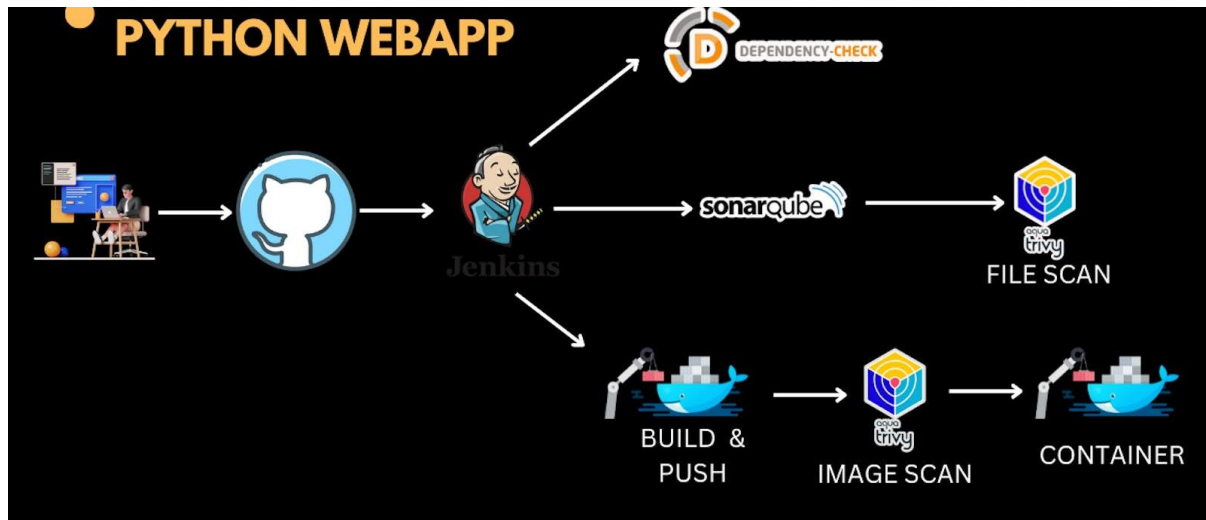


## CI-CD DevSecOps project with Jenkins | Python webapp



Step 1 — Launch an AWS T2 Large Instance.

Step 2 — Install Jenkins, Docker and Trivy

2A — To Install Jenkins

2B — Install Docker

2C — Install Trivy

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

3A — Install Plugin

3B — Configure Java and Maven in Global Tool Configuration

3C — Create a Job

Step 4 — Install OWASP Dependency Check Plugins

Step 5 — Configure Sonar Server in Manage Jenkins

Step 6 — we have to install make package

Step 7 — Docker Image Build and Push

Step 8 — Deploy the image using Docker

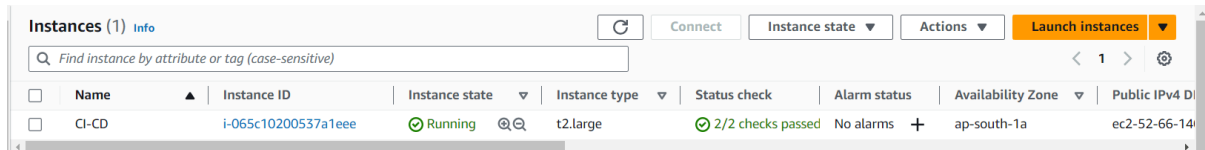
Step 9 — Access the Real World Application

Step 10 — Terminate the AWS EC2 Instance

<https://github.com/Milky19/Python-System-Monitoring.git>

## STEP1:Launch an Ubuntu(22.04) T2 Large Instance

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 and open all ports (not best case to open all ports but just for learning purposes it's okay).



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
<input type="checkbox"/>	CI-CD	i-065c10200537a1eee	Running	t2.large	2/2 checks passed	No alarms	ap-south-1a	ec2-52-66-14

## Step 2 — Install Jenkins, Docker and Trivy

### 2A — To Install Jenkins

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

```
vi jenkins.sh
```

```
#!/bin/bash
```

```
sudo apt update -y
```

```
#sudo apt upgrade -y
```

```
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee  
/etc/apt/keyrings/adoptium.asc
```

```
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc]  
https://packages.adoptium.net/artifactory/deb ${awk -F= '/^VERSION_CODENAME/{print$2}'  
/etc/os-release} main" | tee /etc/apt/sources.list.d/adoptium.list
```

```
sudo apt update -y
```

```
sudo apt install temurin-17-jdk -y
```

```
/usr/bin/java --version
```

```
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
```

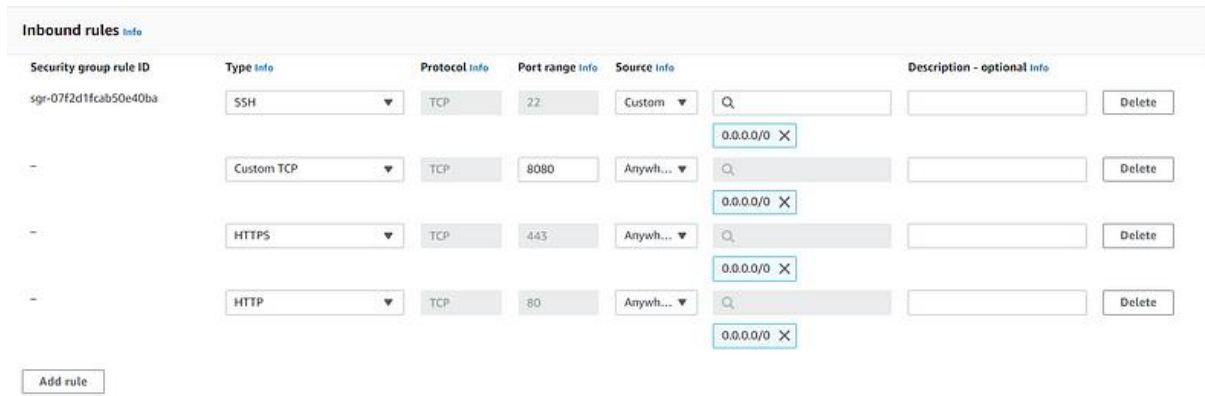
```
sudo apt-get update -y
```

```
sudo apt-get install jenkins -y
```

```
sudo systemctl start jenkins
```

