**What Is Jenkins?**

Jenkins is defined as an open-source solution comprising an automation server to enable continuous integration and continuous delivery (CI/CD), automating the various stages of software development such as test, build, and deployment. This article explains how Jenkins works, its key features and use cases, and the functionality of a Jenkins pipeline.

1. **Continuous Integration and Continuous Delivery: -**

As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

1. **Easy installation: -**

Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Linux, macOS and other Unix-like operating systems.

1. **Easy configuration: -**

Jenkins can be easily set up and configured via its web interface, which includes on-the-fly error checks and built-in help.

1. **Plugins: -**

With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain.

1. **Extensible: -**

Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.

1. **Distributed: -**

Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.

**Step 1: Update System Packages: -**

Redhat Linux (RHEL 9.0)

$ sudo yum update

**Step 2: Install Java Development Kit (JDK): -**

$ sudo yum install vim -y

$ sudo yum install wget -y

$ sudo yum install httpd -y

$ sudo yum install fontconfig java-17-openjdk

$ sudo java --version

( note: update-alternatives --config java )

**Step 3: Add Jenkins Repository: -**

(Note: not use in vm Box in this command only used for ASW)

$ sudo yum install wget -y

$ sudo wget -O /etc/yum.repos.d/jenkins.repo \

<https://pkg.jenkins.io/redhat-stable/jenkins.repo>

$ sudo rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key>

**Step 4: Install Jenkins: -**

(Note:- download all jenkins file for oracle vm box <https://github.com/gauravkhandate/jenkins.git>)

Cloud network

$ sudo yum install Jenkins -y

$ sudo systemctl daemon-reload

$ sudo systemctl start Jenkins

$ sudo systemctl status jenkins

**Step 5: Configure Firewall: -**

$ sudo firewall-cmd --permanent --add-port=8080/tcp

$ sudo firewall-cmd --permanent --add-port={6443,2379,2380,10250,10251,10252,25,465}/tcp

$ sudo firewall-cmd --reload

$ sudo firewall-cmd --list-all

**Step 6: Configure Docker: -**

$ sudo yum install -y yum-utils

$ sudo yum-config-manager --add-repo <https://download.docker.com/linux/rhel/docker-ce.repo>

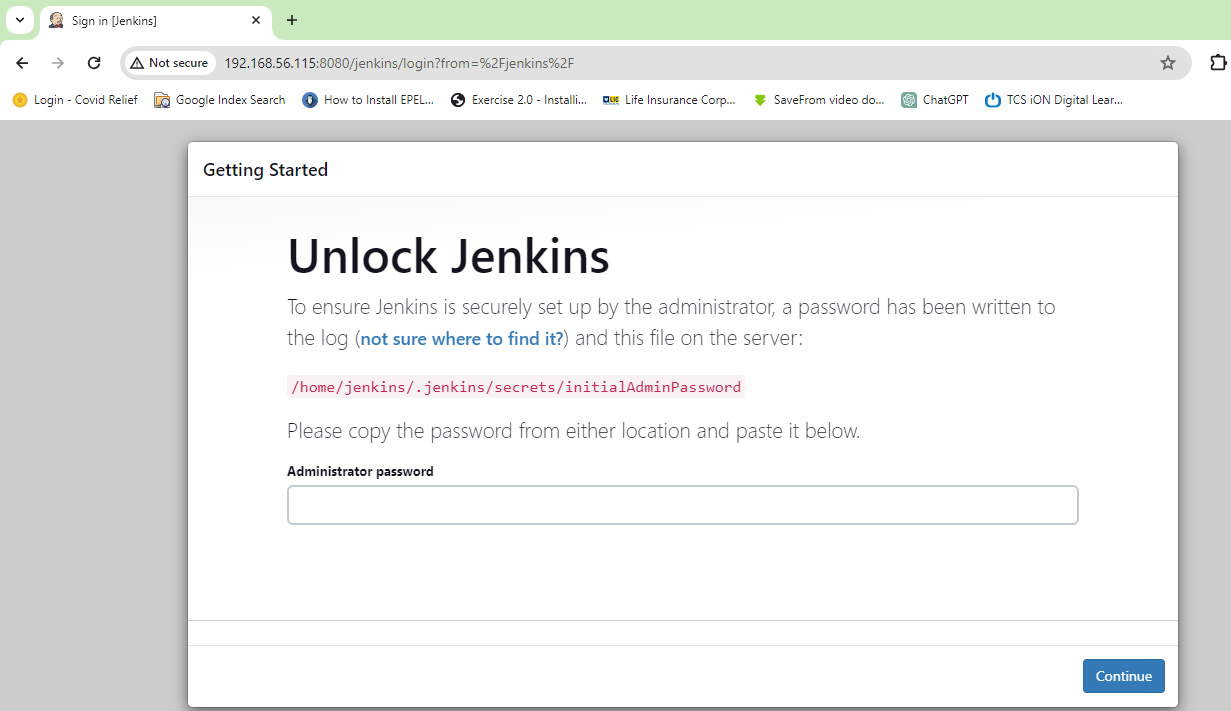
$ sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

