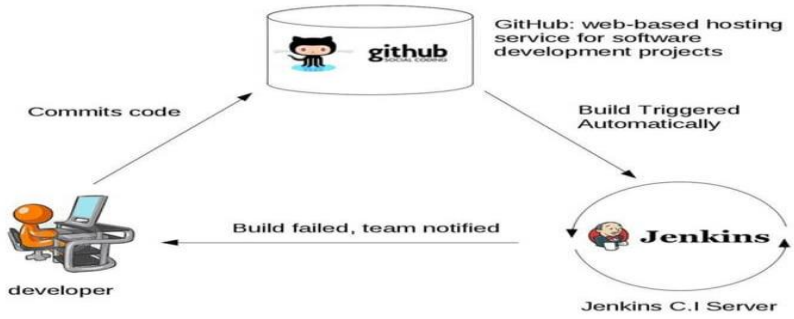
Jenkins is an open source Continuous Integration Server written in Java that allows continuous development, test and deployment of codes.

Continuous Integration is a process of integrating code changes from multiple developers in a single project many times. The software is tested immediately after a code commit.
With each code commit, code is built and tested. If the test is passed, the build is tested for deployment.

If the deployment is successful, the code is pushed to production.



- Jenkins is being managed by the community which is very open. Every month, they hold public meetings and take inputs from the public for the development of Jenkins project.

- So far around 280 tickets are closed, and the project publishes stable release every three months.

- As technology grows, so does Jenkins. So far Jenkins has around 320 plugins published in its plugins database. With plugins, Jenkins becomes even more powerful and feature rich.

- Jenkins tool also supports cloud-based architecture so that you can deploy Jenkins in cloud-based platforms.

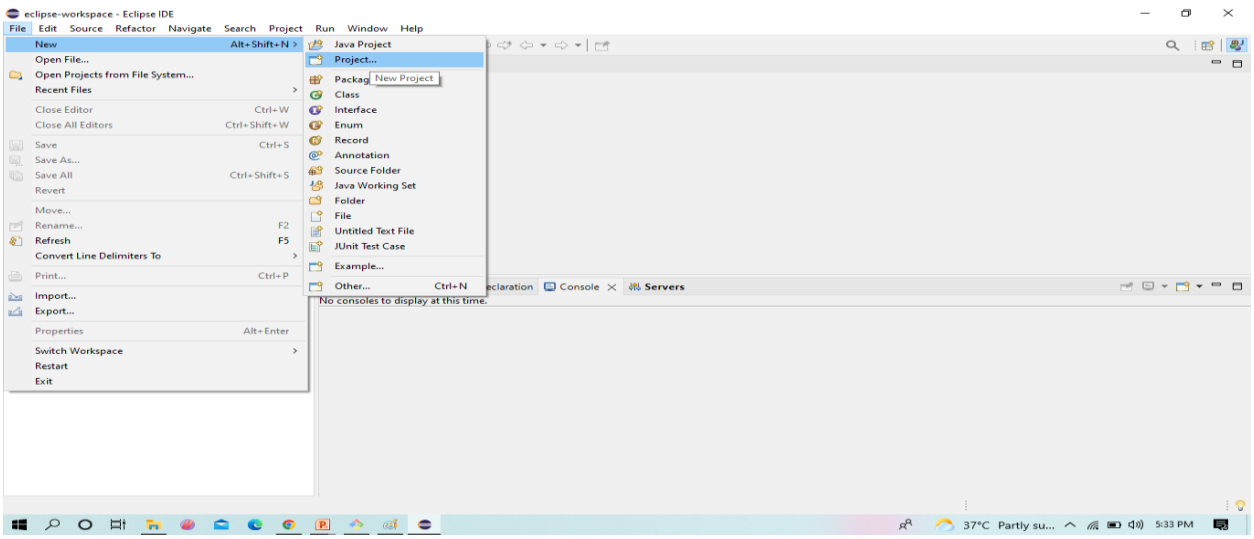
- The reason why Jenkins became popular is that it was created by a developer for developers.
- Its interface is out dated and not user friendly compared to current UI trends.
-

- Though Jenkins is loved by many developers, it's not that easy to maintain it because Jenkins runs on a server and requires some skills as server administrator to monitor its activity.

-
- One of the reasons why many people don’t implement Jenkins is due to its difficulty in installing and configuring Jenkins.
-
- Continuous integrations regularly break due to some small setting changes. Continuous integration will be paused and therefore requires some developer attention.

Now let us demonstrate simple maven web project

Go to- > File - > New -> Maven Project



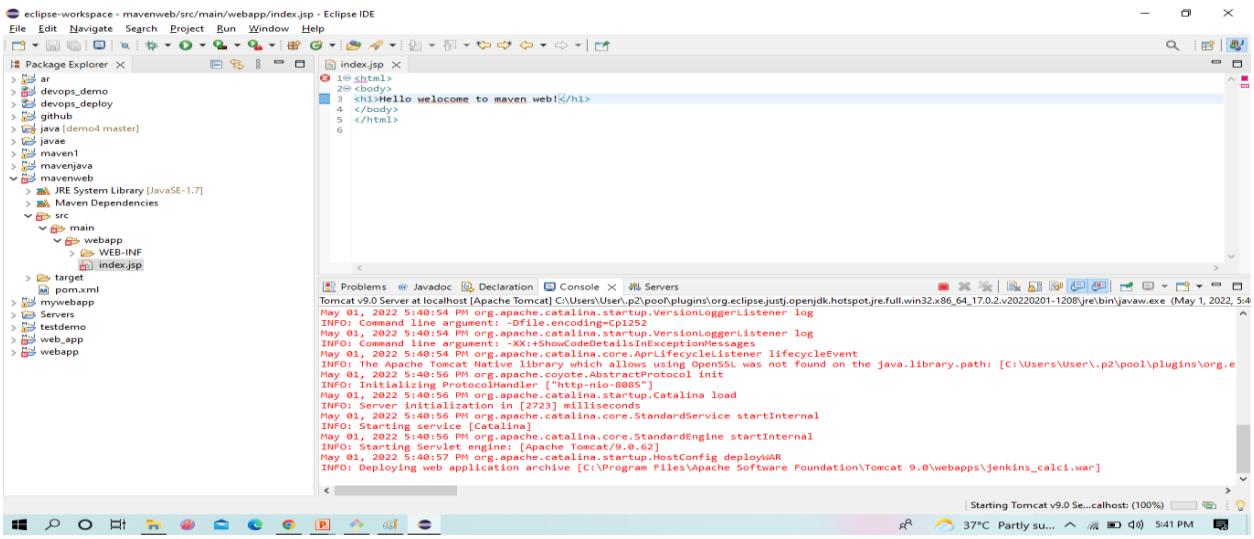
Select the Maven Project

Click -> Next

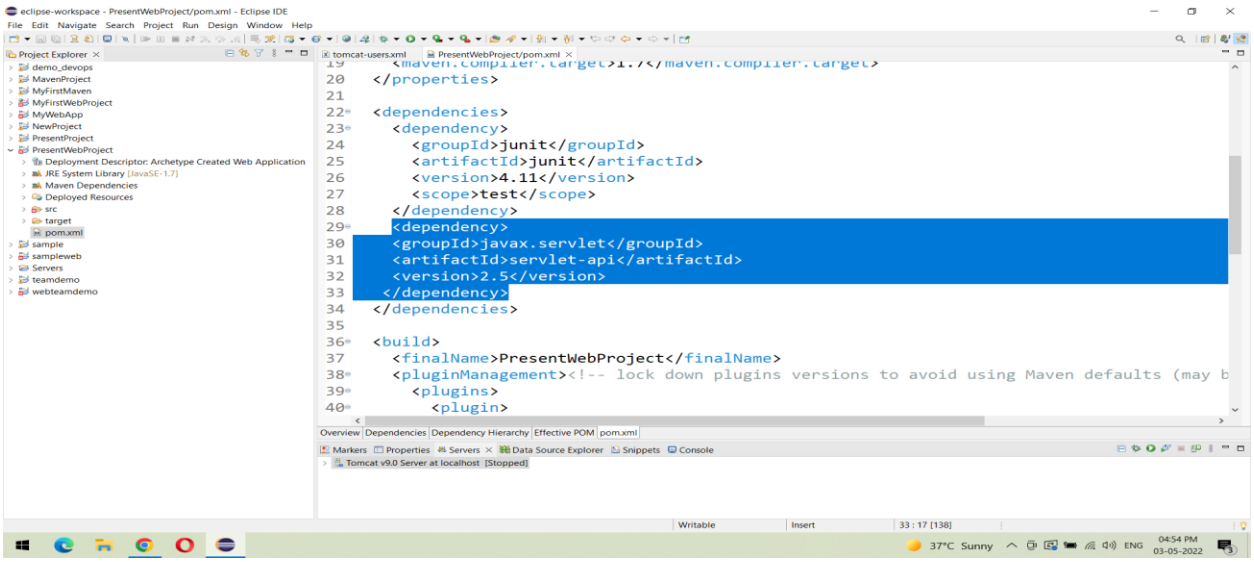
Select the org.apache.maven.archetypes with webapp archetype and click Next

Provide the Group id , Artifact id(file nam) and click Finish, Maven web Project is created

