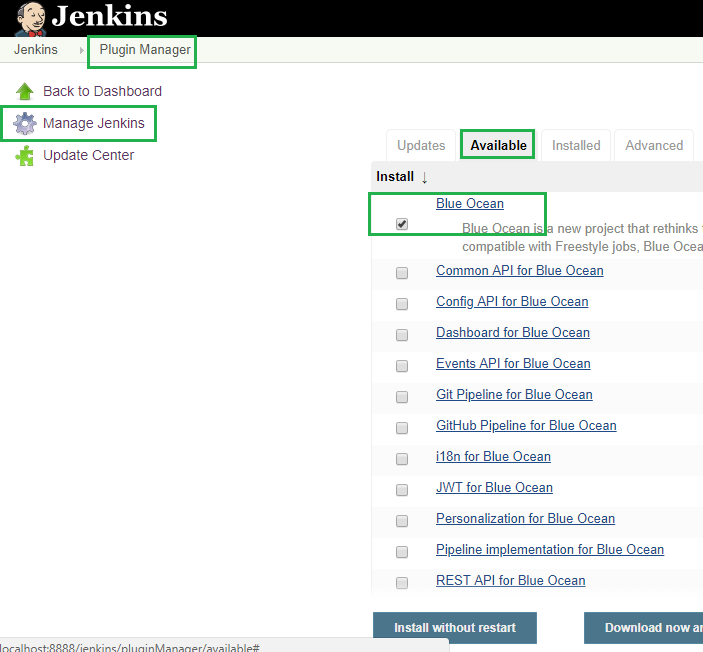
**Install Blue Ocean Plugin**:

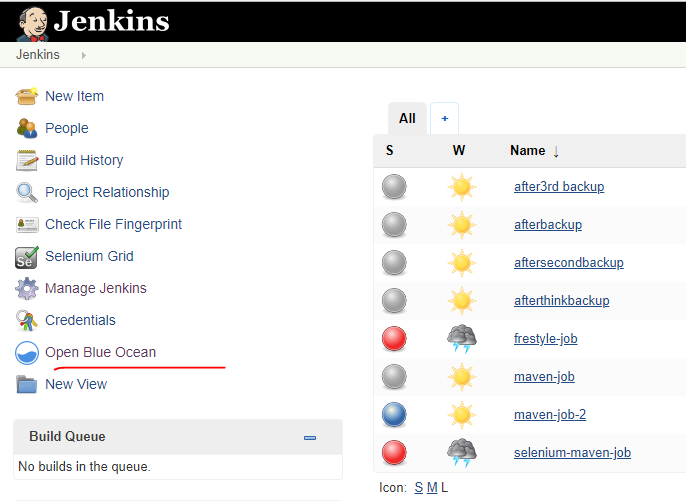
Manage Jenkins 🡪 Manage Plugins 🡪 Available 🡪 Search for “Blue Ocean” 🡪 Select it and Click on “Install without Restart”



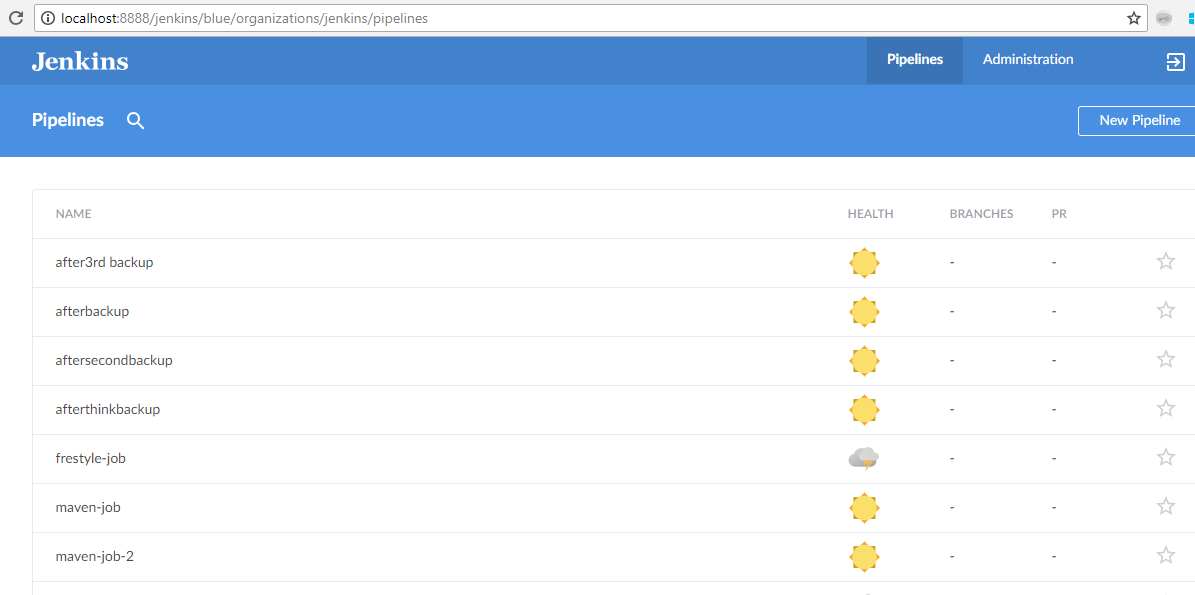
**Plugin Installation status**:



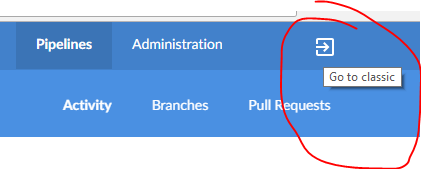
**Jenkins Home page**:



**Jobs view in Blue Ocean**: We can see/run the same jobs under “Blue Ocean” too.

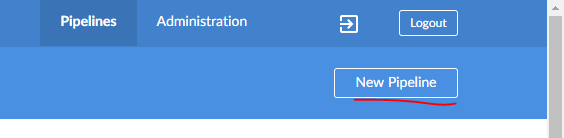


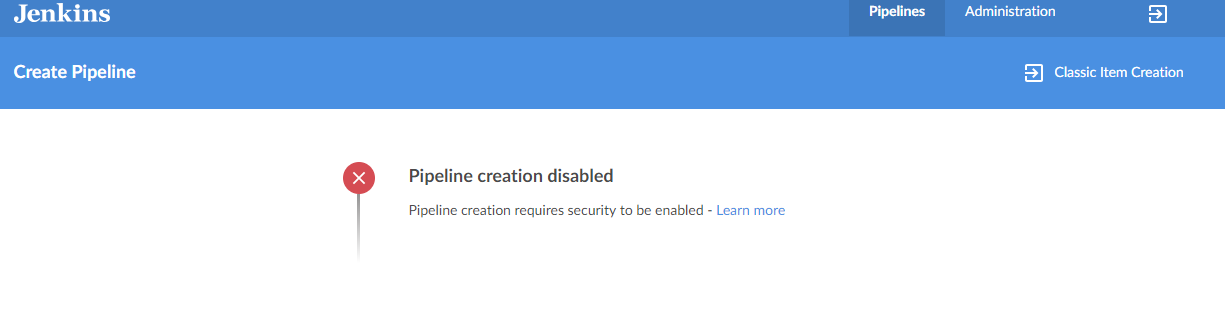
If you want to switch back to classic Jenkins view, click on the arrow as shown in below image.



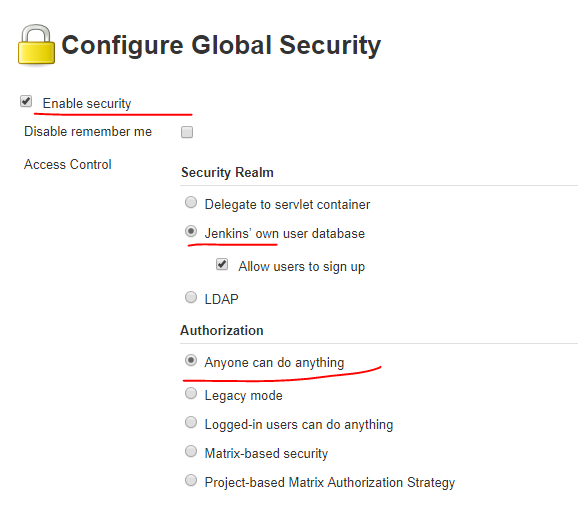
**Create a Pipeline**:

“Open Blue Ocean” 🡪 Click on “New Pipeline” button.

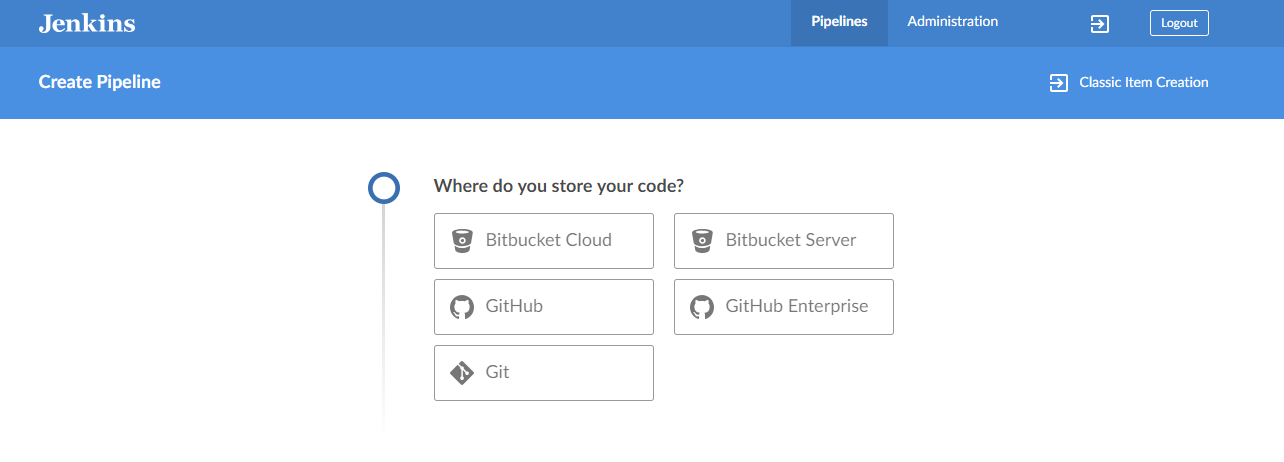




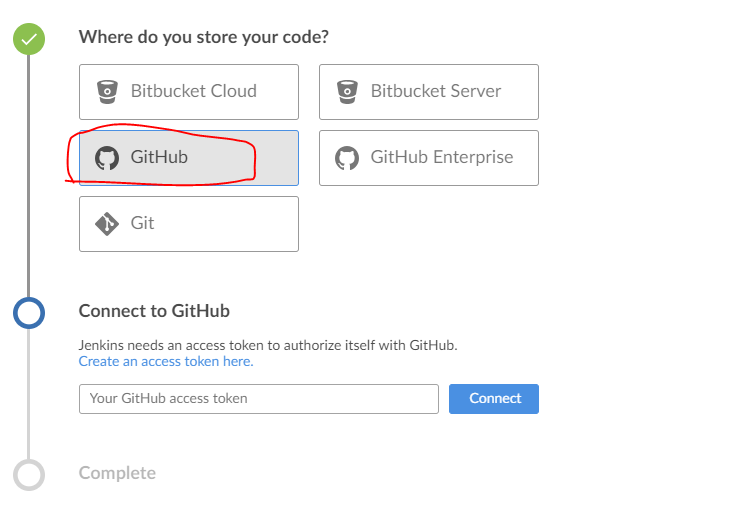
If you faced this issue as shown in above image, just do “Enable Security” as shown in below image. To enable it, go to “Manage Jenkins” 🡪 Configure Global Security 🡪 Enable Security.



Once you enabled the security or if it is enabled you will see the below image after you clicked on “New Pipeline” button. The entire purpose is to where to save the autogenerated pipeline script i.e., Jenkinsfile. For this, we need to connect to any version control system Git, GitHub, Bitbucket etc.

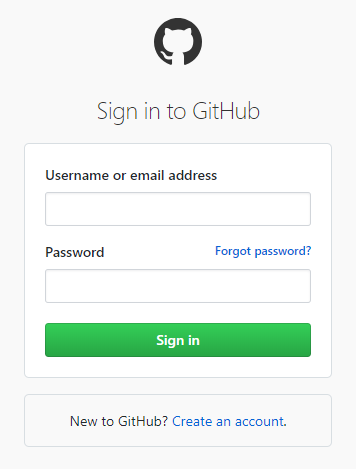


Here you can choose your SCM either Git, Bitbucket, GitHub etc. I am choosing GitHub.

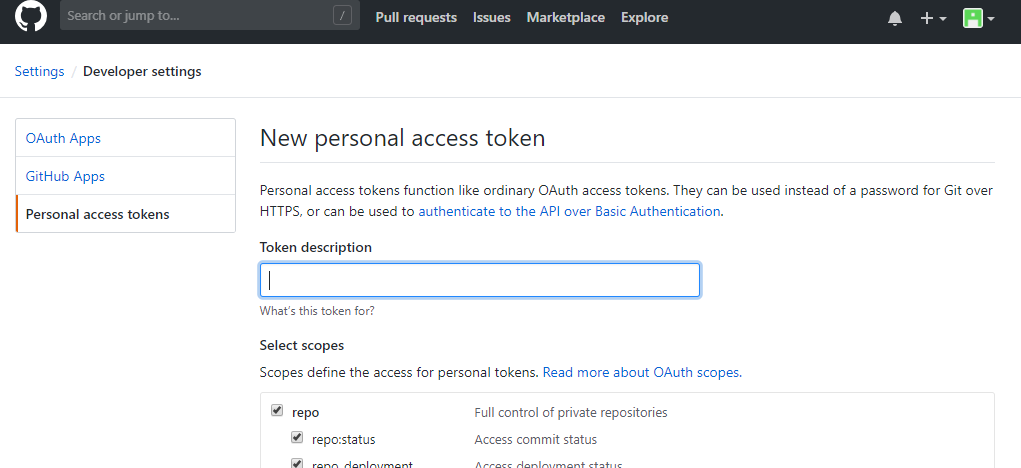


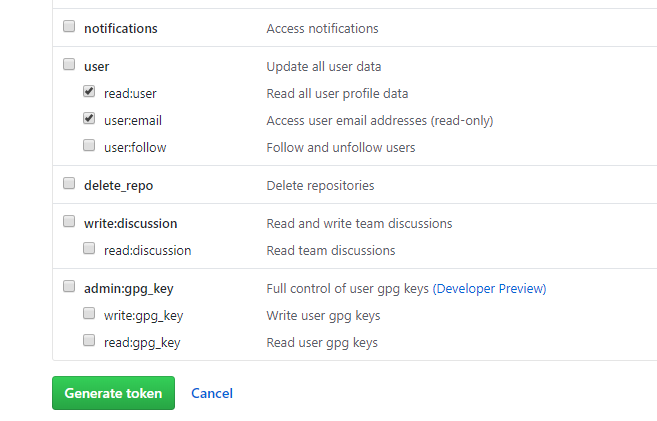
As showing in the above image, we need to generate the access token in GitHub and paste the token here. Click on “Create an access token here”, you will directly redirected to GtiHub website.