$ sudo systemctl start docker

$ sudo systemctl enable --now docker

$ sudo chmod 666 /var/run/docker.sock

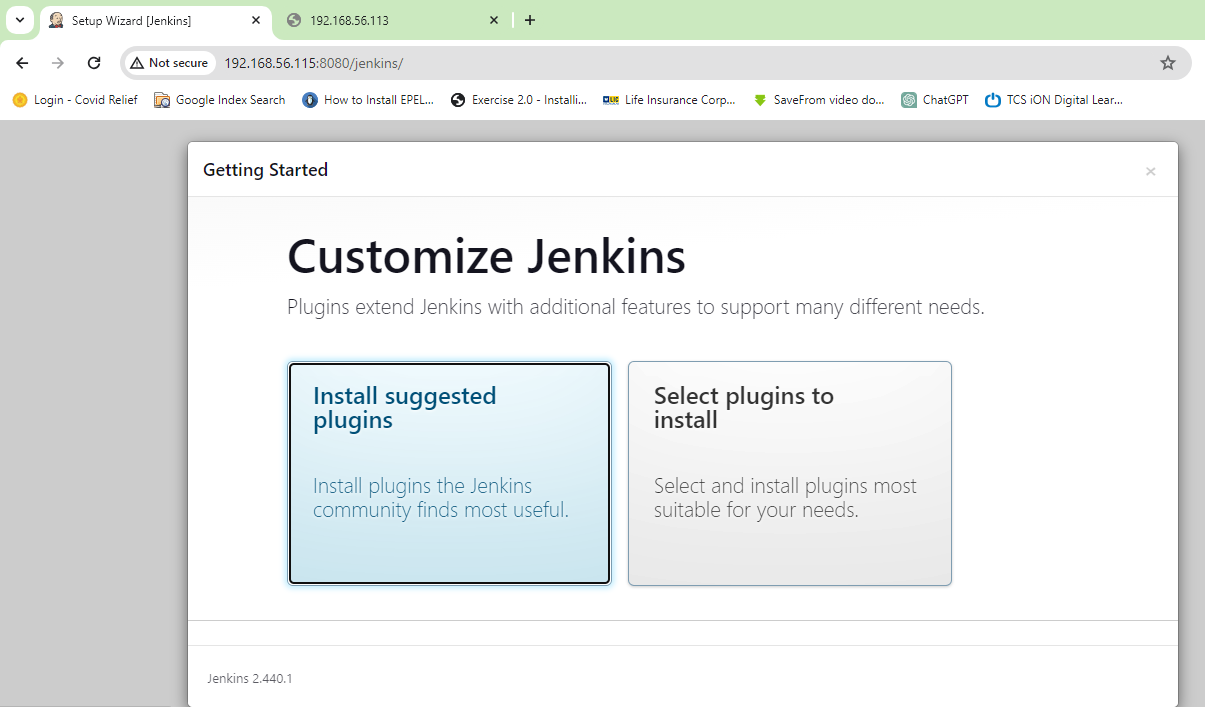
**Step 7: configure jenkins: -**



Note: -

* Copy this link and cat your terminal

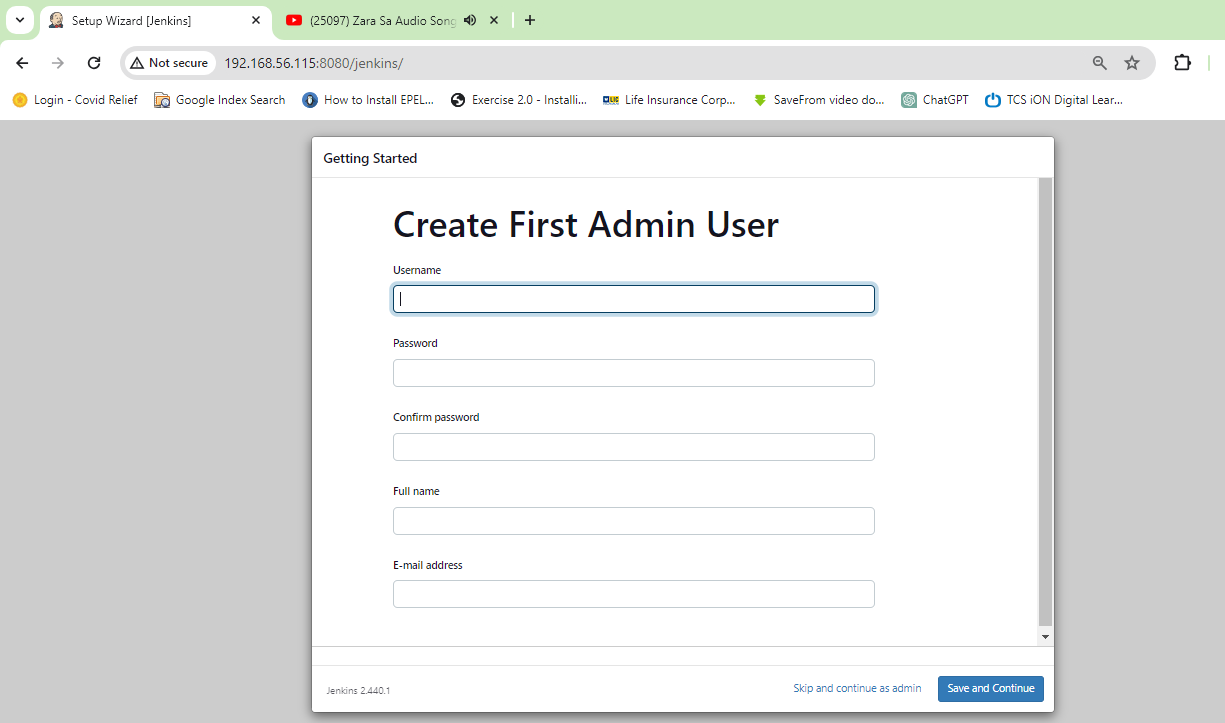
Ex. cat /home/jenkins/.jenkins/secrets/initialAdminPassword



