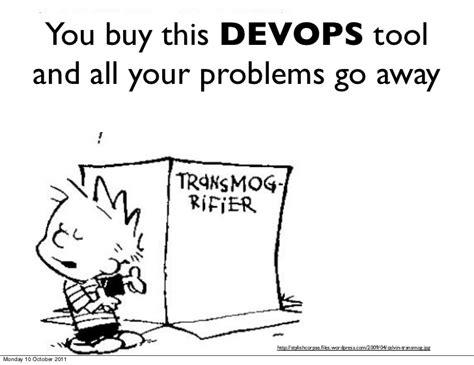
**What and Why DevOps?**

DevOps is a set of practices and tools that aims to improve collaboration and communication between software development and operations teams, and to streamline the software development process from code to deployment.

In our conversation, we have talked about various DevOps practices and tools. For example, we have used Git for version control, Jenkins for continuous integration, Docker for containerization, and configuration management tools Ansible for deployment.

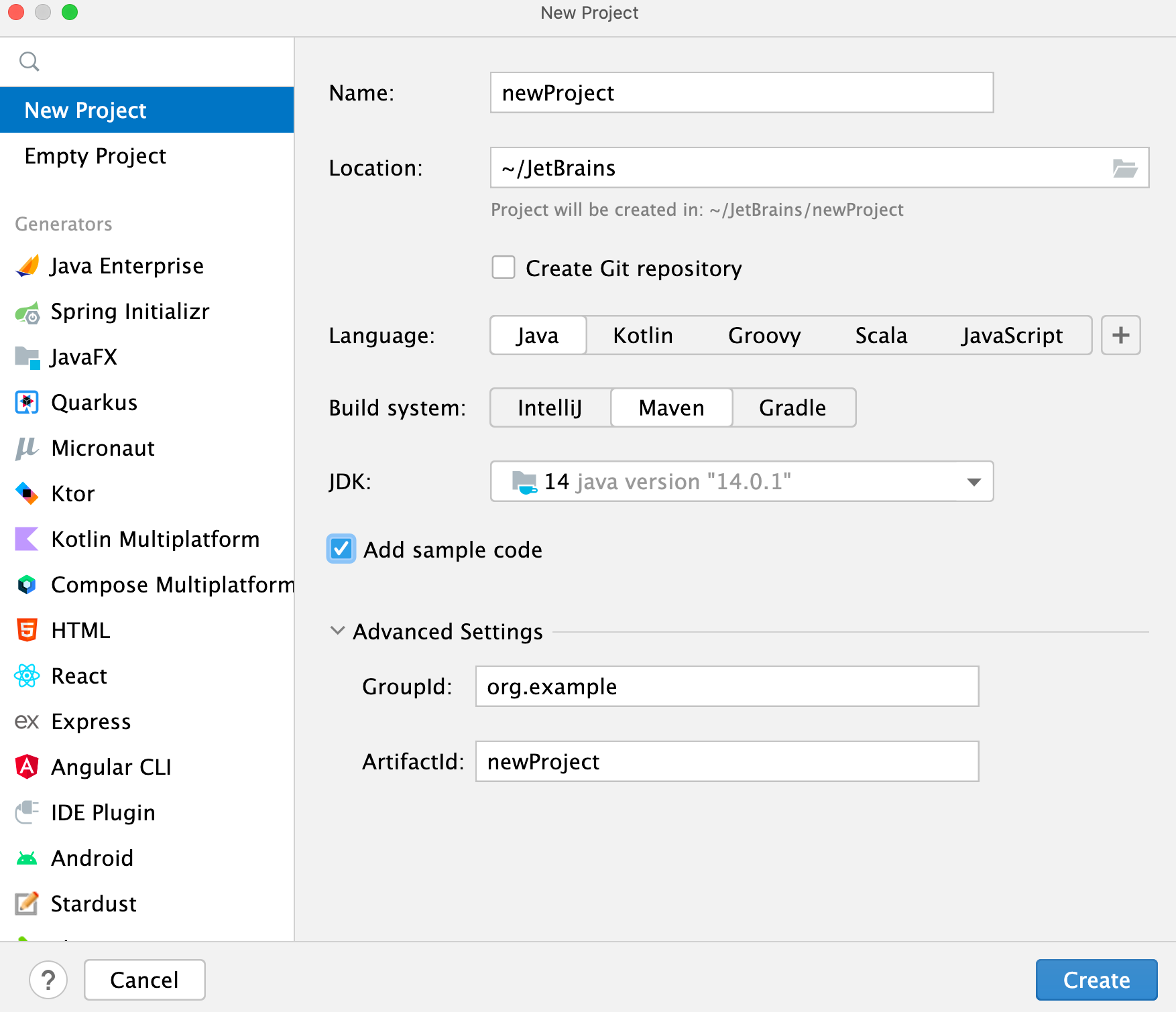
By using these tools and practices, we can automate the software development process, reduce the time between development and deployment, and ensure the quality and consistency of the software. This leads to faster delivery of software products, improved efficiency, and better collaboration between teams.



