# International Institute of Information Technology, Bangalore


Academic Year 2019-20

**Project Report on Devops Calculator Assignment**

## Submitted By

Shivani Shrivastava

MT2019104

GitHub : https://github.com/ShrivastavaShivani

DockerHub : https://hub.docker.com/u/shrivastavashivani

# Introduction

The project aims at building DevOps Continuous Integration and Continuous Deployment Pipeline. So, this project has a very simple source code for Calculator. Following tools used to build this project:

1. IntelliJ
2. AWS EC2
3. Github
4. Webhook
5. Jenkins
6. Docker
7. Docker Hub
8. Rundeck

# WorkFlow of Project

The project starts with writing source code in intelliJ. Develop a Maven project on Java Language. The code is then pushed to github from where jenkins pulls it using git plugin and builds the jar file using Maven Integration plugin. The jar file is then tested using JUnit plugin. Jenkins then uses Docker Build and Publish plugin to make docker images which are then pushed to Docker Hub. Jenkins then finally invokes Rundeck. Rundeck deletes all the previous images and containers related to the new container in which it's going to deploy new image and the it pulls docker image from Docker Hub and deploys the image on any other docker container (same in our case).

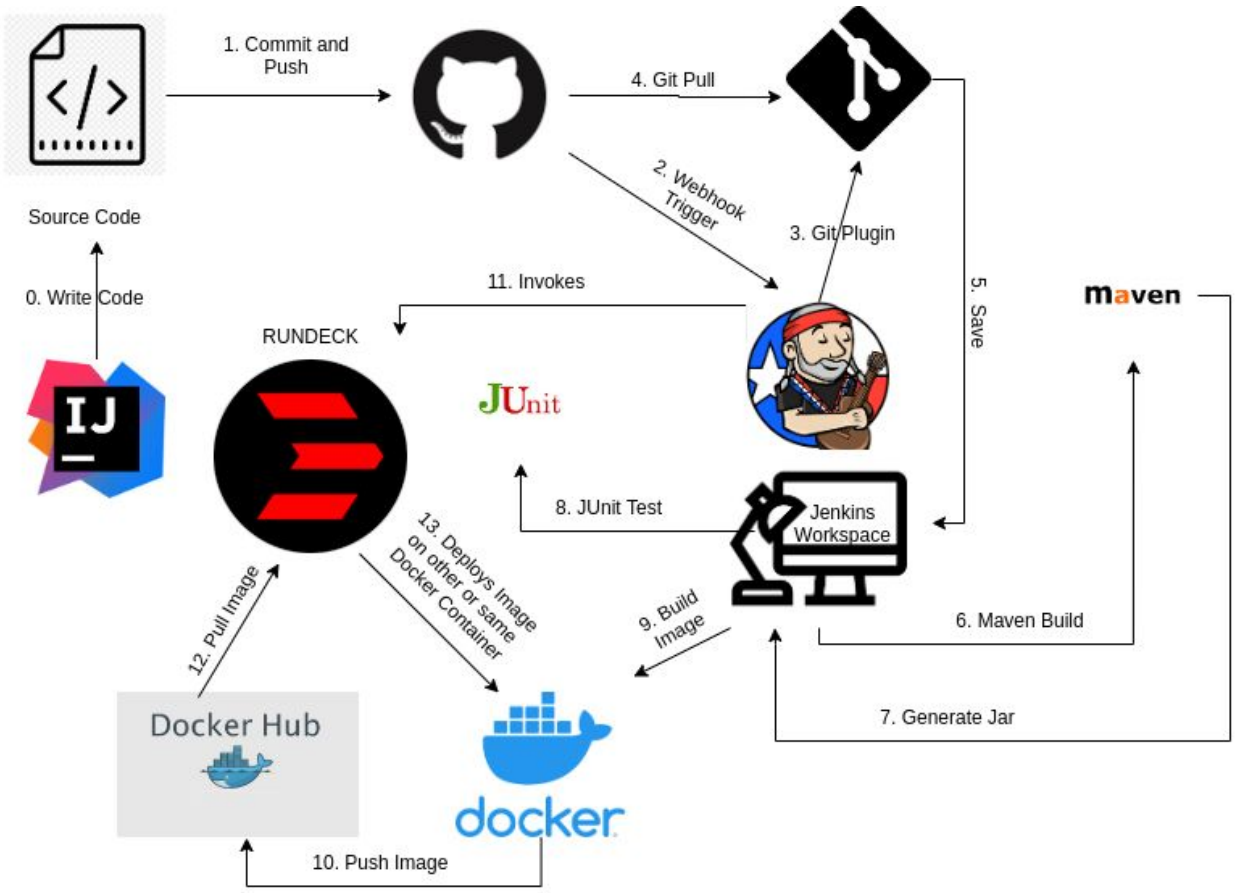


Fig 1. Workflow

# Source Code

The very first step in making a devops project is coding. We used IntelliJ IDEA for coding the calculator program in java. For testing utilities we made a MAVEN project. Install IntelliJ on Ubuntu 18.04 using following commands

```
$sudo snap install intellij-idea-ultimate --classic
```

Start a new maven project and select java version 8.

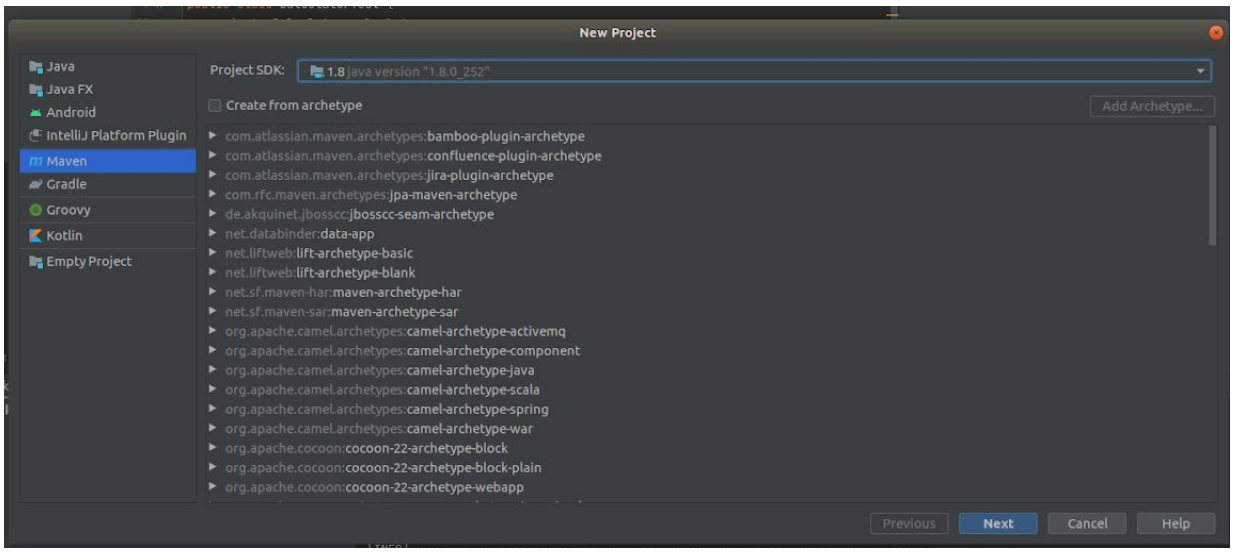


Fig 2. IntelliJ project config

Give your project a suitable name, location, group_id, artifact_id and version.

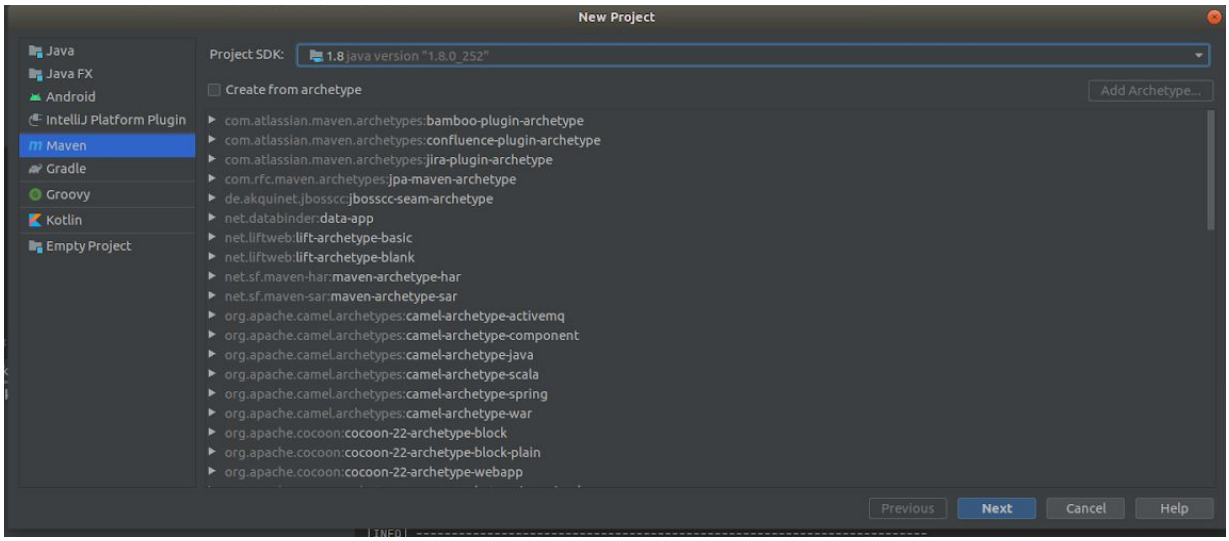


Fig 3. IntelliJ project config

Create two new java classes, Calculator.java and CalculatorTest.java and write code inside it. Remember to write code only for addition operation, as the rest of the 3 will be pushed after building the pipeline as a part of Continuous Integration. The final source code is as follows:

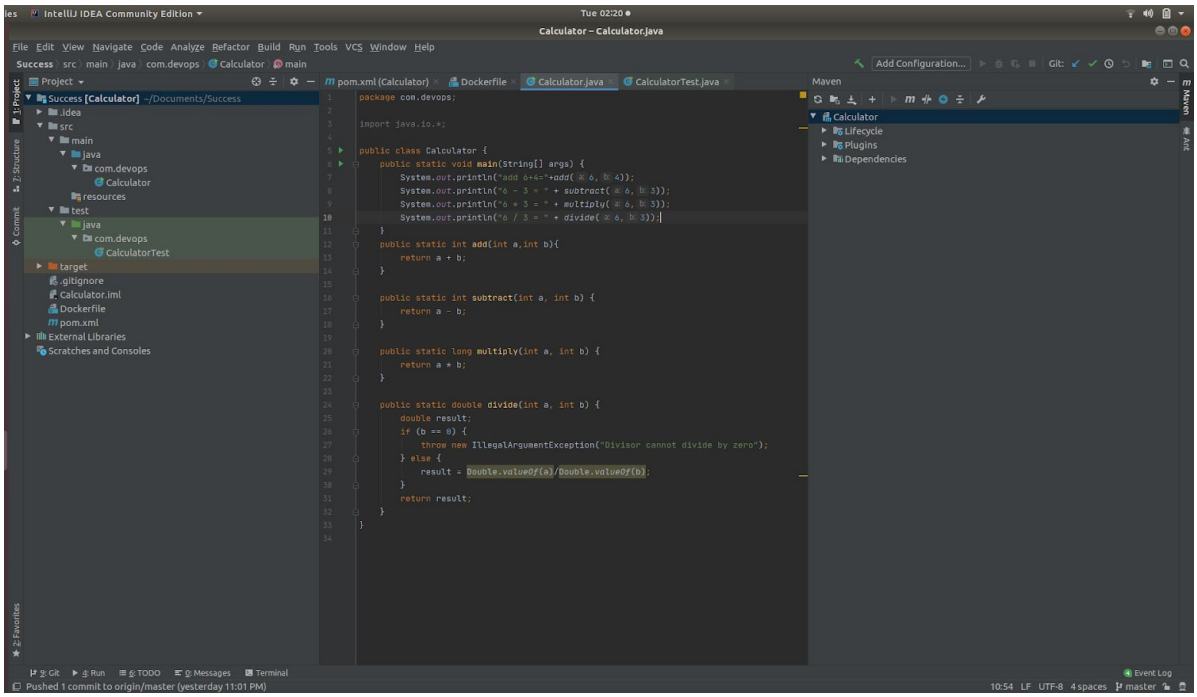


Fig 4. Calculator.java

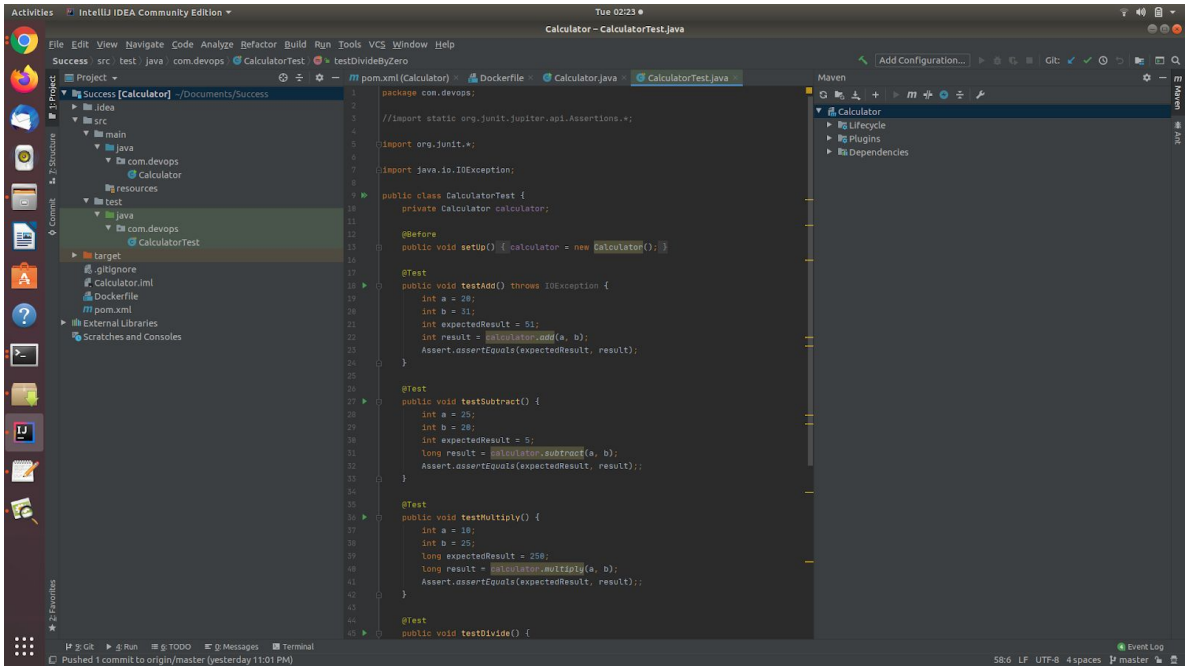


Fig 5. Calculator.java

The next job is to look at the services we will need in our project and write the correct pom.xml file. This is the which contains all the dependencies we require to build our project. The very basic need of our project is to have JUnit and maven dependencies included in pom.xml. Another important file is Dockerfile which Jenkins will need during Docker Build and Publish step. Jenkins will read Dockerfile to build and push docker image of our project to Docker Hub.

