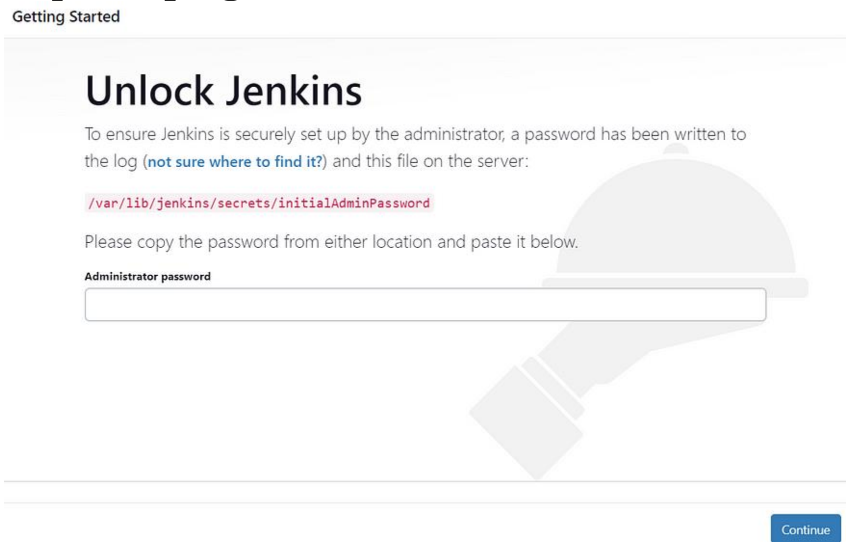

Inbound rules Info

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

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

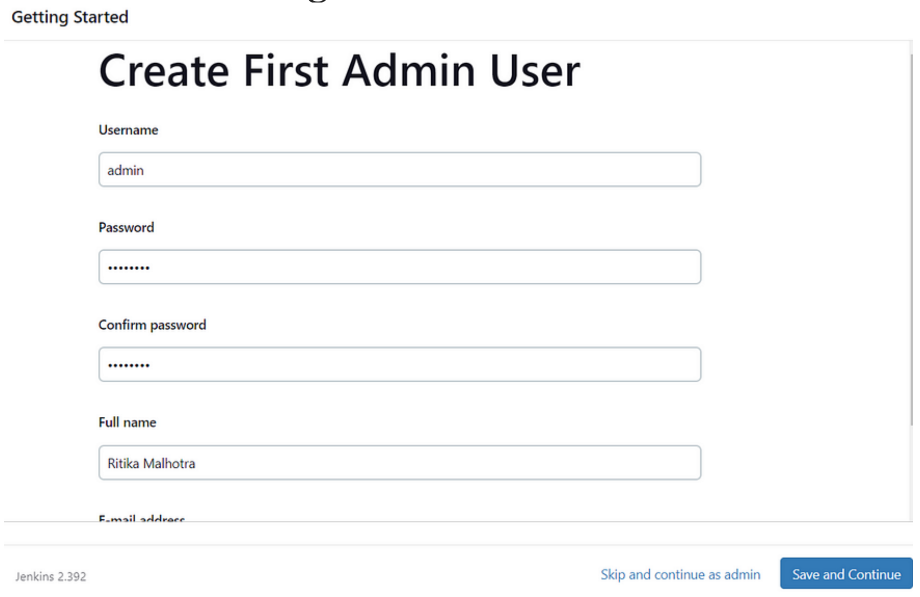
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/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: ••••••••

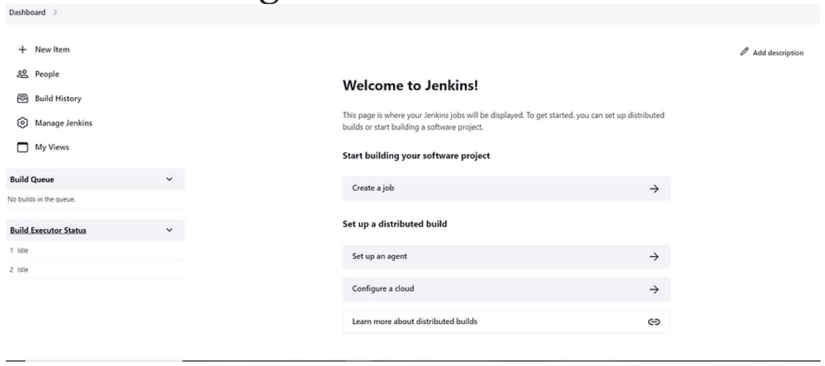
Confirm password: ••••••••

Full name: Ritika Malhotra

E-mail address

Jenkins 2.392

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 →

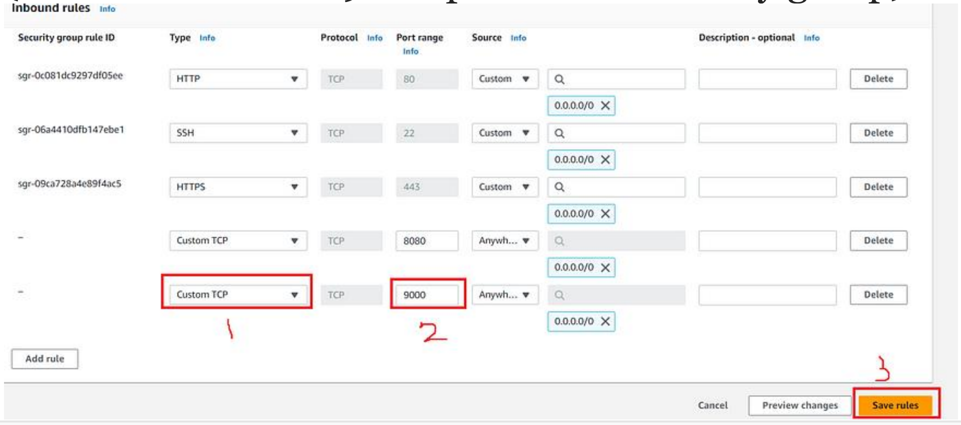
Configure a cloud →

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
```

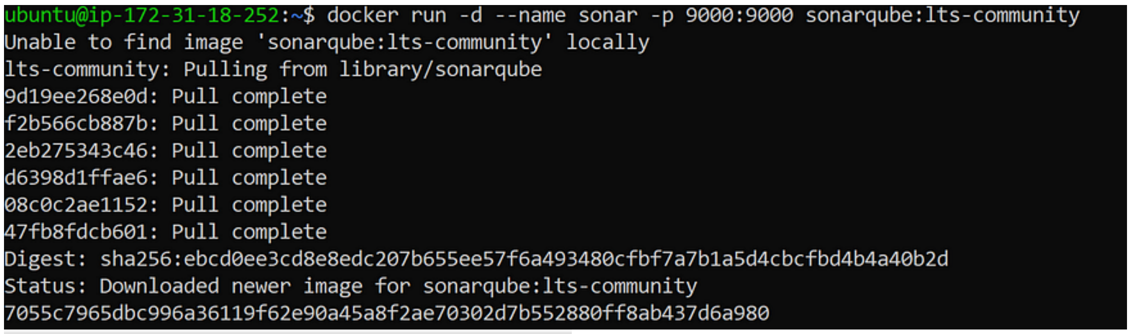
After the docker installation, we create a sonarqube container (Remember added 9000 port in the security group)



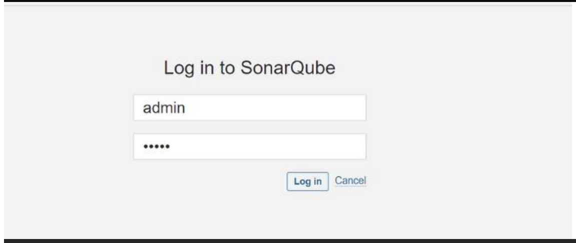
Inbound rules Info

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 -p9000:9000sonarqube:lts-community
```



