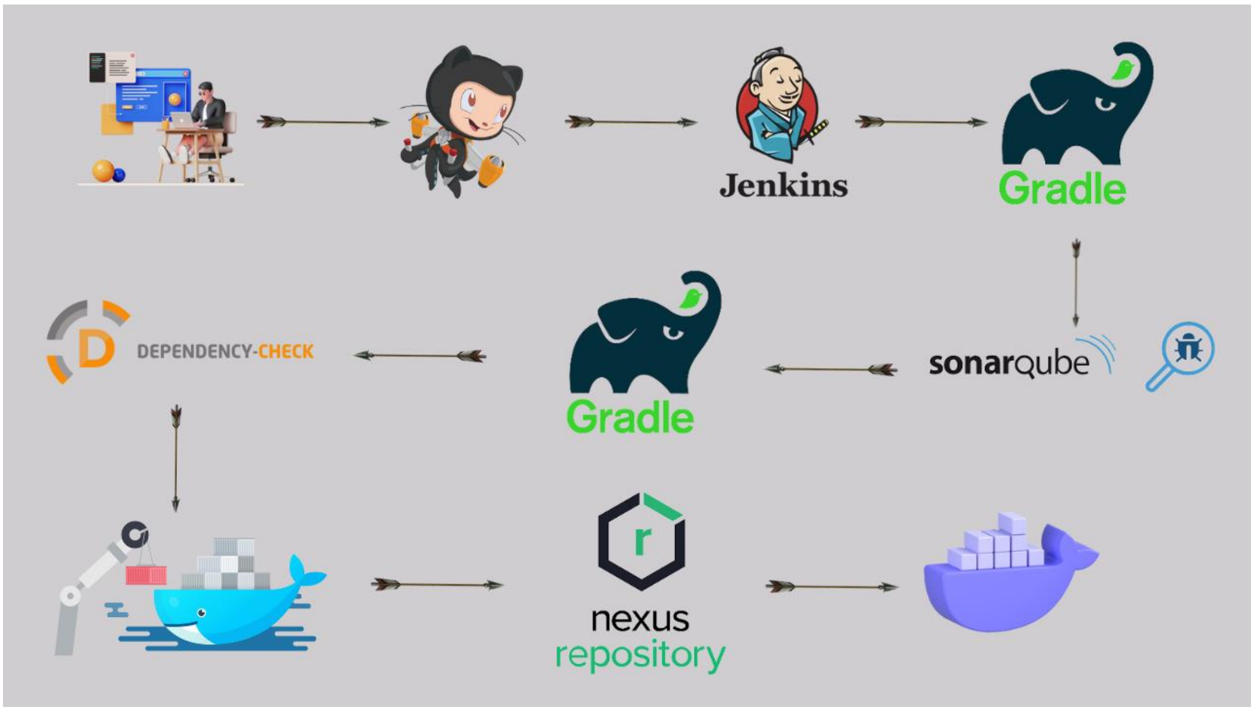
COMPLETE CI/CD GRADLE PROJECT DEVSECOPS



Hello friends, we will be deploying a Gradle Java Based Application. This is an everyday use case scenario used by several organizations. We will be using Jenkins as a CICD tool and deploying our application on Docker.
We will be deploying our application using Docker Container ,which docker image is stored in nexus repository.

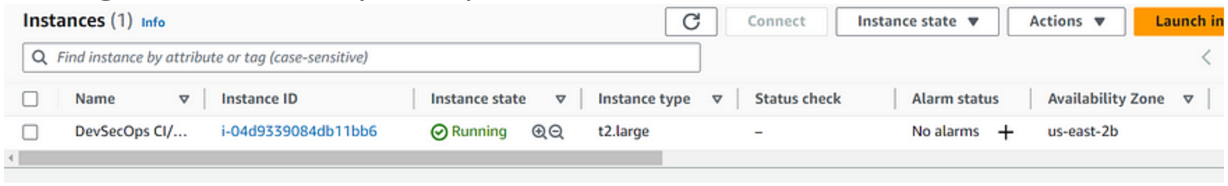
Steps:-

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,Gradle
Step 4 — Create a Pipeline Project in Jenkins using Declarative Pipeline
Step 5 — Install OWASP Dependency Check Plugins
Step 6 — launch t2medium instance for Nexus and setup
Step 7 — Docker Image Build and Push to nexus
Step 8 — Access the Real World Application
Step 9 — Terminate the AWS EC2 Instance

References

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

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



Instances (1) Info

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

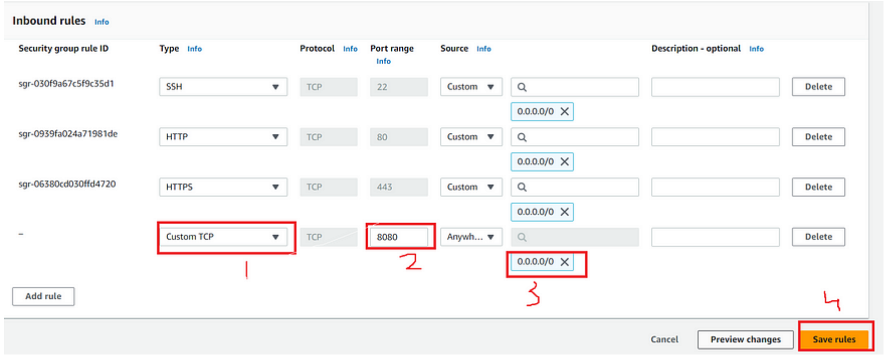
Step 2 — Install Jenkins, Docker and Trivy