Sign In:

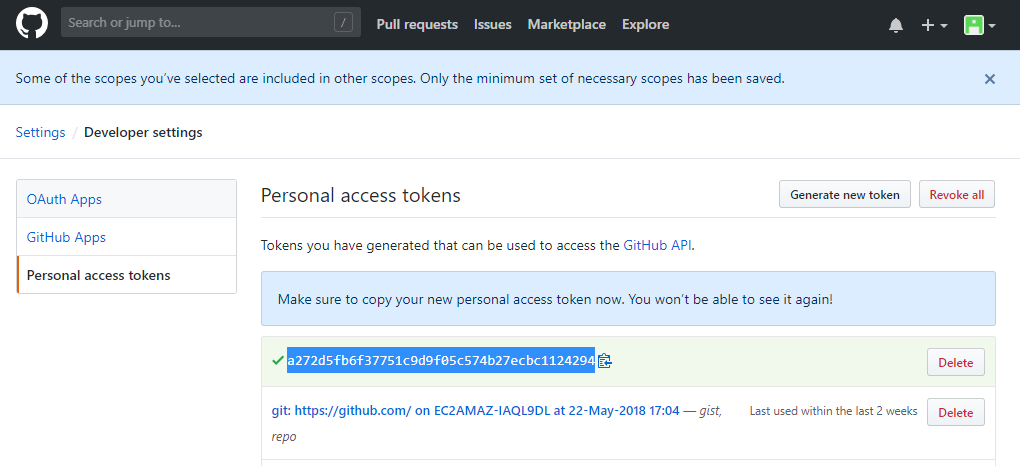


**Give a description about the key. And then click on “Generate Token”.**

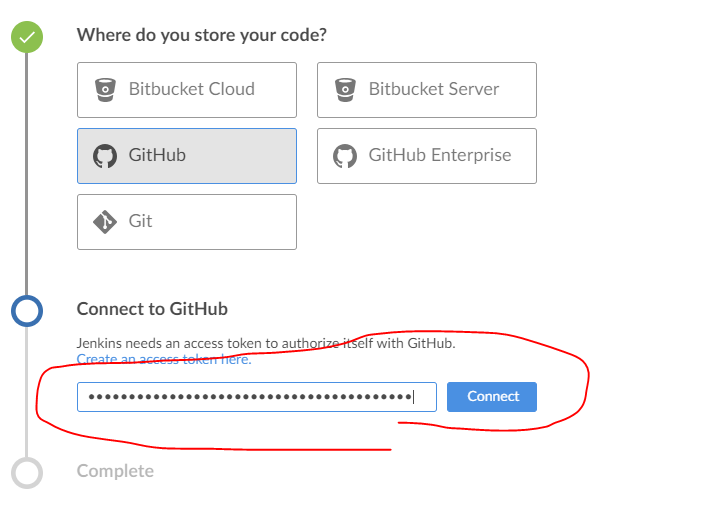




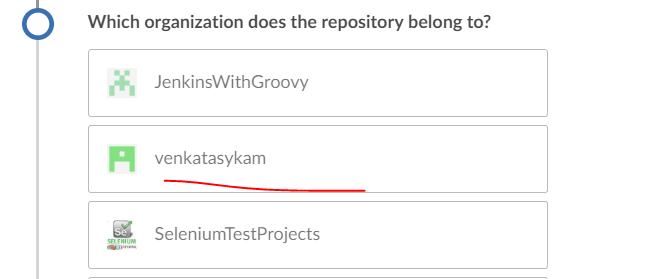
**Copy the key**



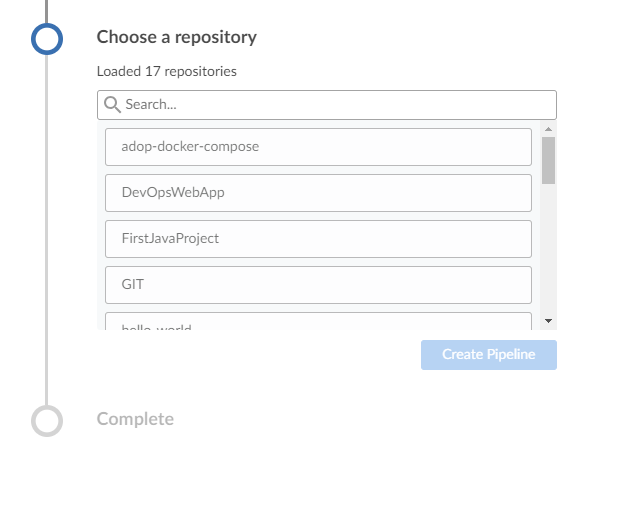
**Paste the key here as shown in below image.**



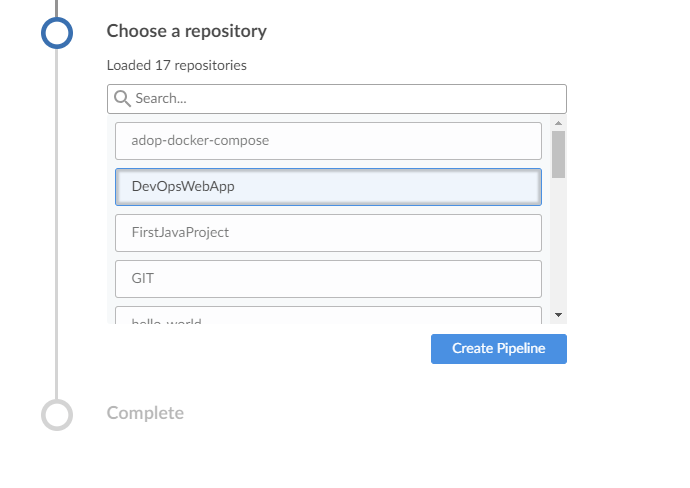
**Choose your GutHub organization or your GitHub account name.**



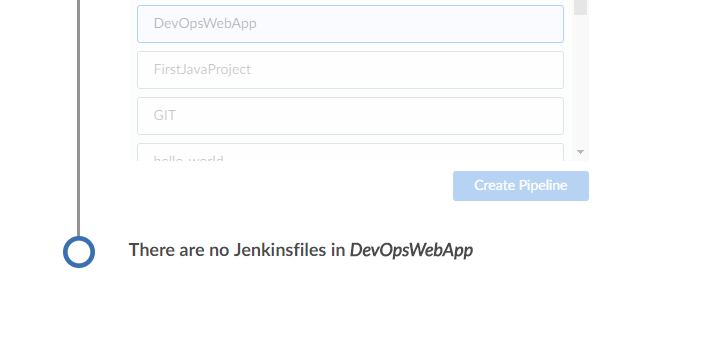
**Choose the repo you want:**



**Repo select and click on “Create Pipeline”**

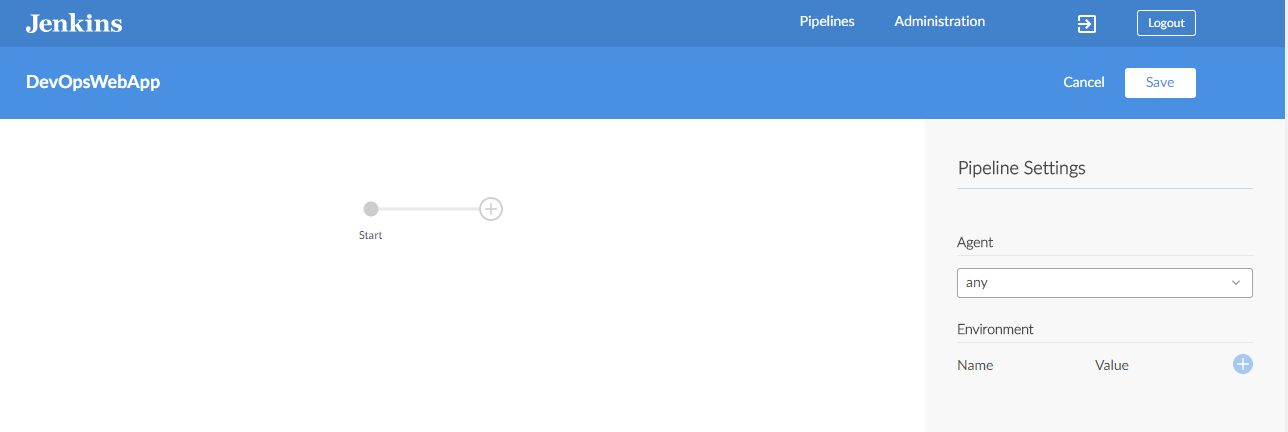


**Pipeline creating In-process:**

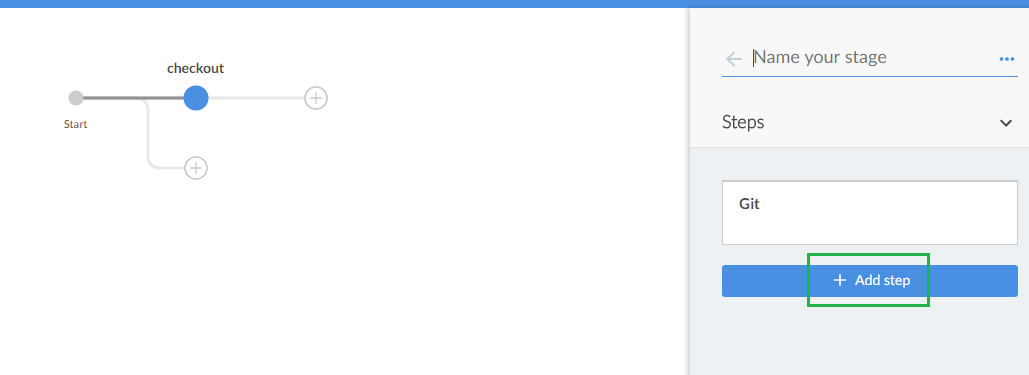


**This is your pipeline job: DevOpsWebapp (as same as GitHub repo name). Here you can add many stages you want. Ex: code checkout 🡪 Build 🡪 deploy 🡪 test etc**

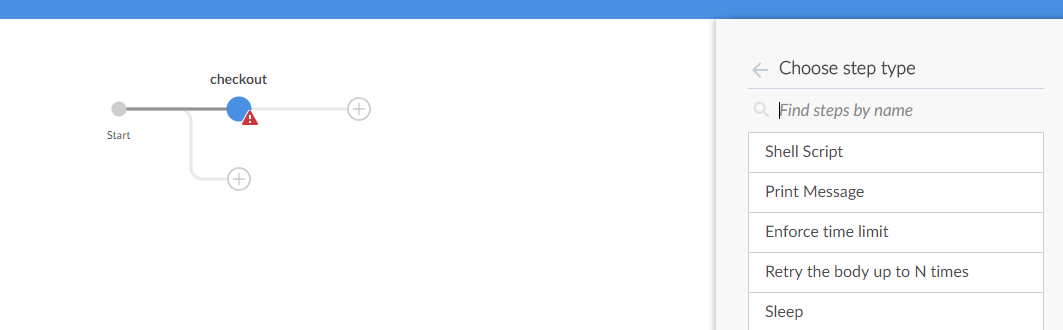
**Click on plus symbol to add a first stage.**



**We must give a name to stage & need to add at least a single step. Here I am giving a stage name as “checkout” & adding a step “Git”.**

****

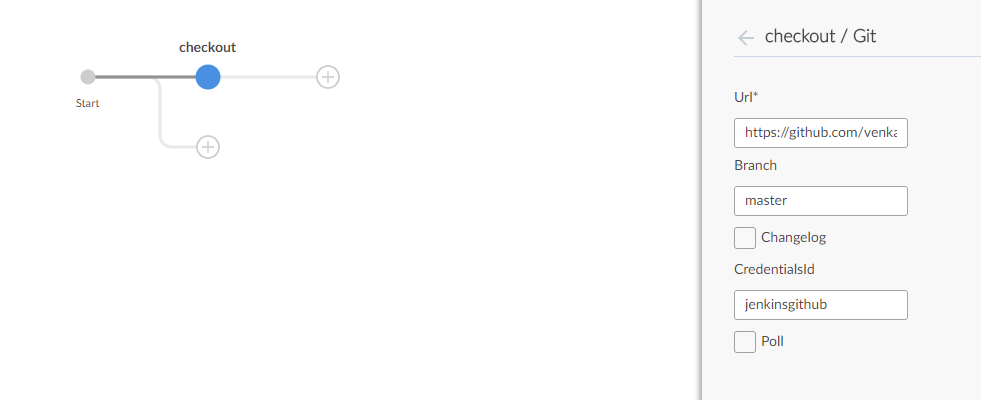
**Just click on “Add Step”. And then choose the step you want to add.**



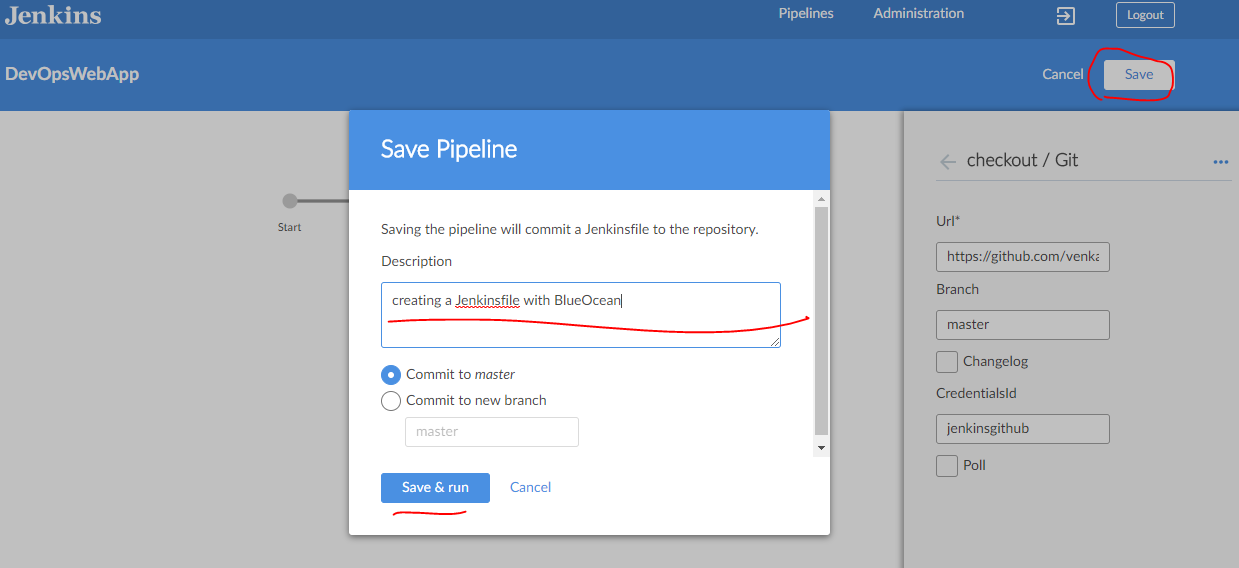
**Choose “Git” to check out the code as per our requirement.**