Note: -

* Install suggested plugins

Create first Admin User detail

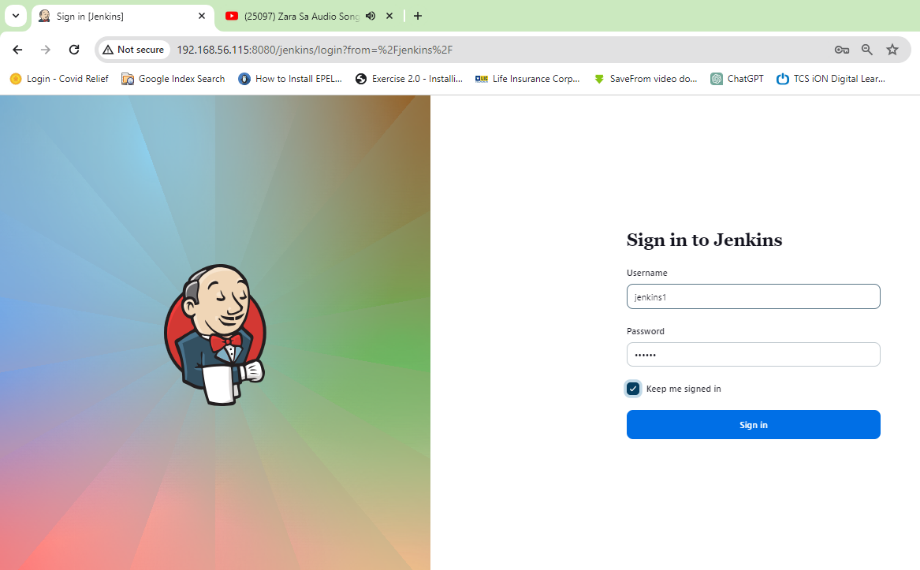


Than Instance Configuration

Clk save and finish

Than jenkins is ready

Clk Start using jenkins



Sign in to Jenkins and create CICD project

**Configure SonarQube forManage Plugins Jenkins: -**

Goto “Dashboard”

“Manage Jenkins”

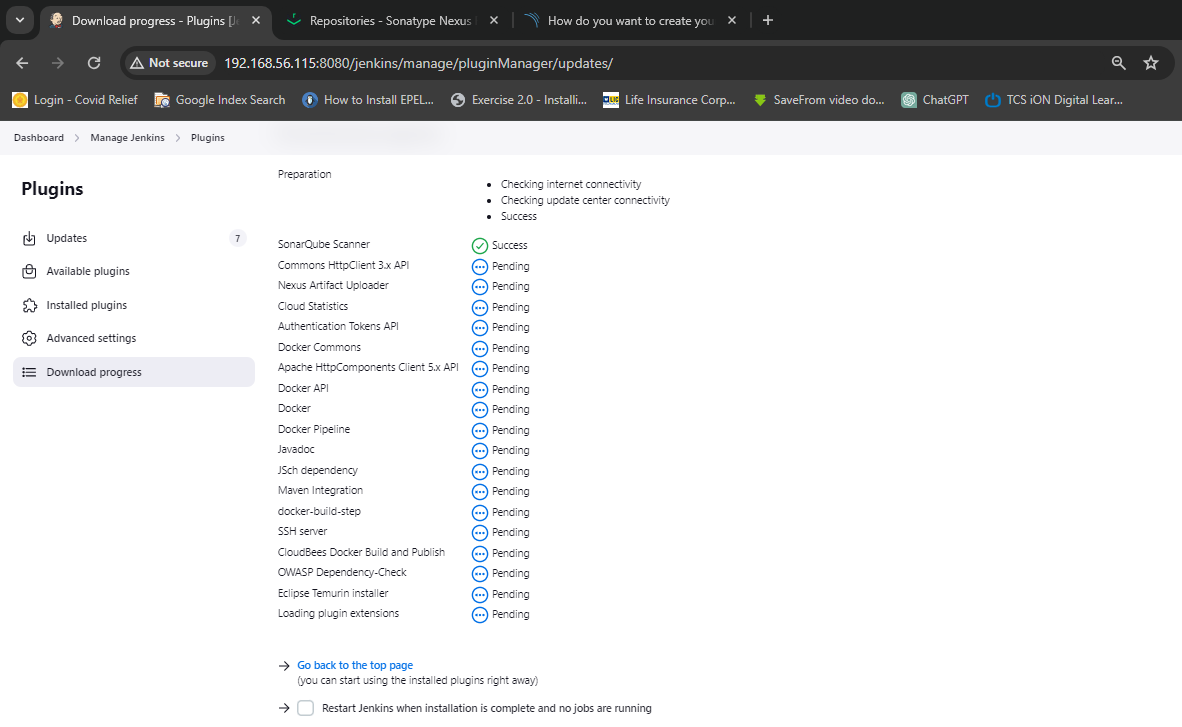
“Plugins” ----- ------ “Available Plugins”

|

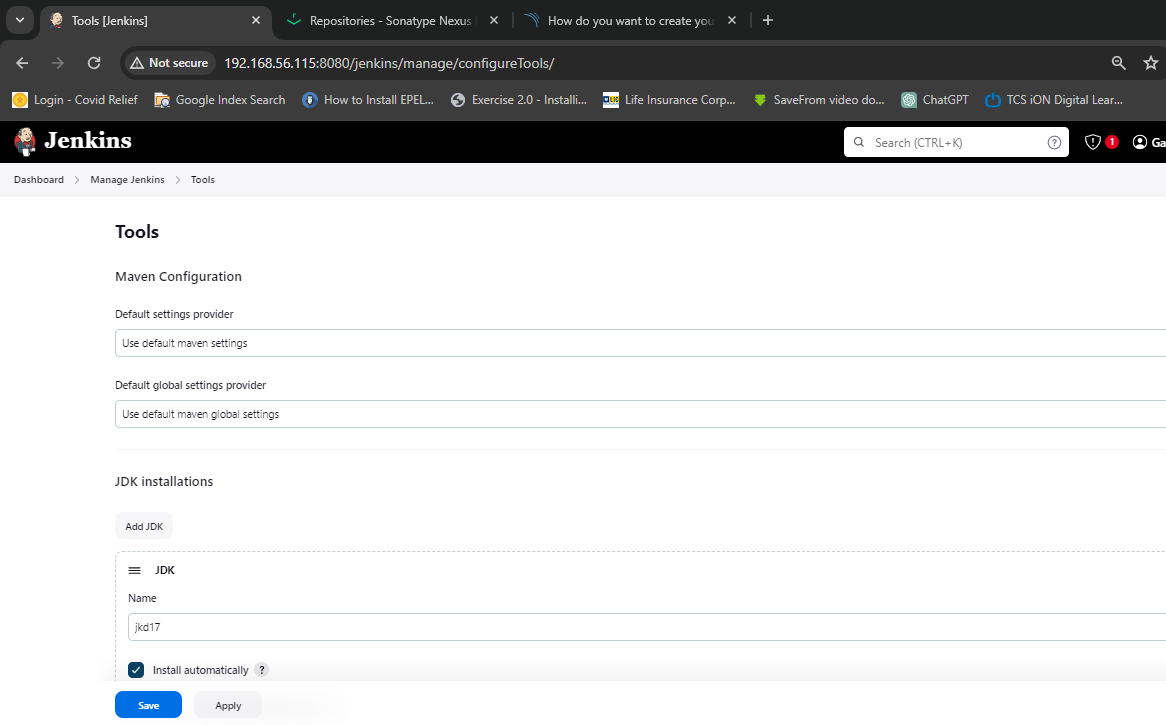
|

1. SonarQube Scanner
2. Nexus Artifact Uploader
3. Docker , Docker Pipeline , docker-build-step , CloudBees Docker Build and Publish
4. OWASP Dependency-Check
5. Eclipse Temurin installer

