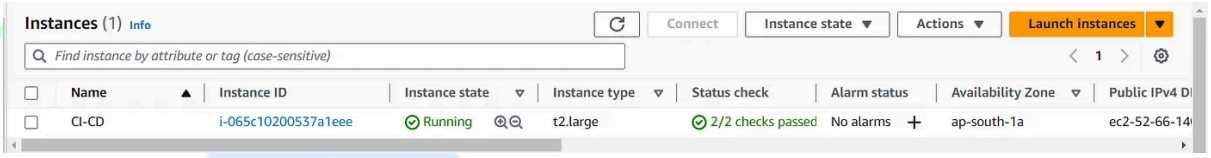
DEPLOY THE REACTJS APP IN KUBERNETES WITH DEVSECOPS CICD PIPELINE

STEP1:Launch an Ubuntu(22.04) T2 Large Instance 30 GB storage



Step 2 —

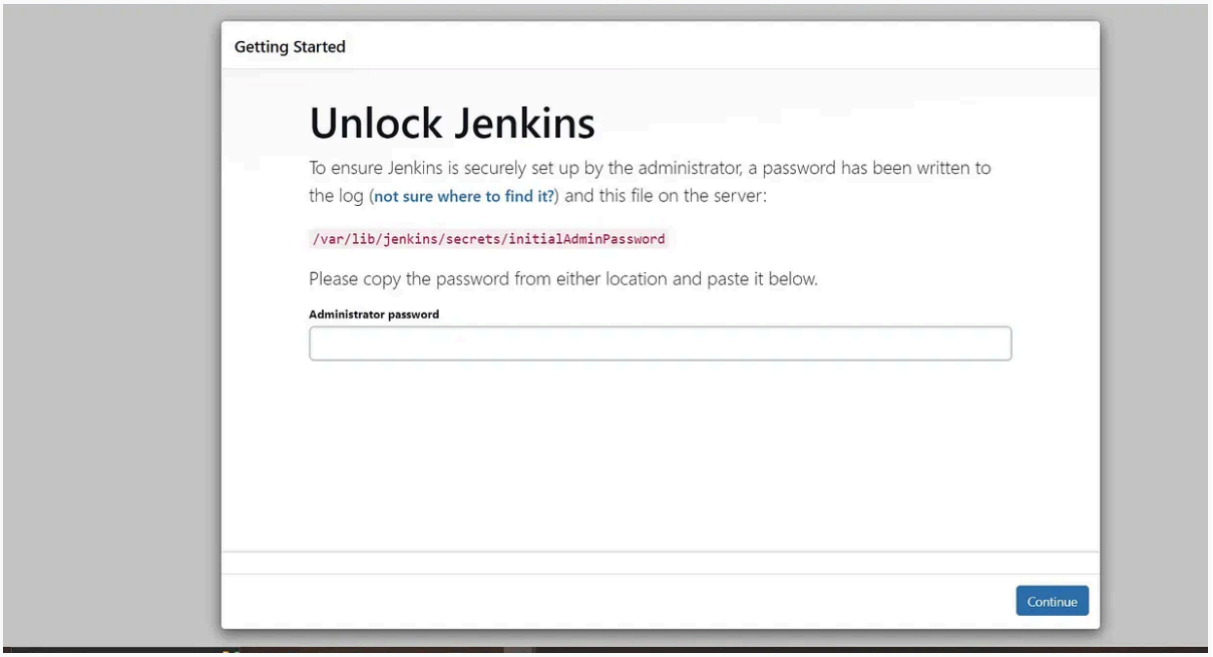
2A - Install Jenkins

https://github.com/AWS-AZURE-Bootcamp5/tools-setup-project1/blob/main/jenkins.sh

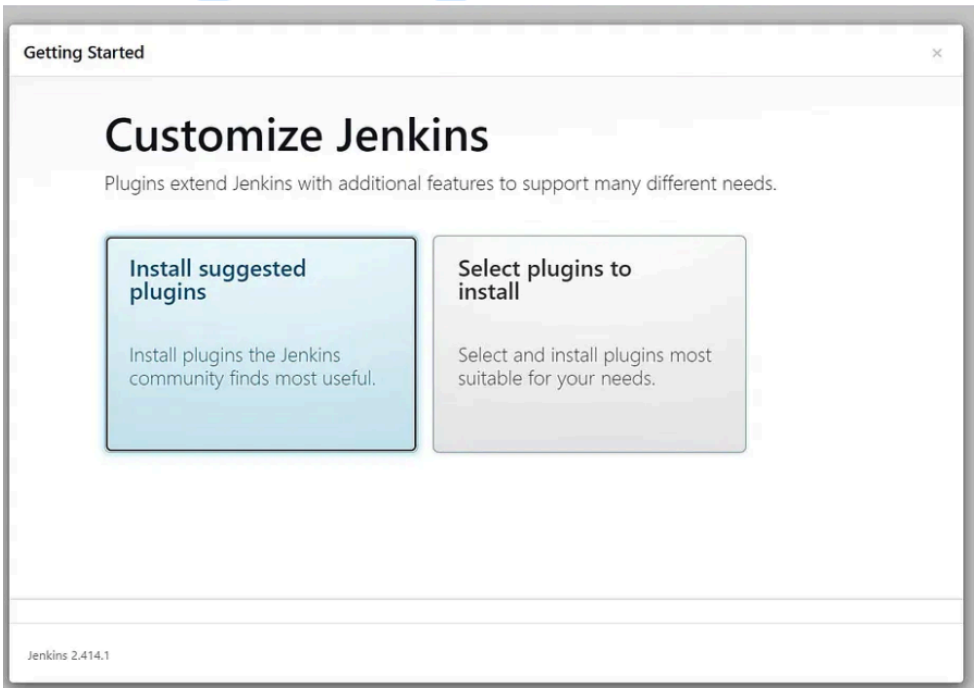
Go to AWS EC2 Security Group and open Inbound Port 8080 and in server get the password