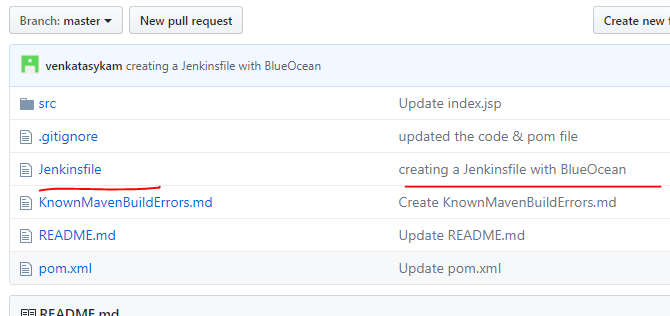
**In the step “Git”🡪 We must enter the project/repo url, branch name, credentials id details etc.**



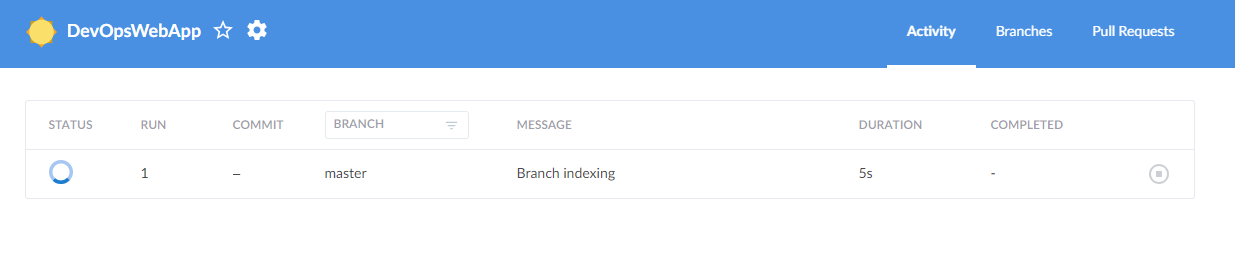
**Write a proper commit message & then click on “Save & run”.**



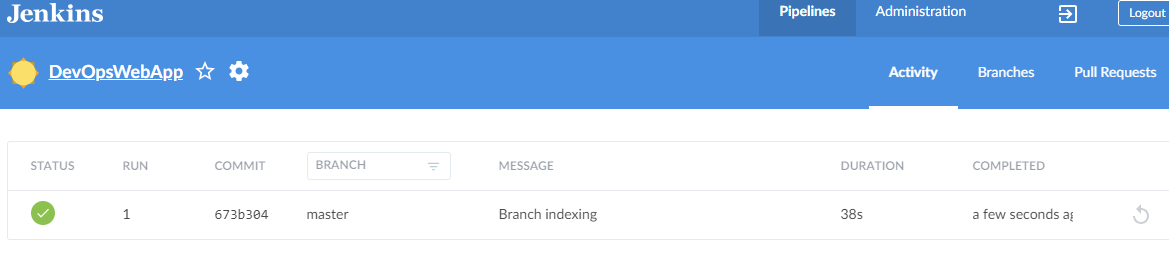
**Go to your github repo and see, ‘Jenkinsfile’ should be created once you clicked on “Save & run”.**



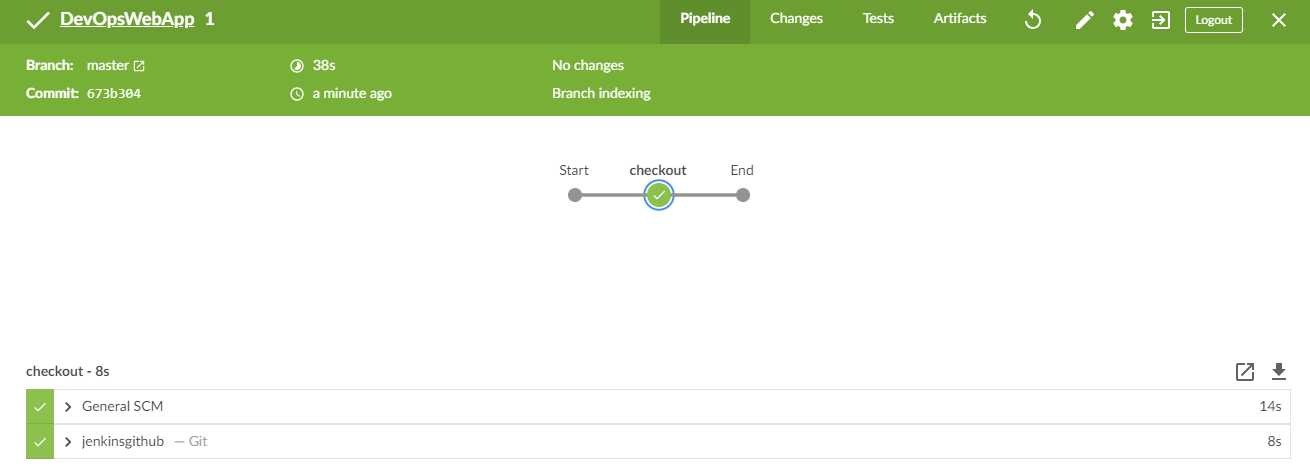
**Our pipeline Build is running. Build is running once the pipeline stages are saved.**



**Build is done.**

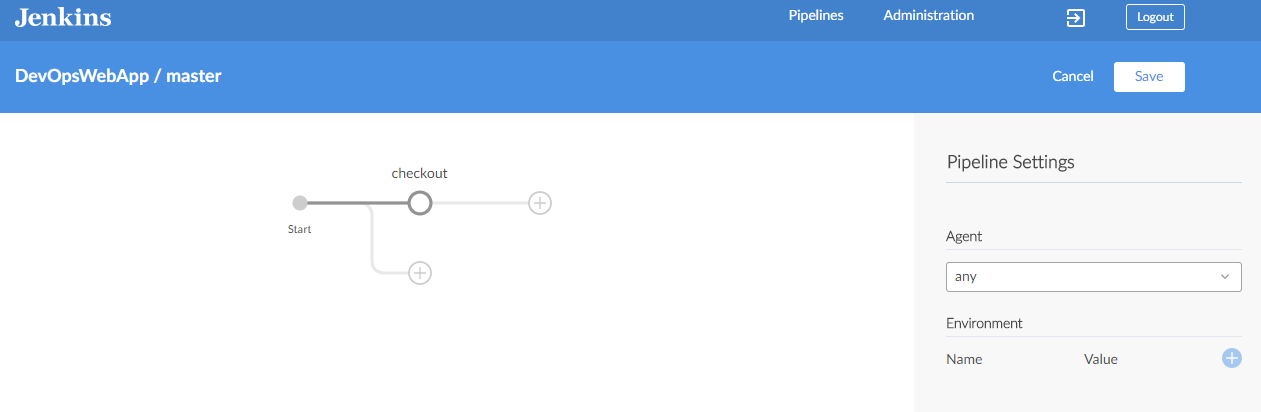


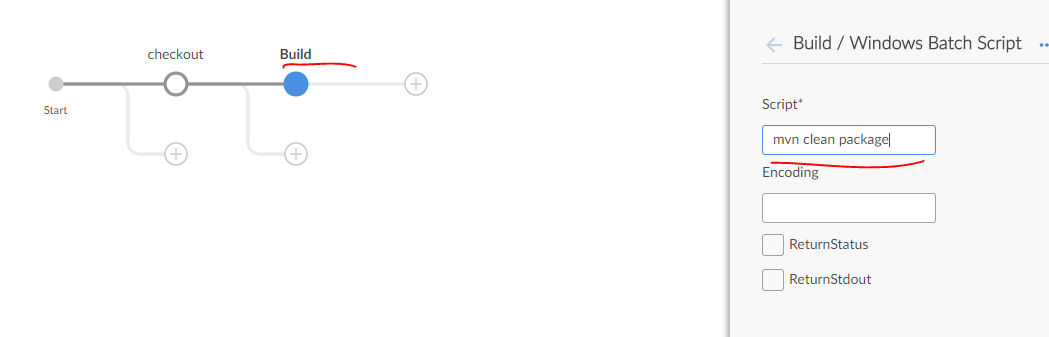
**Click on “1” under “Run”**

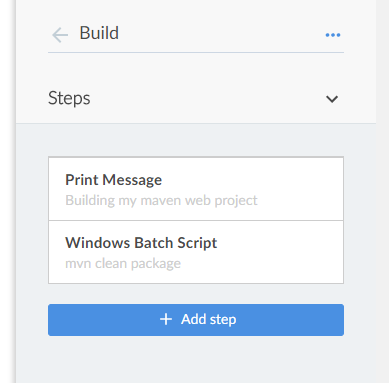


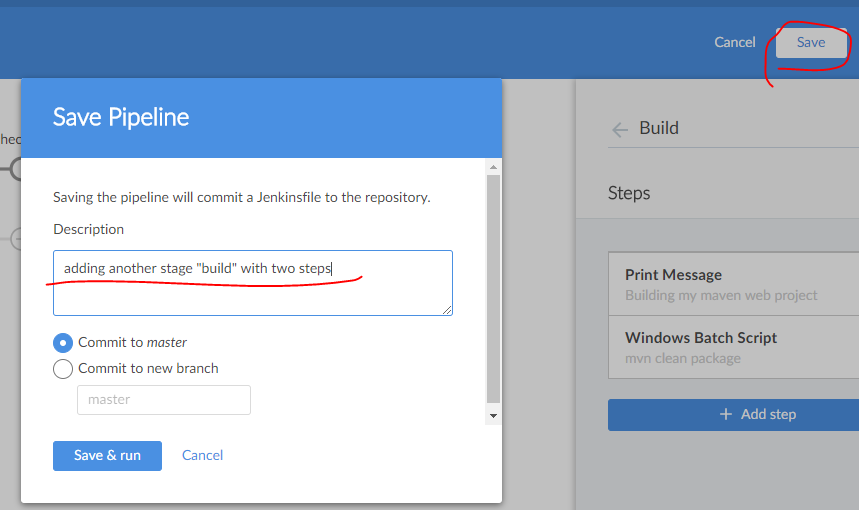
**Click on Edit to add more stages.**



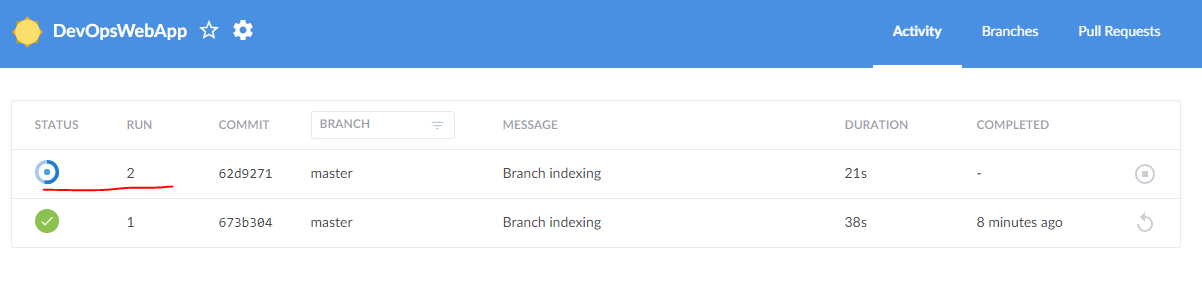




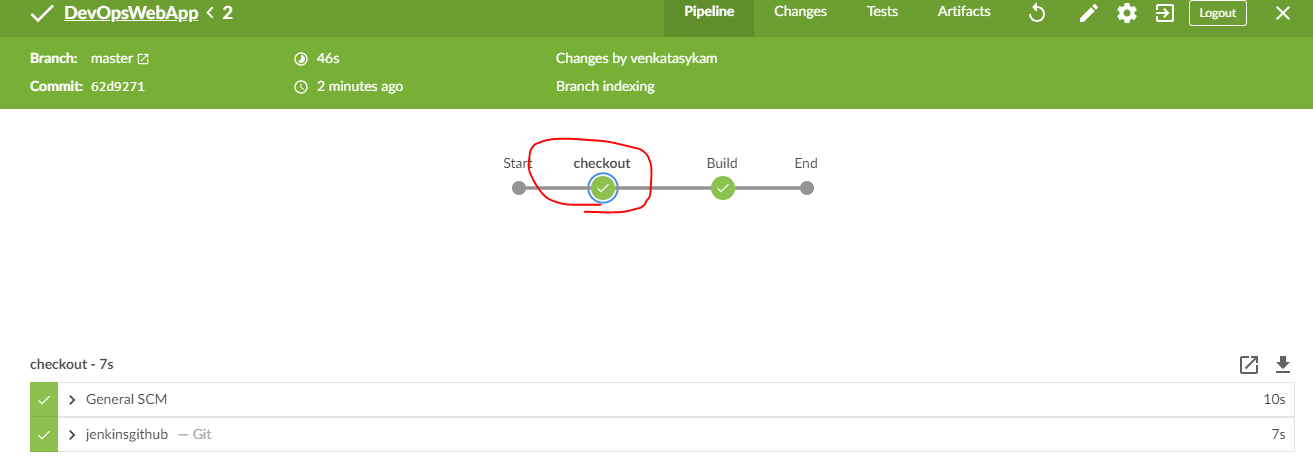




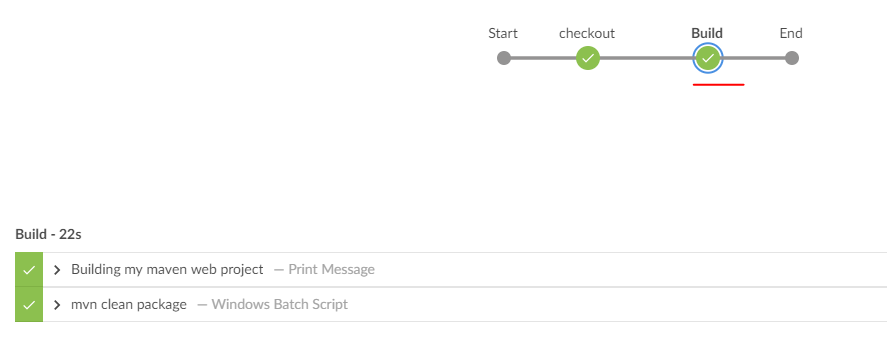
**Give me proper commit message & click on “Save & run”. Once the new stage saved, job triggered automatically. Click on build#2 will see how many stages are running, what exactly is running in each stage etc.**



