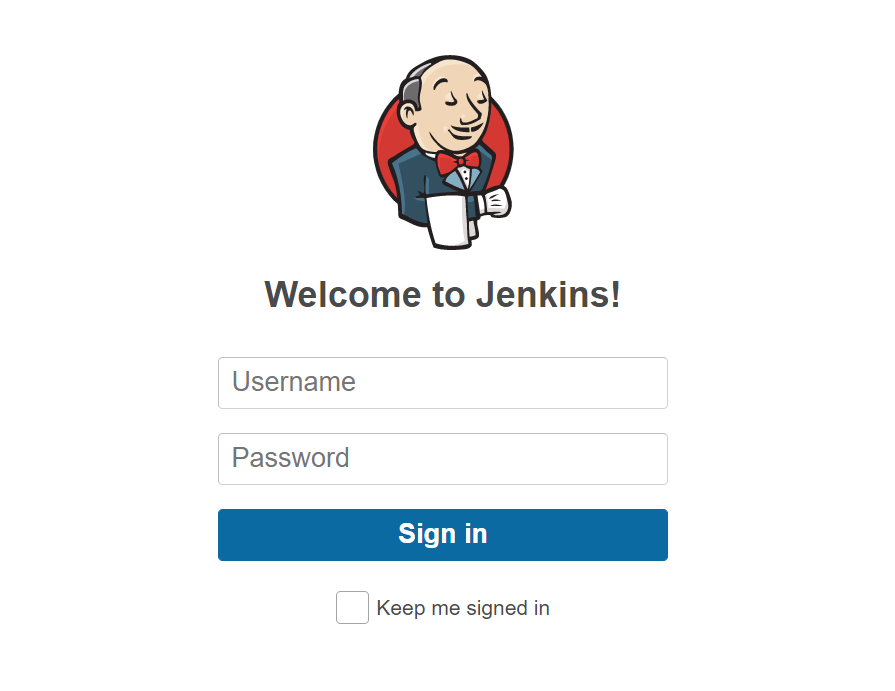
**Sample Project – To Build A CI/CD Pipeline Using Jenkins**

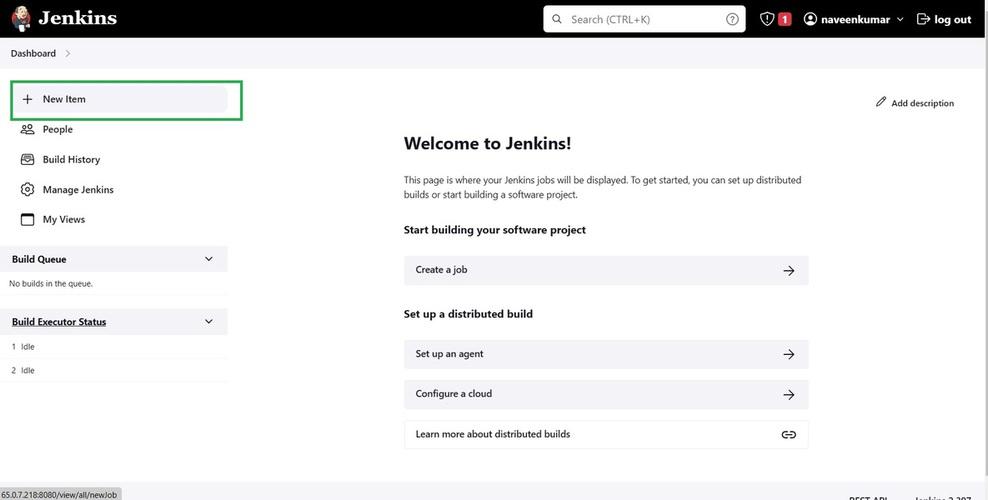
In this project, we will try to learn a basic CI/CD pipeline that is required to build a Java web application .war file by using [Jenkins](https://www.geeksforgeeks.org/jenkins/). To install Jenkins refers to the [Jenkins installation](https://www.geeksforgeeks.org/how-to-install-and-configure-jenkins-on-debian-linux-ubuntu-kali-mint/).

**Step 1:** Login into your Jenkins account as shown below.

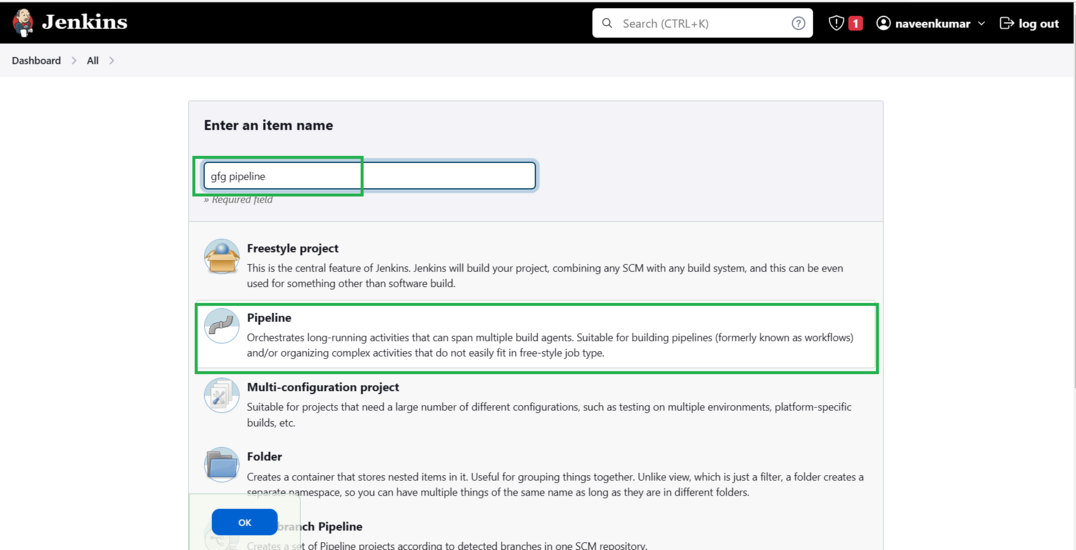
**Step 2.**Once logged in, the user will be redirected to the Jenkins console, here’s the reference for the same.



