**CI/CD**

**ASSIGNMENT 2**

**SUBMITTED BY:**

NITISH KUMAR SINGLA

500071913

R171218126

**SUBMITTED TO:**

HITESH KUMAR SHARMA SIR

In this Assignment, I am going to tell you how to create an Automated Continuous Integration and Delivery Pipeline.

The following tools will be used to create CI/CD Pipeline:

* Source Code Management Tool : Github
* Build Tool : Maven
* CI/CD Tool : Jenkins
* Testing Framework : Junit Testing Framework
* Artifact Repository (Production Server) : Sonatype Nexus
* Automated Trigger : Github Webhooks

Continuous Integration Pipeline basically consists of the following things :

* Committing the Source Code to a Source Code Repository (Github Repository)
* Creating the Build of the Source Code using Apache Maven
* Testing the Source Code using Unit Testing Framework such as Junit

Continuous Delivery basically involves staging and deployment of the Build Application to a Production Server to make it available to the end users.

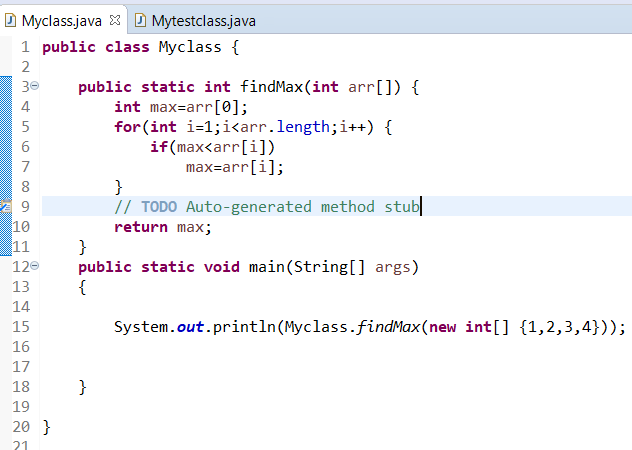


**So let’s start creating our own CI/CD Pipeline.**

First we are going to create a Maven Project. If you want to know how to create a Maven Project in Eclipse in detail, you can go through my other blog named : “Jenkins Integration with Github and Maven”.

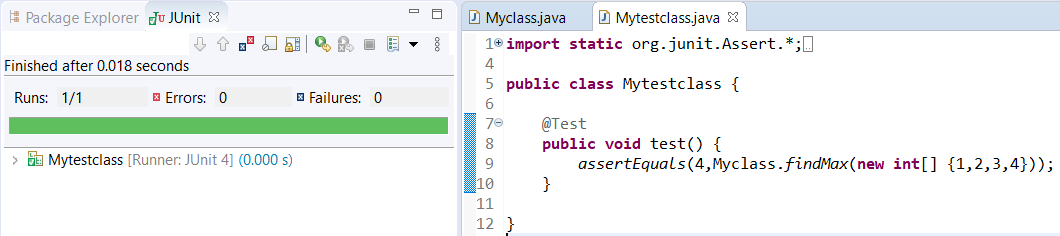
**Writing the Source Code**

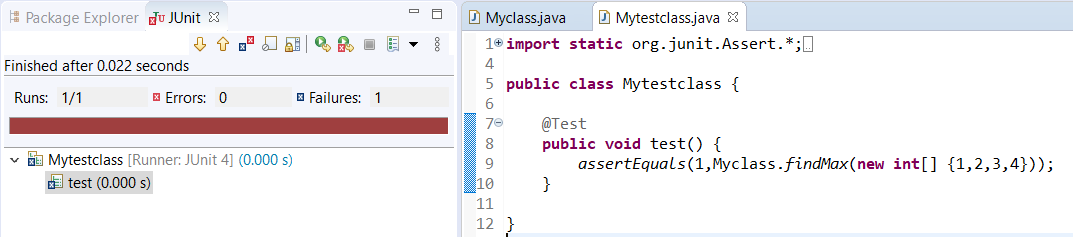
Create a Class in the Maven Project and write your source code in that class. I have created a class named “Myclass” and in that I have written a program in Java which gives you the Maximum Element out of the given input array as output.



**Testing the Source Code using Junit Test Case**

Now for testing this source code, Let’s create a Junit Test Class in which we will write a Test Case according to our source code. I have created a Junit Test Class named “Mytestclass” and in that I have mentioned a test case using **assertEquals(Expected,Actual);** function.



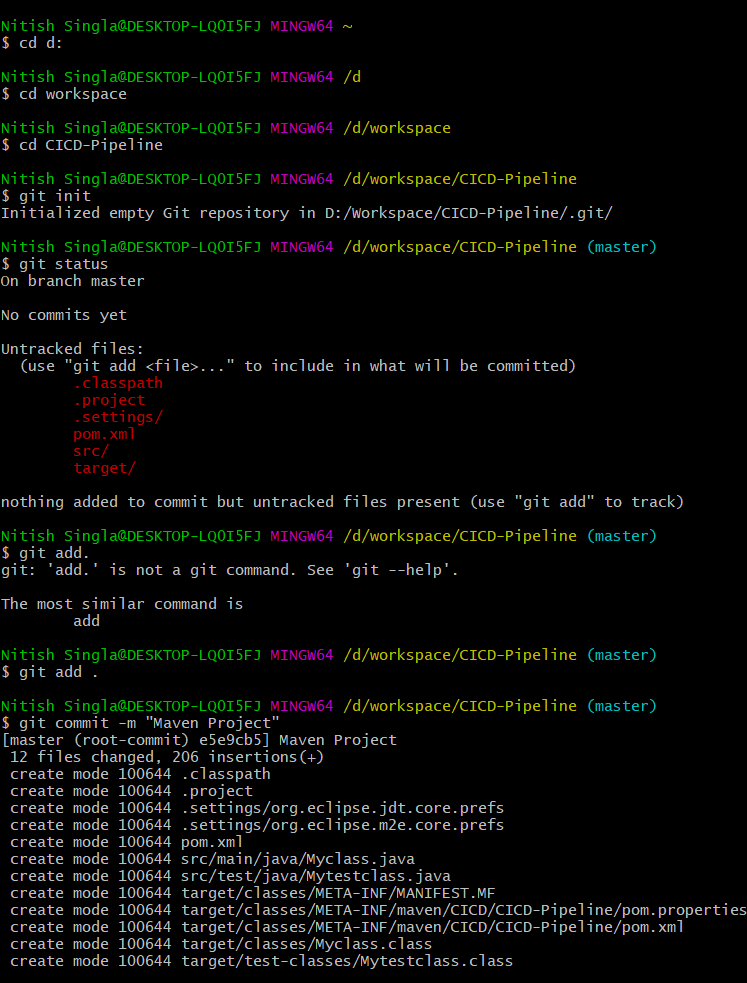


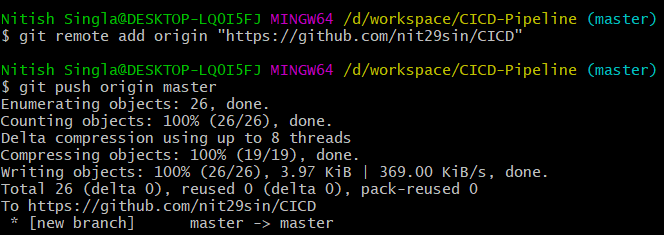
Now in pom.xml, I have added Junit dependency which is necessary to run unit test in Maven. I have downloaded the dependency from the central repository “mvnrepository.com”.