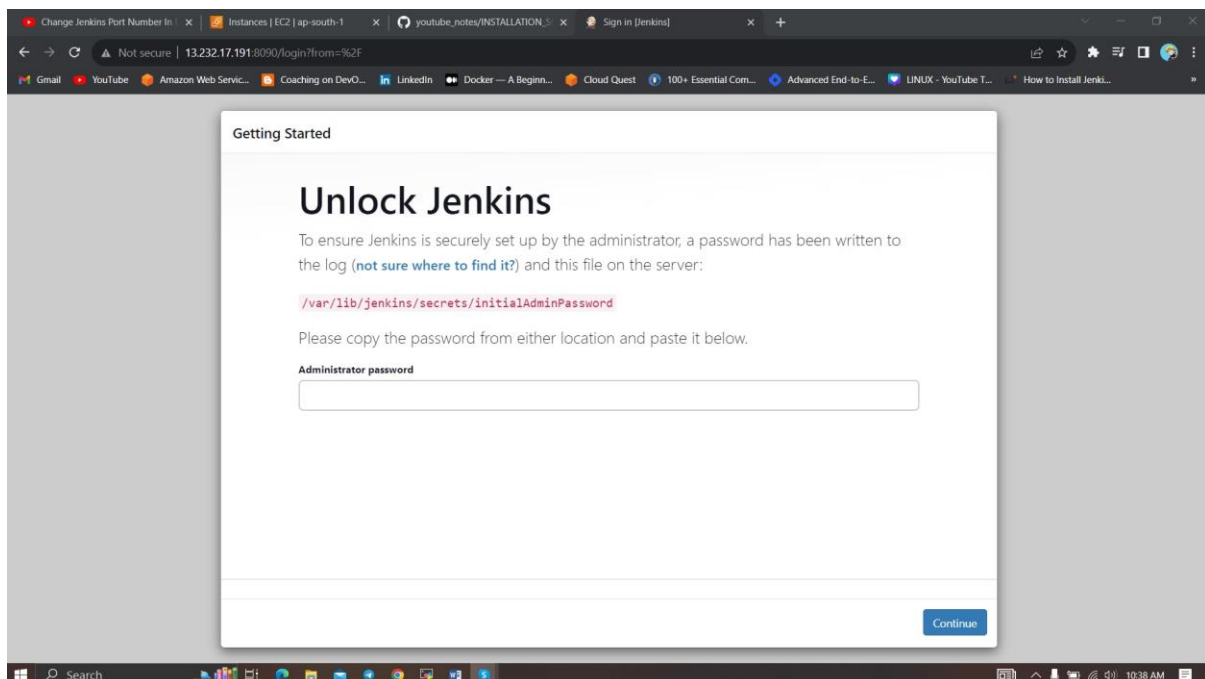
```
sudo systemctl status jenkins
```

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.