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

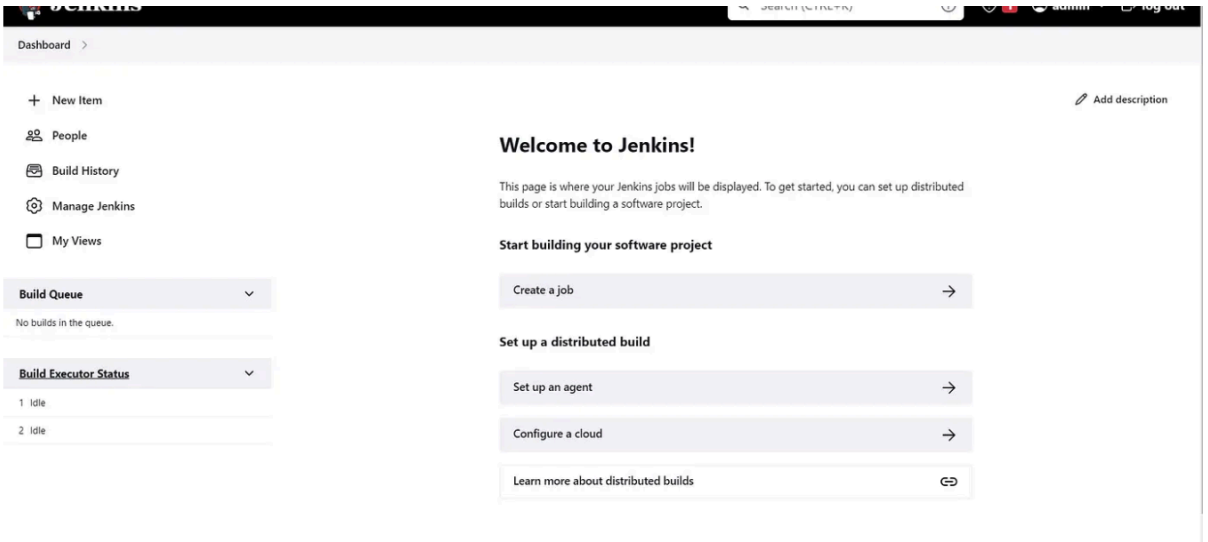
—



Unlock Jenkins



Create a user click on save and continue.



2B — Install Docker and setup Sonarqube

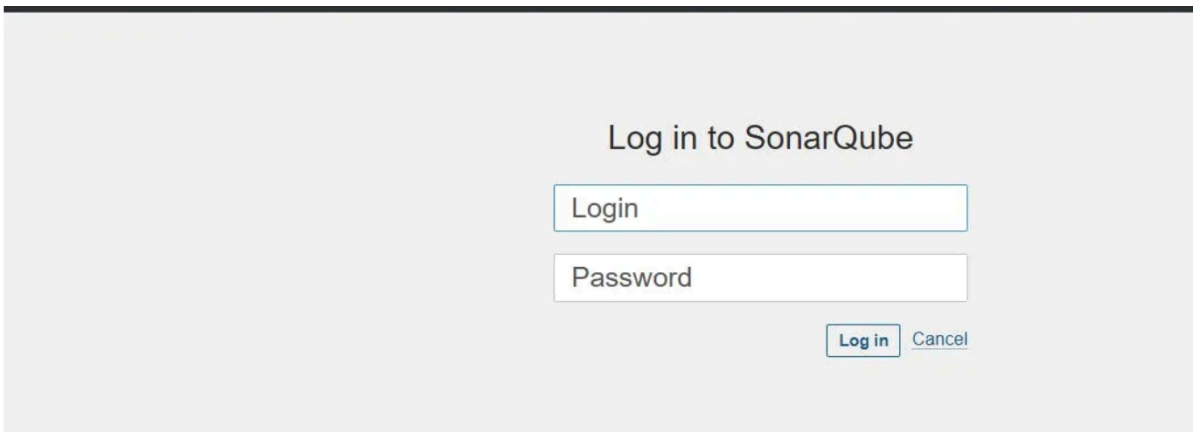
https://github.com/AWS-AZURE-Bootcamp5/tools-setup-project1/blob/main/docker.sh