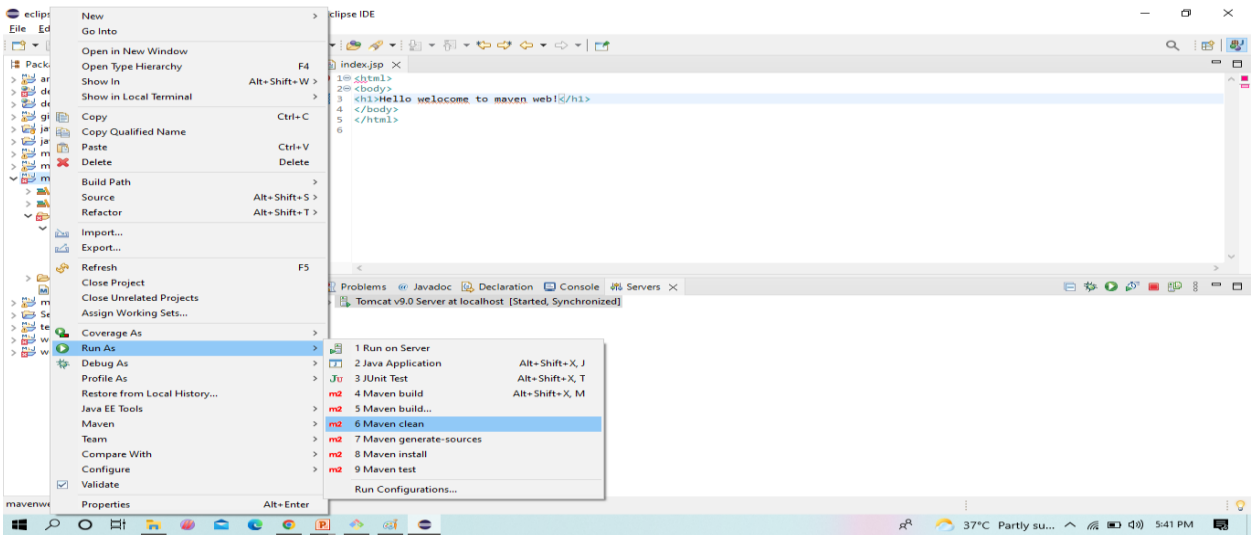
Here the project created , Check for index .java file containing Hello Welcome to maven web



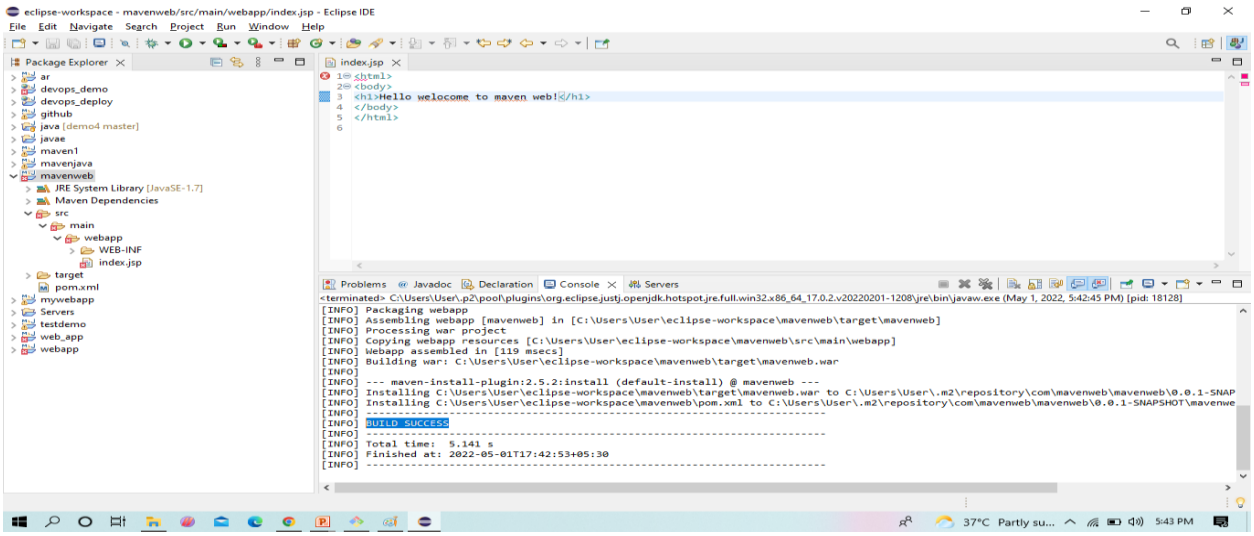
In pom.xml add the dependencies as shown below.



To run the project right click and Run as Maven Clean - It clears out the existing classes previous one



Here we can see the console for Build Success.



Now again run the project right click and Run as Maven Install – to add artifacts to the project

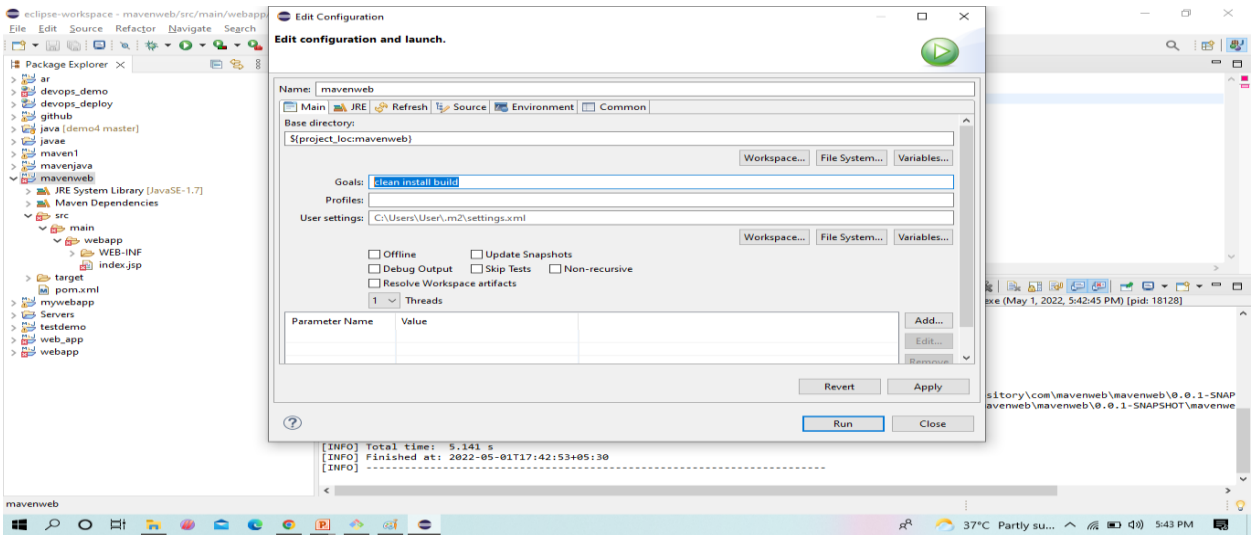
Here we can see the console for Build Success

Again run the project Maven test – to test the project

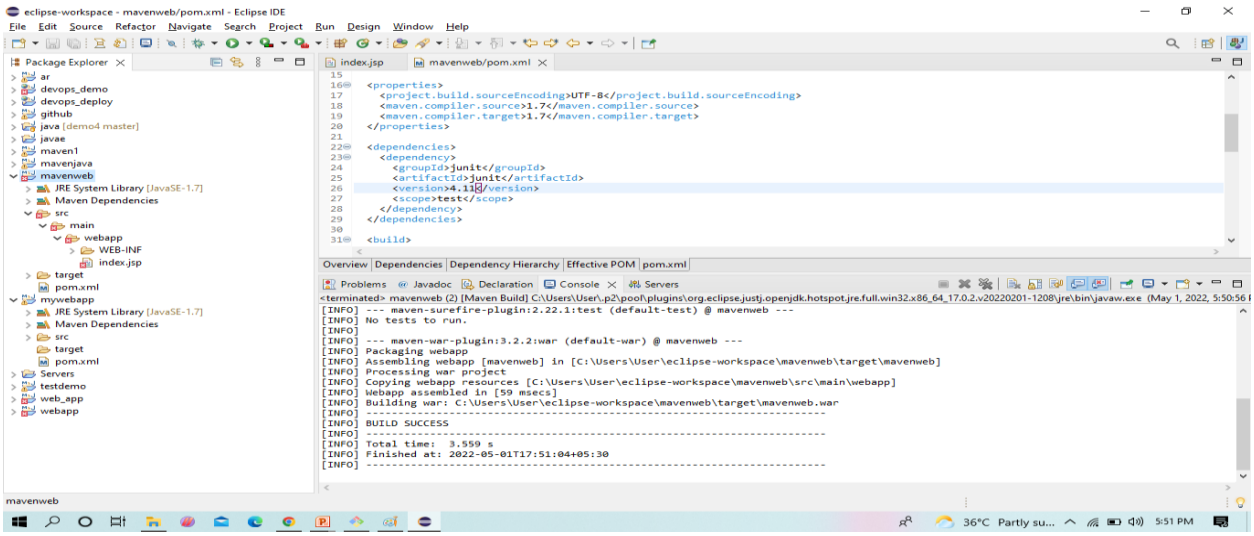
Here we can see the console for Build Success

Now run the project Run as Maven Build – it creates JAR/WAR files to the project

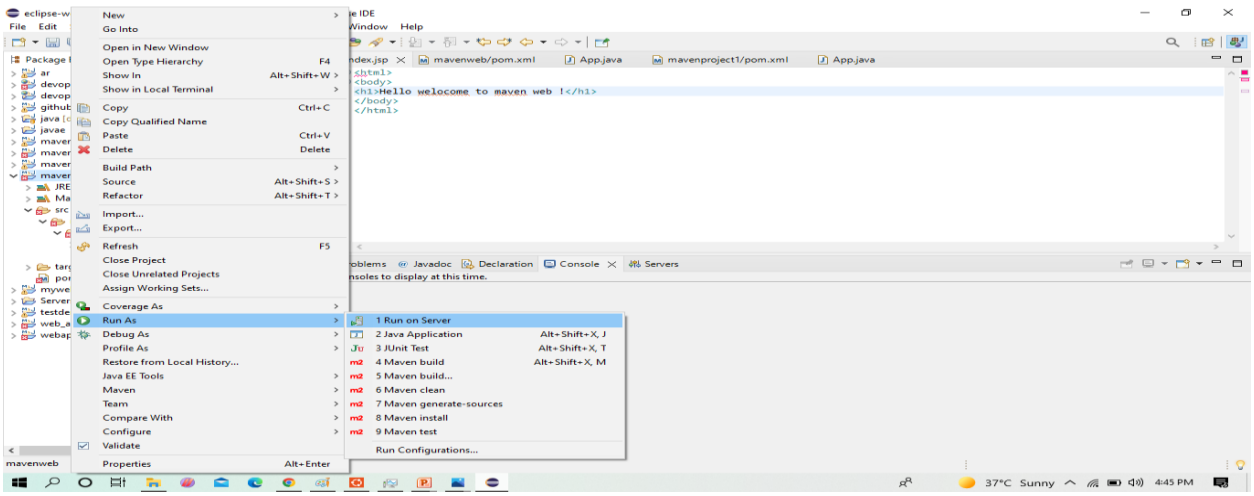
Here specify the goals -> they clean install test or Package ,click -> Apply & Run