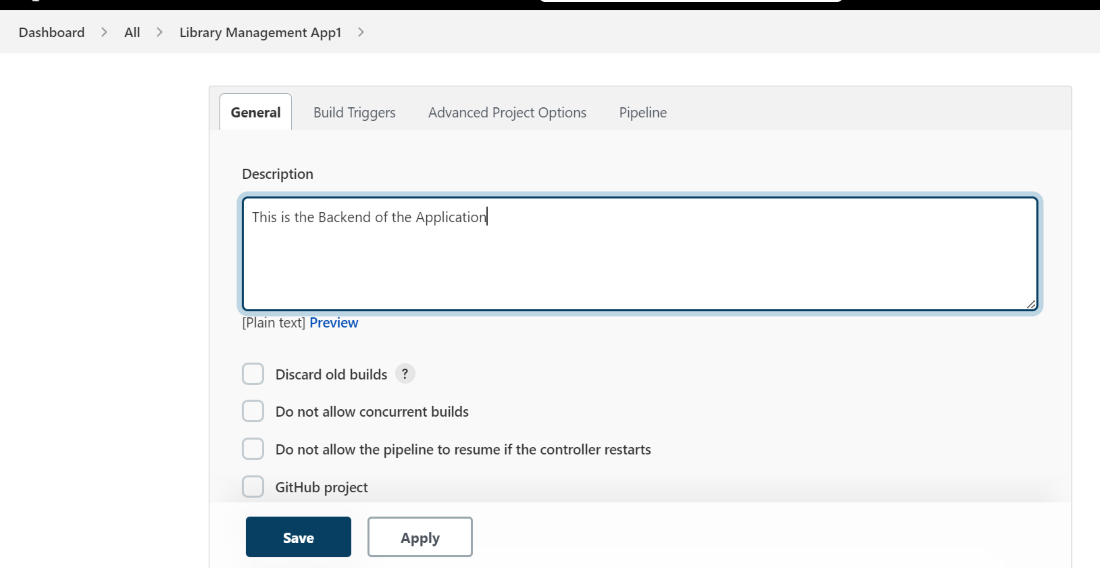
**Step 3**. To create a new project select the option available in the Dashboard which is **“New Item”** Refer to the image provided below:



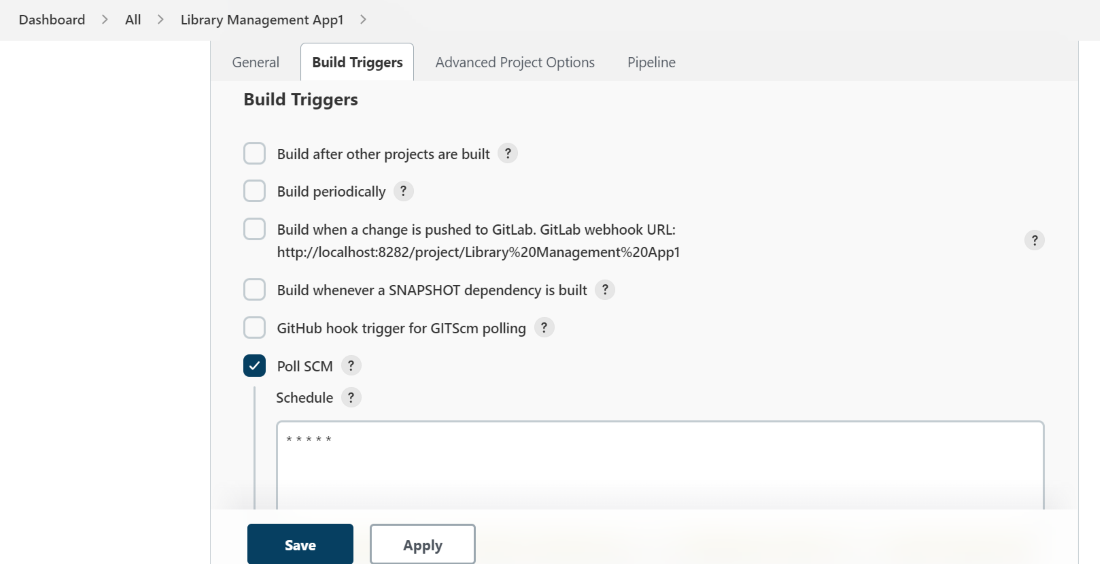
**Step 4**. Now a list of options will be visible on the screen, along with a field to name the pipeline. Add a suitable name and select the **“Pipeline”** option to proceed. Refer to this screenshot.