2A — To Install Jenkins

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

```
sudo apt-get update

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo
tee \
    /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
    https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
    /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update
sudo apt install openjdk-17-jdk
sudo apt install openjdk-17-jre

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

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

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



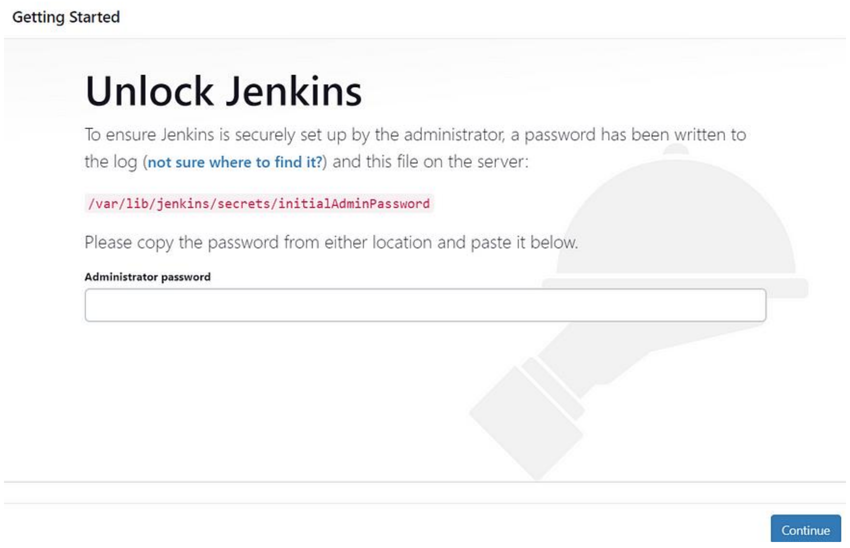
Inbound rules Info

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

Now, grab your Public IP Address

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

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


Getting Started

Unlock Jenkins

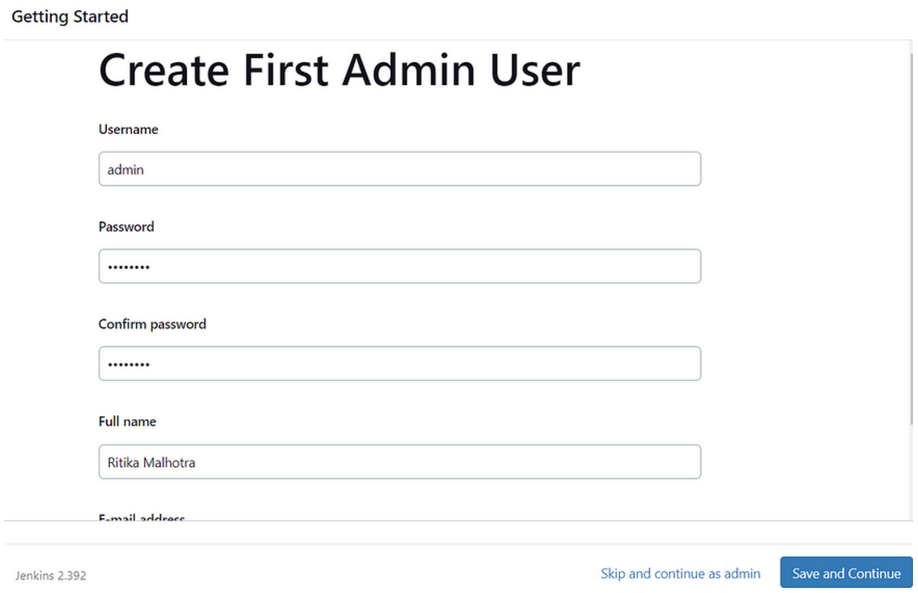
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

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

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

Administrator password

Jenkins will now get installed and install all the libraries.


Getting Started

Create First Admin User

Username: admin

Password: ••••••••

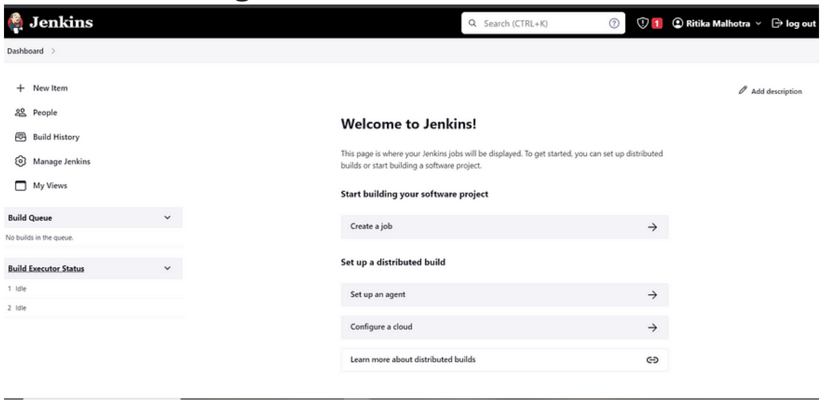
Confirm password: ••••••••

Full name: Ritika Malhotra

E-mail address

Jenkins 2.392

Jenkins Getting Started Screen



Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

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

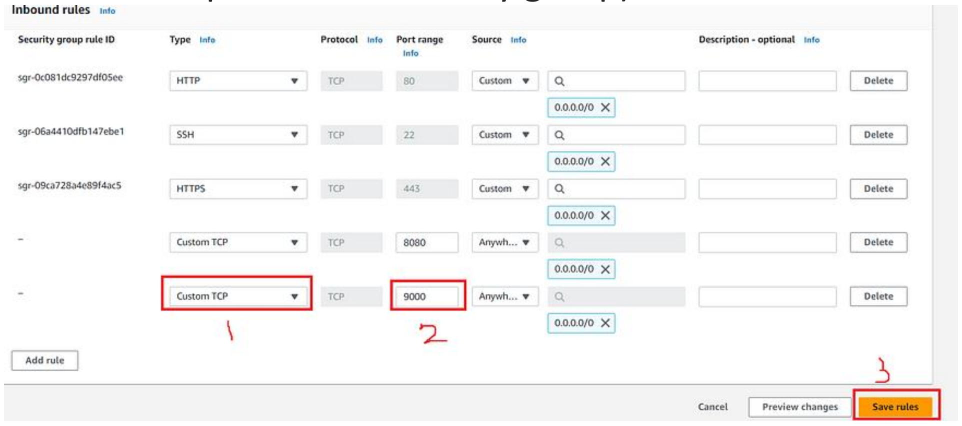
Configure a cloud

Learn more about distributed builds

2B — Install Docker

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

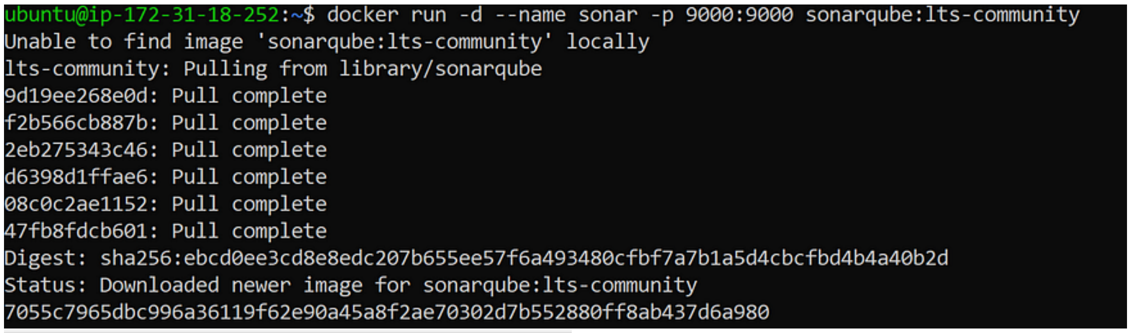
After the docker installation, we create a sonarqube container (Remember added 9000 port in the security group)



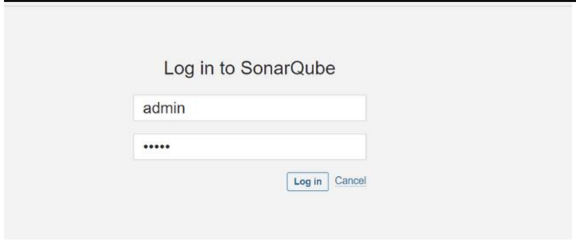
Inbound rules

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 -p 9000:9000 sonarqube:lts-community
```