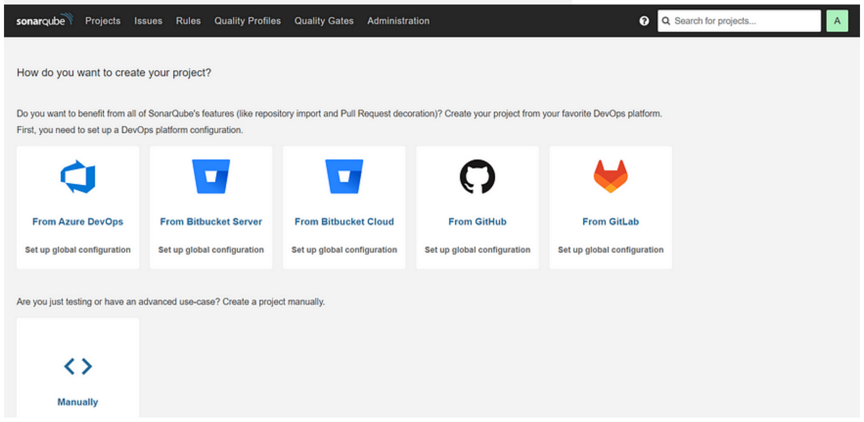
```
ubuntu@ip-172-31-18-252:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
9d19ee268e0d: Pull complete
f2b566cb887b: Pull complete
2eb275343c46: Pull complete
d6398d1ffae6: Pull complete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



Log in to SonarQube

admin

•••••



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.

From Azure DevOps
Set up global configuration
From Bitbucket Server
Set up global configuration
From Bitbucket Cloud
Set up global configuration
From GitHub
Set up global configuration
From GitLab
Set up global configuration

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

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

Install Plugin

Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

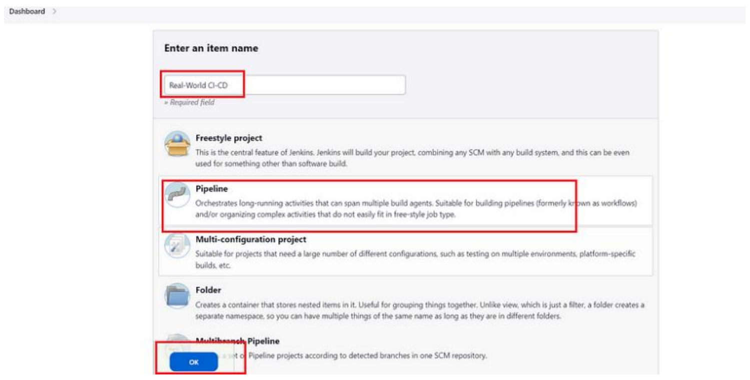
2 → SonarQube Scanner (Install without restart)

Configure Java and Maven in Global Tool Configuration

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

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.

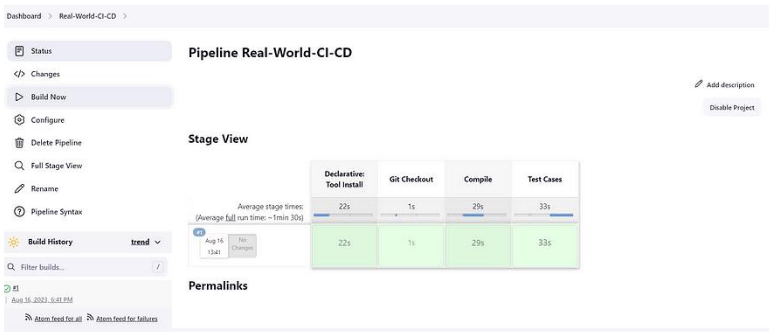
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
[illegible] Pipeline projects according to detected branches in one SCM repository.

Enter this in Pipeline Script,

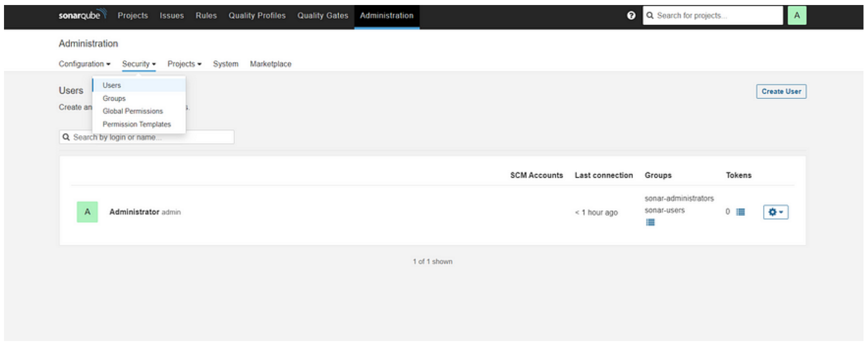
```
pipeline {
    agent any

