INFOTRIXS INTERNSHIP TASK – 02

Team Leader – Mrs. Shivani Mane

Team Member – Mast. Tejas Lamkhade

Project Name – Deploying Web Application on EC2: Description: Deploy a web application on an Amazon EC2 instance using an appropriate programming language and web framework. Implementation: Set up an EC2 instance, install necessary software, configure security groups, and deploy the web application code. Use Elastic IP for a static IP address and Elastic Load Balancer for load balancing if required.

Full Stack CICD Virtual Browser Project

Step-by-Step Implementation:-

Step-1 Create EC2 Instance :-

1. **Sign in to AWS Console:** Log in to your AWS Management Console.
2. **Navigate to EC2 Dashboard:** Go to the EC2 Dashboard by selecting “Services” in the top menu and then choosing “EC2” under the Compute section.
3. **Launch Instance:** Click on the “Launch Instance” button to start the instance creation process.

4. **Choose an Amazon Machine Image (AMI):** Select an appropriate AMI for your instance. For example, you can choose Ubuntu image.
5. **Choose an Instance Type:** In the "Choose Instance Type" step, select `t2.large` as your instance type. Proceed by clicking "Next: Configure Instance Details."
6. **Configure Instance Details:**

- For "Number of Instances," set it to 1 (unless you need multiple instances).
- Configure additional settings like network, subnets, IAM role, etc., if necessary.
- For "Storage," click "Add New Volume" and set the size to 8GB (or modify the existing storage to 16GB).
- Click "Next: Add Tags" when you're done.

7. Add Tags (Optional): Add any desired tags to your instance. This step is optional, but it helps in organizing instances.

8. Configure Security Group:

- Choose an existing security group or create a new one.
- Ensure the security group has the necessary inbound/outbound rules to allow access as required.

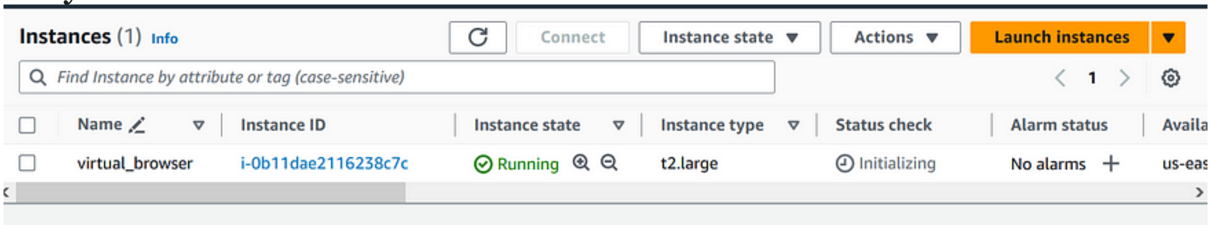
9. Review and Launch: Review the configuration details. Ensure everything is set as desired.

10. Select Key Pair:

- Select "Choose an existing key pair" and choose the key pair from the dropdown.
- Acknowledge that you have access to the selected private key file.
- Click "Launch Instances" to create the instance.

11. Access the EC2 Instance: Once the instance is launched, you can access it using the key pair and the instance's public IP or DNS.

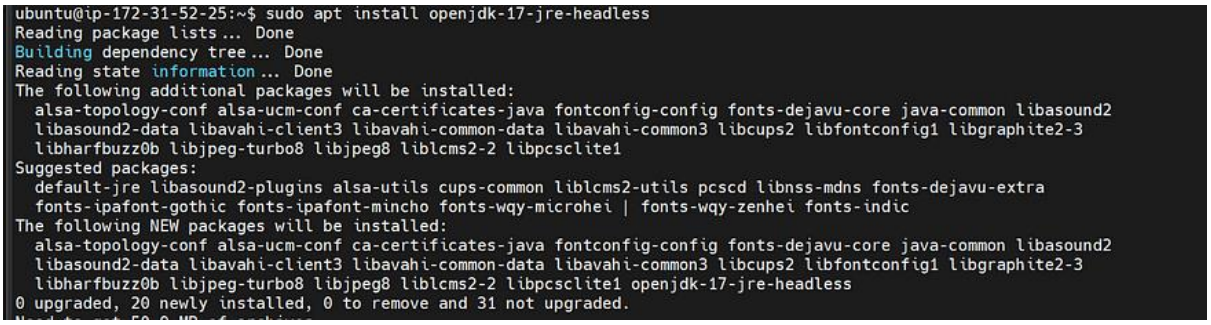
Ensure you have necessary permissions and follow best practices while configuring security groups and key pairs to maintain security for your EC2 instance.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availa
☐	virtual_browser	i-0b11dae2116238c7c	Running	t2.large	Initializing	No alarms	us-eas

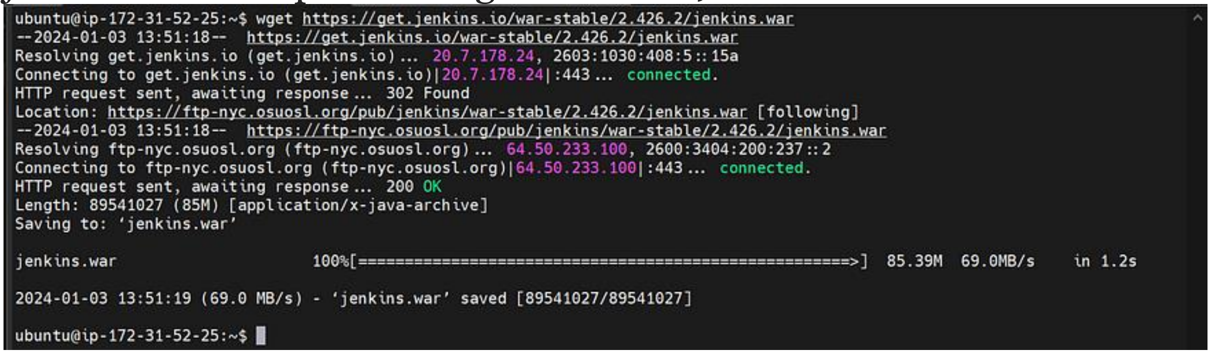
Step 2 Connect to Instance and Install Required Packages:-

2 A:- we are install jdk-17 for this demo.



```
ubuntu@ip-172-31-52-25:~$ sudo apt install openjdk-17-jre-headless
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf ca-certificates-java fontconfig-config fonts-dejavu-core java-common libasound2
  libasound2-data libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libfontconfig1 libgraphite2-3
  libharfbuzz0b libjpeg-turbo8 libjpeg8 liblcms2-2 libpcsclite1
Suggested packages:
  default-jre libasound2-plugins alsa-utils cups-common liblcms2-utils pcscd libnss-mdns fonts-dejavu-extra
  fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  alsa-topology-conf alsa-ucm-conf ca-certificates-java fontconfig-config fonts-dejavu-core java-common libasound2
  libasound2-data libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libfontconfig1 libgraphite2-3
  libharfbuzz0b libjpeg-turbo8 libjpeg8 liblcms2-2 libpcsclite1 openjdk-17-jre-headless
0 upgraded, 20 newly installed, 0 to remove and 31 not upgraded.
```

(Jenkins is run on 8080 by default but in this demo we are run an jenkins on 8081 port through WAR file.)