Install all these plugins



**Configure Tools: -**



Tools: -

1. “JDK installations”

‘Add jdk’

JDK

name: - jdk17

Install automatically: - Install from adoptium.net

Version: - jdk-17.0.8+7

1. “Git installations”

‘Git’

Name: - Default

Path to Git executable: - git

1. “SonarQube Scanner installations”

‘SonarQube Scanner’

Name: - Sonar-Scanner

‘Install automatically’ check

Install from Maven central: -

Version: - SonarQube Scanner 5.0.1.3006

1. “Maven installation”

‘Maven’

Name: - maven3

‘Install automatically’ check

‘Install from Apache’

Version: - 3.6.3

1. “Dependency-Check installations”

‘Dependency-Check’

Name: - DC

‘Install automatically’ check

‘Install from github.com’

Version: - dependency-check 6.5.1

1. “Docker installations”

‘Docker’

Name: - docker

‘Install automatically’ check

‘Download from docker.com’

Docker version: - latest

Then “Save”

**Configure Docker to Jenkins Credentials: -**

Goto “Jenkins Dashboard”

‘Manage Jenkins’

‘Credentials’

‘Domains’ clk (global)

‘Global credentials’ clk ‘add Credentials’

‘New credentials’

‘Kind’

‘Scope’

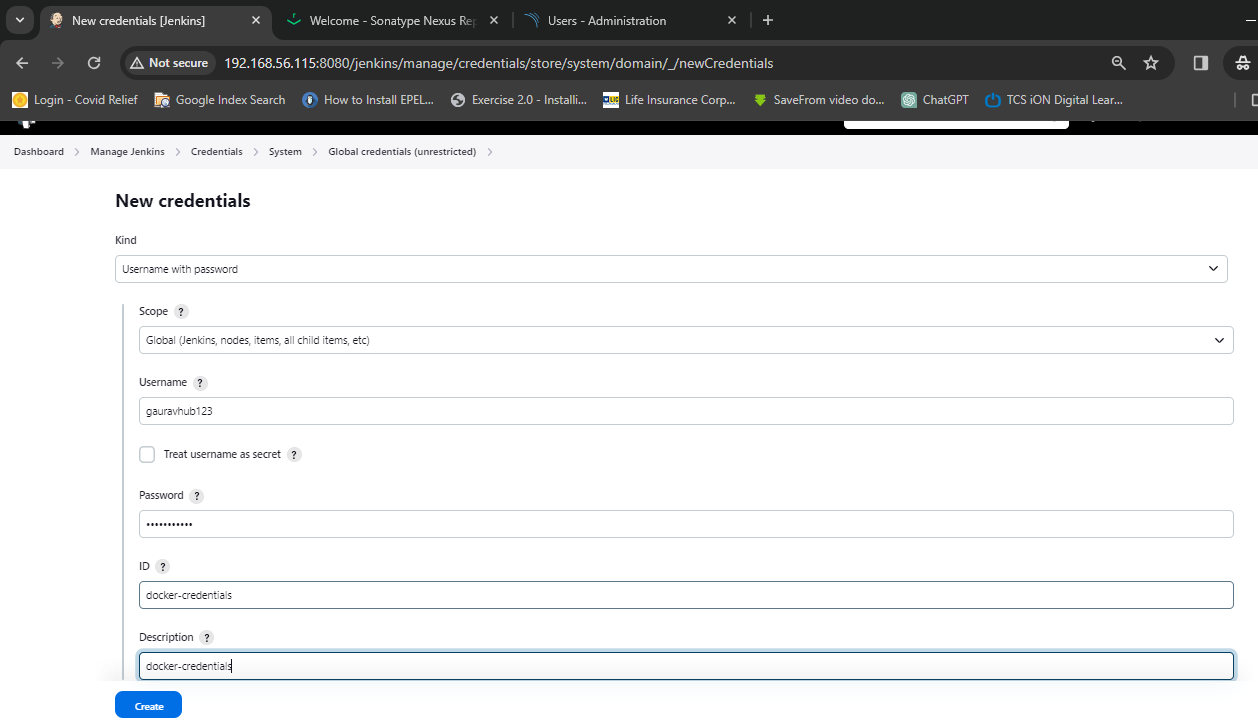
Username: - gauravhub123

Password:- Redhat@1991

ID: - docker-credential

Description: - docker-credential

“Create” clk



**Configure SonarQube Server to connected: -**

Goto “Dashboard”

‘Manage Jenkins’

‘System’

‘SonarQube servers’