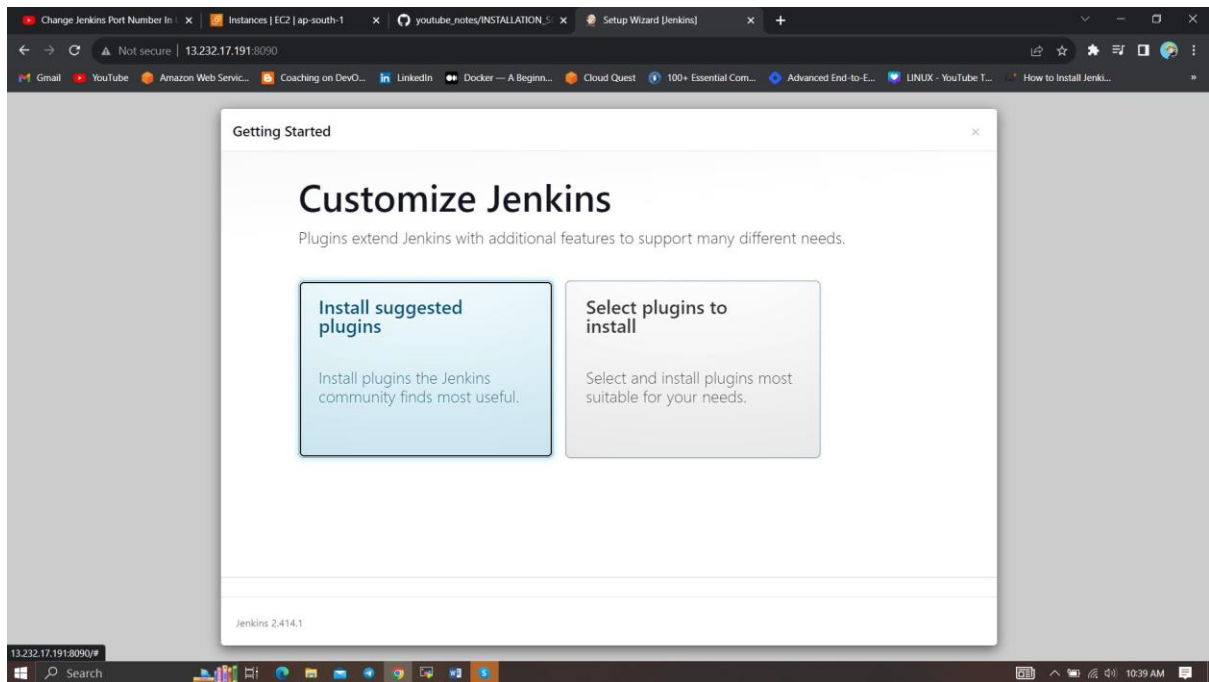


<EC2 Public IP Address:8080>

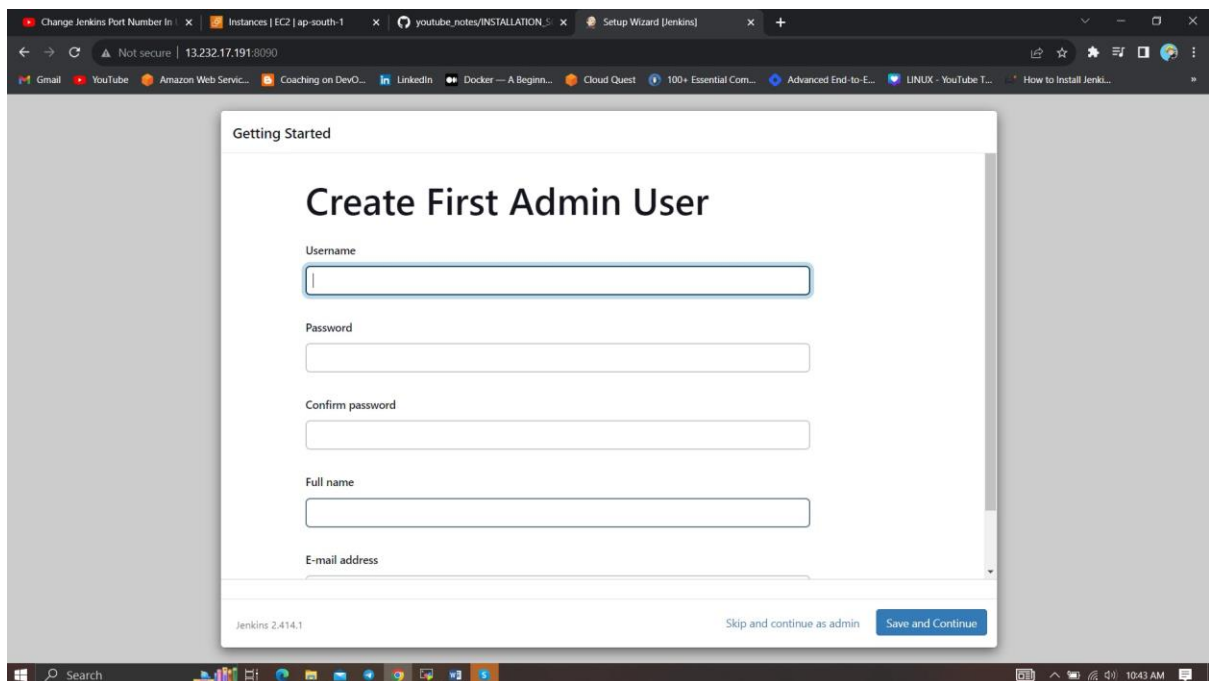
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



