**EXPERIMENT NO:6**

Continuous Integration with Jenkins: Setting Up a CI Pipeline, Integrating Jenkins with Maven/Gradle, Running Automated Builds and Tests.

**What is A CI/CD Pipeline?**

CI/CD stands for [Continuous Integration / Continuous Deployment](https://www.geeksforgeeks.org/what-is-ci-cd/)first let us try to understand what is a pipeline. In computing, a pipeline is a set of stages or processes linked together to form a processing system. Each stage in the pipeline takes an input, processes it in accordance with a set of rules, and then sends the outputs to the stage that follows. Frequently, the pipeline’s overall output is its final step’s output. like the procedures outlined below

* Test code
* Build Application
* Push Repository
* Deploy to Server

**What is Continuous Integration (CI)?**

Continuous integration means whenever new code is committed to remote repositories like [GitHub](https://www.geeksforgeeks.org/introduction-to-github/), GitLab, etc. Continuous Integration (CI) will continuously build, tested, and merged into a shared repository.

**Benefits of Continuous Integration (CI)**

* We can maintain the reports of the projects
* Deployments can be made within the given time
* Bugs can be found quickly.

**Continuous Deployment/Delivery (CD)?**

**Continuous Deployment**

Continuous Deployment means automating the further stages of the pipeline automatically or manually deploying the application/code to different environments like Dev, Test, and Production. Automating the build is the main component of Continuous Integration and  Continuous Deployment.

**Continuous Delivery**

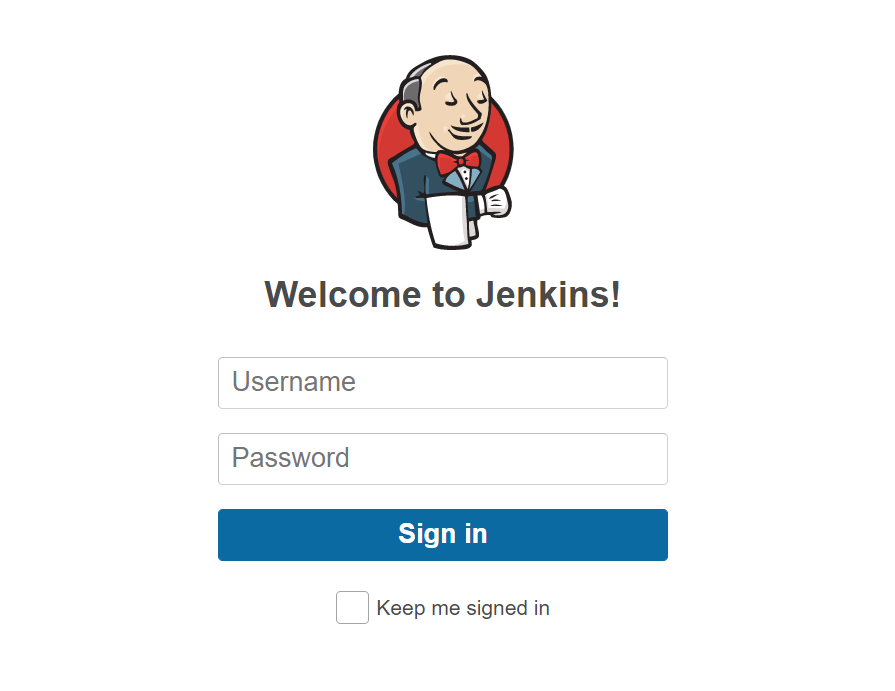
Each and every build that passed all automated tests and was able to be fully automated and delivered into production only required one click of human intervention is called Continuous Delivery.