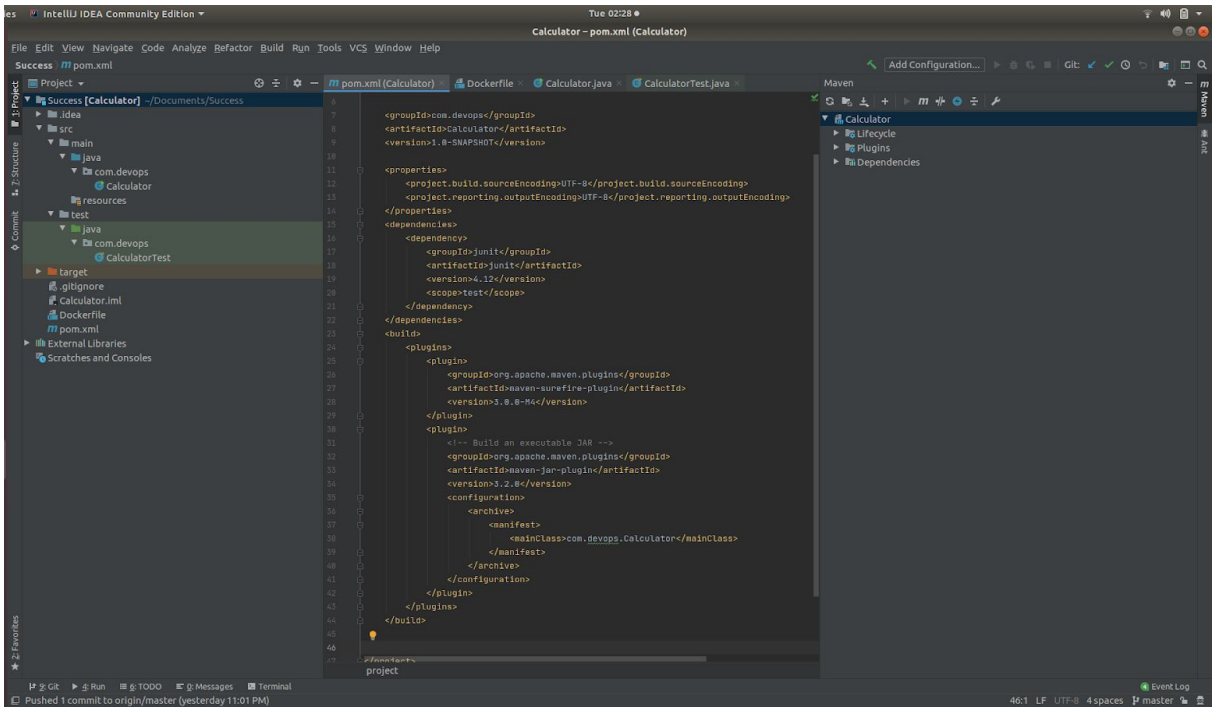


Fig 6. pom.xml

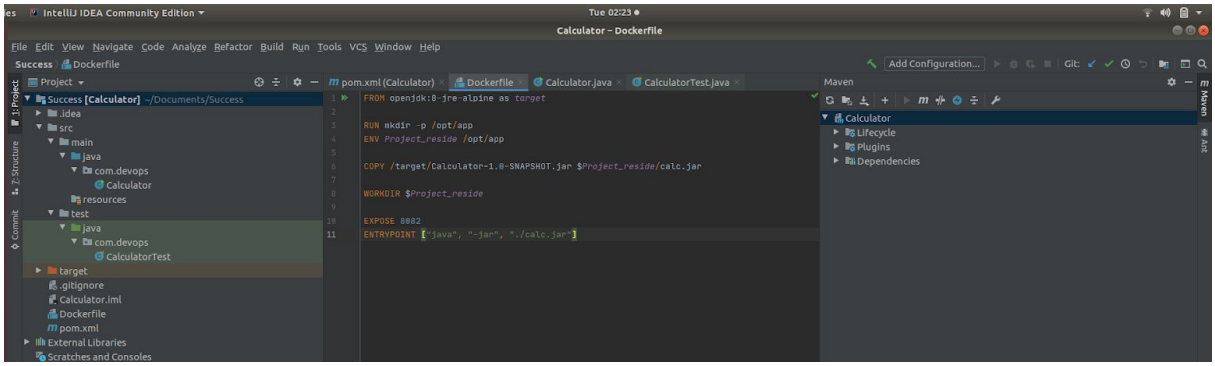


Fig 7. Dockerfile

# AWS EC2 Instance

To make the pipeline reachable to the outer world, I decided to build the devops pipeline through aws (Amazon Web Services). As this project is just to learn DevOps tools the ec2 instance is taken in free tier. By following the steps given below setup two ec2 instances.

1. Make an aws account. Some services are free for a duration of one year. (Do not worry about the credit or debit card details as after the full usage of the free tier they do not charge without notifying.)
2. Log in to your aws account, hover on services and click on EC2 under compute services.
3. Launch a new instance and follow as shown in pictures below.

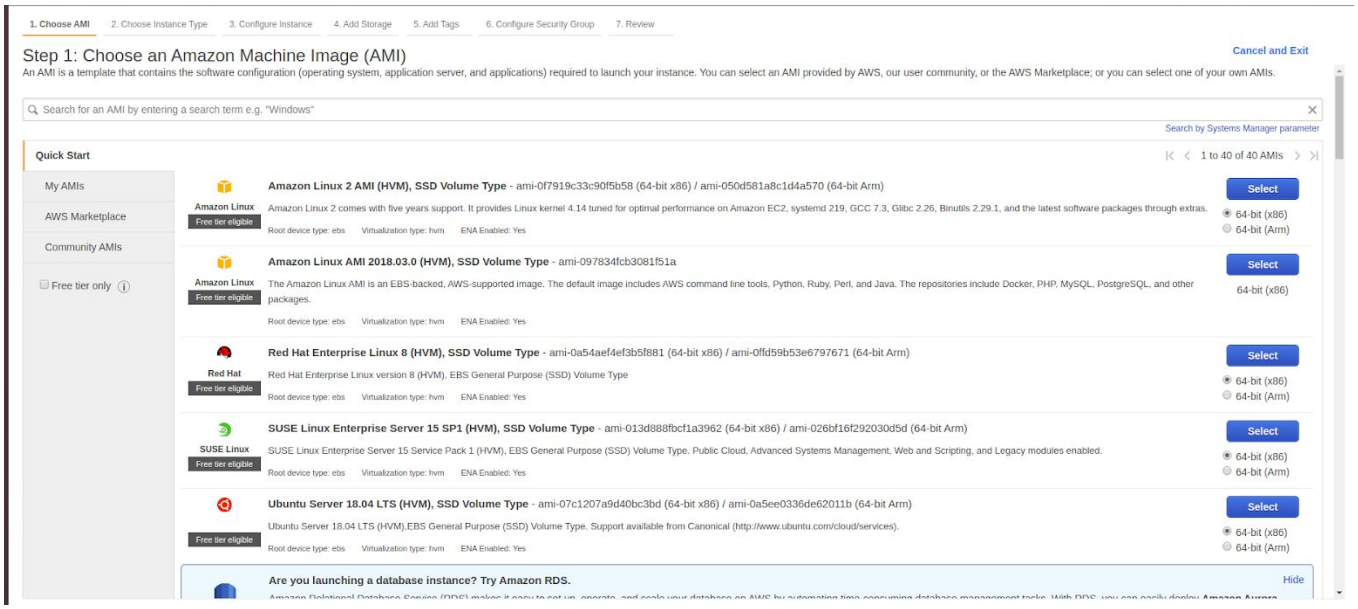


Fig 8. AWS EC2 configuration

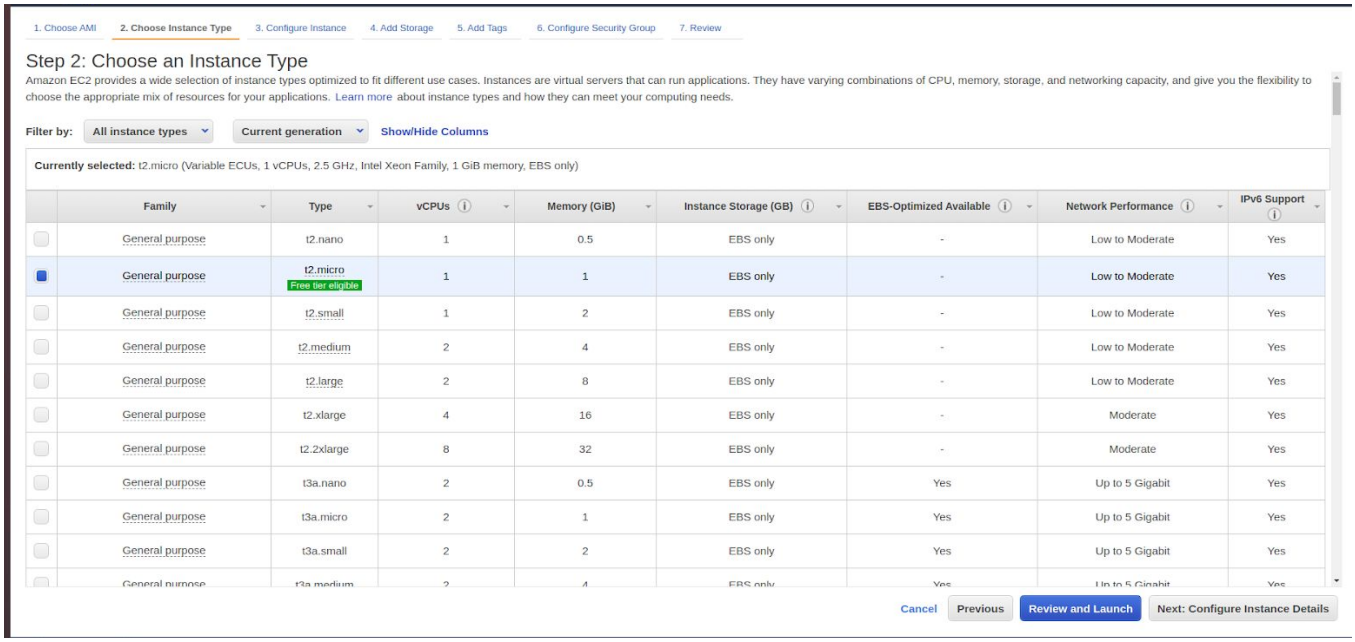


Fig 9. Select Instance Type

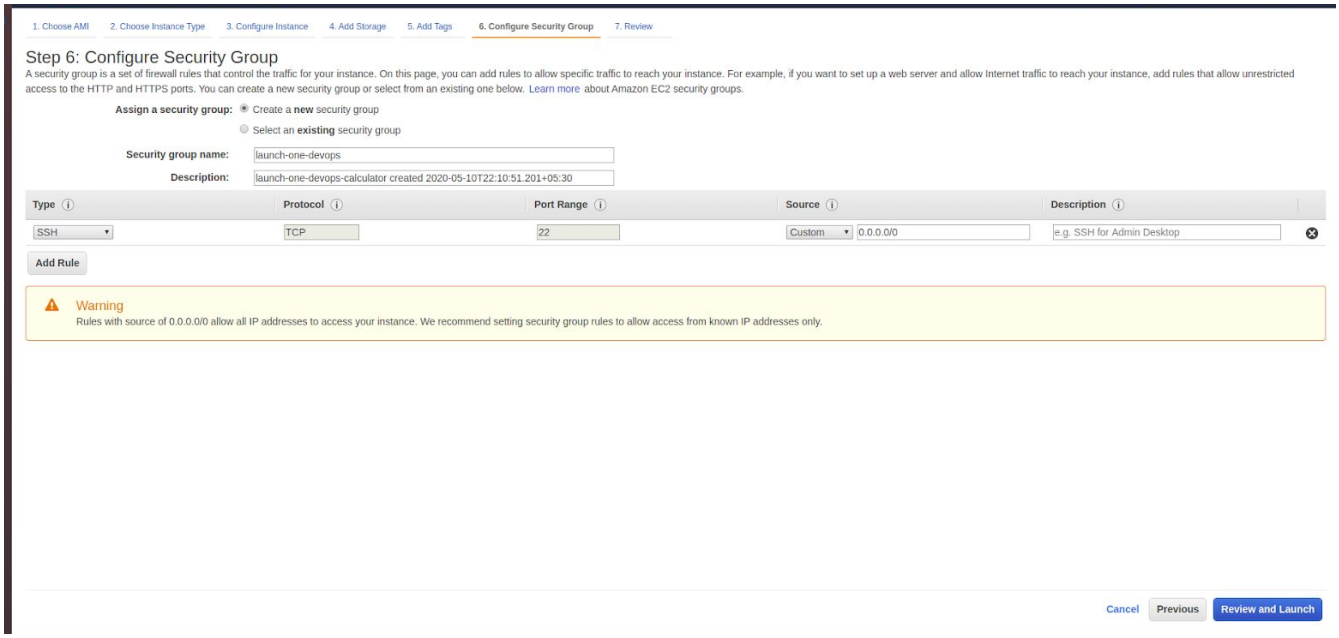


Fig 10. Create a new Network Security Group

Add inbound rules for the ports into the security groups which will be accessed from this instance for building pipeline. This EC2 instance is for Jenkins and Docker working.