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

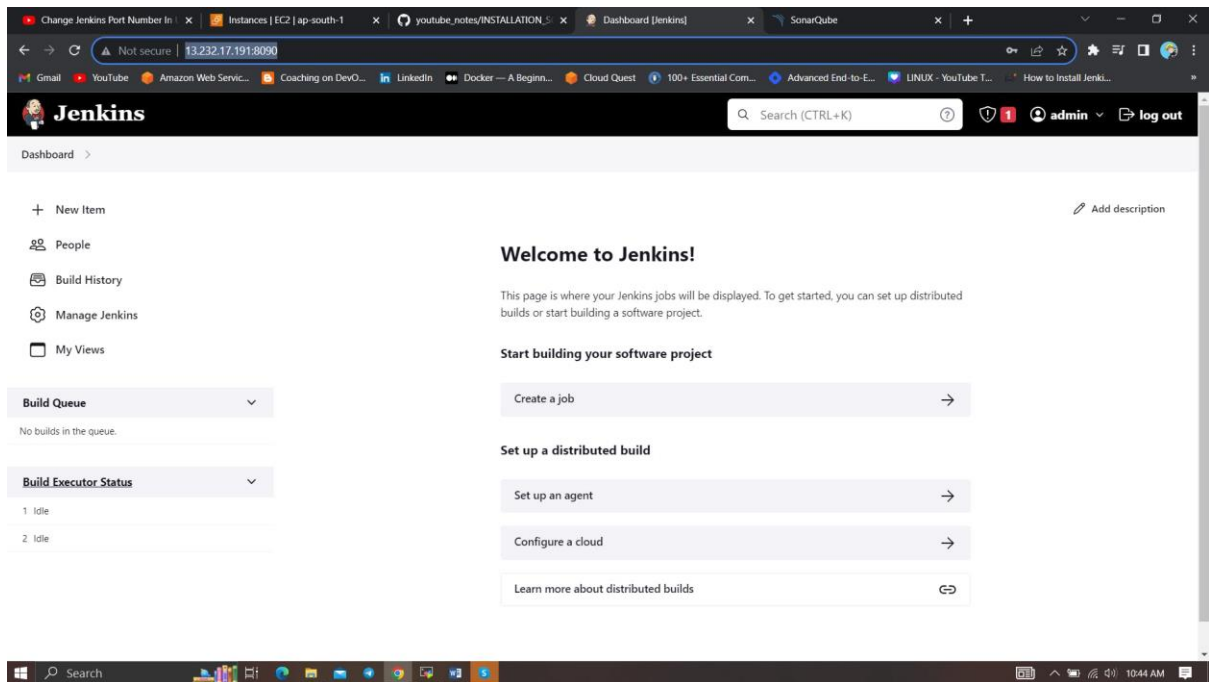


Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

**Jenkins Getting Started Screen.**



## 2B — Install Docker

`sudo apt-get update`

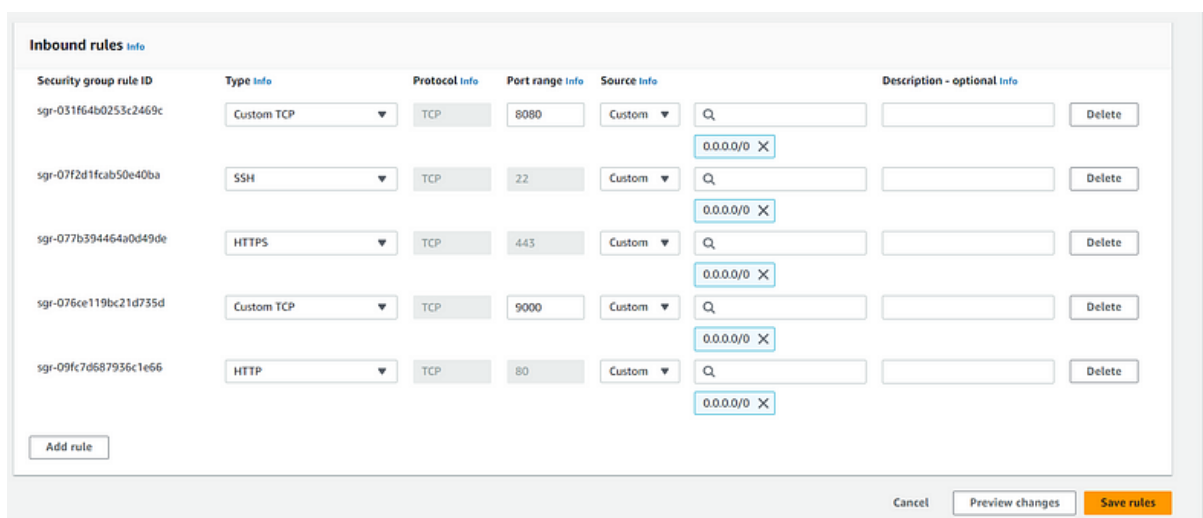
`sudo apt-get install docker.io -y`

`sudo usermod -aG docker $USER #my case is ubuntu`

`newgrp docker`

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

After the docker installation, we create a sonarqube container  
(Remember to add 9000 ports in the security group).



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

```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ 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
44ba2882f8eb: Pull complete
2cabec57fa36: Pull complete
c20481384b6a: Pull complete
bf7b1ee74f8: Pull complete
38617fae714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b66c96bf9ad3d62289436af7f752fdb04993092d0ca3065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS                    NAMES
4b66c96bf9ad   sonarqube:lts-commu   "/opt/sonarqube/dock...  9 seconds ago Up 5 seconds    0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   sonar
ubuntu@ip-172-31-42-253:~$
```

Now our sonarqube is up and running

Log in to SonarQube

Login

Password

Log in Cancel

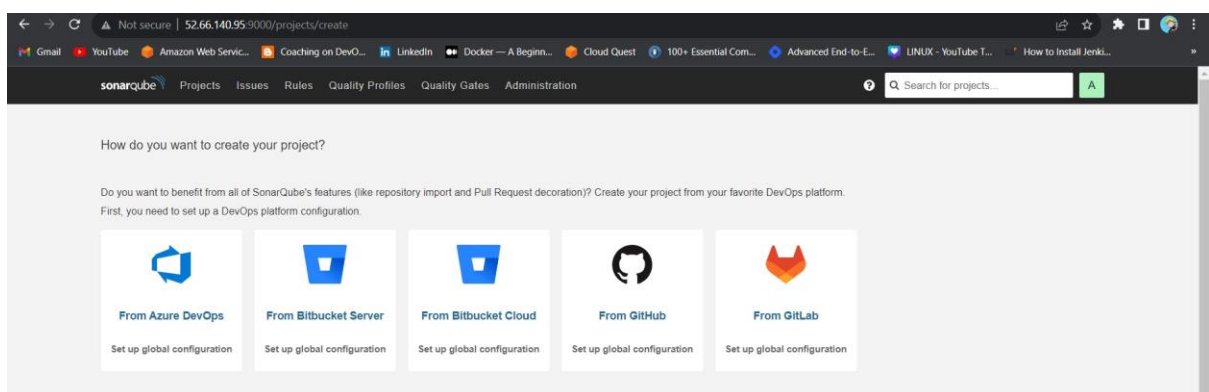
Enter username and password, click on login and change password

**username admin**

**password admin**

The screenshot shows a web browser window with the address bar displaying `52.66.140.95:9000/account/reset_password`. The browser tabs include 'Instances | EC2 | ap-south-1', 'petstore Config [Jenkins]', and 'SonarQube'. The page content is a white box with the title 'Update your password'. Below the title, it says 'This account should not use the default password.' and 'Enter a new password'. A note states 'All fields marked with \* are required'. There are three input fields: 'Old Password \*', 'New Password \*', and 'Confirm Password \*'. An 'Update' button is at the bottom of the form.

Update New password, This is Sonar Dashboard.



## 2C — Install Trivy

`vi trivy.sh`

`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 log in to Jenkins and start to configure our Pipeline in Jenkins