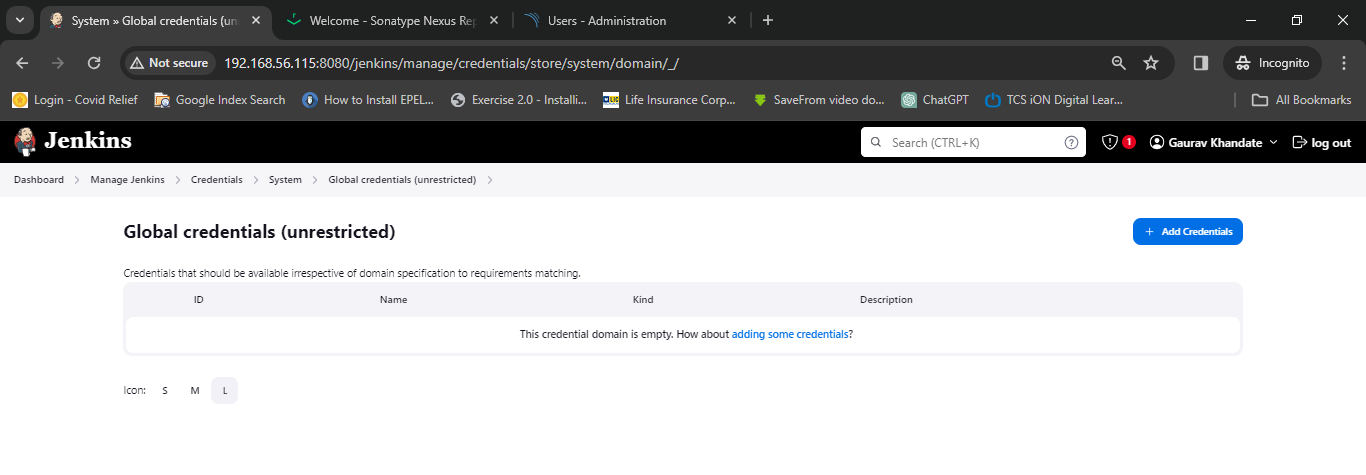
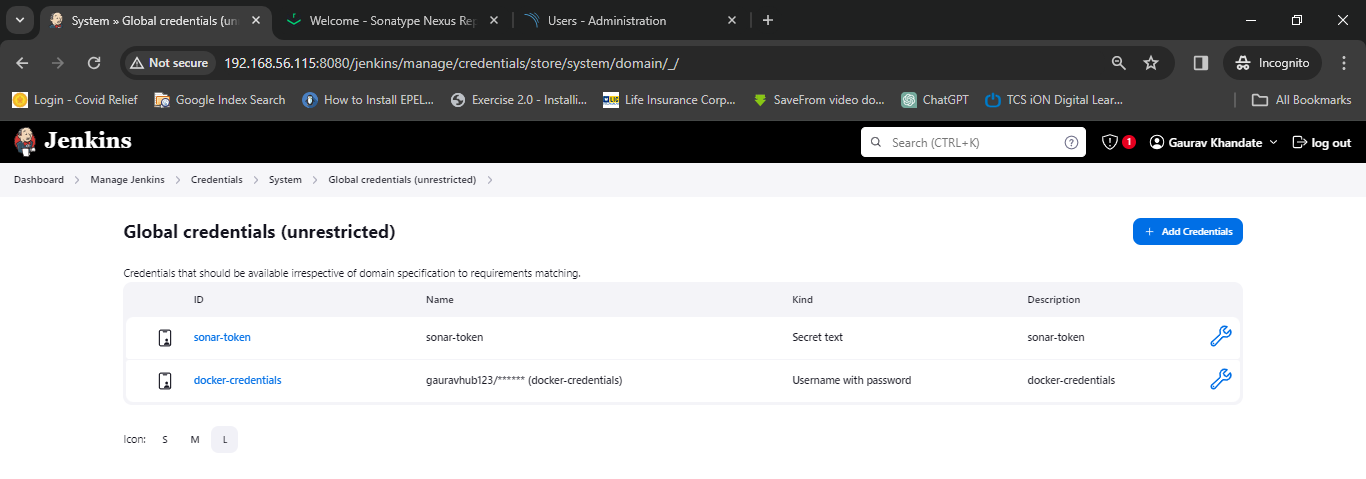
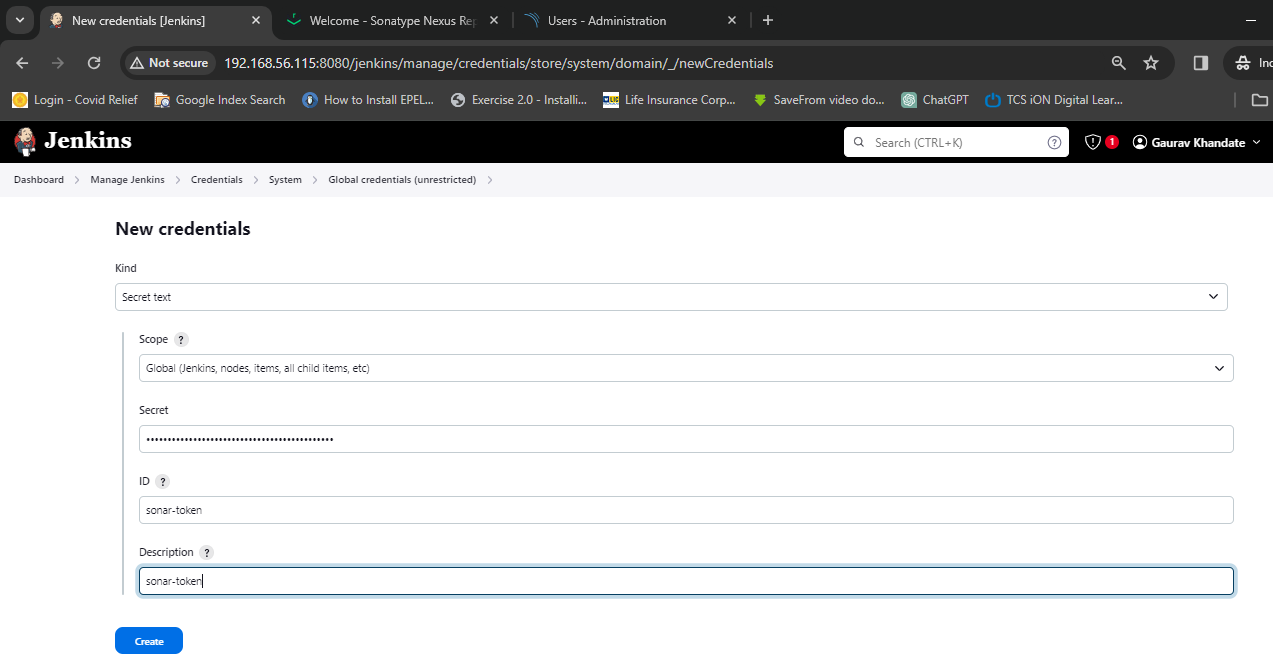
‘Add SonarQube’