    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"
            }
        }

         stage("Test Cases"){
            steps{
                sh "mvn test"
            }
        }
     }
}
```

The stage view would look like this,

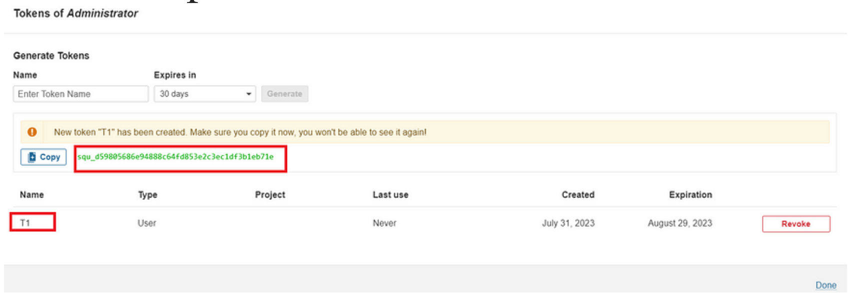


Step 4: Configure Sonar Server in Manage Jenkins

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



Click on Update Token



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

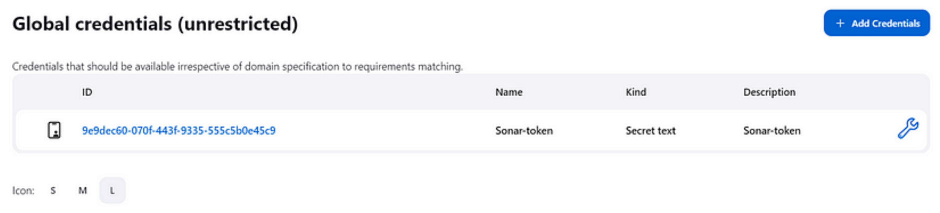
squ_d59805686e94888c64fd853e2c3ec1df3b1eb71e

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

Copy this Token

Goto Dashboard → Manage Jenkins → Credentials → Add Secret Text.

It should look like this



Global credentials (unrestricted)

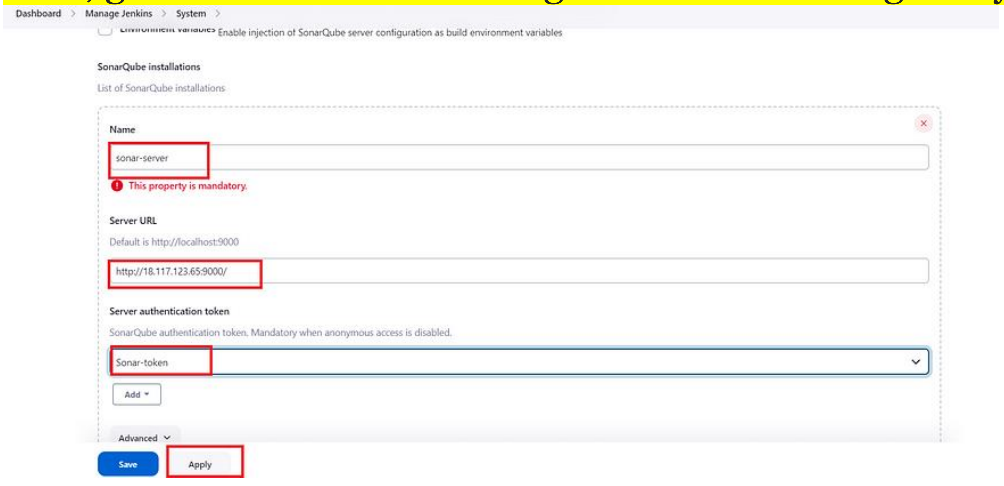
Add Credentials

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

Icon: S M L

Now, goto Dashboard → Manage Jenkins → Configure System



Dashboard > Manage Jenkins > 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

Add

Advanced

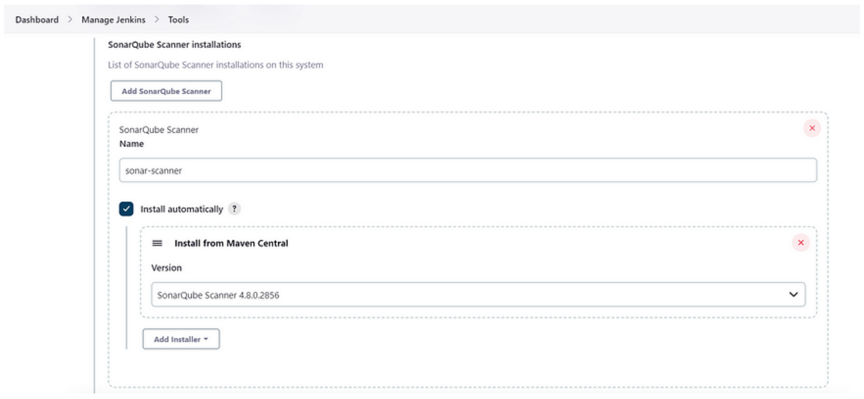
Save Apply

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.


Dashboard > Manage Jenkins > Tools

SonarQube Scanner installations

List of SonarQube Scanner installations on this system

Add SonarQube Scanner

SonarQube Scanner

Name

sonar-scanner

Install automatically

Install from Maven Central

Version

SonarQube Scanner 4.8.0.2856

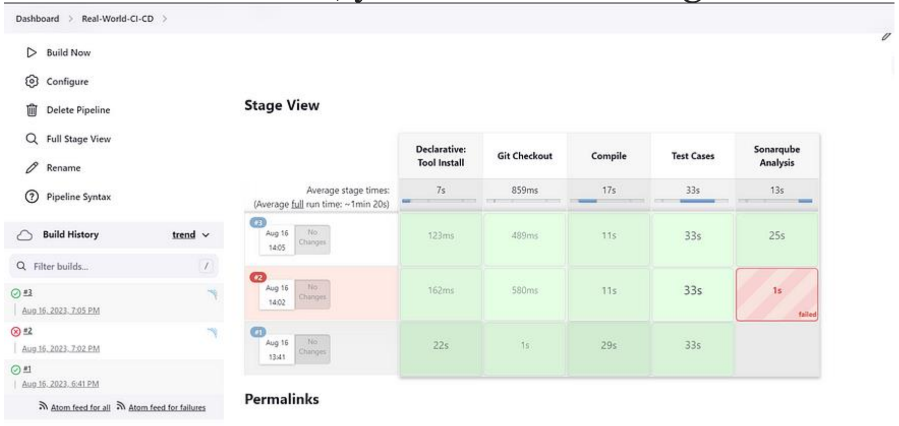
Add Installer

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

```
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{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Petclinic \
                        -Dsonar.java.binaries=. \
                        -Dsonar.projectKey=Petclinic '''