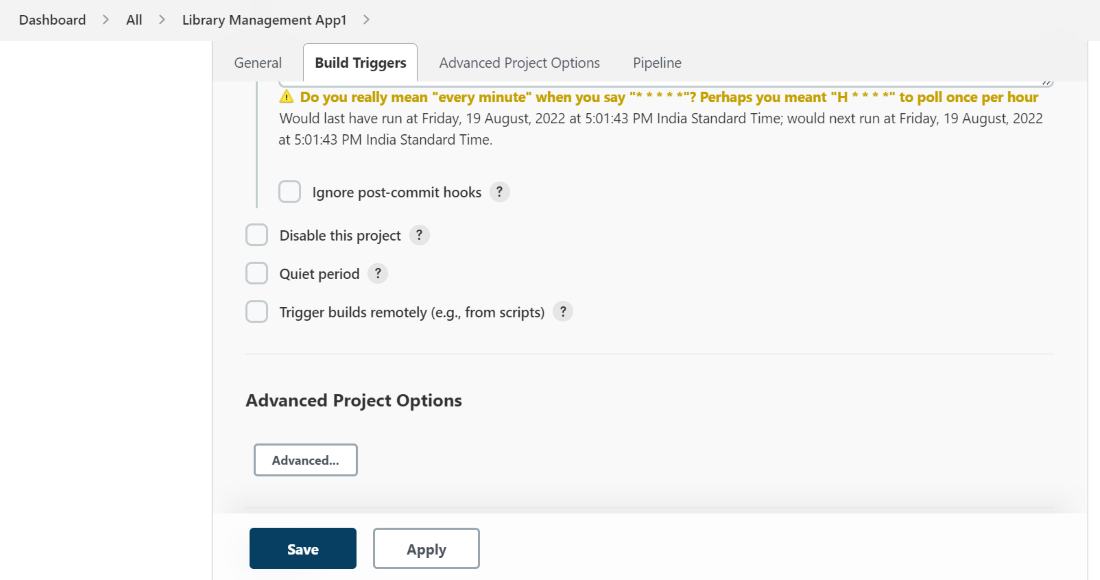
**Step 5:**Once redirected, the configuration page will appear. This is the most important page as here all the details will be filled in. At first, there is the **General** section where the user can add a description based on the project for which the pipeline has to be created. And establish the connection to compute from where the pipeline will access the project. Refer to the screenshot to understand better.



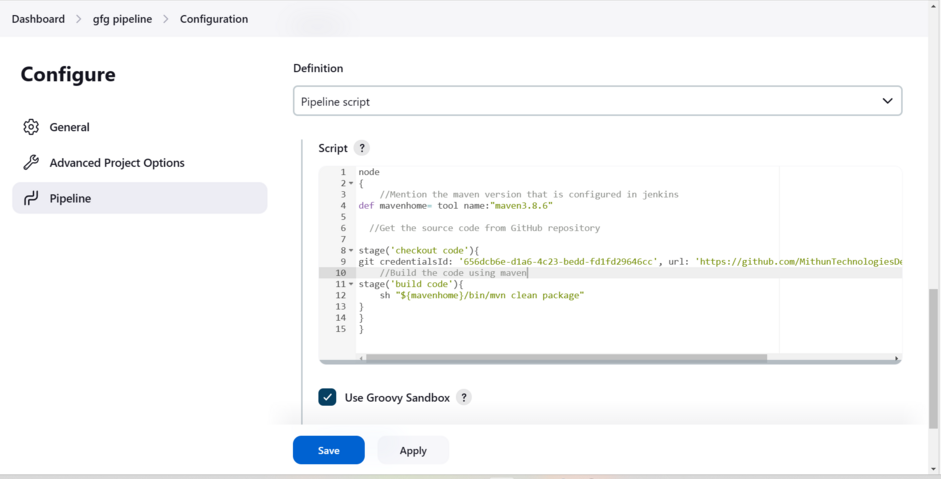
**Step 6.**Now comes the second section, i.e. **“Build triggers”.**Here, we need to specify the branch and repository and give the credentials too. And add additional behaviors if required so far. Refer to the screenshot to have a better understanding.



**Step 7.**The next section is **“Advanced Project Options”,**as the name suggests it is related to the special pipelines only, simpler projects do not require any specifications in this section. Please refer to the screenshot given below for the same.



**Step 8.**This is the last section i.e. **“Pipeline”**. Here the user specifies from where the scripts will be imported including the path to the file, repository, credentials, etc. Refer to the screenshot attached below for reference.



**Sample Pipelinescript To Deploy The Web Application Into The Tomcat Server**

node

{

//Mention the tools which have been configured

def mavenhome= tool name:"\*\*\*\*\*"

// Mention how to trigger the Pipeline and how many Builds must be there and so on

properties([buildDiscarder(logRotator(artifactDaysToKeepStr:

'', artifactNumToKeepStr: '5', daysToKeepStr: '

', numToKeepStr: '5')), pipelineTriggers([pollSCM('\* \* \* \* \*')])])

// Getting the code from the GitHub

stage('checkout code'){

git branch: 'development', credentialsId: '\*\*\*\*\*\*\*', url: '\*\*\*\*\*\*\*\*'

}

//Building the code in to packages by using maven

stage('build'){

sh "${mavenhome}/bin/mvn clean package"

//Executing the code quality report by using SonarQube

}

stage('execute sonarqube package'){

sh "${mavenhome}/bin/mvn clean sonar:sonar"

//Uploading the package into nexus

}

stage('upload buildartifact'){

sh "${mavenhome}/bin/mvn clean deploy"

//Deploying th application into Tomcat

}

stage('tomcat'){

sshagent(['\*\*\*\*\*\*\*\*\*\*']) {

sh "scp -o StrictHostKeyChecking=no target

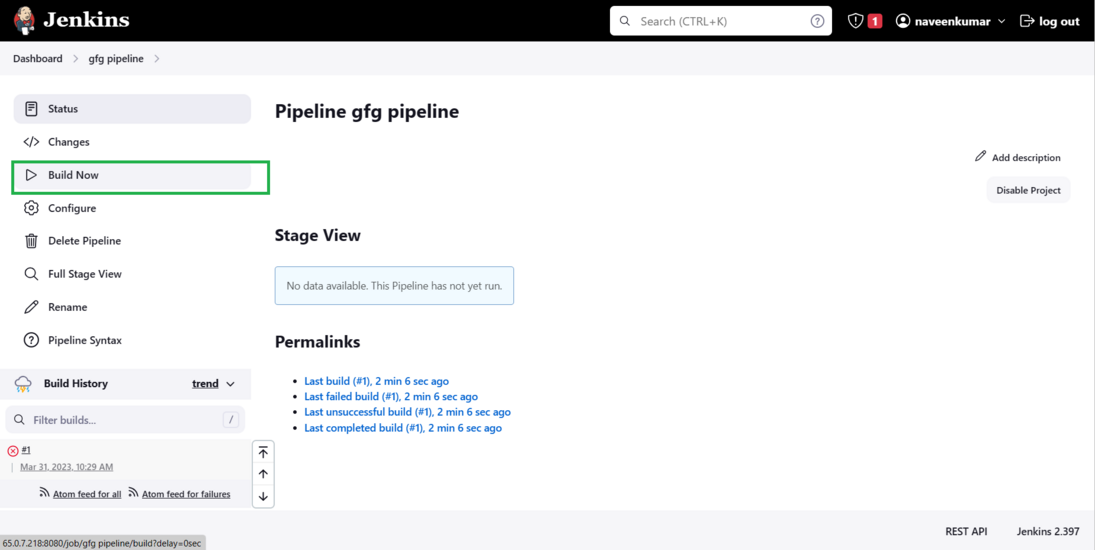
/maven-web-application.war ec2-user@\*\*\*\*\*\*\*:/opt/apache-tomcat-9.0.64/webapps/"

