

Jenkins installation: using amz linux 2022

```
cd /mnt

yum install tomcat-native.x86_64 -y
amazon-linux-extras install java-openjdk11 -y

mkdir tools
cd tools
wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.73/bin/apache-tomcat-9.0.73.zip
unzip apache*tomcat*.zip

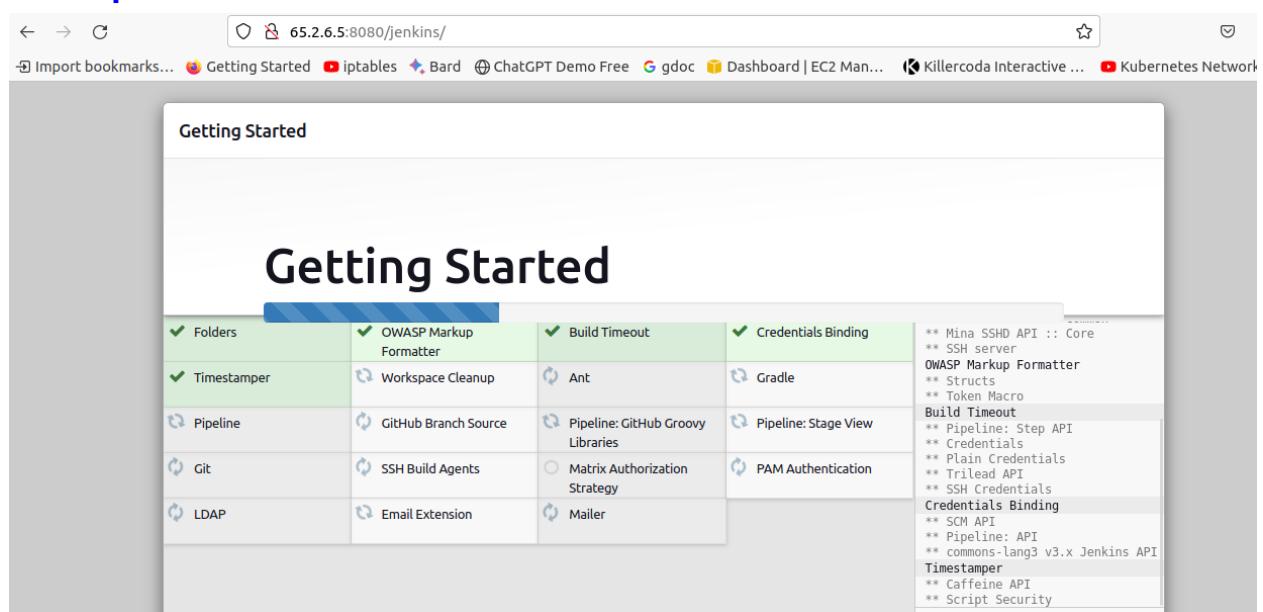
cd apache-tomcat-9.*/ 
cd webapps
wget https://get.jenkins.io/war-stable/2.375.1/jenkins.war

cd ../bin
chmod +x *.sh

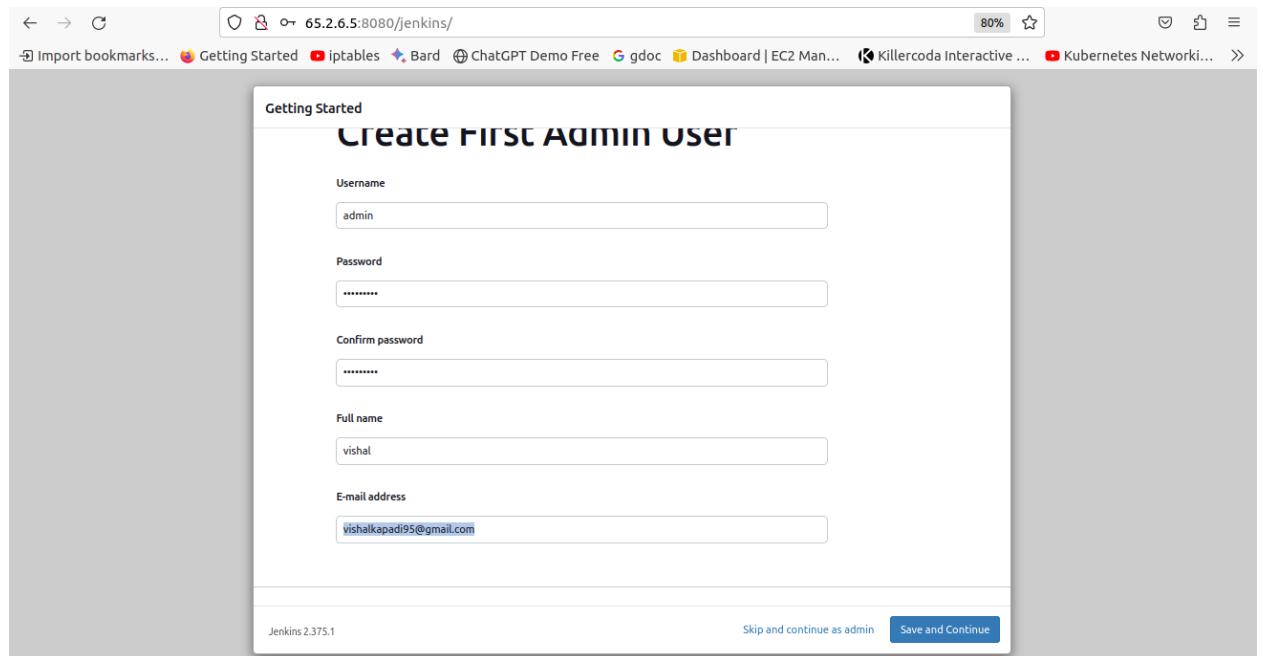
## start jenkins or not ##
read -p "Start Jenkins? (yes/no) " answer

if [ "$answer" == "yes" ]
then
    bash "startup.sh"
    sleep 3
    echo "Started Jenkins successfully."
else
    echo "Start Jenkins manually later on."
fi
#
```

- Access jenkins from url <http://public-ip:8080/jenkins/>  
In my case it is http://13.233.38.126:8080/jenkins/



The screenshot shows the Jenkins 'Getting Started' page. A sidebar on the left lists various Jenkins features with green checkmarks: Folders, OWASP Markup Formatter, Build Timeout, Credentials Binding, Timestamper, Workspace Cleanup, Ant, Gradle, Pipeline, GitHub Branch Source, Pipeline: GitHub Groovy Libraries, Pipeline: Stage View, Git, SSH Build Agents, Matrix Authorization Strategy, PAM Authentication, LDAP, Email Extension, and Mailer. A tooltip for 'Build Timeout' is open, listing sub-plugins: Mina SSHD API, SSH server, OWASP Markup Formatter, Structs, Token Macro, Pipeline: Step API, Credentials, Plain Credentials, Trilead API, SSH Credentials, SCM API, Pipeline: API, commons-lang3 v3.x Jenkins API, Caffeine API, and Script Security.



The screenshot shows the 'Create FIRST ADMIN USER' form. It has fields for Username (admin), Password (redacted), Confirm password (redacted), Full name (vishal), and E-mail address (vishalkapadi95@gmail.com). At the bottom, there are 'Skip and continue as admin' and 'Save and Continue' buttons. A note at the bottom left says 'Jenkins 2.375.1'.

# Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

[t.me/vishalk17](https://t.me/vishalk17)

[github.com/vishalk17](https://github.com/vishalk17)

The screenshot shows the Jenkins dashboard at <http://65.2.6.5:8080/jenkins/>. The left sidebar includes links for 'New item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main area features a 'Welcome to Jenkins!' message and a 'Start building your software project' section with links to 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. On the far right, there's an 'Add description' button.

## Installing apache-maven

```
mkdir -p /mnt/tools
cd /mnt/tools
wget https://dlcdn.apache.org/maven/maven-3/3.9.3/binaries/apache-maven-3.9.3-bin.zip
unzip apache-maven-*.zip

# Add Maven to PATH
echo 'export PATH="/mnt/tools/apache-maven-3.9.3/bin:$PATH"' >> ~/.bash_profile
source ~/.bash_profile

echo " Maven installation completed"
```

**Installing sonar-scanner :**

```
# Update package manager  
sudo yum update -y  
  
# Install OpenJDK 11  
sudo amazon-linux-extras install java-openjdk11 -y  
  
# Create tools directory  
sudo mkdir -p /mnt/tools  
  
# Download SonarScanner  
wget -P /mnt/tools  
https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.6.2.2472-linux  
.zip  
  
cd /mnt/tools/  
unzip sonar-scanner-cli-*  
  
  
cd sonar-scanner-cli-*  
  
# Add SonarScanner to PATH  
echo 'export PATH="/mnt/tools/sonar-scanner-4.6.2.2472-linux/bin:$PATH"' >> ~/.bash_profile  
source ~/.bash_profile
```

### Sonarqube Installation :

- Let we use container for it

vi docker-compose.yml :

```
version: '3'
services:
  sonarqube:
    image: sonarqube
    ports:
      - 8090:9000
    networks:
      - sonarnet
    environment:
      - SONARQUBE_JDBC_URL=jdbc:h2:tcp://sonarqube-db:9092/sonar
      - SONARQUBE_JDBC_USERNAME=sonar
      - SONARQUBE_JDBC_PASSWORD=sonar
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs

networks:
  sonarnet:

volumes:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
```

Access SonarQube Web UI: Open a web browser and navigate to `http://public-ip:9000`. This will bring up the SonarQube Web UI.

In my case , as per my Dockercompose file it is <http://13.233.38.126:8090>

- Log in to SonarQube: Use the default credentials to log in. The default username is "admin" and the default password is "admin". You will be prompted to change the password after the first login.

## Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

t.me/vishalk17

github.com/vishalk17

The screenshot shows the SonarQube web interface at the URL <http://65.2.6.5:8090/projects/create>. At the top, there are links for 'Import bookmarks...', 'Getting Started', 'iptables', 'Bard', 'ChatGPT Demo Free', 'gdoc', 'Dashboard | EC2 Man...', 'Killercoda Interactive ...', 'Kubernetes Networki...', and more. Below the header, the SonarQube logo and navigation menu (Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, More, Search) are visible. A prominent heading asks 'How do you want to create your project?'. It provides five options for importing from external platforms: 'From Azure DevOps', 'From Bitbucket Server', 'From Bitbucket Cloud', 'From GitHub', and 'From GitLab', each with a corresponding icon and a 'Set up global configuration' link. Below these, a section titled 'Are you just testing or have an advanced use-case? Create a project manually.' features a large text input field with a double-angle bracket icon and a 'Manually' button.

- Create a new project: Once logged in, click on the "manually Create new project" button on the SonarQube Web UI. Provide a unique project key, name, and display name for your project.

The screenshot shows the 'Create a project' form in the SonarQube web interface. The URL is <http://65.2.6.5:8090/projects/create?mode=manual>. The form fields are as follows:

- Project display name \***: Hello-World-app-test
- Project key \***: Hello-World-app-test
- Main branch name \***: main

Below the form, there is a note: 'The name of your project's default branch [Learn More](#)'. At the bottom right of the form is a 'Next' button.

# Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

t.me/vishalk17

github.com/vishalk17

The screenshot shows the SonarQube interface for setting up a new project. At the top, there's a navigation bar with links like 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', 'More', and a search bar. Below the navigation, a section titled 'Set up project for Clean as You Code' provides instructions on how to focus on recent changes. It includes a link to 'Defining New Code'. A question 'What should be the baseline for new code for this project?' has three options: 'Use the global setting' (selected), 'Previous version', and 'Define a specific setting for this project'. The 'Previous version' option is described as recommended for regular releases. The 'Define a specific setting for this project' section contains three sub-options: 'Previous version', 'Number of days', and 'Reference branch'. Each sub-option has its own description and recommendation. At the bottom, there are 'Back' and 'Create project' buttons.

The screenshot shows the SonarQube dashboard for the 'Hello-World-app-test' project. The top navigation bar is identical to the previous screenshot. The main content area displays the project name 'Hello-World-app-test' and a dropdown menu showing 'main'. Below this, there are tabs for 'Overview', 'Issues', 'Security Hotspots', 'Measures', 'Code', and 'Activity'. On the right, there are 'Project Settings' and 'Project Information' dropdowns. The 'Overview' tab is selected, showing a summary of the repository analysis. It includes sections for 'How do you want to analyze your repository?' and 'Do you want to integrate with your favorite CI? Choose one of the following tutorials.' There are six integration tutorial cards: 'With Jenkins', 'With GitHub Actions', 'With Bitbucket', 'With GitLab CI', 'With Azure Pipelines', and 'With CircleCI'.