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

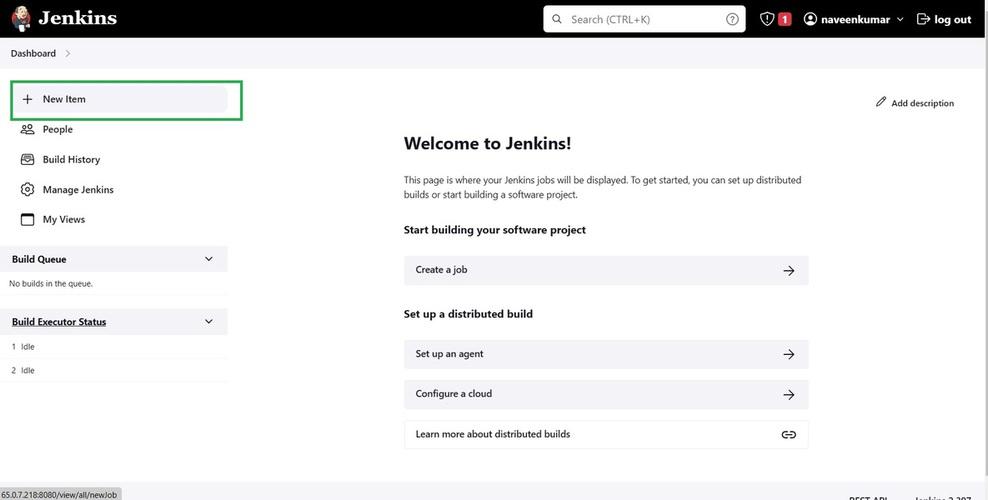
In this project, we will try to learn a basic CI/CD pipeline that is required to build a Java web application .war file by using [Jenkins](https://www.geeksforgeeks.org/jenkins/). To install Jenkins refers to the [Jenkins installation](https://www.geeksforgeeks.org/how-to-install-and-configure-jenkins-on-debian-linux-ubuntu-kali-mint/).

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

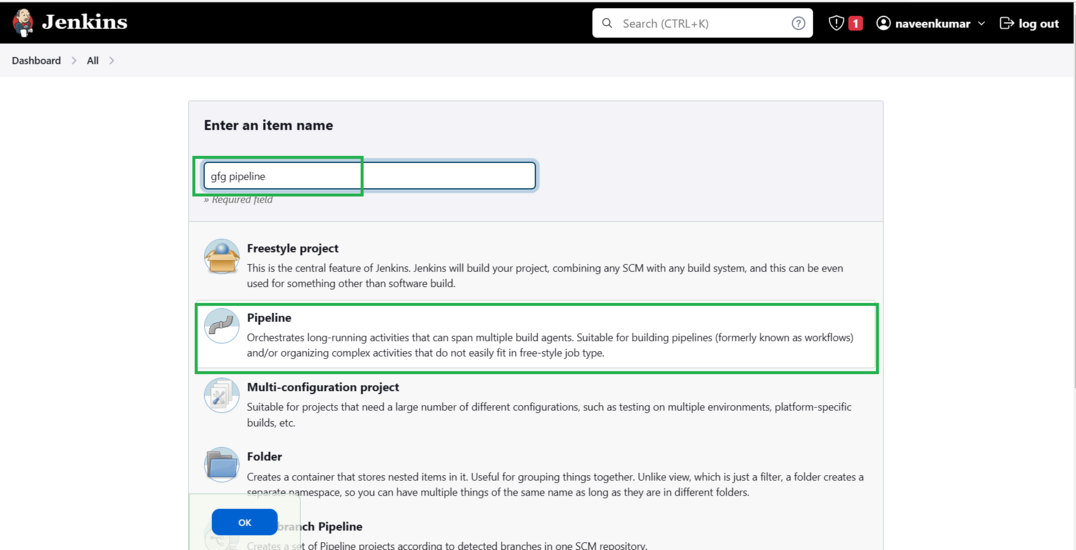
**Step 2.**Once logged in, the user will be redirected to the Jenkins console, here’s the reference for the same.



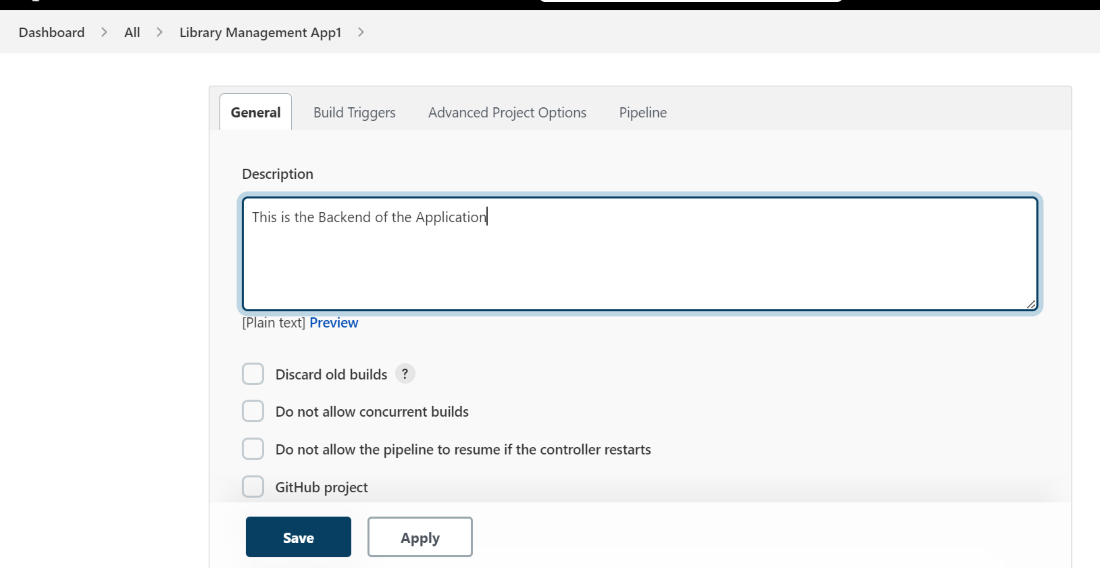
**Step 3**. To create a new project select the option available in the Dashboard which is **“New Item”** Refer to the image provided below:



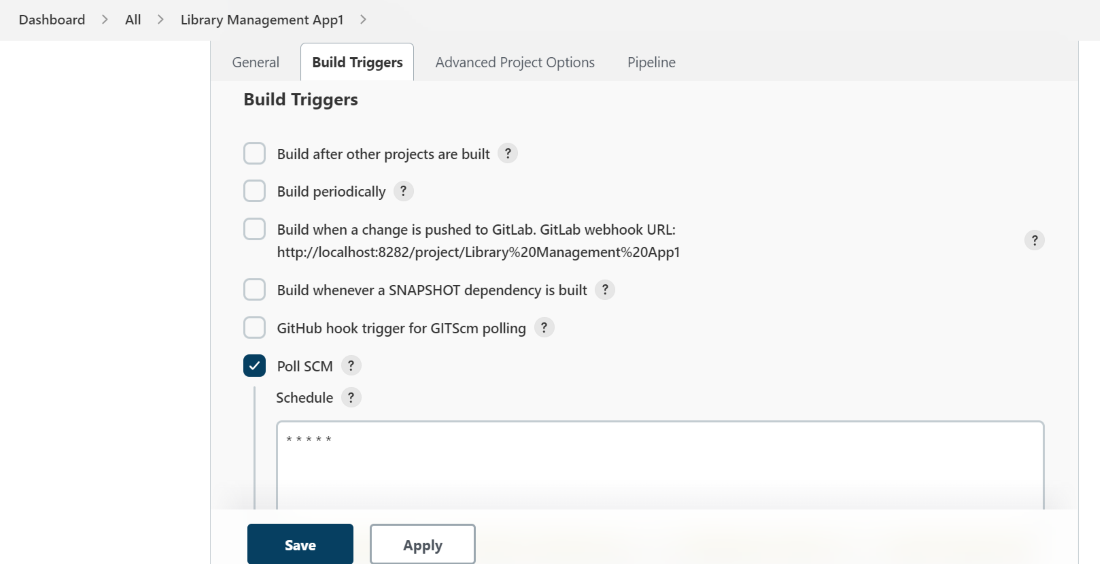
**Step 4**. Now a list of options will be visible on the screen, along with a field to name the pipeline. Add a suitable name and select the **“Pipeline”** option to proceed. Refer to this screenshot.



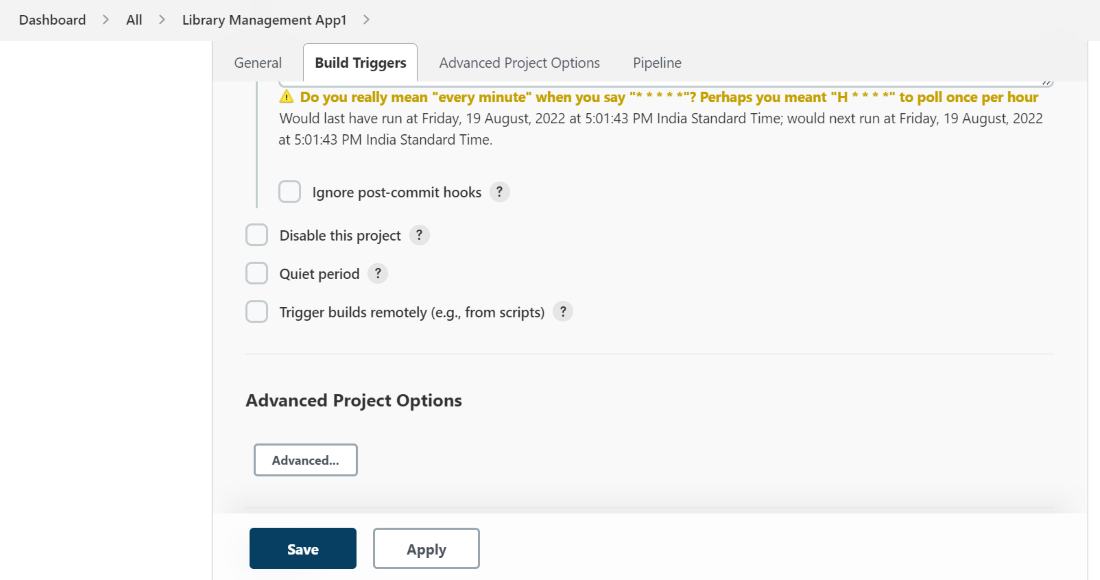
**Step 5:**Once redirected, the configuration page will appear. This is the most important page as here all the details will be filled in. At first, there is the **General** section where the user can add a description based on the project for which the pipeline has to be created. And establish the connection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