}

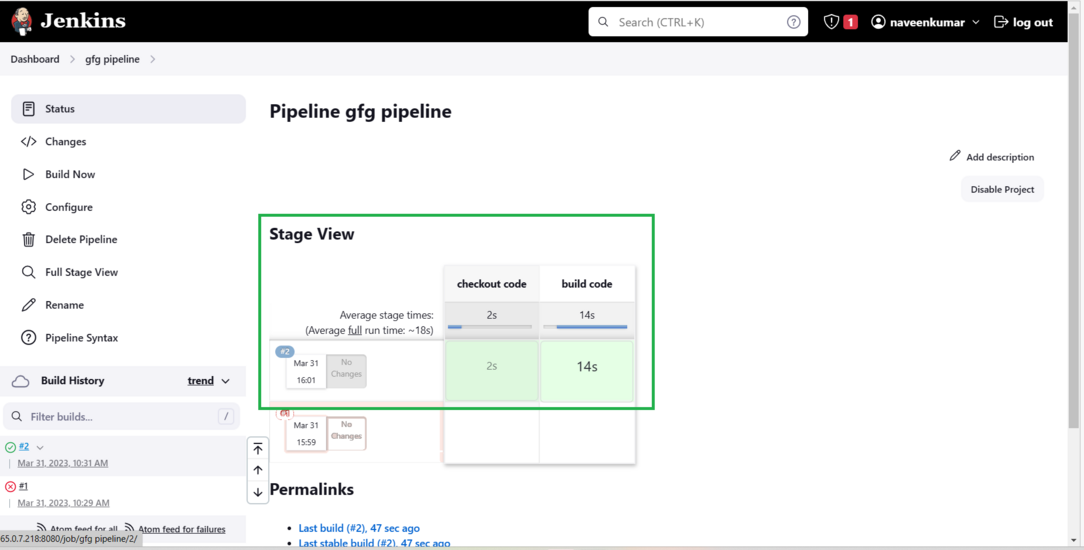
}

Replace all the values (\*) that have been mentioned above with your values.

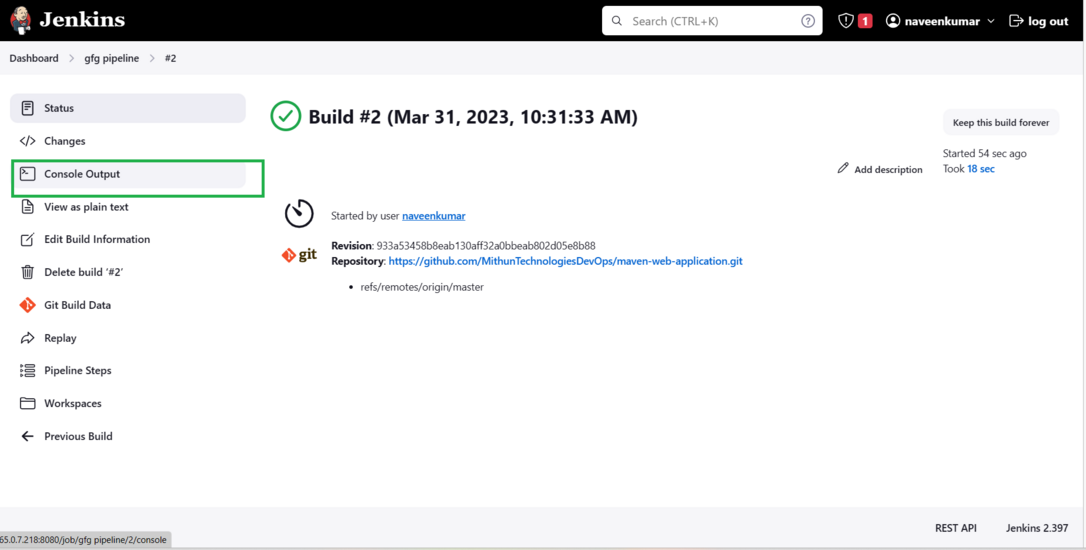
**Step 9.**After writing the pipeline is done click on save it will be directly redirected to the Dashboard of the project there we can use, the **“Build Now”**option to run the pipeline and check if it is successful or not, by using stage view or console output.