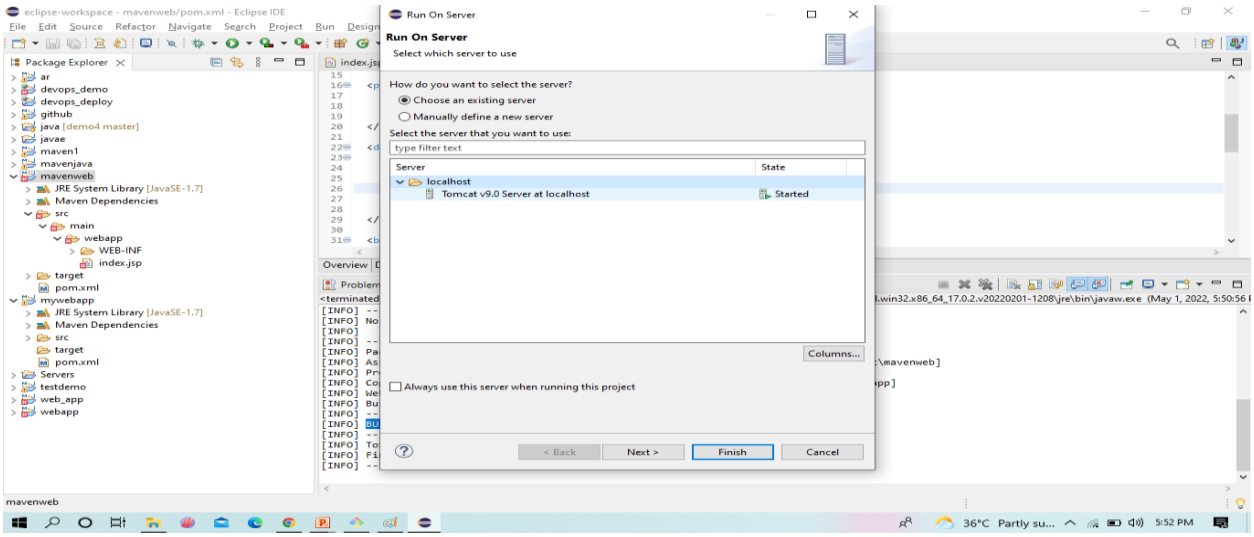
We see the console output



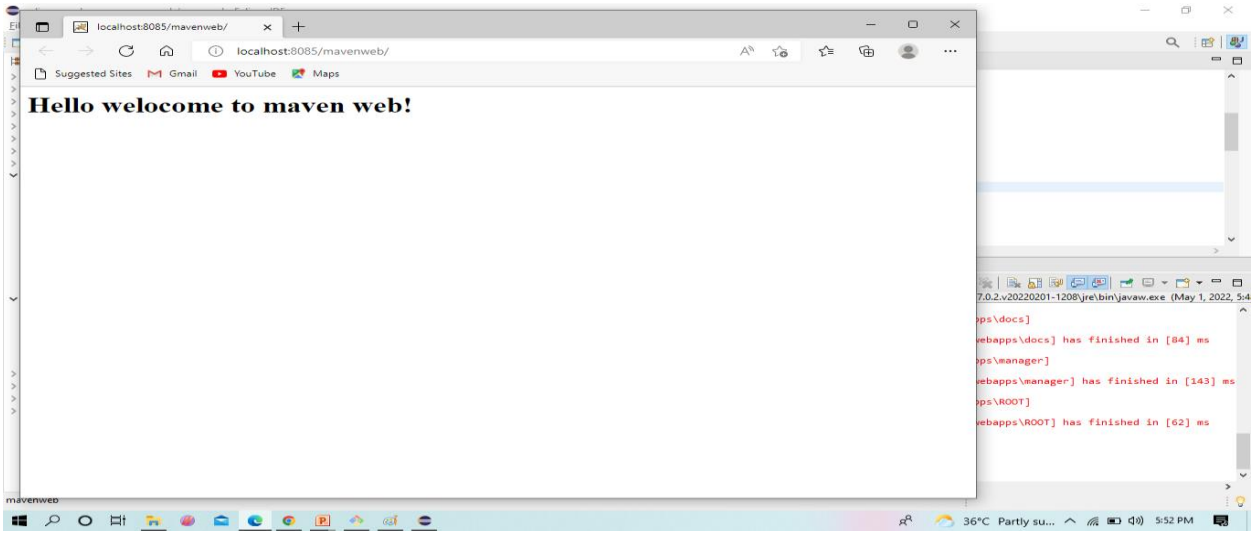
Now run the project as Run on server



Here we have to select the tomcat server ,Click -> Finish



Here we can see output here



Now copy the path from maven

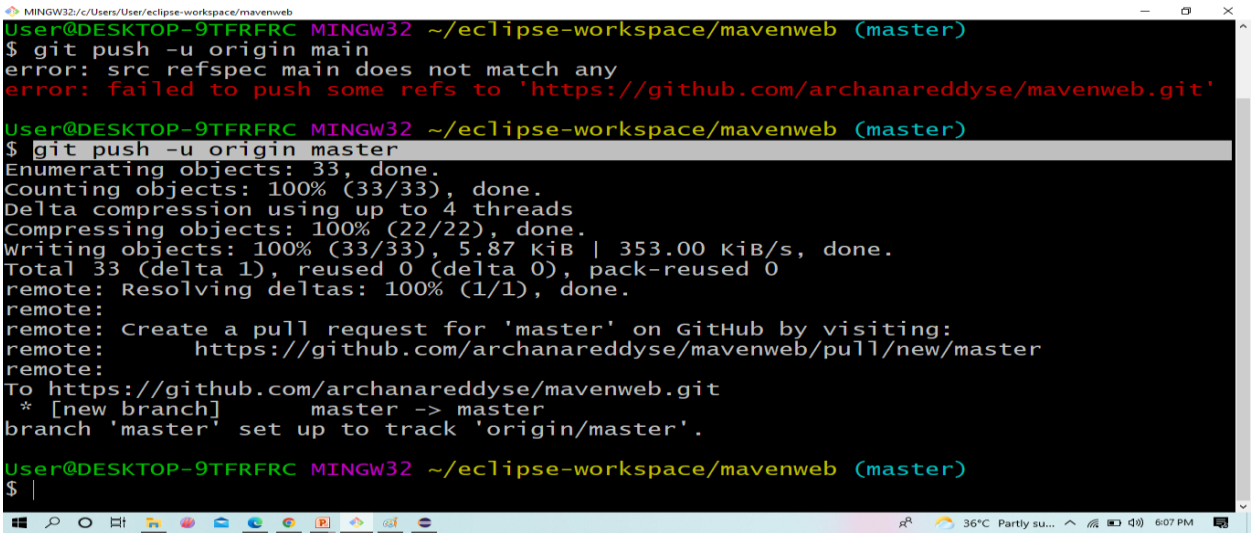
Go to gitbash and change directory into that path

$ git init

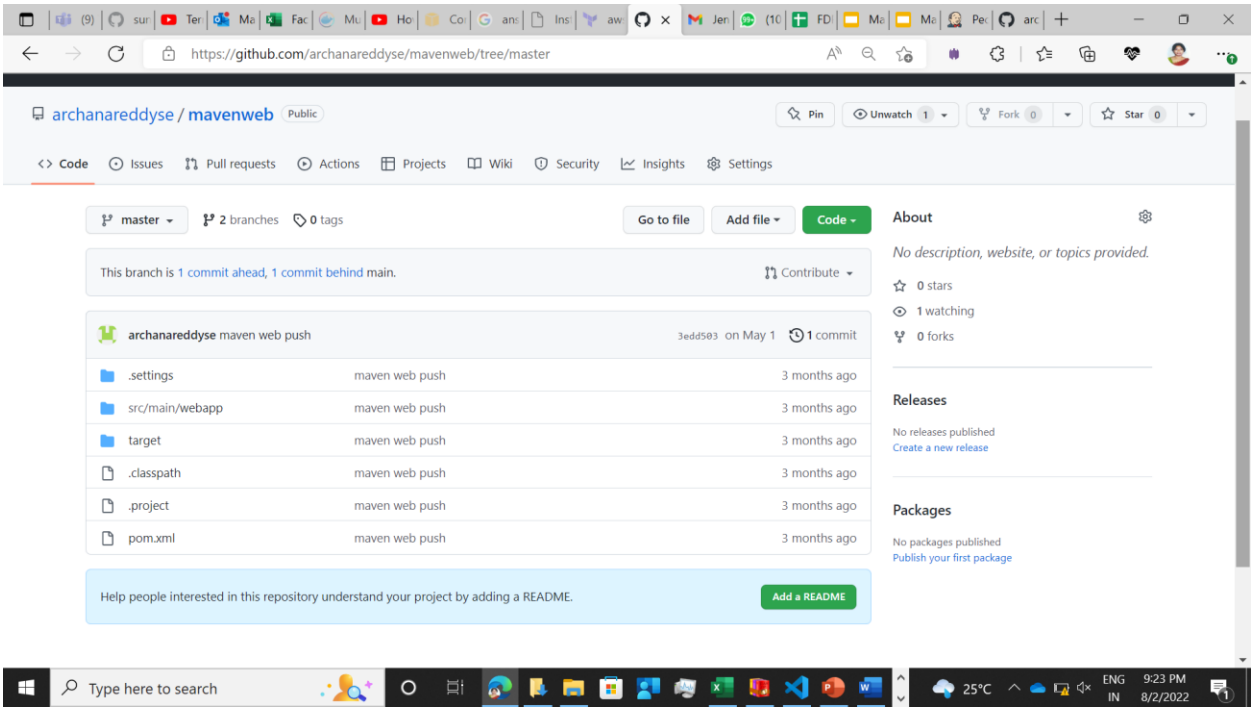
$ git add.