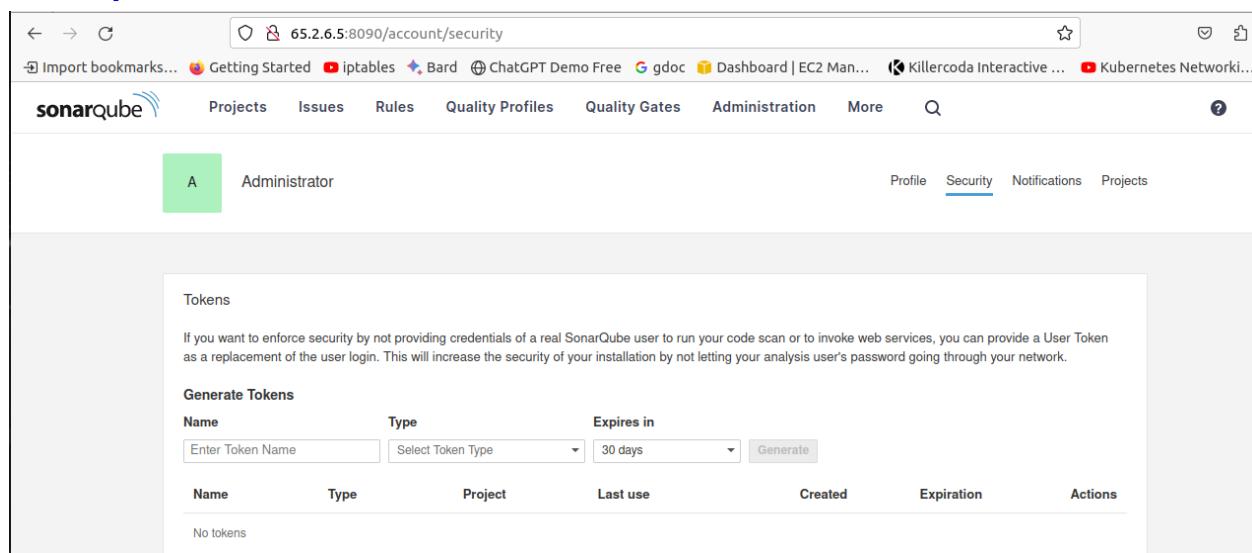
- Generate an authentication token: Go to "My Account" -> "Security" -> "Tokens" and generate a new authentication token. Make sure to copy the generated token as you will need it later for the analysis.

The screenshot shows the SonarQube dashboard for the 'Hello-World-app-test' project. The top navigation bar and project details are the same as the previous screenshots. The 'Overview' tab is selected. A dropdown menu is open on the right side, showing options: 'Administrator', 'My Account' (which is currently selected), and 'Log out'. The 'My Account' option is highlighted with a blue background.

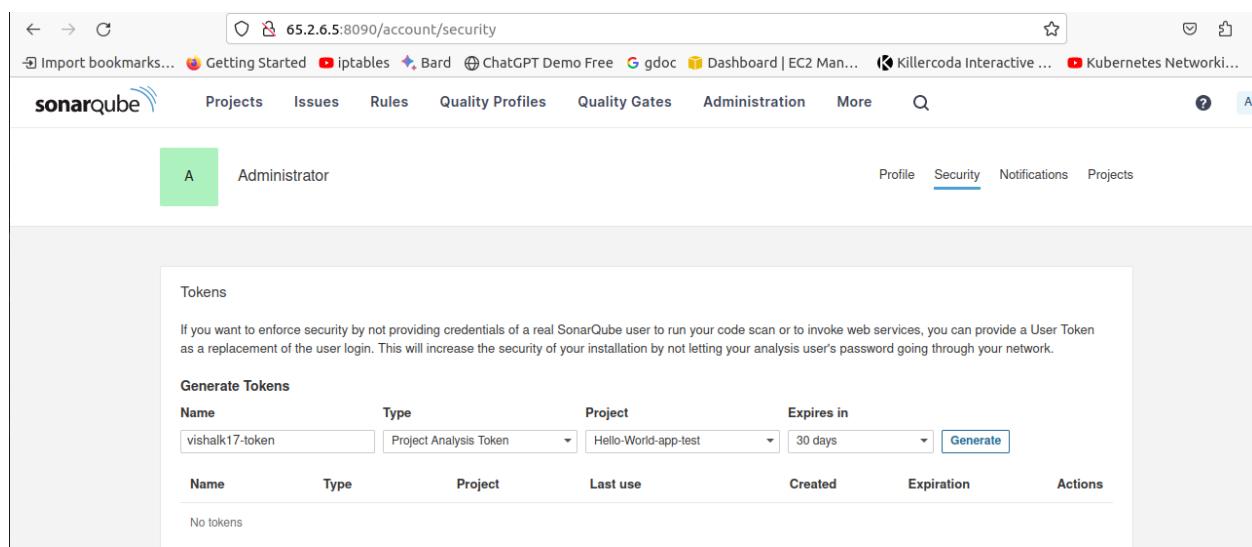
# Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

 t.me/vishalk17

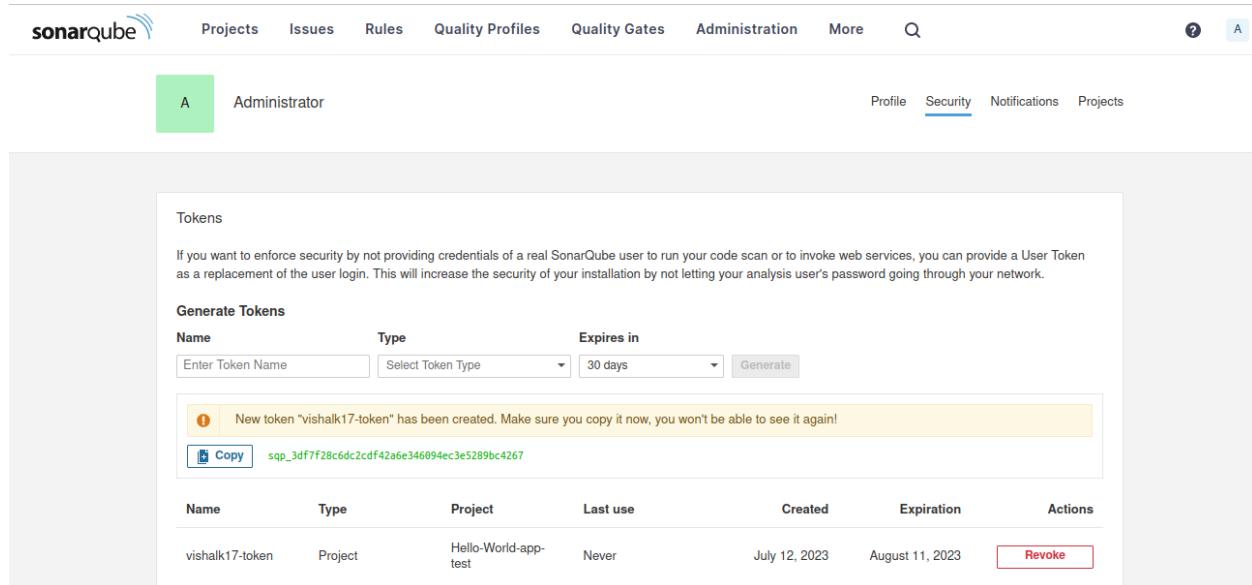
 github.com/vishalk17



The screenshot shows the SonarQube security interface. At the top, there's a navigation bar with links like 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', 'More', and a search bar. Below this is a user profile section with a green 'A' icon and the name 'Administrator'. A 'Profile' tab is selected. The main content area is titled 'Tokens'. It contains a sub-section 'Generate Tokens' with fields for 'Name' (text input), 'Type' (dropdown), and 'Expires in' (dropdown set to 30 days). A 'Generate' button is present. Below this is a table header for tokens with columns: Name, Type, Project, Last use, Created, Expiration, and Actions. The table body shows a single row: 'No tokens'.



This screenshot shows the same SonarQube security interface as the previous one, but with more detailed information in the 'Generate Tokens' form. The 'Name' field is filled with 'vishalk17-token', 'Type' is 'Project Analysis Token', 'Project' is 'Hello-World-app-test', and 'Expires in' is '30 days'. The 'Generate' button is visible. The table below shows a single token entry: 'vishalk17-token' (Type: Project, Project: Hello-World-app-test, Last use: Never, Created: July 12, 2023, Expiration: August 11, 2023).



This screenshot shows the SonarQube security interface after a token has been generated. A yellow message box at the top of the token table area says: 'New token "vishalk17-token" has been created. Make sure you copy it now, you won't be able to see it again!'. Below this message, there is a 'Copy' button next to the token ID 'sqp\_3df7f28c6dc2cdf42a6e346094ec3e5289bc4267'. The token table shows the single entry: 'vishalk17-token' (Type: Project, Project: Hello-World-app-test, Last use: Never, Created: July 12, 2023, Expiration: August 11, 2023). A red 'Revoke' button is visible in the 'Actions' column for this token.

- Generated token is : sqp\_3df7f28c6dc2cdf42a6e346094ec3e5289bc4267
- 

Lets test and do the sonarqube analysis on github project I have created,  
So that we can integrated with jenkins to do ci cd later on

What we have:

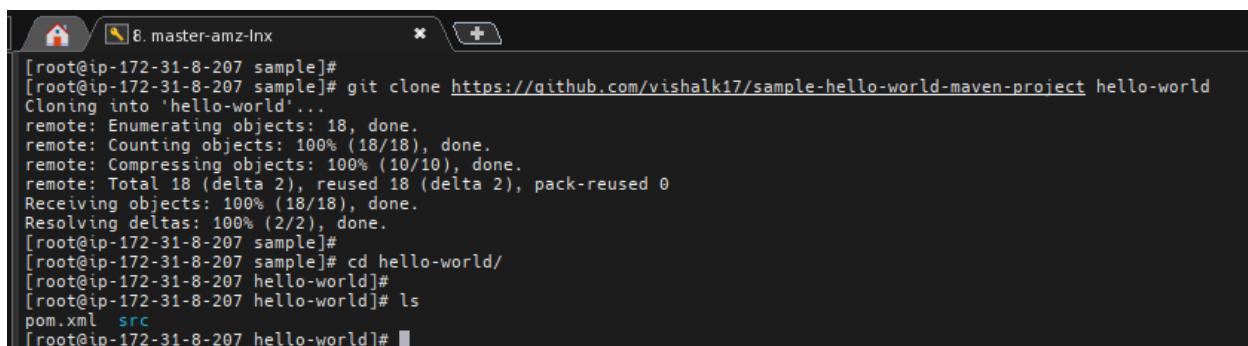
Sonarqube url as per container created: <http://13.233.38.126:8090>

Project key from sonarqube : Hello-World-app-test

Auth token: sqp\_3df7f28c6dc2cdf42a6e346094ec3e5289bc4267

- Clone git url to the location

Git url : <https://github.com/vishalk17/sample-hello-world-maven-project>



```
[root@ip-172-31-8-207 sample]# git clone https://github.com/vishalk17/sample-hello-world-maven-project hello-world
Cloning into 'hello-world'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 18 (delta 2), reused 18 (delta 2), pack-reused 0
Receiving objects: 100% (18/18), done.
Resolving deltas: 100% (2/2), done.
[root@ip-172-31-8-207 sample]#
[root@ip-172-31-8-207 sample]# cd hello-world/
[root@ip-172-31-8-207 hello-world]#
[root@ip-172-31-8-207 hello-world]# ls
pom.xml  src
[root@ip-172-31-8-207 hello-world]#
```

- cd hello-world

Note : I have integrated sonarqube properties in pom.xml , so you only need to replace above things there

- Open the pom.xml

```
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <!-- SonarQube properties -->
    <sonar.host.url>http://localhost:9000</sonar.host.url>
    <sonar.login>YOUR SONARQUBE LOGIN TOKEN</sonar.login>
    <sonar.projectKey>my-webapp</sonar.projectKey>
    <sonar.projectName>My Web App</sonar.projectName>
    <sonar.projectVersion>1.0-SNAPSHOT</sonar.projectVersion>
    <sonar.language>java</sonar.language>
</properties>

<dependencies>
    <dependency>
```