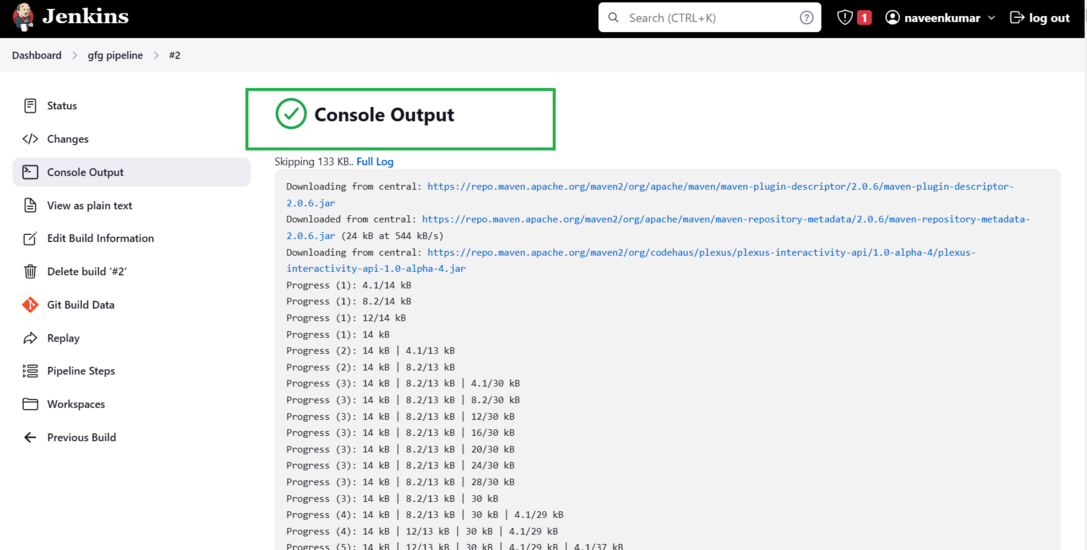
we can see the outcome of the pipeline in the stage view where as shown in the image below.

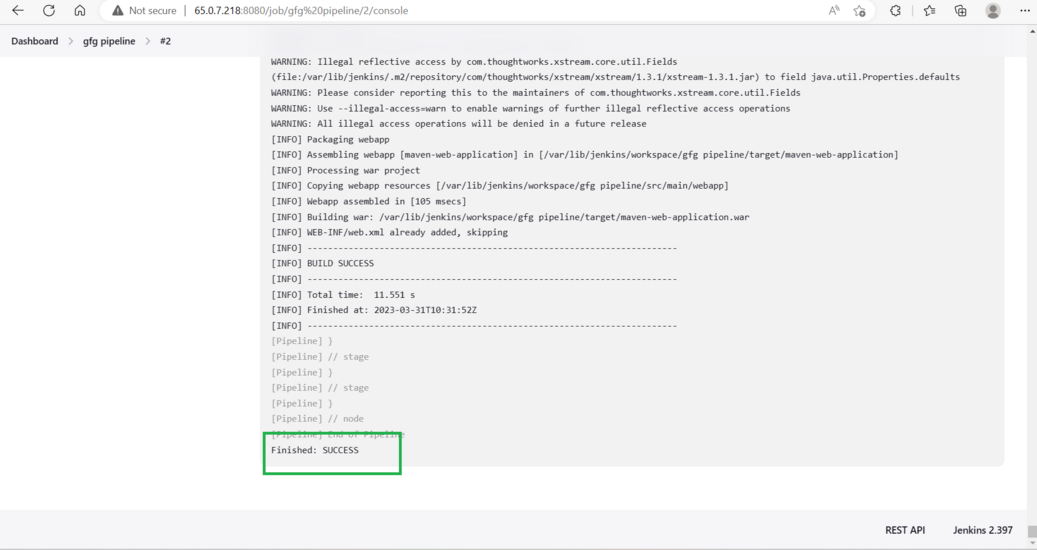


And we can also see the console output where we can see logs of each and every step which is performed.



Click on the console output to see the logs of each and every stage that is performed by using the pipeline.



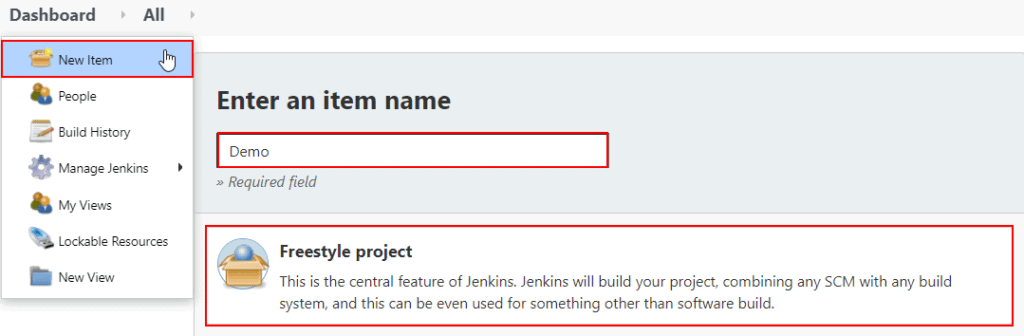


Scroll down to the end of the console output there we can see the status of the pipeline if it is “**Finished: success” The**pipeline which we have written was a success. If it marks as a fail we see the logs in the console we can find the reason why the stage was getting failed.