After the docker installation, we created 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
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436af7f752fdb04993092d0ca3065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED         STATUS         PORTS                                       NAMES
4b60c96bf9ad   sonarqube:lts-community   "/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:~$
```

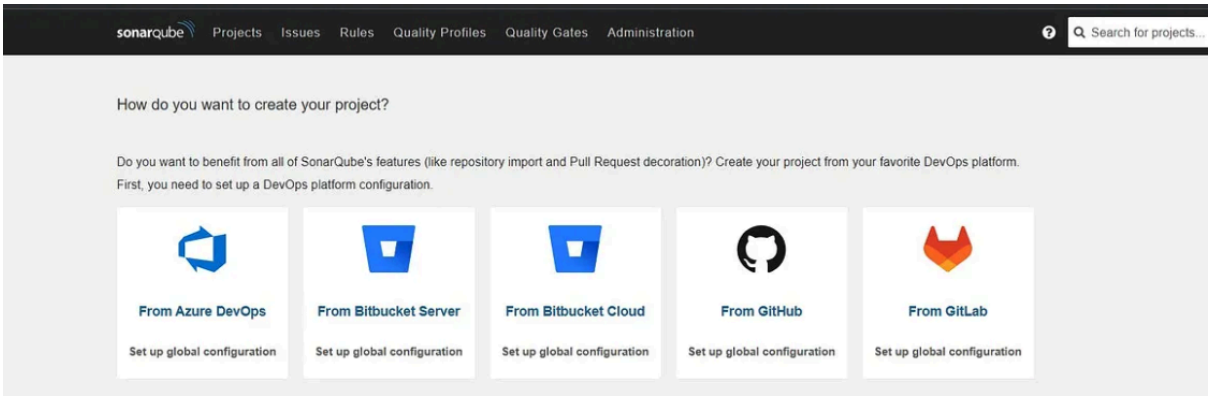
Now our Sonarqube is up and running



Enter username and password, click on login and change password

username admin / password admin

Update New password, This is Sonar Dashboard.



2C — Install Trivy [IAC]

https://github.com/AWS-AZURE-Bootcamp5/tools-setup-project1/blob/main/trivy.sh