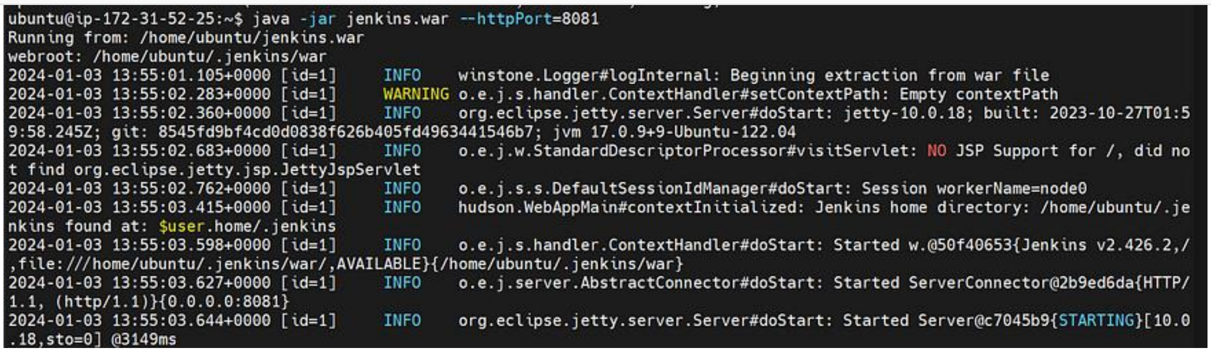


```
ubuntu@ip-172-31-52-25:~$ wget https://get.jenkins.io/war-stable/2.426.2/jenkins.war
--2024-01-03 13:51:18--  https://get.jenkins.io/war-stable/2.426.2/jenkins.war
Resolving get.jenkins.io (get.jenkins.io)... 20.7.178.24, 2603:1030:408:5::15a
Connecting to get.jenkins.io (get.jenkins.io)|20.7.178.24|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://ftp-nyc.osuosl.org/pub/jenkins/war-stable/2.426.2/jenkins.war [following]
--2024-01-03 13:51:18--  https://ftp-nyc.osuosl.org/pub/jenkins/war-stable/2.426.2/jenkins.war
Resolving ftp-nyc.osuosl.org (ftp-nyc.osuosl.org)... 64.50.233.100, 2600:3404:200:237::2
Connecting to ftp-nyc.osuosl.org (ftp-nyc.osuosl.org)|64.50.233.100|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 89541027 (85M) [application/x-java-archive]
Saving to: 'jenkins.war'

jenkins.war                100%[=====================================================>]  85.39M  69.0MB/s    in 1.2s

2024-01-03 13:51:19 (69.0 MB/s) - 'jenkins.war' saved [89541027/89541027]

ubuntu@ip-172-31-52-25:~$ 
```

WAR file link:- https://www.jenkins.io/download/

after that run an command

```
java -jar jenkins.war --httpPort=8081
```



```
ubuntu@ip-172-31-52-25:~$ java -jar jenkins.war --httpPort=8081
Running from: /home/ubuntu/jenkins.war
webroot: /home/ubuntu/.jenkins/war
2024-01-03 13:55:01.105+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from war file
2024-01-03 13:55:02.283+0000 [id=1]     WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2024-01-03 13:55:02.360+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-10.0.18; built: 2023-10-27T01:5
9:58.245Z; git: 8545fd9bf4cd0d0838f626b405fd4963441546b7; jvm 17.0.9+9-Ubuntu-122.04
2024-01-03 13:55:02.683+0000 [id=1]     INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did no
t find org.eclipse.jetty.jsp.JettyJspServlet
2024-01-03 13:55:02.762+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: Session workerName=node0
2024-01-03 13:55:03.415+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home directory: /home/ubuntu/.je
nkins found at: $user.home/.jenkins
2024-01-03 13:55:03.598+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started w.@50f40653{Jenkins v2.426.2,/
,file:///home/ubuntu/.jenkins/war/,AVAILABLE}{/home/ubuntu/.jenkins/war}
2024-01-03 13:55:03.627+0000 [id=1]     INFO    o.e.j.server.AbstractConnector#doStart: Started ServerConnector@2b9ed6da{HTTP/
1.1, (http/1.1)}{0.0.0.0:8081}
2024-01-03 13:55:03.644+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started Server@c7045b9{STARTING}[10.0
.18,sto=0] @3149ms
```