### Simple CI/CD Pipeline

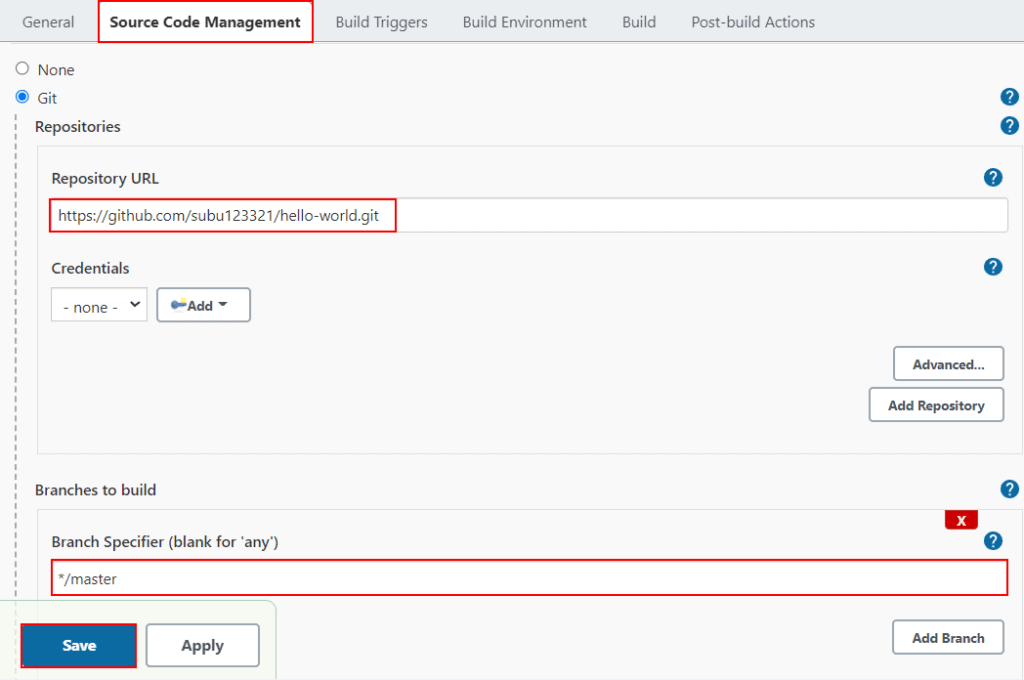
### Continuous Integration

**Step 1)** Create New Item, select Freestyle Project and provide a name to your item.



**Step 2)** Switch to the Source Code Management window and paste your Github repository link. Specify your branch name of the repository below and Save it.

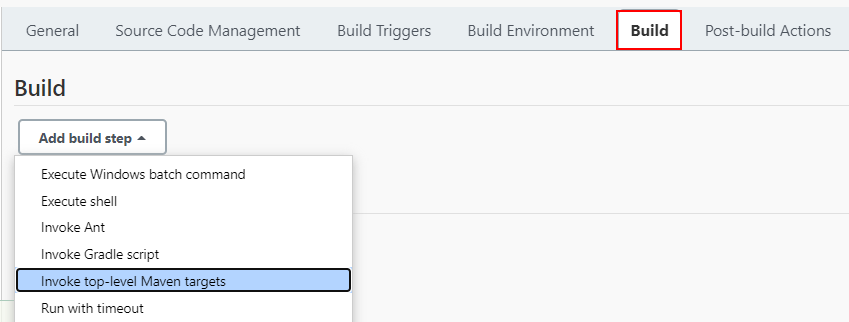
[Note: The above-linked Github repository ‘gihub.com/subu123321/hello-world‘ contains a ‘pom.xml‘ file used for Java compilation and generates a web app. It will be deployed to the server]



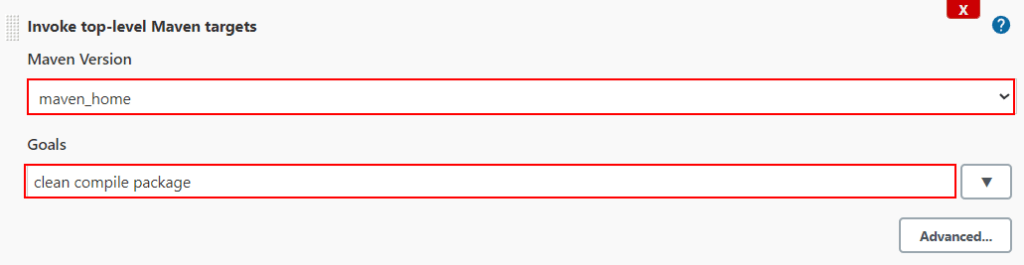
**Step 3)** Now click on Build Now button from the menu. With this step, all the repository files will be fetched by Jenkins. Click on Configure to go back to the same settings page.



**Step 4)** Click on Build Tab and select build step as ‘Invoke top-level Maven targets‘.



**Step 5)** Select your maven name from the drop-down menu. Fill the goals with the multiple jobs you need to perform and separate them with one space. These goals are available in your repository, and you need to invoke them using Maven. Save it and again click on the ‘Build Now‘ button from the menu as we did in the previous steps. Now the maven commands will be executed that will generate a war file.



**Step 6)**If you want to check the war file created in the previous steps, visit the workspace on your Jenkins dashboard or just run the directory commands in your server. Your directories and project name can vary, so you can use the ‘ls‘ command to see the list inside that directory and also keep in mind the directory name is case sensitive.

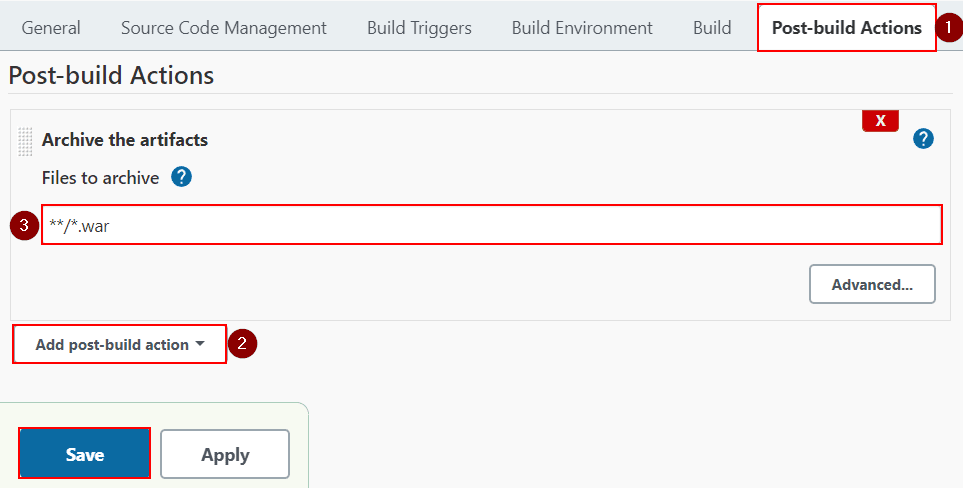
cd /var/lib/jenkins/workspace/

ls

cd /Demo

cd /webapp/target

**Step 7)**Now go back to Configure and visit the ‘Post Build Actions‘ tab. Click the drop-down and select ‘Archive the Artifacts‘ from the options. In the field, write down ‘\*\*/\*.war‘ as shown in the image below. It will fetch all the directories and get the war file wherever it is present. Click again on Build Now button, and you will now see the Artifacts in the Jenkins dashboard.