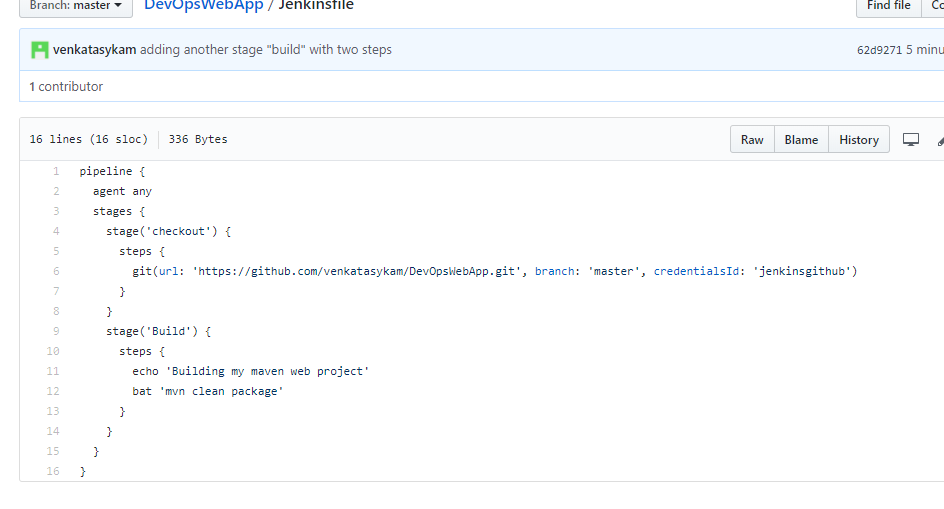
**Stage-1: Code checkout**



**Stage-2: Code Build**



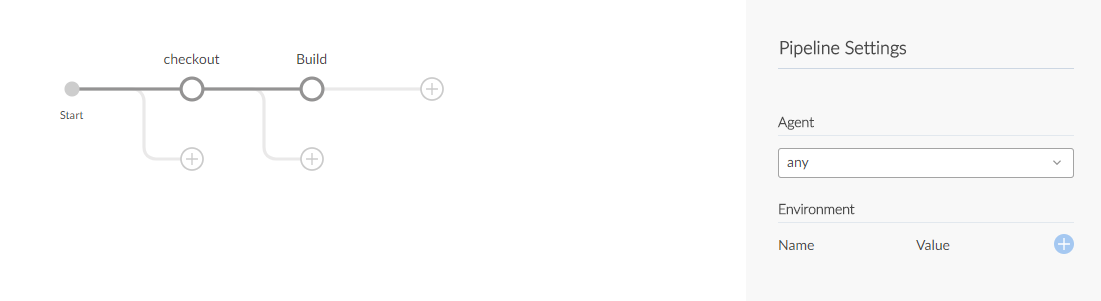
**See the pipeline script which is auto generated by BlueOcean plugin.**

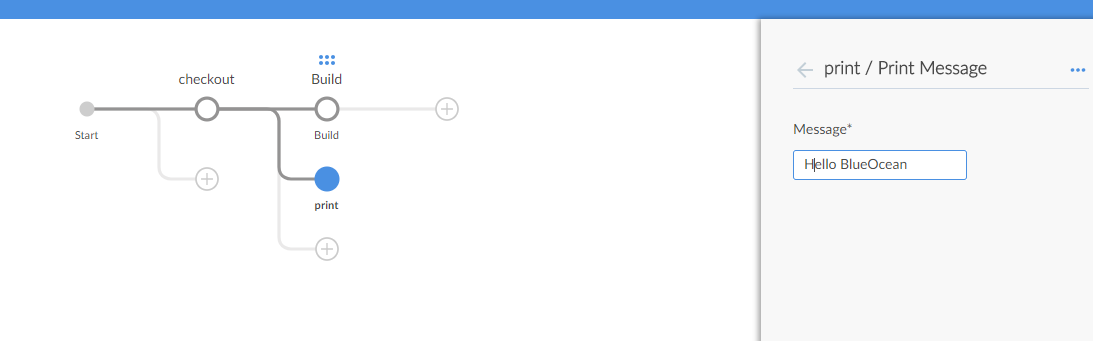


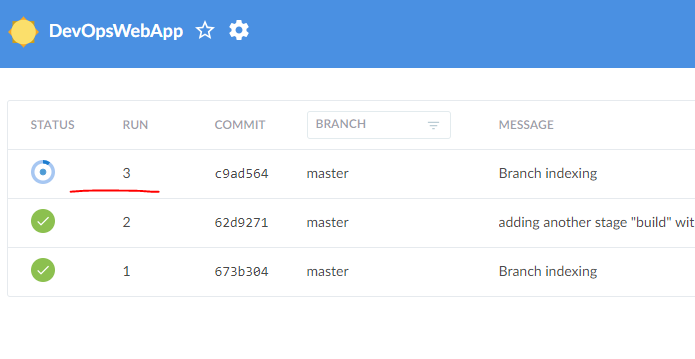
**Add a parallel Stage:**

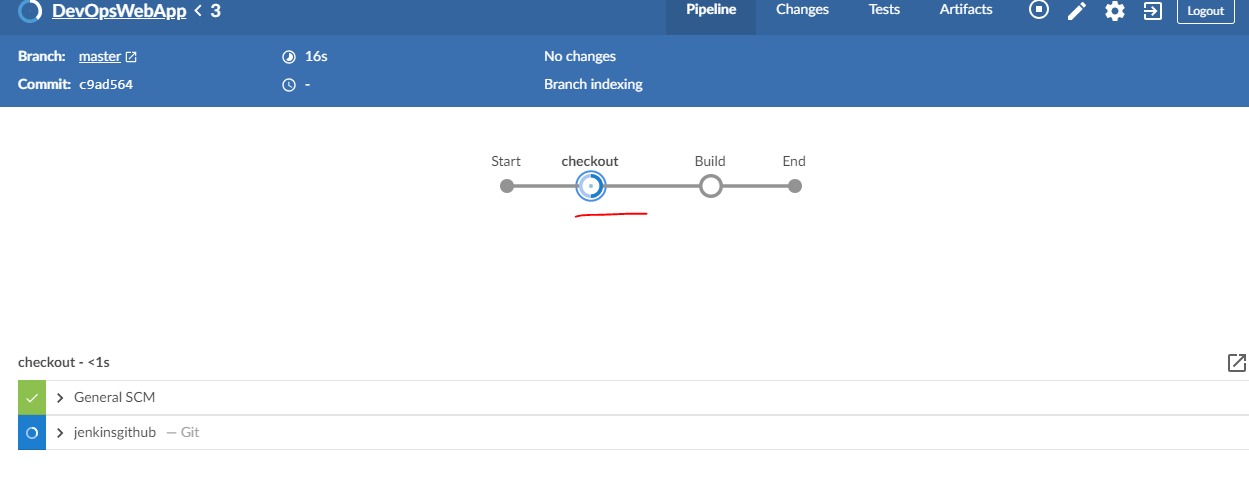
**Click on Edit to add more steps**

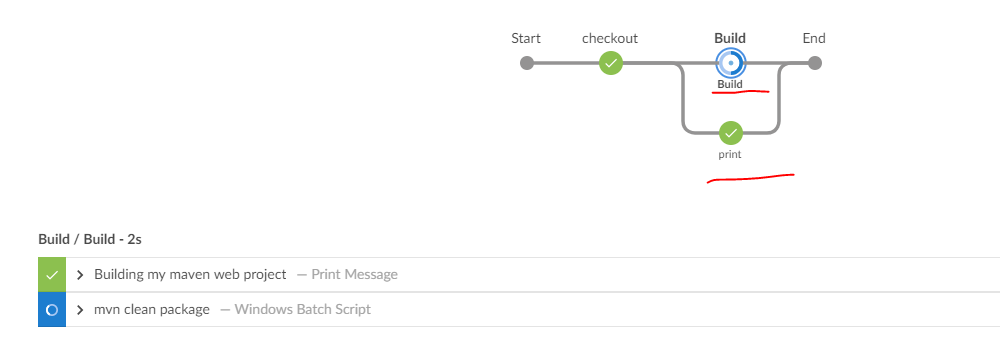


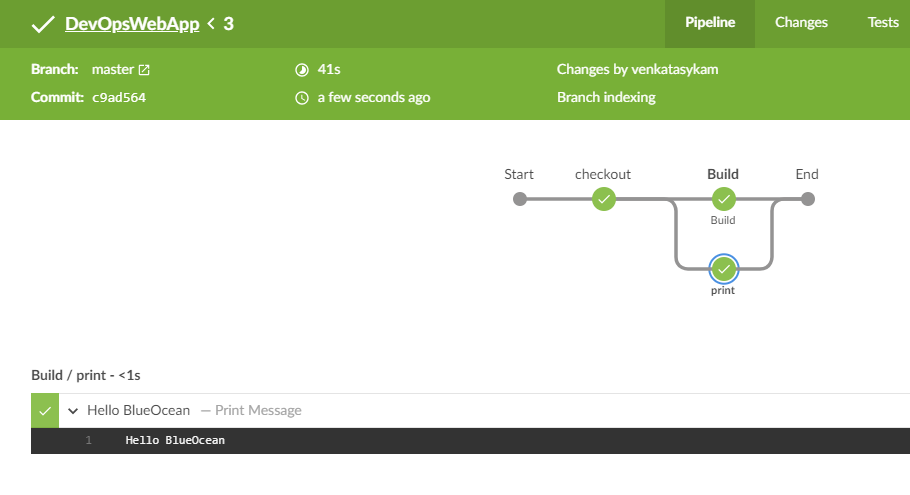






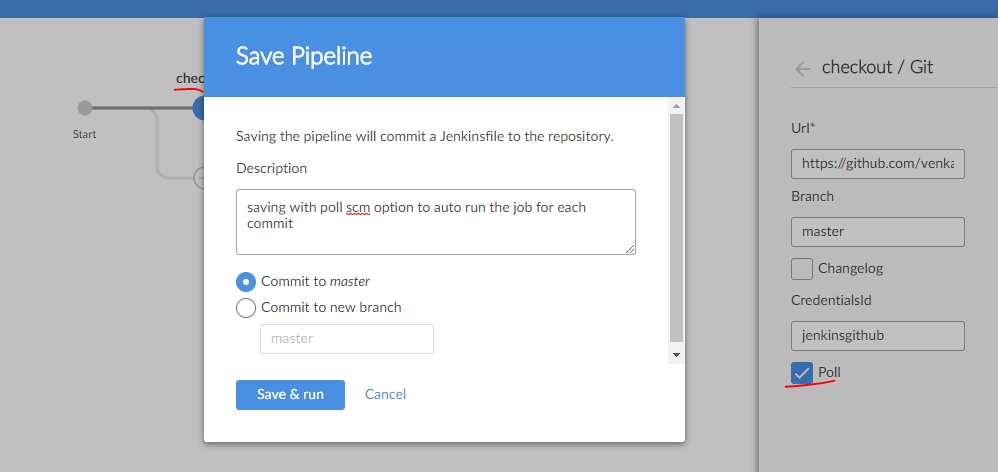




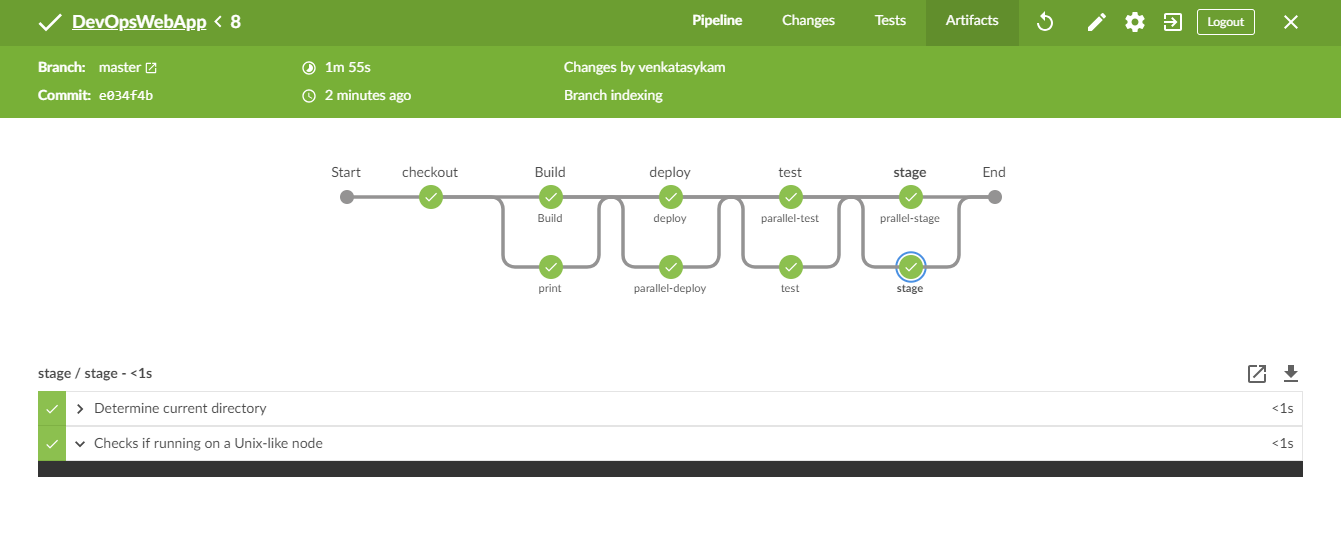


Auto trigger the build once the commit/changes done in GitHub repo: (make sure you integrated Jenkins with Github)

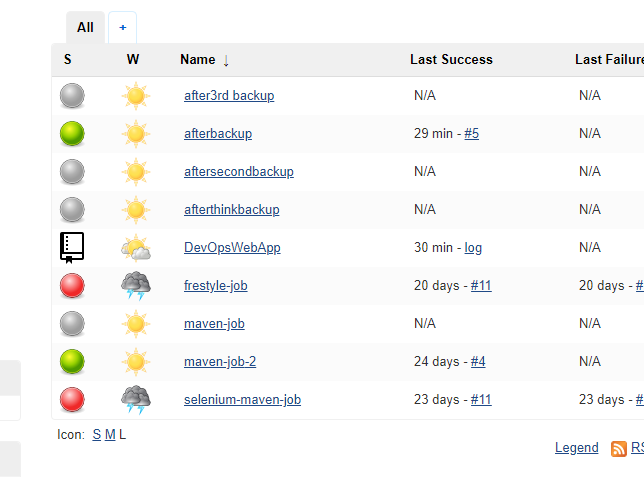
Choose the “Poll option” to auto trigger the Jenkins job “DevOpsWebapp”



Example: An overview of all the stages.

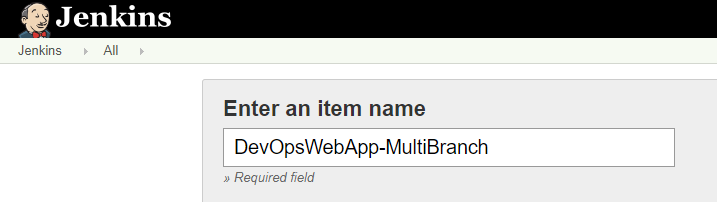


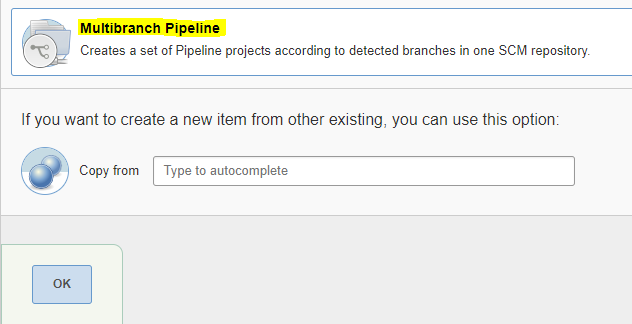
Switched to classic view**:** It’s a multi branch pipeline view.