- Replace sonar properties with things we have

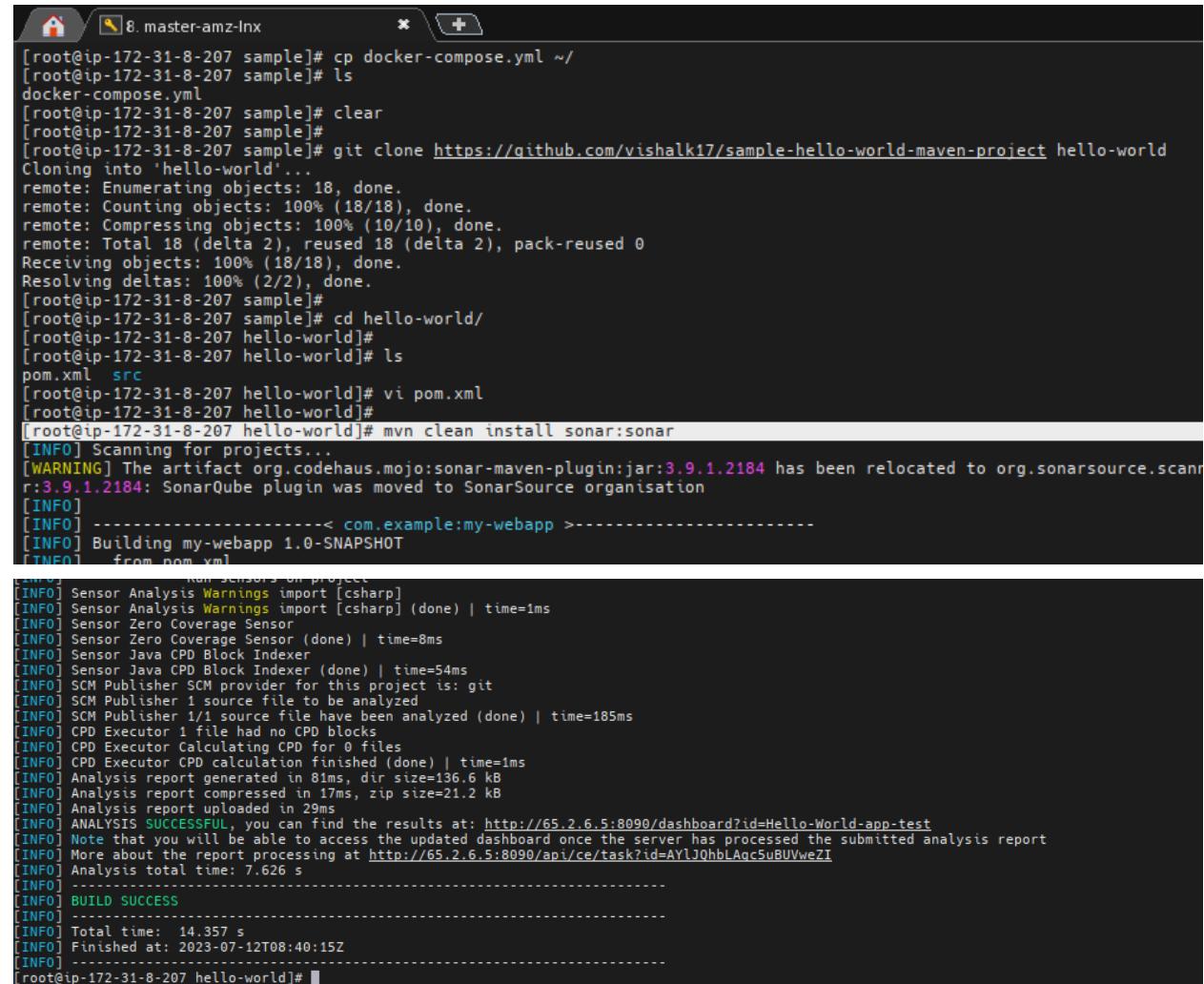
```
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <!-- SonarQube properties -->
    <sonar.host.url>http://localhost:8090</sonar.host.url>
    <sonar.login>sqp_3df7f28c6dc2cdf42a6e346094ec3e5289bc4267</sonar.login>
    <sonar.projectKey>Hello-World-app-test</sonar.projectKey>
    <sonar.projectName>Hello-World-app-test</sonar.projectName>
    <sonar.projectVersion>1.0-SNAPSHOT</sonar.projectVersion>
    <sonar.language>java</sonar.language>
</properties>

<dependencies>
```

- Save and exit

Lets have a analysis

- mvn clean install sonar:sonar



```
[root@ip-172-31-8-207 sample]# cp docker-compose.yml ~/
[root@ip-172-31-8-207 sample]# ls
docker-compose.yml
[root@ip-172-31-8-207 sample]# clear
[root@ip-172-31-8-207 sample]#
[root@ip-172-31-8-207 sample]# git clone https://github.com/vishalk17/sample-hello-world-maven-project hello-world
Cloning into 'hello-world'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 18 (delta 2), reused 18 (delta 2), pack-reused 0
Receiving objects: 100% (18/18), done.
Resolving deltas: 100% (2/2), done.
[root@ip-172-31-8-207 sample]#
[root@ip-172-31-8-207 sample]# cd hello-world/
[root@ip-172-31-8-207 hello-world]#
[root@ip-172-31-8-207 hello-world]# ls
pom.xml  src
[root@ip-172-31-8-207 hello-world]# vi pom.xml
[root@ip-172-31-8-207 hello-world]#
[root@ip-172-31-8-207 hello-world]# mvn clean install sonar:sonar
[INFO] Scanning for projects...
[WARNING] The artifact org.codehaus.mojo:sonar-maven-plugin:jar:3.9.1.2184 has been relocated to org.sonarsource.scanner:3.9.1.2184: SonarQube plugin was moved to SonarSource organisation
[INFO]
[INFO] -----< com.example:my-webapp >-----
[INFO] Building my-webapp 1.0-SNAPSHOT
[INFO]   from pom.xml

[INFO]   Run Sensors on project
[INFO] Sensor Analysis Warnings import [csharp]
[INFO] Sensor Analysis Warnings import [csharp] (done) | time=1ms
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=8ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=54ms
[INFO] SCM Publisher SCM provider for this project is: git
[INFO] SCM Publisher 1 source file to be analyzed
[INFO] SCM Publisher 1/1 source file have been analyzed (done) | time=185ms
[INFO] CPD Executor 1 file had no CPD blocks
[INFO] CPD Executor Calculating CPD for 0 files
[INFO] CPD Executor CPD calculation finished (done) | time=1ms
[INFO] Analysis report generated in 81ms, dir size=136.6 kB
[INFO] Analysis report compressed in 17ms, zip size=21.2 kB
[INFO] Analysis report uploaded in 29ms
[INFO] ANALYSIS SUCCESSFUL, you can find the results at: http://65.2.6.5:8090/dashboard?id=Hello-World-app-test
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://65.2.6.5:8090/api/ce/task?id=AYlJ0hbLAqcSuBUVweZI
[INFO] Analysis total time: 7.626 s
[INFO]
[INFO] -----< com.example:my-webapp >-----
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 14.357 s
[INFO] Finished at: 2023-07-12T08:40:15Z
[INFO] -----< com.example:my-webapp >-----
```

# Ci-cd-git-jenkins-nexus- sonarqube-mvn-ansible

Lets have a look on sonarqube dashboard

[t.me/vishalk17](https://t.me/vishalk17)

[github.com/vishalk17](https://github.com/vishalk17)

The screenshot shows the SonarQube dashboard for the 'Hello-World-app-test' project. At the top, there's a banner indicating 'The last analysis has warnings' with a 'See details' link. Below the banner, the 'Overview' tab is selected. On the left, a 'Quality Gate Status' section shows a green checkmark and the word 'Passed'. A message below says 'Enjoy your sparkling clean code!'. To the right, under 'Measures', there are six cards: 'Reliability' (0 New Bugs), 'Maintainability' (0 New Code Smells), 'Security' (0 New Vulnerabilities), 'Security Review' (0 New Security Hotspots), 'Coverage' (Coverage on 0 New Lines to cover), and 'Duplications' (0.0% Duplications). Below the measures, there's a 'Coverage' chart showing one issue at 2:10 PM on July 12, 2023. The chart includes a legend for Bugs, Code Smells, Vulnerabilities, and New Code.

- Everything seems ok

After compilation , our final binary file created inside target dir.

```
[root@ip-172-31-8-207 hello-world]# ls
pom.xml  src  target
[root@ip-172-31-8-207 hello-world]# cd target
[root@ip-172-31-8-207 target]# ls
classes  generated-sources  maven-archiver  maven-status  my-webapp-1.0-SNAPSHOT  my-webapp-1.0-SNAPSHOT.war  sonar
[root@ip-172-31-8-207 target]#
```

### Setup nexus artifact.

vi docker-compose-nexus.yml

```
version: '3'
services:
  nexus:
    image: sonatype/nexus3:latest
    container_name: nexus-instance
    ports:
      - 8092:8081
    volumes:
      - nexus-data:/nexus-data

  volumes:
    nexus-data:
```

docker-compose -f docker-compose-nexus.yml up -d

```
[root@ip-172-31-8-207 nexus]# docker-compose ls
NAME          STATUS        CONFIG FILES
nexus         running(1)   /mnt/sample/nexus/docker-compose-nexus.yml
sample        running(1)   /mnt/sample/docker-compose.yml
[root@ip-172-31-8-207 nexus]# docker ps
CONTAINER ID IMAGE           COMMAND            CREATED          STATUS          PORTS          NAMES
7253a4fd5056  sonatype/nexus3:latest  "/opt/sonatype/nexus..."  14 minutes ago  Up 2 minutes  0.0.0.0:8092->8081/tcp, :::8092->8081/tcp  nexus-instance
8e220320c2bf  sonarqube        "/opt/sonarqube/dock..."  3 hours ago   Up 3 hours   0.0.0.0:8090->9000/tcp, :::8090->9000/tcp  sample-sonarqube-1
```

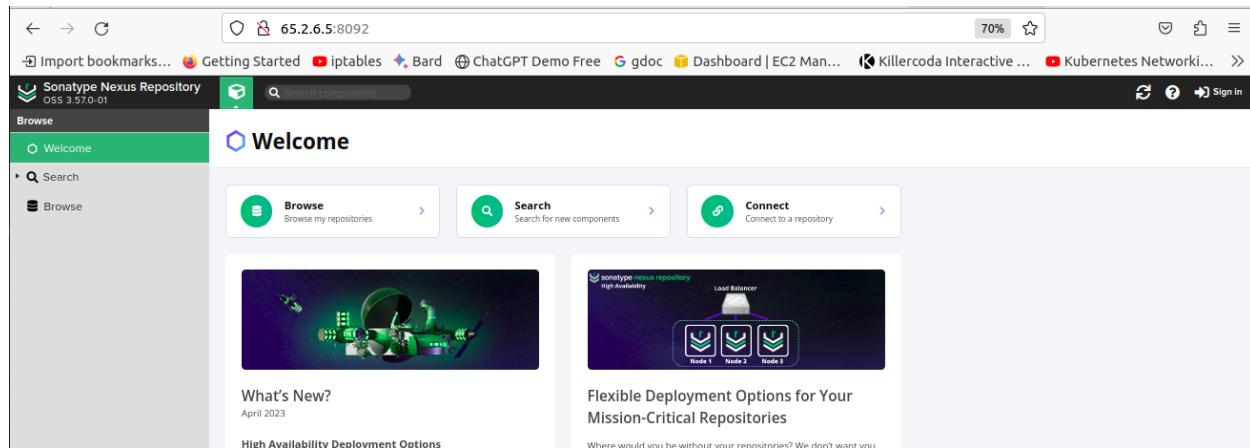
## Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

Lets access nexus url

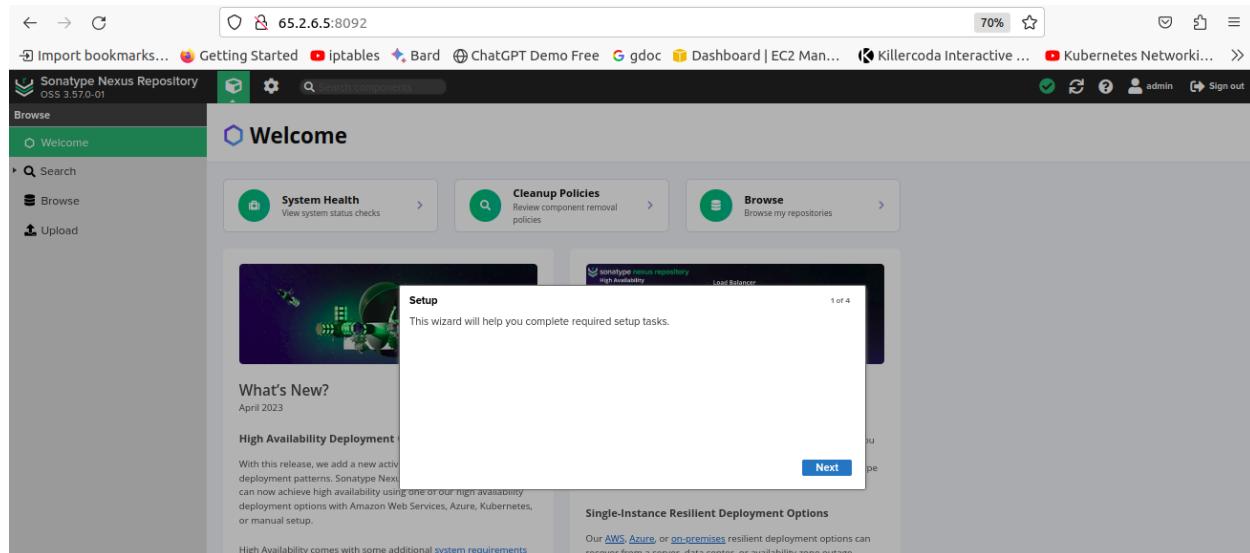
 t.me/vishalk17

 github.com/vishalk17

As per docker-compose file my url is : <http://13.233.38.126:8092/>



- Click on sign in
- Click username as a admin
- Password you will get from /nexus-data/admin-password  
`cat /nexus-data/admin-password`



- Setup admin acc

## Create a Maven Repository:

- Click on the "Settings" cogwheel icon in the left sidebar.
- Under the "Repositories" section, click on "Repositories".
- Click on the "Create Repository" button.
- Select "Maven2 (hosted)" as the recipe.
- Fill in the required information, such as the repository name and storage location.
- Click on the "Create Repository" button to create the Maven repository.

The screenshot shows the Sonatype Nexus Repository interface. The left sidebar has a green header 'Administration' and a 'Repository' section with sub-options: Repositories, Blob Stores, Proprietary Repositories, Content Selectors, and Cleanup Policies. The main content area is titled 'Repository' and contains tabs for Blob Stores, Cleanup Policies, Proprietary Repositories, Repositories, Routing Rules, and Content Selectors. The 'Repositories' tab is active.

This screenshot shows the 'Repositories' management page. The left sidebar shows the 'Repository' section with 'Repositories' selected. The main table lists existing repositories with columns: Name, Type, Format, Blob Store, Status, URL, Health check, and Firewall Re... . The repositories listed are maven-central, maven-public, maven-releases, maven-snapshots, nuget-group, nuget-hosted, and nuget.org-proxy.