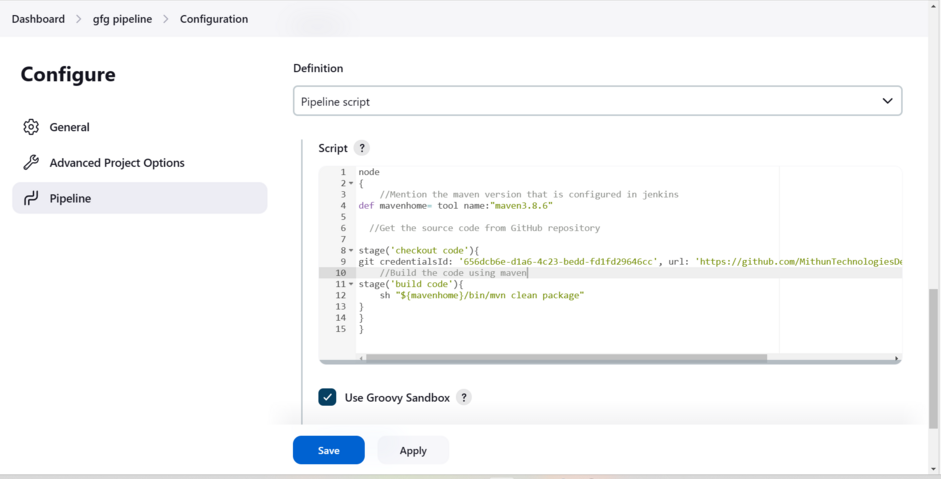
**Step 6.**Now comes the second section, i.e. **“Build triggers”.**Here, we need to specify the branch and repository and give the credentials too. And add additional behaviors if required so far. Refer to the screenshot to have a better understanding.