$ git commit -m

Now push into the git hub .



https://github.com/archanareddyse/mavenweb.git



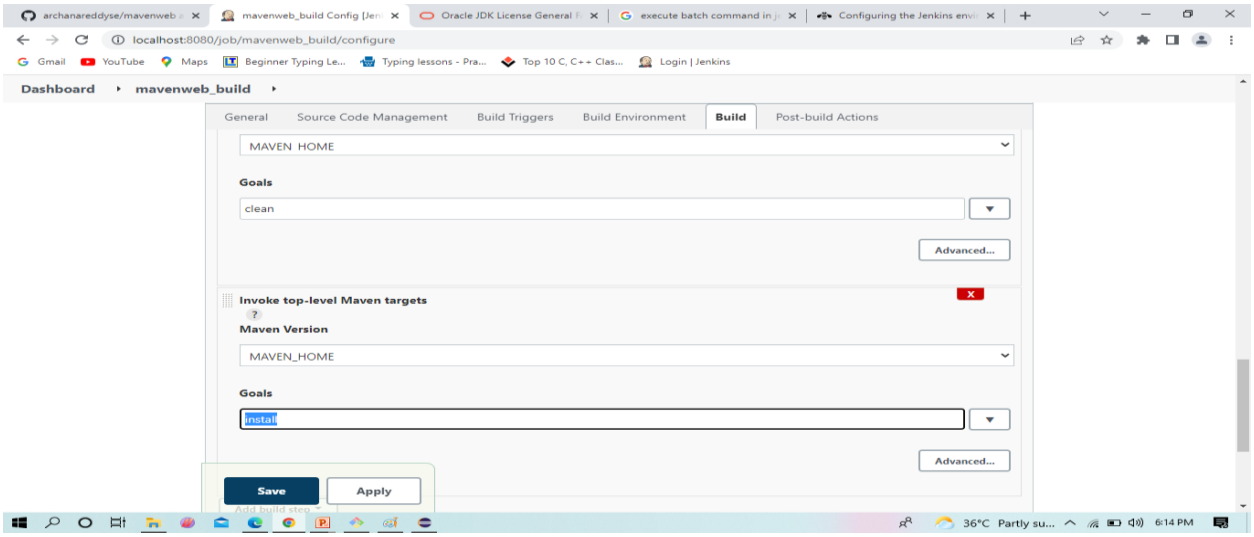
Now come to jenkins ,here we will create the new item

Name it, and select as -> freestyle project ,click on-> ok.

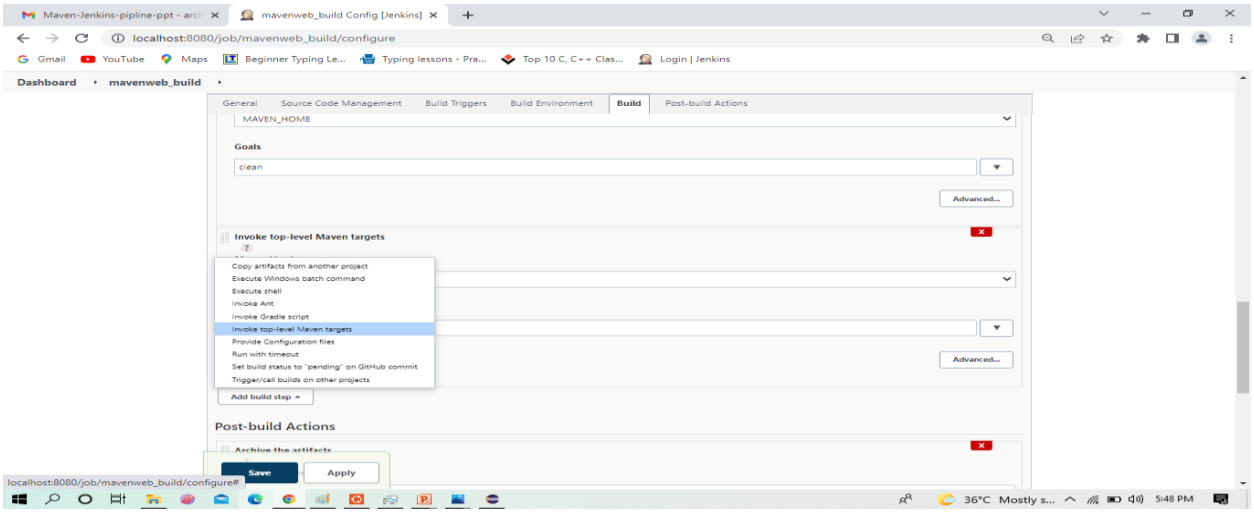
Go to source code management click on git

Paste the url of git https://github.com/archanareddyse/mavenweb.git

Check master /main here

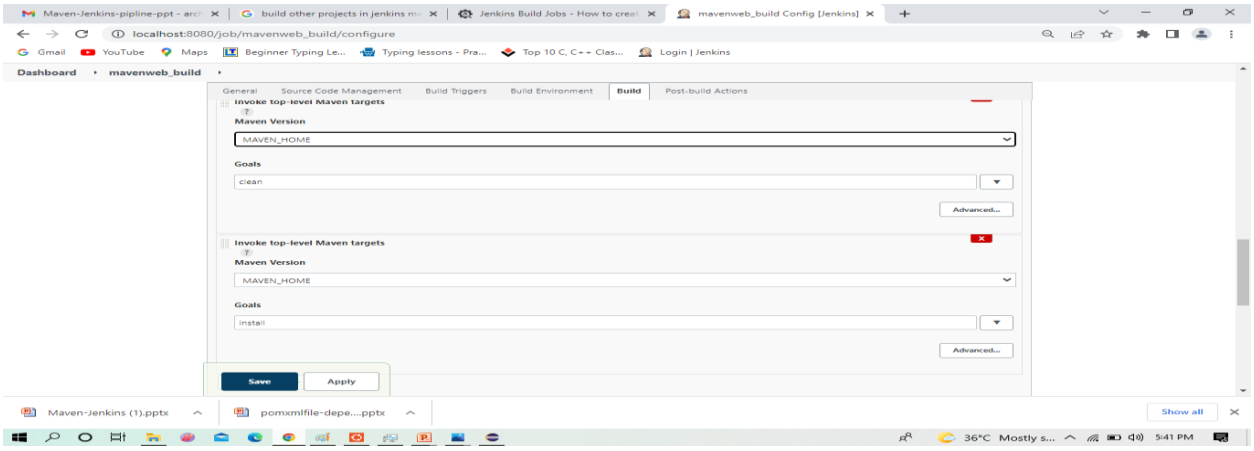


Now in Build, Invoke top-level Maven targets



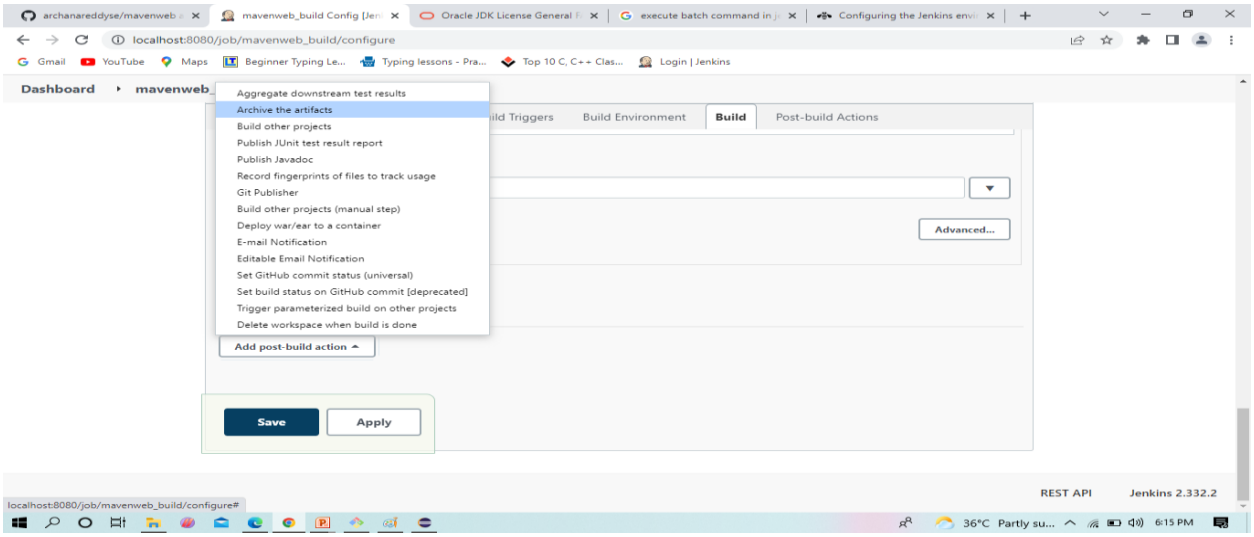
Select the Maven path which is already set in the global credentials in Manage Jenkins