Name: - sonarqube

Server URL: - <http://192.168.56.116:9000>

Server authentication token: - sonar-toke

“Apply and save “ clk

**Configure Nexus for Manage Plugins Jenkins: -**

Goto “Dashboard”

“Manage Jenkins”

“Plugins” ----- ------ “Available Plugins”

|

|

‘Config File Provider’

Go back and manage Jenkins on ‘Managed files’ option are show

**Configure to Nexus server to Jenkins: -**

Goto “Dashboard”

“Manage Jenkins”

‘Managed files’

Add a new Config: -

New configuration: - Global Maven settings.xml

ID: - global-maven

Next

Edit configuration File: -

Server credentials

‘content’ add on this line

<server>

<id> maven-releases</id>

<username>admin </username>

<password>redhat</password>

</server>

server>

<id> maven-snapshots</id>

<username>admin </username>

<password>redhat</password>

</server>

“Submit” clk

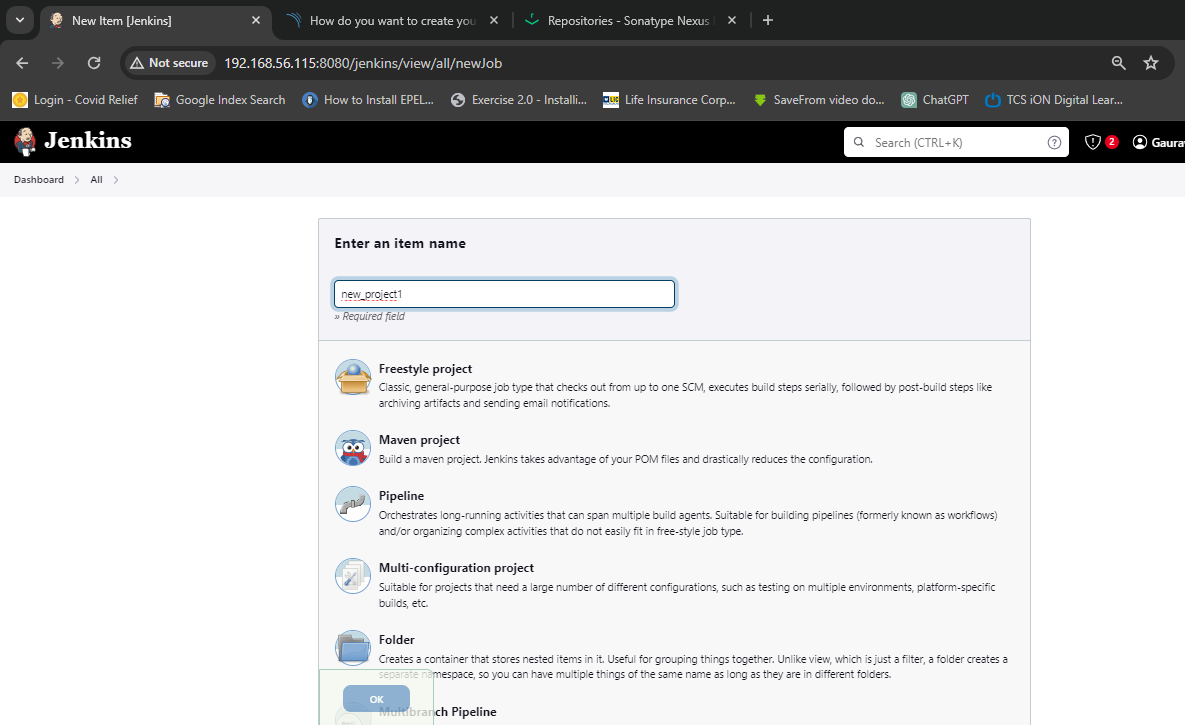
**Creating pipe line: -**

Goto “Dashboard”

Clk “new item”

Clk “Enter an item name to create pipeline project”

Clk “ok”



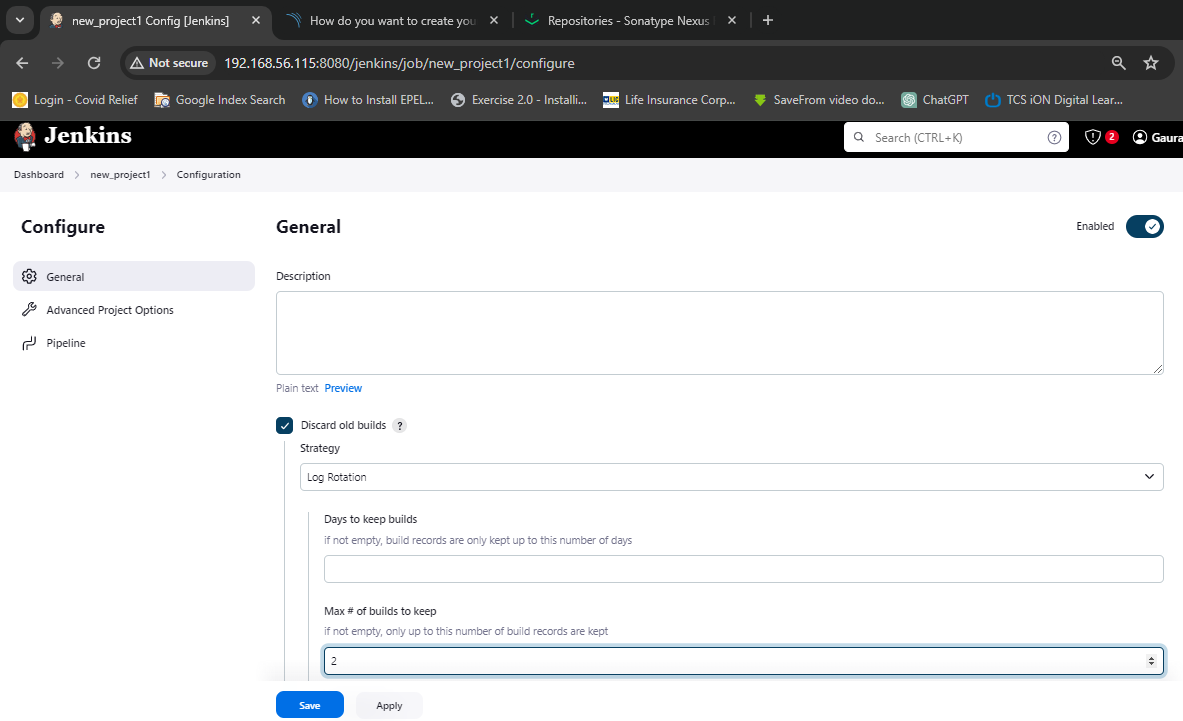
Goto “ new\_project and configure”