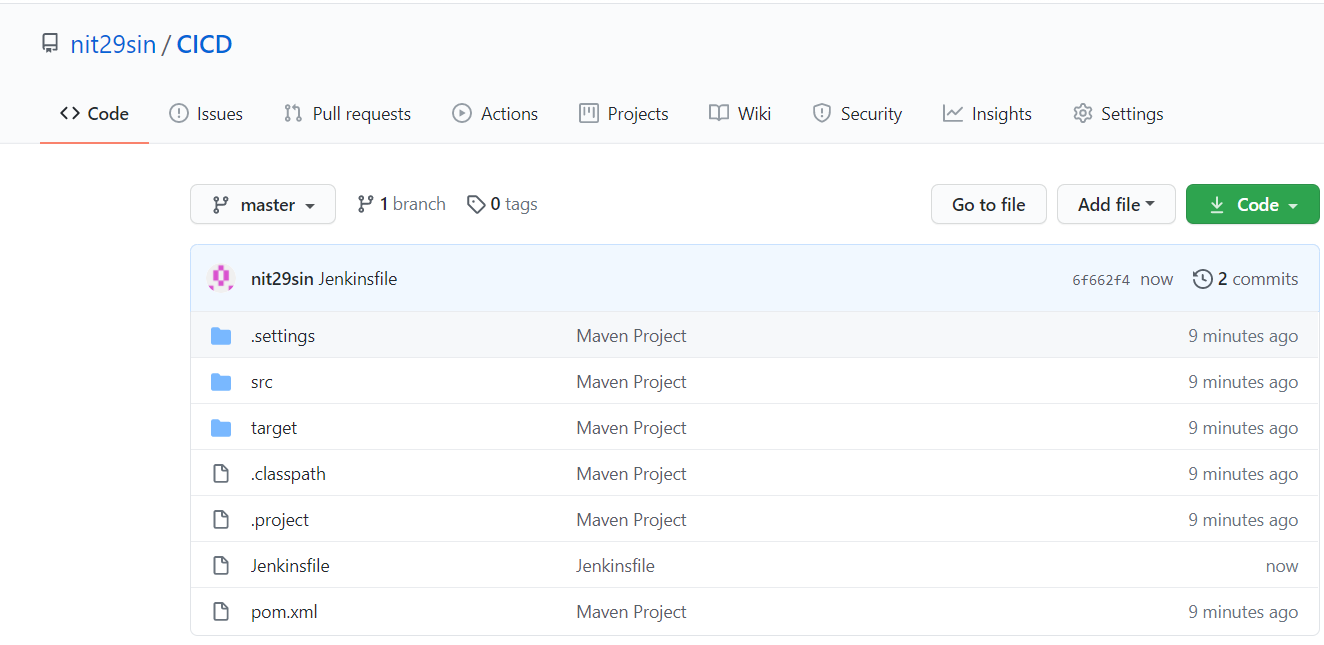


**Importing the Maven Project to a Github Repository**

Now import all the contents of Maven Project to a Github Repository. I have imported using Git Bash in Windows. You can also simply drag and drop the contents in the repository.





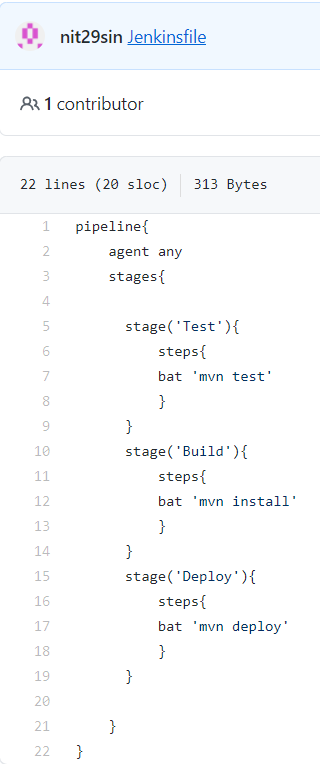


**Jenkinsfile**

So, Jenkinsfile is a text file that contains the definition of a Jenkins Pipeline and is checked into source control (Github Repository). So basically we are going define the Pipeline in a Jenkinsfile which Jenkins will then load directly from source control.

So here I have created a 3 Stage Pipeline.

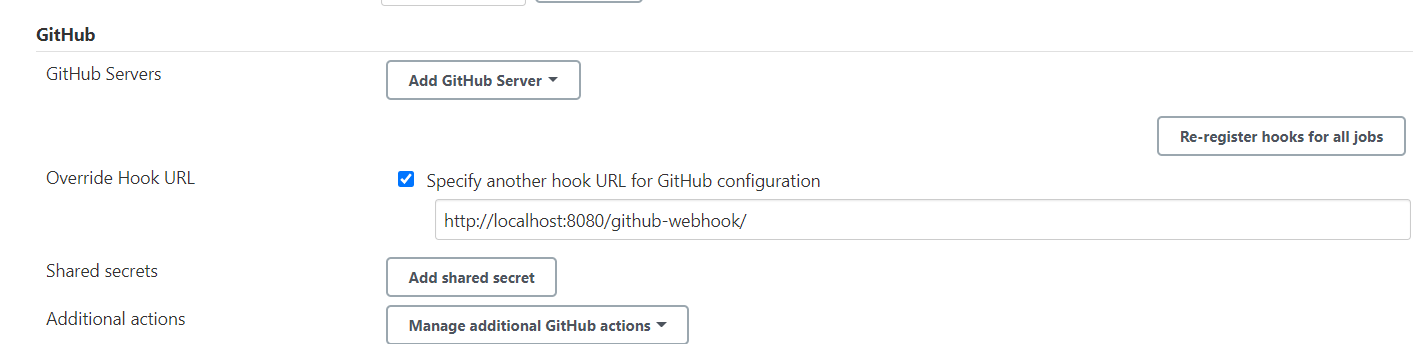
* First, my source code will be tested using mvn test command
* Then, the build of the source code will be created using mvn install command
* Then finally the build application will be deployed in the Nexus Server using mvn deploy command.