Now we follow the goals of as starting with clean and install

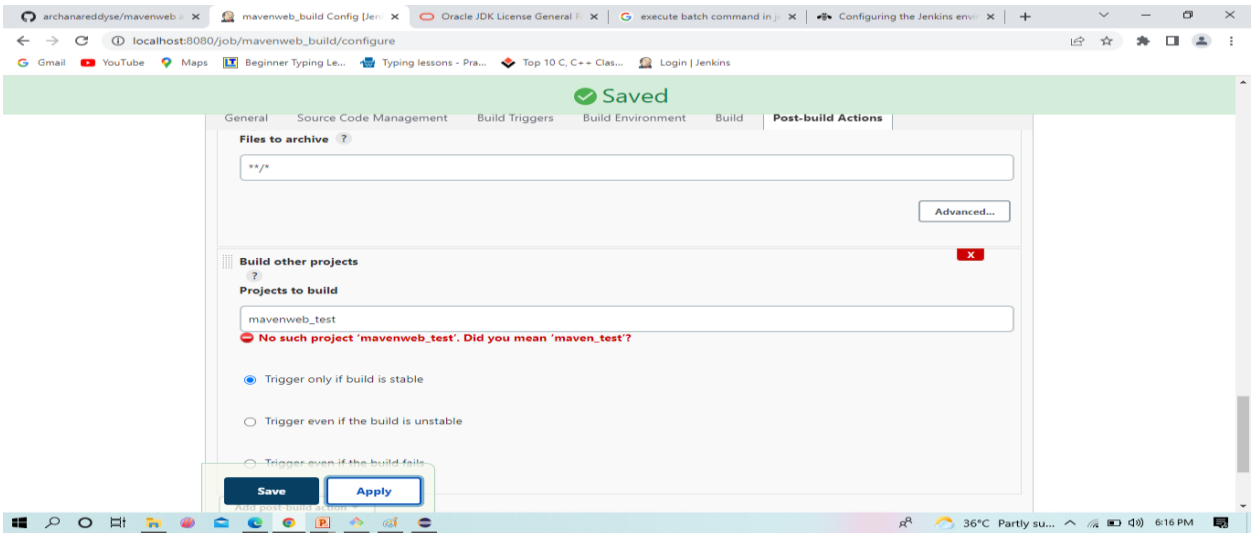


Now we go for post build select the Archive artifacts -in Jenkins is a feature that allows us to store the output files after we build the project with Jenkins.

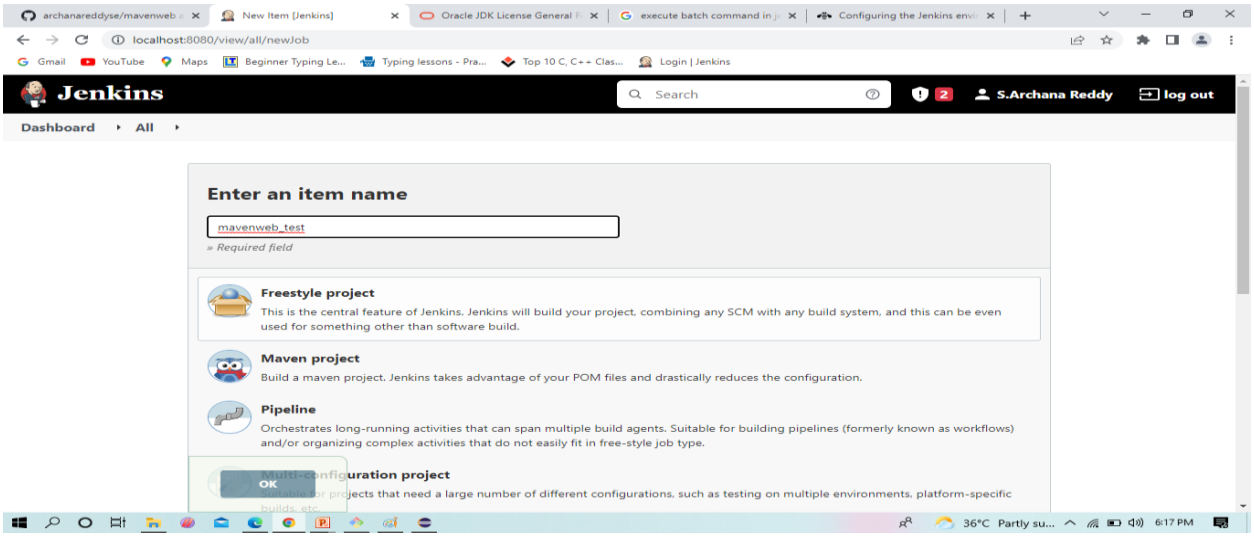
To archive the artifacts we go for **/*.



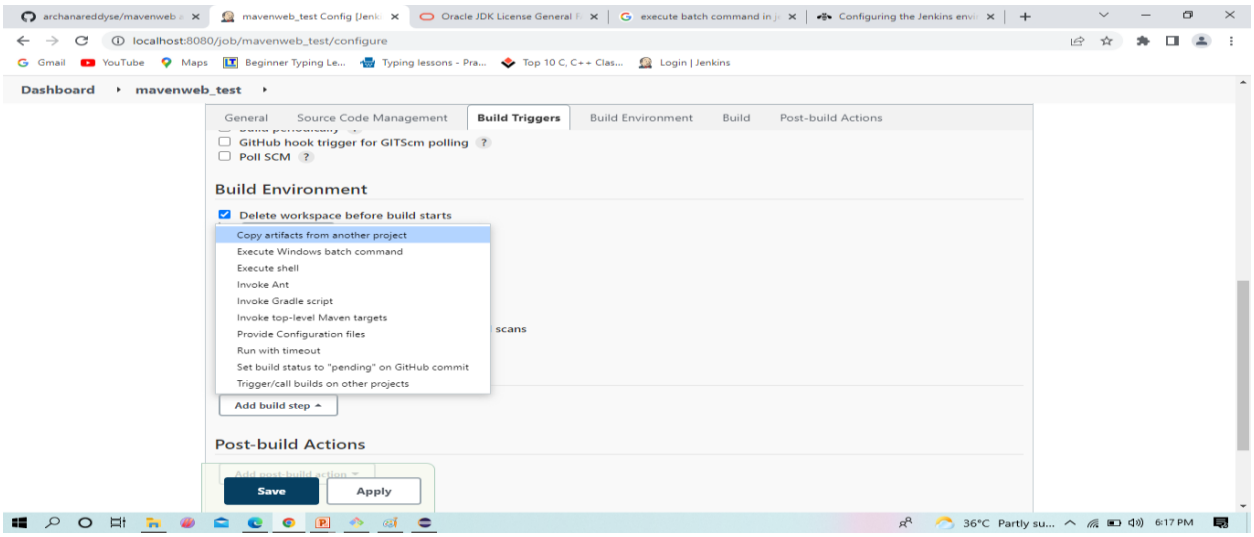
Now we here select the build other projects, where we will create a test project which will be triggered by the build project, click on Apply and Save



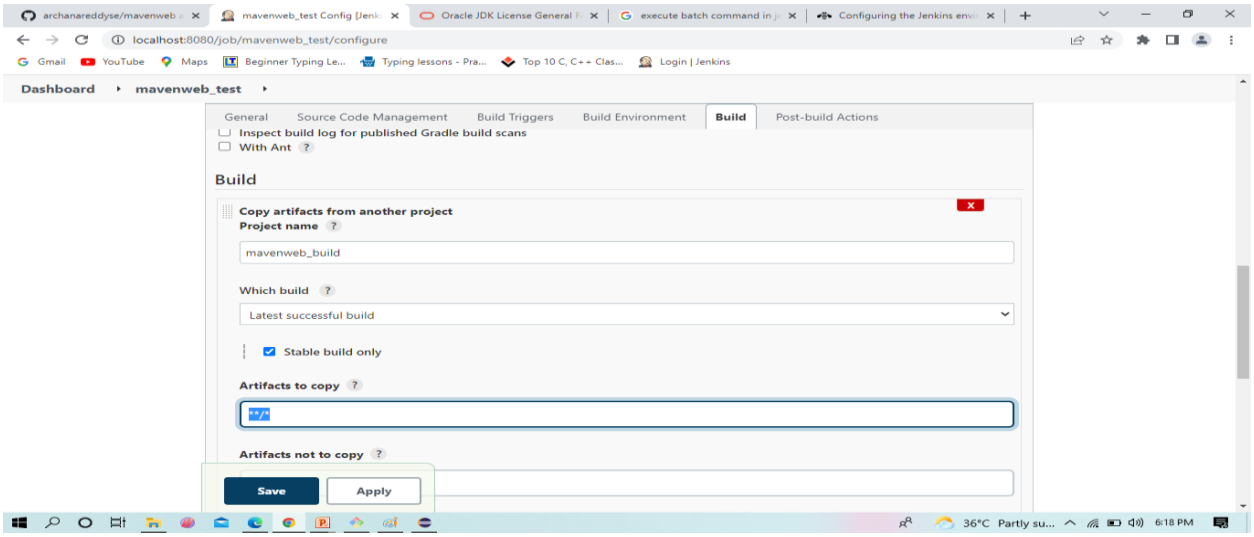
Create a new freestyle project test as shown and click ok