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

3A — Install Plugin

Go to Manage Jenkins →Plugins → Available Plugins →

Install below plugins

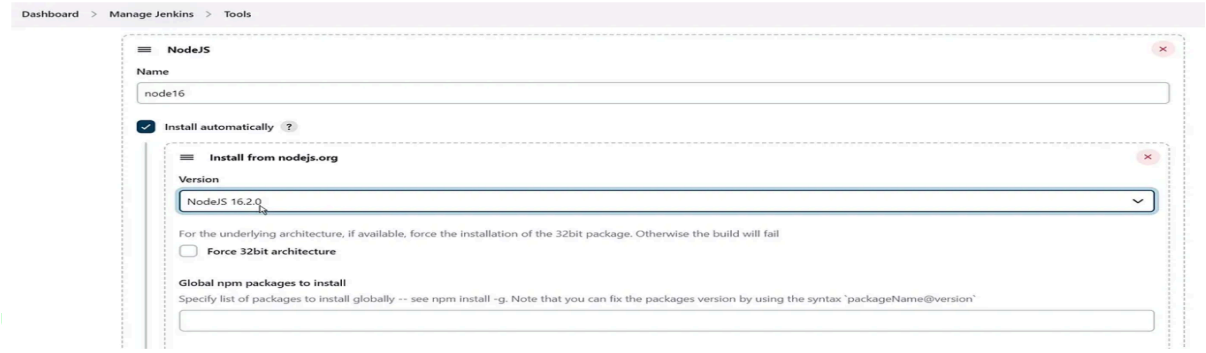
1 → Eclipse Temurin Installer (Install without restart)

2 → Sonarqube Scanner (Install without restart)

3 → NodeJS Plugin (Install Without restart)

3B — Configure Java and Nodejs in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save



3C — Create a Job

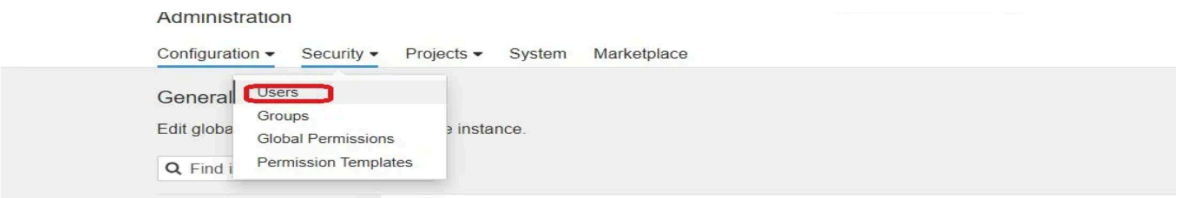
create a job as Devsecops_demo Name, select pipeline and click ok.

Step 4 — Configure Sonar Server in Manage Jenkins

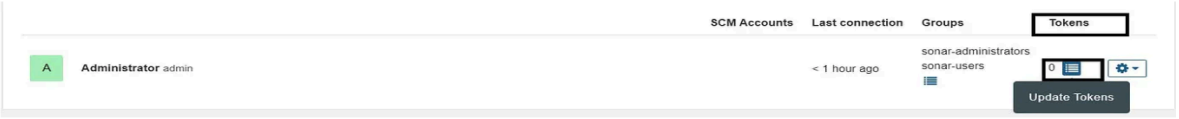
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000.

Goto your Sonarqube Server.