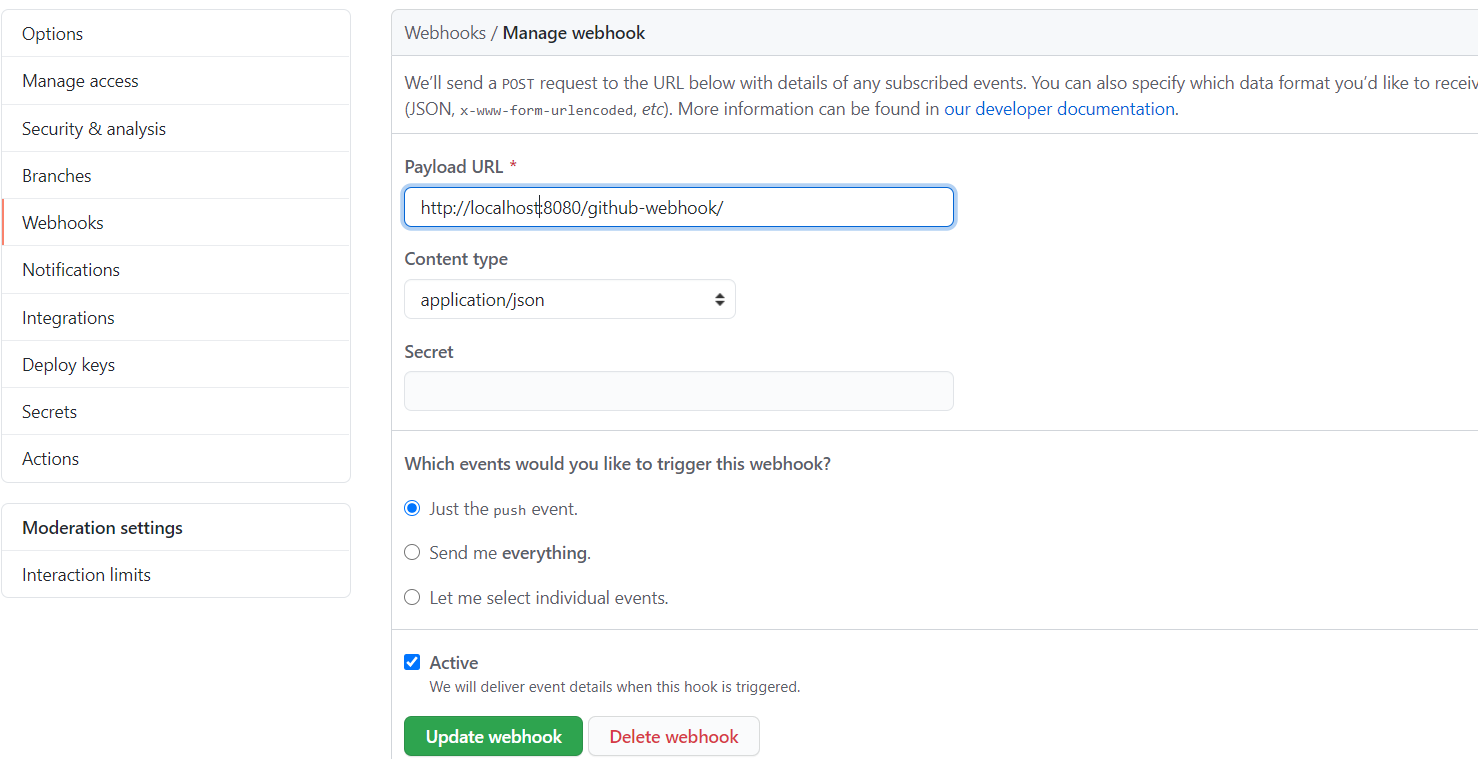
**Github Webhooks**

Github Webhooks is used for Automatic Trigger of the Pipeline whenever a Developers pushes new changes in the source code. This means that the moment a Developer pushes new changes in the repository then the Pipeline created in Jenkins will run automatically and all the command will be executed automatically. Testing, Building and Deploying will e done automatically without any user intervention (without having to do it manually).

To setup Github Webhooks and integrate it with Jenkins, first go to Jenkins Dashboard and then go to Manage Jenkins 🡪 Configure System. Then there go to Github Section and Click on Advanced. Copy the Override Hook URL.



Then go to your Github Repository which contains the Maven Project. Go to Settings 🡪 Webhooks. Paste the copied URL in Payload URL. Instead of Local Server you would have to mention the IP Address of the Server on which you are working. Select Content Type as application/json and select “Just the push event”.



**Creating a Pipeline project in Jenkins**

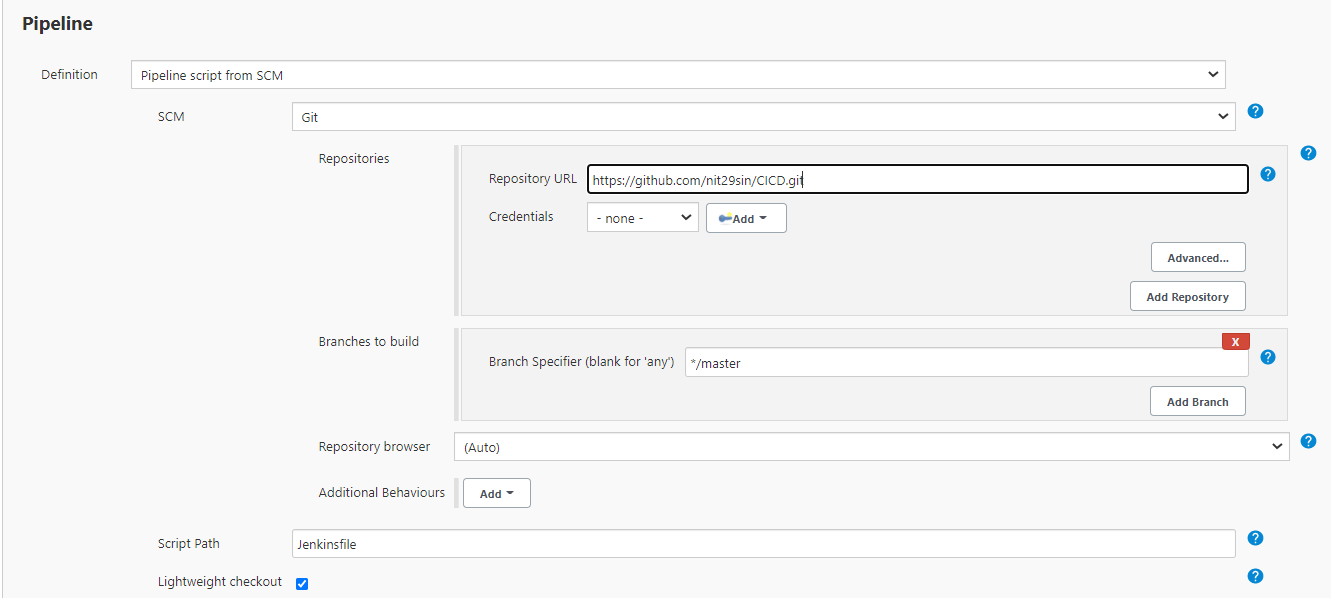
If you want to know how to install and run Jenkins on your machine, then refer to my other blog named “Getting started with Jenkins”.

Now, we will create a Pipeline Project in Jenkins. Start Jenkins on your machine using java -jar jenkins.war command and then go to <https://localhost:8080>, enter you credentials and log in.

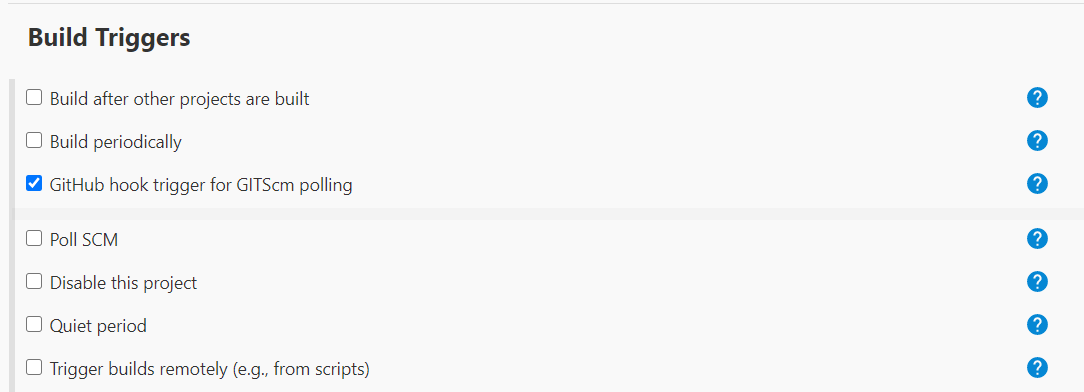
Now, go to New Item 🡪 Pipeline. Enter the name for your project. I have entered the name as CICD. Then click on Apply 🡪 Save.

Now go to the Pipeline Tab and select “Pipeline script from SCM” in the definitions section. We have selected this because we have defined our Jenkinsfile in Github Repository and now the Jenkins will load the Jenkinsfile directly from the source code.

Now select “Git” in the SCM section and paste the url of the Github Repository which contains the Maven Project and Jenkinsfile.