                    }
               }
          }

```

```
          stage("quality gate"){
steps {
          script {
          waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
          }
}
}
stage ('sonarqube Analysis'){
        steps{
          script{
            withSonarQubeEnv(credentialsId: 'Sonar-token') {
             sh 'mvn sonar:sonar'
            }
          }
        }
    }
      }
}
```

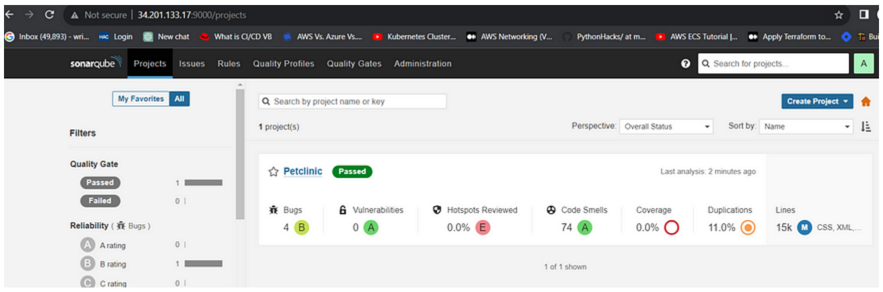
Click on Build now, you will see the stage view like this



Stage View

	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	

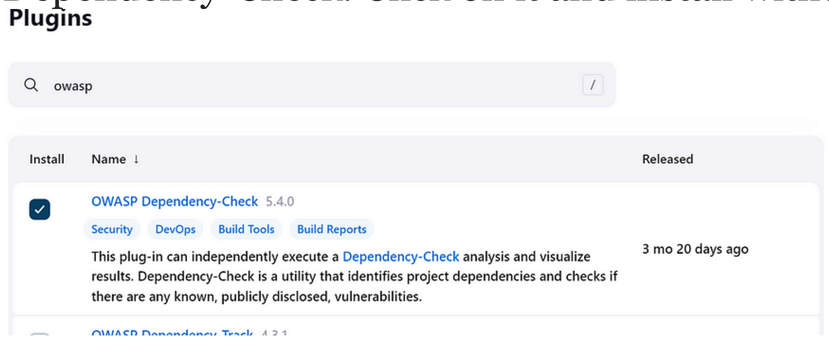
To see the report, you can goto Sonarqube Server and goto Projects.



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.



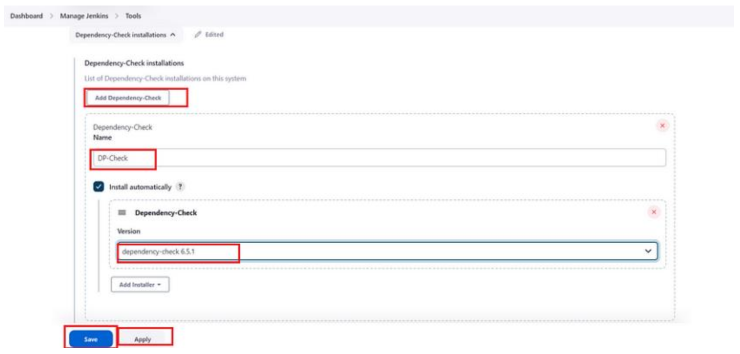
OWASP Dependency-Check 5.4.0

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.

3 mo 20 days ago

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

Goto Dashboard → Manage Jenkins → Tools →


Dashboard > Manage Jenkins > Tools
Dependency-Check installations
Edited
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.

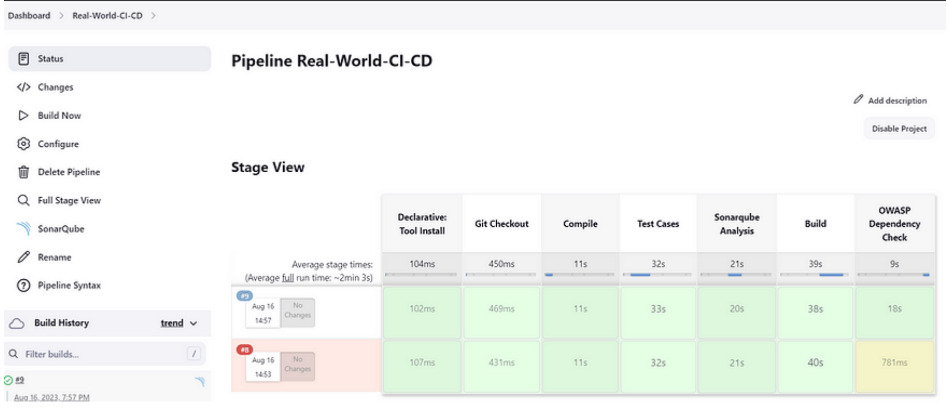
Now goto configure → Pipeline and add this stage to your pipeline

```
   stage("OWASP Dependency Check"){
            steps{
                dependencyCheck additionalArguments: '--scan ./ --format HTML
', odcInstallation: 'DP-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-
report.html'
            }
        }
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{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Petclinic \
                    -Dsonar.java.binaries=. \
                    -Dsonar.projectKey=Petclinic '''

                }
            }
        }


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

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

    }
}
```

The stage view would look like this,

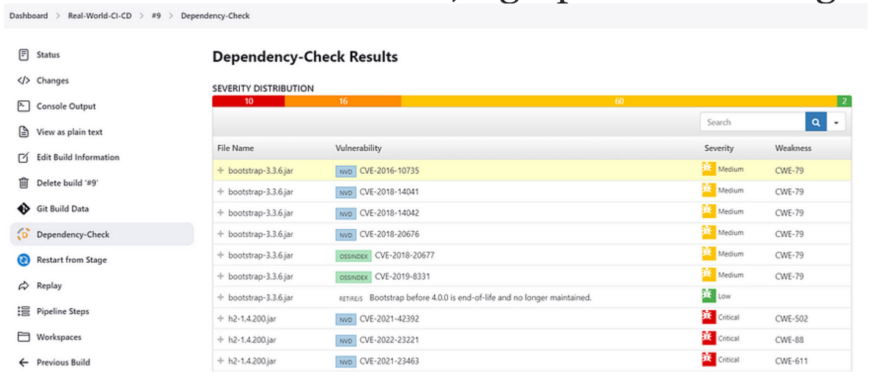

Dashboard > Real-World-CI-CD >

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

Dependency-Check Results

SEVERITY DISTRIBUTION

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

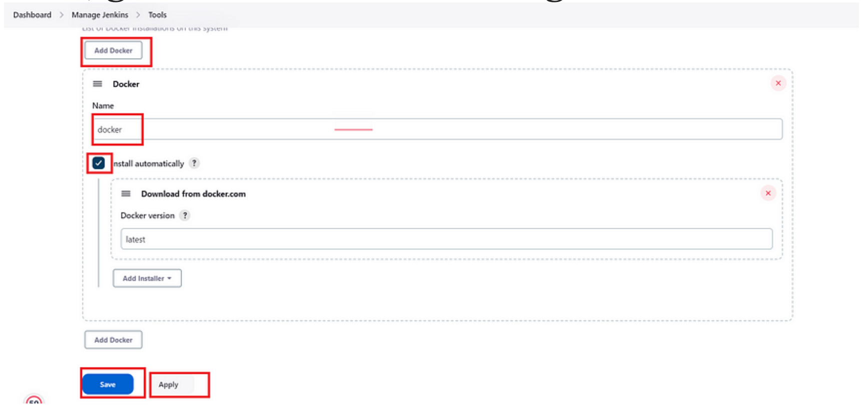
We need to install Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

- 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

Docker

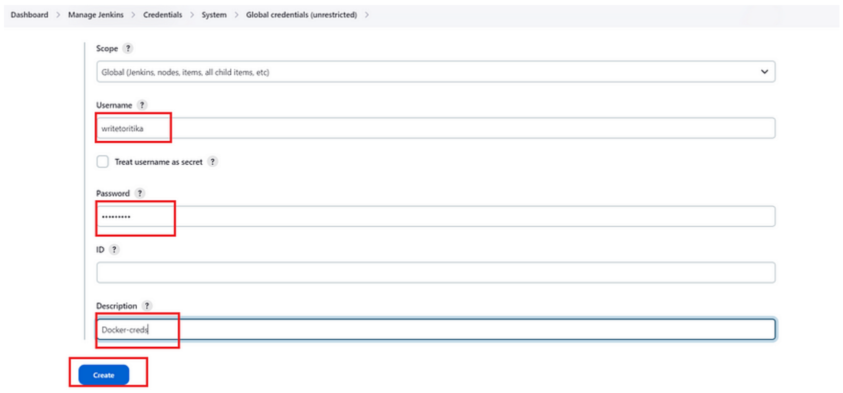
Name: docker

[✓] 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: •••••••••

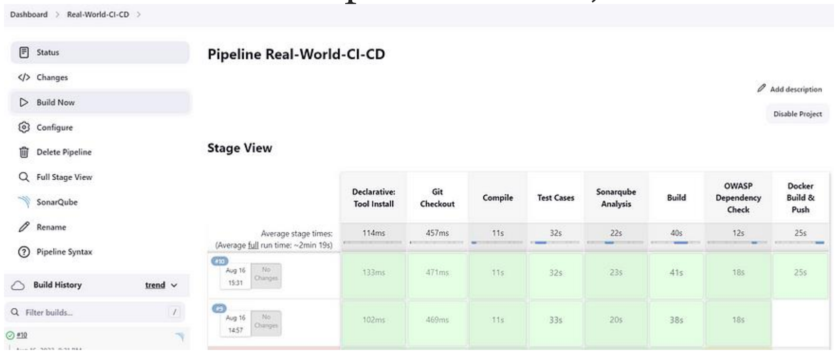
ID:

Description: Docker-creds

Add this stage in Pipeline Script