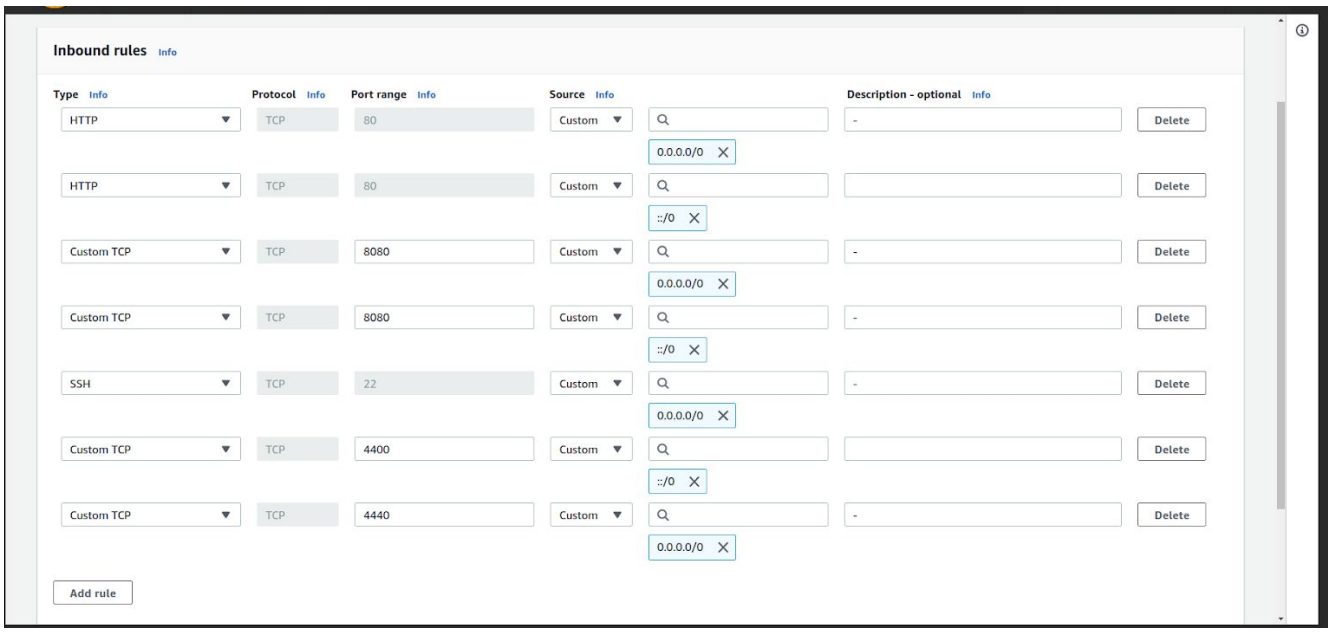


Fig 11. Add inbound rules for NSG

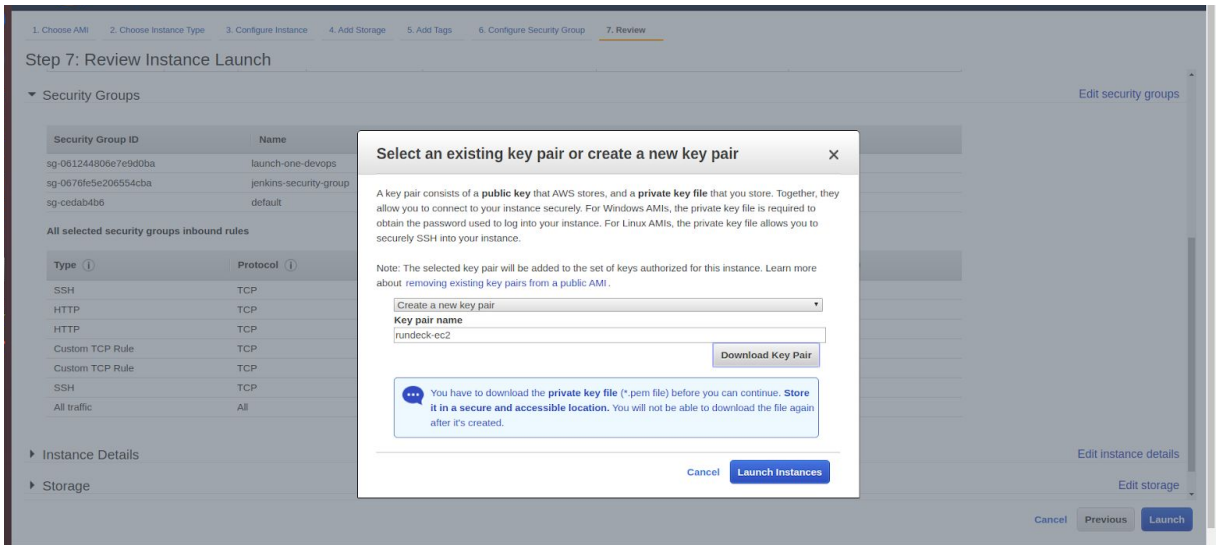


Fig 12. Download key pair file

Do not forget to download the key pair file before launching the instance.

Perform the same steps to create another EC2 instance which will be for Rundeck but this time no need to create a new security group just select it.

## SSH Connection

To connect to aws EC2 instance from local machine follow the given steps:

1. Open an SSH client i.e. terminal inside the folder where the key-pair rundeck-ec2.pem file is present.
2. The wizard automatically detects the key used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command `$chmod 400 rundeck-ec2.pem`
4. Connect to your instance using its Public DNS: (xyz is DNS Public IPv4) ec2-xyz.us-east-2.compute.amazonaws.com with the command

```
$ssh -i "rundeck-ec2.pem"
ec2-user@ec2-xyz.us-east-2.compute.amazonaws.com
```

## Github

Go to https://github.com and sign up in github. Go to intelliJ and allow VCS - Version Control Services and add github credentials in it. After that, Commit and Push the whole intelliJ project on Github. You might want to remove an error inside jenkins regarding Github plugin at that time you might need to install git in EC2 instance. You can do that using following command

```
$sudo yum install git
```

## Docker

Install docker using following commands:

```
$sudo yum update -y //updates system
$sudo amazon-linux-extras install docker //install docker
$sudo service docker start //starts docker
$sudo usermod -a -G docker ec2-user //adds ec2-user in docker group
```

After adding ec2-user to docker make sure you reboot your instance.

## Docker Hub

Go to https://hub.docker.com/ and make an account in Docker Hub. Do not create any repository manually. Just create an account and note down its credentials.

## Jenkins

Install jenkins in ec2 instance using following commands:

```
$sudo yum update -y //update system
$sudo yum install java-1.8.0-openjdk-devel //install jdk
$curl --silent --location
http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee
/etc/yum.repos.d/jenkins.repo //get GPG key
$sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
//add repo to system
$sudo yum install jenkins //install jenkins
$sudo systemctl start jenkins //start jenkins
$systemctl status jenkins.service // check status
```

Now, our machine is ready with jenkins installed in it. Open jenkins in the web browser at http://xyz:8080. Port no. 8080 is the default port for jenkins.

Copy the initial admin password using following command to Unlock Jenkins:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Install initial suggested plugins and wait for the download to complete.

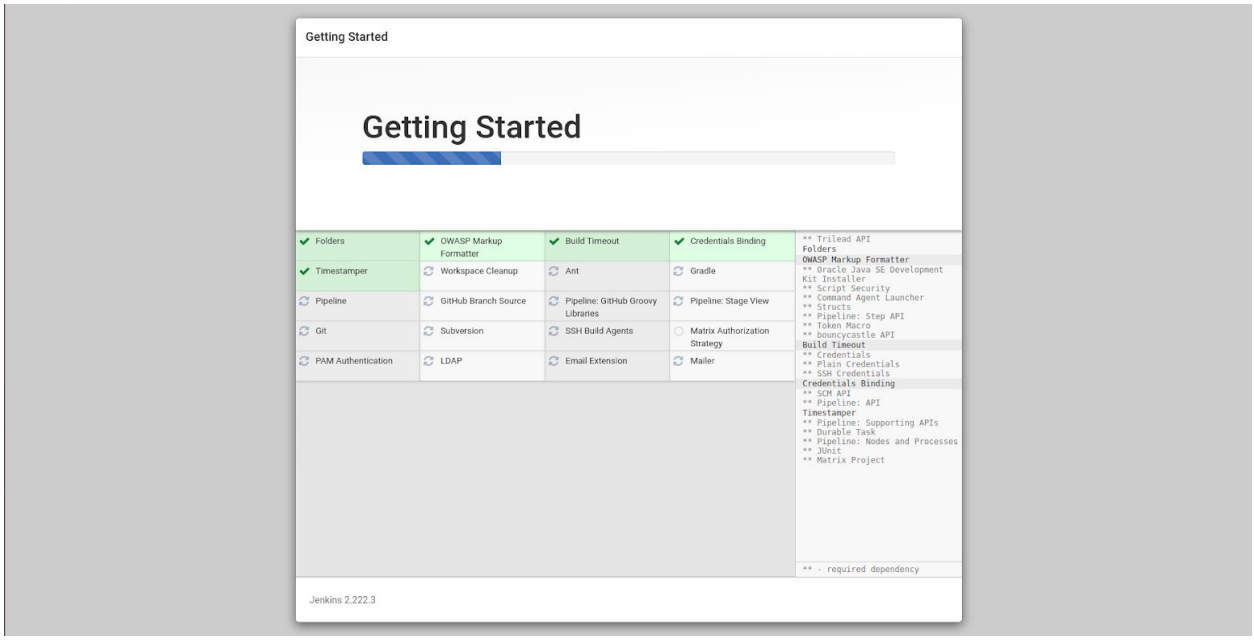


Fig 13. Getting Started with Jenkins

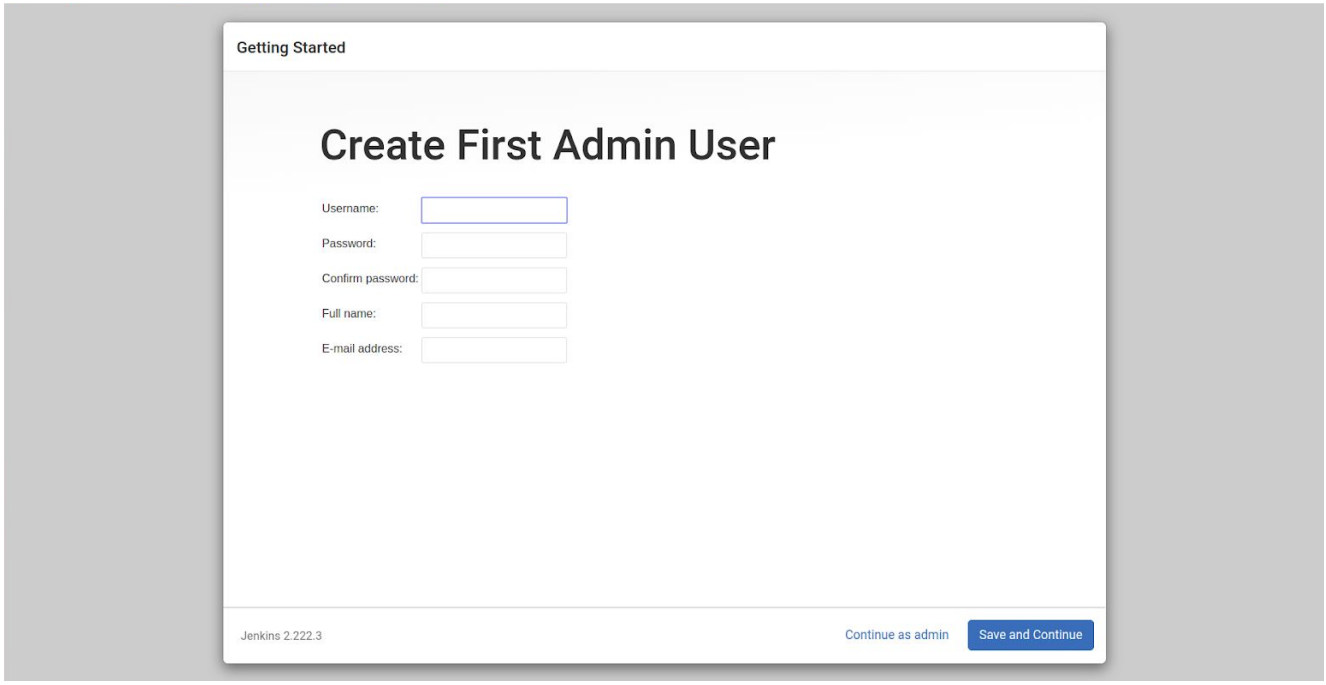


Fig 14. Jenkins User Account Setup

Save and Finish !

Your jenkins is now ready to use.

## Plugins

Next step is to install all the necessary plugins which we will need in our project. Click on Manage Jenkins and go to Manage Plugin. Under the Available plugin tab check the following plugins and install them.

1. CloudBees Docker Build and Publish plugin
2. Docker plugin
3. Docker-build-step
4. Git plugin
5. Github plugin
6. Junit plugin
7. Maven Integration plugin
8. Pipeline plugin
9. Pipeline : GitHub Groovy Libraries
10. Rundeck plugin

## Webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, Webhook sends a POST request to each of the URLs you provide.