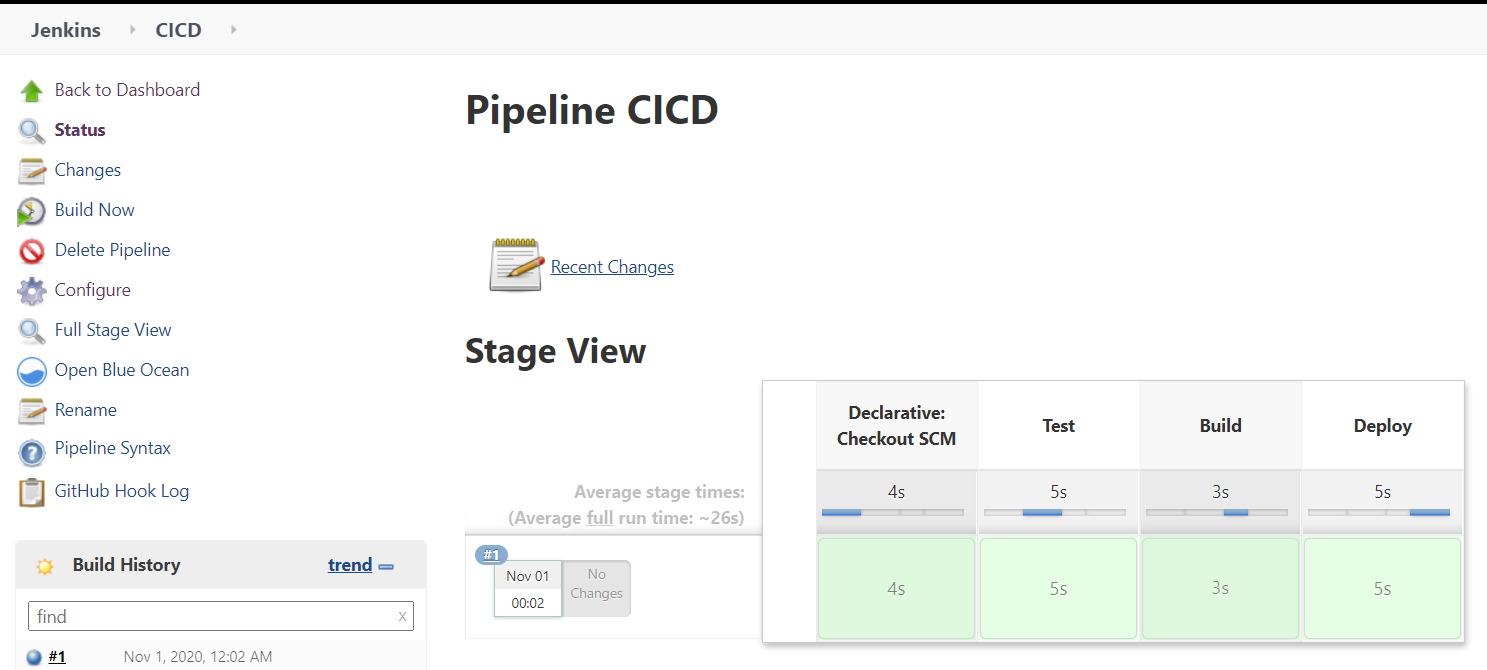


Now in Build Triggers Tab, select “GitHub hook trigger for GITScm polling”. With this option, Pipeline defined in Jenkinsfile will run automatically whenever new changes are pushed to the Github Repository.



Now, just click on Apply 🡪 Save.

Now Click on Build Now to execute and run the pipeline. After successful execution of Pipeline, you can see the stage view. Stage View shows which stages executed successfully and the average time taken by each stage to run and total stage time.



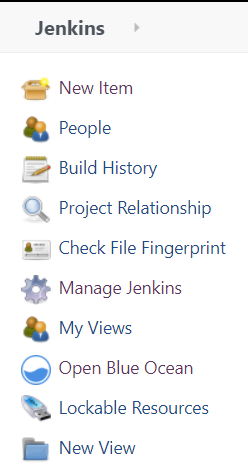
**Blue Ocean**

**Blue Ocean** is a new user experience for **Jenkins** based on a personalizable, modern design that allows users to graphically create, visualize and diagnose CICD Pipelines.

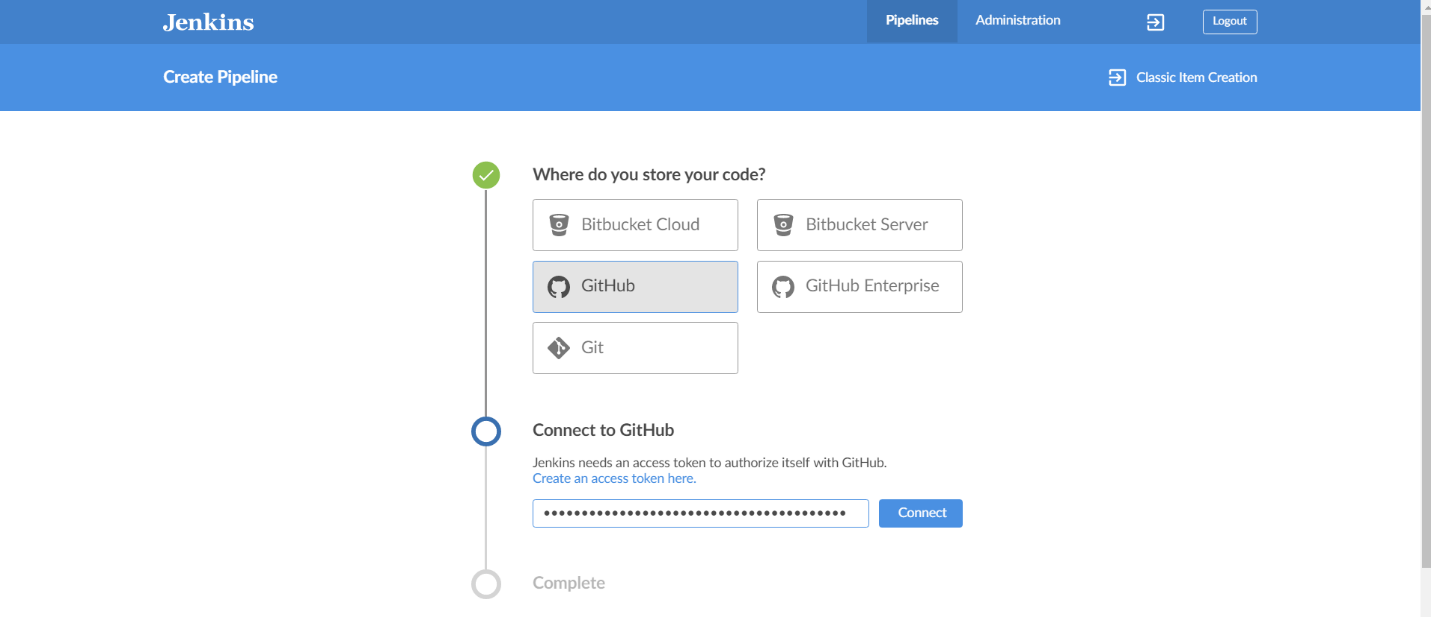
Blue Ocean is just a different and better way to visualize and implement the Pipeline. The process and concept behind creating the pipeline is exactly the same, It just provides you with a better user experience to implement the Pipeline.