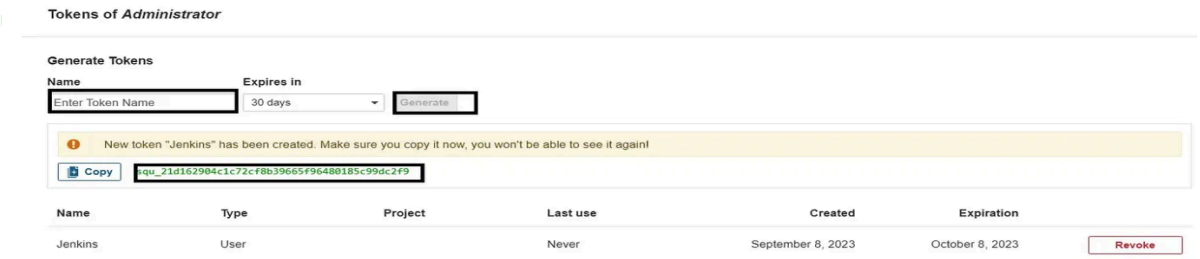
Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

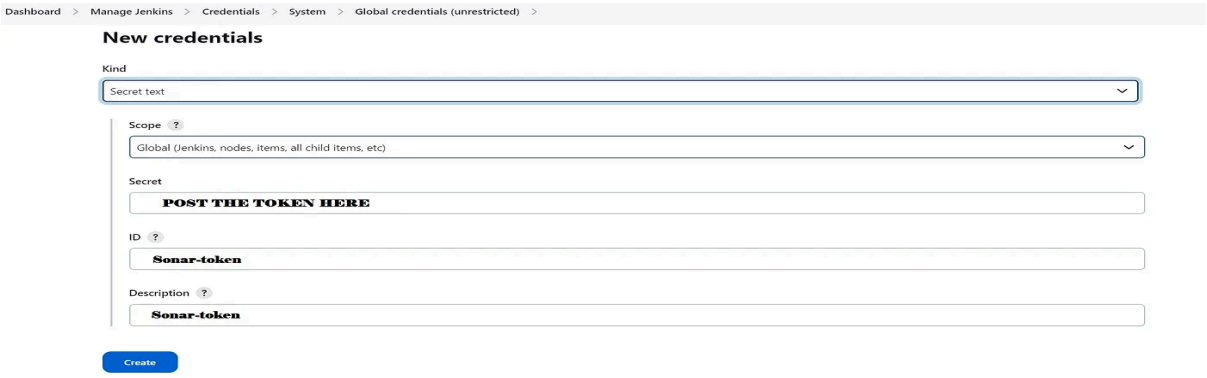


Create a token with a name and generate



copy Token

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



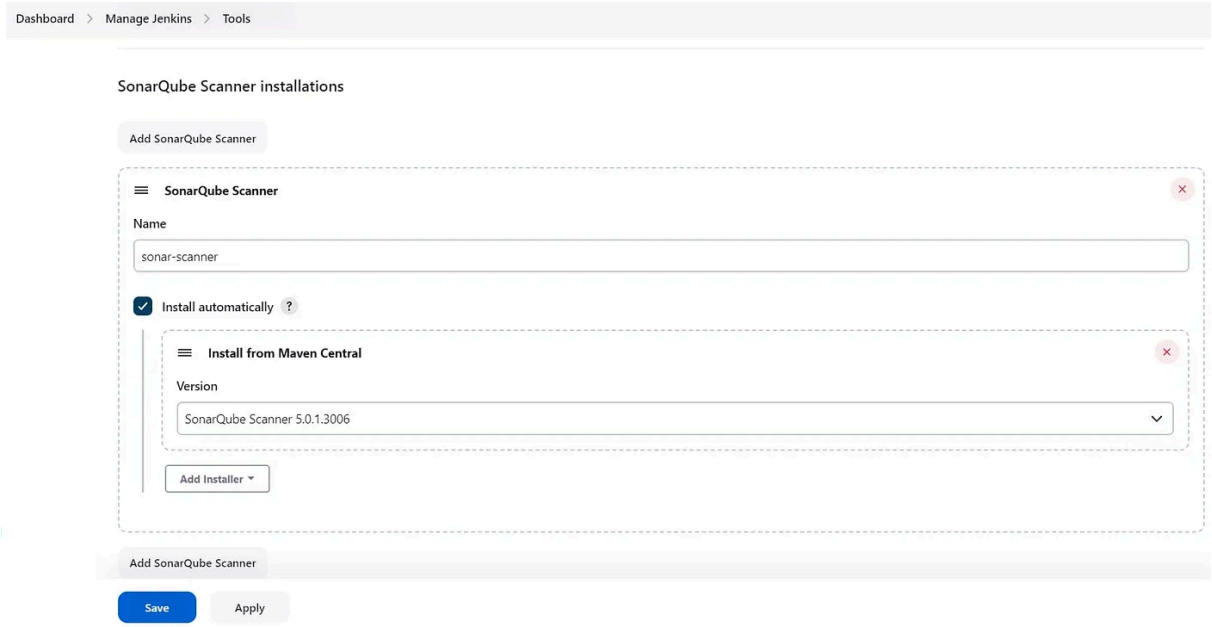
Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

Click on Apply and Save

The Configure System option is used in Jenkins to configure different server

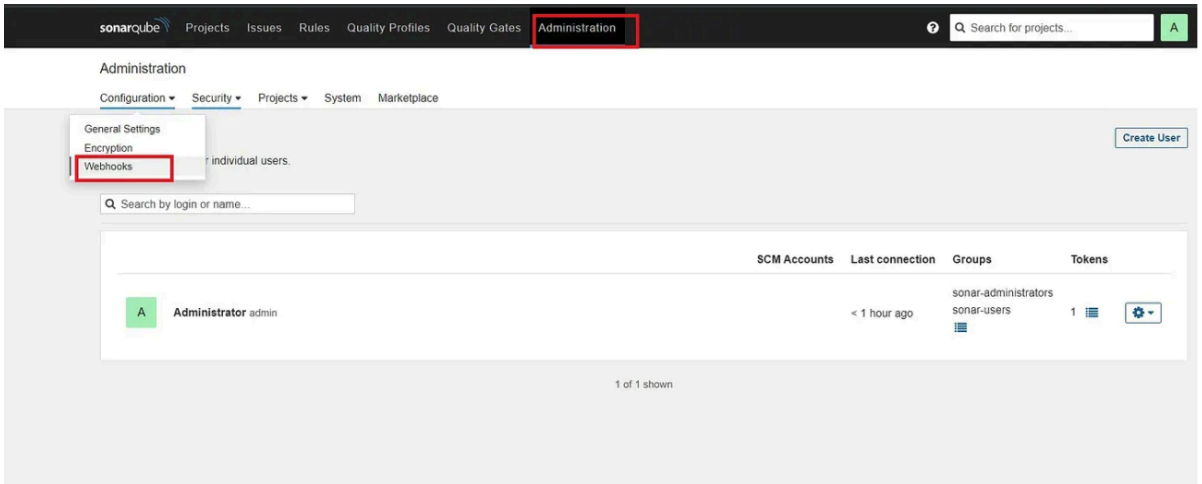
Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

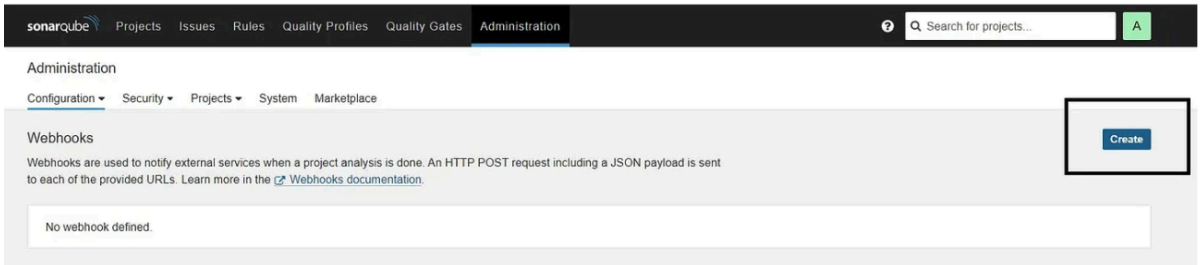


In the Sonarqube Dashboard add a quality gate also

Administration → Configuration →Webhooks

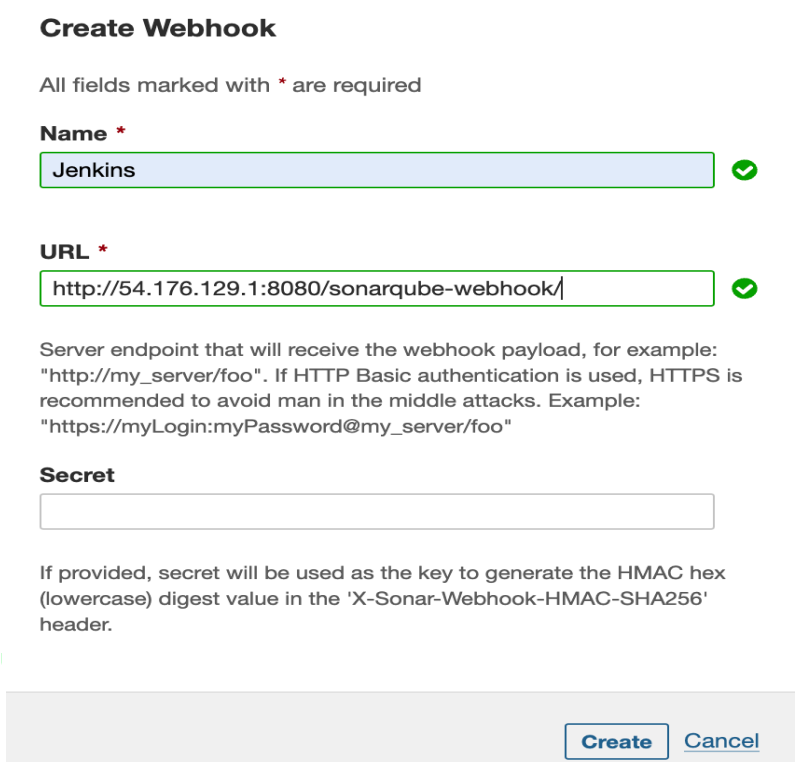


Click on Create



Add details