**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.

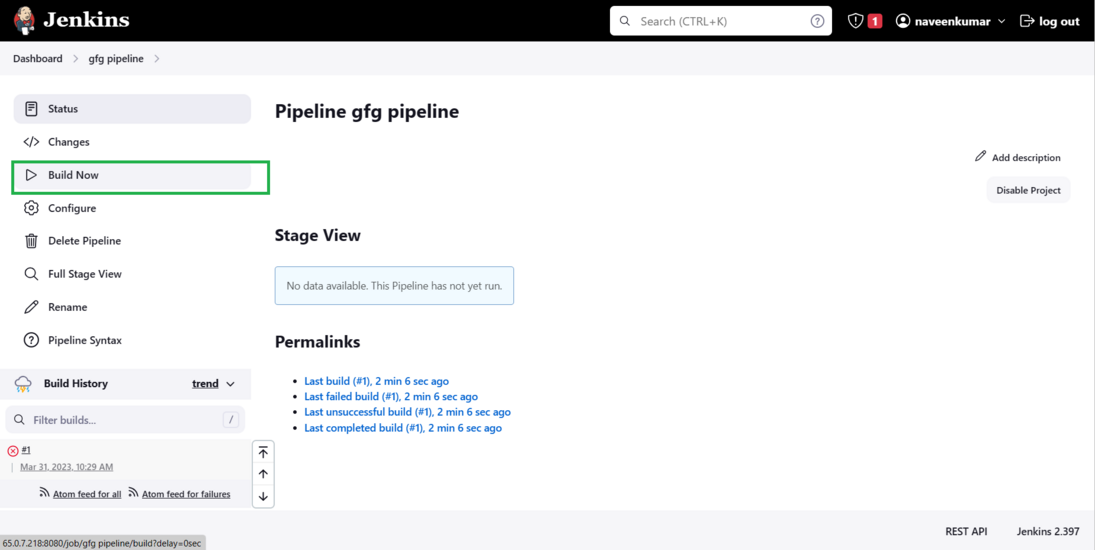


**Sample Pipelinescript To Deploy The Web Application Into The Tomcat Server**

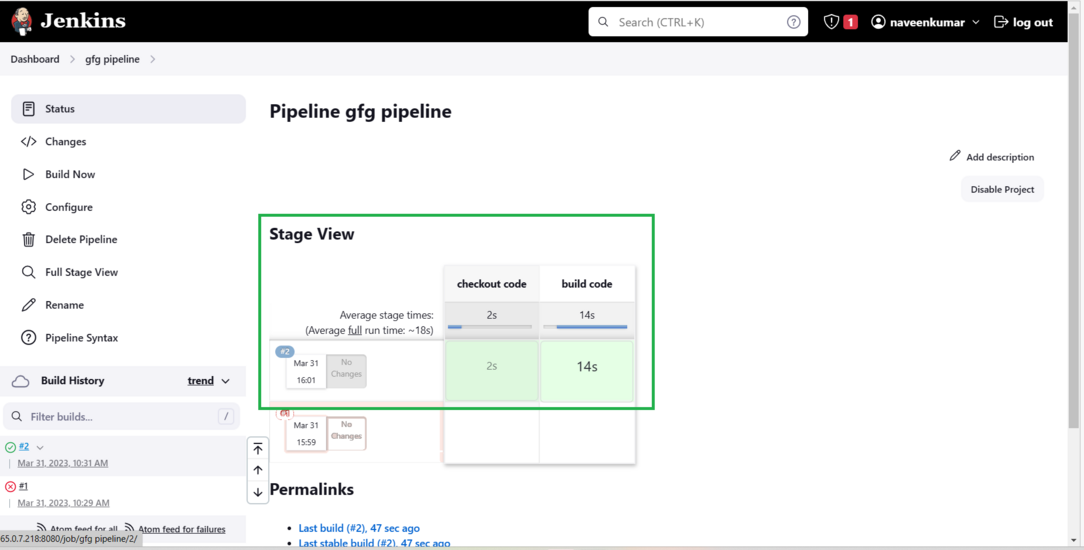
node  
{  
 //Mention the tools which have been configured  
   
 def mavenhome= tool name:"\*\*\*\*\*"  
   
 // Mention how to trigger the Pipeline and how many Builds must be there and so on   
   
 properties([buildDiscarder(logRotator(artifactDaysToKeepStr:   
 '', artifactNumToKeepStr: '5', daysToKeepStr: '  
 ', 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.

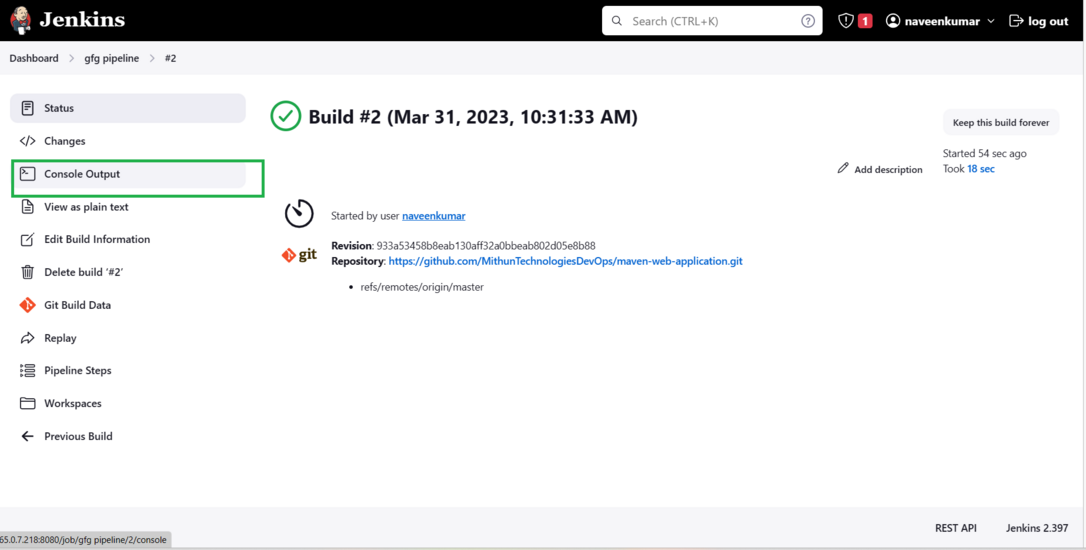
**Step 9.**After writing the pipeline is done click on save it will be directly redirected to the Dashboard of the project there we can use, the **“Build Now”**option to run the pipeline and check if it is successful or not, by using stage view or console output.