Overall, DevOps is a way to bring together development and operations teams to work together efficiently and effectively, and to create high-quality software products that meet the needs of users.

**BUILD PROJECT**

**Step1: Create a “New Project” in intellij**

-

Select Maven as “Build system” and hit “Create”

**Step2: Add below dependencies in pom.xml**

~ junit-jupiter-engine

~ mockito-core

~ log4j-api

~ log4j-core

Hit “reload project” to download and refresh dependencies.

=======================================================================

**Step3: Under the Main class write your calculator program.**

Under Test folder create MainTest class and write your test cases here.

Create “log4j2.xml” file under “main/resources”

Add the xml configuration for your logger.

Run the Main and MainTest class to see if working perfectly.

**Step4: Push project to github**

First, make sure that Git is installed on your system. You can check this by running the following command in the terminal:

**git –version**

If Git is installed, the version number will be displayed. If not, you can install it by running:

**sudo apt-get update**

**sudo apt-get install git**

Once Git is installed, navigate to the directory where your Java code file is located using the **cd** command. For example:

**cd ~/Documents/ScientificCalculator**

Initialize Git in this directory by running the following command:

**git init**

Add the URL for the remote repository to your local repository by running the following command, replacing **your-username** and **your-repo-name** with the appropriate values:

**git remote add origin** [**https://github.com/your-username/your-repo-name.git**](https://github.com/your-username/your-repo-name.git)

Add your Java code file to the Git repository by running the following command:

**git add MyProgram.java**

Commit the changes to the repository by running the following command:

**git commit -m "Initial commit"**

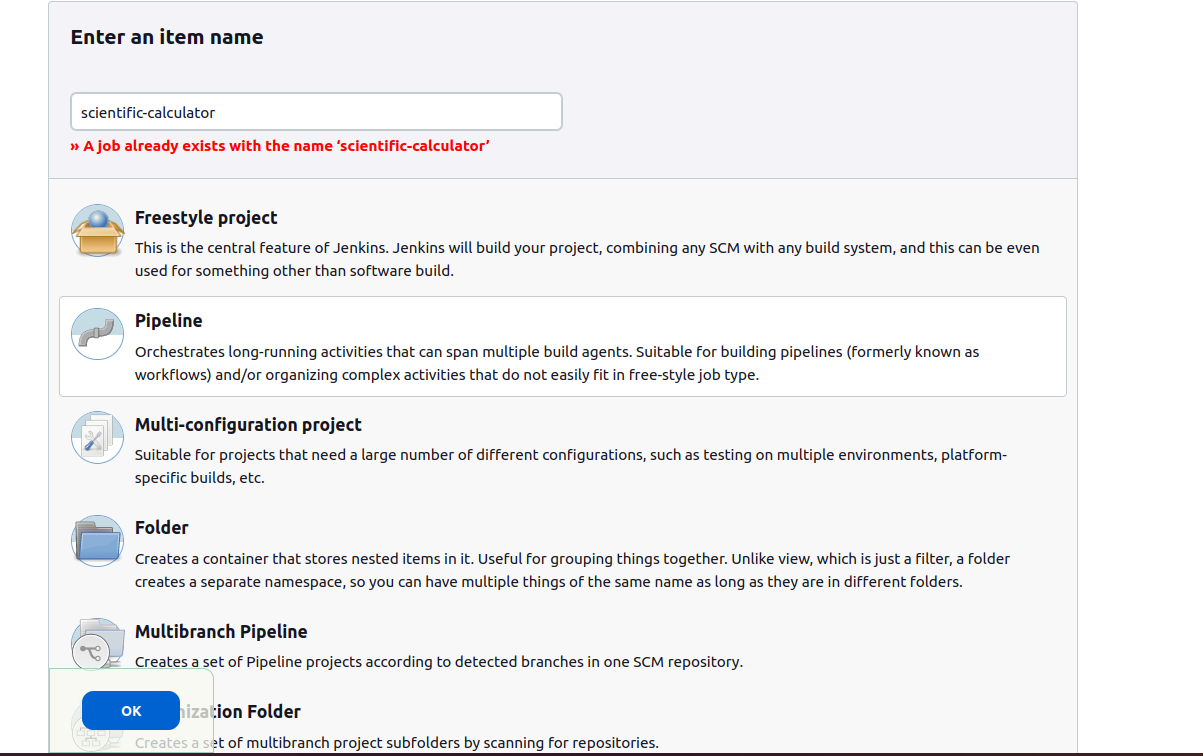
Push the changes to the remote repository by running the following command:

**git push -u origin main**

**Step4: Create a Jenkins Pipeline project**

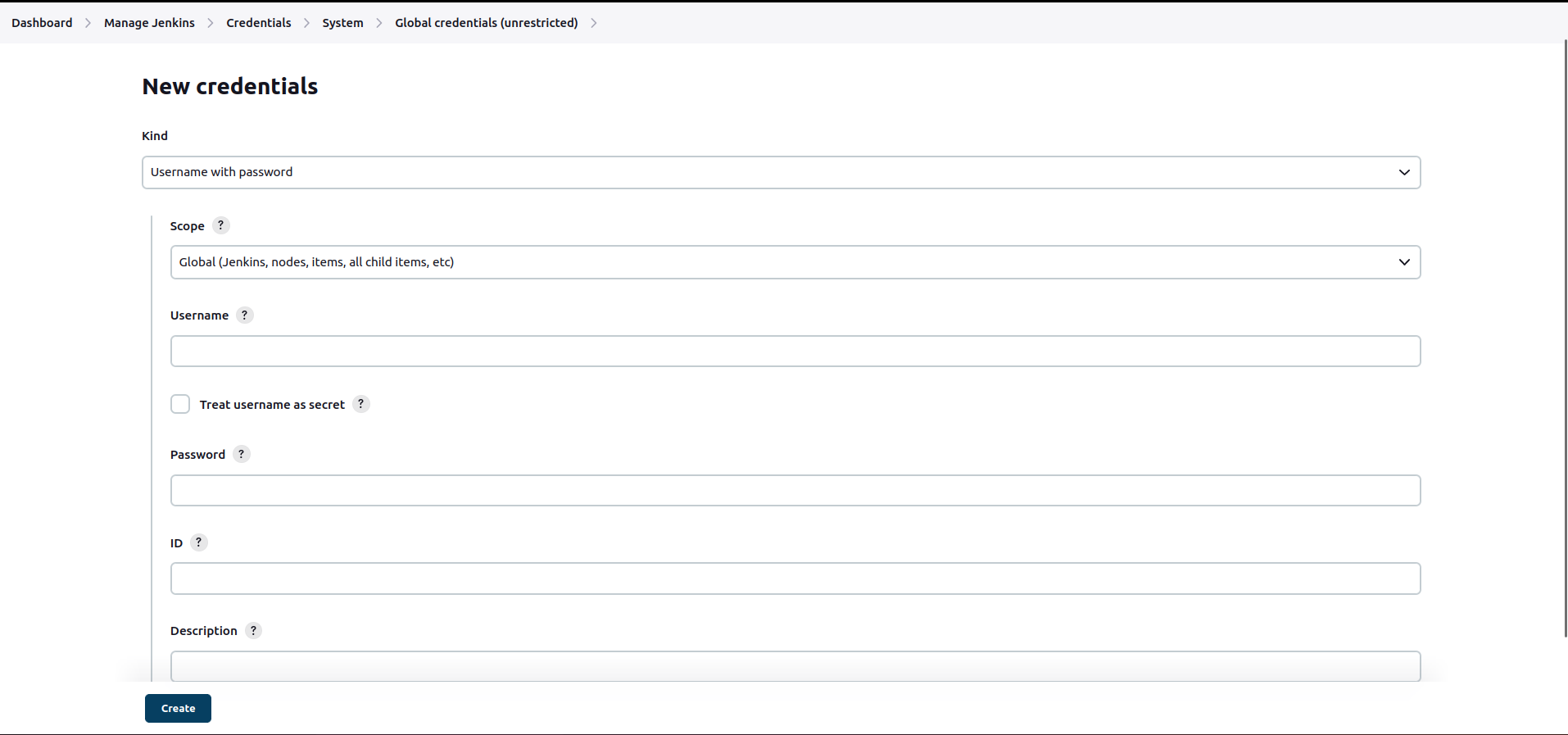
Write any name and select Pipeline.