### Continuous Deployment

**Step 8)** We need to install Apache Tomcat, and for this, you need to visit the [Tomcat Download](https://tomcat.apache.org/download-90.cgi) page. In the core section, hover over the ‘tar.gz’ link and copy it. Now, use the below commands in your server one by one.

* First, four commands will create one temporary directory and user group to access the file. Here, use the command curl -O ‘paste tomcat download link‘ as shown in the command below.
* Use further commands to create a tomcat directory and extract the gzip file. Just cross-check the version number of the Tomcat that you are downloading and extracting.
* Now the permission of the files needs to be configured with the below commands. In the last command, replace it with your username by which you are accessing the server.

sudo groupadd tomcat

sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat

cd

cd /tmp

curl -O https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.54/bin/apache-tomcat-9.0.54.tar.gz

sudo mkdir /opt/tomcat

sudo tar xzvf apache-tomcat-9.0.54.tar.gz -C /opt/tomcat --strip-components=1

cd /opt/tomcat

sudo chgrp -R tomcat /opt/tomcat

sudo chmod -R g+r conf

sudo chmod g+x conf

cd ..

sudo chown -R jenkinsuser:jenkinsuser tomcat/

**Step 9)** We need to update the port number from **8080** to **8090** in the server.xml file. We are updating it as this port number is already in use by Jenkins, and we have created 8090 in Azure VM for Tomcat. Use the below commands to edit the file. When you enter the file, click the **INSERT** button to edit. Now search for a similar code, as shown in the image below. Update the port number to 8090. To save the file, press the Esc key, type :wq and click on Enter button.

cd

cd /opt/tomcat/conf

vi server.xml

Tomcat Port Number

**Step 10)** Similarly, we need to edit the ‘tomcat-users.xml’ file to update the roles that enable us to deploy files using Tomcat. In the file before tomcat-users ending code, paste the below roles code. To save the file press, the Esc key, type :wq and press Enter.

cd

cd /opt/tomcat/conf

vi tomcat-users.xml

<role rolename="manager-gui"/>

 <role rolename="manager-script"/>

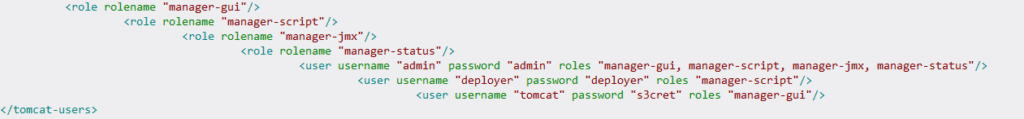
 <role rolename="manager-jmx"/>

 <role rolename="manager-status"/>

 <user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx, manager-status"/>

 <user username="deployer" password="deployer" roles="manager-script"/>

<user username="tomcat" password="s3cret" roles="manager-gui"/>



**Step 11)** We also need to update the context.xml file to remove the IP restriction. Use the same steps to edit the file with the below commands. Remove all the content present inside ‘context‘ as shown in the image below and save it.

cd

cd /opt/tomcat/webapps/manager/META-INF

vi context.xml

Context Update

**Step 12)** Now, all the files are edited successfully. To update the Tomcat, we need to restart the system to accept all our changes. Use the below commands in the server for shutdown and startup of Tomcat. With this step, Tomcat is ready to deploy our container.