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

**3A — Install Plugin**

**Goto Manage Jenkins → Plugins → Available Plugins →**

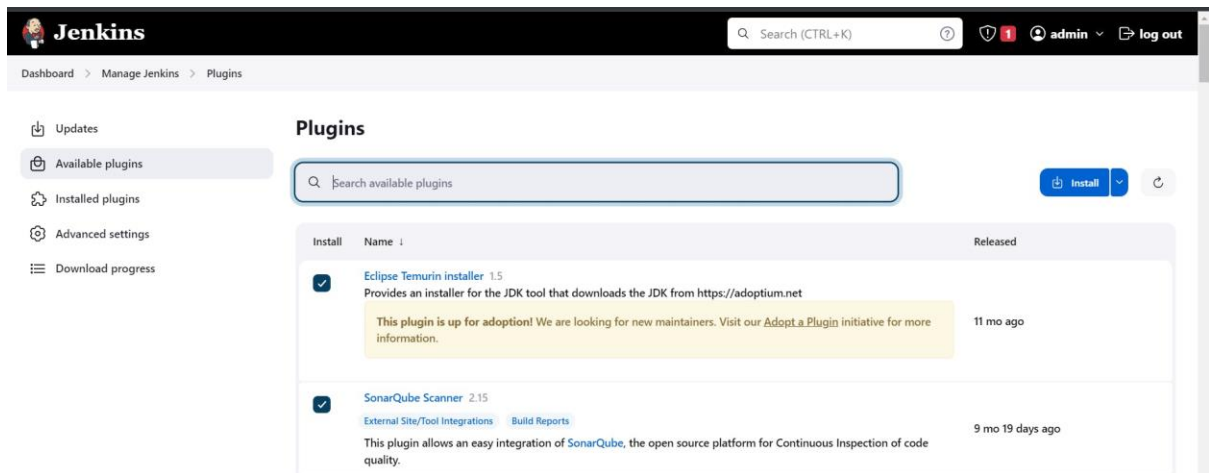
**Install below plugins**

**1 → Eclipse Temurin Installer (Install without restart)**

**2 → SonarQube Scanner (Install without restart)**

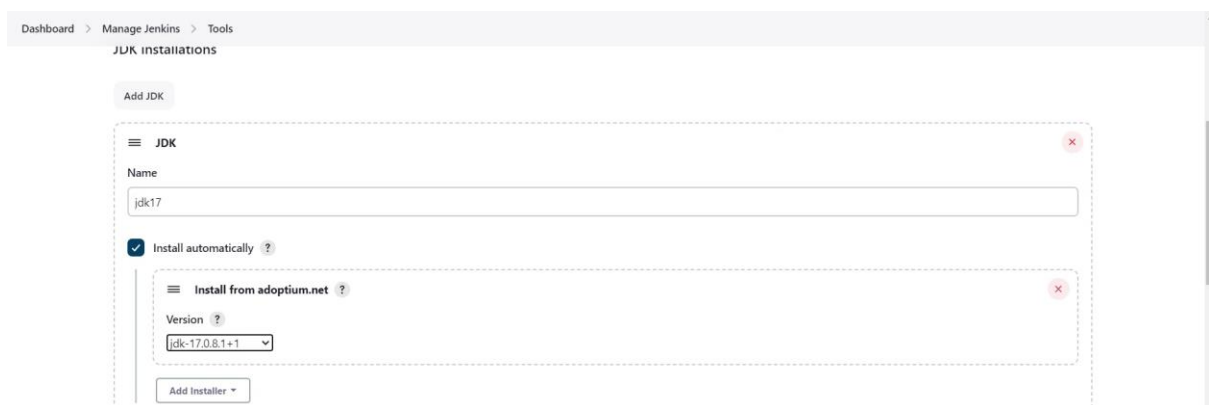
**3 → NodeJs Plugin (Install Without restart)**





## 3B — Configure Java and Maven in Global Tool Configuration

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



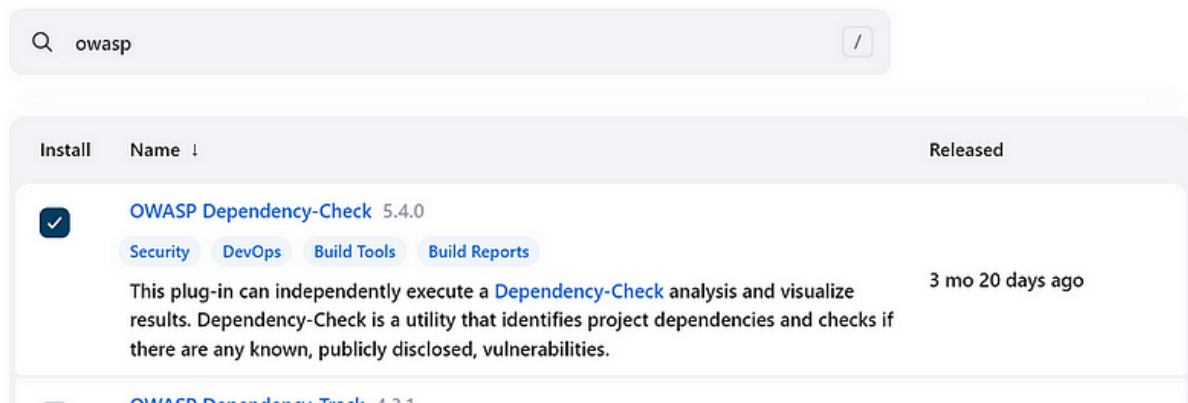
## 3C — Create a Job

Label it as Dotnet CI-CD, click on Pipeline and OK.

## Step 4 — Install OWASP Dependency Check Plugins

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

### Plugins



First, we configured the Plugin and next, we had to configure the Tool

Goto Dashboard → Manage Jenkins → Tools →

## Dependency-Check installations

Add Dependency-Check

### Dependency-Check

Name

DP-Check

☒ Install automatically ?

### Install from github.com

Version

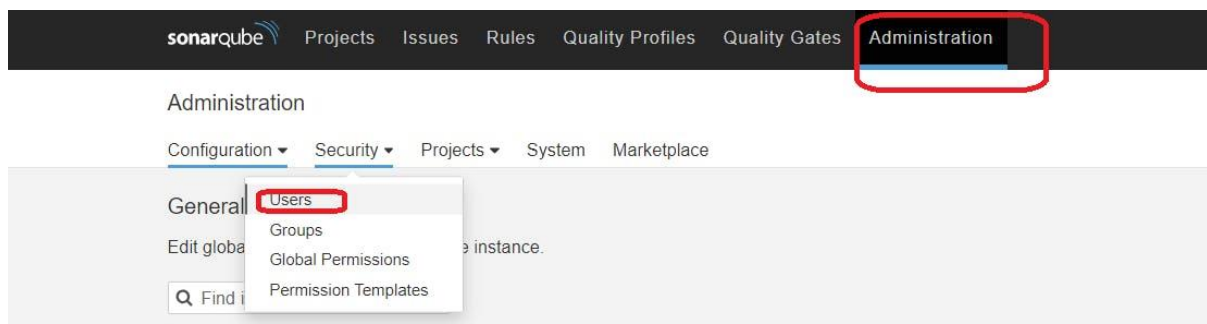
dependency-check 6.5.1

Add Installer ▾

Click on Apply and Save here.

## Step 5 — 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

SCM Accounts

Last connection

Groups

Tokens

A

Administrator admin

< 1 hour ago

sonar-administrators

sonar-users

0

Update Tokens

## Create a token with a name and generate

### Tokens of Administrator

#### Generate Tokens

Name

Expires in

Enter Token Name

30 days

Generate

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

Copy

sq\_u\_21d152904c1c72cf8b39665f96480185c99dc2f9

Name	Type	Project	Last use	Created	Expiration	
Jenkins	User		Never	September 8, 2023	October 8, 2023	Revoke

## Copy this Token

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

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

New credentials

Kind

Secret text

Scope ?

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

Secret

POST THE TOKEN HERE

ID ?

Sonar-token

Description ?

Sonar-token

Create

## You will this page once you click on create

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

ID	Name	Kind	Description	
Sonar-token	sonar	Secret text	sonar	

Now, go to Dashboard → Manage Jenkins → Configure System

Dashboard > Manage Jenkins > System

### SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

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

SonarQube installations

List of SonarQube installations

Name

sonar-server

Server URL

Default is http://localhost:9000

http://13.232.17.191:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Sonar-token

Add

Save Apply

Click on Apply and Save

**The 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 a sonar scanner in the tools.

Dashboard > Manage Jenkins > Tools

### SonarQube Scanner installations

Add SonarQube Scanner

☰ SonarQube Scanner

Name

sonar-scanner

☒ Install automatically ?