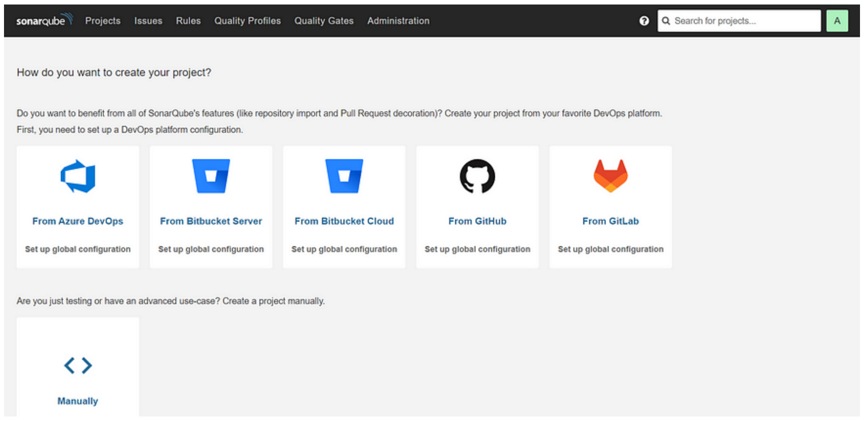
```
ubuntu@ip-172-31-18-252:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
9d19ee268e0d: Pull complete
f2b566cb887b: Pull complete
2eb275343c46: Pull complete
d6398d1ffae6: Pull complete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



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.

2C — Install Trivy

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

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

echo "deb [signed-by=/usr/share/keyrings/trivy.gpg]
https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo
tee -a /etc/apt/sources.list.d/trivy.list

sudo apt-get update

sudo apt-get install trivy -y
```

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

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

3A — Install Plugin

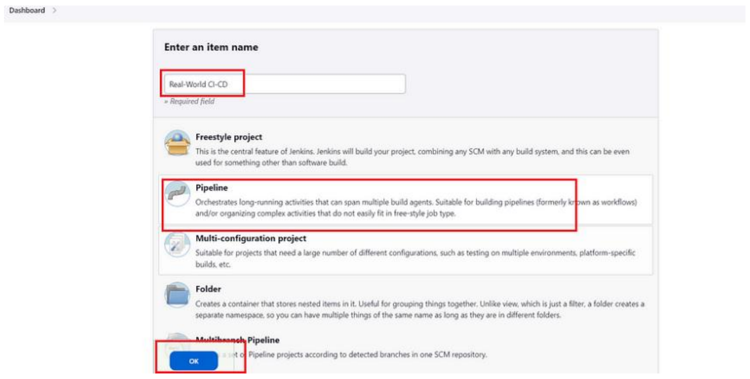
Goto Manage Jenkins →Plugins → Available Plugins →
Install below plugins
1 → Eclipse Temurin Installer (Install without restart)
2 → SonarQube Scanner (Install without restart)

3B — Configure Java and Maven in Global Tool Configuration

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

3C — Create a Job

Label it as Gradle, click on Pipeline and Ok.



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.