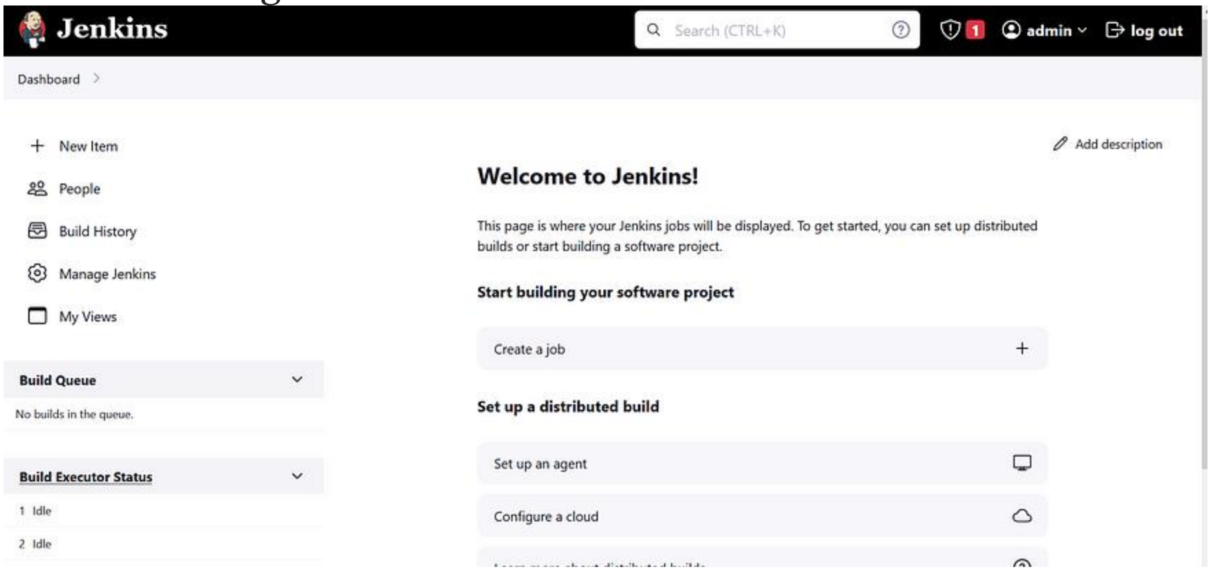
(Please continue that session in mobaxtream and duplicate an session in new tab and added next commands in that terminal)