☰ Install from Maven Central

Version

SonarQube Scanner 5.0.1.3006

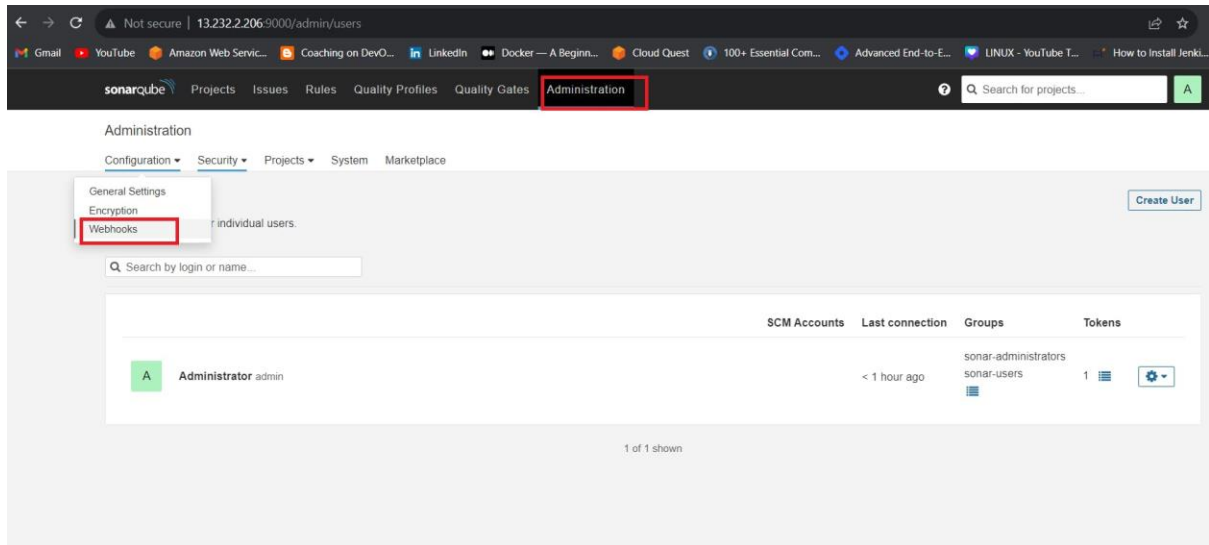
Add Installer

Add SonarQube Scanner

Save Apply

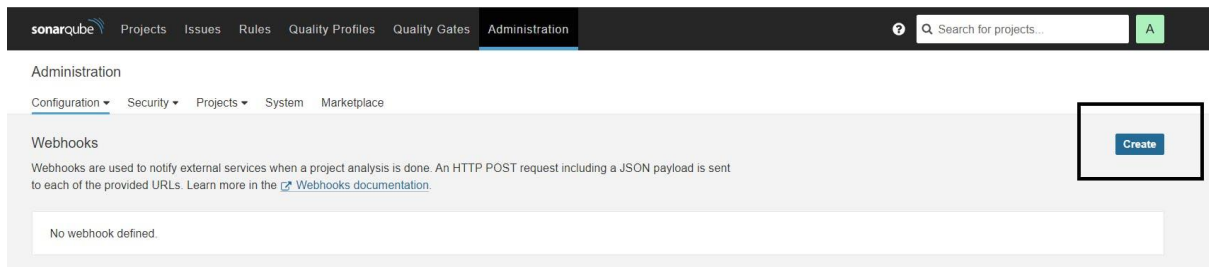
In the Sonarqube Dashboard add a quality gate also

## Administration--> Configuration-->Webhooks



The screenshot shows the SonarQube Administration interface. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration (highlighted with a red box). Below the navigation bar, the 'Administration' section is active, and the 'Configuration' sub-menu is expanded, showing 'General Settings', 'Encryption', and 'Webhooks' (highlighted with a red box). The 'Webhooks' page displays a table with columns for SCM Accounts, Last connection, Groups, and Tokens. The table shows one entry for 'Administrator admin' with a last connection of '< 1 hour ago' and two groups: 'sonar-administrators' and 'sonar-users'. A 'Create User' button is visible in the top right corner.

## Click on Create

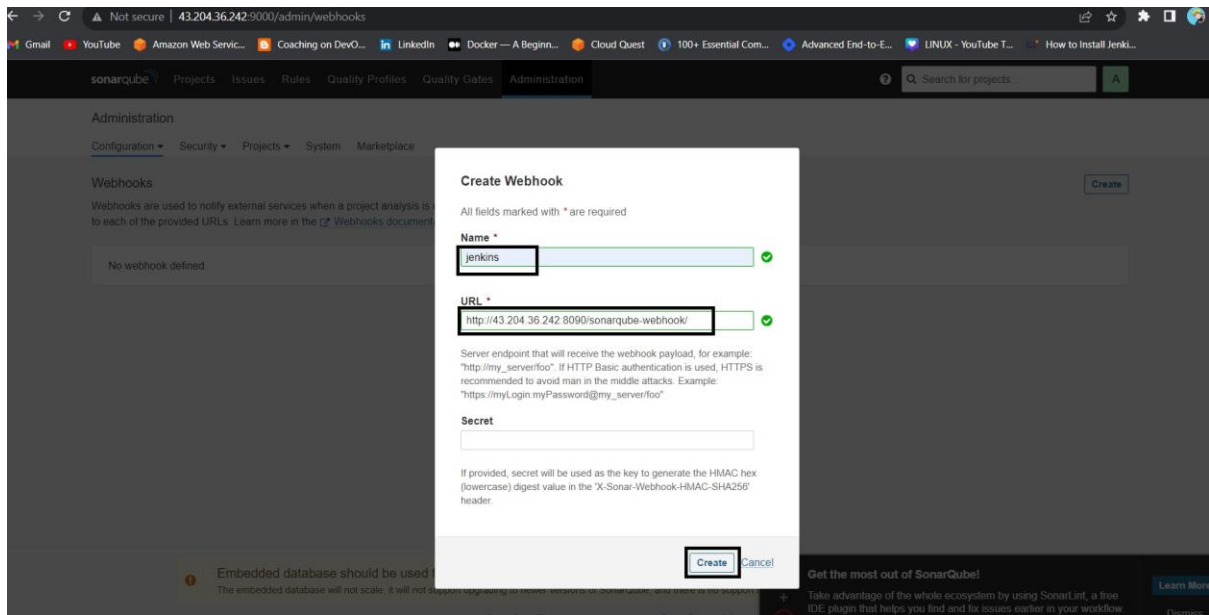


The screenshot shows the SonarQube Webhooks page. The top navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The 'Webhooks' sub-menu is active, and the 'Create' button is highlighted with a red box. The page content includes a description of webhooks and a 'No webhook defined.' message.