```
    stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: ‘docker', toolName: 
'docker') {
 sh "docker build -t petclinic1 ."
                       sh "docker tag petclinic1 Milky19/pet-clinic123:latest
"
  sh "docker push Milky19/pet-clinic123:latest "

                    }
                }
            }
        }
```

You will see the output like below,



Dashboard > Real-World-CI-CD >

Pipeline Real-World-CI-CD

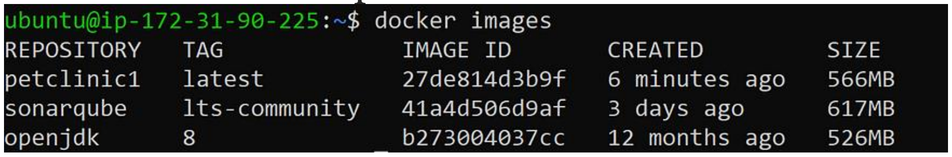
Stage View

	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
```

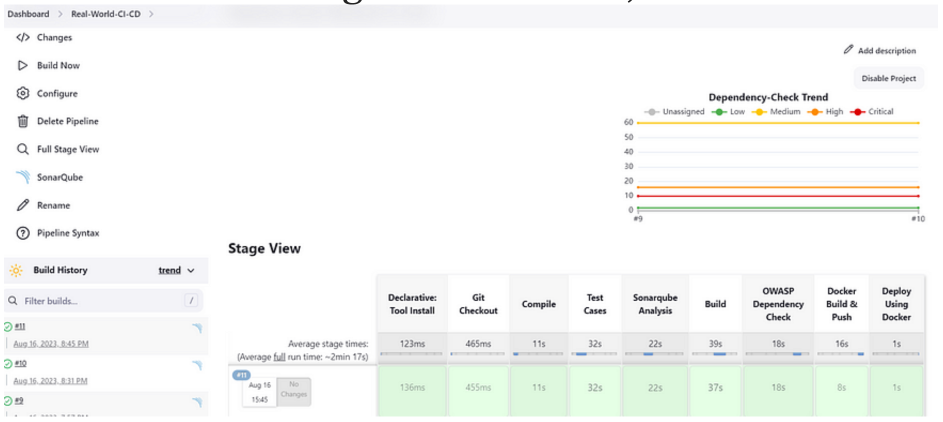
When you log in to Dockerhub, you will see a new image is created

Step 7 — Deploy image using Docker

Add this stage to your pipeline syntax

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

You will see the Stage View like this,



	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 — Install Tomcat on Port 8083 and finally deploy on Apache Tomcat using groovy pipeline script mentioned

Before we add Pipeline Script, we need to install and configure Tomcat on our server.

Here are the steps to install Tomcat 9

##################----INSTALL TOMCAT----##################

Commands are in Yellow color

--> change to opt directory

cd /opt

--> Download tomcat file using wget command

```
sudo wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.65/bin/apache-tomcat-9.0.65.tar.gz
```

```
sudo wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.80/bin/apache-tomcat-9.0.80.tar.gz (Another link )
```

--> Unzip tar file

```
sudo tar -xvf apache-tomcat-9.0.65.tar.gz
```

--> move to conf directory and change port in tomcat server to another port from default port

```
cd /opt/apache-tomcat-9.0.65/conf
vi server.xml
```

--> update tomcat users xml file for manager app login.

```
cd /opt/apache-tomcat-9.0.65/conf
sudo vi tomcat-users.xml
```

```
# ---add-below-line at the end (2nd-last line)----
<user username="admin" password="admin1234" roles="admin-gui, manager-gui"/>
```

--> create a symbolic links for direct start and stop of tomcat

```
sudo ln -s /opt/apache-tomcat-9.0.65/bin/startup.sh /usr/bin/startTomcat
sudo ln -s /opt/apache-tomcat-9.0.65/bin/shutdown.sh /usr/bin/stopTomcat
```

```
sudo vi /opt/apache-tomcat-9.0.65/webapps/manager/META-INF/context.xml
```

comment:

```
<!-- Valve className="org.apache.catalina.valves.RemoteAddrValve"
  allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
```

```
sudo vi /opt/apache-tomcat-9.0.65/webapps/host-manager/META-INF/context.xml
```

comment:

```
<!-- Valve className="org.apache.catalina.valves.RemoteAddrValve"
  allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
```

```
sudo stopTomcat
sudo startTomcat
```

Certainly! To allow both the `ubuntu` and `jenkins` users to copy the `petclinic.war` file to the `/opt/apache-tomcat-9.0.65/webapps/` directory without entering passwords, you can add the appropriate entries to the `/etc/sudoers` file. Here's how you can do it:

Open a terminal.