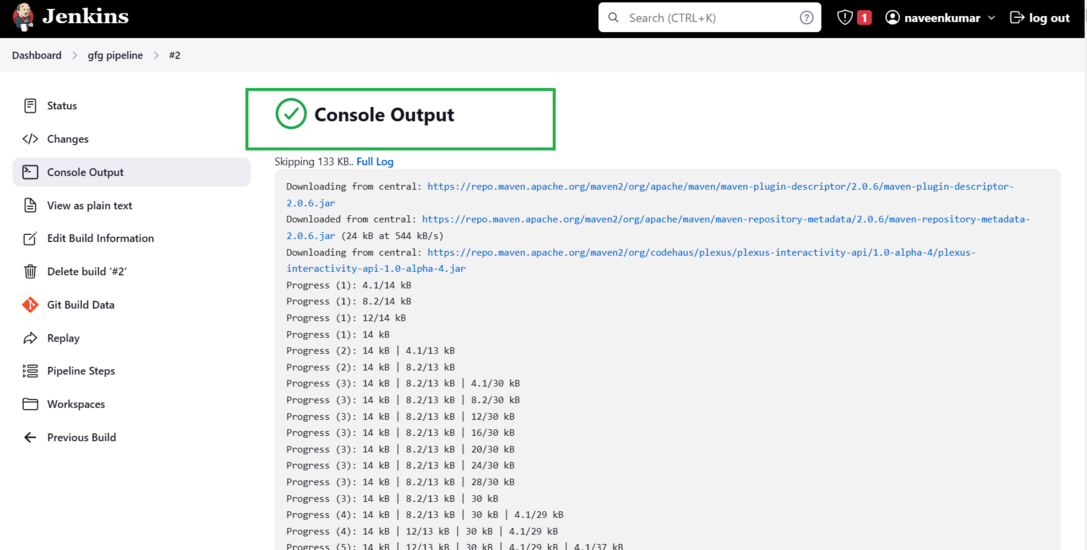
we can see the outcome of the pipeline in the stage view where as shown in the image below.

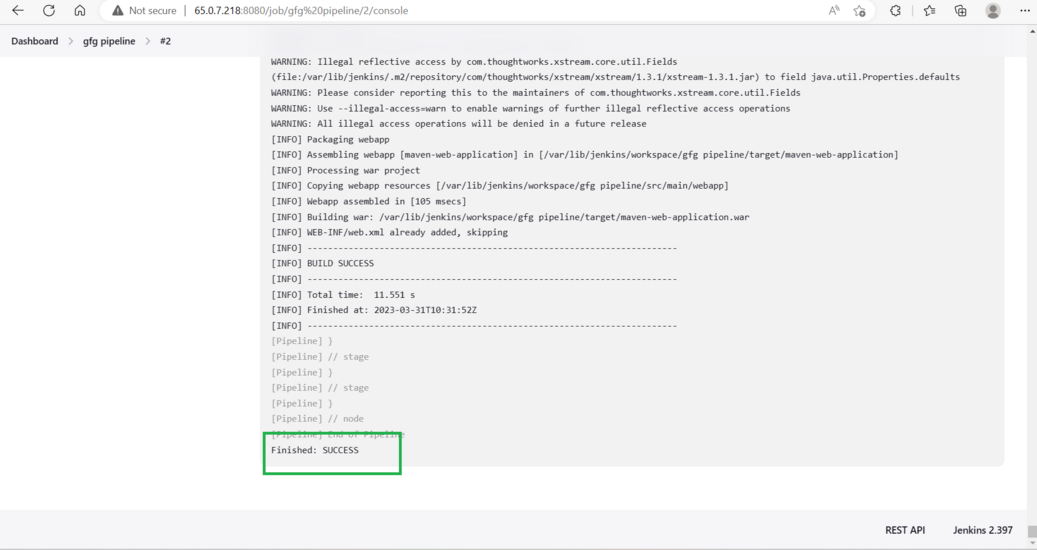


And we can also see the console output where we can see logs of each and every step which is performed.



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.