Name ↑	Type	Format	Blob Store	Status	URL	Health check	Firewall Re...
maven-central	proxy	maven2	default	Online - Ready to Connect	<a href="#">copy</a>	Analyze	<a href="#">...</a>
maven-public	group	maven2	default	Online	<a href="#">copy</a>	<a href="#">...</a>	<a href="#">...</a>
maven-releases	hosted	maven2	default	Online	<a href="#">copy</a>	<a href="#">...</a>	<a href="#">...</a>
maven-snapshots	hosted	maven2	default	Online	<a href="#">copy</a>	<a href="#">...</a>	<a href="#">...</a>
nuget-group	group	nuget	default	Online	<a href="#">copy</a>	<a href="#">...</a>	<a href="#">...</a>
nuget-hosted	hosted	nuget	default	Online	<a href="#">copy</a>	<a href="#">...</a>	<a href="#">...</a>
nuget.org-proxy	proxy	nuget	default	Online - Ready to Connect	<a href="#">copy</a>	Analyze	<a href="#">...</a>

This screenshot shows the 'Create repository' dialog box. The left sidebar shows the 'Repository' section with 'Repositories' selected. The dialog box has fields for 'Name' (maven-central), 'Type' (proxy), 'Format' (maven2), and 'Blob Store' (default). Below the dialog, a table lists the created repositories: maven-central, maven-public, maven-releases, maven-snapshots, nuget-group, nuget-hosted, and nuget.ora-proxy.

Name ↑	Type	Format	Blob Store
maven-central	proxy	maven2	default
maven-public	group	maven2	default
maven-releases	hosted	maven2	default
maven-snapshots	hosted	maven2	default
nuget-group	group	nuget	default
nuget-hosted	hosted	nuget	default
nuget.ora-proxy	proxy	nuget	default

## Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

[t.me/vishalk17](https://t.me/vishalk17)

[github.com/vishalk17](https://github.com/vishalk17)

The screenshot shows the Sonatype Nexus Repository interface. The left sidebar is titled 'Administration' and includes sections for Repository, Security, and Roles. Under 'Repository', 'Repositories' is selected, showing a list of existing repositories: Recipe ↑, go (proxy), helm (hosted), helm (proxy), maven2 (group), maven2 (hosted), maven2 (proxy), npm (group), npm (hosted), npm (proxy), nuget (group), nuget (hosted), nuget (proxy), p2 (proxy), and pip (group). The 'maven2 (hosted)' repository is highlighted.

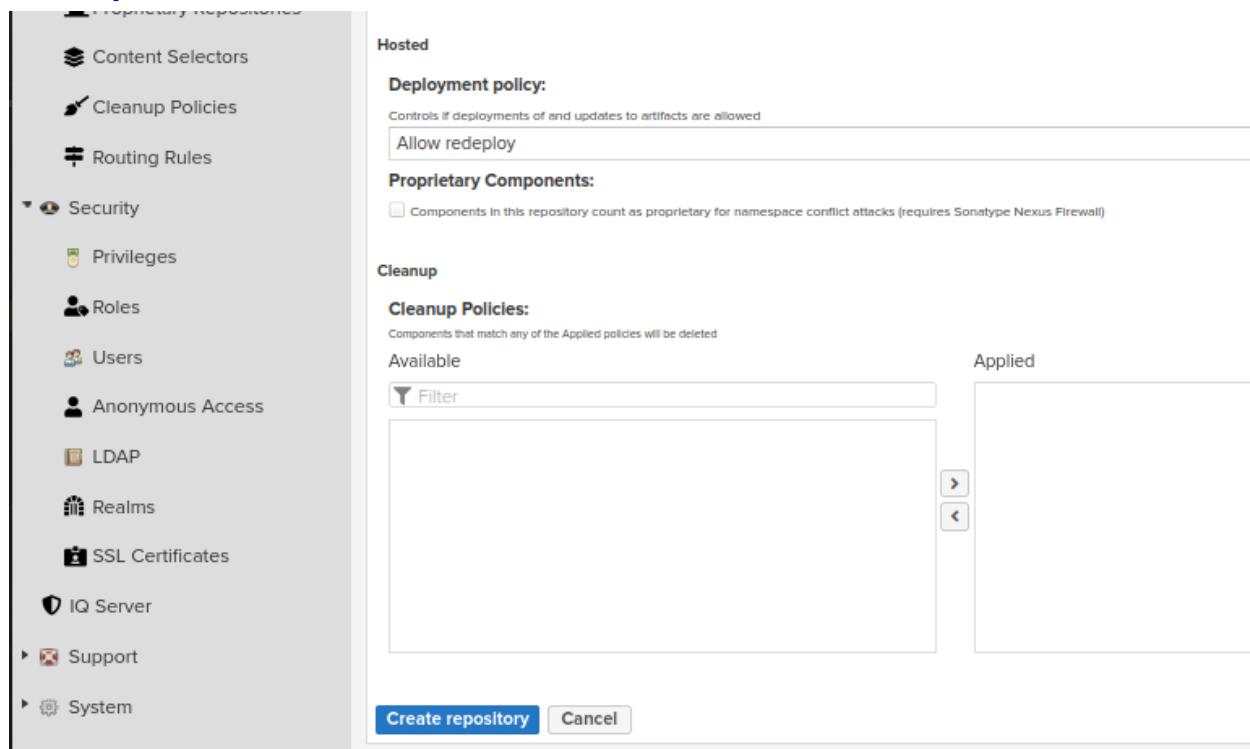
- Give name of the repository: [hello-world-pkg](#)

The screenshot shows the 'Create Repository' dialog for 'maven2 (hosted)'. The 'Name:' field is set to 'hello-world-pkg'. The 'Online:' checkbox is checked. Under 'Maven 2', the 'Version policy:' is set to 'Release' and the 'Layout policy:' is 'Strict'. In the 'Content Disposition:' section, 'Inline' is selected. The 'Storage' section shows 'Blob store:' set to 'default'. Under 'Strict Content Type Validation:', there is an unchecked checkbox for validating MIME types.

# Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

 t.me/vishalk17

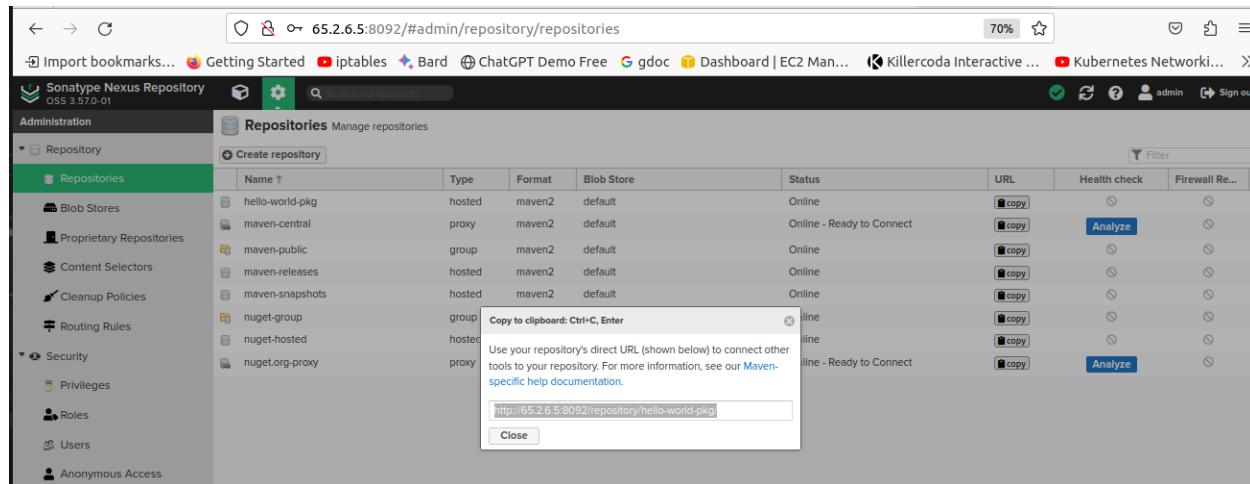
 github.com/vishalk17



The screenshot shows the 'Create repository' dialog in the Sonatype Nexus Repository Manager. The left sidebar lists various configuration sections: Content Selectors, Cleanup Policies, Routing Rules, Security (Privileges, Roles, Users, Anonymous Access, LDAP, Realms, SSL Certificates), IQ Server, Support, and System. The main area is titled 'Hosted' and contains sections for 'Deployment policy:' (Allow redeploy checked) and 'Proprietary Components:' (checkbox for namespace conflict attacks). Below this is the 'Cleanup' section with 'Cleanup Policies:' (Available and Applied tabs, empty lists). At the bottom are 'Create repository' and 'Cancel' buttons.

- get url from repository

In our case it is <http://13.233.38.126:8092/repository/hello-world-pkg/>



The screenshot shows the 'Repositories' list in the Sonatype Nexus Repository Manager. The left sidebar is identical to the previous screenshot. The main table lists repositories: hello-world-pkg (hosted, maven2, default, Online, URL: http://65.2.6.5:8092/repository/hello-world-pkg), maven-central (proxy, maven2, default, Online - Ready to Connect, URL: http://65.2.6.5:8092/repository/maven-central), maven-public (group, maven2, default, Online, URL: http://65.2.6.5:8092/repository/maven-public), maven-releases (hosted, maven2, default, Online, URL: http://65.2.6.5:8092/repository/maven-releases), maven-snapshots (hosted, maven2, default, Online, URL: http://65.2.6.5:8092/repository/maven-snapshots), nuget-group (group, proxy, offline, URL: http://65.2.6.5:8092/repository/nuget-group), nuget-hosted (hosted, proxy, offline, URL: http://65.2.6.5:8092/repository/nuget-hosted), and nuget.org-proxy (proxy, offline, URL: http://65.2.6.5:8092/repository/nuget.org-proxy). A context menu is open over the 'hello-world-pkg' row, with options 'Copy to clipboard: Ctrl+C, Enter' and 'Use your repository's direct URL (shown below) to connect other tools to your repository. For more information, see our Maven-specific help documentation.' Below the menu is a text input field containing the URL: http://65.2.6.5:8092/repository/hello-world-pkg/.

Create another repository for snap releases:

The screenshot shows the Sonatype Nexus Repository interface. The left sidebar is titled 'Administration' and includes sections for 'Repository' (selected), 'Blob Stores', 'Proprietary Repositories', 'Content Selectors', 'Cleanup Policies', 'Routing Rules', 'Security', and 'Privileges'. The main content area is titled 'Repositories' and shows a list of existing repositories: go (proxy), helm (hosted), helm (proxy), maven2 (group), maven2 (hosted) (which is selected), maven2 (proxy), npm (group), npm (hosted), npm (proxy), nuget (group), nuget (hosted), and nuget (proxy). A search bar at the top right says 'Search components'.

- select maven2 hosted  
Repo name: hello-world-pkg-snapshot  
Version policy : snapshot  
Layout policy : permissive  
Deployment policy : allow redeploy  
Create repo.

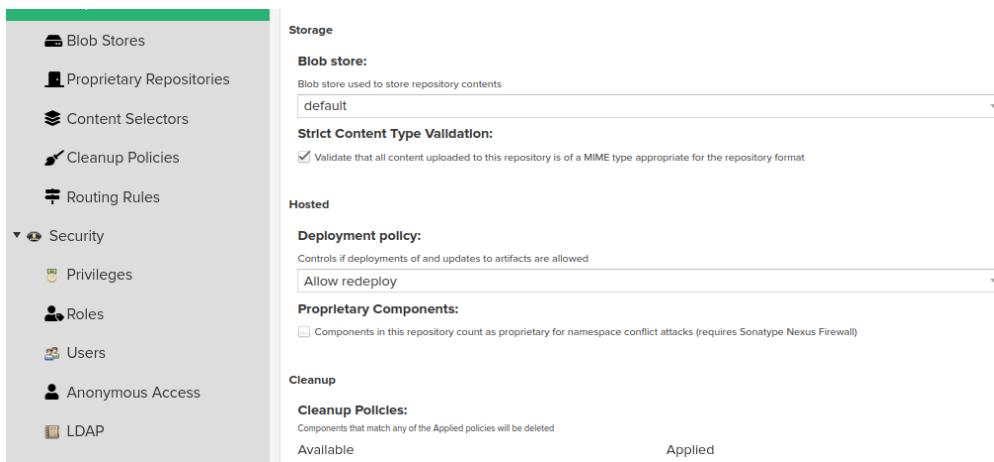
The screenshot shows the 'Create Repository' dialog for 'maven2 (hosted)'. The left sidebar is identical to the previous screenshot. The main form has the following fields:

- Name:** hello-world-pkg-snapshot
- Online:** checked
- Maven 2**
  - Version policy:** Snapshot
  - Layout policy:** Permissive
  - Content Disposition:** Inline

## Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

 t.me/vishalk17

 github.com/vishalk17



The screenshot shows the 'Storage' configuration page in the Nexus Repository Manager. On the left sidebar, under 'Security', the 'Privileges' section is selected. The main content area displays the 'Storage' configuration with the following settings:

- Blob store:** default
- Strict Content Type Validation:** checked (Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format)
- Deployment policy:** Controls if deployments of and updates to artifacts are allowed (Allow redeploy)
- Proprietary Components:** Components in this repository count as proprietary for namespace conflict attacks (requires Sonatype Nexus Firewall) (unchecked)
- Cleanup Policies:** Components that match any of the Applied policies will be deleted (Available)
- Applied:** Available

Ok now setup nexus integration in pom.xml so that . after build it will upload to the nexus repo.