## Add details

**#in url section of quality gate**

**<http://jenkins-public-ip:8080>/sonarqube-webhook/**



Let's go to our Pipeline and add the below code Pipeline Script.

```
pipeline{
    agent any

    tools{
        jdk 'jdk17'
    }

    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/Python-System-Monitoring.git'
```

```

    }
}
stage("Sonarqube Analysis "){
    steps{
        withSonarQubeEnv('sonar-server') {
            sh "' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Python-Webapp \
            -Dsonar.projectKey=Python-Webapp '"
        }
    }
}
stage("quality gate"){
    steps {
        script {
            waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
        }
    }
}
stage("TRIVY File scan"){
    steps{
        sh "trivy fs . > trivy-fs_report.txt"
    }
}
stage("OWASP Dependency Check"){
    steps{
        dependencyCheck additionalArguments: '--scan ./ --format XML ', odciInstallation: 'DP-
Check'
        dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
    }
}

```



```
}  
  
}  
  
}
```

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

### Stage View

	Declarative: Tool Install	clean workspace	Checkout From Git	Sonarqube Analysis	quality gate	TRIVY File scan	OWASP Dependency Check
Average stage times: (Average full run time: ~2min 12s)	204ms	375ms	1s	19s	682ms	5s	48s
#1 Sep 14 16:04 No Changes	204ms	375ms	1s	19s	682ms (paused for 6s)	5s	48s












### SonarQube Quality Gate

Python-Webapp **Passed**  
server-side processing: **Success**

 Latest Dependency-Check

To see the report, you can go to Sonarqube Server and go to Projects

☆ Python-Webapp **Passed** Last analysis: 12 minutes ago

 Bugs	 Vulnerabilities	 Hotspots Reviewed	 Code Smells	Coverage	Duplications	Lines
10 	0 	0.0% 	1 	0.0% 	0.0% 	345  HTML, Pyt...

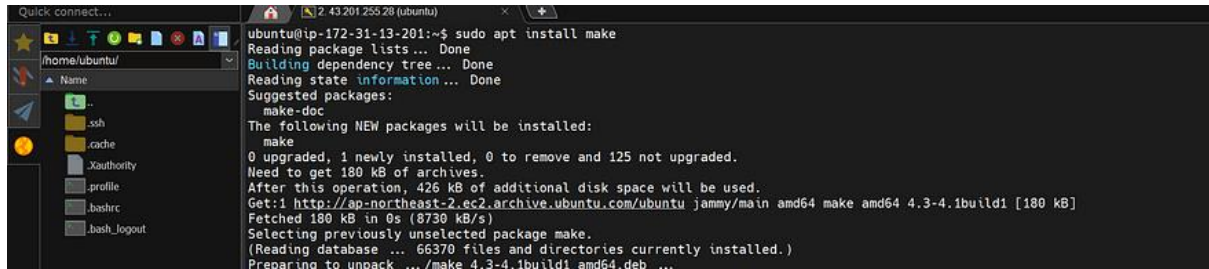
You can see the report has been generated and the status shows as passed. You can see that there are 522 lines. To see a detailed report, you can go to issues.

## Step 6 — we have to install make package

**sudo apt install make**

# to check version install or not

make -v



```
ubuntu@ip-172-31-13-201:~$ sudo apt install make
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 125 not upgraded.
Need to get 180 kB of archives.
After this operation, 426 kB of additional disk space will be used.
Get:1 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 make amd64 4.3-4.1build1 [180 kB]
Fetched 180 kB in 0s (8730 kB/s)
Selecting previously unselected package make.
(Reading database ... 66370 files and directories currently installed.)
Preparing to unpack .../make_4.3-4.1build1_amd64.deb ...
```

## Step 7 — Docker Image Build and Push

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

Docker

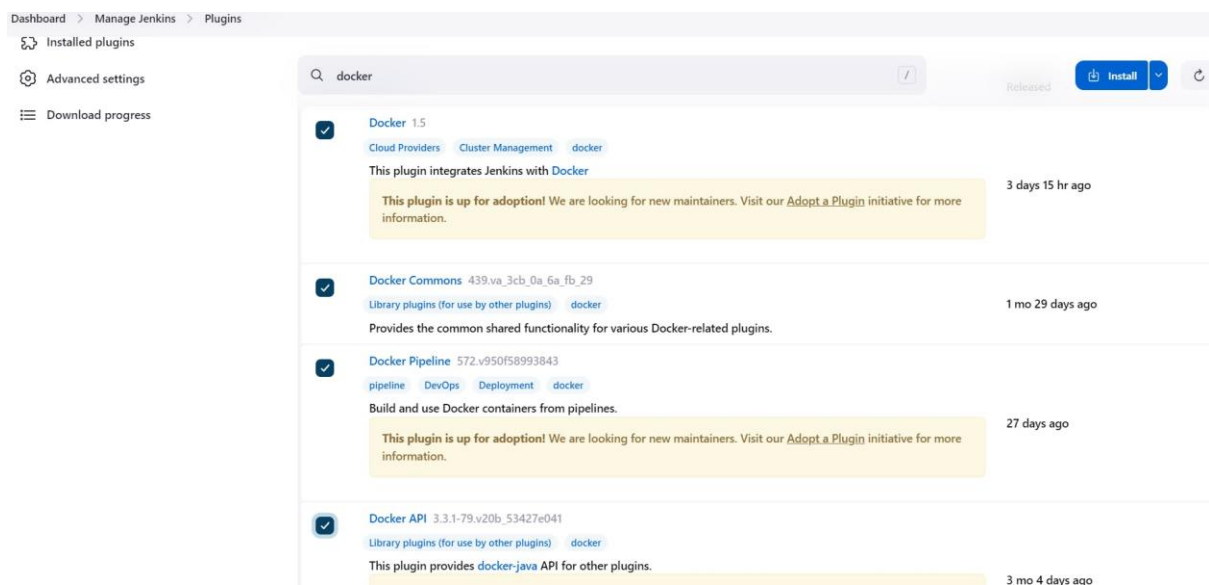
Docker Commons

Docker Pipeline

Docker API

docker-build-step

and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →

Dashboard > Manage Jenkins > Tools

## Docker installations

Add Docker

≡ Docker

Name

docker

☒ Install automatically ?