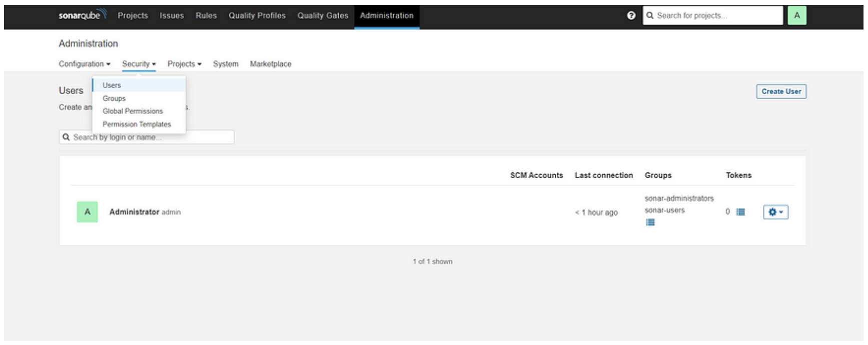
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

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
… Pipeline projects according to detected branches in one SCM repository.

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

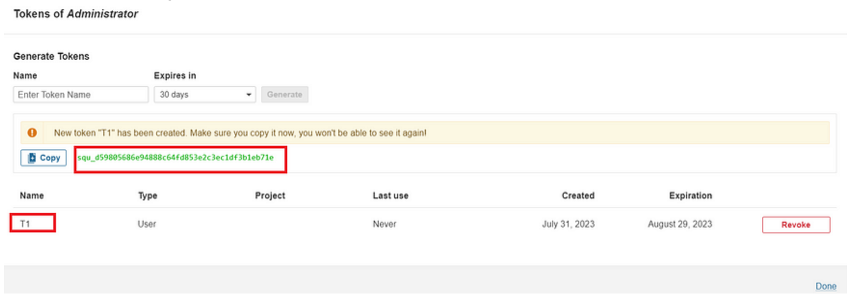


Users

	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: Enter Token Name — Expires in: 30 days

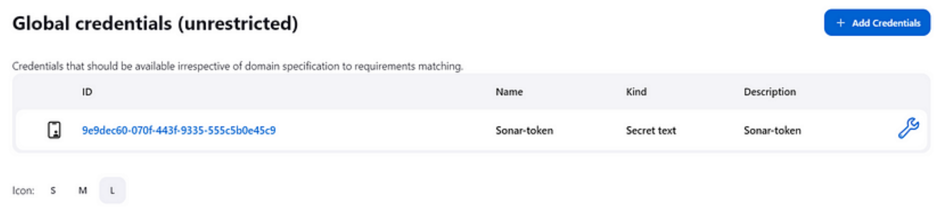
New token "T1" has been created. Make sure you copy it now, you won't be able to see it again!

squ_d59805686e94888c64fd853e2c3ec1df3b1eb71e

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

Copy this Token

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

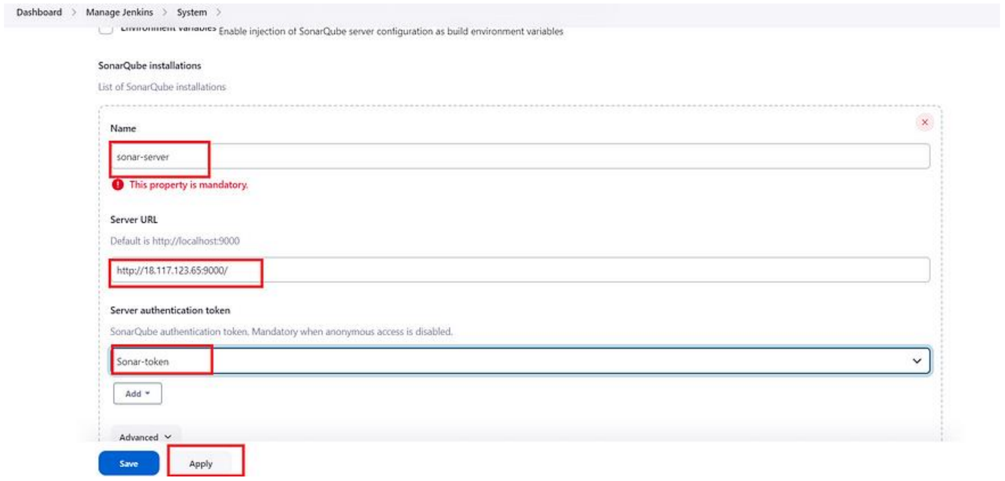


Global credentials (unrestricted)

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


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.