- I have setup following things:

**Nexus username :** admin

**Nexus password :** v17041995

**Repo url for releases :** <http://13.233.38.126:8092/repository/hello-world-pkg/>

**Repo url for snapshot :** <http://13.233.38.126:8092/repository/hello-world-pkg/>

Add following things in pom.xml

```
<distributionManagement>

    <repository>
        <id>nexus-releases</id>
        <url>http://13.233.38.126:8092/repository/hello-world-pkg/</url>
    </repository>

    <snapshotRepository>
        <id>nexus-snapshots</id>
        <url>http://13.233.38.126:8092/repository/hello-world-pkg-snapshot/</url>
    </snapshotRepository>

</distributionManagement>
```

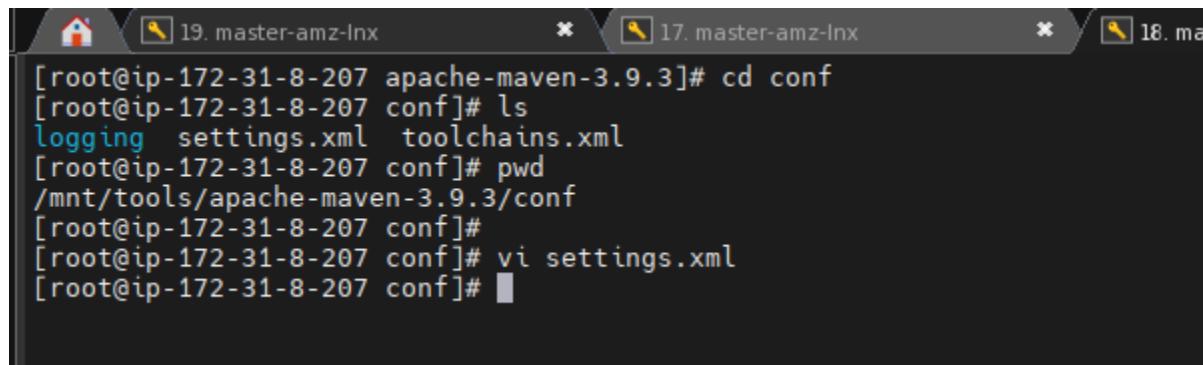
```
<distributionManagement>  
  
    <repository>  
        <id>nexus-releases</id>  
        <url>http://65.2.6.5:8092/repository/hello-world-pkg/</url>  
    </repository>  
  
    <snapshotRepository>  
        <id>nexus-snapshots</id>  
        <url>http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/</url>  
    </snapshotRepository>  
  
</distributionManagement>
```

- Save and exit

We have maven configs in `/mnt/tools/apache*maven*/conf`

- `vi settings.xml`
- Add following configs ,save,exit

```
<server>  
    <id>nexus-releases</id>  
    <username>admin</username>  
    <password>v17041995</password>  
</server>  
<server>  
    <id>nexus-snapshots</id>  
    <username>admin</username>  
    <password>v17041995</password>  
</server>
```



The screenshot shows a terminal window with three tabs open:

- Tab 19: master-amz-Inx
- Tab 17: master-amz-Inx
- Tab 18: master-amz-Inx

The active tab (Tab 19) displays the following command-line session:

```
[root@ip-172-31-8-207 apache-maven-3.9.3]# cd conf  
[root@ip-172-31-8-207 conf]# ls  
logging  settings.xml  toolchains.xml  
[root@ip-172-31-8-207 conf]# pwd  
/mnt/tools/apache-maven-3.9.3/conf  
[root@ip-172-31-8-207 conf]#  
[root@ip-172-31-8-207 conf]# vi settings.xml  
[root@ip-172-31-8-207 conf]#
```



```
<!-- Another sample, using keys to authenticate.
<server>
  <id>siteServer</id>
  <privateKey>/path/to/private/key</privateKey>
  <passphrase>optional; leave empty if not used.</passphrase>
</server>
-->

<server>
  <id>nexus-releases</id>
  <username>admin</username>
  <password>v17041995</password>
</server>
<server>
  <id>nexus-snapshots</id>
  <username>admin</username>
  <password>v17041995</password>
</server>

</servers>

<!-- mirrors -->
```

Lets have a test whether build is getting uploaded or not to nexus

mvn deploy

```
[INFO] [INFO] --- deploy:3.1.1:deploy (default-deploy) @ my-webapp ---
[INFO] [INFO] Downloading from nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/1.0-SNAPSHOT/_metadata.xml
[INFO] [INFO] Uploading to nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/1.0-SNAPSHOT/my-w
[INFO] [INFO] -1.0-20230712.133334-1.pom
[INFO] [INFO] Uploading to nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/1.0-SNAPSHOT/my-w
[INFO] [INFO] -1.0-20230712.133334-1.pom (1.8 kB at 14 kB/s)
[INFO] [INFO] Uploading to nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/1.0-SNAPSHOT/my-w
[INFO] [INFO] -1.0-20230712.133334-1.war
[INFO] [INFO] Uploading to nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/1.0-SNAPSHOT/my-w
[INFO] [INFO] -1.0-20230712.133334-1.war (2.9 kB at 25 kB/s)
[INFO] [INFO] Downloading from nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/maven-metadat
[INFO] [INFO] Uploading to nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/1.0-SNAPSHOT/mave
[INFO] [INFO] n-metadata.xml
[INFO] [INFO] Uploading to nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/1.0-SNAPSHOT/maven-
[INFO] [INFO] metadata.xml (764 B at 15 kB/s)
[INFO] [INFO] Uploading to nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/maven-metadata.xm
[INFO] [INFO] Uploading to nexus-snapshots: http://65.2.6.5:8092/repository/hello-world-pkg-snapshot/com/example/my-webapp/maven-metadata.xml
[INFO] [INFO]  B at 5.8 kB/s)
[INFO] [INFO] -----
[INFO] [INFO] BUILD SUCCESS
[INFO] [INFO] -----
[INFO] [INFO] Total time: 2.679 s
[INFO] [INFO] Finished at: 2023-07-12T13:33:36Z
[INFO] [INFO] -----
```

Got successful , lets have a look on nexus repo

Check nexus url : <http://13.233.38.126:8092/#browse/browse:hello-world-pkg-snapshot>

The screenshot shows the Sonatype Nexus Repository interface. The URL in the address bar is 65.2.6.5:8092/#browse/browse:hello-world-pkg-snapshot. The left sidebar has a green 'Browse' button highlighted. The main content area shows a tree view under 'com'. It starts with 'example', then 'my-webapp', then '1.0-SNAPSHOT'. Inside '1.0-SNAPSHOT' are several files: 'my-webapp-1.0-20230712.133334-1.pom', 'my-webapp-1.0-20230712.133334-1.pom.md5', 'my-webapp-1.0-20230712.133334-1.pom.sha1', 'my-webapp-1.0-20230712.133334-1.war', 'my-webapp-1.0-20230712.133334-1.war.md5', 'my-webapp-1.0-20230712.133334-1.war.sha1', 'maven-metadata.xml', 'maven-metadata.xml.md5', 'maven-metadata.xml.sha1', 'maven-metadata.xml', 'maven-metadata.xml.md5', and 'maven-metadata.xml.sha1'. The 'my-webapp-1.0-20230712.133334-1.war' file is currently selected.

----- move forward to release instead of snapshot -----

Little bit change in pom.xml to make release version

Why i m doing :

- Not able to get what i expect it from metadata.xml snapshot repo
- I want to run the script that will automatically fetch latest version of war file  
And download the binary , so that I can later on use it for ci-cd
- With release version , i m getting what I expecting
- I will use shell script to retrieve latest version of war file from metadata.xml

Did following change as you can see in below commit :

- Link :

<https://github.com/vishalk17/sample-hello-world-maven-project/commit/1b7978937013c923fab17ded4aefecc5f2d0e322>

```
pom.xml
@@ -6,7 +6,7 @@
 6   6
 7   7     <groupId>com.example</groupId>
 8   8     <artifactId>my-webapp</artifactId>
 9   -   <version>1.0-SNAPSHOT</version>
 9  +   <version>1.0</version>
10  10
11  11     <packaging>war</packaging>
12  12
@@ -18,7 +18,7 @@
18  18     <sonar.login>sqp_3df7f28c6dc2cdf42a6e346094ec3e5289bc4267</sonar.login>
19  19     <sonar.projectKey>Hello-World-app-test</sonar.projectKey>
20  20     <sonar.projectName>Hello-World-app-test</sonar.projectName>
21  -   <sonar.projectVersion>1.0-SNAPSHOT</sonar.projectVersion>
21  +   <sonar.projectVersion>1.0</sonar.projectVersion>
22  22     <sonar.language>java</sonar.language>
23  23   </properties>
24  24
```

If I again deploy then war will go in release repo that is

In my case it is : <http://13.233.38.126:8092/repository/hello-world-pkg/>

What if I update release version in pom.xml ?? how it will looks like, lets have a look ,

- Update release version in pom.xml to 1.1 from 1.0
- Link :

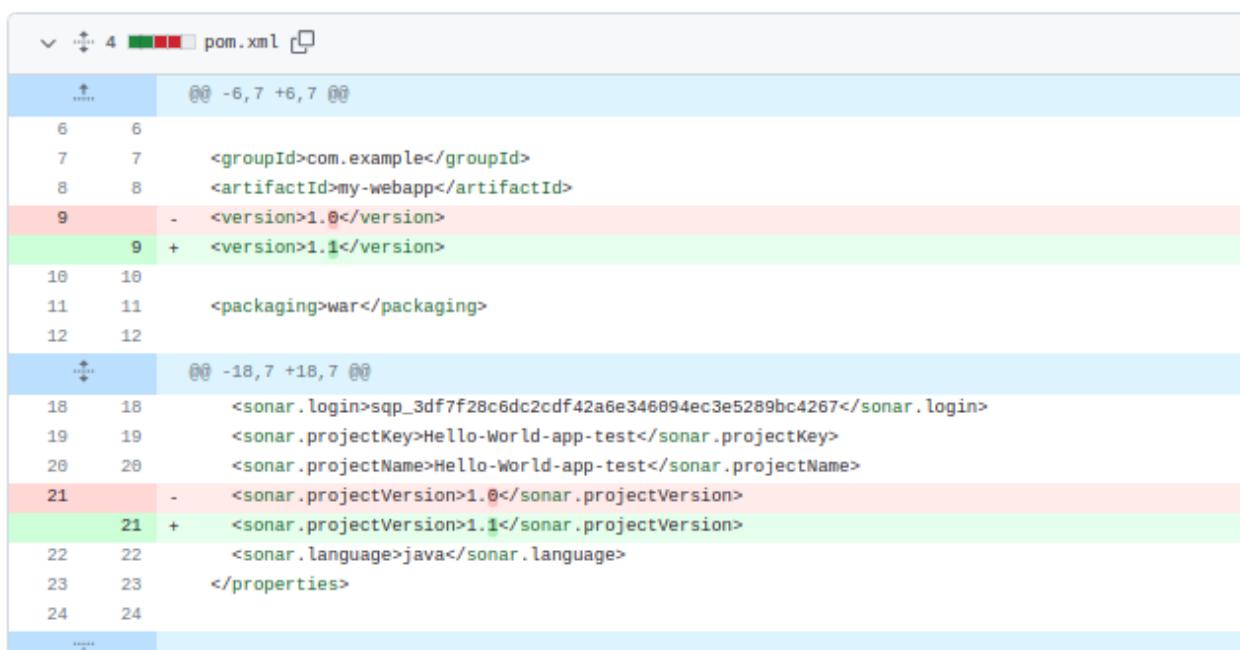
<https://github.com/vishalk17/sample-hello-world-maven-project/commit/151f5bb145880d58580d1a0ddb3c9d595d4c9e1b>

pom.xml: release 2.0

89 master

vishalk17 committed 12 minutes ago

Showing 1 changed file with 2 additions and 2 deletions.



```
diff --git a/pom.xml b/pom.xml
--- a/pom.xml
+++ b/pom.xml
@@ -6,7 +6,7 @@
 6   6
 7   7     <groupId>com.example</groupId>
 8   8     <artifactId>my-webapp</artifactId>
- 9   -     <version>1.0</version>
+ 9   +     <version>1.1</version>
 10  10
 11  11     <packaging>war</packaging>
 12  12
@@ -18,7 +18,7 @@
 18  18     <sonar.login>sqp_3df7f28c6dc2cdf42a6e346894ec3e5289bc4267</sonar.login>
 19  19     <sonar.projectKey>Hello-World-app-test</sonar.projectKey>
 20  20     <sonar.projectName>Hello-World-app-test</sonar.projectName>
- 21  -     <sonar.projectVersion>1.0</sonar.projectVersion>
+ 21  +     <sonar.projectVersion>1.1</sonar.projectVersion>
 22  22     <sonar.language>java</sonar.language>
 23  23   </properties>
```

Lets do maven deploy again :