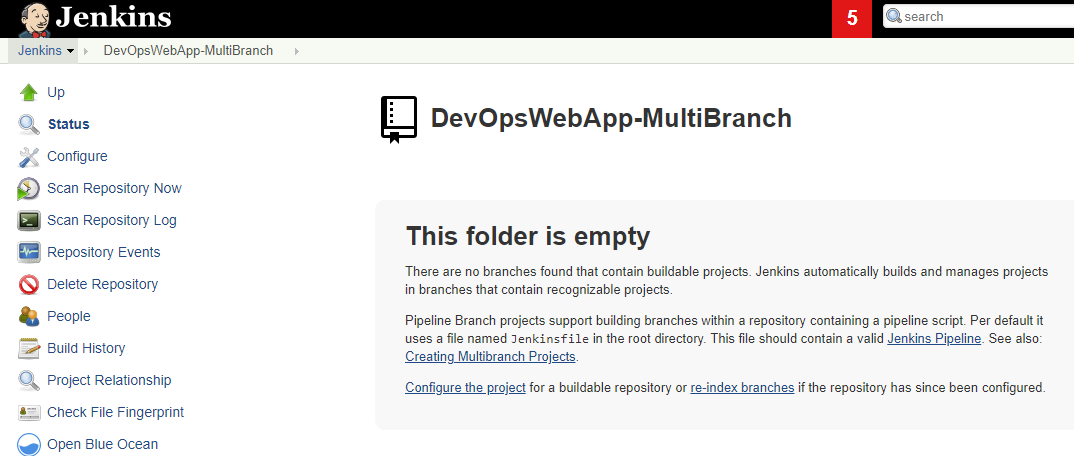


**Let’s create a new item/job with multibranch template:**

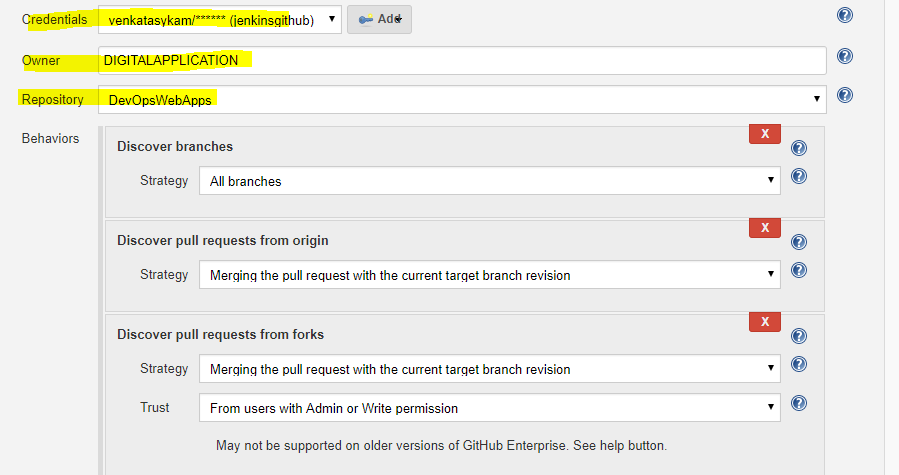
**Click on new item 🡪 Enter Item Name 🡪 Choose the project template as “Multibranch Pipeline” 🡪 Click OK.**



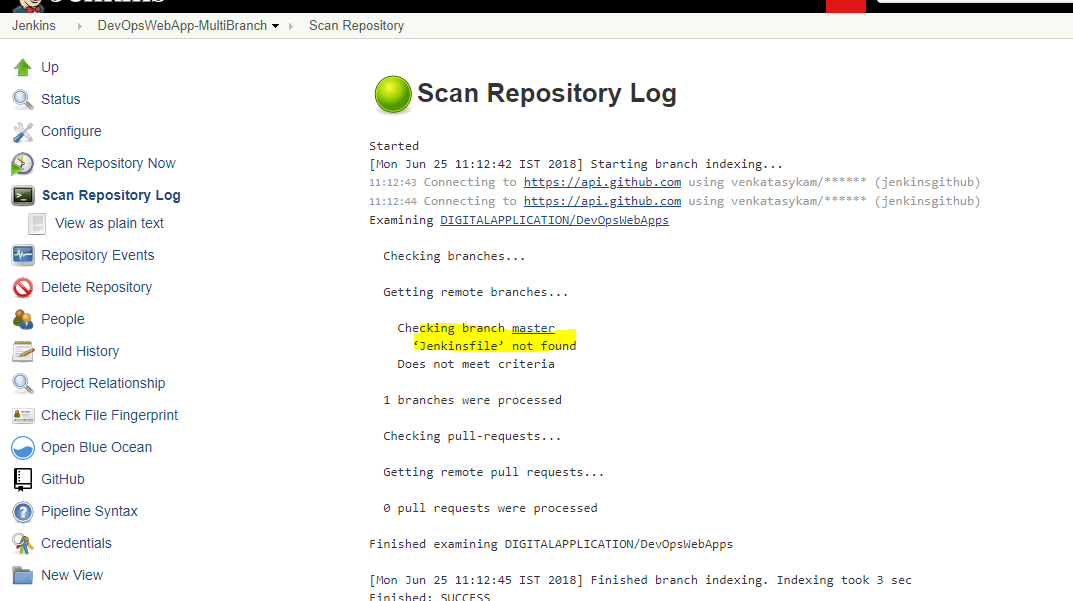




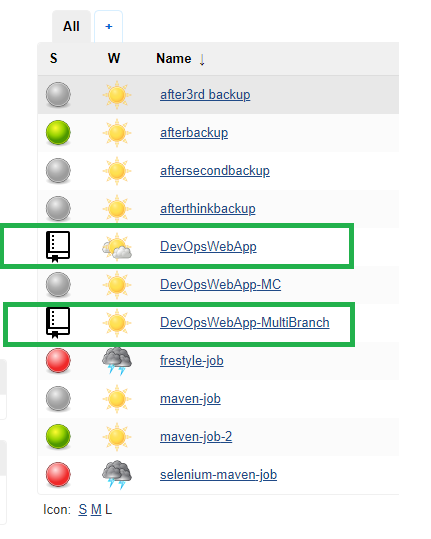
**Configure 🡪 add the github credentials, owners, repos.**



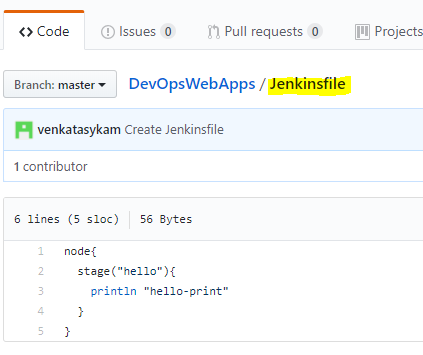
Once the job config saved, job will scan **all the branches** for “Jenkinsfile” in all the branches & it will try to execute the “Jenkinsfile” (pipeline script)



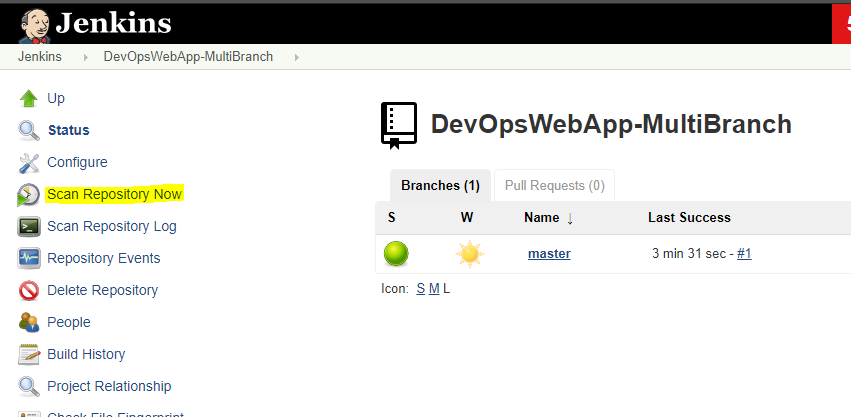
**Whatever we are creating pipeline with BlueOcean, that is a Jenkins “Multibranch Pipeline” project.**

****

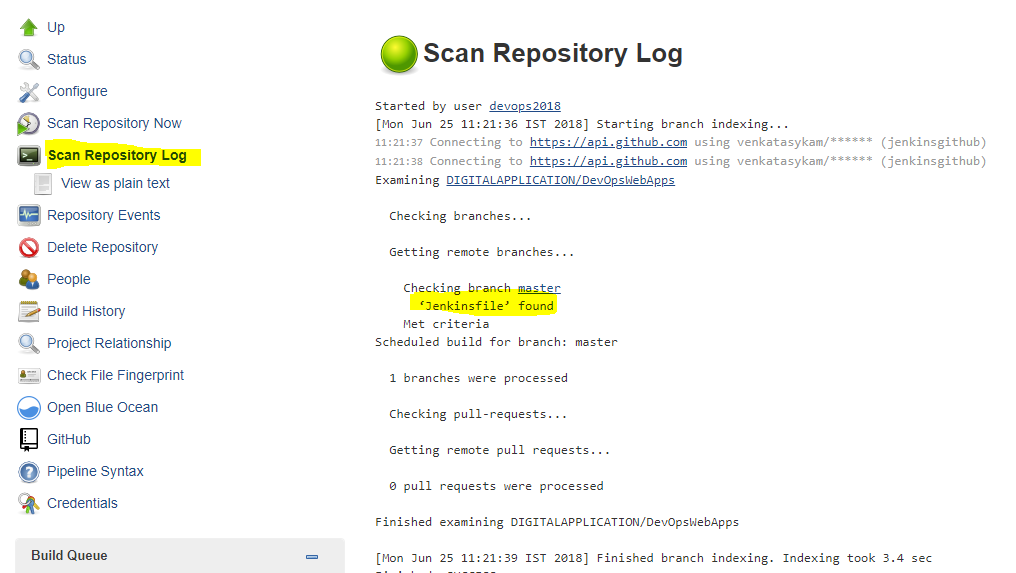
**Created a Jenkins file in GitHub repo and written a simple pipeline script.**



**Go to Jenkins job 🡪 Click on “Scan Repository Now”.**



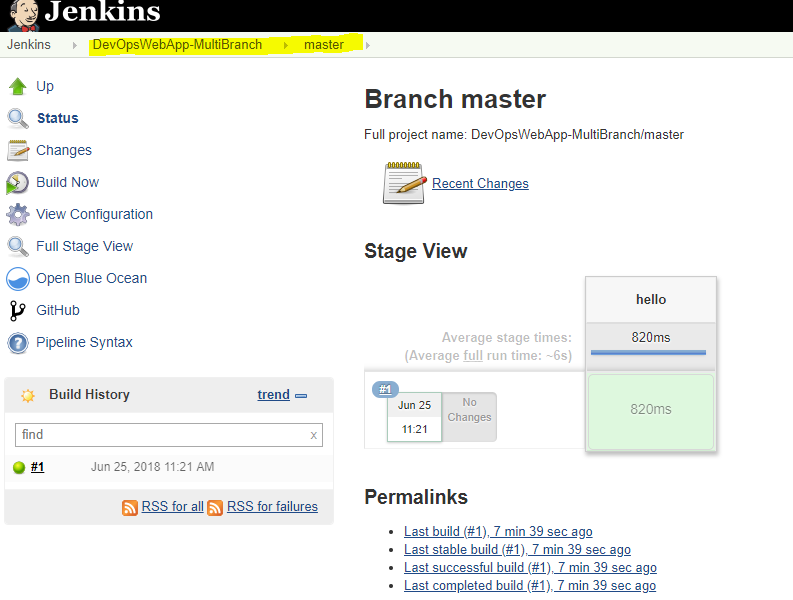
**Click on “Scan Repository Log” to check the status of the scan log. This time Jenkinsfile found and executed.**



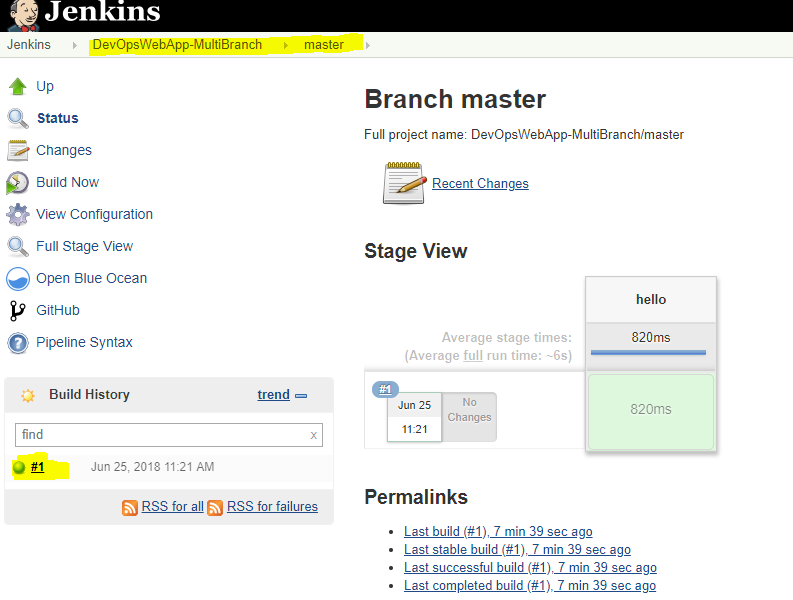
**For every branch, job is triggered & executed the pipeline script from “Jenskinfile”. Click on “master”, the build which is triggered for the branch “master” since the “Jenkinsfile” found in this branch.**

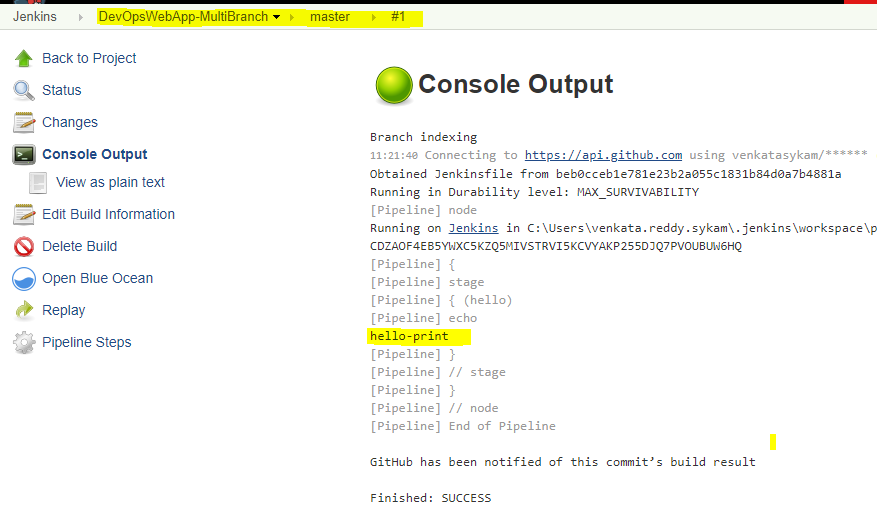


**Once you clicked on “master”, you will be navigated to the page (as show in below). Which is exactly same as any other Jenkins job.**