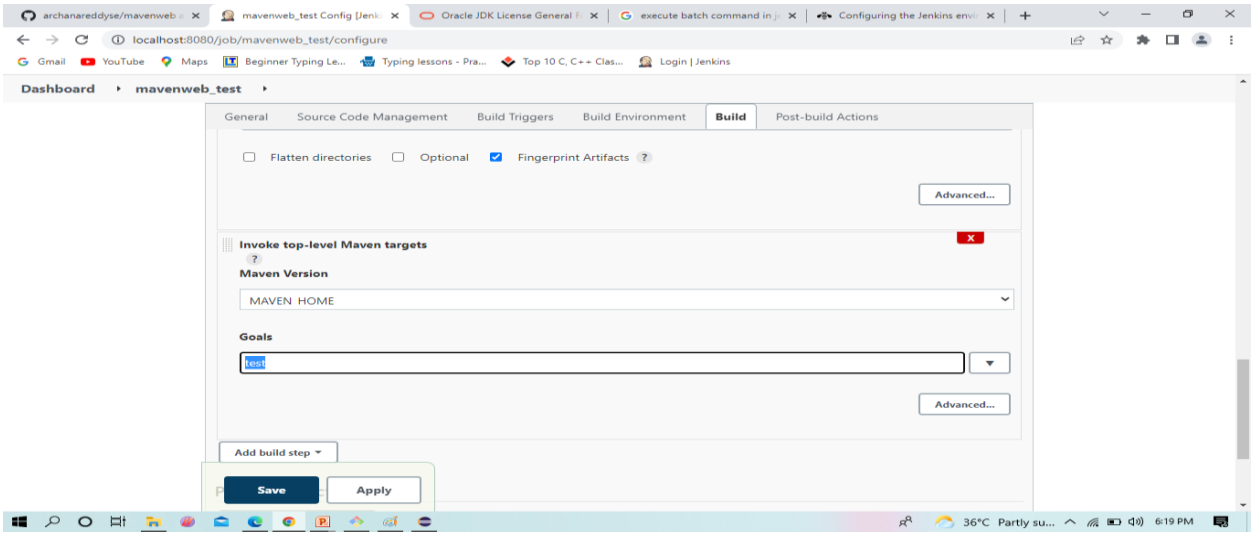
In Build environment check the box as shown below, this is to discard old builds
To forward the artifacts of the previous project to the current test project, select copy the artifacts from another project in Build as shown



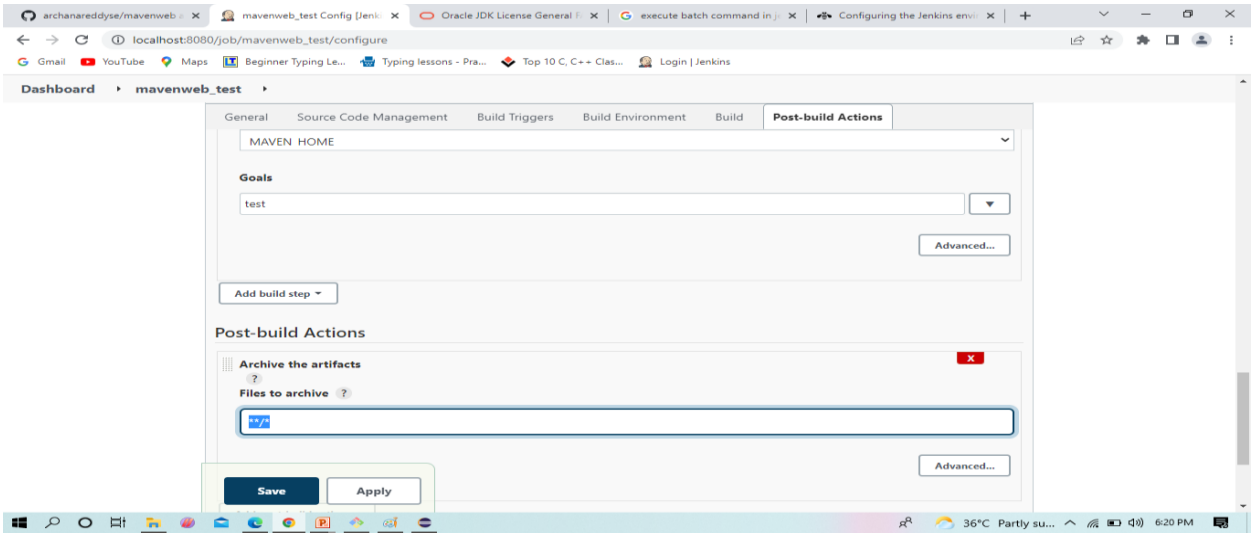
Give the name of the project from which we want to copy the artifacts and check the box ->stable build only->to copy all the artifacts type **/*



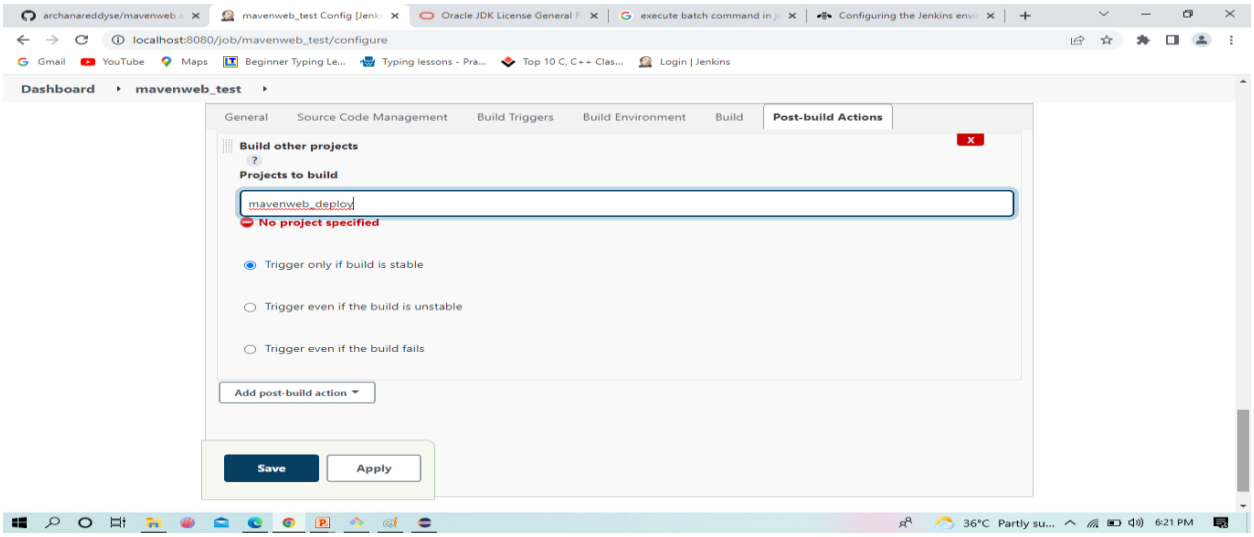
Now in Build, Invoke top-level Maven targets, choose the Maven Path and select goal as test



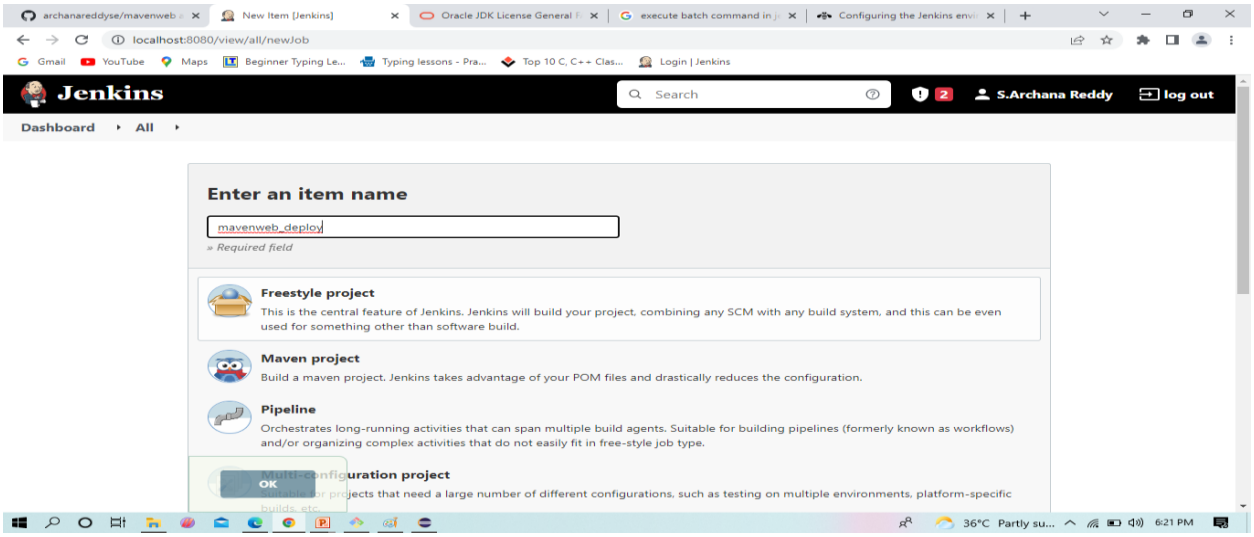
To archive the artifacts we go for **/*



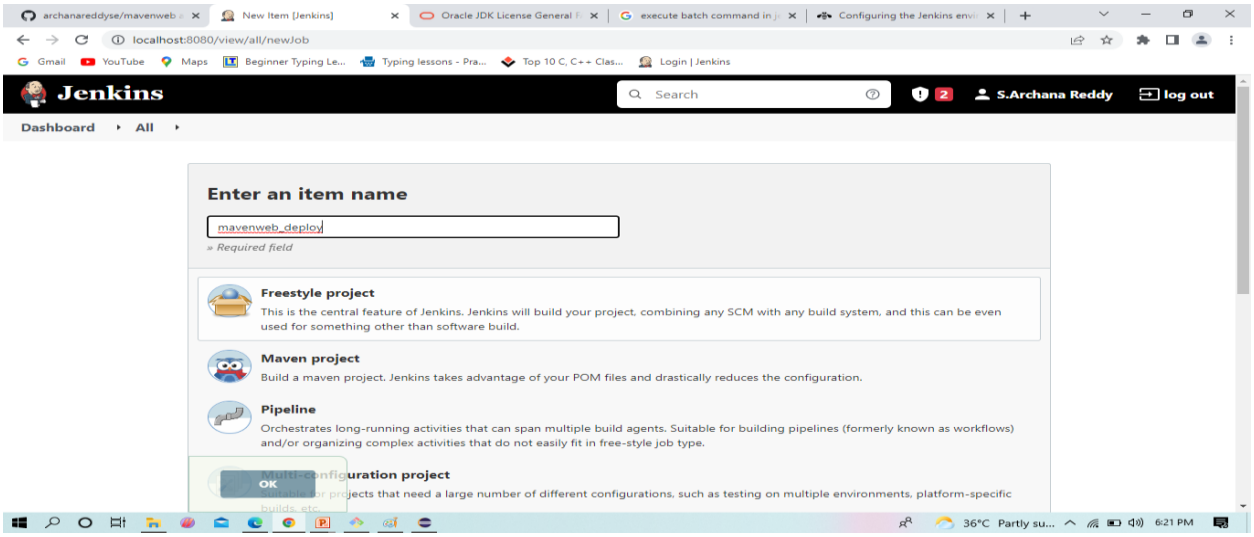
Now we here select the build other projects, where we will create a deploy project which will be triggered by the test project, click Apply and Save