Now, grab your Public IP Address

```
<EC2 Public IP Address:8081>
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

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

Jenkins Getting Started Screen.



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

Set up a distributed build

No builds in the queue.

1 Idle

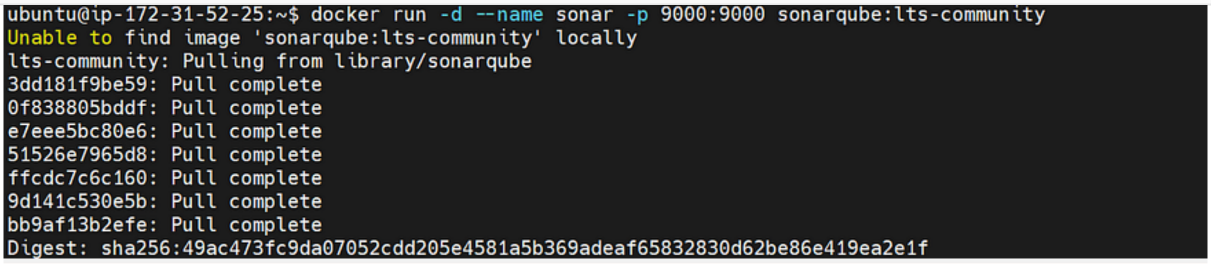
2 Idle

2 B:- Install Docker

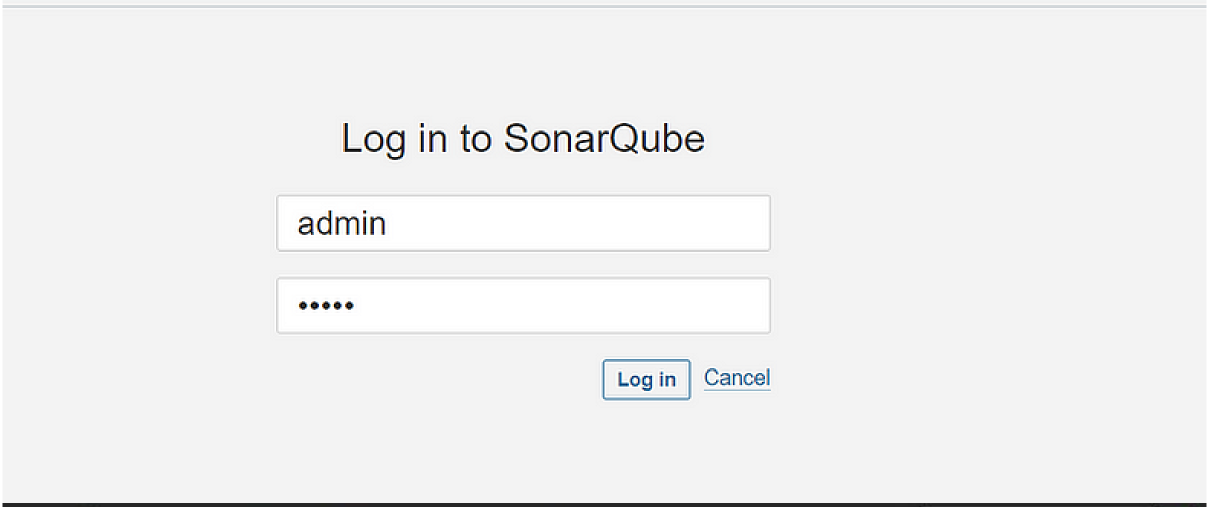
```
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER
sudo chmod 777 /var/run/docker.sock
sudo docker ps
```

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

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



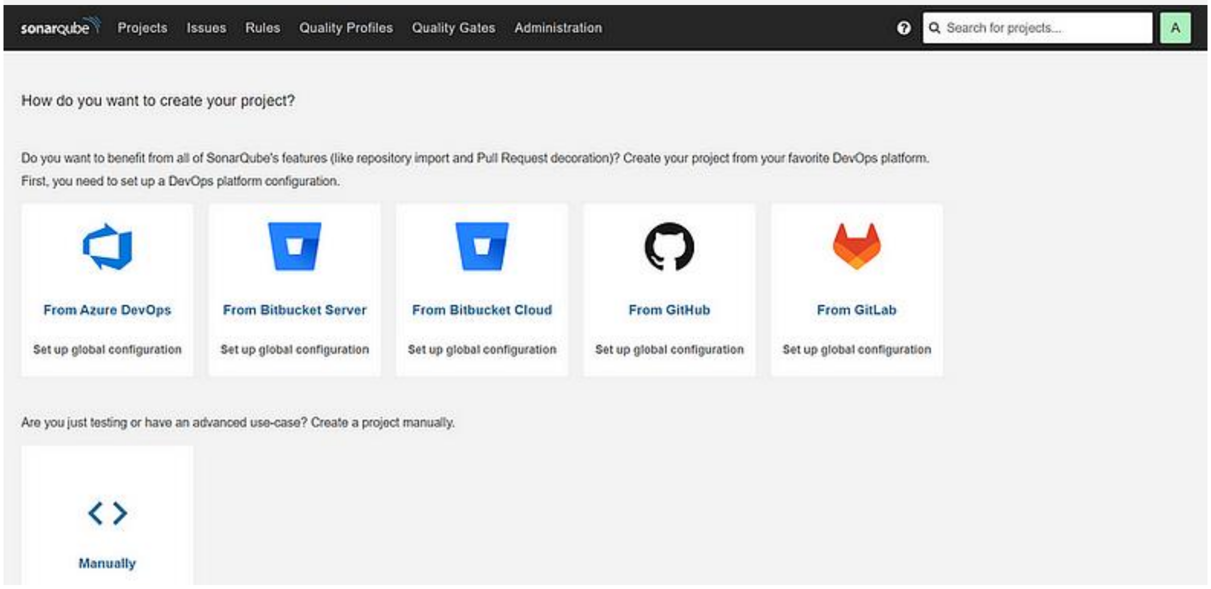
```
ubuntu@ip-172-31-52-25:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
3dd181f9be59: Pull complete
0f838805bddf: Pull complete
e7eee5bc80e6: Pull complete
51526e7965d8: Pull complete
ffcdc7c6c160: Pull complete
9d141c530e5b: Pull complete
bb9af13b2efe: Pull complete
Digest: sha256:49ac473fc9da07052cdd205e4581a5b369adeaf65832830d62be86e419ea2e1f
```



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.

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

3A — Install Plugin

Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Install OWASP ((Install without restart)

2 → SonarQube Scanner (Install without restart)

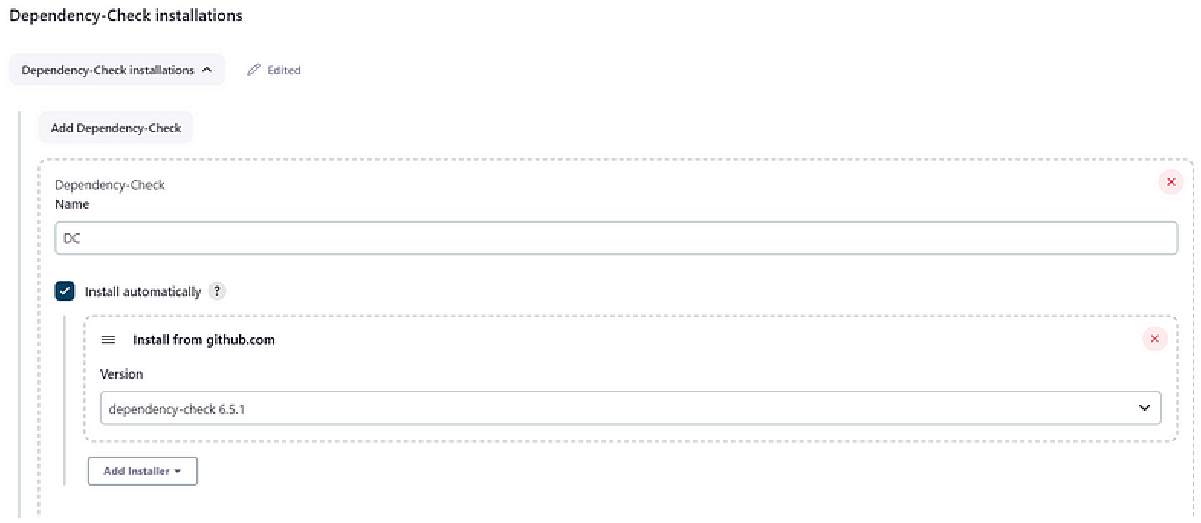
3 → 1 → Eclipse Temurin Installer (Install without restart)

4 → docker (docker, docker-pipleine , docker-build-step, cloudbees docker)

Step 4 Configure an Tools in jenkins:-

Goto Dashboard → Manage Jenkins → Tools →

and install owasp, docker and sonar-scanner tools versions.