Use the `sudo` command to edit the sudoers file using a text editor like `visudo`:

```
sudo visudo
```

Scroll down to an appropriate section (e.g., just below the line with `%sudo ALL=(ALL:ALL) ALL`) and add the following lines:

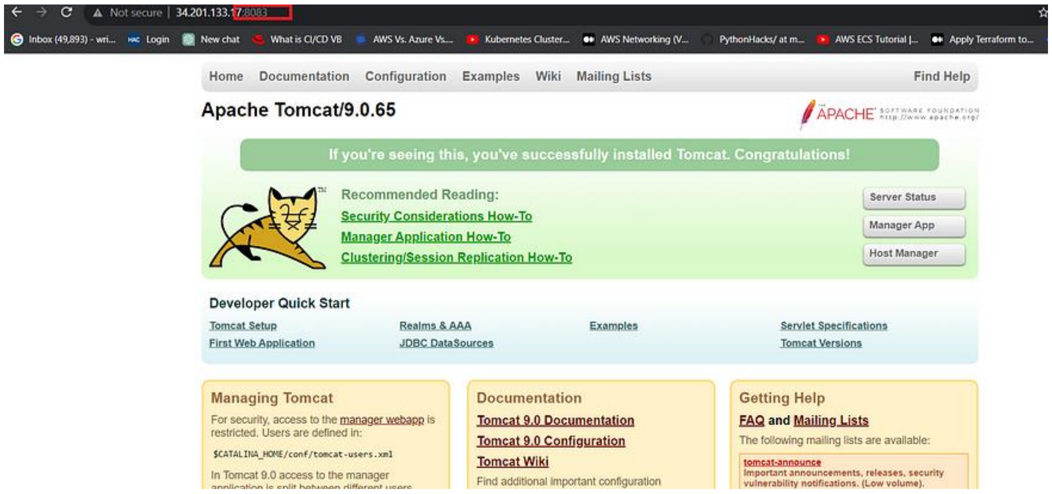
```
ubuntu ALL=(ALL) NOPASSWD: /bin/cp
/var/lib/jenkins/workspace/petclinic/target/petclinic.war /opt/apache-tomcat-9.0.65/webapps/
jenkins ALL=(ALL) NOPASSWD: /bin/cp
/var/lib/jenkins/workspace/petclinic/target/petclinic.war /opt/apache-tomcat-9.0.65/webapps/
```

Save the file and exit the text editor.

By adding these lines, you're allowing both the `ubuntu` user and the `jenkins` user to run the specified `cp` command without being prompted for a password.

After making these changes, both users should be able to run the Jenkins job that copies the `petclinic.war` file to the specified directory without encountering permission issues. Always ensure that you're cautious when editing the sudoers file and that you verify the paths and syntax before saving any changes.

SincePort 8080 is being used by Jenkins, we have used Port 8083 to host Tomcat Server


Not secure | 34.201.133.17:8083

Inbox (49,893) - wri... Login New chat What is CI/CD VB AWS Vs. Azure Vs... Kubernetes Cluster... AWS Networking (V... PythonHacks/ at m... AWS ECS Tutorial |... Apply Terraform to...

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/9.0.65

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To

Server Status
Manager App
Host Manager

Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 9.0 access to the manager

Documentation
Tomcat 9.0 Documentation
Tomcat 9.0 Configuration
Tomcat Wiki
Find additional important configuration

Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).

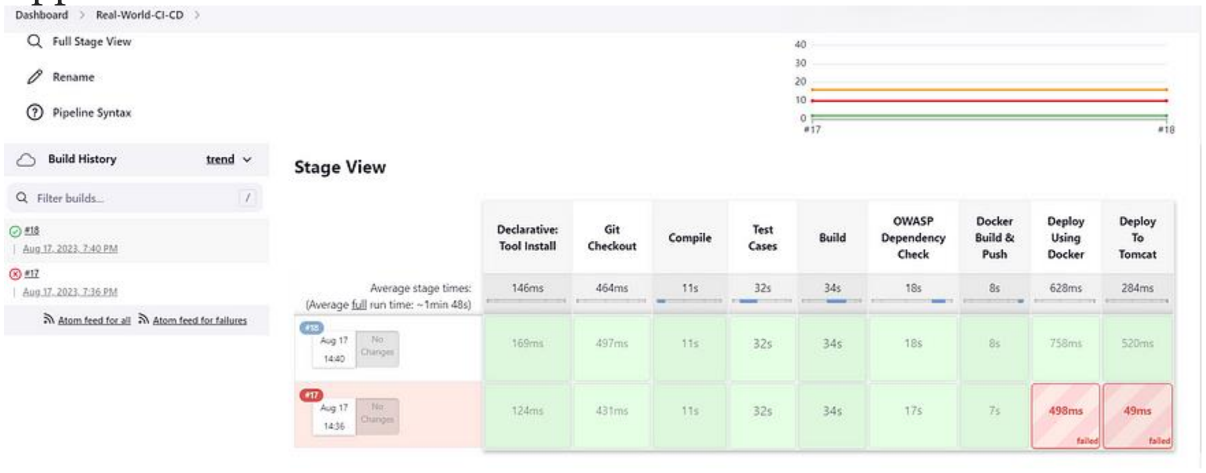
Add this stage to your Pipeline script

```
  stage("Deploy To Tomcat"){
            steps{
                sh "cp  /var/lib/jenkins/workspace/Real-World-CI-CD/target/petclinic.war /opt/apache-tomcat-9.0.65/webapps/ "
            }
        }
```

Kindly note that this application can be deployed via Docker and also via Tomcat Server.

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

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

        stage("Docker Build & Push"){
            steps{
                script{
                        withDockerRegistry(credentialsId: 'docker', toolName:
'docker') {


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

                             sh "docker push hanvitha/pet-clinic123:latest "

                    }
                }
            }
        }

        stage("Deploy Using Docker"){
            steps{
                sh " docker run -d --name pet12 -p 8082:8082 hanvitha/pet-
clinic123:latest "
            }
        }