```
#in url section of quality gate
<http://jenkins-public-ip:8080>/sonarqube-webhook/
```



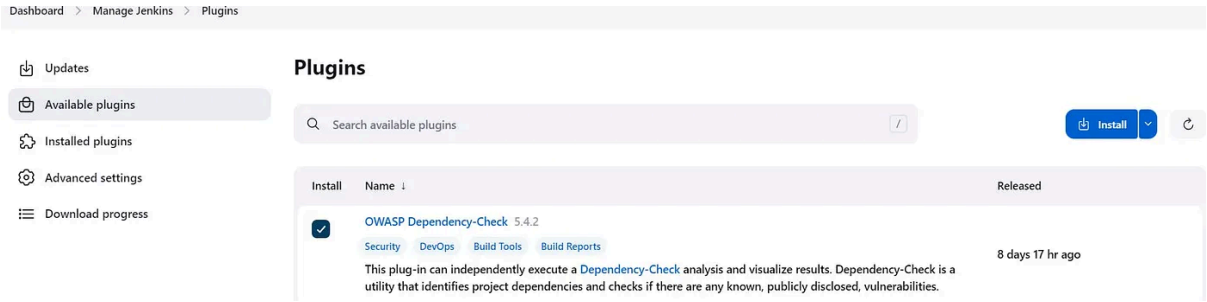
Let's go to our Pipeline and add the script in our Pipeline Script.

https://github.com/AWS-AZURE-Bootcamp5/Devsecops-Project1/blob/main/Jenkinsfile1

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

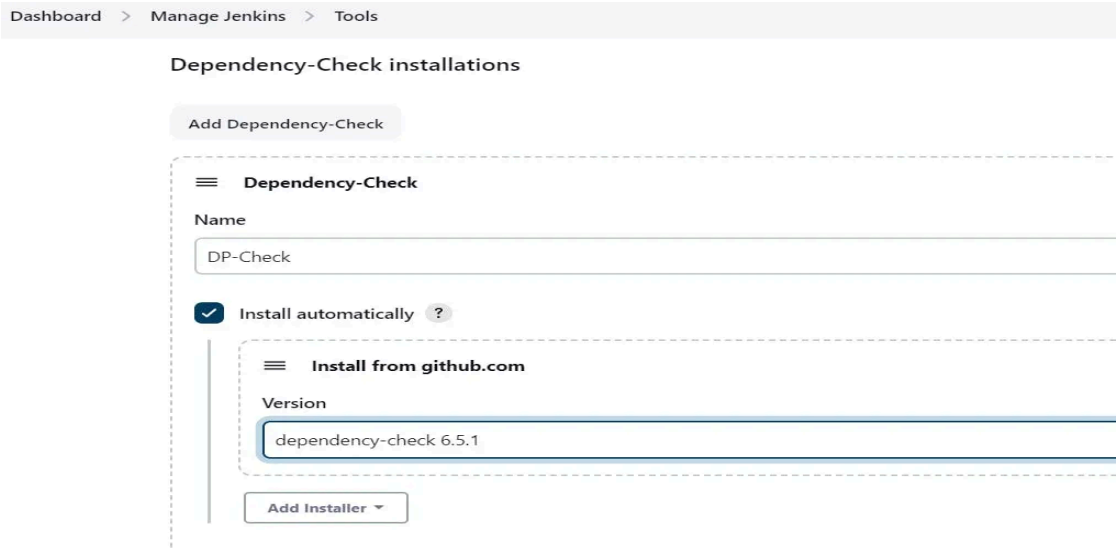
Step 5 — Install OWASP Dependency Check Plugins

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



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

Go to Dashboard → Manage Jenkins → Tools →



Now go configure → Pipeline and add OWASP and TRIVY stage to your pipeline and build.

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN
5s	379ms	1s	16s	520ms	1min 12s	1min 45s	13s
169ms	294ms	1s	28s	926ms (paused for 741ms)	2min 24s	3min 31s	27s

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

Dependency-Check Results

SEVERITY DISTRIBUTION

13 | 39 | 13

Search

File Name	Vulnerability	Severity	Weakness
ansi-html:0.0.7	NVD CVE-2021-23424	High	NVD-CWE-noinfo
ansi-regex:4.1.0	NVD CVE-2021-3807	High	CWE-1333
async:2.6.3	NVD CVE-2021-43138	High	CWE-1321
browserslist:4.14.2	NVD CVE-2021-23364	Medium	CWE-1333
css-what:3.4.2	OSSINDEX CVE-2022-21222	High	CWE-1333
decode-uri-component:0.2.0	NVD CVE-2022-38778	Medium	CWE-20
decode-uri-component:0.2.0	NVD CVE-2022-38900	High	CWE-20
ejs:2.7.4	OSSINDEX CVE-2022-29078	High	CWE-94
eventsource:1.1.0	NVD CVE-2022-1650	Critical	CWE-212
express:4.17.1	OSSINDEX CVE-2022-24999	High	CWE-1321

Step 6 — 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