Dependency-Check installations

Dependency-Check
Name

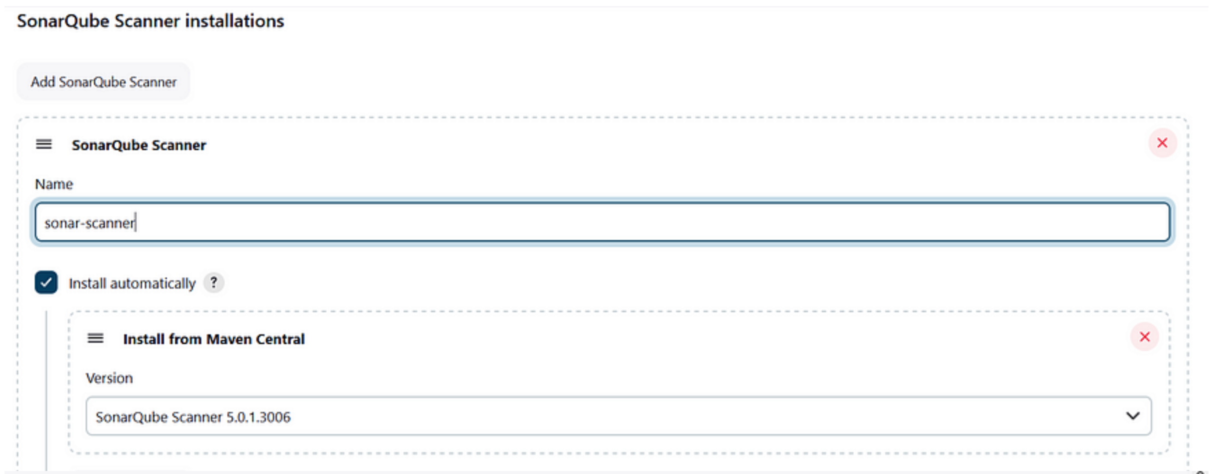
DC

Install automatically ✓

Install from github.com

Version

dependency-check 6.5.1


SonarQube Scanner installations

Add SonarQube Scanner

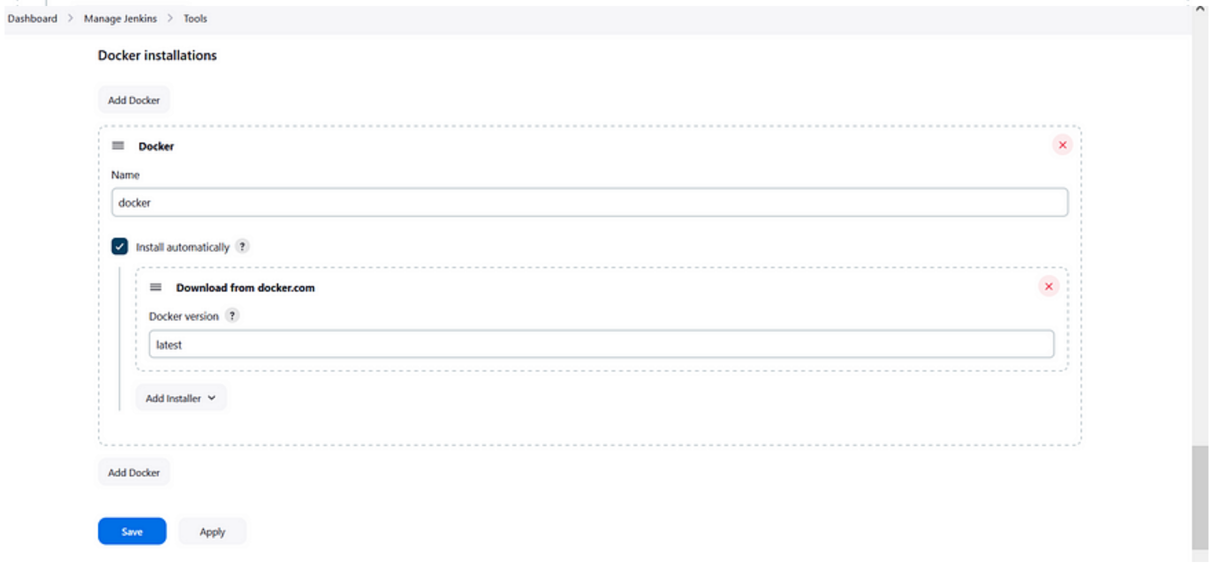
SonarQube Scanner

Name: sonar-scanner

Install automatically ✓

Install from Maven Central

Version: SonarQube Scanner 5.0.1.3006


Dashboard > Manage Jenkins > Tools

Docker installations

Add Docker

Docker

Name: docker

Install automatically ✓

Download from docker.com

Docker version: latest

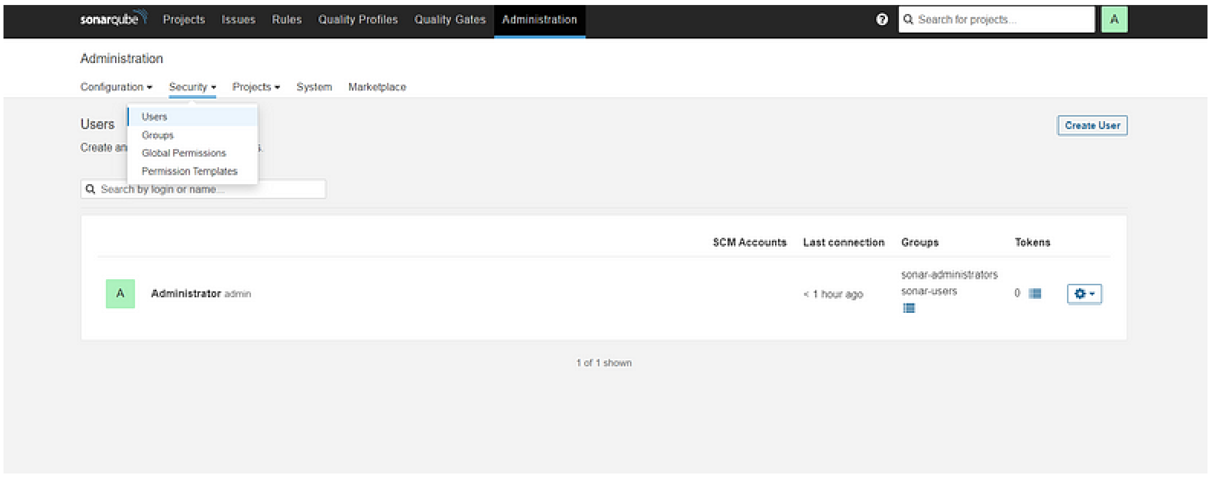
Add Installer

Add Docker

Save Apply

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

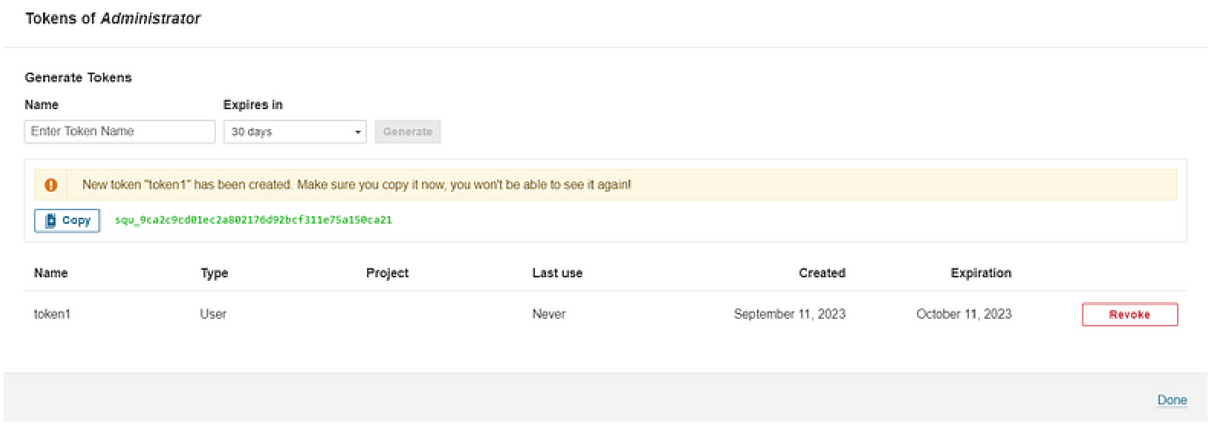


Users

	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	Expires in	
Enter Token Name	30 days	Generate

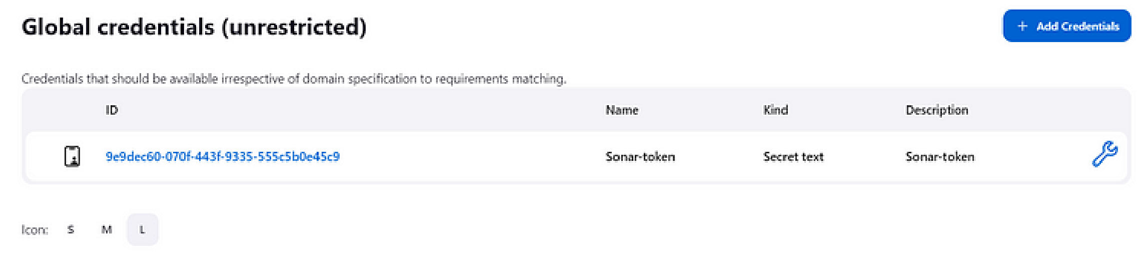
New token "token1" has been created. Make sure you copy it now, you won't be able to see it again!

Copy `squ_9ca2c9cd01ec2a802176d92bcf311e75a150ca21`

Name	Type	Project	Last use	Created	Expiration	
token1	User		Never	September 11, 2023	October 11, 2023	Revoke

Copy this Token

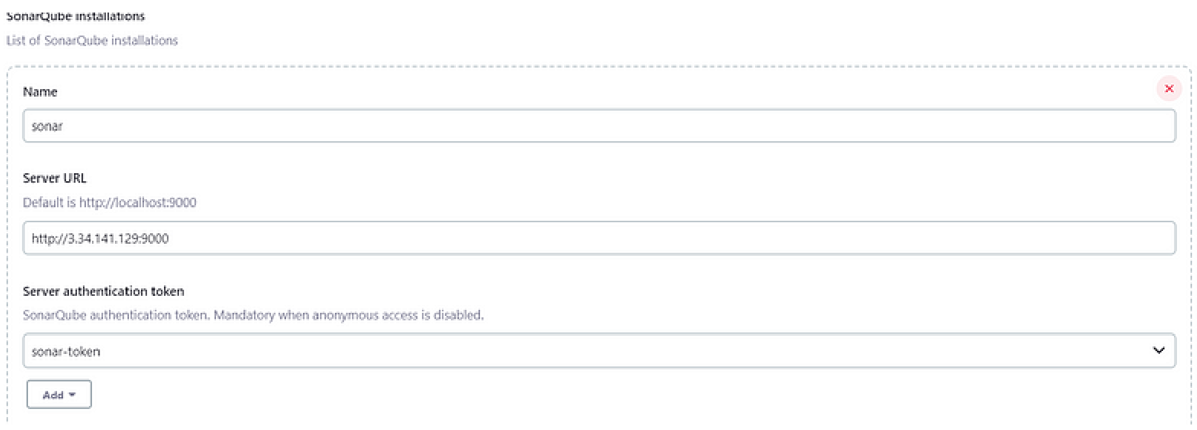
Goto Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this


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


SonarQube installations

List of SonarQube installations

Name

sonar

Server URL

Default is http://localhost:9000

http://3.34.141.129:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

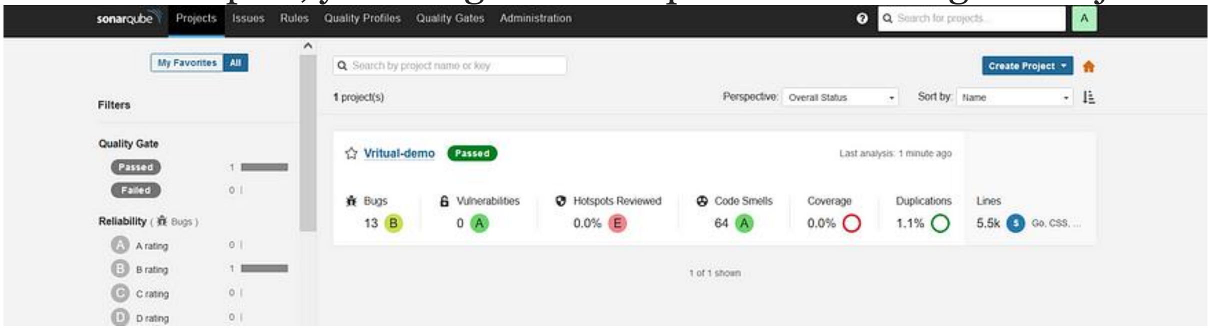
sonar-token

Click on Apply and Save.

Lets goto our Pipeline and add Stages in our Pipeline Script.