For setting up Webhook follow given steps:

1. Login to Github
2. Go inside the repository you want to give access to aws to pull files from.
3. Click on the Settings tab and then go to the Webhook section.
4. Add webhook and enter the url where jenkins will be hosted i.e. http://xyz:8080 (remember xyz is Public DNS IPv4)
5. As soon as you add the webhook, github will make a dummy post call to the jenkins webhook. Make sure the call is successful to the jenkins. It can be verified from the status icon on left of added webhook which should be a green tick.

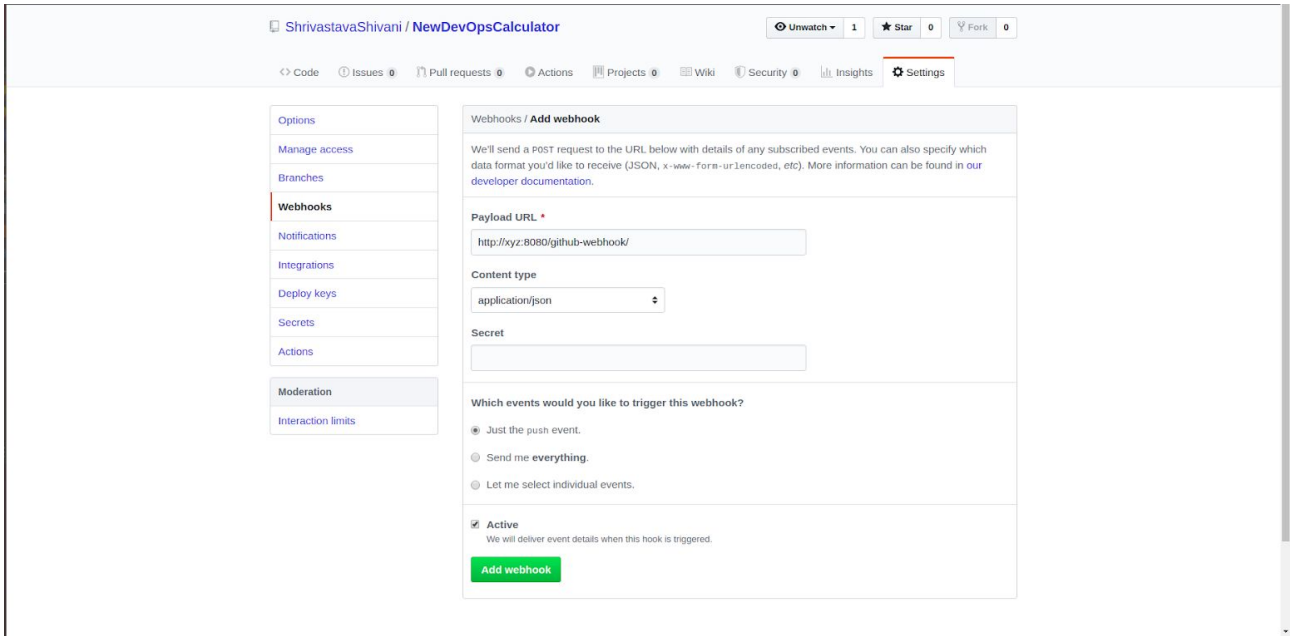


Fig15. Setting up Webhook

# Creating Items

Create 3 new items in Jenkins -

1. Calculator-1-Build
2. Calculator-2-Test
3. Calculator-3-Main

Configure each one of them as shown in figures below

For configuring Calculator-1-Build project, you need to give access to your github account. Add correct username and password and proceed. In the Build step below, you might need to add Maven for Invoking top level maven targets to build jar file from the dockerfile commands. This can be done by going to Manage Jenkins then into Global Tool Configuration and then in Maven section click on Add Maven. Give a name to your maven version (mine - Maven_latest) and select one version out of the drop down bar. Click on save and you are done!

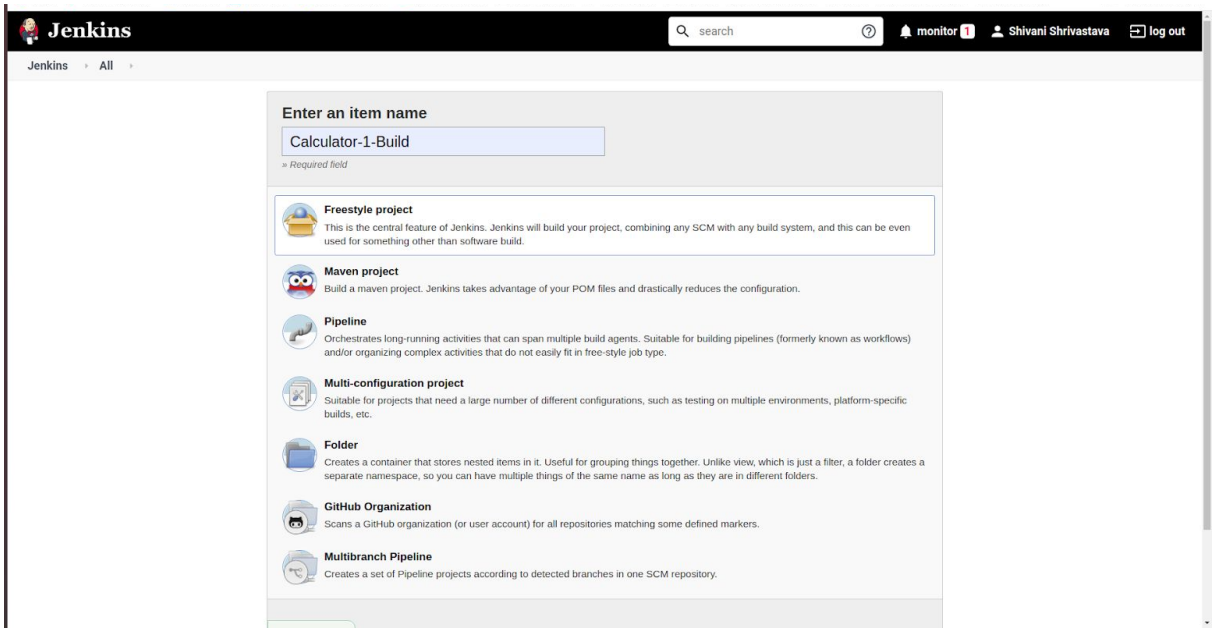


Fig16. Calculator-1-Build setup-1

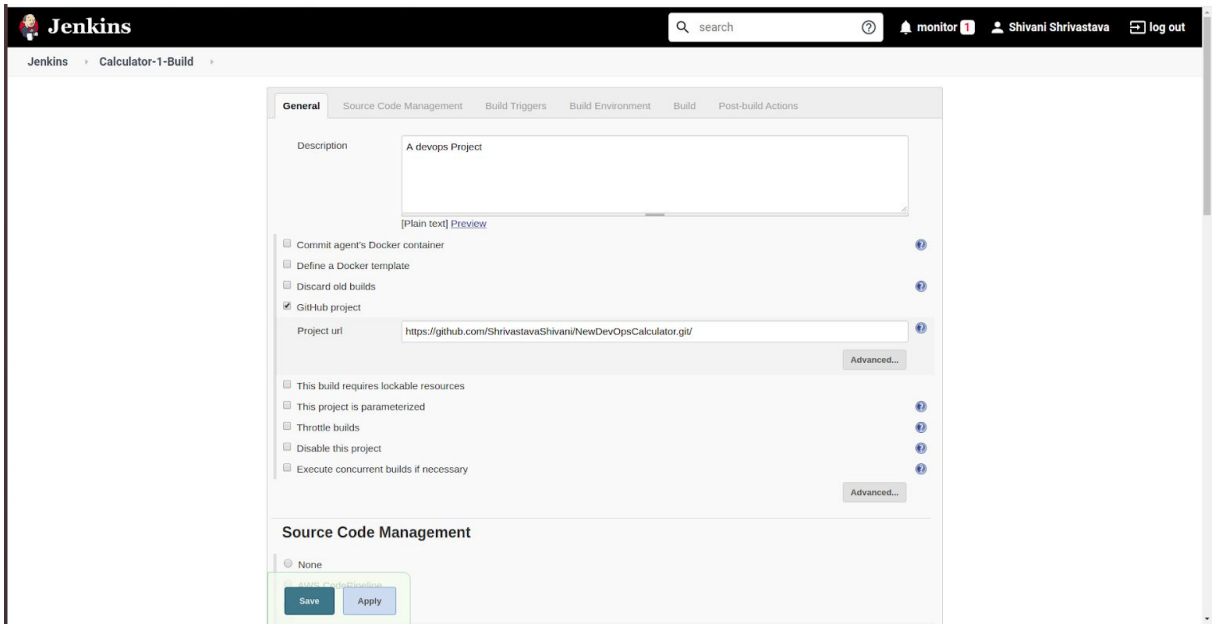


Fig17. Calculator-1-Build setup-2

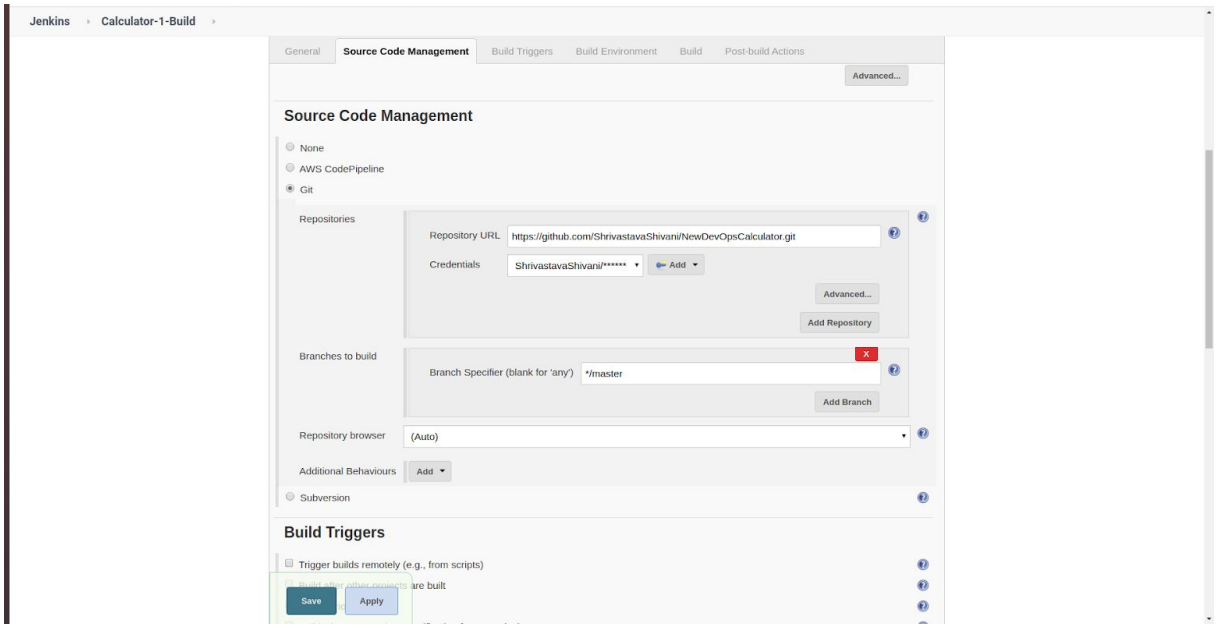


Fig18. Calculator-1-Build setup-3

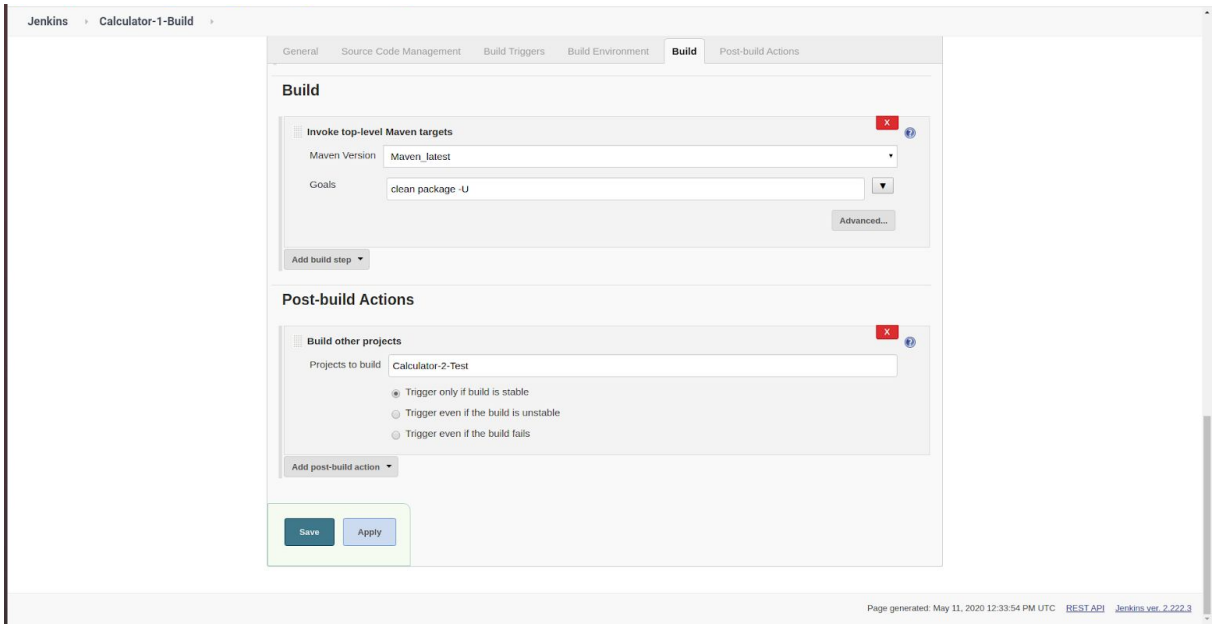


Fig19. Calculator-1-Build setup-4

Click on Apply and then Save. To check individually, you can now click on Build Now and see if Console Output is Success.