        stage("Deploy To Tomcat"){
            steps{
                sh "cp  target/petclinic.war /opt/apache-tomcat-
9.0.65/webapps/ "
            }
        }
    }
}
```

And you can access your application on Port 8083. This is a Real World Application that has all Functional Tabs.


Dashboard > Real-World-CI-CD >

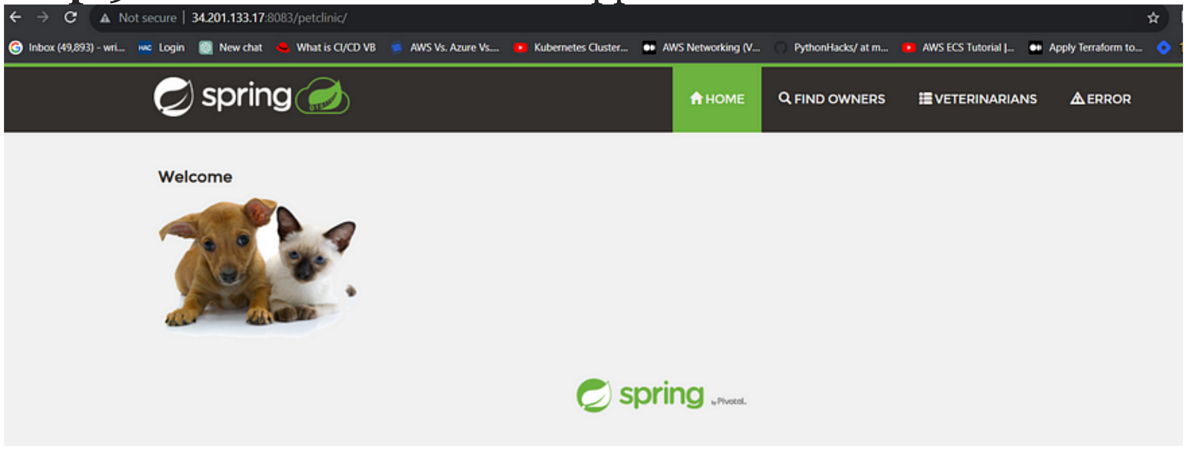
Full Stage View
Rename
Pipeline Syntax

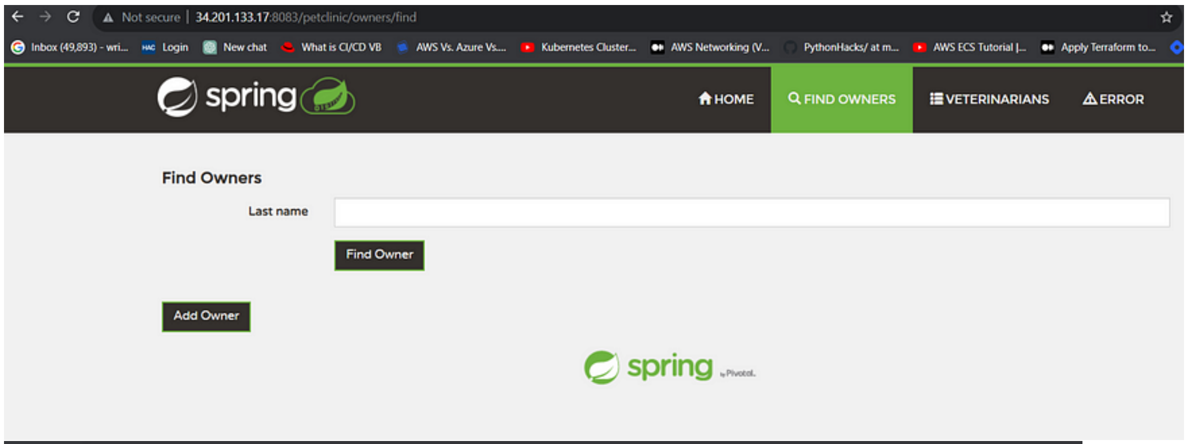
Build History trend
Filter builds...
#18 Aug 17, 2023, 7:40 PM
#17 Aug 17, 2023, 7:36 PM
Atom feed for all Atom feed for failures

Stage View

	Declarative: Tool Install	Git Checkout	Compile	Test Cases	Build	OWASP Dependency Check	Docker Build & Push	Deploy Using Docker	Deploy To Tomcat
Average stage times: (Average full run time: ~1min 48s)	146ms	464ms	11s	32s	34s	18s	8s	628ms	284ms
#18 Aug 17 14:40 No Changes	169ms	497ms	11s	32s	34s	18s	8s	758ms	520ms
#17 Aug 17 14:36 No Changes	124ms	431ms	11s	32s	34s	17s	7s	498ms failed	49ms failed

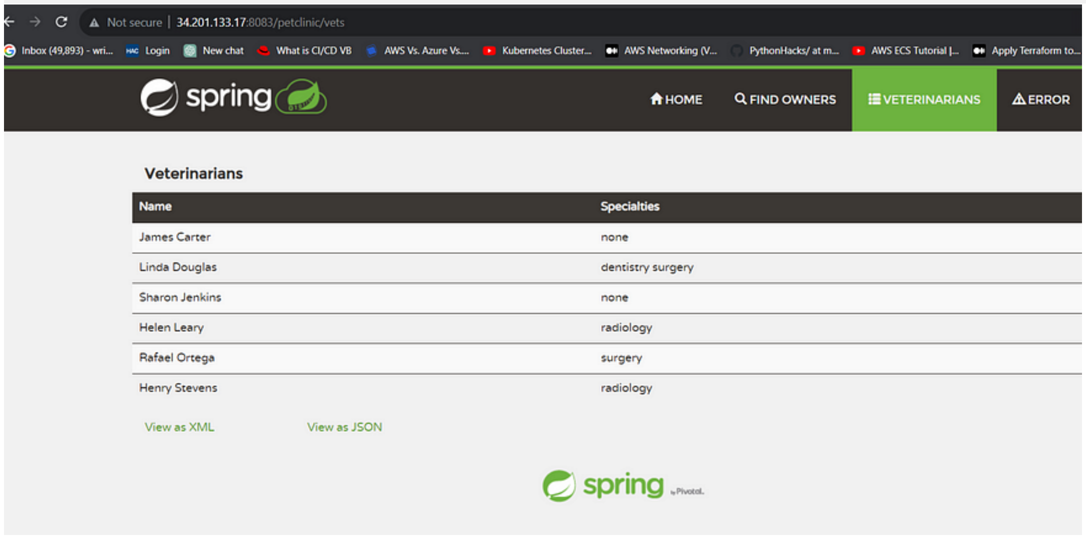
Step 9: Access the Real World Application




Not secure | 34.201.133.17:8083/petclinic/owners/find

Find Owners

Last name


Not secure | 34.201.133.17:8083/petclinic/vets

Veterinarians

Name	Specialties
James Carter	none
Linda Douglas	dentistry surgery
Sharon Jenkins	none
Helen Leary	radiology
Rafael Ortega	surgery
Henry Stevens	radiology

STEP: 10 Take Two Ubuntu 20.04 instances one for k8s master and other one for worker also install on Jenkins machine (only kubectl)

Kubectl on Jenkins to be installed

sudo apt update
sudo apt install curl
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version –client

Part 1 ----------------------------Master ------------

sudo su
hostname master
bash

clear

Node --------------

sudo su
hostname master
bash
clear

Part 2 ---Both Master & Node -------------

sudo apt-get update && sudo apt-get upgrade -y

sudo apt-get install -y docker.io
sudo usermod –aG docker Ubuntu
newgrp docker
sudo chmod 777 /var/run/docker.sock

sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

snap install kube-apiserver

Part 3 --- Master ---------------

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Node ----------

paster the kube adm join command which is in this format: sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --discovery-token-ca-cert-hash <hash>

Part 4 --

Master -----------------

kubectl get nodes

CONGRATULATIONS FOR YOUR NEW KUBERNETES CLUSTER ON UBUNTU ON EC2

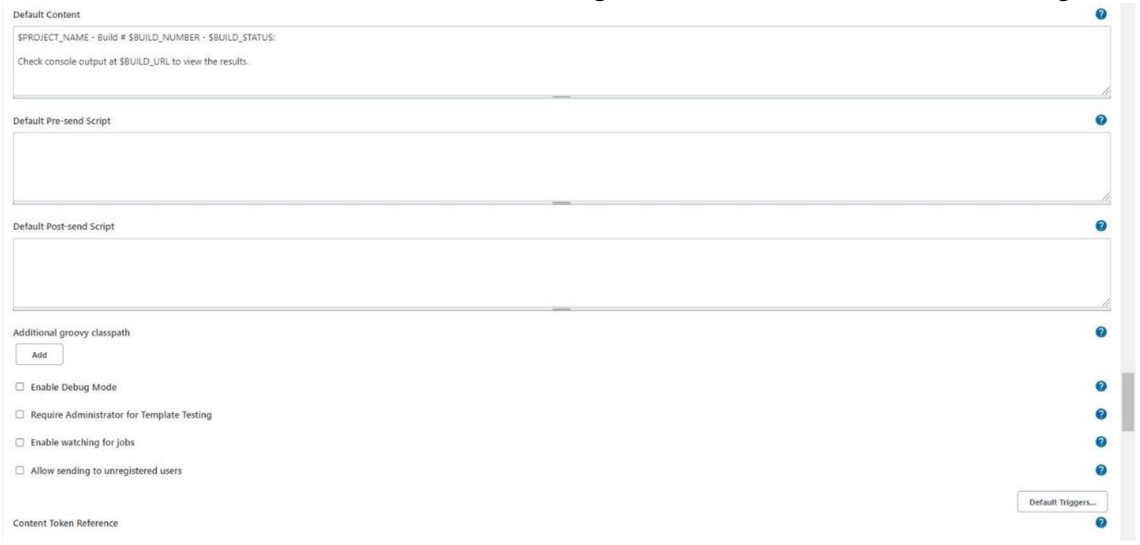
Copy config file to Jenkins master or to local file manager and save it
Install Kubernetes plugins

Install Kubernetes Plugin, once its installed successfully. goto manage jenkins --> manage credentials --> Click on jenkins global --> add credentials

Configuring mail server in Jenkins (Gmail)

Install Email Extension Plugin in Jenkins
Once plugin installed in jenkins, click on manage jenkins --> configure system there under E-mail Notification section configure the details as shown in below image

this is to just verify mail configuration
Now under Extended E-mail Notification section configure the details as shown in below images



Default Content