```
pipeline {
    agent any

      environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/Tejas-
24ytj/Vitual-Browser.git'
            }
        }

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

        stage(" Sonarqube Analysis "){
            steps{
                 withSonarQubeEnv('sonar') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Virtual-demo \
                    -Dsonar.projectKey=Virtual-demo '''

                 }
            }
        }
      }
  }
```

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



1 project(s)

Perspective: Overall Status   Sort by: Name

Quality Gate
Passed 1
Failed 0

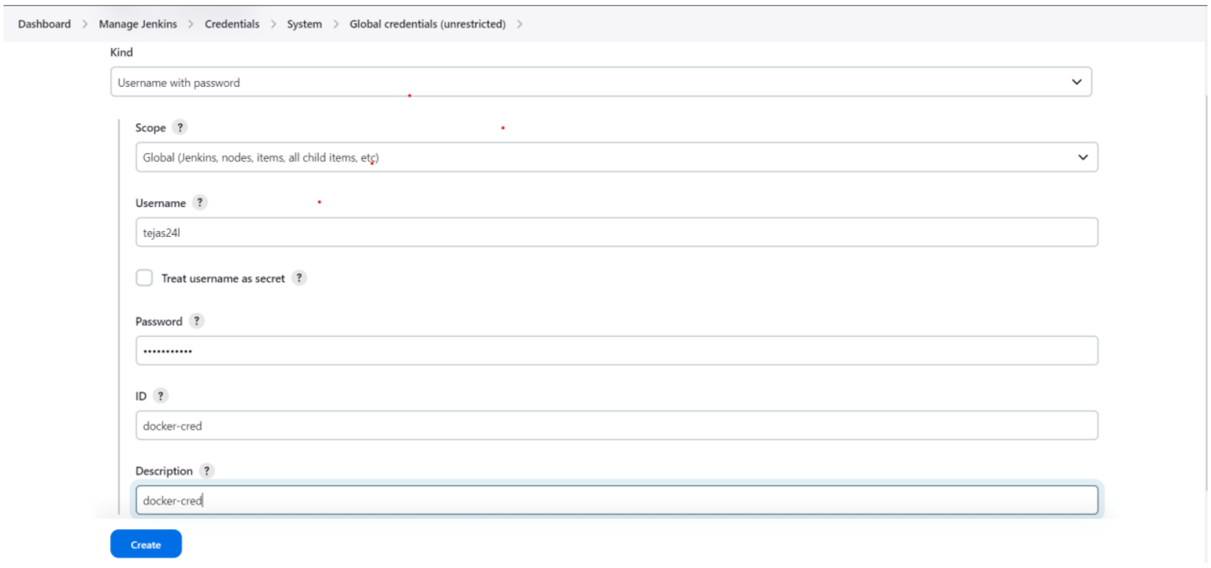
Reliability (Bugs)
A rating 0
B rating 1
C rating 0
D rating 0

Vritual-demo Passed   Last analysis: 1 minute ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
13 B	0 A	0.0% E	64 A	0.0%	1.1%	5.5k S Go, CSS, ...

1 of 1 shown

Add DockerHub Username and Password under Global Credentials.


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

Kind: Username with password

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

Username: tejas24l

Treat username as secret

Password: ••••••••••••

ID: docker-cred

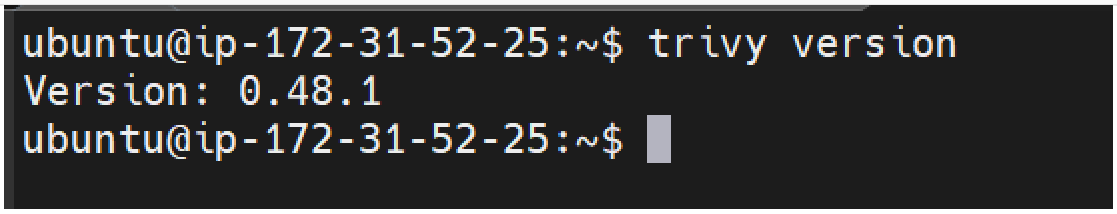
Description: docker-cred

added this stage to pipeline