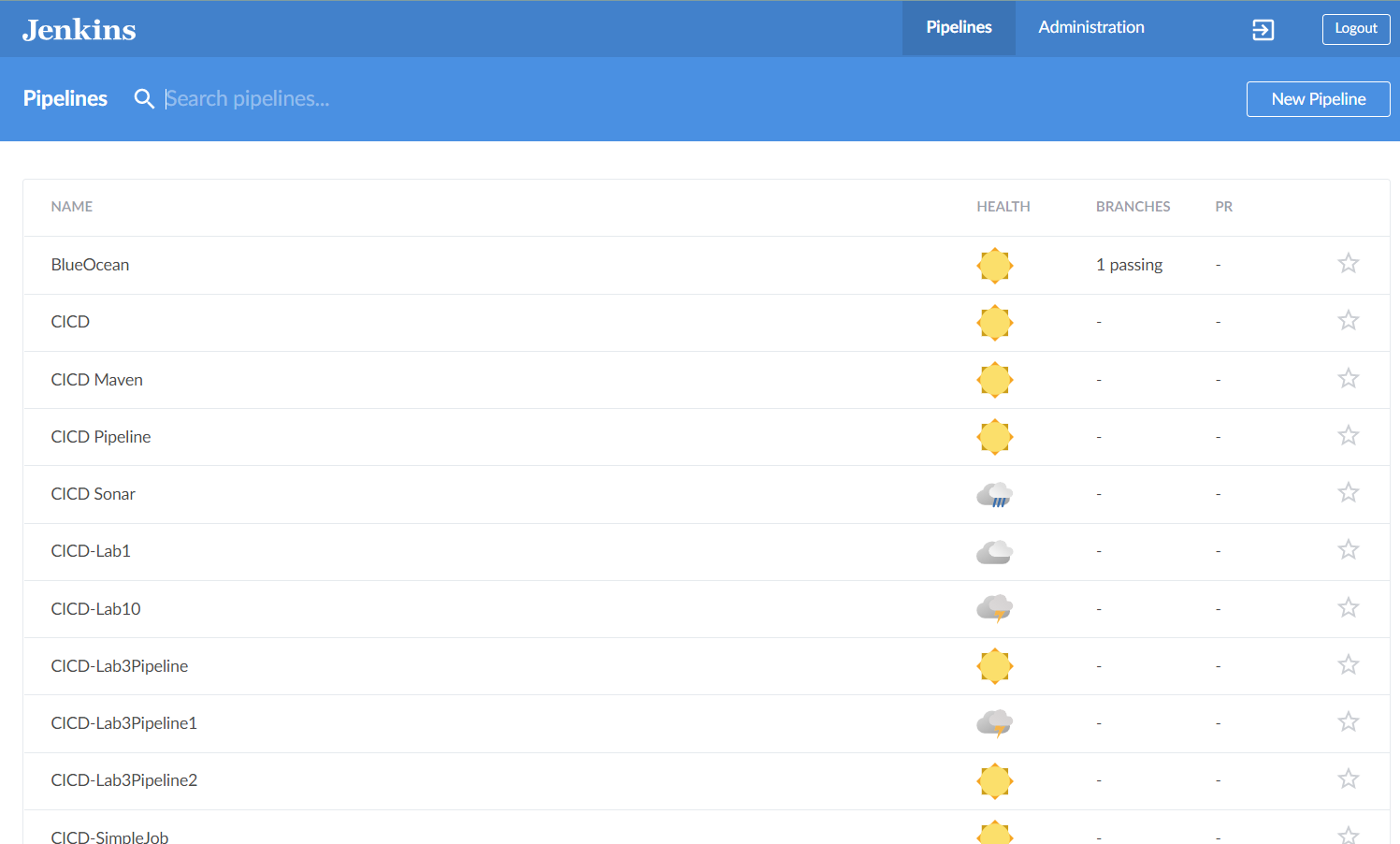
So for using Blue Ocean, first you have to install BlueOcean Plugin. Go to Manage Jenkins 🡪 Manage Plugins and then search for Blue Ocean Plugin and install it.

Then Go to main Jenkins Dashboard and Click on Open Blue Ocean.



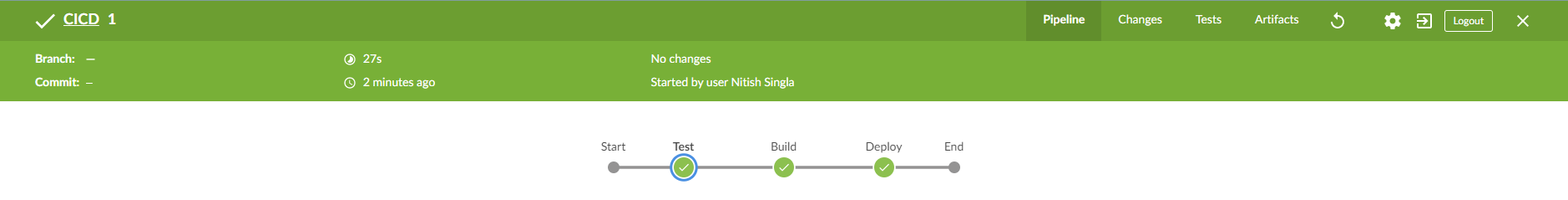
Then go to your Pipeline project that you have already created and then Click on Open Blue Ocean. Then Enter Github Credentials to integrate BlueOcean with Github. Then Select the desired Github Repository with a Maven Project and Jenkinsfile in it.