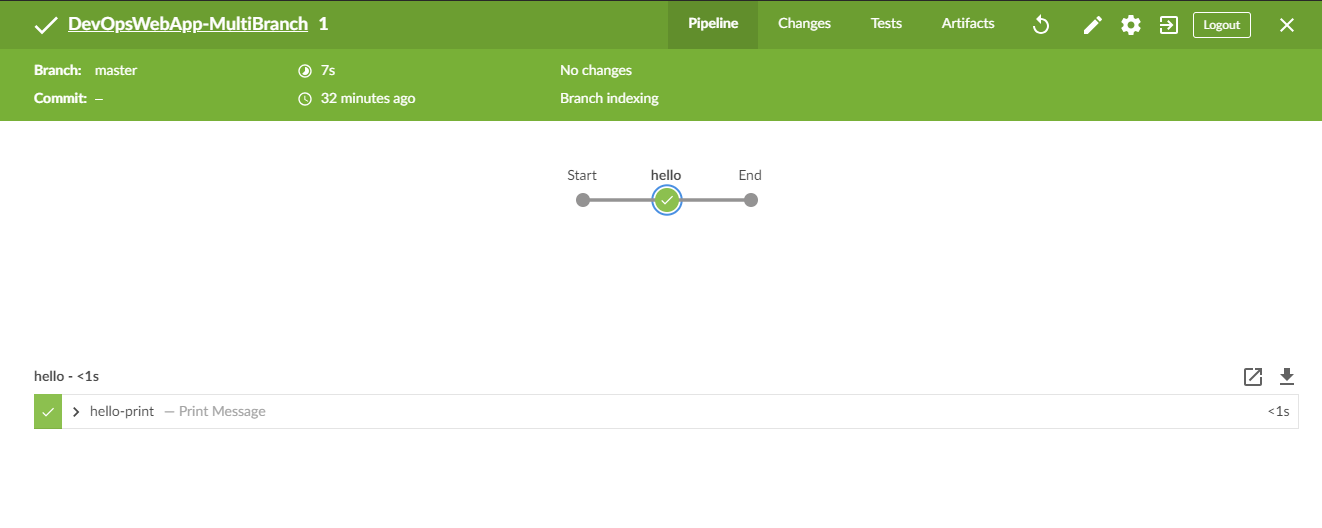


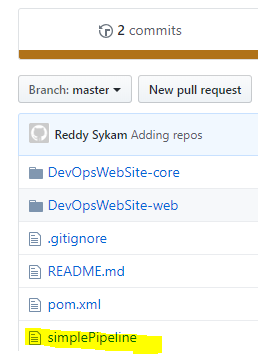
**Click on Build#1 to check the build console**



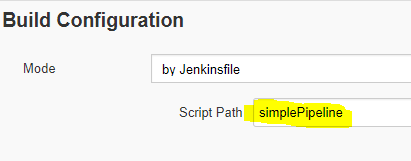


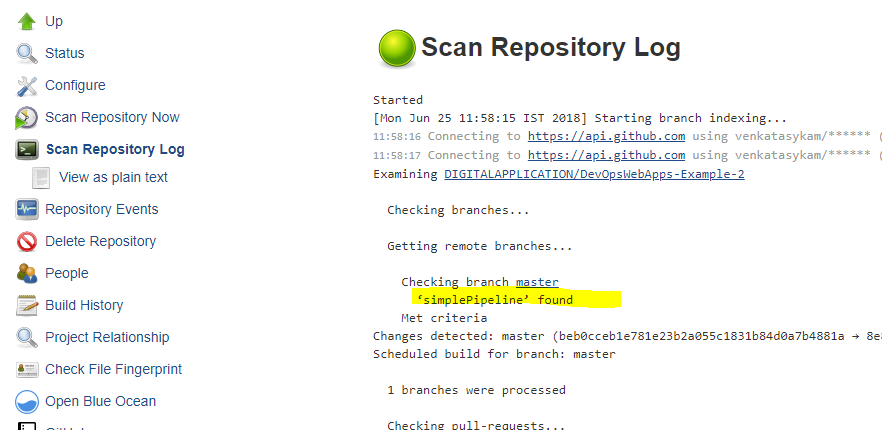
**Open the the same job in Blue Ocean.**

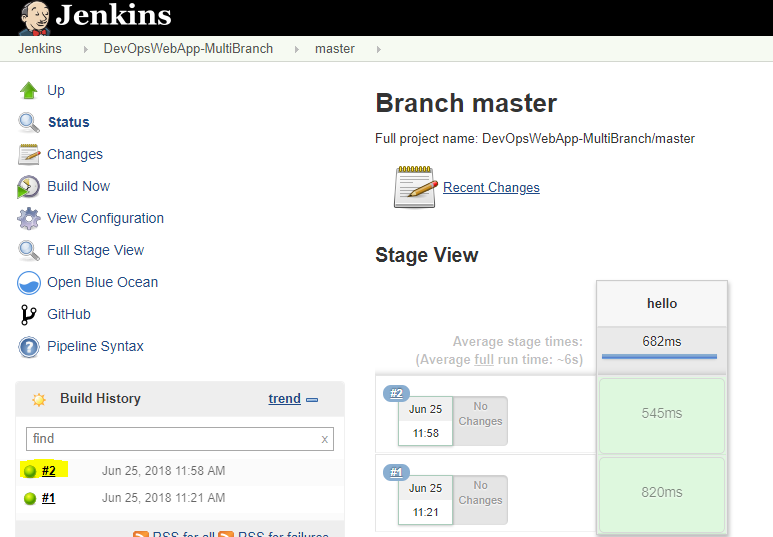
**Not mandatory to keep the pipeline script file name as “Jenkinsfile”, we can give any name. Just rename the file and update the job config with new file name. (By default Jenkins job configuration refer “Jenkinsfile”)**



**Job config:**



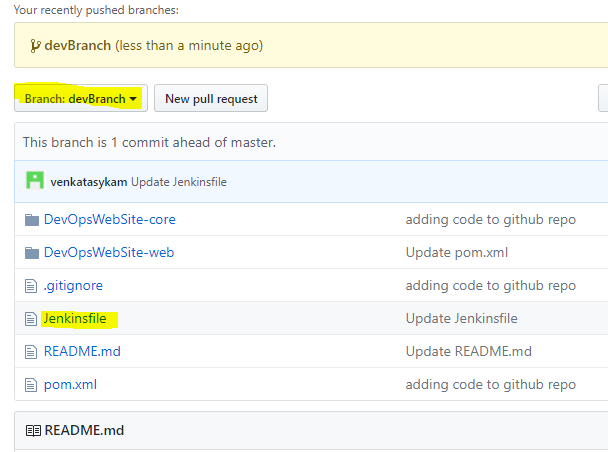


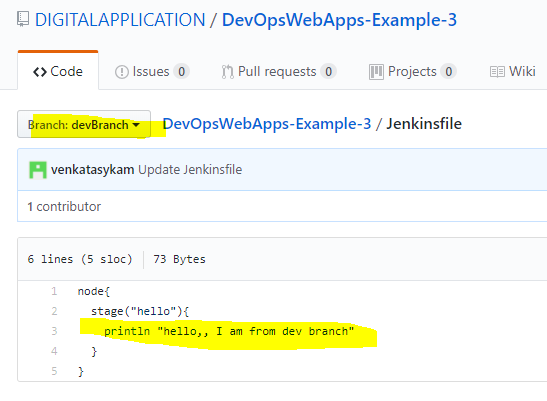


**Example with Multi branches in guithub: So far we have only one branch in our git repo.**

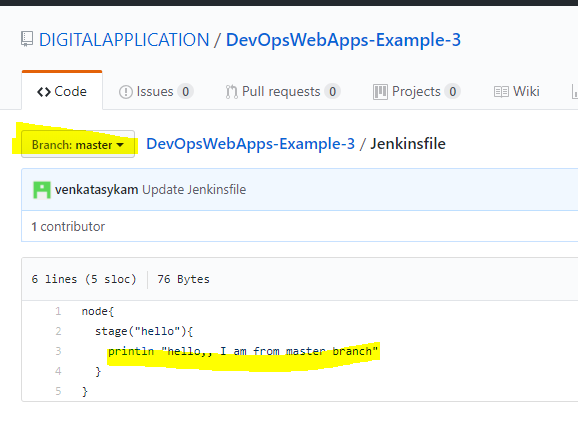
**Created another repo for our practice:** <https://github.com/DIGITALAPPLICATION/DevOpsWebApps-Example-3>

**Created a new branch “devBranch” from ‘master’ & updated the pipeline script as below.**

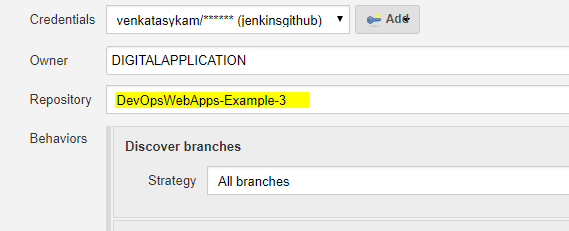




**Pipeline script from “master” branch:**



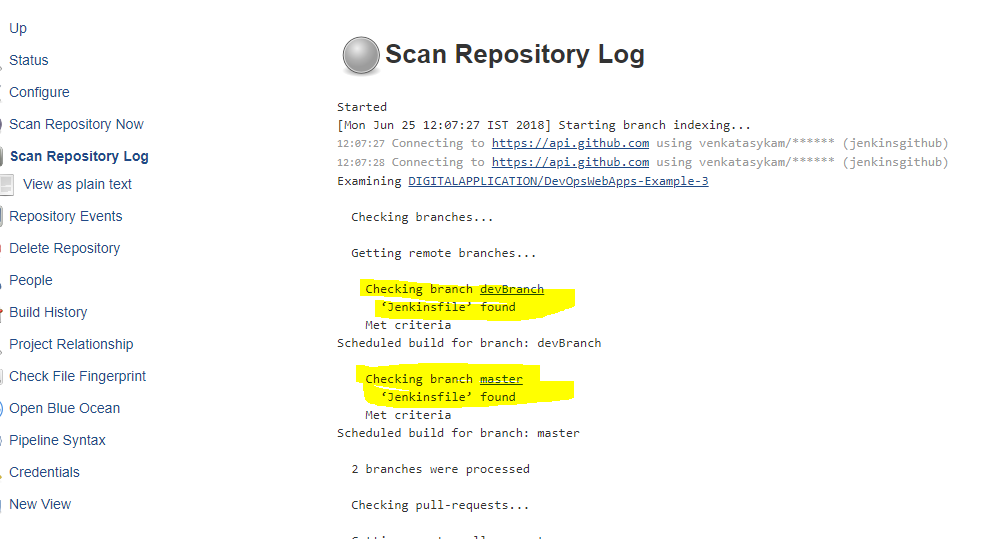
**Crete a new Jenkins job for this repo: DevOpsWebApps-Example-3**



**Nothing to update, default config.**

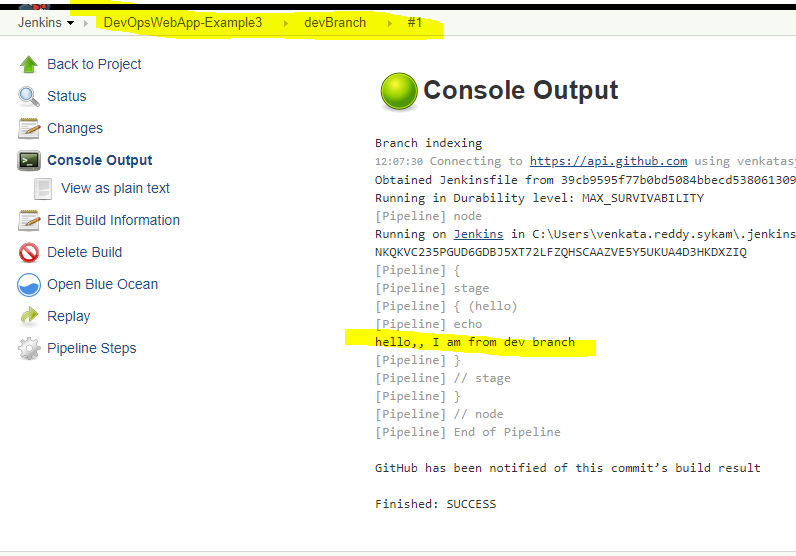


**Save the job config: Auto scan started & running the pipeline script as configured in “Jenkinsfile”.**

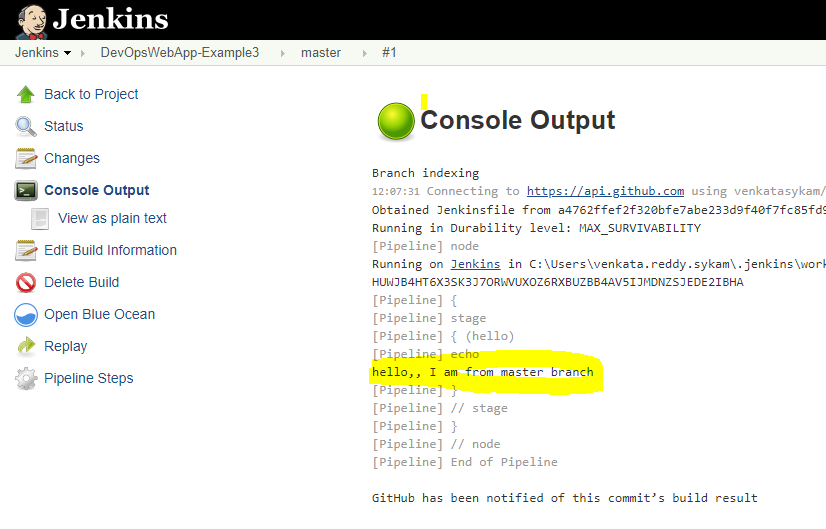




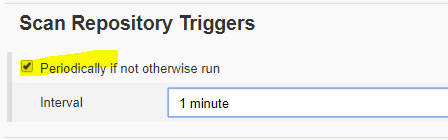
**devBranch console:**



**Build Console for master:**

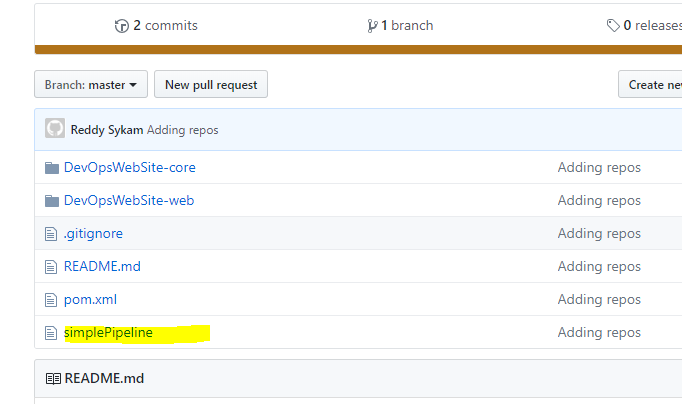


**Auto trigger the job or each commit/push: Job Configuration 🡪 Choose the below option 🡪 Select the time interval.**



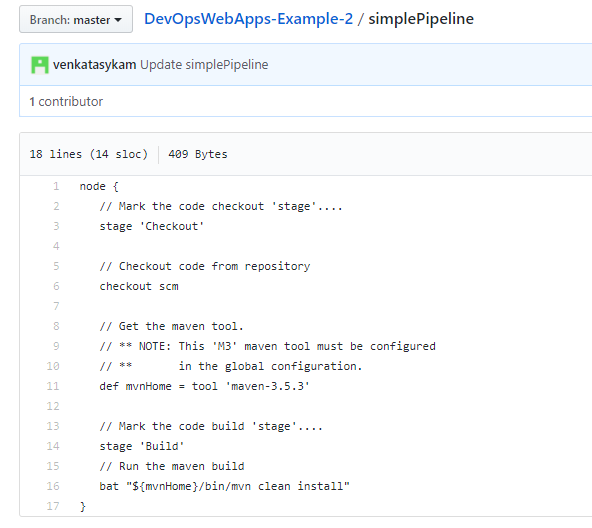
**Save the configuration and do some check-ins in the branch.**