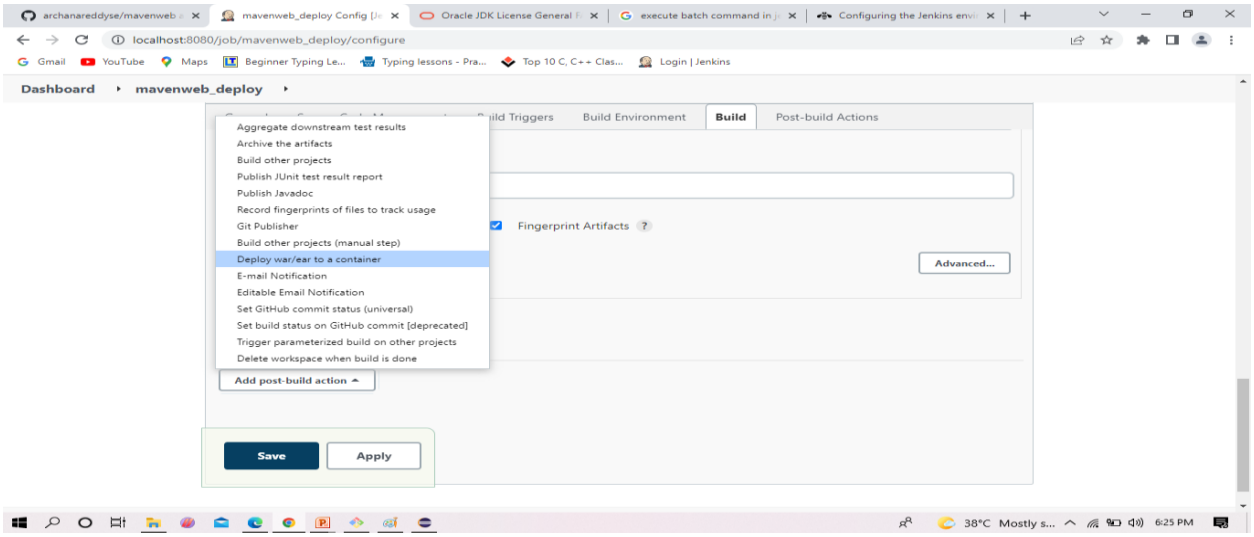
Create a new freestyle project test as shown and click ok



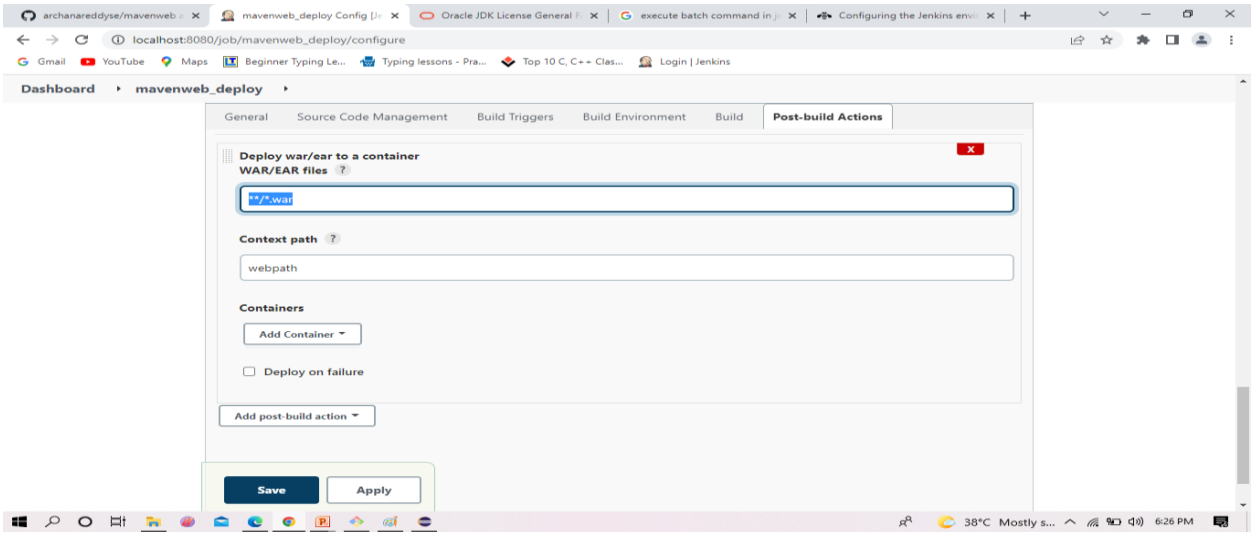
Give the name of the project from which we want to copy the artifacts and check the box ->stable build only->to copy all the artifacts type **/*



Now here we go for Add post –build action where we select the Deploy war/ear to a container.
This **plugin** takes a **war/ear** file and deploys that to a running remote application server at the end of a build.



Deploy war/ear to a container takes the artifacts as **/*.war.

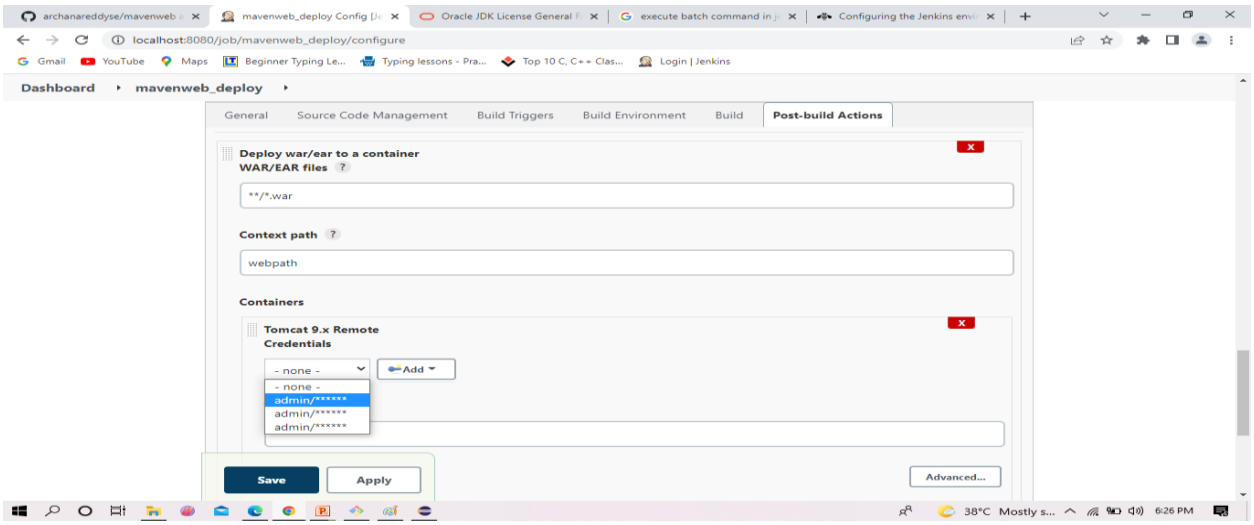


Here we select the tomcat version .

Here we add the credentials of tomcat .

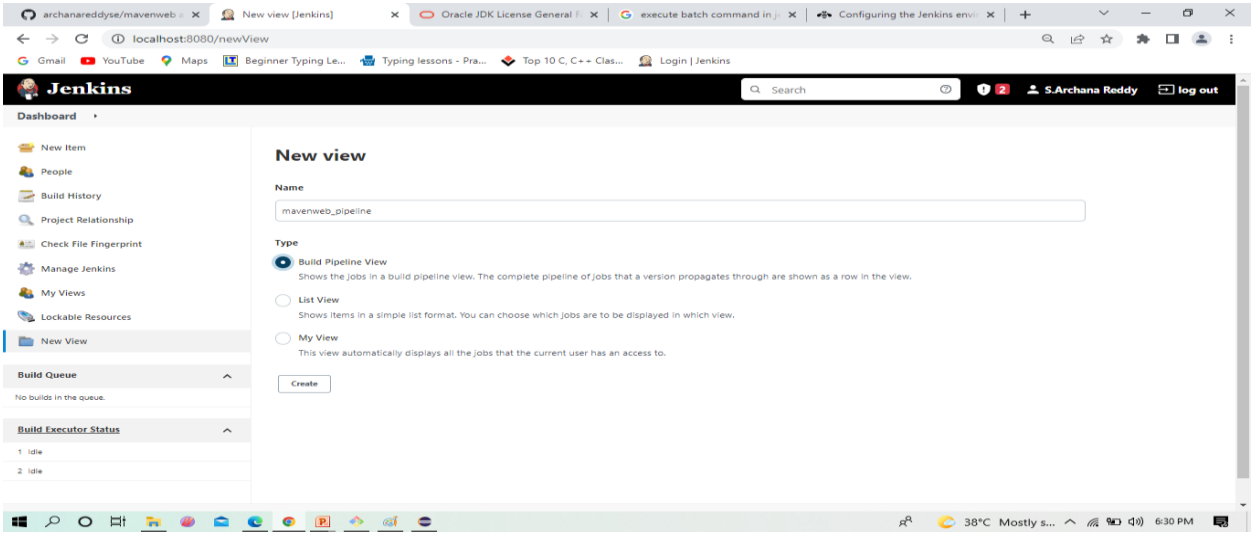
Here we added the credentials of tomcat and tomcat URL also.