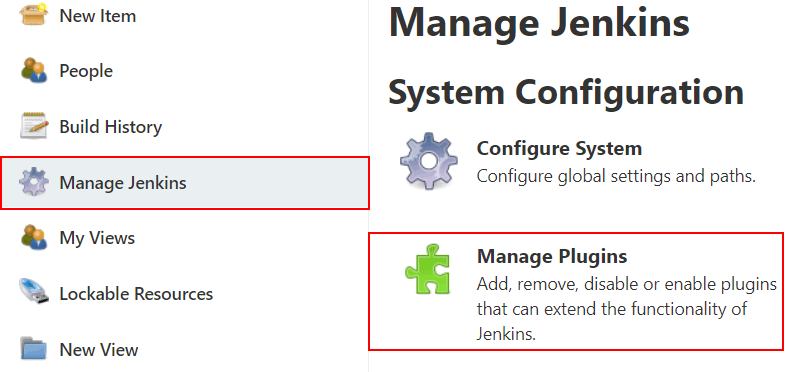
cd

cd /opt/tomcat/bin/

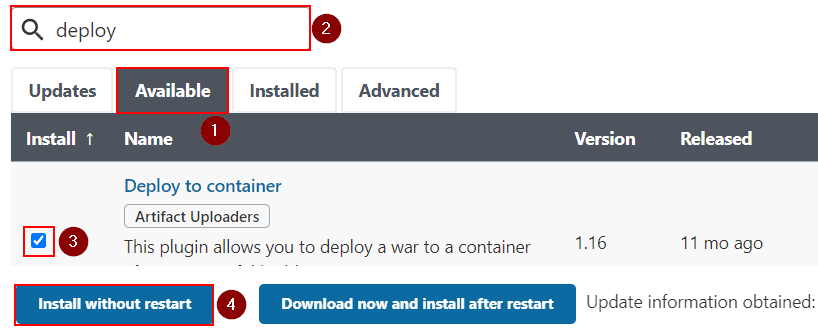
./shutdown.sh

./startup.sh

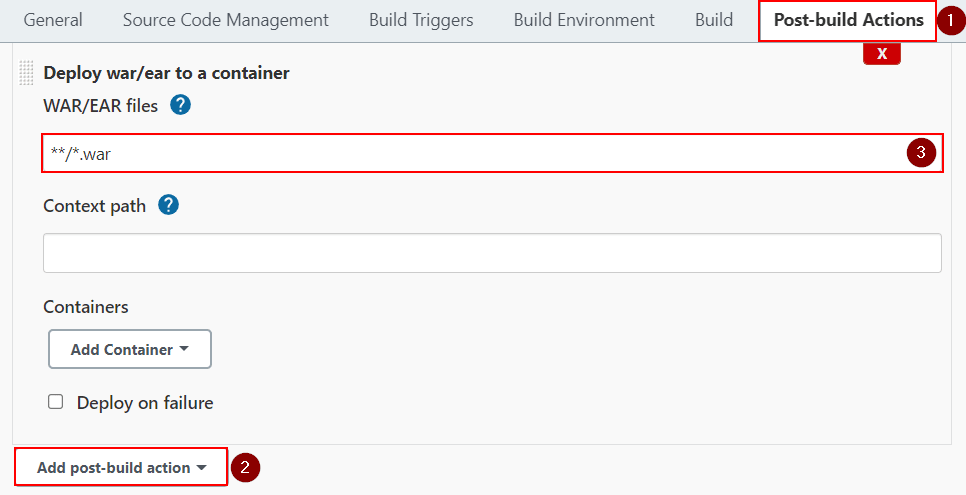
**Step 13)** In the Jenkins Dashboard, click on Manage Jenkins and then visit Manage Plugins.



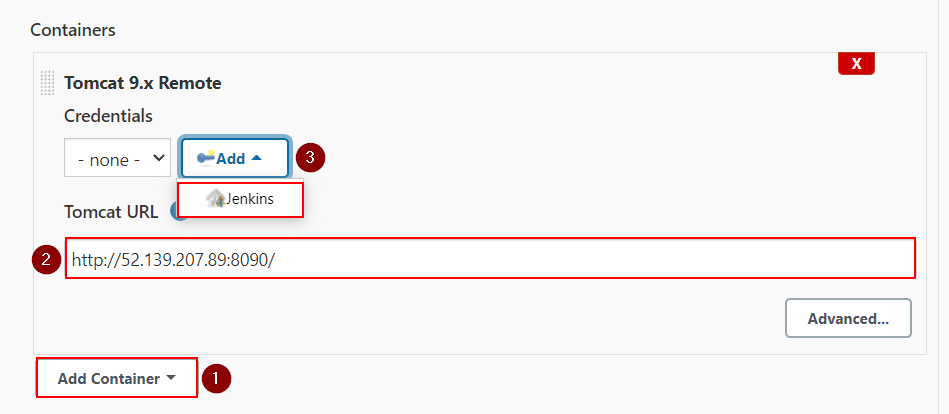
**Step 14)** Click on the Available tab and search for the ‘Deploy to Container‘ plugin. Select the plugin and click on the ‘Install without restart‘ button.



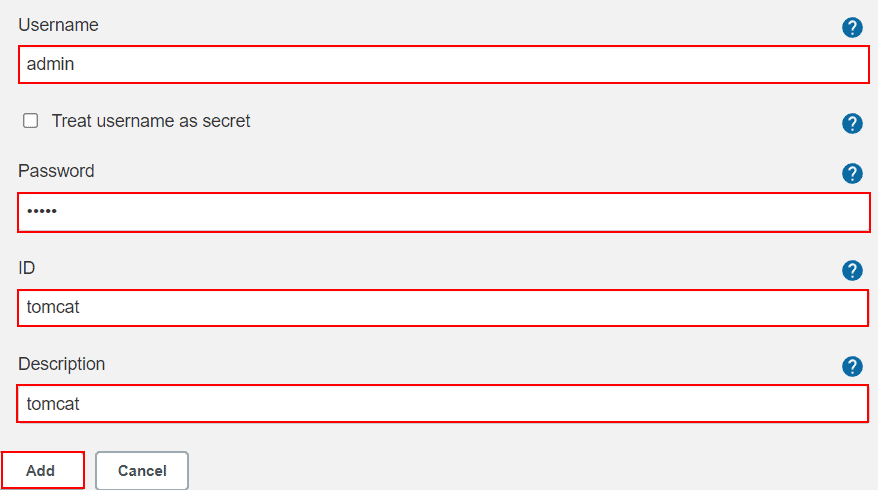
**Step 15)** Go back to your Configure window and select the Post-build Actions tab.  Select the ‘Add post-build action‘ drop-down button and select the ‘Deploy war/ear to a container‘ plugin. Fill the same path of your war file here, as shown in the image below.



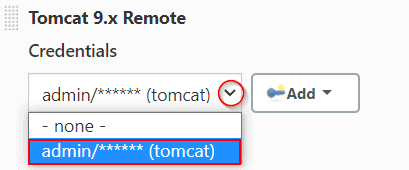
**Step 16)**Now click on the ‘Add Container‘ button and select the ‘Tomcat 9.x Remote‘ as we are using version 9 of the Tomcat. Fill in the URL of the same virtual machine with the new port number **8090**. On the credentials drop-down button, select Jenkins.



**Step 17)**In this window, fill in the username and password that we have used in the ‘tomcat-users.xml‘ file **(**in Step 10**)**. Fill in the ID, description and click on the button ‘Add‘.



**Step 18)**Click on the credentials drop-down button and select the recently created credential. Save all the settings and again click on the ‘Build Now‘ button from the Jenkins dashboard. If the Build is successful, the war file will get deployed.



**Final Step)** Visit the address of your virtual machine, and you will see your deployment on the page. In our case, the URL is ‘52.139.207.89:8090/webapp/‘ with port number 8090 that we created for Tomcat.