```
$PROJECT_NAME - Build # $BUILD_NUMBER - $BUILD_STATUS:

Check console output at $BUILD_URL to view the results.
```

Default Pre-send Script

Default Post-send Script

Additional groovy classpath

- ☐ Enable Debug Mode
- ☐ Require Administrator for Template Testing
- ☐ Enable watching for jobs
- ☐ Allow sending to unregistered users

Content Token Reference

By using below code i can send customized mail

```
post {
             always {
                    mail bcc: '', body: "<br>Project: ${env.JOB_NAME} <br>Build
Number: ${env.BUILD_NUMBER} <br> URL de build: ${env.BUILD_URL}", cc: '', charset:
'UTF-8', from: '', mimeType: 'text/html', replyTo: '', subject:
"${currentBuild.result} CI: Project name -> ${env.JOB_NAME}", to:
"krishna04.b@gmail.com";
             }
       }
```

also which ever the mail you use for authentication in that mail setting "Less secure apps access" should be enabled

Step 11: Terminate the AWS EC2 Instance

Total script:

```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        maven 'maven3'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanWs()
            }
        }
        stage('Checkout From Git'){
            steps{
                git branch: 'main', url: 'https://github.com/Milky19/Petclinic-
Real.git'
            }
        }
        stage('mvn compile'){
            steps{
                sh 'mvn clean compile'
            }
        }
        stage('mvn test'){
            steps{
                sh 'mvn test'
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Petclinic \
                    -Dsonar.java.binaries=. \
                    -Dsonar.projectKey=Petclinic '''
                }
            }
        }
        stage("quality gate"){
           steps {
                 script {
                     waitForQualityGate abortPipeline: false, credentialsId:
'Sonar-token'
                    }
                }
        }
        stage('mvn build'){
            steps{
                sh 'mvn clean install'
            }
        }
        stage("OWASP Dependency Check"){
            steps{
                dependencyCheck additionalArguments: '--scan ./ --format HTML ',
odcInstallation: 'DP-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-
report.html'
            }
        }
        stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName:
'docker'){
                       sh "docker build -t petclinic1 ."
                       sh "docker tag petclinic1 hanvitha/petclinic1:latest "
                       sh "docker push hanvitha/petclinic1:latest "
                    }
                }
            }
        }
        stage("TRIVY"){
            steps{
                sh "trivy image hanvitha/petclinic1:latest> trivy.txt"
            }
        }
        stage('Clean up containers') {   //if container runs it will stop and
remove this block
          steps {
           script {
             try {
                sh 'docker stop pet1'
                sh 'docker rm pet1'
                } catch (Exception e) {
                  echo "Container pet1 not found, moving to next stage"
                }
            }
          }
        }
        stage ('Manual Approval'){
          steps {
           script {
             timeout(time: 10, unit: 'MINUTES') {
              def approvalMailContent ="""
              Project: ${env.JOB_NAME}
              Build Number: ${env.BUILD_NUMBER}
              Go to build URL and approve the deployment request.
              URL de build: ${env.BUILD_URL}
              """
             mail(
             to: 'krishna04.b@gmail.com',
             subject: "${currentBuild.result} CI: Project name -
>${env.JOB_NAME}",
             body: approvalMailContent,
             mimeType: 'text/plain'
             )
            input(
            id: "DeployGate",
            message: "Deploy ${params.project_name}?",
            submitter: "approver",
            parameters: [choice(name: 'action', choices: ['Deploy'], description:
'Approve deployment')]
            )
          }
         }
       }
    }
        stage('Deploy to conatiner'){
            steps{
                sh 'docker run -d --name pet1 -p 8082:8080
hanvitha/petclinic1:latest'
```

```
            }
        }
        stage("Deploy To Tomcat"){
            steps{
                sh "sudo
cp  /var/lib/jenkins/workspace/petclinic/target/petclinic.war /opt/apache-tomcat-
9.0.65/webapps/ "
            }
        }
        stage('Deploy to kubernets'){
            steps{
                script{
                    withKubeConfig(caCertificate: '', clusterName: '',
contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess:
false, serverUrl: '') {
                       sh 'kubectl apply -f deployment.yaml'
                  }
                }
            }
        }
    }
    post {
     always {
        emailext attachLog: true,
            subject: "'${currentBuild.result}'",
            body: "Project: ${env.JOB_NAME}<br/>"+
                "Build Number: ${env.BUILD_NUMBER}<br/>"+
                "URL: ${env.BUILD_URL}<br/>",
            to: 'hanvitha@gmail.com',
            attachmentsPattern: 'trivy.txt'
        }
    }
}


// try this approval stage also

stage('Manual Approval') {
  timeout(time: 10, unit: 'MINUTES') {
    mail to: 'krishna04.b@gmail.com',
         subject: "${currentBuild.result} CI: ${env.JOB_NAME}",
         body: "Project: ${env.JOB_NAME}\nBuild Number: ${env.BUILD_NUMBER}\nGo
to ${env.BUILD_URL} and approve deployment"
    input message: "Deploy ${params.project_name}?",
           id: "DeployGate",
           submitter: "approver",
           parameters: [choice(name: 'action', choices: ['Deploy'], description:
'Approve deployment')]
  }
}
```