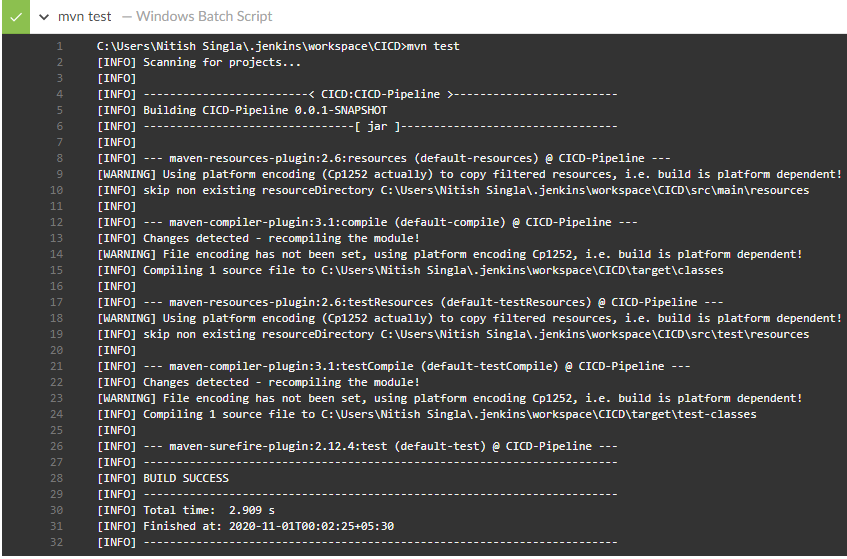


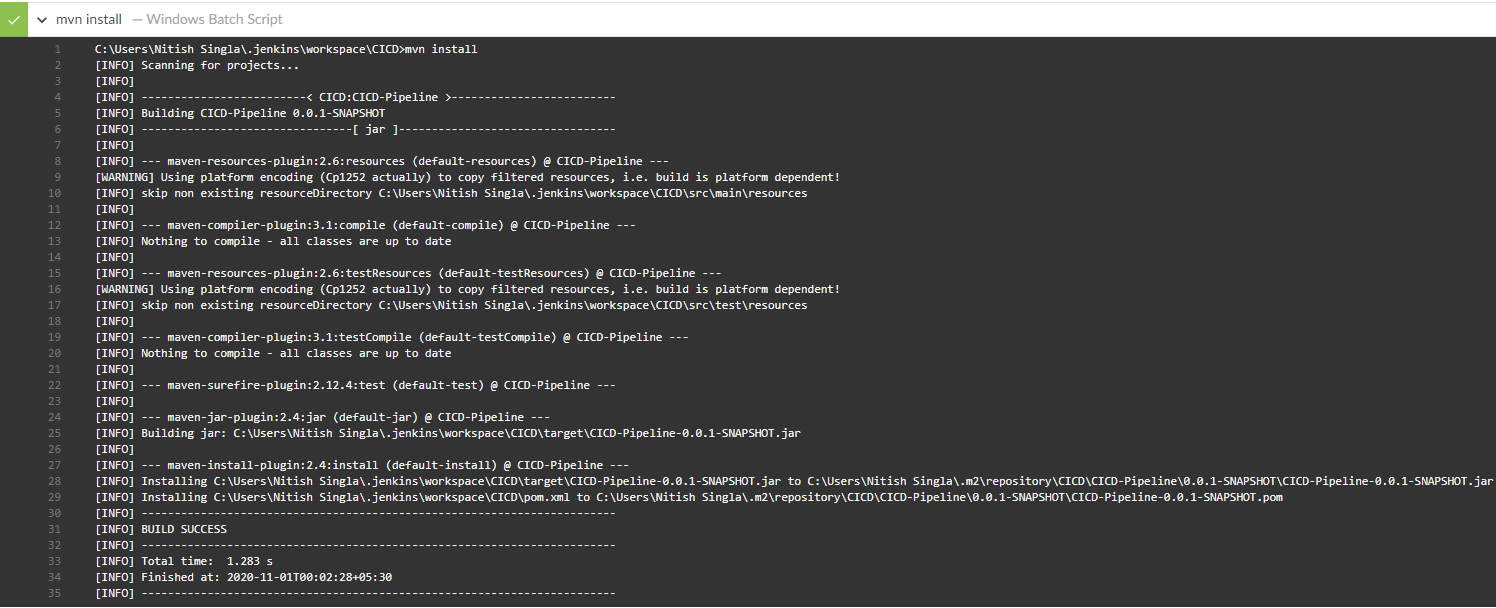


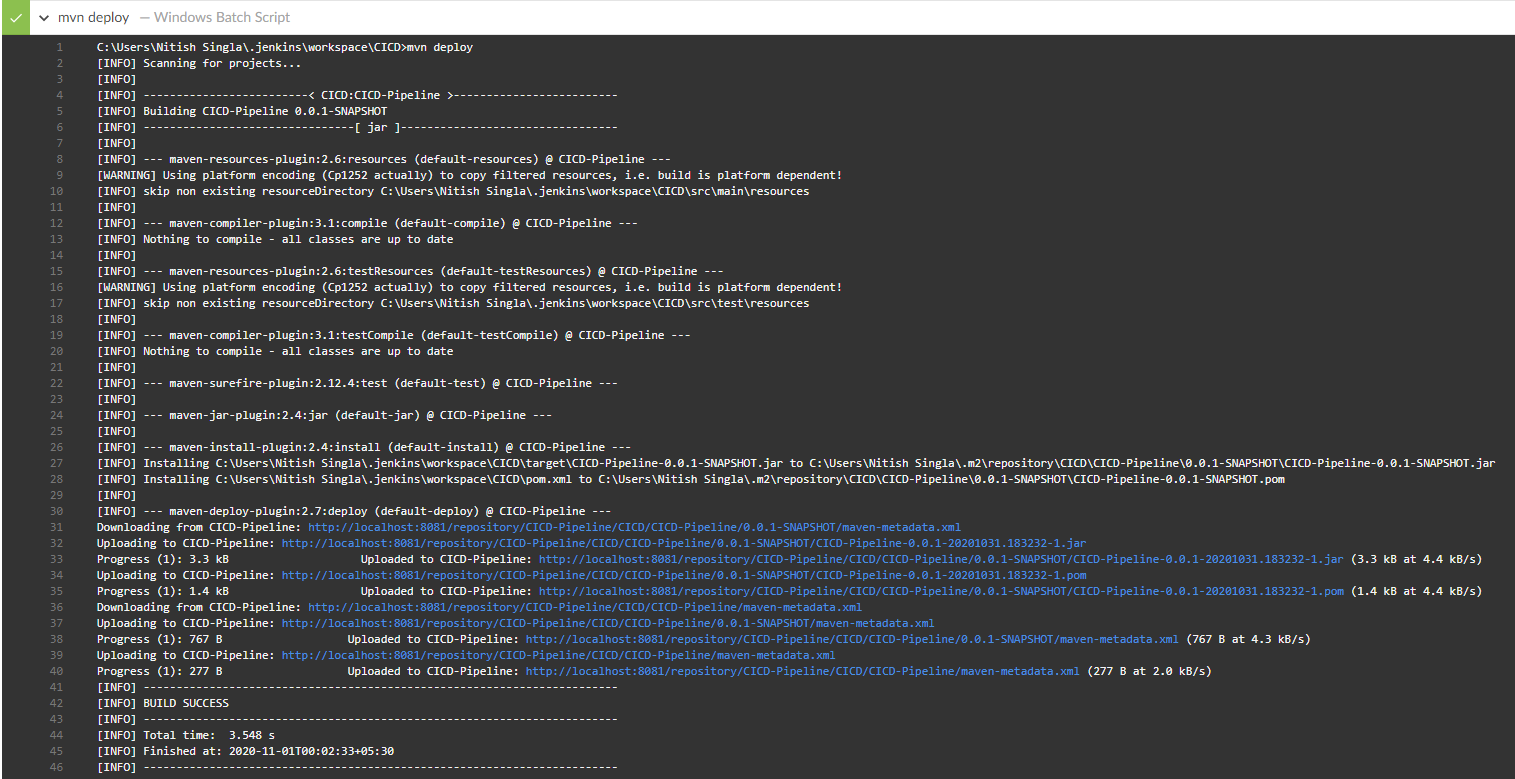
Now your Maven Project is integrated with BlueOcean. Now Just click on Run Button to execute and run the Pipeline.

After the successful completion, you can see tick marks on each stage and you can also view Console Output for each stage.