Clk “Discard old builds”

“Strategy”

“Days to keep builds ‘100’”

“Max # of builds to keep ‘2’”



“ Create Pipeline”

“Definition ‘Pipeline script’”

‘Script’

pipeline {

agent any

tools {

maven 'maven3'

jdk 'jdk17'

}

environment {

SCANNER\_HOME= tool 'sonar-scanner'

}

stages {

stage('Git Checkout') {

steps {

git 'https://github.com/gauravkhandate/Project1.git'

}

}

stage('Compile') {

steps {

sh "mvn compile"

}

}

stage('Unit Test') {

steps {

sh "mvn test -DskipTests=true"

}

}

stage('SonarQube Analysis') {

steps {

withSonarQubeEnv('sonar') {

sh '''$SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=EKART -Dsonar.projectKey=EKART -Dsonar.java.binaries=. '''

}

}

}

stage('Hello') {

steps {

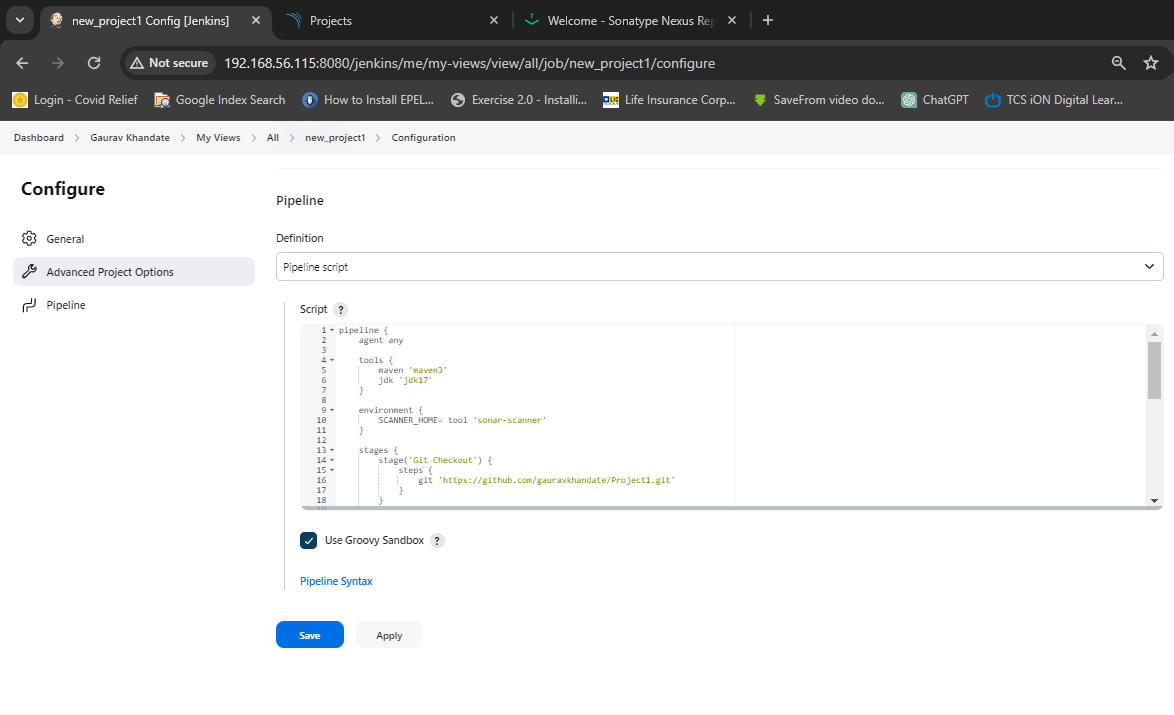
echo 'Hello World'