```
stage("Docker Build & Tag"){
            steps{
                script{
                    withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker') {


dir('/home/ubuntu/.jenkins/workspace/virtual_browser/.docker/firefox') {
      sh "docker build -t tejas24l/vb:latest ."
         }
                     }
                }
            }
        }
```

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
```

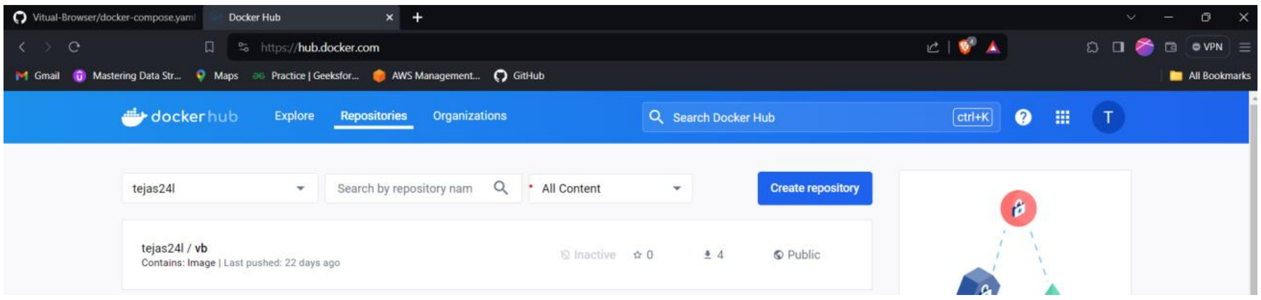


```
ubuntu@ip-172-31-52-25:~$ trivy version
Version: 0.48.1
ubuntu@ip-172-31-52-25:~$
```

Add this stage to Pipeline Script

```
stage("TRIVY"){
            steps{
                sh "trivy image tejas241/vb:latest > trivy.txt"
            }
        }
stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker'){
                               sh "docker push tejas241/vb:latest "
                         }
                }
            }
        }
```

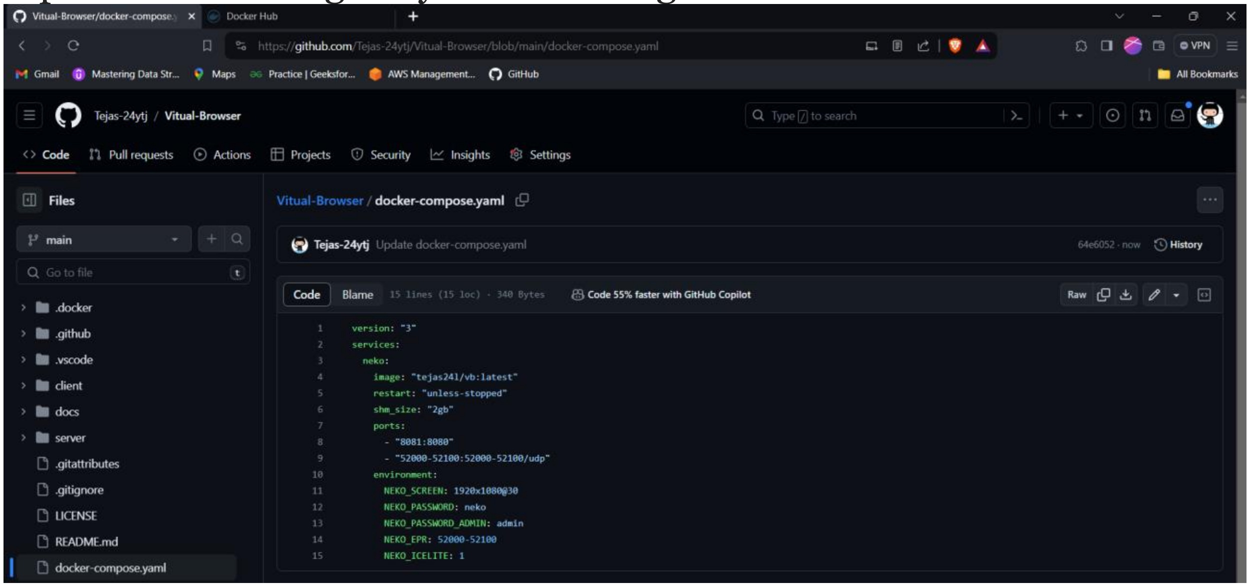
you will see the image pushed to dockerhub



step 6 Install docker-compose to ec2:-

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

After that we can edit an docker-compose file in github and we replace an old image to your new image.