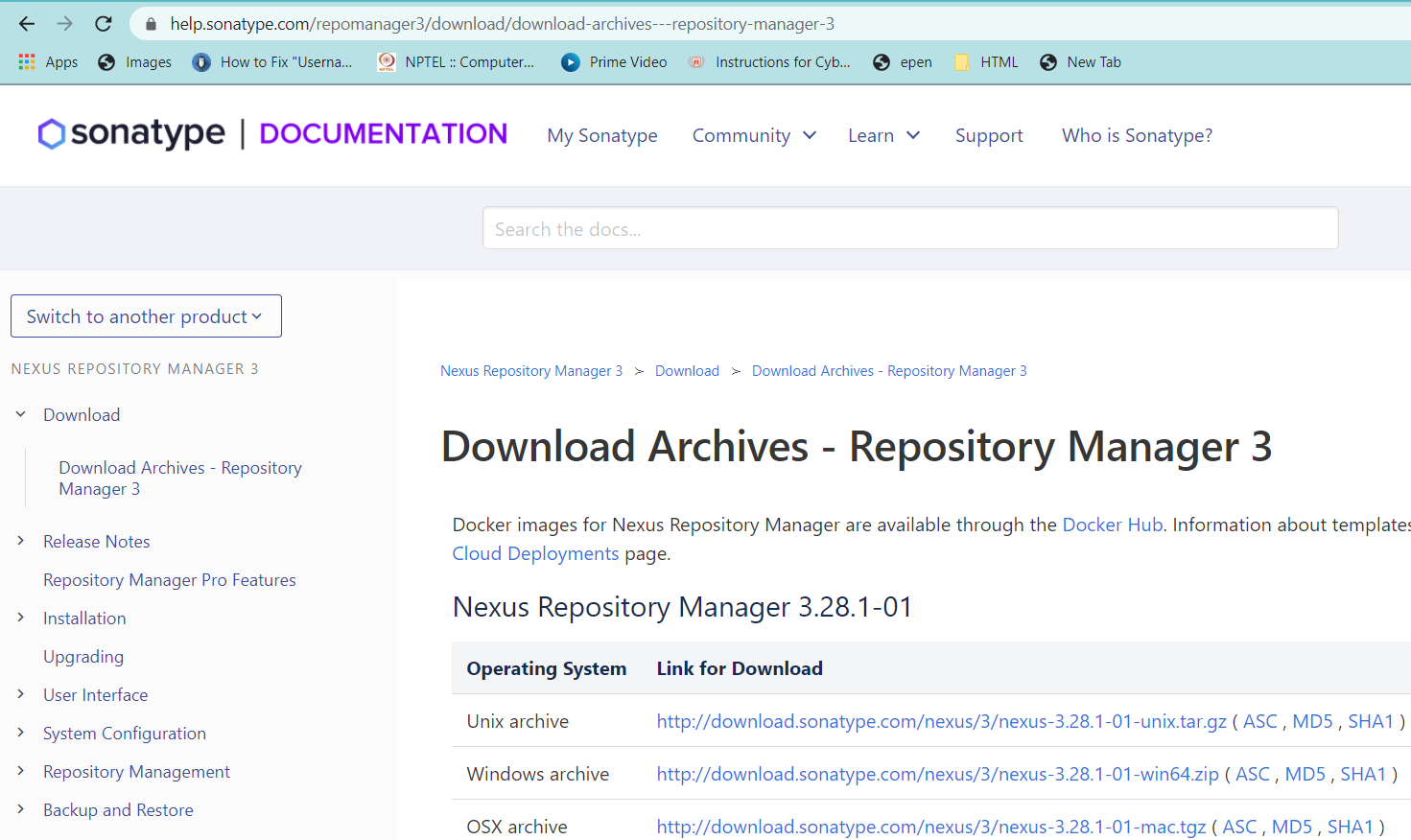
It’s kind of easy to work with Blue Ocean. Using Blue Ocean, You can also run the projects that you have previously created in Jenkins.

**Sonatype Nexus**

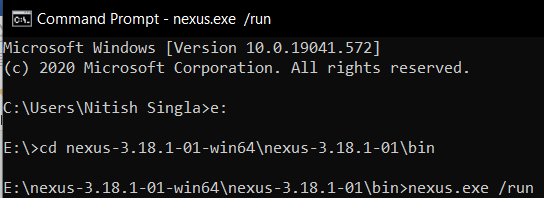
Sonatype Nexus is an artifact repository manager where to can store and manage your binaries and build artifacts. All of your build artifacts can be deployed to a Nexus Server at the end using mvn deploy command defined in the Jenkinsfile.

Now, I am going to tell you how to download and install Nexus and how to run it.

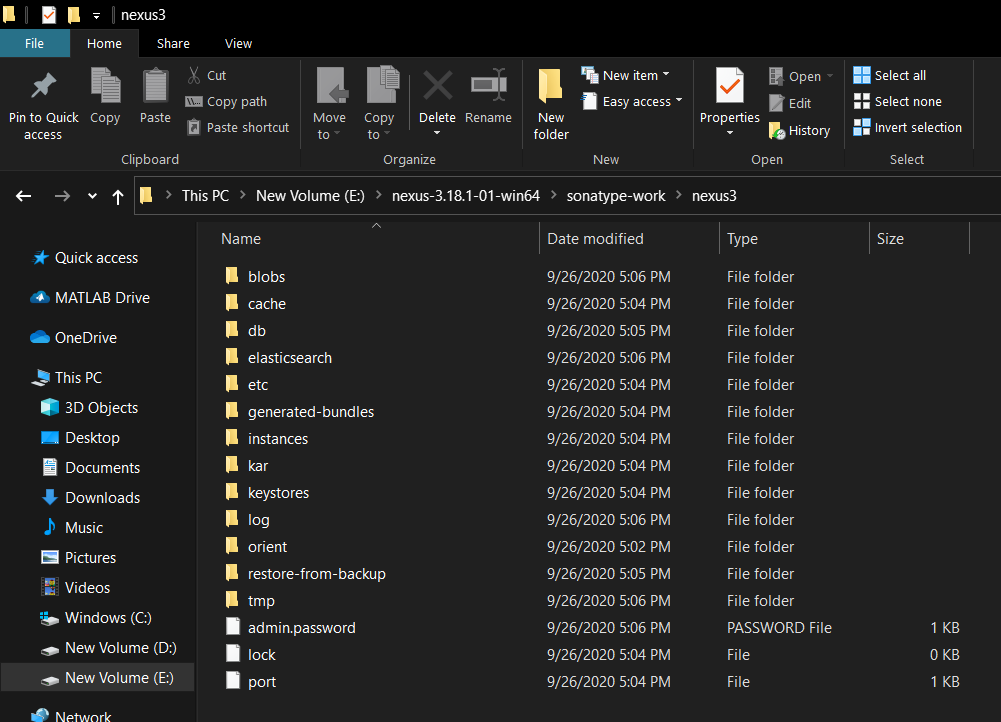
For Installation, go to <https://help.sonatype.com/repomanager3/download/download-archives---repository-manager-3> and download Nexus Repository Manager according to your System Requirements.



After downloading, install the Nexus Tool to a desired folder. Then go to Command Prompt and then go to the drive where have you installed it. Now go to cd desired folder\nexus-3.18.1-01-win64\nexus-3.18.1-01\bin. I have installed my Nexus in E: drive, E:\nexus-3.18.1-01-win64\nexus-3.18.1-01\bin and then run command nexus.exe /run.