Now Apply and Save.



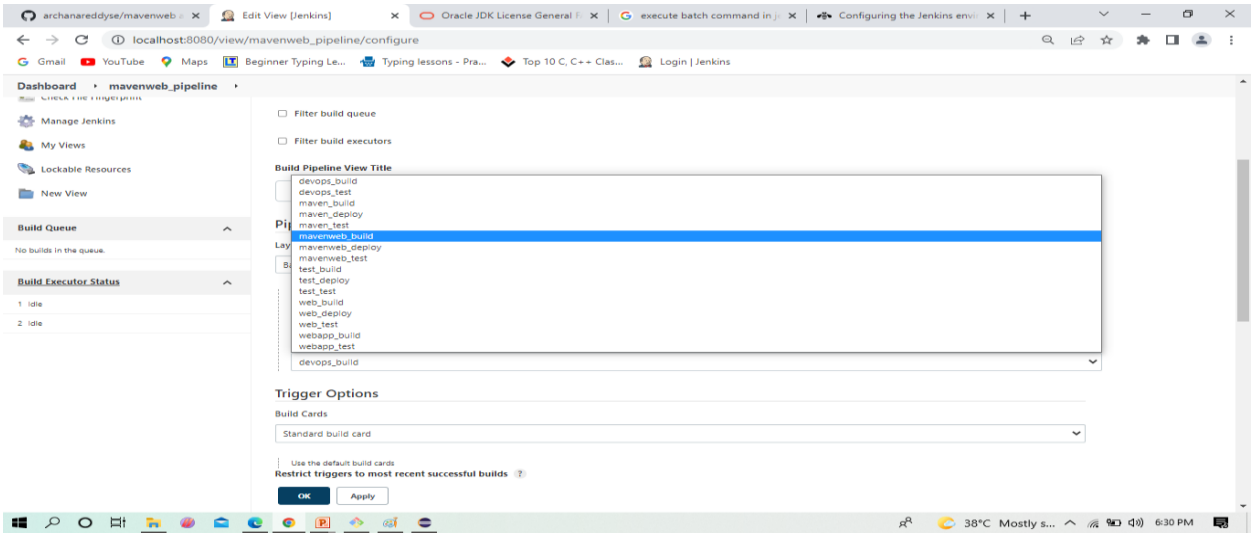
We Create a pipeline by clicking on + symbol in the dashboard ->a pipeline is a collection of events or jobs which are interlinked with one another in a sequence

Give a name to the pipeline->select Build Pipeline View->create

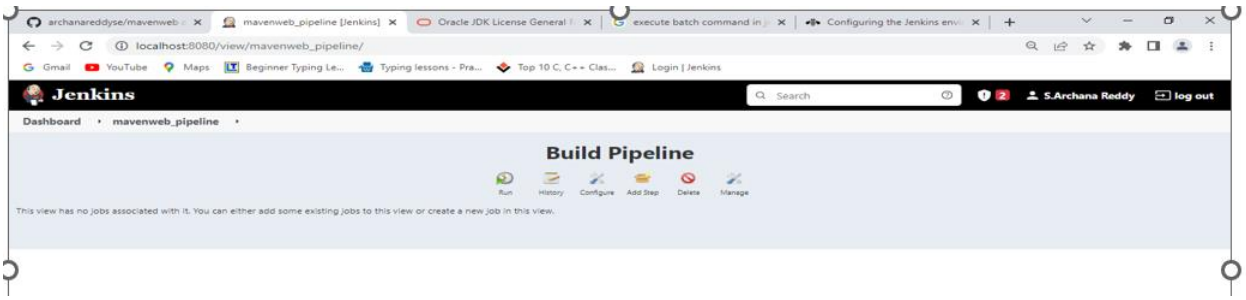


Select the first project to trigger the execution->build session of your project

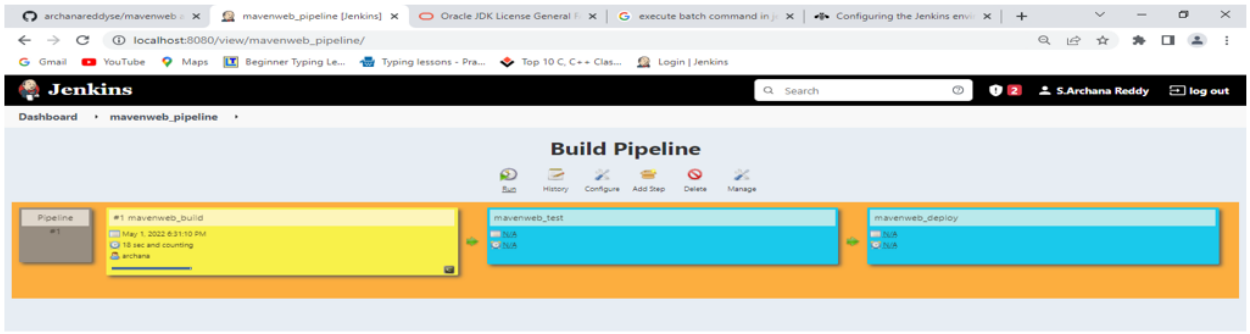
Apply and Save



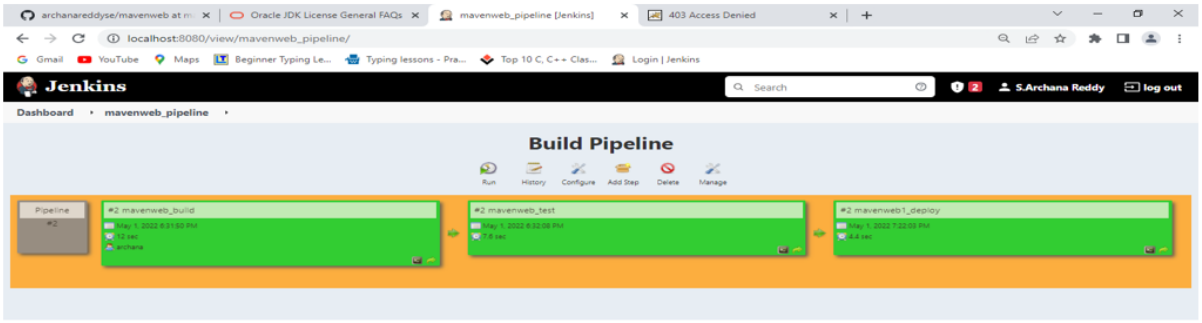
This is the console after save .now click on run.



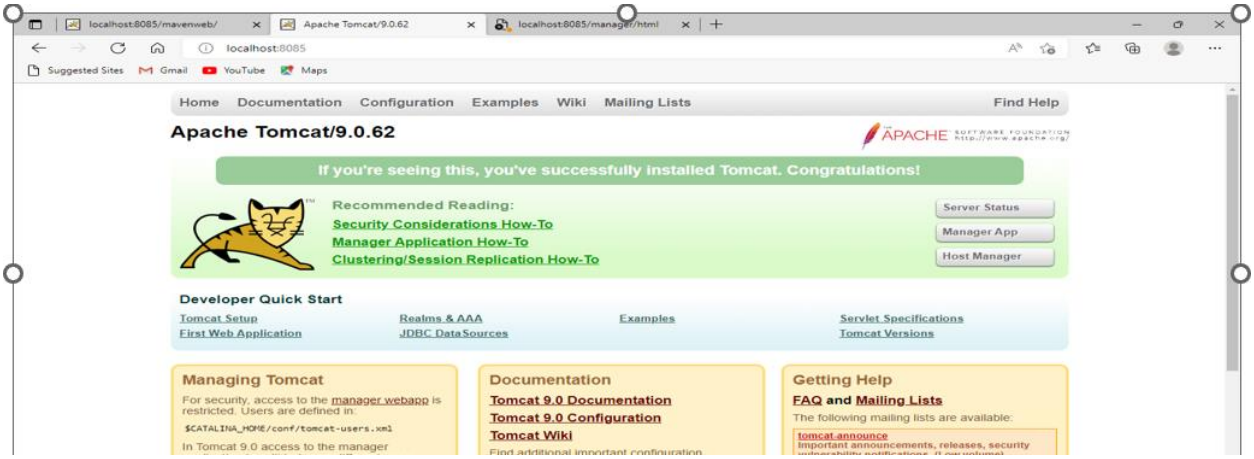
After Click on Run -> click on the small black box to open the console to check if the build is success



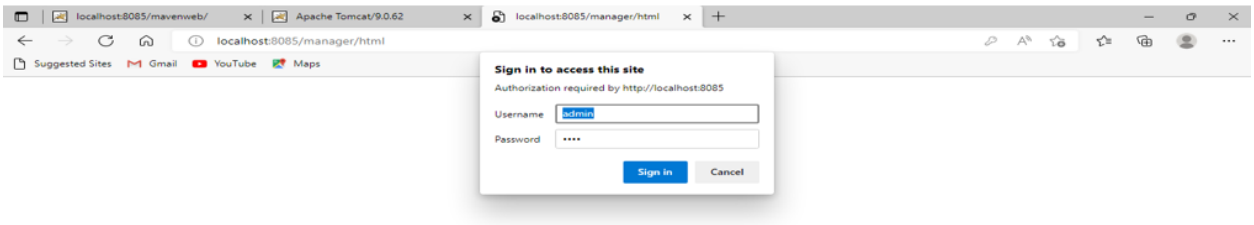
Now we see all the build has success.



Now we can run the local host of tomcat, click -> manager App



It ask for user credentials for login ,provide the credentials of tomcat



After clicking on our project we can see our output here .