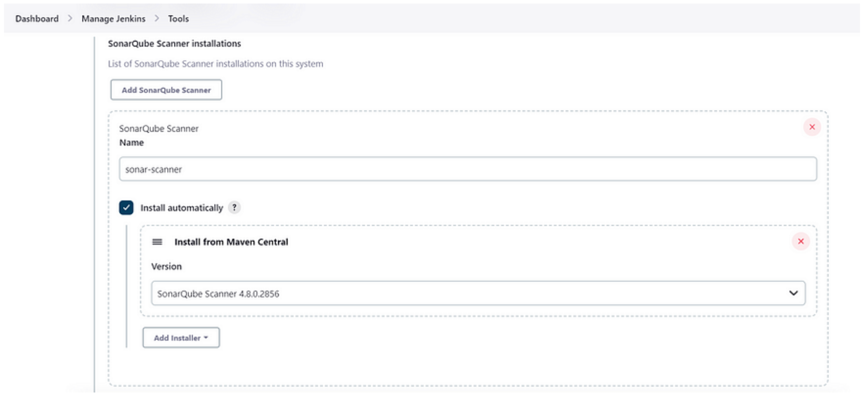
Sonar-token

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

SonarQube Scanner

Name: sonar-scanner

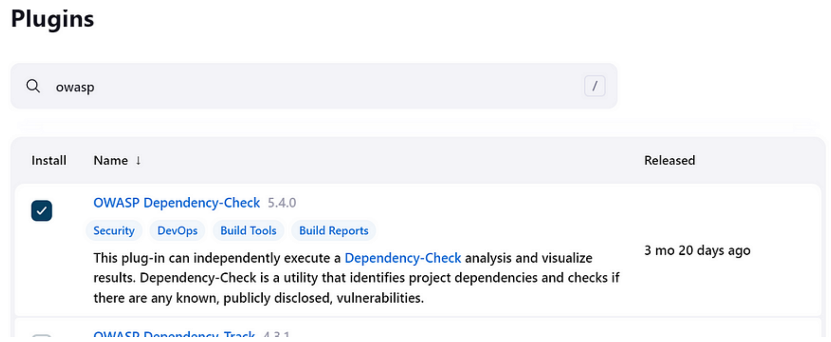
Install automatically: ✓

Install from Maven Central

Version: SonarQube Scanner 4.8.0.2856

Step 5 — Install OWASP Dependency Check Plugins

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

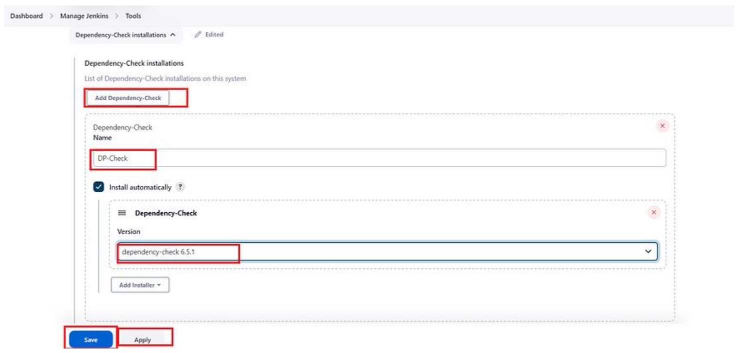

Plugins

owasp

Install	Name ↓	Released
✓	OWASP Dependency-Check 5.4.0 Security DevOps Build Tools Build Reports 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
Add Dependency-Check
Add Installer
Save
Apply

Dependency-Check installations
List of Dependency-Check installations on this system

Dependency-Check
Name: DP-Check

[✓] Install automatically

Dependency-Check
Version: dependency-check 6.5.1

Click on apply and Save here.

Step 6 — Nexus repo launch
Take an Ubuntu t2 medium instance and paste the below commands

Prequesites :
4 cpus max and min used 2 in this video
20 gb storage
java 8
Aws account

```
sudo apt update                                   #update packages
```

Now, install the Java OpenJDK 8 via the apt command below. Input Y when asked to confirm the installation and press ENTER to proceed.

```
sudo apt install openjdk-8-jdk                    # install java8
```

Once Java is installed, verify the Java version on your system using the following command.

```
java -version
```

To install Nexus, you will need to create a new dedicated Linux user with a valid shell and
also need to set up the max open files for both hard and soft limits to '65536'.

Run the following command to create a new dedicated user for the Nexus with the name 'nexus'

```
sudo useradd -d /opt/nexus -s /bin/bash nexus
sudo passwd nexus
```

Add password

Next, set the ulimit to '65536' using the below command. This will only affect the system on the current system temporarily.
To make it permanent, you can create a new config file that you will do in the next step.

```
ulimit -n 65536
```

TO set up ulimit permanently, create a new config file '/etc/security/limits.d/nexus.conf' using nano editor.

```
sudo nano /etc/security/limits.d/nexus.conf
    nexus - nofile 65536
```

```
ulimit -a
```

Download the Nexus Repository Manager package via the wget command as below. If the download process is finished,
you will see the file 'nexus-3.41.1-01-unix.tar.gz' on your current working directory.

```
sudo wget https://download.sonatype.com/nexus/3/nexus-3.41.1-01-unix.tar.gz
```

Now extract the file 'nexus-3.41.1-01-unix.tar.gz' via the tar command below. And you should get two directories, the 'nexus-3.41.1-01' and 'sonatype-work'.

The directory 'nexus-3.41.1-01' is the main directory for the Nexus package, and the directory 'sonatype-work' is the main working directory for Nexus.

```
sudo tar xzf nexus-3.41.1-01-unix.tar.gz
```

Next, move those extracted directories to '/opt' using the following command.
The Nexus package directory will be '/opt/nexus' and the Nexus working directory will be '/opt/sonatype-work'.

```
sudo mv nexus-3.41.1-01 /opt/nexus
sudo mv sonatype-work /opt/
```

Lastly, change the ownership of both directories to the user and group 'nexus' via the chown command below.

```
sudo chown -R nexus:nexus /opt/nexus /opt/sonatype-work
```

Next, you will set up your Nexus installation by editing some of the Nexus configuration files.

Open the file '/opt/nexus/bin/nexus.rc' using nano editor.

```
sudo nano /opt/nexus/bin/nexus.rc
```

Uncomment the option 'run_as_user' and change the value to 'nexus'. With this configuration, you will be running the Nexus application as the system user 'nexus'.

```
run_as_user='nexus'
```

Save the file and exit the editor when you are done.