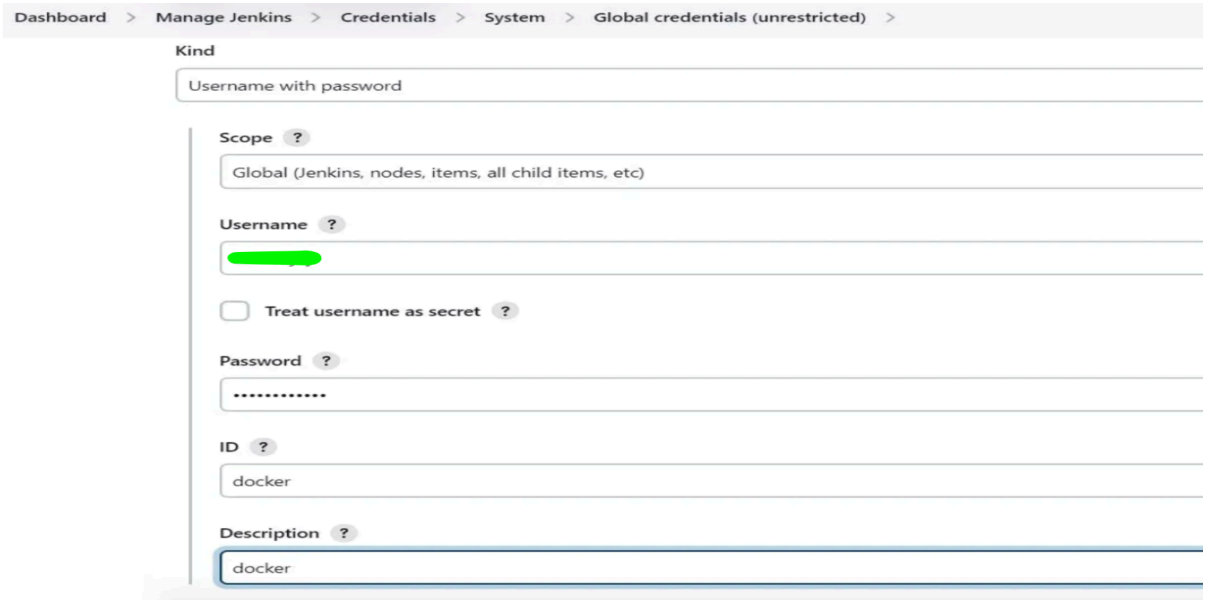
Docker Pipeline

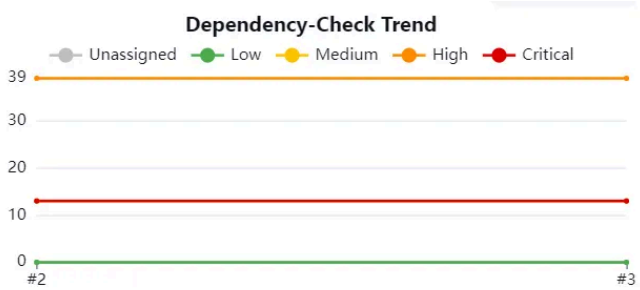
Docker API

Docker-build-step

Now, goto Dashboard → Manage Jenkins → Tools →

Add Docker Hub Username and Password under Global Credentials





Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY
3s	366ms	1s	19s	451ms	1min 20s	2min 1s	16s	3min 9s	4s
154ms	341ms	1s	25s	315ms	1min 36s	2min 31s	23s	3min 9s	4s

Now Run the container to see if the game coming up or not by adding below stage

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY	Deploy to container
144ms	284ms	1s	25s	410ms	1min 47s	2min 43s	23s	2min 7s	36s	789ms
146ms	251ms	1s	26s	305ms	1min 36s	2min 35s	23s	1min 50s	2min 8s	1s

<Jenkins-public-ip:3000>

ASSIGNMENT < CREATE AND INTEGRATE THE k8 CLUSTER WITH CICD PIPELINE >

Step 8 — Kubernetes Setup

Take-Two Ubuntu 20.04 instances one for k8s master and the other one for worker. T2.MEDIUM 15 GB

Install Kubectl on Jenkins machine also.

https://github.com/AWS-AZURE-Bootcamp5/tools-setup-project1/blob/main/kubectl.sh

Part 1 — — — — — Master Node — — — — — —

Set the hostname for Master Server
sudo hostnamectl set-hostname K8s-Master

Part 1.1 — — — — — Worker Node — — — — — —

Set the hostname for Worker Server
sudo hostnamectl set-hostname K8s-Worker

Part 2 — — — — — — Both Master & Node — — —

Install Kubeadm/Kubelet/kubectl

https://github.com/AWS-AZURE-Bootcamp5/tools-setup-project1/blob/main/kubeadm.sh

Part 3 — — — —ON Master Node — — —

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