Now configure the Calculator-2-Test project as shown in figures.

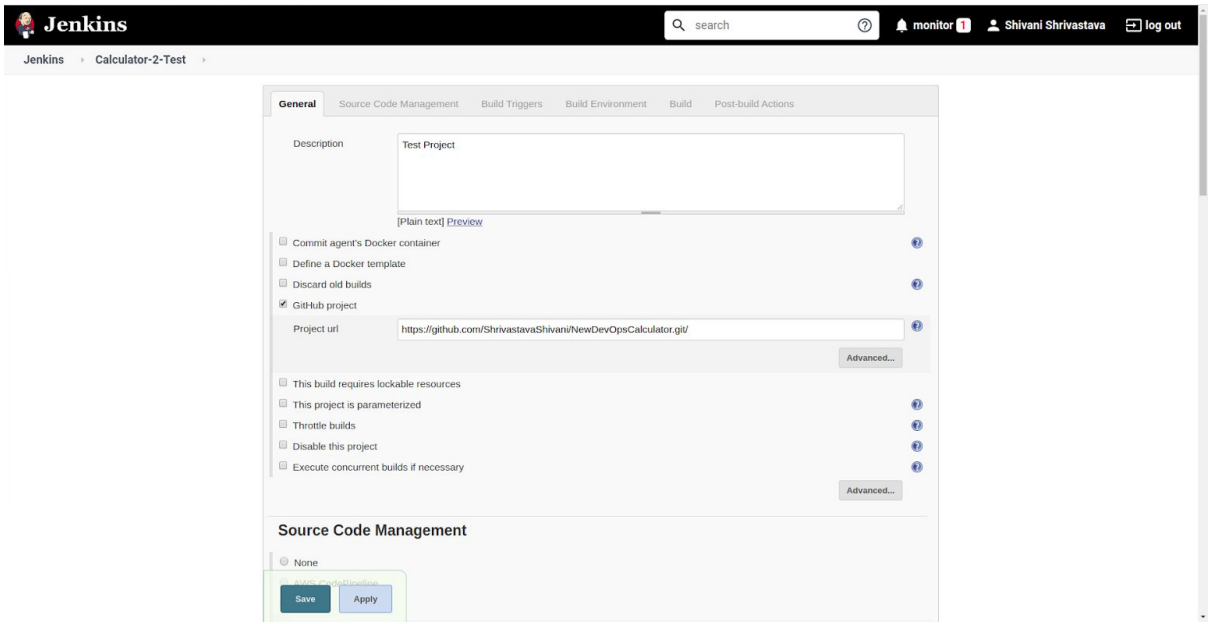


Fig20. Calculator-2-Test setup-1

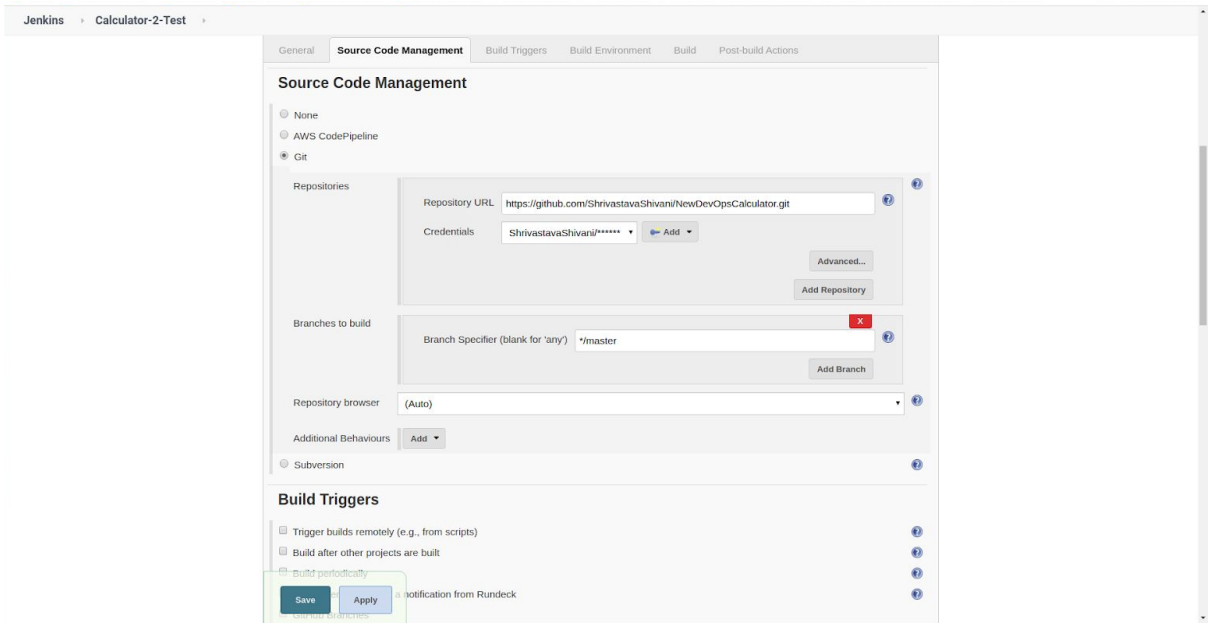


Fig21. Calculator-2-Test setup-2

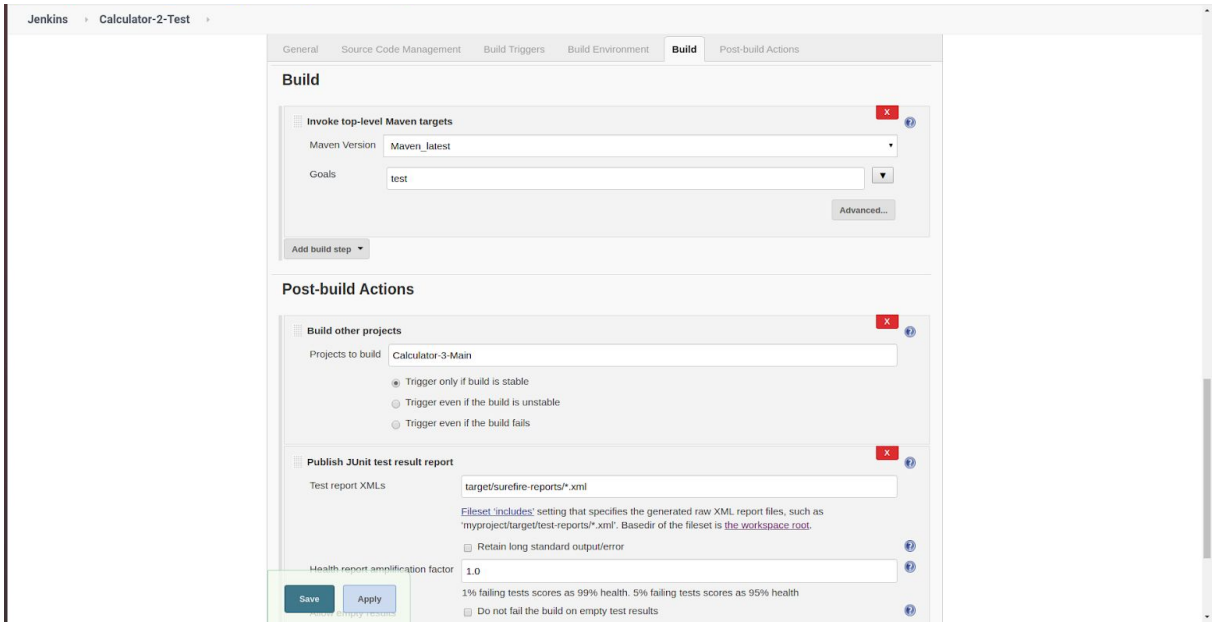


Fig22. Calculator-2-Test setup-3

Apply and Save. Build using build now and check if it's giving success.

Let us now configure the third Calculator-3-Main project

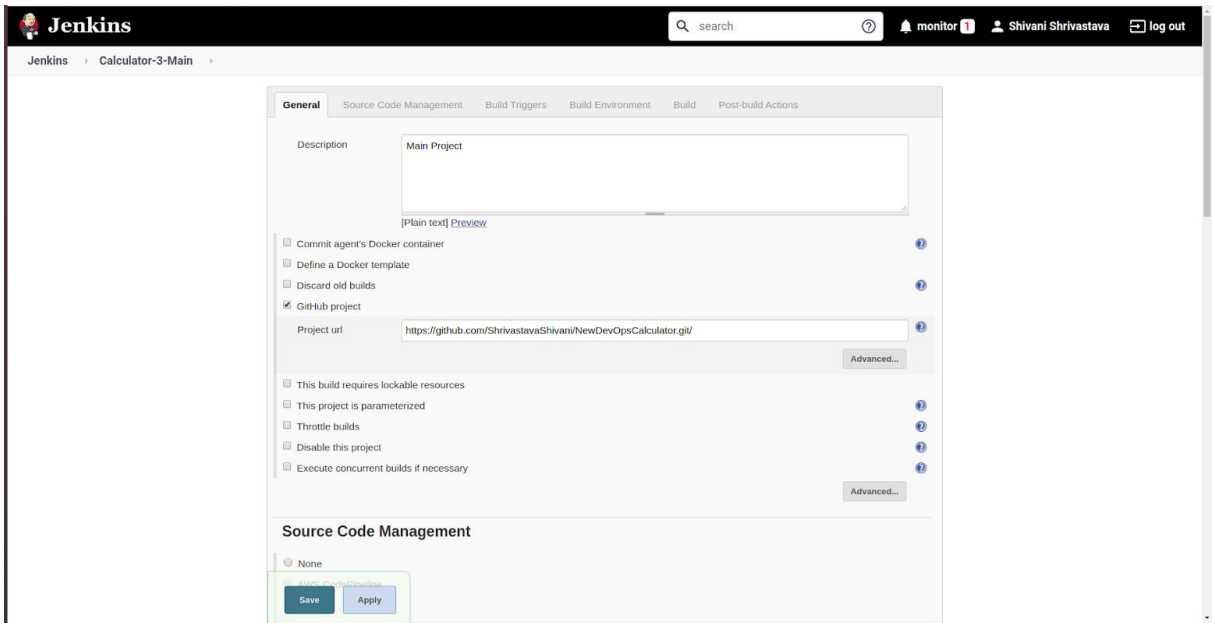


Fig23. Calculator-3-Main setup-1

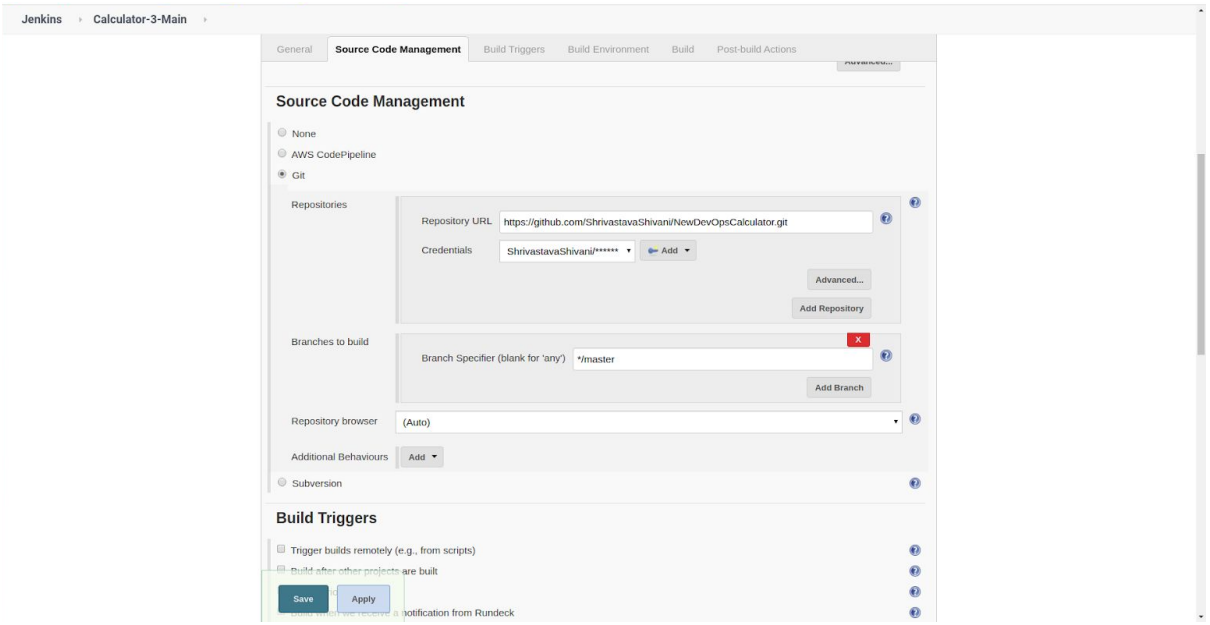


Fig24. Calculator-3-Main setup-2

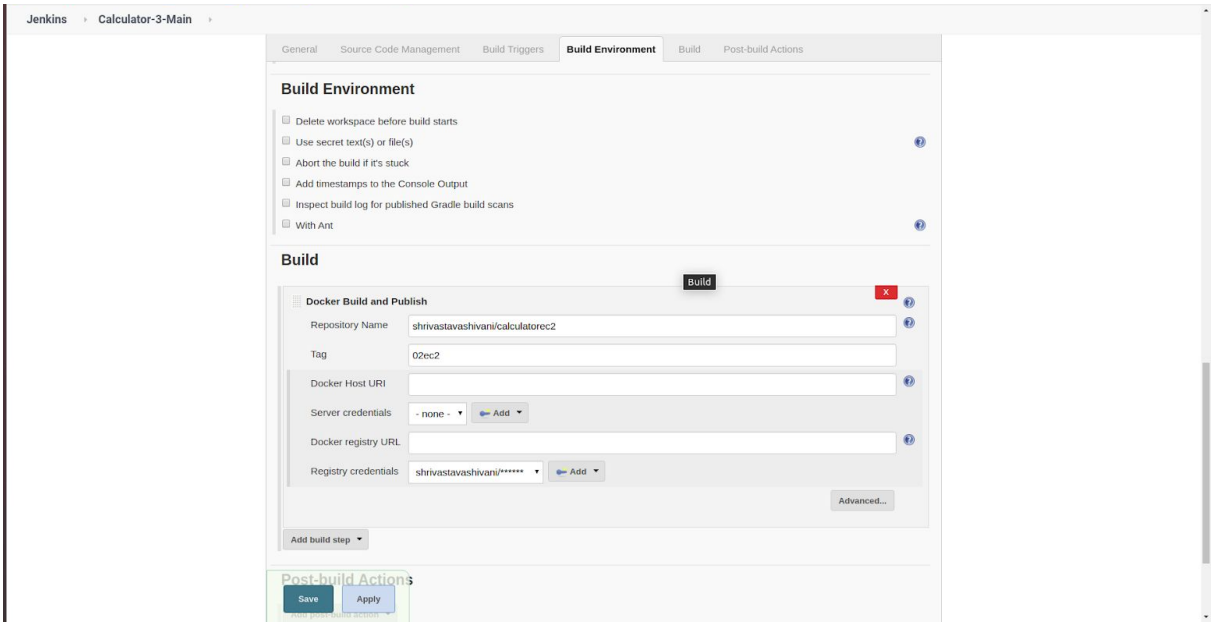


Fig25. Calculator-3-Main setup-3

In Docker Build and Publish, Repository Name should be your docker hub username followed by the name you want to build a repository for. Also, new credentials need to be added here which will contain username and password of docker hub.