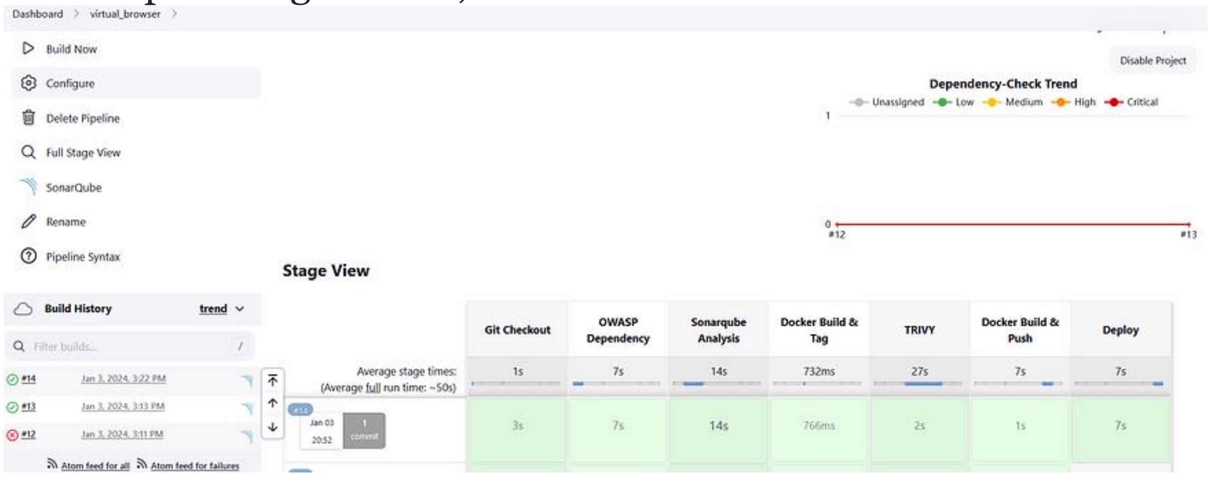


```
version: "3"
services:
  neko:
    image: "tejas24l/vb:latest"
    restart: "unless-stopped"
    shm_size: "2gb"
    ports:
      - "8081:8080"
      - "52000-52100:52000-52100/udp"
    environment:
      NEKO_SCREEN: 1920x1080@30
      NEKO_PASSWORD: neko
      NEKO_PASSWORD_ADMIN: admin
      NEKO_EPR: 52000-52100
      NEKO_ICELITE: 1
```

added next stage to pipeline

```
  stage("Deploy"){
            steps{
                sh "docker-compose up -d"
                }
            }
```

the complete stage view is,

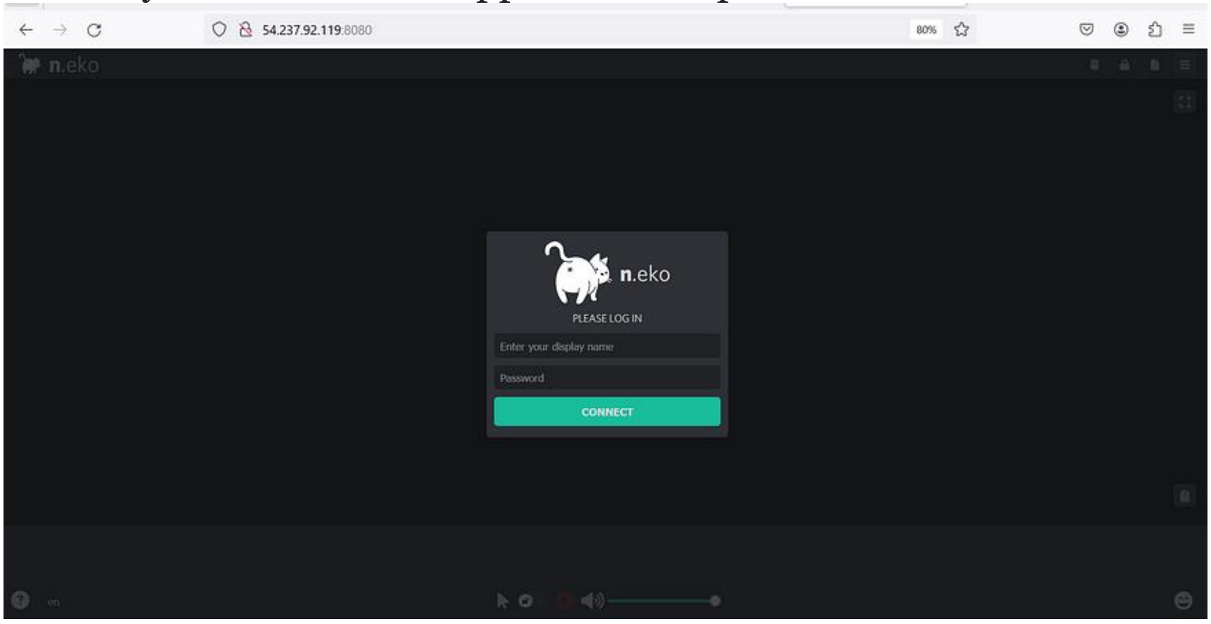


Build	Date
#14	Jan 3, 2024, 3:22 PM
#13	Jan 3, 2024, 3:13 PM
#12	Jan 3, 2024, 3:11 PM

Stage View

	Git Checkout	OWASP Dependency	Sonarqube Analysis	Docker Build & Tag	TRIVY	Docker Build & Push	Deploy
Average stage times: (Average full run time: ~50s)	1s	7s	14s	732ms	27s	7s	7s
#14 Jan 03 20:52 1 commit	3s	7s	14s	766ms	2s	1s	7s

so finally we can acces an application on port 8080.



The complete Pipeline for this project

```
pipeline {
    agent any

      environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/Tejas-
24ytj/Vitual-Browser.git'
            }
        }

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

        stage(" Sonarqube Analysis "){
            steps{
                 withSonarQubeEnv('sonar') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Virtual-demo \
                    -Dsonar.projectKey=Virtual-demo '''

                 }
            }
        }

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


dir('/home/ubuntu/.jenkins/workspace/virtual_browser/.docker/firefox') {
      sh "docker build -t tejas241/vb:latest ."
       }
                        }
                }
            }
        }

        stage("TRIVY"){
            steps{
                sh "trivy image tejas241/vb:latest > trivy.txt"
            }
        }
        stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'docker'){
                          sh "docker push tejas241/vb:latest "
                    }
                }
            }
        }


        stage("Deploy"){
            steps{
                sh "docker-compose up -d"
            }
        }
    }
}
```