≡ Download from docker.com

Docker version ?

latest

Add Installer ▼

Add DockerHub Username and Password under Global Credentials

Kind

Username with password

Scope ?

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

Username ?

sevenajay

☐ Treat username as secret ?

Password ?

.....

ID ?

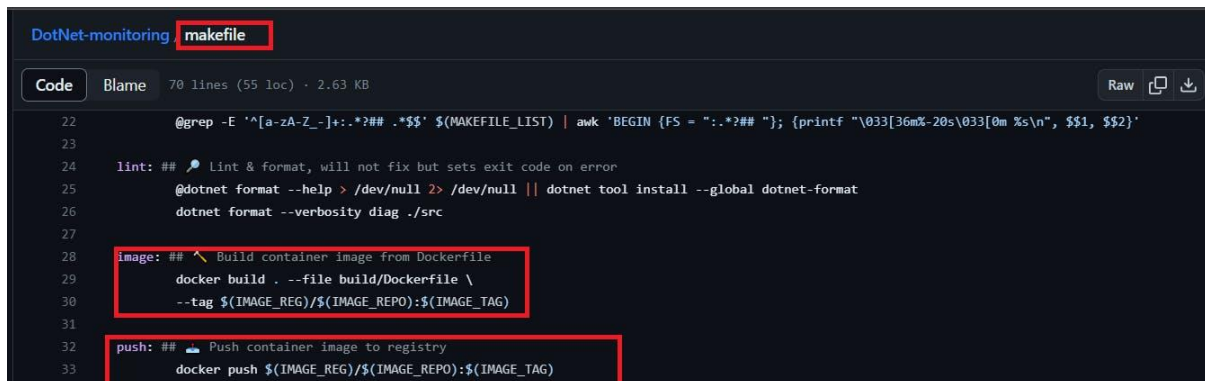
docker

Description ?

docker

Create

In the makefile, we already defined some conditions to build, tag and push images to dockerhub.



```
DotNet-monitoring makefile
Code Blame 70 lines (55 loc) · 2.63 KB
22 @grep -E '^[a-zA-Z_-]+:.*?## .*$$' $(MAKEFILE_LIST) | awk 'BEGIN {FS = ":.*?## "}; {printf "\033[36m%-20s\033[0m %s\n", $$1, $$2}'
23
24 lint: ## 🐞 Lint & format, will not fix but sets exit code on error
25 @dotnet format --help > /dev/null 2> /dev/null || dotnet tool install --global dotnet-format
26 dotnet format --verbosity diag ./src
27
28 image: ## 🏗️ Build container image from Dockerfile
29 docker build . --file build/Dockerfile \
30 --tag $(IMAGE_REG)/$(IMAGE_REPO):$(IMAGE_TAG)
31
32 push: ## 📦 Push container image to registry
33 docker push $(IMAGE_REG)/$(IMAGE_REPO):$(IMAGE_TAG)
```

that's why we are using make image and make a push in the place of docker build -t and docker push

Add this stage to Pipeline Script

```
stage("Docker Build & tag"){
    steps{
        script{
            withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){
                sh "make image"
            }
        }
    }
}

stage("TRIVY"){
    steps{
        sh "trivy image hanvitha/python-system-monitoring:latest > trivy.txt"
    }
}
```

```

stage("Docker Push"){
  steps{
    script{
      withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){
        sh "make push"
      }
    }
  }
}

```

When all stages in docker are successfully created then you will see the result You log in to Dockerhub, and you will see a new image is created

Declarative: Tool Install	clean workspace	Checkout From Git	Sonarqube Analysis	quality gate	TRIVY File scan	OWASP Dependency Check	Docker Build & tag	TRIVY	Docker Push
204ms	375ms	1s	19s	682ms	5s	48s	24s	3s	15s
204ms	375ms	1s	19s	682ms (paused for 6s)	5s	48s	24s	3s	15s

## Step 8 — Deploy the image using Docker

Add this stage to your pipeline syntax

```

stage("Deploy to container"){
  steps{
    sh "docker run -d --name python1 -p 5000:5000 hanvitha/python-system-
monitoring:latest"
  }
}

```

Declarative: Tool Install	clean workspace	Checkout From Git	Sonarqube Analysis	quality gate	TRIVY File scan	OWASP Dependency Check	Docker Build & tag	TRIVY	Docker Push	Deploy to container
204ms	375ms	1s	19s	682ms	5s	48s	24s	3s	15s	1s
204ms	375ms	1s	19s	682ms (paused for 6s)	5s	48s	24s	3s	15s	1s

sgr-076ce119bc21d735d	Custom TCP	TCP	9000	Custom	Q		Delete
sgr-09fc7d687936c1e66	HTTP	TCP	80	Custom	Q	0.0.0.0/X	Delete
-	Custom TCP	TCP	5000	Anywh...	Q	0.0.0.0/X	Delete

Add rule

And you can access your application on Port 5000. This is a Real World Application that has all Functional Tabs.

<public-ip of jenkins:5000>

## Step 9 — Access the Real World Application

Python Demo Info Monitor

### System Information

Hostname	6974d5b0d44c
Boot Time	2023-09-10 08:21:11
OS Platform	Linux
OS Version	#26~22.04.1-Ubuntu SMP Mon Apr 24 01:58:15 UTC 2023
Python Version	3.9.17
Processor & Cores	2 x
System Memory	8GB (44.3% used)
Network Interfaces	<ul style="list-style-type: none"> <li>lo - 127.0.0.1</li> <li>eth0 - 172.17.0.3</li> </ul>

v1.4.2 [Ben Coleman, 2018-2021]