Next, open the config file '/etc/nexus/bin/nexus.vmoptions' using the nano editor to set up the max heap memory for Nexus.
****

```
sudo vi /opt/nexus/bin/nexus.vmoptions
```

```
-Xms1024m
-Xmx1024m
-XX:MaxDirectMemorySize=1024m

-XX:LogFile=./sonatype-work/nexus3/log/jvm.log
-XX:-OmitStackTraceInFastThrow
-Djava.net.preferIPv4Stack=true
-Dkaraf.home=.
-Dkaraf.base=.
-Dkaraf.etc=etc/karaf
-Djava.util.logging.config.file=/etc/karaf/java.util.logging.properties
-Dkaraf.data=./sonatype-work/nexus3
-Dkaraf.log=./sonatype-work/nexus3/log
-Djava.io.tmpdir=./sonatype-work/nexus3/tmp
```
****

To run nexus as service using Systemd

```
sudo nano /etc/systemd/system/nexus.service
```
****

```
[Unit]
Description=nexus service
After=network.target

[Service]
Type=forking
LimitNOFILE=65536
ExecStart=/opt/nexus/bin/nexus start
ExecStop=/opt/nexus/bin/nexus stop
User=nexus
Restart=on-abort
```

```
[Install]
WantedBy=multi-user.target
```
****
****

```
sudo systemctl daemon-reload
sudo systemctl start nexus.service
sudo systemctl enable nexus.service
sudo systemctl status nexus.service
```

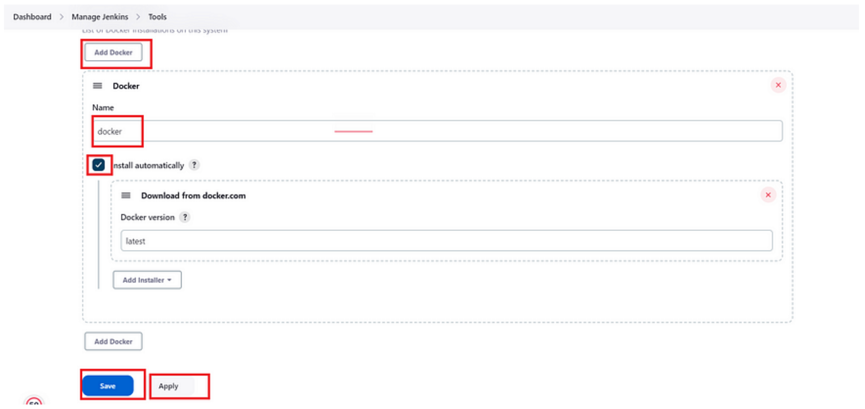
if the nexus service is not started, you can the nexus logs using below command

```
tail -f /opt/sonatype-work/nexus3/log/nexus.log
```

in ec2 instance add 8081 port for Nexus

```
sudo cat /opt/nexus/sonatype-work/nexus3/admin.password
```

Step 7 – Docker build and push to docker
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

Add Docker

Docker

Name: docker

[✓] Install automatically

Download from docker.com

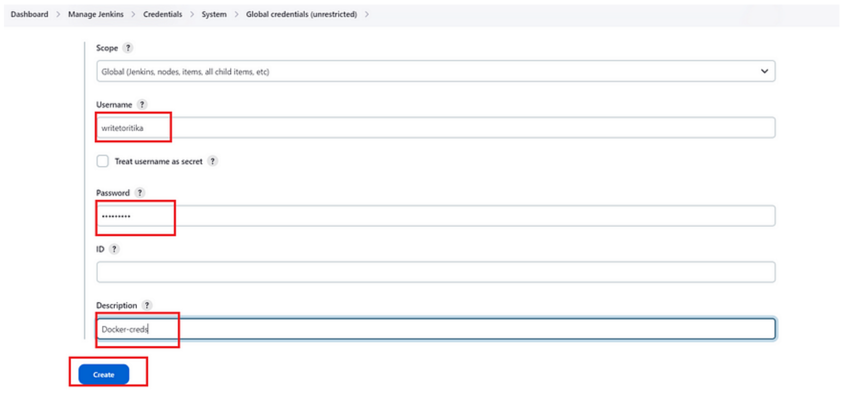
Docker version: latest

Add Installer

Add Docker

Save | Apply

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: ••••••••••

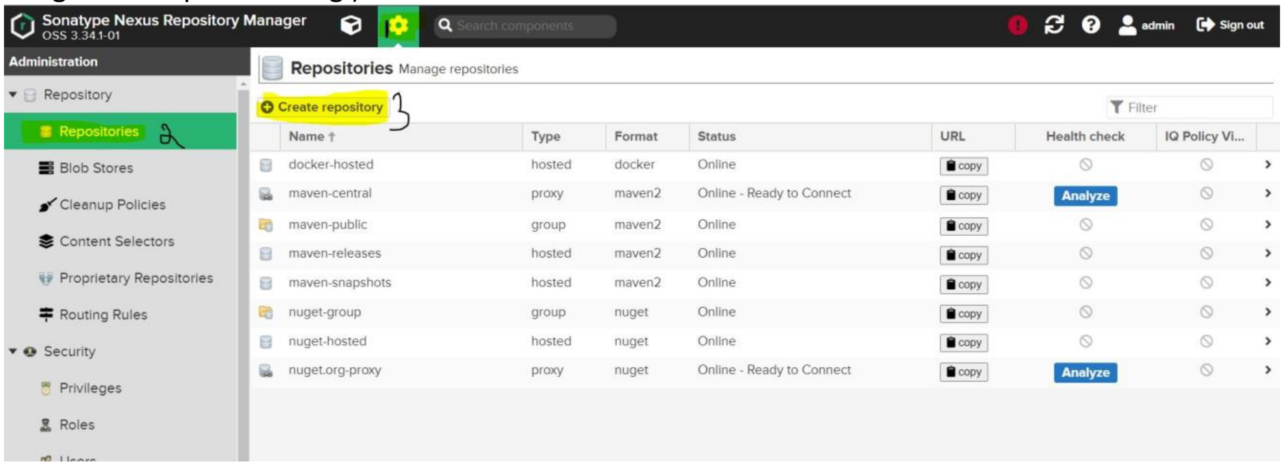
ID:

Description: Docker-creds

Create

Initial setup

In nexus click on gear button --> click on repositories --> click on create repository (below image can help in creating)