**Another example:** <https://github.com/DIGITALAPPLICATION/DevOpsWebApps-Example-2.git>

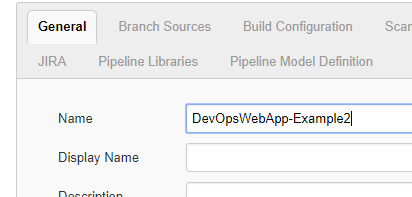


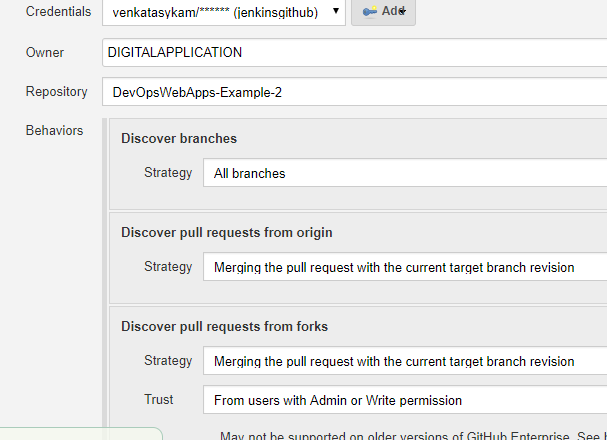
**Updating the pipeline script as below.**

|  |
| --- |
| node {  // Mark the code checkout 'stage'....  stage 'Checkout'  // Checkout code from repository  checkout scm  // Get the maven tool.  // \*\* NOTE: This 'M3' maven tool must be configured  // \*\* in the global tool configuration in Jenkins.  def mvnHome = tool 'maven-3.5.3'  // Mark the code build 'stage'....  stage 'Build'  // Run the maven build - if you are running this script on Linux box, then replace 'bat' with 'sh' i.e., sh "${mvnHome}/bin/mvn clean install"  bat "${mvnHome}/bin/mvn clean install"  } |



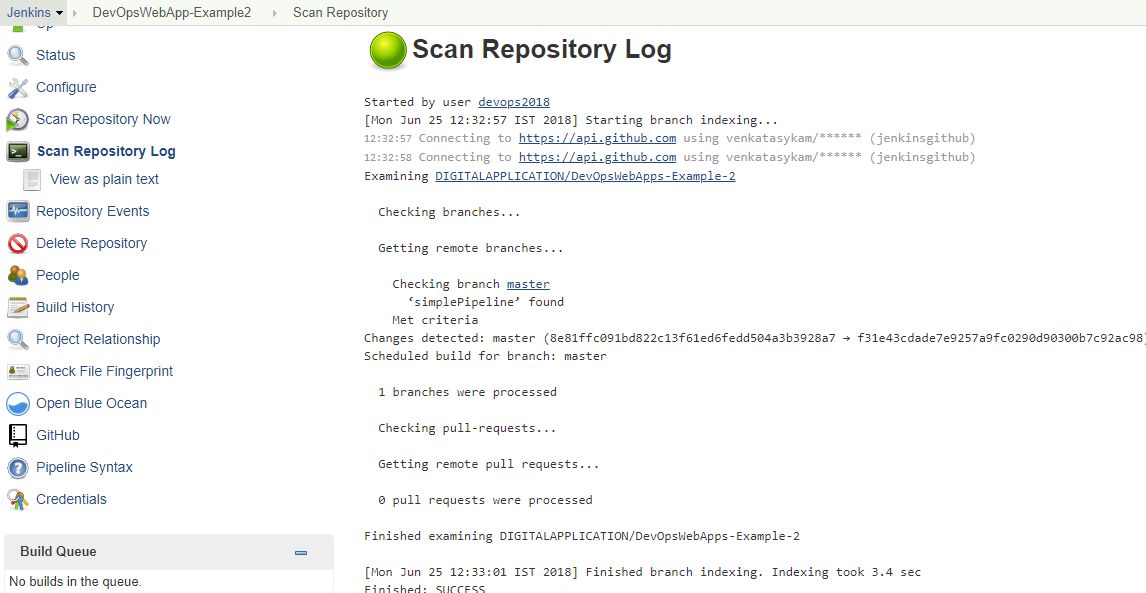
**Jenkins Job Configuration:**

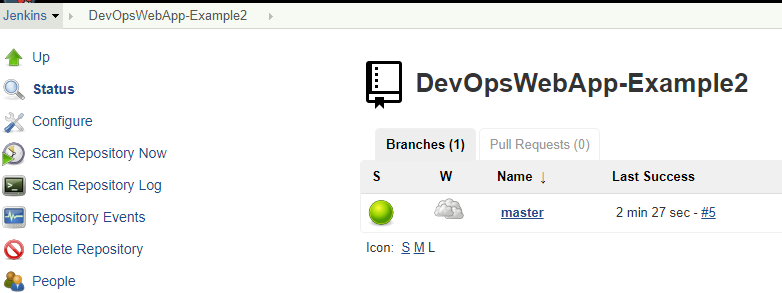


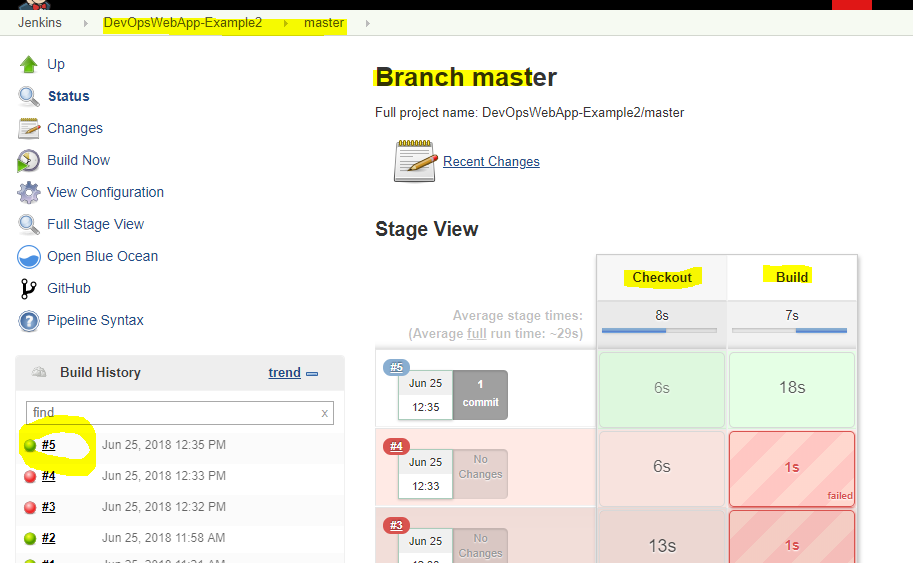


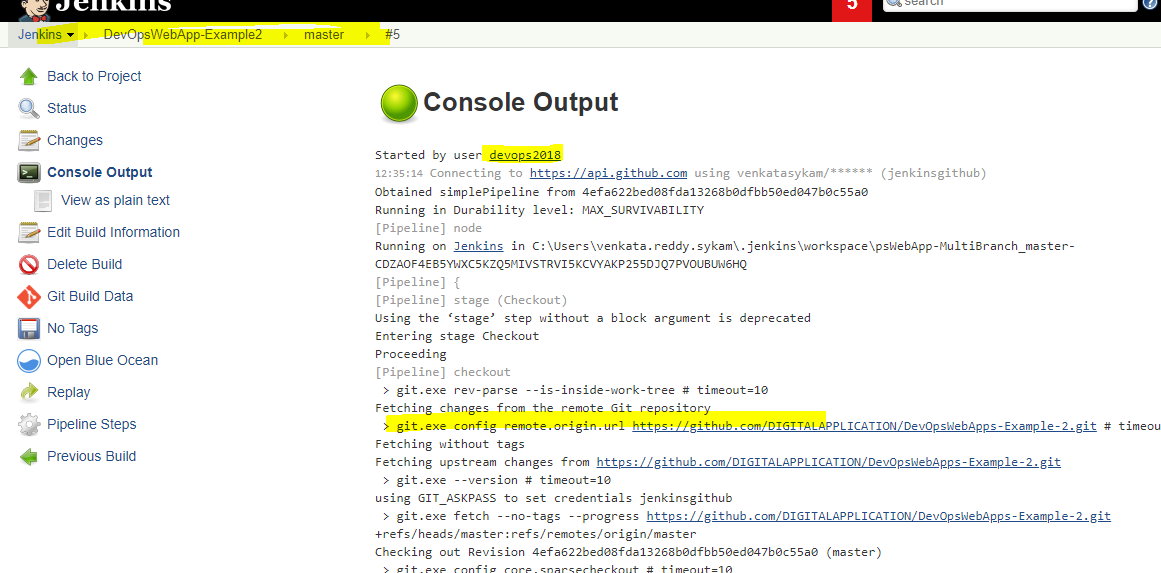


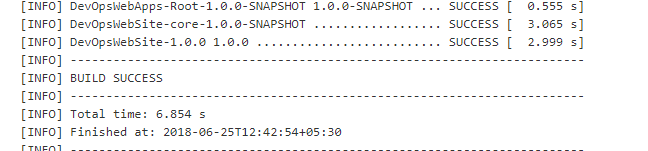
**Once we saved the job configuration, automatic scan will be started.**





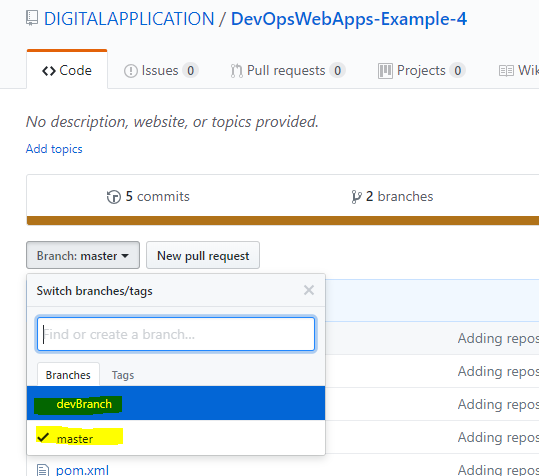




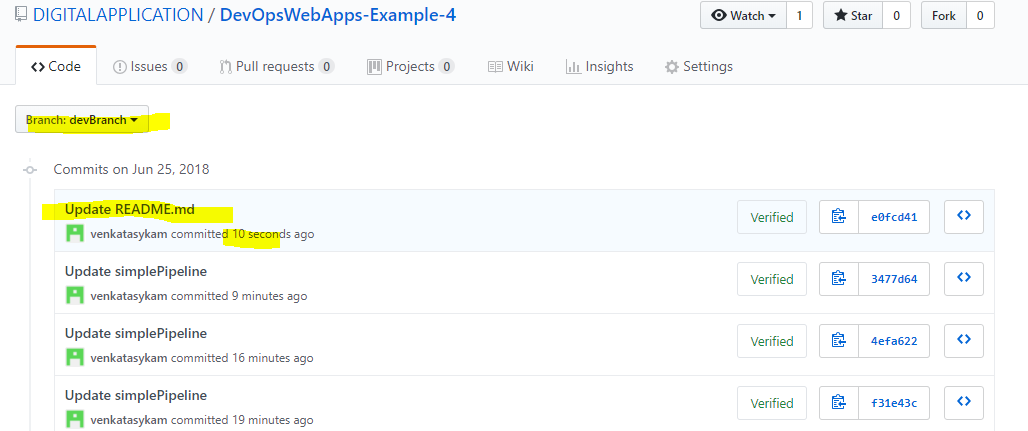


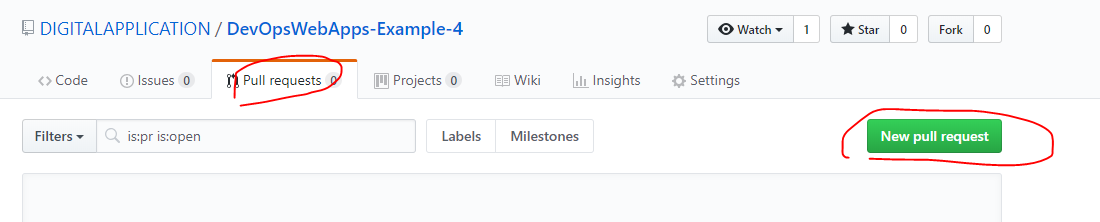
Another example for Pull Requests: <https://github.com/DIGITALAPPLICATION/DevOpsWebApps-Example-4>

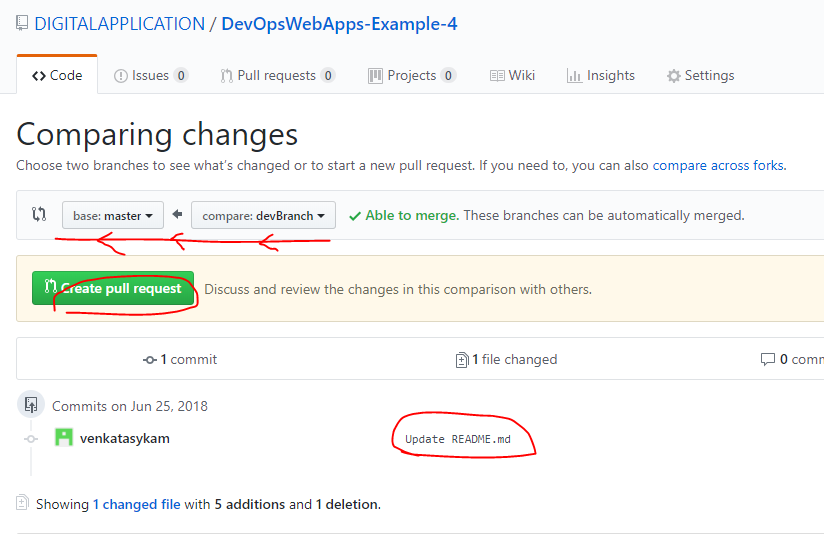
There are two branches: master, devBranch.