Before adding Post-Build Actions, you must install and configure rundeck and add a job as shown after this section. This will assign a Job identifier to the rundeck job which will be provided in the Job Identifier. Also, Rundeck instance need to be added in Manage Jenkins before this step.

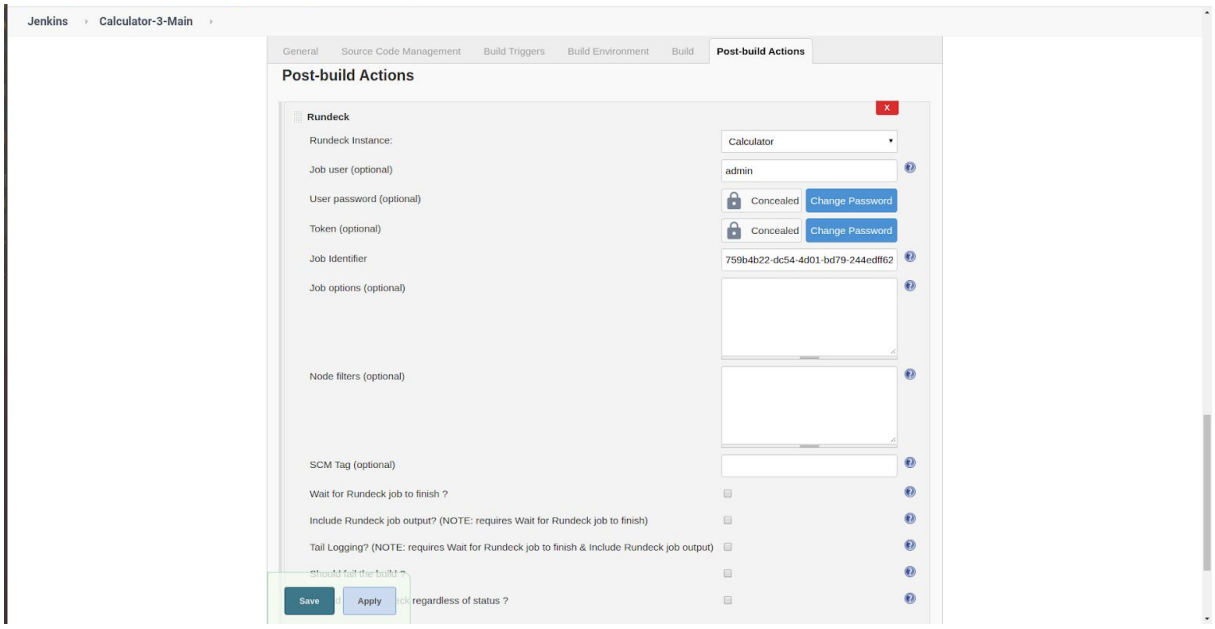


Fig26. Calculator-3-Main setup-4

# Rundeck

Use another ec2 instance rundeck-ec2 for installing Rundeck. Login in terminal with ssh command for the new rundeck ec-2 instance. Install rundeck in this instance using following commands:

```
$Sudo yum update
$sudo rpm -Uvh http://repo.rundeck.org/latest.rpm
$sudo yum install rundeck java
$Sudo yum update rundeck
$Start rundeck
$sudo service rundeckd start
```

Now after installation, check if the Java version is 11.0.7. If yes, then install Java with version 1.8.0 using following command as rundeck runs on Java8 support.

```
$sudo yum install java-1.8.0-openjdk
```

Switch between java versions using following command:

```
$sudo update-alternatives --config java
```

Reboot system.

Before login to Rundeck dashboard you have to update public dns name at parameter name “grails.serverURL” in the configuration file /etc/rundeck/rundeck-config.properties. For doing so use command:

```
$sudo nano /etc/rundeck/rundeck-config.properties
```

And then edit line -

```
grails.serverURL = http://ec2-xyz.compute-1.amazonaws.com:4440
```

After editing press Ctrl+S to save and Ctrl+X to exit.

Open Rundeck in http://ec2-xyz.compute-1.amazonaws.com:4440 url in web browser.

Enter Username : admin

Password : admin

Create new Project in Rundeck as shown below:

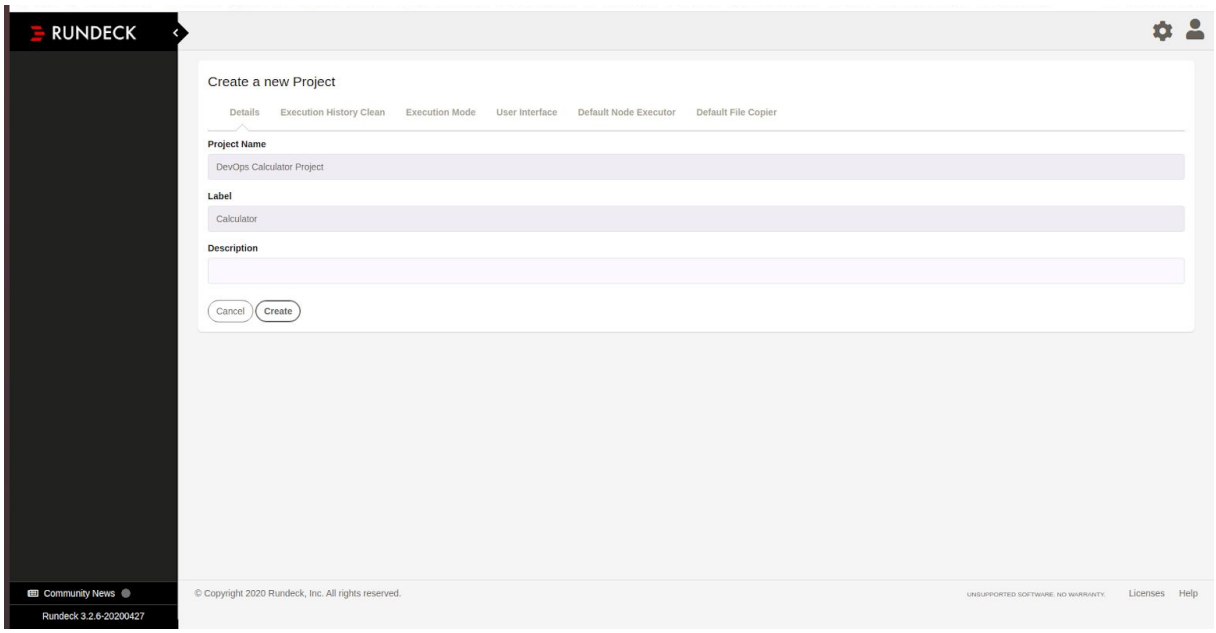


Fig 27. Rundeck Create New Project

Now, add a new Job to this rundeck project. This job will pull image from dockerhub and make run it in docker container for deployment.

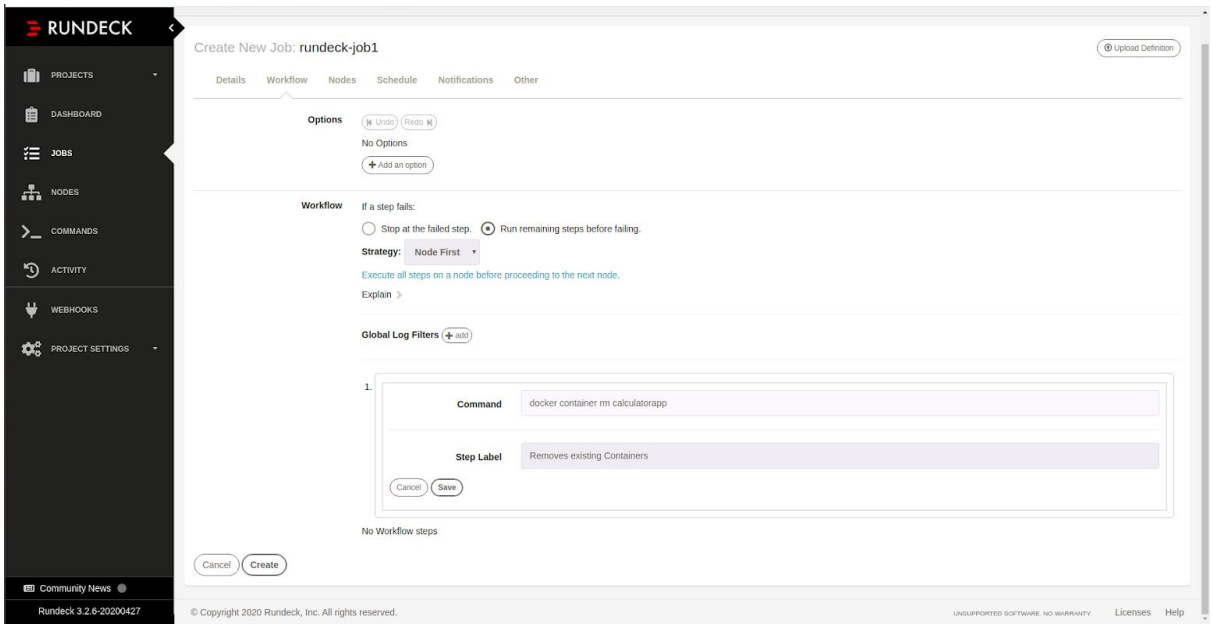


Fig28. Rundeck job setup-1

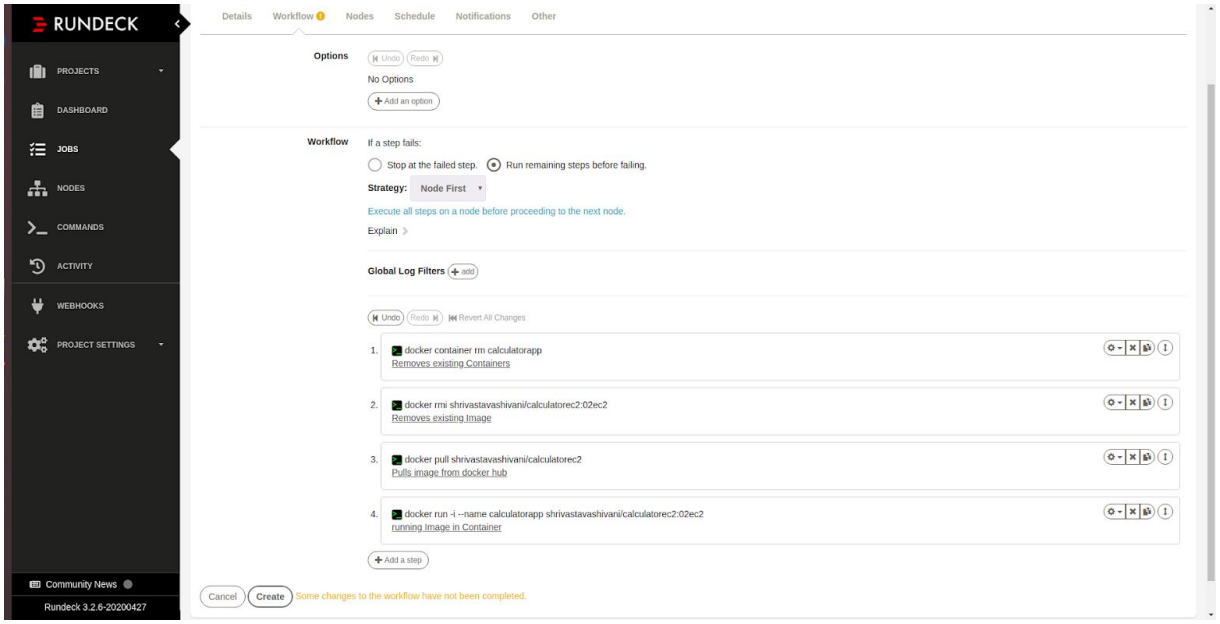


Fig29. Rundeck job setup 2

The first two commands in rundeck job are to remove any previous image of docker or container with the same name as the one rundeck will create after pulling image from docker hub. This is why it is important to check Run remaining steps before failing.