Sonatype Nexus Repository Manager
OSS 3.34.1-01

Search components | admin | Sign out

Administration

- Repository
  - Repositories
  - Blob Stores
  - Cleanup Policies
  - Content Selectors
  - Proprietary Repositories
  - Routing Rules
- Security
  - Privileges
  - Roles
  - Users

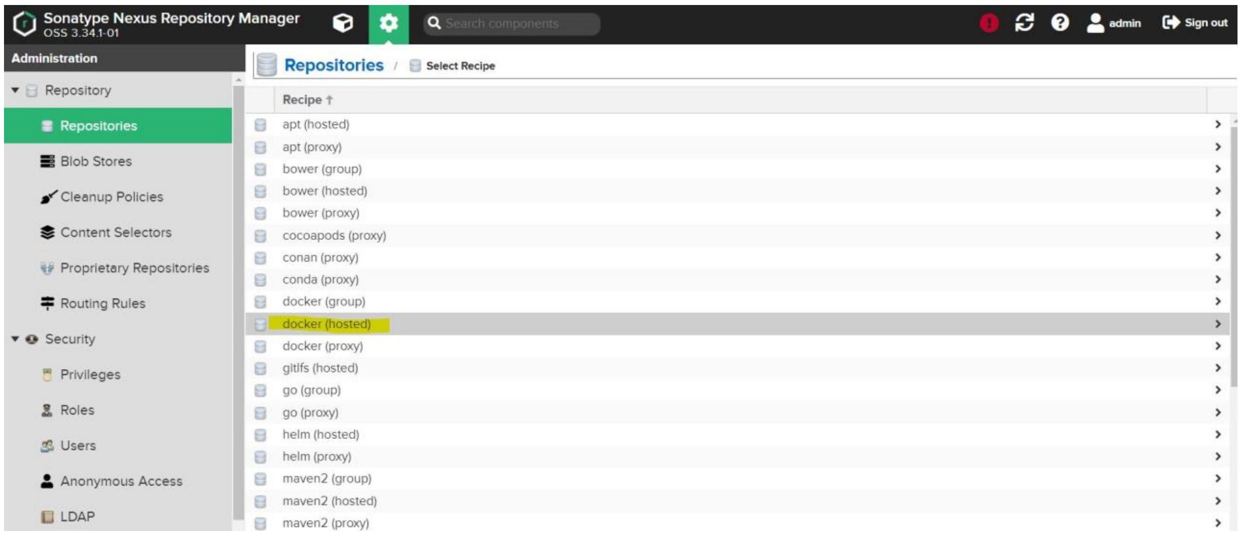
Repositories Manage repositories

Create repository

Filter

Name ↑	Type	Format	Status	URL	Health check	IQ Policy Vi...
docker-hosted	hosted	docker	Online	copy		
maven-central	proxy	maven2	Online - Ready to Connect	copy	Analyze	
maven-public	group	maven2	Online	copy		
maven-releases	hosted	maven2	Online	copy		
maven-snapshots	hosted	maven2	Online	copy		
nuget-group	group	nuget	Online	copy		
nuget-hosted	hosted	nuget	Online	copy		
nuget.org-proxy	proxy	nuget	Online - Ready to Connect	copy	Analyze	

once we click on create repository (types of repository will be listed) --> click on docker(hosted)


Sonatype Nexus Repository Manager
OSS 3.34.1-01

Search components

admin Sign out

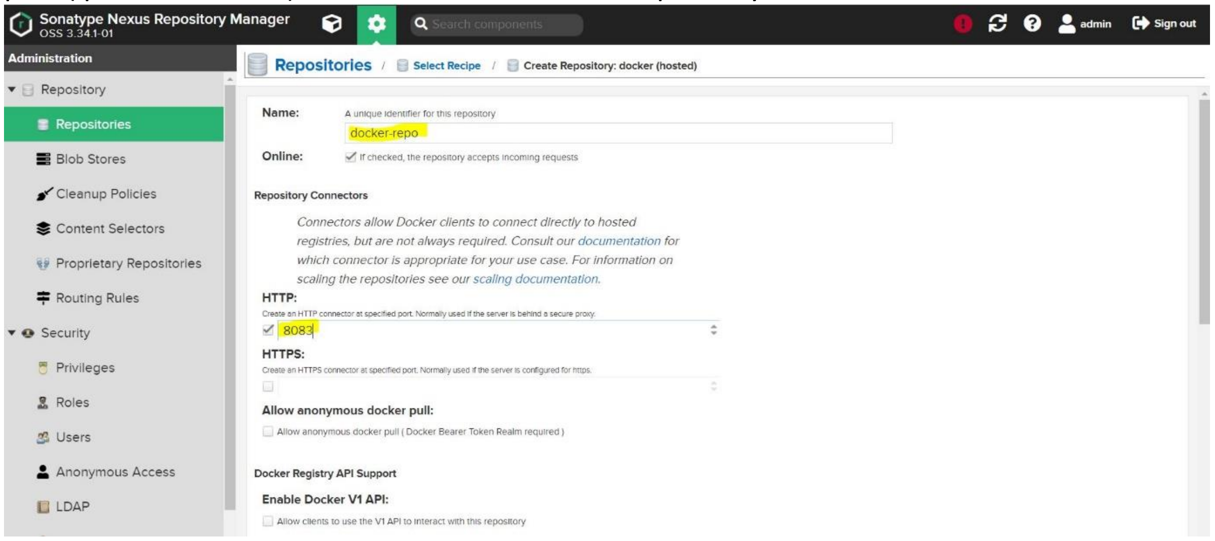
Administration

- Repository
  - Repositories
  - Blob Stores
  - Cleanup Policies
  - Content Selectors
  - Proprietary Repositories
  - Routing Rules
- Security
  - Privileges
  - Roles
  - Users
  - Anonymous Access
  - LDAP