}

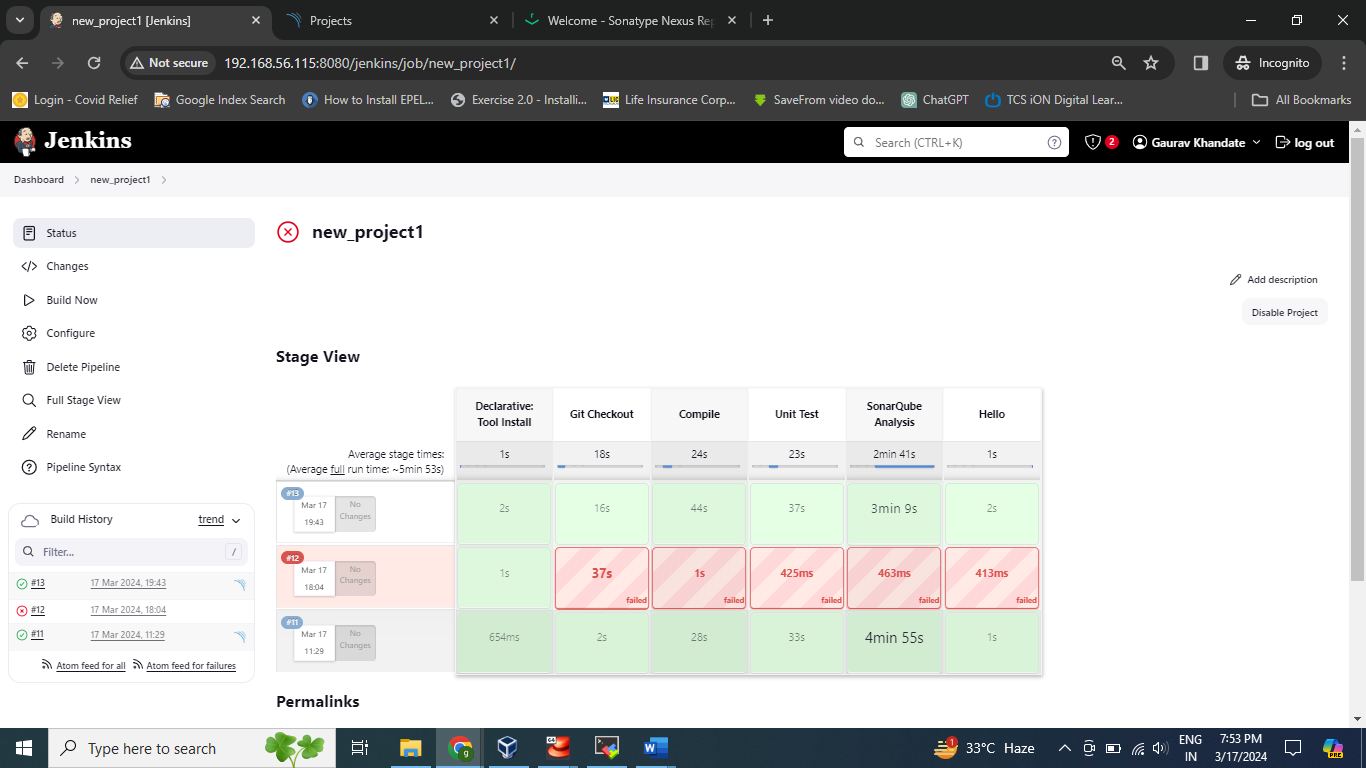
}

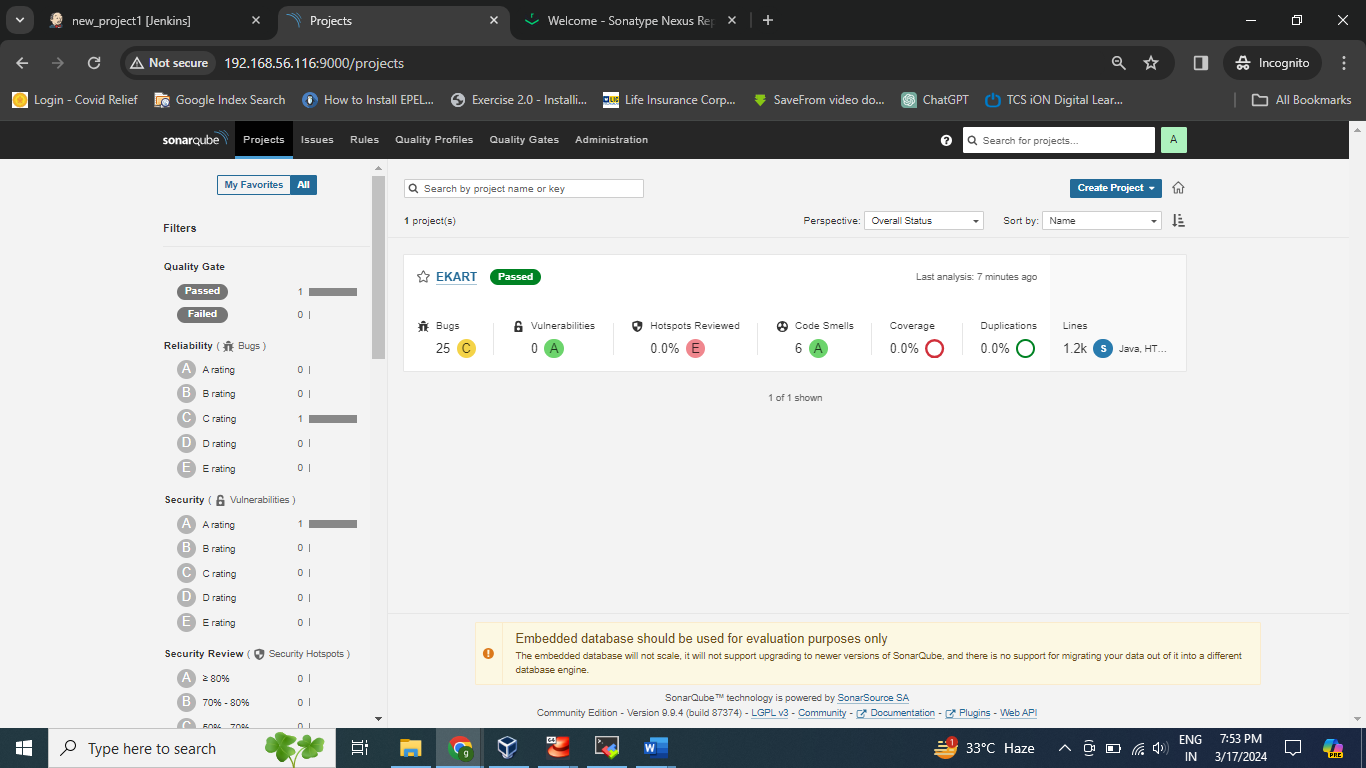
}

}



First project done





**Configure To Nexus Server**

stage('OWASP Dependency Check') {

steps {

dependencyCheck additionalArguments: ' --scan ./', odcInstallation: 'DC'

dependencyCheckPublisher pattern: '\*\*/dependency-check-report.xml'

}

}

stage('Build') {

steps {

sh "mvn package -DskipTests=true"

}

}

stage('Deploy To Nexus') {

steps {

withMaven(globalMavenSettingsConfig: 'global-maven', jdk: 'jdk17', maven: 'maven3', mavenSettingsConfig: '', traceability: true) {

sh "mvn deploy -DskipTests=true"

}

}

}

In This command to used to connected to nexus server

There are few dependency plugins install

“Pipeline Maven Integration” are installed