# Adding Rundeck Instance in Jenkins

We will need to add a rundeck instance in jenkins in order to configure post build actions of item 3 Calculator-3-Main. Go to Manage Jenkins and then in Configure System and add Rundeck in Rundeck section as shown below:

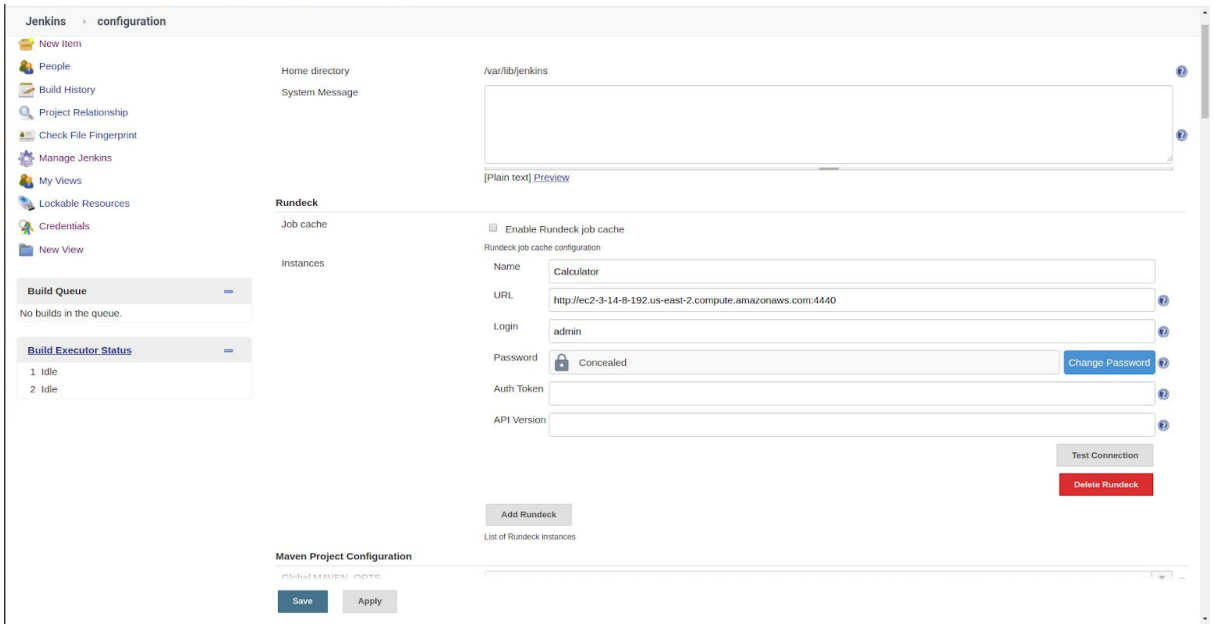


Fig 30. Rundeck instance in Jenkins

# Build Pipeline

Install Build Pipeline plugin from Manage Jenkins -> Manage Plugins. After this on the dashboard, click on '+' for creating pipeline view. Add view name and select Build Pipeline. Click OK.

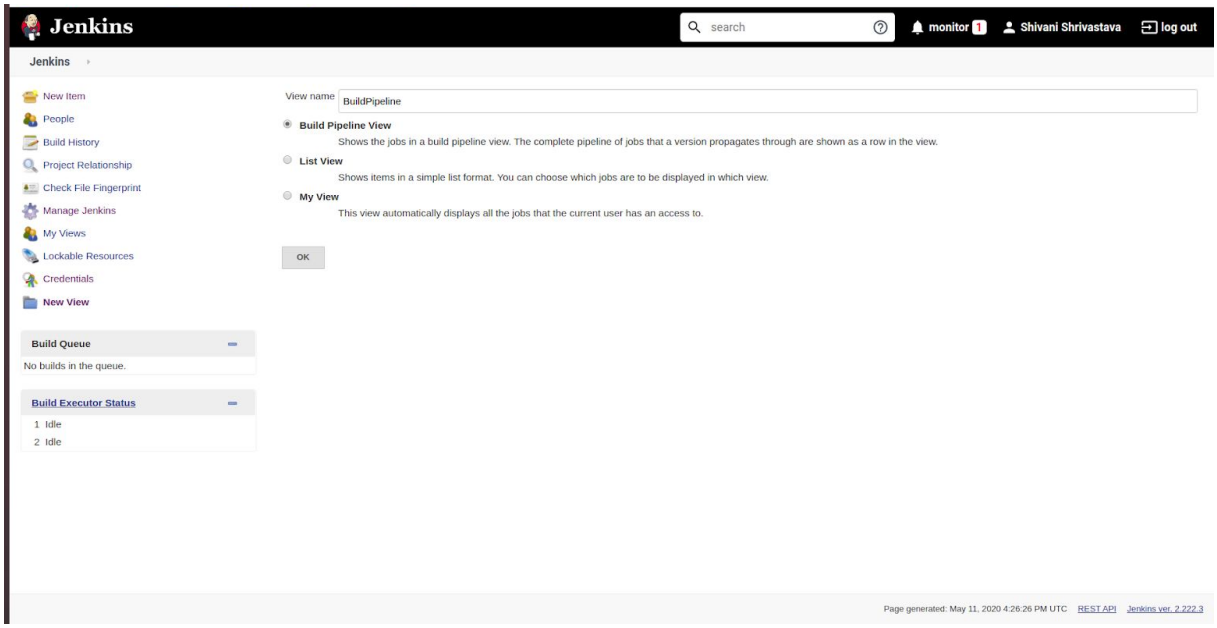


Fig 31. Build Pipeline View setup 1

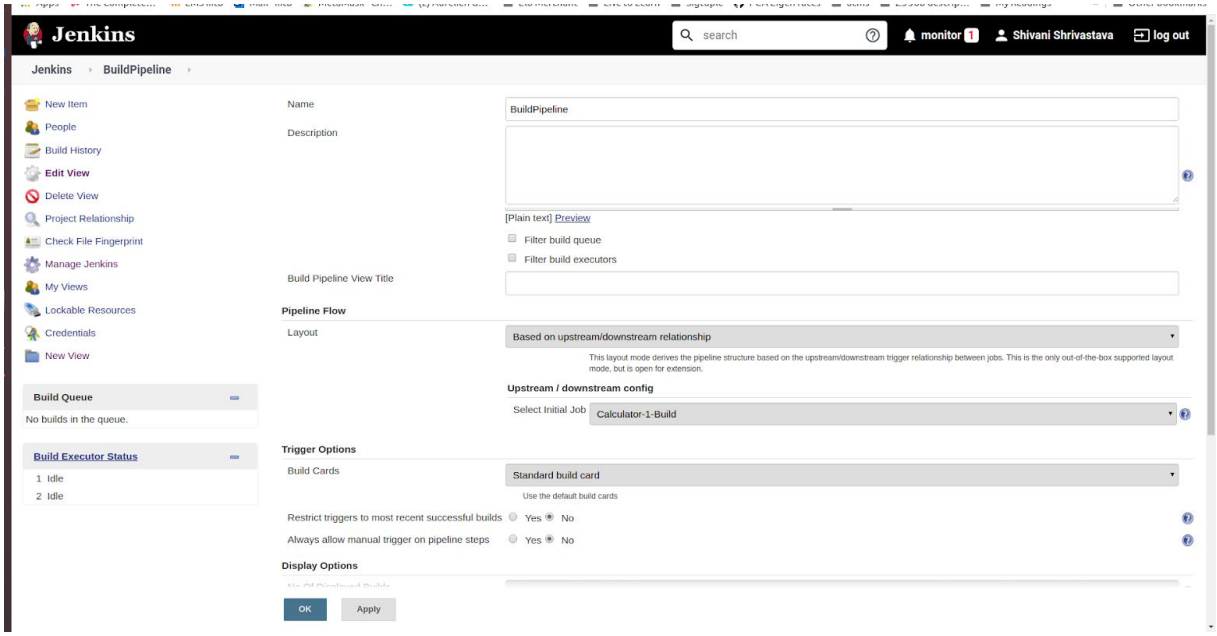


Fig 32. Build Pipeline View Setup 2

Provide description of the pipeline view and select initial job as Calculator-1-Build and keep the rest options as default. Click on Apply and then OK.
Pipeline is ready.

# Outputs

Whole pipeline is ready to use now. Start adding other operations subtraction, multiplication and division one after the other and build the projects in jenkins and see the results in pipeline view.