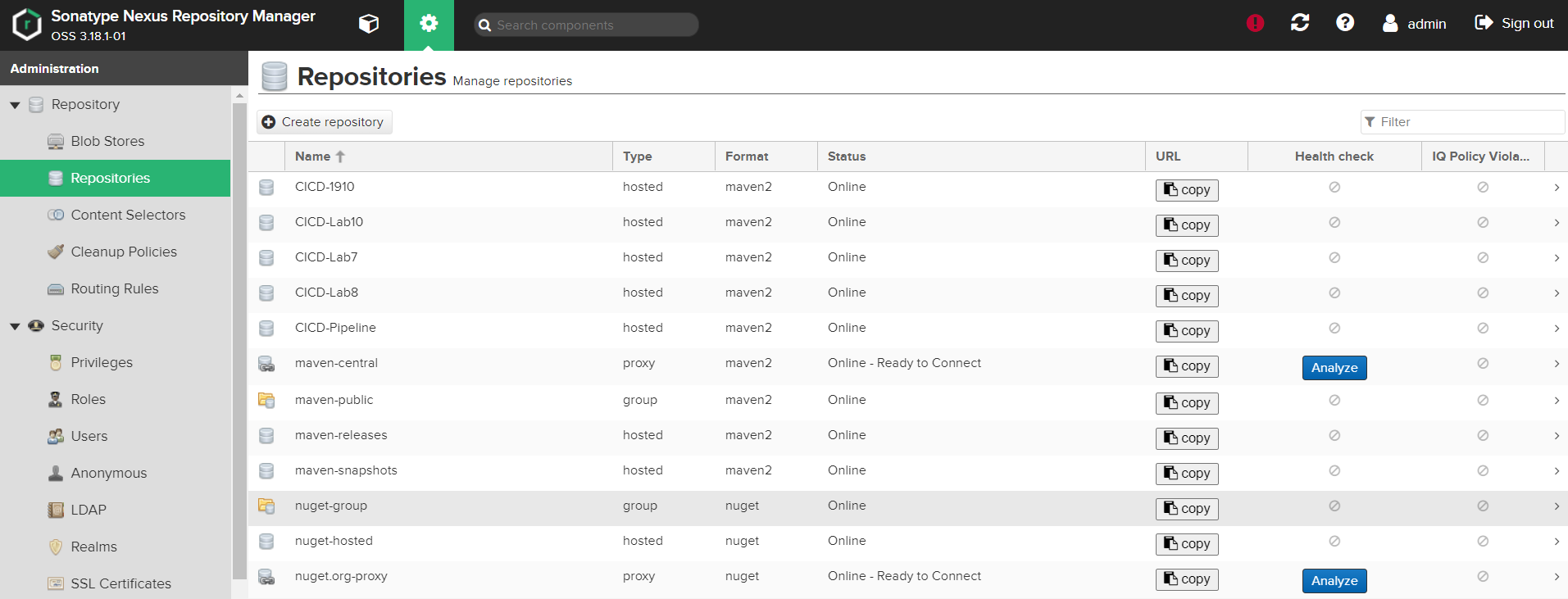


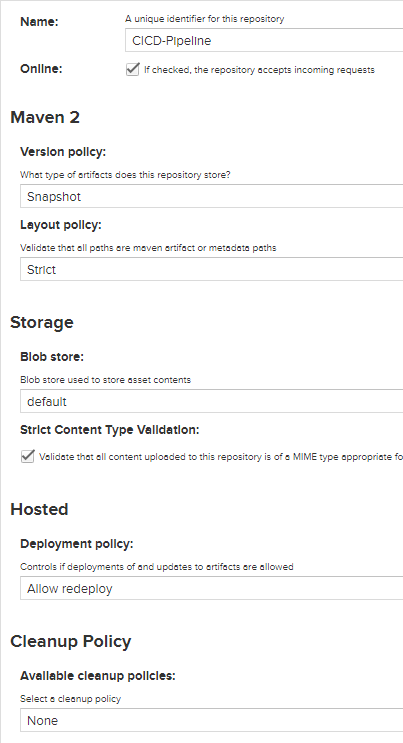
Go to E:\nexus-3.18.1-01-win64\sonatype-work\nexus3 and Open admin.pasword file to get password for first time authentication.

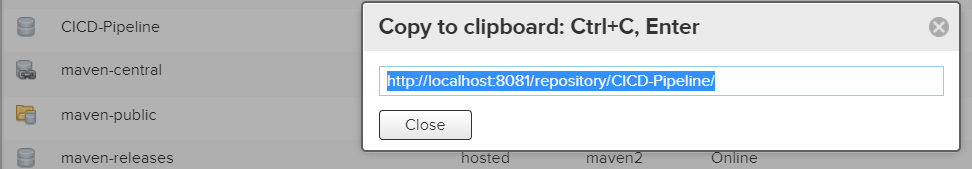


Then go to url <https://localhost:8081> and enter username as admin and password from admin.password file.

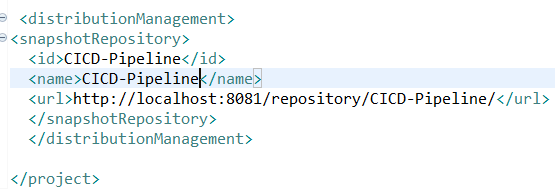
Now to create a Nexus Repository, go to Setting Icon 🡪 Repositories 🡪 Create Repository and then Select Recipe as maven2 (hosted) as we are going to deploy Maven Build Artifact and then enter the desired information like Repository name and then click on Create Repository.







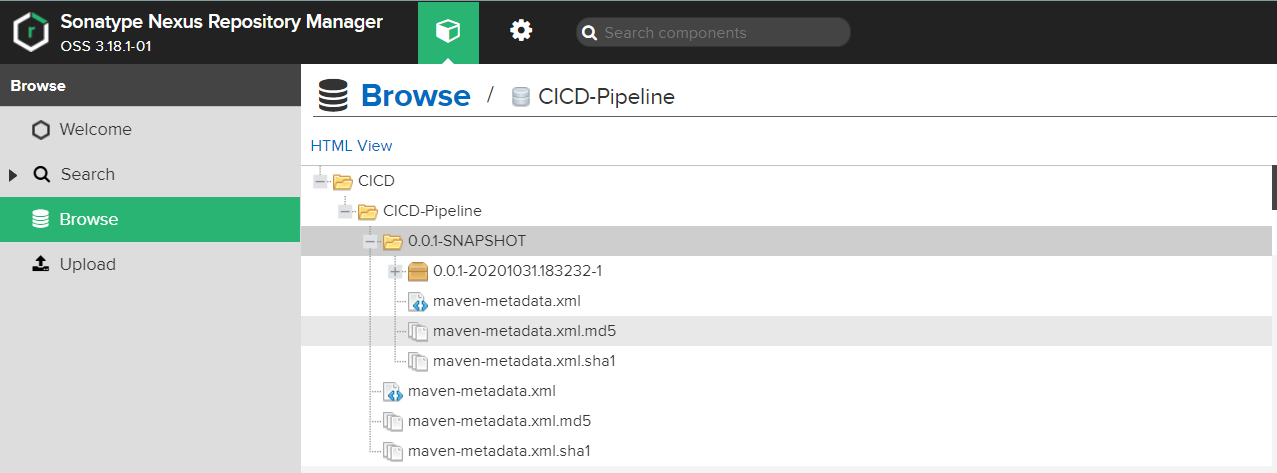
Now copy the url of the Nexus Repository created and paste it in the <distributionmanagement> <snapshotrepository> element. Also, in this element the id and name should be same as that of Nexus Repository created. Like I have created a Reposittory named CICD-Pipeline and kept the id and name same as CICD-Pipeline.



Now go to the path where you have installed your Apache Maven and then go to conf 🡪 settiings.xml. In settings.xml, we are going to create a server in the server tag to integrate Nexus with Maven. In the severs tag, add the id tag and enter name in it as same as that entered in the Nexus Repository and in the username and password tag add the Nexus Credentials that we use to log in.



After running the Pipeline, the build artifact created will be deployed to the Nexus Repository as shown below. You can go to Repositories Section in Nexus Dahboard, select the desired repsoitory and see the Build Artifcat present there.



**Conclusion**

Now, I am going to quickly tell you what we have done till now.

* We have created a Maven Project and written the source code in a Java Class.
* Then, we have created a Junit Test Case in a Test Class in the Maven Project.
* Then, we have defined the dependency of Junit in pom.xml.
* Then, import the Maven Project to a Github Repository.
* Create a Jenkinsfile in it and define a 3 Stage Pipeline. 1st Stage is testing the source code using mvn test. 2nd Stage is Building the Maven Project using mvn install. 3rd Stage is deploying the source code using mvn deploy.
* Then, we created a Pipeline in Jenkins and selected the option to run Jenkinsfile from SCM (Github Repository).
* Then, we have created a Nexus Repository, defined the url of the repository in pom.xml and created the Nexus Server in Apache Maven settings.xml file to integrate Nexus with Maven.
* Then, we just run our Pipeline created in Jenkins. You can also run it using Blue Ocean for better user experience.

So, We have finally created our own CI/CD Pipeline.

**Note: I have posted the blog for the same. Check it out at :** https://medium.com/@singlanitish29/automated-ci-cd-pipeline-8124eb8dc136?source=friends\_link&sk=eb2eed6b46000fed7585f9a6bd00f108