`mvn deploy`

```
1.war  
[INFO] [INFO] --- deploy:3.1.1:deploy (default-deploy) @ my-webapp ---  
Uploading to nexus-releases: http://65.2.6.5:8092/repository/hello-world-pkg/com/example/my-webapp/1.1/my-webapp-1.1.pom  
Uploaded to nexus-releases: http://65.2.6.5:8092/repository/hello-world-pkg/com/example/my-webapp/1.1/my-webapp-1.1.pom (1.8 kB at 14 kB/s)  
Uploading to nexus-releases: http://65.2.6.5:8092/repository/hello-world-pkg/com/example/my-webapp/1.1/my-webapp-1.1.war  
Uploaded to nexus-releases: http://65.2.6.5:8092/repository/hello-world-pkg/com/example/my-webapp/1.1/my-webapp-1.1.war (2.8 kB at 41 kB/s)  
Downloading from nexus-releases: http://65.2.6.5:8092/repository/hello-world-pkg/com/example/my-webapp/maven-metadata.xml  
Downloaded from nexus-releases: http://65.2.6.5:8092/repository/hello-world-pkg/com/example/my-webapp/maven-metadata.xml (296 B at 18 kB/s)  
Uploading to nexus-releases: http://65.2.6.5:8092/repository/hello-world-pkg/com/example/my-webapp/maven-metadata.xml  
Uploaded to nexus-releases: http://65.2.6.5:8092/repository/hello-world-pkg/com/example/my-webapp/maven-metadata.xml (325 B at 7.4 kB/s)  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 2.428 s  
[INFO] Finished at: 2023-07-13T05:07:02Z  
[INFO] -----  
[root@ip-172-31-8-207 hello-world]# ls
```

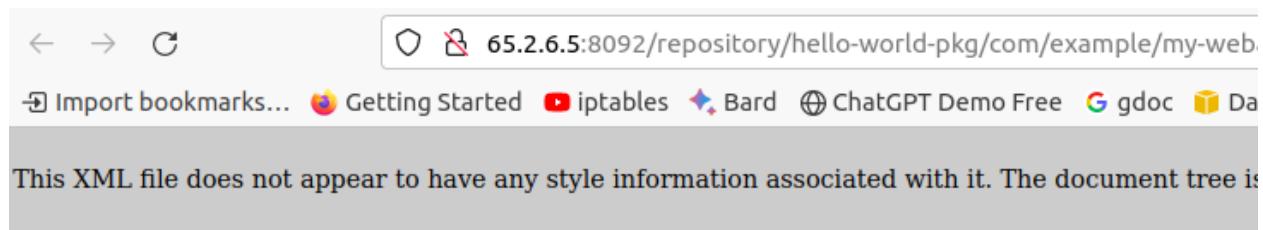
- Got successful

Lets have a look on nexus repo :

My nexus url : <http://13.233.38.126:8092/#browse/browse:hello-world-pkg>

The screenshot shows the Sonatype Nexus Repository interface. The URL in the browser is <http://65.2.6.5:8092/#browse/browse:hello-world-pkg>. The left sidebar has a green 'Browse' button selected. The main area shows a tree view of the repository structure under 'hello-world-pkg': com > example > my-webapp > 1.0 > 1.1 > maven-metadata.xml, maven-metadata.xml.md5, maven-metadata.xml.sha1.

- Lets check what inside of maven-metadata.xml



This XML file does not appear to have any style information associated with it. The document tree is as follows:

```
-<metadata>
  <groupId>com.example</groupId>
  <artifactId>my-webapp</artifactId>
  -<versioning>
    <release>1.1</release>
    -<versions>
      <version>1.0</version>
      <version>1.1</version>
    </versions>
    <lastUpdated>20230713050700</lastUpdated>
  </versioning>
</metadata>
```

Make a shell script to retrieve latest version and download latest binary whenever executing the bash script

- get-web-app-latest-war.sh

```
#!/bin/bash

# Define the Nexus URL and the artifact coordinates.
NEXUS_URL="http://13.233.38.126:8092/repository/hello-world-pkg/com/example/my-webapp/maven-metadata.xml"
GROUP_ID="com.example"
ARTIFACT_ID="my-webapp"
war_path="http://13.233.38.126:8092/repository/hello-world-pkg/com/example/my-webapp"
# Get the latest version of the artifact.
LATEST_VERSION=$(curl -s "$NEXUS_URL" | grep -oP "(?=<>version>)[^<]*(?=</version>)" | sort -V | tail -1)

# remove any previous war app
sudo rm -rf my-webapp.war

# Download the latest artifact.
curl -o my-webapp.war "$war_path/$LATEST_VERSION/my-webapp-$LATEST_VERSION.war"
```

## Integration of ansible in jenkins :

### Requirement

- Ansible setup on both server
- Setup of inventory file

#### In jenkins

- Ansible plugin integration
- Ansible credentials

- Installation of ansible on both server
- I am using amz linux 2022 on both server

Ip of both servers:

Master server: 13.233.38.126

Slave server : 13.234.110.144

I already have the script, so without wasting time , lets have a look:

You can check here:

[https://github.com/vishalk17/devops/blob/main/ansible/general\\_setup.sh](https://github.com/vishalk17/devops/blob/main/ansible/general_setup.sh)



```
#!/bin/bash

# ***** allow further script only if root user executing the script start*****
if [[ $EUID == 0 ]]
then

    ## install require pkges ##
    amazon-linux-extras install ansible2 -y
    yum install git -y
    sleep 4

    ### add user with password start ####
    # -m : The user's home directory will be created if it does not exist.
    # -p EncryptedPasswordHere : The encrypted password, as returned by crypt().
    # username : Add this user to the Linux system,
    #
    # useradd -m username
    #
    # change/set password of ansible user
    # echo "newuser:newpassword" | chpasswd
    #
    useradd -m ansible
    echo "ansible:ansible" | chpasswd
    #
    ### add user with password End ####

    ### provide ansible as a user root privilege start #####
    #
    echo 'ansible ALL=(ALL)      NOPASSWD: ALL' >> /etc/sudoers
    #
    ### provide ansible as a user root privilege end #####

    ##      edit /etc/ansible/ansible.cfg start ####
    #      uncomment inventory line to enable hosts file
    #      uncomment sudo_user line
    #      replace lines by sed command
    #      sed -i "s%old_line%new_line%g" /path/to/the/file
    #
    sed -i "s%#inventory      = /etc/ansible/hosts%inventory      = /etc/ansible/hosts%g" /etc/ansible/ansible.cfg
    sed -i "s%#sudo_user      = root%sudo_user      = root%g" /etc/ansible/ansible.cfg
    #
    ##      edit /etc/ansible/ansible.cfg End ####

    ##      edit /etc/ssh/sshd_config start ####
    #
    #      uncomment PermitRootLogin yes
    #      uncomment PasswordAuthentication yes
    #      comment PasswordAuthentication no
    #      restart sshd service
    #
    sed -i "s%#PermitRootLogin yes%PermitRootLogin yes%g" /etc/ssh/sshd_config
    sed -i "s%#PasswordAuthentication yes%PasswordAuthentication yes%g" /etc/ssh/sshd_config
    sed -i "s%PasswordAuthentication no%#PasswordAuthentication no%g" /etc/ssh/sshd_config

    service sshd restart
    #
    ##      edit /etc/ssh/sshd_config end ####
    #
else
    echo " you are normal user, only root user can execute this script $0"
fi
#
# ***** allow further script only if root user executing the script End*****
```

- Execute above script on both setup this is general setup to do
- Above script will create necessary things
  - Our ansible user name : ansible
  - Our ansible password : ansible
- Setup inventory file on ansible server
  - It is located in /etc/hosts/host.conf

**vi /etc/ansible//hosts**

- Add nodes ip in the group
- In my case my group is **deploy-war-app**
- In our case slave1 ip are as follows
  - Public-ip: 13.234.110.144
  - Private ip : 172.31.36.98
  - Our both ansible server and node (slave one in same vpc)

```
## 10.25.1.57

# Here's another example of host ranges, this time
# leading 0s:

## db-[99:101]-node.example.com

[deploy-war-app]
172.31.36.98
"/etc/ansible//hosts" 46L, 1046B
```

- Save and exit
- Follow next steps using ref provide below
- [https://github.com/vishalk17/devops/blob/main/ansible\\_notes\\_vishalk17.pdf](https://github.com/vishalk17/devops/blob/main/ansible_notes_vishalk17.pdf)
  - Transfer ssh keys to other server
  - So that we have authentication from ansible server to its node

## Create a new Jenkins credential:

- Go to the Jenkins dashboard.
- Click on "Manage Jenkins" in the left sidebar.
- Select "Manage Credentials".
- Click on "Jenkins" under "Stores scoped to Jenkins".
- Click on "Global credentials (unrestricted)" or any other appropriate domain.
- Click on "Add Credentials".
- Fill in the necessary details, including the username and password of the Ansible user.
- Click on "OK" to save the credential.

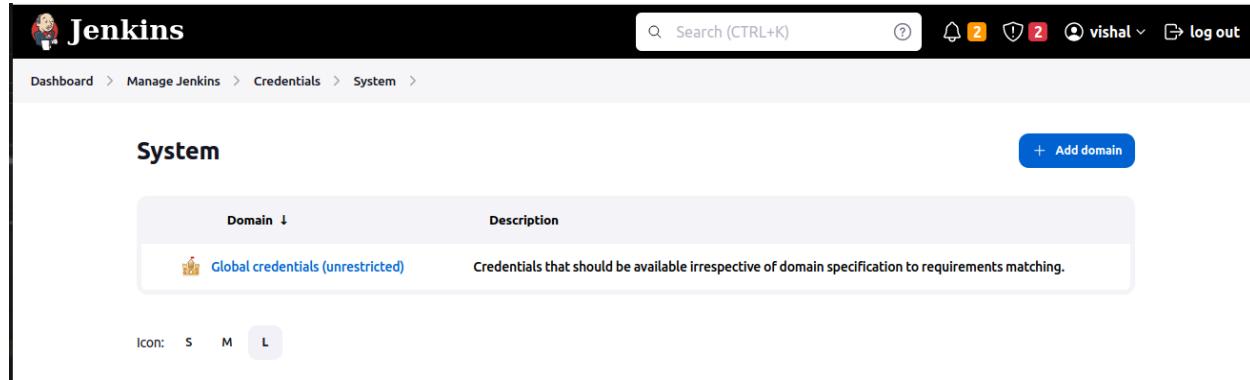
**System Configuration**

- Configure System**  
Configure global settings and paths.
- Global Tool Configuration**  
Configure tools, their locations and automatic installers.
- Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

---

**Security**

- Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.
- Manage Credentials**  
Configure credentials
- Configure Credential Providers**  
Configure the credential providers and types



**Global credentials (unrestricted)**

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

ID	Name	Kind	Description
This credential domain is empty. How about <a href="#">adding some credentials?</a>			

Icon: S M L

## Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

t.me/vishalk17

github.com/vishalk17

Kind: Username with password

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

Username: ansible

Treat username as secret

Password: \*\*\*\*\*

ID: ansible-credentials

Create

- In previous section as per ansible script
- Username : ansible
- Pass: ansible

ID	Name	Kind	Description
ansible-credentials	ansible/*****	Username with password	

+ Add Credentials

### Install the Ansible plugin in Jenkins:

- Go to the Jenkins dashboard.
- Click on "Manage Jenkins" in the left sidebar.
- Select "Manage Plugins".
- Go to the "Available" tab and search for "Ansible".
- Check the box next to the "Ansible" plugin.
- Click on "Install without restart" or "Download now and install after restart" depending on your preference.

Warnings have been published for the following currently installed components:

Jenkins 2.375.1 core and libraries:  
**Multiple security vulnerabilities in Jenkins 2.393 and earlier, LTS 2.375.3 and earlier**  
**CSRF bypass vulnerability**

Fixes for all of these issues are available. Go to the [plugin manager](#) to update the plugin.

**Plugins**

Available plugins: ansible

Install	Name	Released
<input type="checkbox"/>	Ansible 148.v6b_13c6de3a_47	5 mo 13 days ago

pipeline External Site/Tool Integrations DevOps Build Tools Deployment

Invoke Ansible Ad-Hoc commands and playbooks.

Warning: This plugin version may not be safe to use. Please review the following security notices:

- Secrets stored and displayed in plain text

A newer version than being offered for installation exists (version 240.vc26740a\_625c0), so the latest bug fixes or features are not available to you. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. If you are using the latest version of Jenkins offered to you, this plugin release may not be available to your release line yet. See the [plugin documentation](#) for information about its

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 57 min ago [Check now](#)

**Download progress**

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Ansible Success

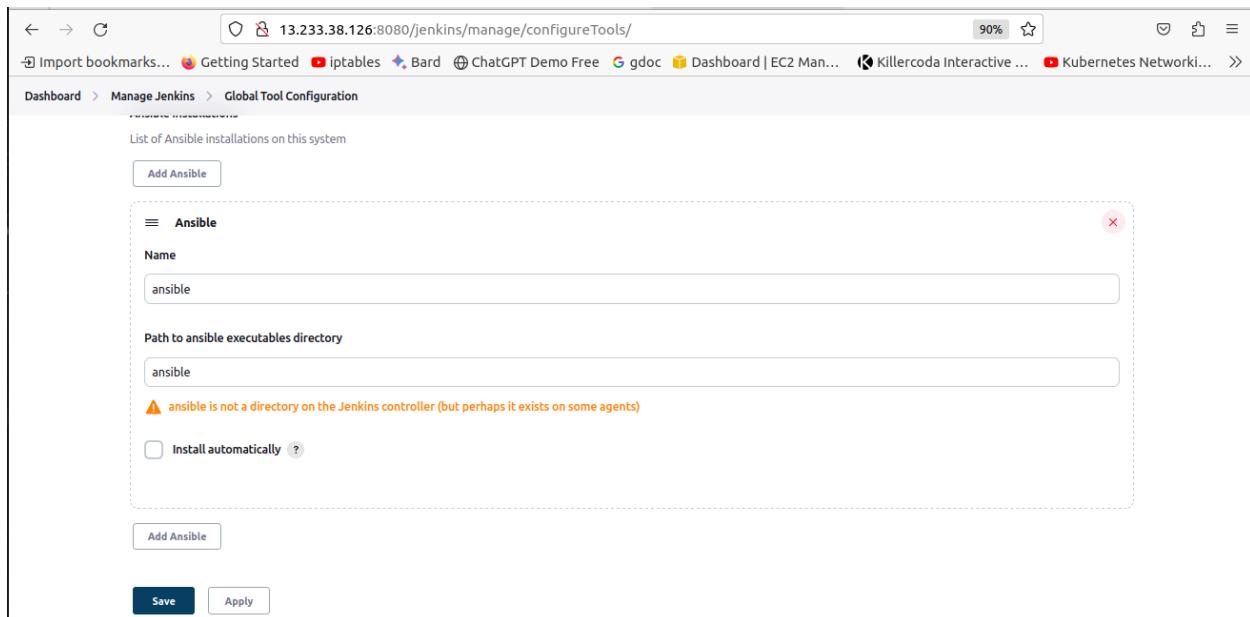
Loading plugin extensions Success

→ [Go back to the top page](#)  
 (you can start using the installed plugins right away)

→  Restart Jenkins when installation is complete and no jobs are running

## Configure Ansible on Jenkins:

- Go to the Jenkins dashboard.
- Click on "Manage Jenkins" in the left sidebar.
- Select "Global Tool Configuration".
- Scroll down to the "Ansible" section.
- Click on "Add Ansible" to add a new Ansible installation.
- Provide a name for the Ansible installation and specify the path to the Ansible executable.
- Click on "Save" to save the configuration.