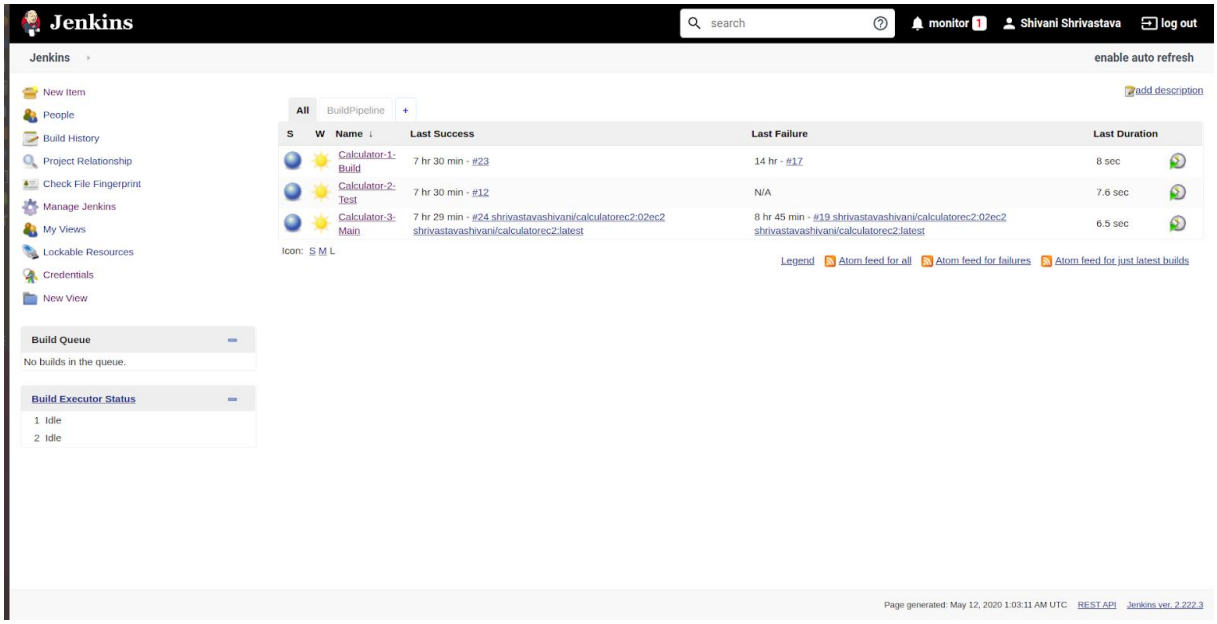


Fig 33. Jenkins Dashboard after everything is done

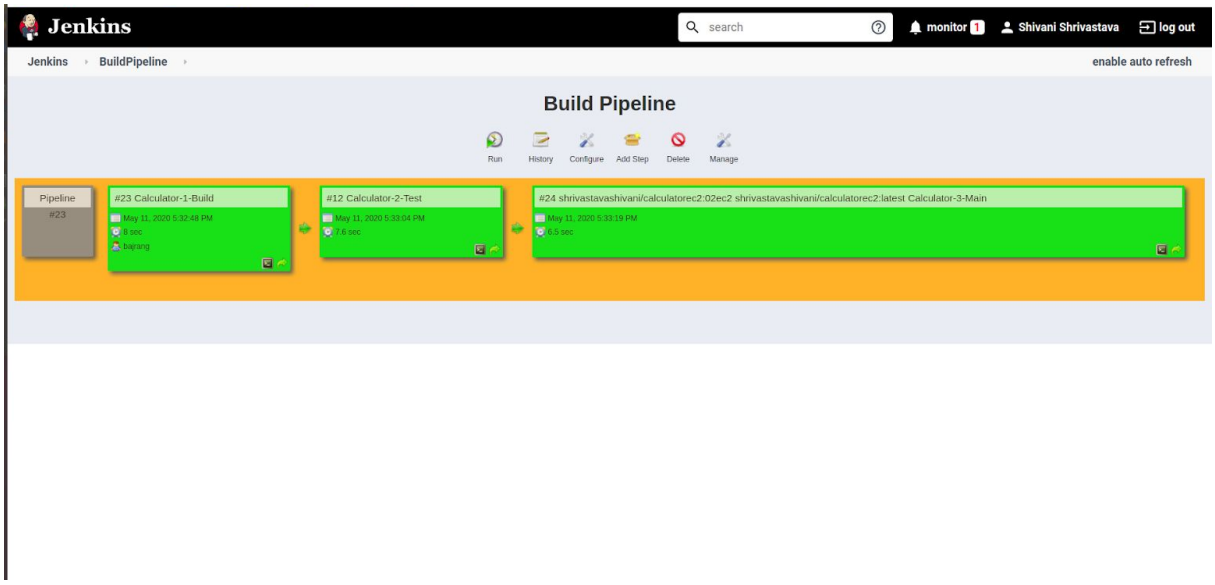


Fig 34. Build Pipeline View

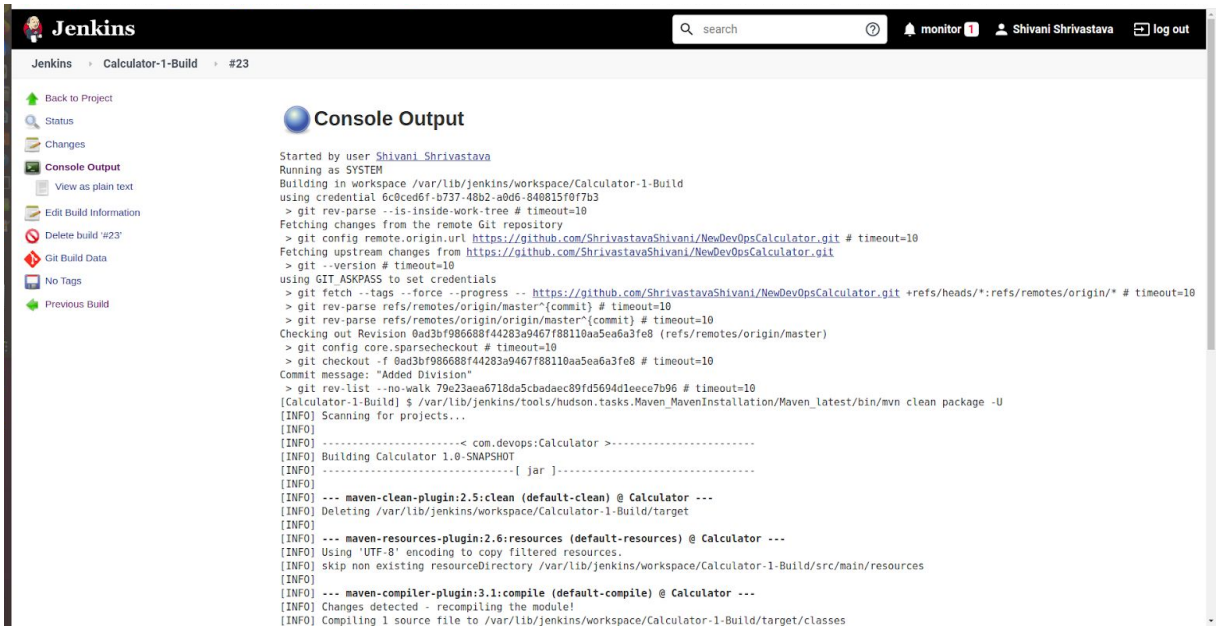


Fig 35. Console Output for Calculator-1-Build

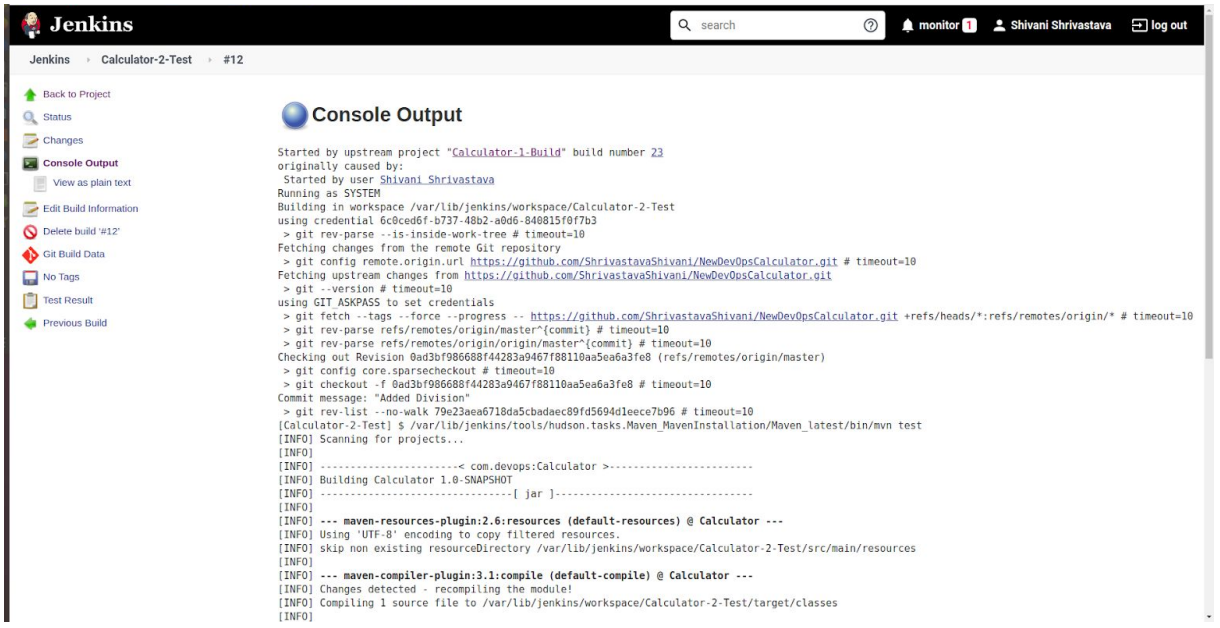


Fig 36. Console Output for Calculator-2-Test

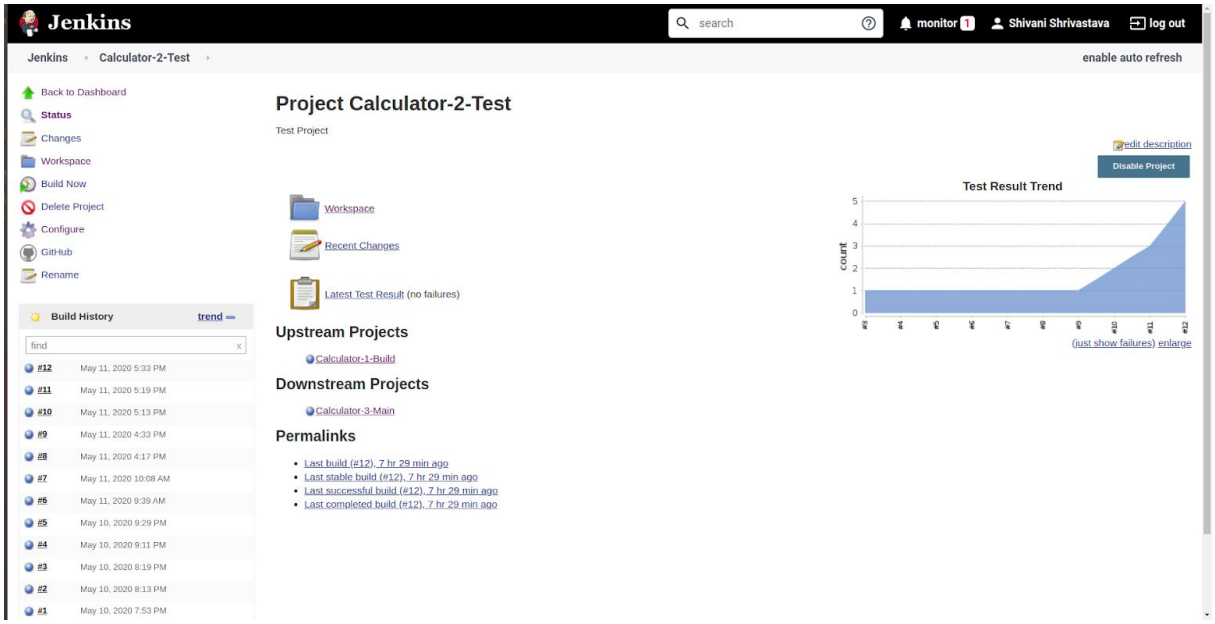


Fig 37. Test Result for Calculator-2-Test

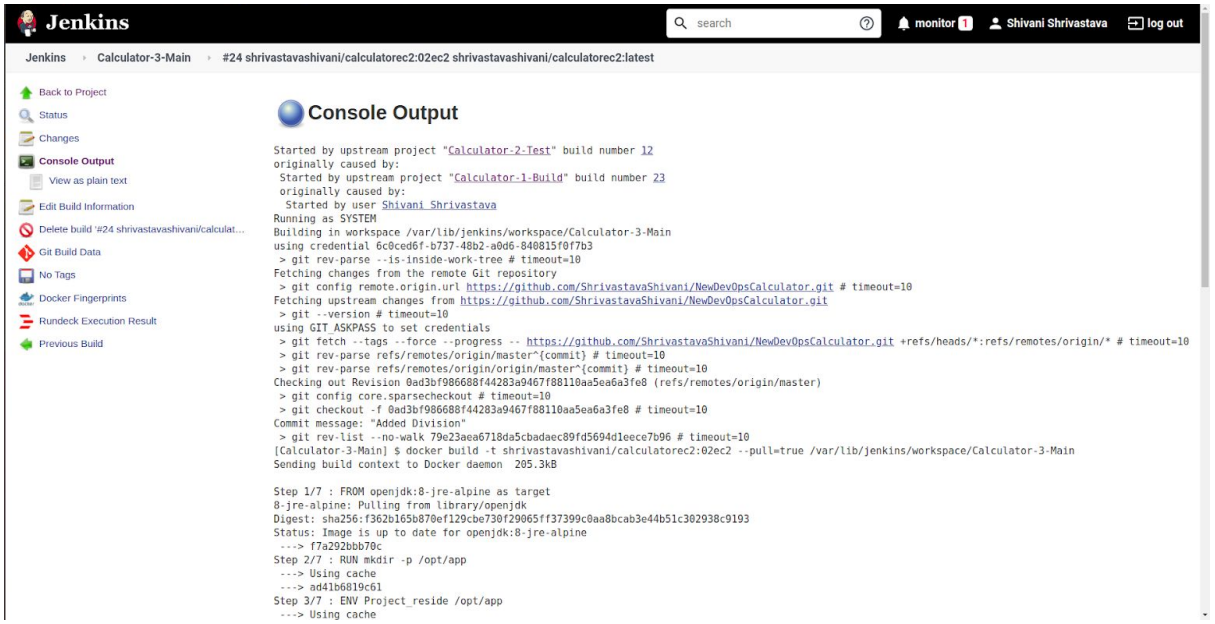


Fig 38. Console Output for Calculator-3-Main

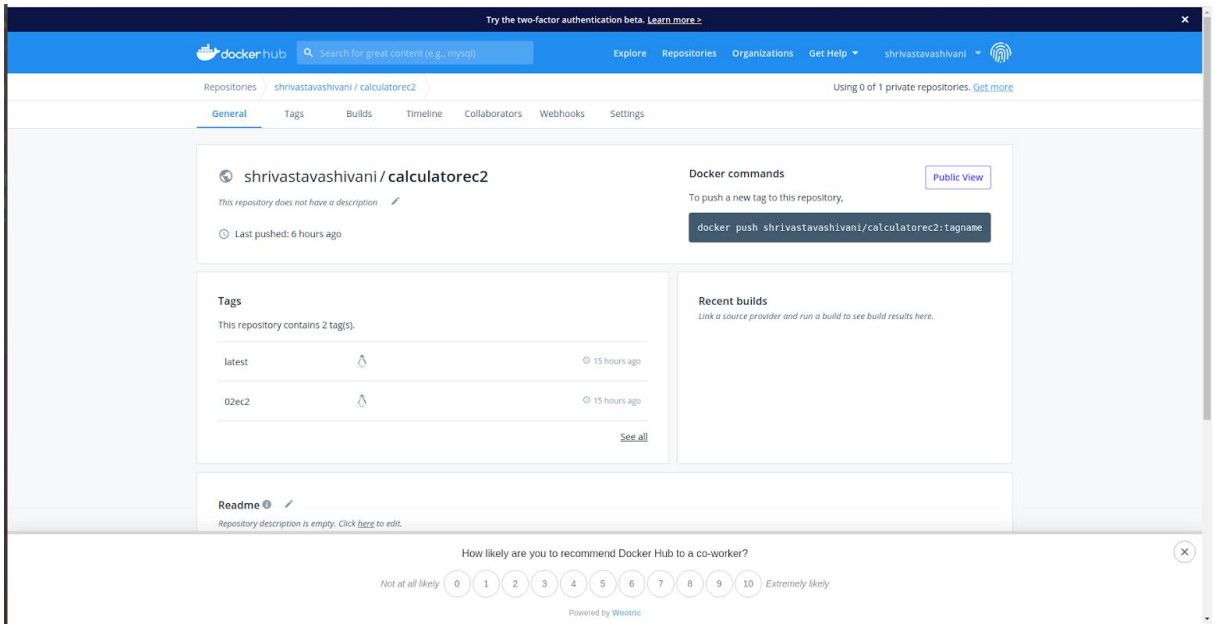


Fig 39. My Docker Hub

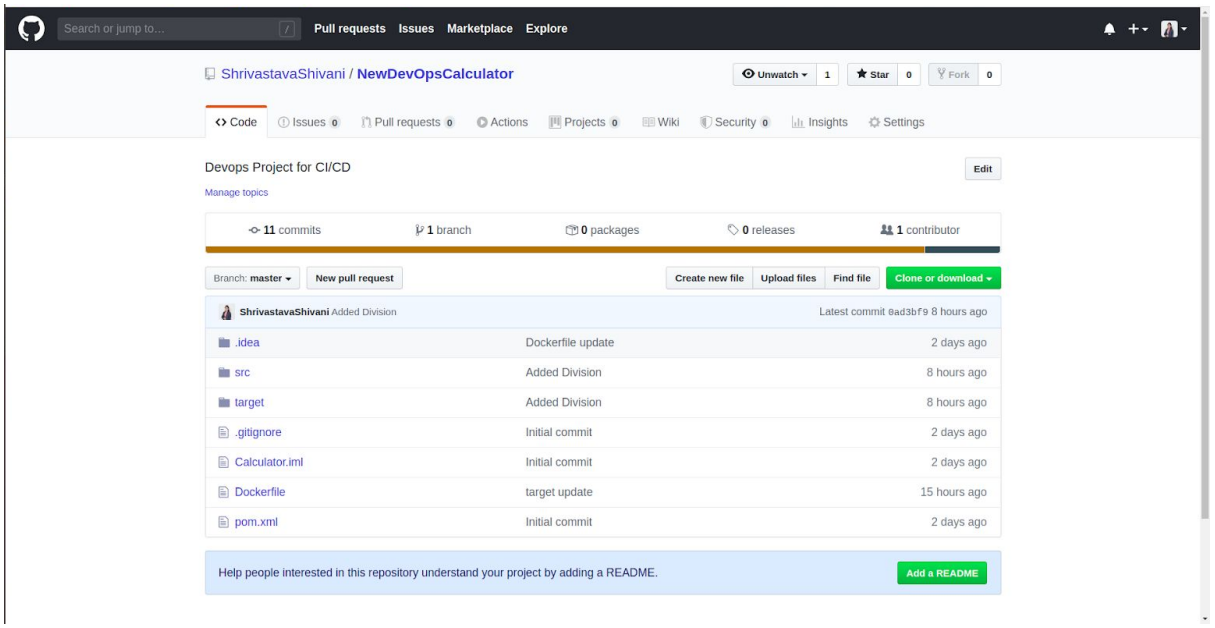


Fig 40. My GitHub