Repositories / Select Recipe

Recipe ↑
apt (hosted)
apt (proxy)
bower (group)
bower (hosted)
bower (proxy)
cocoapods (proxy)
conan (proxy)
conda (proxy)
docker (group)
docker (hosted)
docker (proxy)
gitlfs (hosted)
go (group)
go (proxy)
helm (hosted)
helm (proxy)
maven2 (group)
maven2 (hosted)
maven2 (proxy)

fill out the details in Name (unique name), enable checkbox beside to HTTP and enter a valid port (preferred 8083) once that click on create repository


Sonatype Nexus Repository Manager
OSS 3.34.1-01

Search components

admin Sign out

Administration

- Repository
  - Repositories
  - Blob Stores
  - Cleanup Policies
  - Content Selectors
  - Proprietary Repositories
  - Routing Rules
- Security
  - Privileges
  - Roles
  - Users
  - Anonymous Access
  - LDAP

Repositories / Select Recipe / Create Repository: docker (hosted)

Name: A unique identifier for this repository
docker-repo

Online: ✓ If checked, the repository accepts incoming requests

Repository Connectors

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our documentation for which connector is appropriate for your use case. For information on scaling the repositories see our scaling documentation.

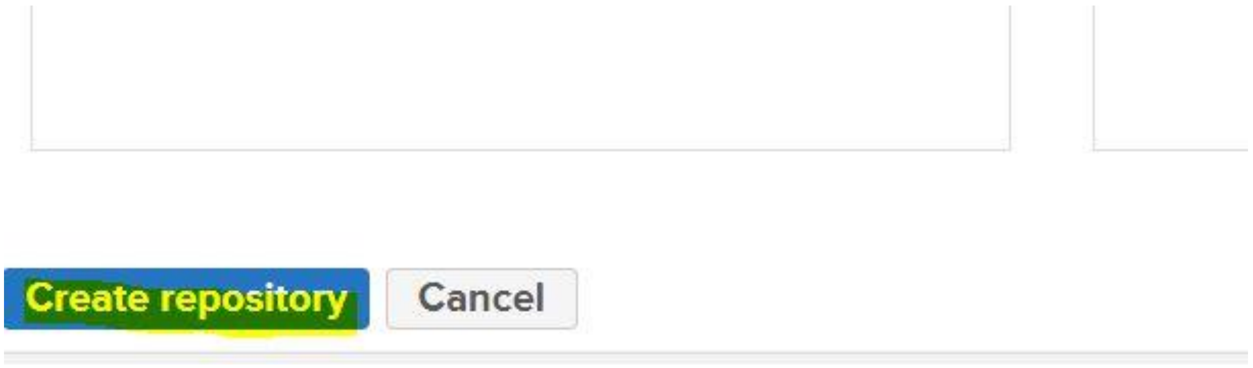
HTTP:
Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy.
✓ 8083

HTTPS:
Create an HTTPS connector at specified port. Normally used if the server is configured for https.
☐

Allow anonymous docker pull:
☐ Allow anonymous docker pull (Docker Bearer Token Realm required)

Docker Registry API Support

Enable Docker V1 API:
☐ Allow clients to use the V1 API to interact with this repository



Once this set up is done in jenkins host we need to setup Insecure Registries. to do that we need to edit or if not present create a file `/etc/docker/daemon.json` in that file add details of nexus

```
{  "insecure-registries":["nexus_machine_ip:8083"] }
```

once that's done we need to execute `systemctl restart docker` this is to apply new changes, also we can verify whether registry is added or not by executing `docker info`
once this is done from jenkins host you can try
`docker login -u nexus_username -p nexus_pass nexus_ip:8083`

```
        pipeline{

    agent any
    tools{
        jdk 'jdk11'
        gradle 'gradle'
    }
    stages{
        stage('Cleanws'){
            steps{
                cleanWs()
            }
        }
        stage('checkout from scm'){
            steps{
                git branch: 'main', url:
'https://github.com/Aj7Ay/Java_Gradle_Responsive_Website.git'
            }
        }
        stage('Gradle compile'){
            steps{
                sh 'chmod +x gradlew'
                sh './gradlew compileJava'
            }
        }
        stage('Test Gradle'){
            steps{
                sh 'chmod +x gradlew'
                sh './gradlew test'
            }
        }
        stage('sonarqube Analysis'){
            steps{
                script{
                      withSonarQubeEnv(credentialsId: 'Sonar-token') {
                        sh 'chmod +x gradlew'
                        sh './gradlew sonarqube'
                    }
                    //quality gate
                    timeout(time: 10, unit: 'MINUTES'){
                        def qg = waitForQualityGate()
                        if (qg.status != 'OK'){
                            error "pipeline is aborted due to qualitygate 
failure: ${qg.status}"
                        }
                    }
                }
            }
        }
        stage('build Gradle'){
            steps{
                sh 'chmod +x gradlew'
                sh './gradlew build'
            }
        }
        stage("OWASP Dependency Check"){
            steps{
                dependencyCheck additionalArguments: '--scan ./ --format HTML ', 
odcInstallation: 'DP-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-
report.html'
            }
        }
        stage('build and push to nexus'){
            steps{
                script{
                    withCredentials([string(credentialsId: 'docker_pass', 
variable: 'docker_password')]) {
                        sh '''
                         docker build -t 43.204.235.20:8083/gradle1:latest .
                         docker login -u admin -p $docker_password 
43.204.235.20:8083

                         docker push 43.204.235.20:8083/gradle1:latest
                         '''
                    }
                }
            }
        }
        stage('deploy to container'){
            steps{
                script{
                    withCredentials([string(credentialsId: 'docker_pass', 
variable: 'docker_password')]) {
```

```
                        sh 'docker run -d --name g1 -p 8082:8080
43.204.235.20:8083/gradle1:latest'
                    }
                }
            }
        }
    }
}
```