- Standalone Ansible already installed thus no need to give full path

## Create Jenkins Job - use of ansible- to deploy war on remote server i.e, slave1

- Tomcat server on slave1
  - Location /mnt/tools/apache
- on server one, i.e., ansible server;
  - Script to use downloading latest war from nexus artifact located in **/mnt/ansible**
- [Make a ansible-playbook](#) to transfer downloaded war to remote server over ansible

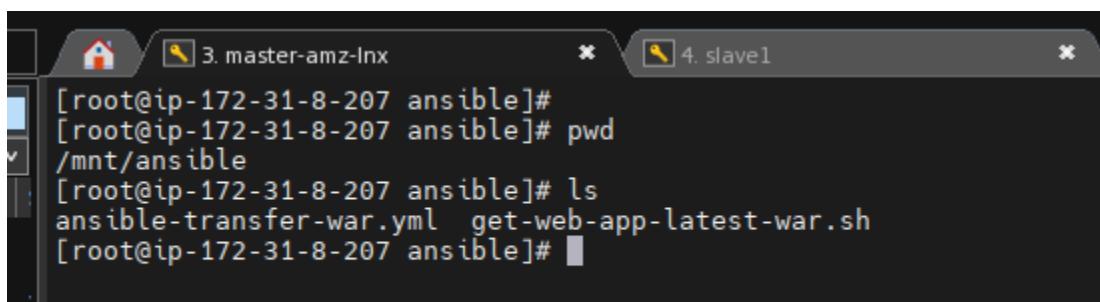
[vi ansible-transfer-war.yml](#)

```
---
- hosts: deploy-war-app
  user: ansible
  become: yes
  connection: ssh
  gather_facts: yes
  tasks:
    - name: stop tomcat server
      command: bash /mnt/tools/apache-tomcat-9.0.78/bin/shutdown.sh

    - name: copy war file to tomcat dir
      copy:
        src: my-webapp.war
        dest: /mnt/tools/apache-tomcat-9.0.78/webapps/

    - name: start tomacat server
      command: bash /mnt/tools/apache-tomcat-9.0.78/bin/startup.sh
```

- Save and exit
- In /mnt/ansible folder we have following things



```
[root@ip-172-31-8-207 ansible]#
[root@ip-172-31-8-207 ansible]# pwd
/mnt/ansible
[root@ip-172-31-8-207 ansible]# ls
ansible-transfer-war.yml  get-web-app-latest-war.sh
[root@ip-172-31-8-207 ansible]#
```

[Lets integrate above thing in jenkins](#)

- Click on Create job or new item
- Select pipeline
- Item name : **step2-deploy-war-ansible-to-slave1**
- Click on ok

The screenshot shows the Jenkins 'Enter an item name' dialog. The input field contains 'step2-deploy-war-ansible-to-slave1' with a note '(Required field)'. Below the input field are three project types: 'Freestyle project', 'Pipeline', and 'Multi-configuration project'. Each type has a description and a small icon.

- Scroll down and write pipeline script

The screenshot shows the Jenkins 'Configure' page for the 'step2-deploy-war-ansible-to-slave1' pipeline. The 'Pipeline' tab is selected. The 'Definition' section shows 'Pipeline script' selected. A large text area labeled 'Script' is empty. Below it is a checkbox for 'Use Groovy Sandbox' which is checked. At the bottom are 'Save' and 'Apply' buttons.

Pipeline script as follows:

```
pipeline {  
    agent {  
        label "built-in" // master node  
    }  
    stages {  
        stage("download latest war of web-app from nexus") {  
            steps {  
                sh "sudo bash /mnt/ansible/get-web-app-latest-war.sh"  
            }  
        }  
        stage("deploy war to slave1 over ansible") {  
            steps {  
                withCredentials([usernamePassword(credentialsId: 'ansible-credentials', passwordVariable: 'ansible_password', usernameVariable: 'ansible_username')]) {  
                    sh "ansible-playbook /mnt/ansible/ansible-transfer-war.yml --user=${ansible_username}  
--extra-vars='ansible_ssh_pass=${ansible_password}'"  
                }  
            }  
        }  
    }  
}
```

1. The `pipeline` block defines the main structure of the Jenkins pipeline.
2. The `agent` block specifies the agent or node on which the pipeline will be executed. In this case, it is set to the "built-in" label, which typically refers to the master node.
3. The `stages` block contains one or more `stage` blocks, representing different stages or phases of the pipeline.
4. The first stage is named "download latest war of web-app from nexus". Inside this stage, there is a `steps` block that contains the actual steps to be executed.
5. The `sh` step is used to execute shell commands. In this case, it runs the command `sudo bash /mnt/ansible/get-web-app-latest-war.sh`. This script is responsible for downloading the latest version of the war file for the web application from a Nexus repository.
6. The second stage is named "deploy war to slave1 over ansible". Inside this stage, there is a `steps` block that contains the steps to be executed.
7. The `withCredentials` block is used to retrieve the Ansible credentials from Jenkins. It takes the credentials ID (`ansible-credentials`) and the variable names (`ansible_password` and `ansible_username`) to store the credential values.
8. The `sh` step is used again to execute the `ansible-playbook` command. It runs the playbook `/mnt/ansible/ansible-transfer-war.yml` using the Ansible credentials obtained in the previous step. The `--user` flag specifies the username for the Ansible connection, and the `--extra-vars` flag passes the password for the Ansible connection.
9. Ansible connection.

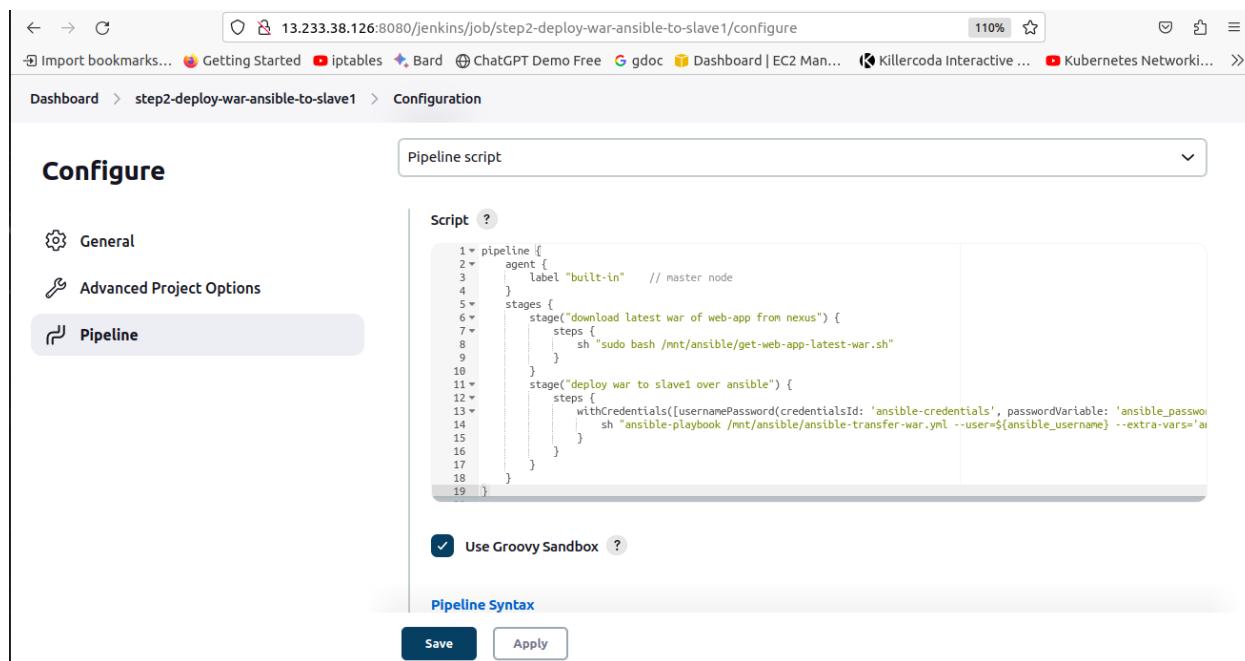
## Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

[t.me/vishalk17](https://t.me/vishalk17)

[github.com/vishalk17](https://github.com/vishalk17)

Overall, this script downloads the latest war file for the web application using the `get-web-app-latest-war.sh` script and then deploys it to a target server using Ansible via the `ansible-playbook` command.

Make sure to replace '`ansible-credentials`' with the actual ID of the Ansible credential stored in Jenkins.



The screenshot shows the Jenkins Pipeline configuration page for a job named "step2-deploy-war-ansible-to-slave1". The "Pipeline" tab is selected in the sidebar. The main area contains a Groovy script for defining the pipeline stages:

```
1 pipeline {
2     agent {
3         label "built-in" // master node
4     }
5     stages {
6         stage("download latest war of web-app from nexus") {
7             steps {
8                 sh "sudo bash /mnt/ansible/get-web-app-latest-war.sh"
9             }
10    }
11    stage("deploy war to slave1 over ansible") {
12        steps {
13            withCredentials([
14                usernamePassword(credentialsId: 'ansible-credentials', passwordVariable: 'ansible_password')
15            ])
16        }
17    }
18 }
19 }
```

Below the script, there is a checkbox for "Use Groovy Sandbox" which is checked. At the bottom are "Save" and "Apply" buttons.

- Save script
- Click on buildnow



The screenshot shows the Jenkins Pipeline step status for the "step2-deploy-war-ansible-to-slave1" pipeline. The "Status" tab is selected. The pipeline has two stages: "download latest war of web-app from nexus" and "deploy war to slave1 over ansible". Both stages are shown as completed successfully.

**Stage View**

download latest war of web-app from	deploy war to slave1 over ansible
-------------------------------------	-----------------------------------

- Successfully deploy

- Try to access that webpage for which we deployed war file on slave1 server
  - web application will be accessible at the following URL:  
`http://<tomcat_server_ip>:<tomcat_port>/my-webapp/hello`
  - Slave1 : public ip is : 13.234.110.144
- Thus my url will be <http://13.234.110.144:8080/my-webapp/hello>

This way I have completed th 2nd step of my ci-cd using jenkins , lets do first step

## First stage of ci-cd using jenkins

### Flow of whole ci-cd :

- Developer push code to the git repository
- Git web-hook configure with jenkins
- Git web-hook trigger jenkins to execute 1st stage
- In first stage after completion ,we trigger 2nd stage

What I m goint to do in first stage:

- Create pipeline script in which
  - Download code from gitub repository
  - Compile using maven and at the same time do sonar-scanner analysis
  - Compiled war file will go in nexus repo
  - trigger 2nd stage

### Configure git web-hook

A git web hook is a way to integrate external services with Git. When a specific event happens in a Git repository, a web hook can be configured to trigger an HTTP POST request to a specified URL. This can be used to automate tasks such as sending notifications, updating build status, or deploying code.

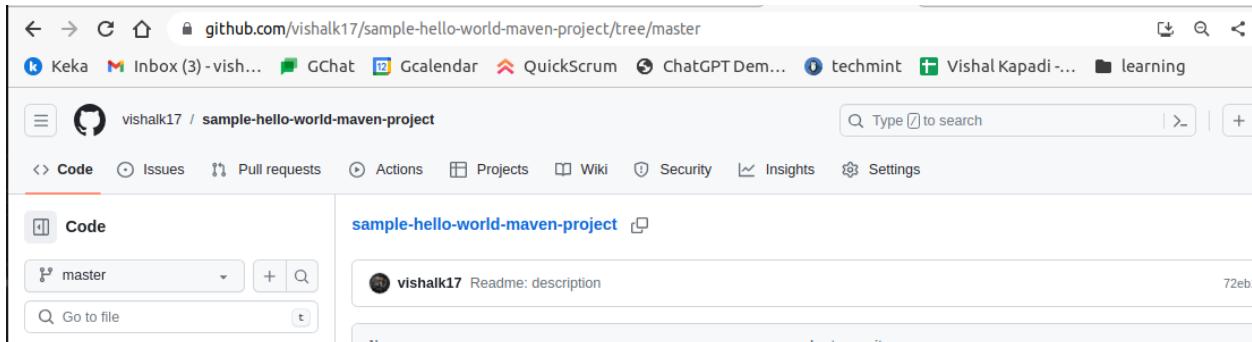
A git web hook with Jenkins is a way to integrate Jenkins with Git. When a specific event happens in a Git repository, a web hook can be configured to trigger a Jenkins job. This can be used to automate tasks such as building, testing, and deploying code.