```
#IMPORTANT### in case your in root exit from it and run below commands

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
```

YOU WILL GET THE TOKEN AND THE COMMAND LIKE BELOW AFTER YOU EXECUTE THE ABOVE COMMAND

```
sudo kubeadm join <master-node-ip>:<master-node-port> --token <token>
--discovery-token-ca-cert-hash <hash>
```

COPY ABOVE TOKEN AND PASTE IN WORKER NODE

Copy the config file FROM K8 MASTER to the local laptop and save it with a name **secret-file.txt** and use this at the Kuberenetes credential section

Welcome | secret-file.txt | squ_0c2a3e5ae5d8c72b37ec49205eebcef85f07 Untitle

Users > praveensingampalli > Downloads > secret-file.txt


```yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENI
    server: https://172.31.1.203:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURJVENDQW
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0
```


PART 4 - Install Kubernetes Plugins

Dashboard > Manage Jenkins > Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

Plugins

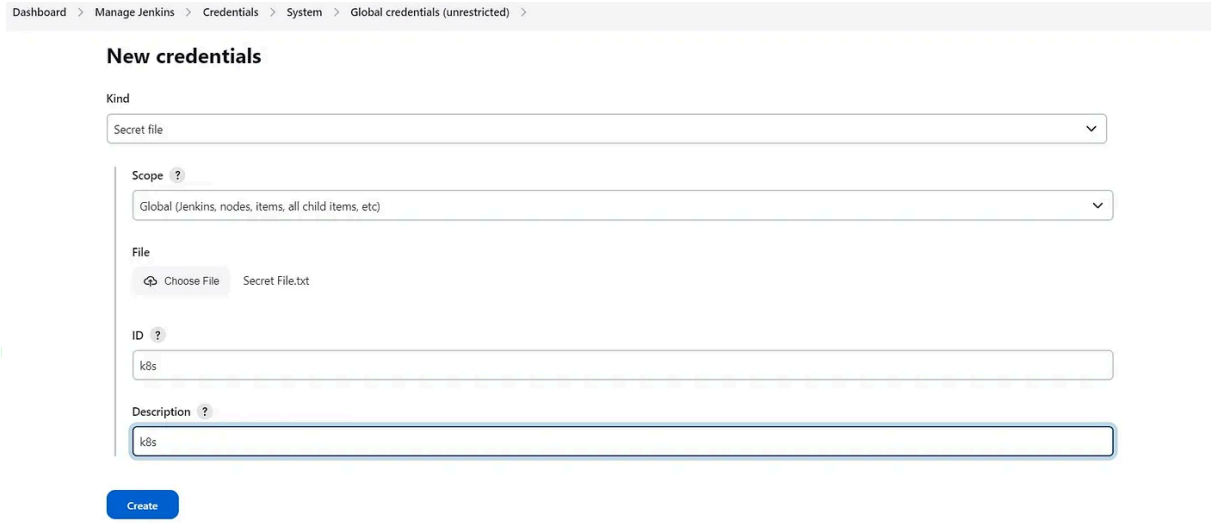
Kuber

Install

Install	Name ↓	Released
☑	Kubernetes Credentials 0.11 kubernetes credentials Common classes for Kubernetes credentials	9 days 16 hr ago
☑	Kubernetes Client API 6.8.1-224.vd388fca_4db_3b_ kubernetes Library plugins (for use by other plugins) Kubernetes Client API plugin for use by other Jenkins plugins.	9 days 17 hr ago
☑	Kubernetes 4029.v5712230ccb_f8 Cloud Providers Cluster Management kubernetes Agent Management This plugin integrates Jenkins with Kubernetes	9 days 15 hr ago
☑	Kubernetes CLI 1.12.1 kubernetes Configure kubectl for Kubernetes	8 days 22 hr ago

PART 5 - Go to manage Jenkins → manage credentials → Click on Jenkins global → add credentials

RUN THE PIPELINE



Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY	Deploy to container	Deploy to kubernets
132ms	264ms	1s	25s	295ms	1min 49s	2min 38s	23s	1min 51s	1min 35s	1s	2s
133ms	261ms	1s	25s	284ms	1min 51s	2min 46s	23s	1min 23s	1min 52s	1s	1s

PART 5 - In the Kubernetes cluster give this command

kubectl get all

kubectl get svc

STEP9:Access from a Web browser with

<public-ip-of-slave:service port>

Step 10: Terminate instances.