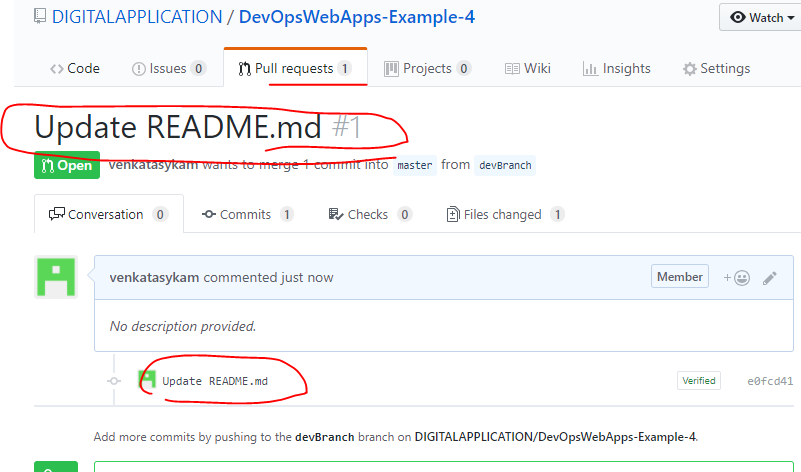


Committing something to “devBranch”. As per the below commit history, made some changes in **devBranch**. Now, I want to push this change to the branch “**master**”. For that, I am going to creating a Pull Request(PR) for master branch.

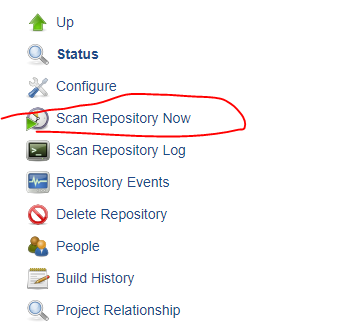


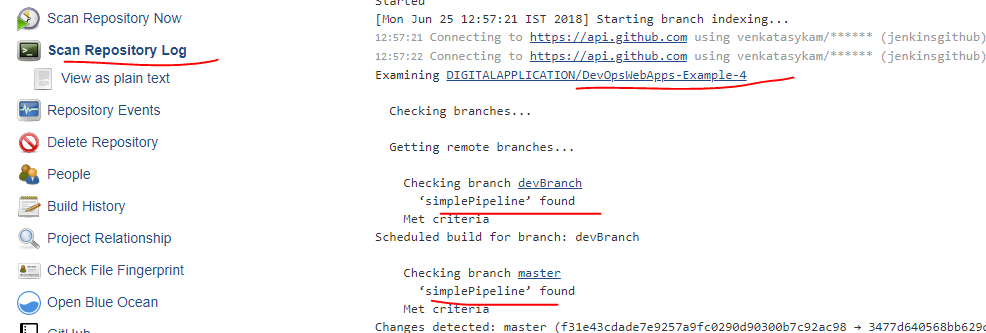


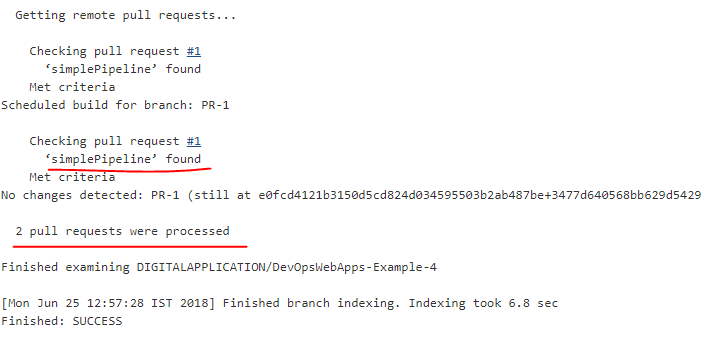


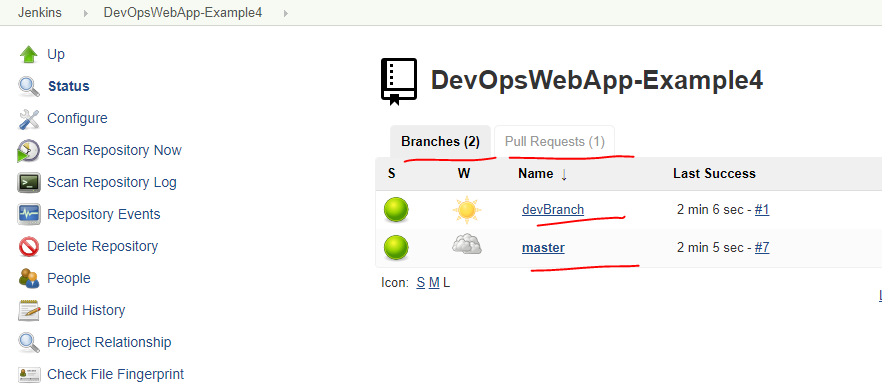


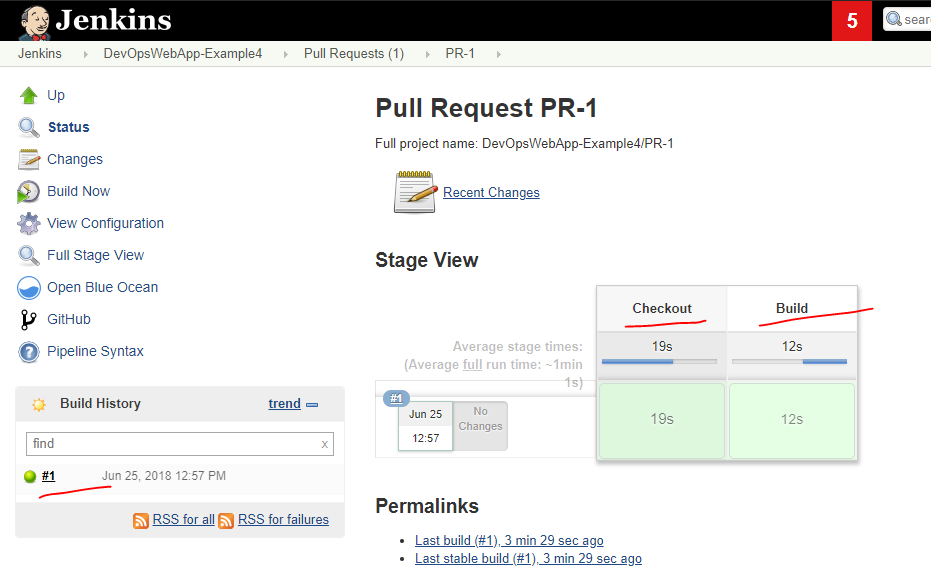
Now, scan the Jenkins pipeline.





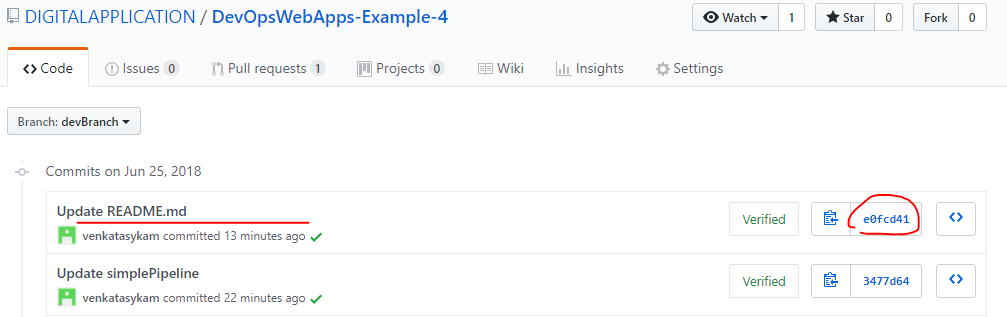




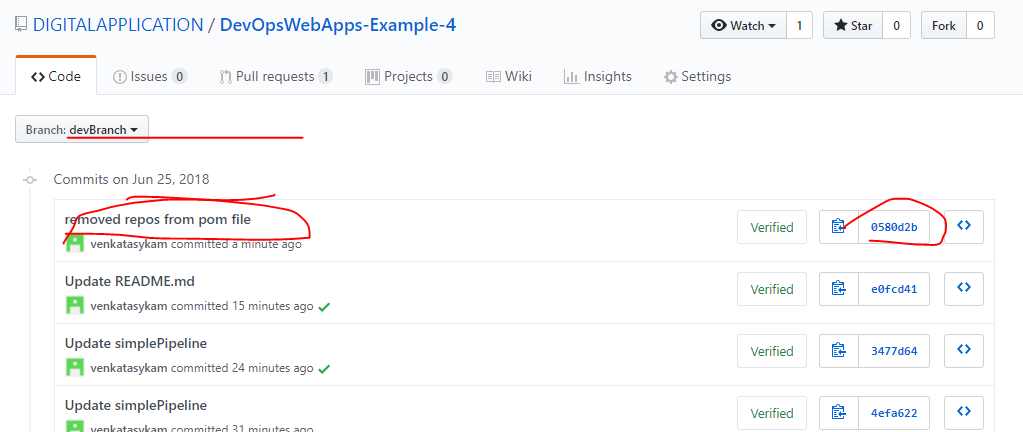


Build is success.

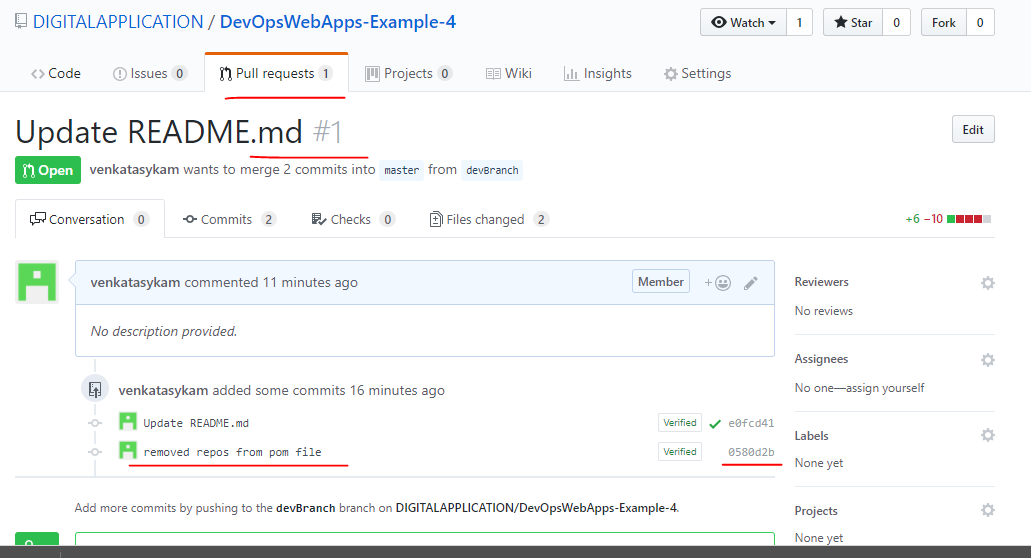




Made some changes in “**devBranch**” again.



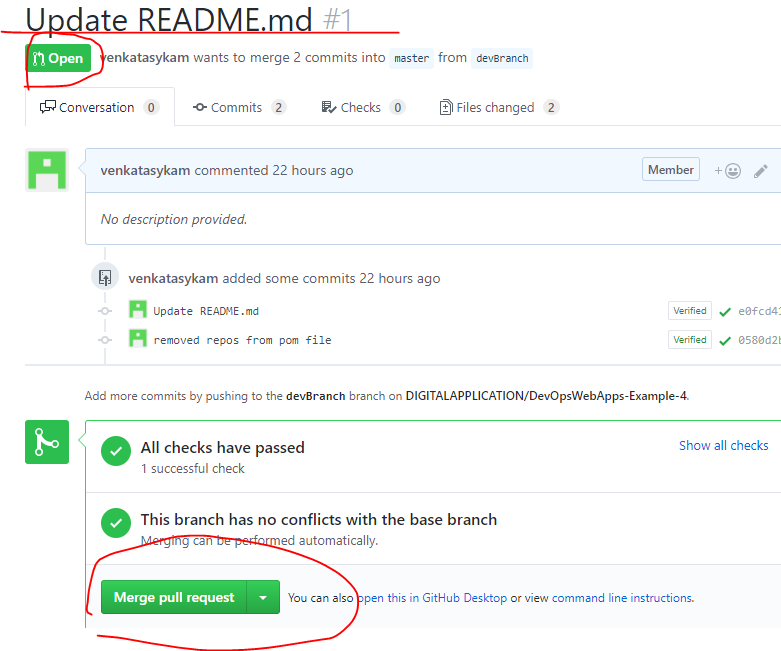
Open the PR#1 and see. Until the PR opens automatically the new changes will be added to the PR since it is the comparision b/w both the branches.

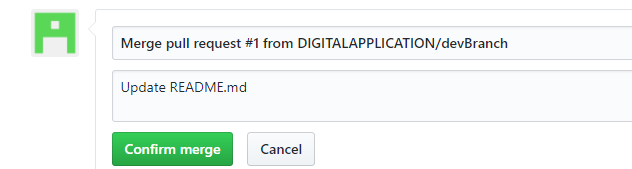


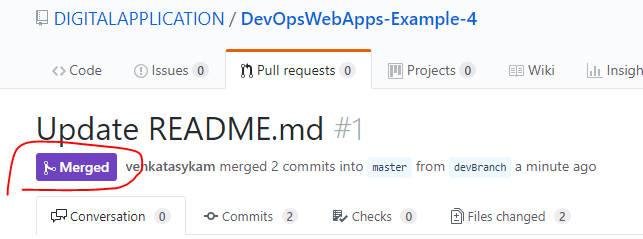
Scan again. This time only PR-1 job should trigger.

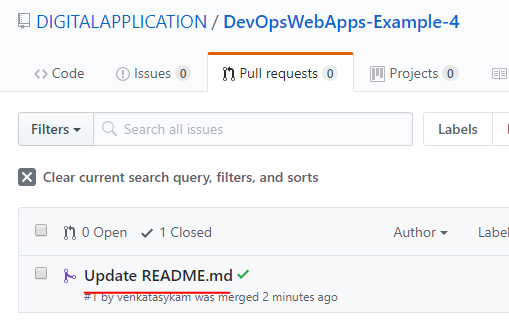


Close the PR: As of now the PR is in open state. Let’s close this PR by “Merge Pull Request”. Click on “Merge Pull Request” to merge the commits from “devBranch” to “master” branch.









Scan now.