For example, you could configure a web hook to be triggered whenever a new commit is pushed to a Git repository. When this happens, the web hook would trigger a Jenkins job that would build, test, and deploy the code to a production server.

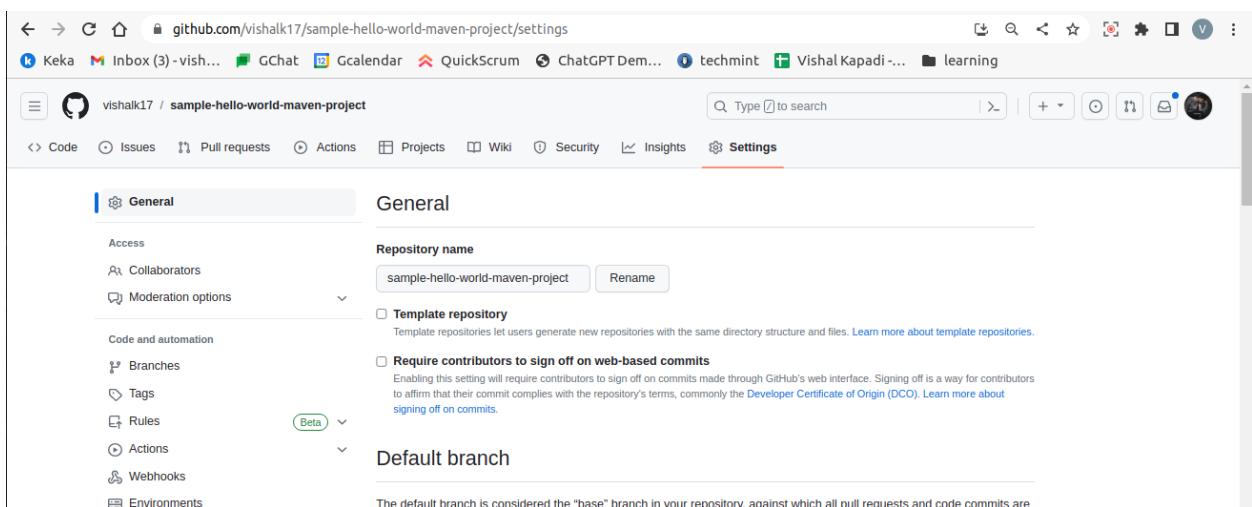
To set up a git web hook with Jenkins, you will need to:

1. Create a Jenkins job that will be triggered by the web hook.
2. Configure the web hook in your Git repository.

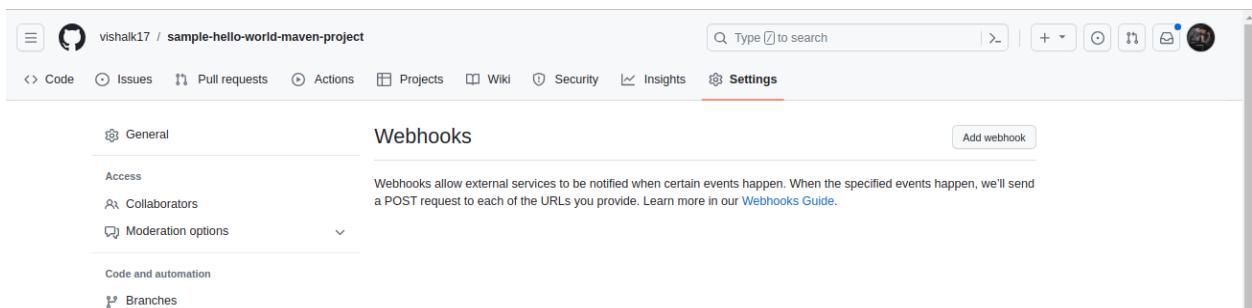
- Go to the repo setting where you want to add git web-hook



- On left side bar, click on webhook



- Click on add web-hook



- Provide the Jenkins URL in the "Payload URL" field.
- The webhook URL of Jenkins is typically in the format:  
[http://<jenkins\\_server>/github-webhook/](http://<jenkins_server>/github-webhook/)

In my case it is

## Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

[t.me/vishalk17](https://t.me/vishalk17)

[github.com/vishalk17](https://github.com/vishalk17)

<http://13.233.38.126:8080/jenkins/github-webhook/>

- Select send me everything
- Click on add-webhook and finish

The screenshot shows the GitHub 'Webhooks / Add webhook' configuration page. On the left, a sidebar menu is open under the 'Webhooks' section, which is highlighted in blue. The main form area contains the following fields:

- Payload URL \***: `http://13.233.38.126:8080/jenkins/github-webhook/`
- Content type**: `application/x-www-form-urlencoded`
- Secret**: An empty text input field.
- Which events would you like to trigger this webhook?**:
  - Just the push event.
  - Send me **everything**.
  - Let me select individual events.
- Active**: A checked checkbox with the note "We will deliver event details when this hook is triggered."
- Add webhook**: A green button at the bottom of the form.

The screenshot shows the GitHub 'Settings' page for the 'sample-hello-world-maven-project'. The 'Webhooks' section is visible, displaying the newly created webhook entry:

URL	Action
<code>http://13.233.38.126:8080/jenkins/github-webhook/</code>	(all events)

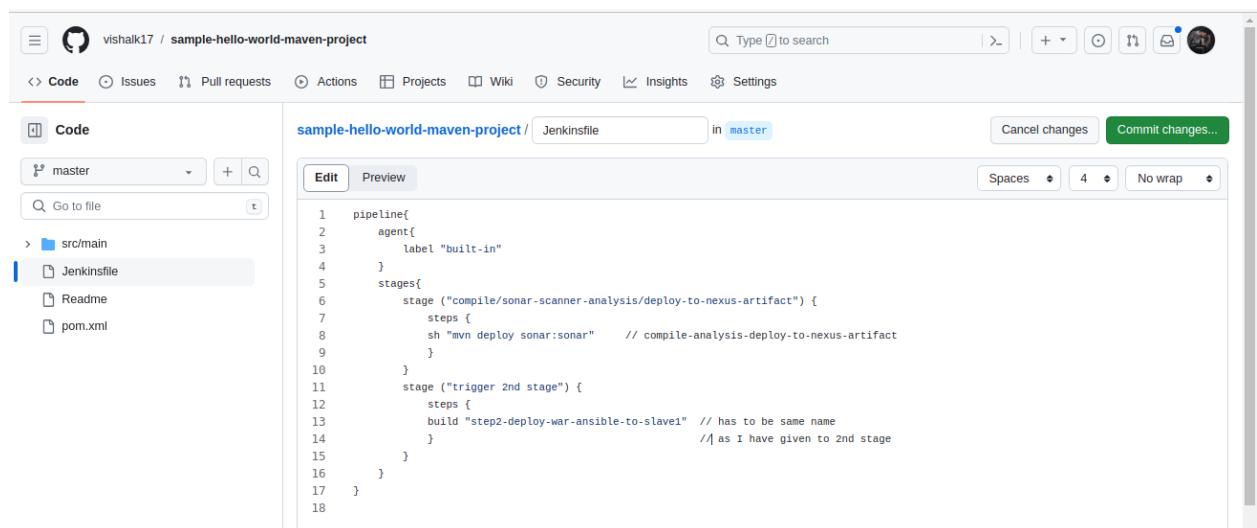
Below the table, there are 'Edit' and 'Delete' buttons. The rest of the page includes standard GitHub navigation links like Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

Create Jenkinsfile in root of ther repo

- So that we can add it in jenkins job directly from there

In Jenkinsfile we need to provide pipeline script

```
pipeline{
    agent{
        label "built-in"
    }
    stages{
        stage ("compile/sonar-scanner-analysis/deploy-to-nexus-artifact") {
            steps {
                sh "mvn deploy sonar:sonar"          # compile-analysis-deploy-to-nexus-artifact
            }
        }
        stage ("trigger 2nd stage") {
            steps {
                build "step2-deploy-war-ansible-to-slave1"      # has to be same name
            }
        }
    }
}
```



# Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

[t.me/vishalk17](https://t.me/vishalk17)

[github.com/vishalk17](https://github.com/vishalk17)

The screenshot shows a GitHub repository named 'sample-hello-world-maven-project'. The 'Code' tab is selected, showing the 'master' branch with files 'src/main', 'Jenkinsfile', 'Readme', and 'pom.xml'. The commit history is displayed on the right, with the most recent commit by 'vishalk17' being 'jenkins: add Jenkinsfile' (commit c8e7982, now). Other commits include 'initial upload: hello-word mvn project | sonarqube integration' (2 days ago), 'jenkins: add Jenkinsfile' (now), 'Readme: description' (1 hour ago), and 'pom.xml: update url endpoints' (4 hours ago).

## Configure job in jenkins:

- Click on Create job or new item
- Select pipeline
- Item name : **Step1-compile-push-artifact-web-hook**
- Click on ok

2nd step job name : **step2-deploy-war-ansible-to-slave1**

The screenshot shows the Jenkins configuration page for the job 'step1-compile-push-artifact-web-hook'. Under the 'General' section, the 'GitHub hook trigger for GITScm polling' checkbox is checked. Other options like 'Build after other projects are built', 'Build periodically', 'Poll SCM', 'Quiet period', and 'Trigger builds remotely' are available but not selected.

- Select GitHub hook trigger for GITScm polling
- Scroll down below

## Ci-cd-git-jenkins-nexus- sonarqube-mvn-ansible

[t.me/vishalk17](https://t.me/vishalk17)

[github.com/vishalk17](https://github.com/vishalk17)

- In pipeline field , select option “ pipeline script from scm”
- In scm field select “git”
  - Provide git url , and specify branch name

Dashboard > deploy my-webapp > step1-compile-push-artifact-web-hook > Configuration

### Configure

#### Pipeline

General

Advanced Project Options

Pipeline

#### Definition

Pipeline script from SCM

#### SCM

Git

#### Repositories

Repository URL

Please enter Git repository.

Dashboard > deploy my-webapp > step1-compile-push-artifact-web-hook > Configuration

### Configure

General

Advanced Project Options

Pipeline

Repository URL

<https://github.com/vishalk17/sample-hello-world-maven-project>

#### Credentials

- none -

Add

Advanced...

Add Repository

#### Branches to build

Branch Specifier (blank for 'any')

\*/\*master

Add Branch

#### Descriptor behaviors

Save

Apply

- Scrolldown and given path to Jenkins file in repo
- In my case it is in root of repo
- Finally click on save

Advanced Project Options

Pipeline

#### Additional Behaviours

Add

#### Script Path

Jenkinsfile

Lightweight checkout

#### Pipeline Syntax

Save

Apply

## Run the job so that we can check ci-cd

- In ci cd , we will update or push code to git repo
- Git will notify to the jenkins
- Jenkins then trigger first job in which
  - Pull source code to the local
  - Compile it using maven and do sonar-scanner analysis
  - Deploy war file to the nexus artifact
  - Trigger 2nd job
    - It download latest war file from nexus artifact
    - Then it will transfer that war to other server using ansible playbook
    - Then ansible playbook start tomcat server

-finally our web page accessible from the web

The screenshot shows the Jenkins Pipeline step1-compile-push-artifact-web-hook dashboard. On the left, there's a sidebar with various pipeline management options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, Pipeline Syntax, and GitHub Hook Log. The main area is titled "Pipeline step1-compile-push-artifact-web-hook". Below the title, there's a "Stage View" section showing three stages: "Declarative: Checkout SCM" (785ms), "compile/sonar-scanner-analysis /deploy-to-nexus-artifact" (9s), and "trigger 2nd stage" (8s). The "trigger 2nd stage" stage is currently executing. Below the stage view, there's a "Build History" section showing two builds: #8 (Jul 14, 2023, 9:09 AM) and #7 (Jul 14, 2023). The build history shows a green bar for the first build and a blue bar for the second.

## Ci-cd-git-jenkins-nexus-sonarqube-mvn-ansible

[t.me/vishalk17](https://t.me/vishalk17)

[github.com/vishalk17](https://github.com/vishalk17)

The screenshot shows the Jenkins interface for a pipeline named "Pipeline step2-deploy-war-ansible-to-slave1". On the left, there's a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area is titled "Stage View" and displays two stages: "download latest war of web-app from nexus" (312ms) and "deploy war to slave1 over ansible" (5s). Below this is a "Build History" section showing build #19 from Jul 14 at 14:39 with "No Changes".

The screenshot shows a browser window displaying the result of the Jenkins pipeline. The URL is 13.234.110.144:8080/my-webapp/hello. The page content is "Hello, World!".

**My devops repo :**

- <https://github.com/vishalk17/devops>

**My telegram channel:**

-  [https://t.me/vishalk17\\_devops](https://t.me/vishalk17_devops)

**Contact:**

Telegram :  t.me/vishalk17

**vishalk17 My youtube Channel :**

-  <https://www.youtube.com/@vishalk17>