Hit OK when done.

-=======================================================================

**Step5: Go-to Dashboard > Manage Jenkins > Credentials > System Global credentials (unrestricted)**

Select “+ Add credentials”



Username: Add your github username

Password: Add your github password

ID: (Optional) give any number/text/combination…

Description: (Optional) write credential details

Hit “Create”

Again select “+ Add credentials”

Now add username and password of your Dockerhub website.

Hit “Create”

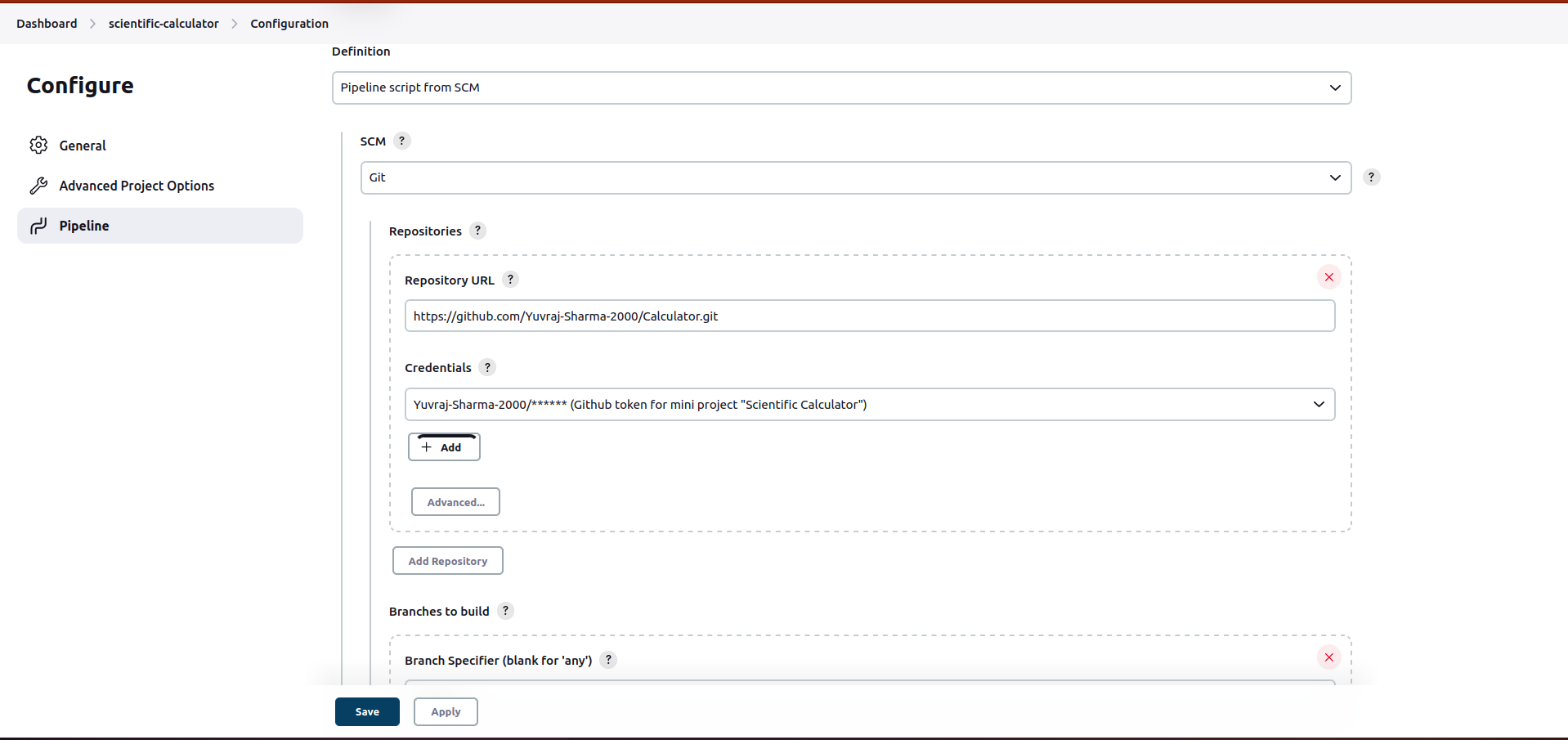
=======================================================================

**Step6: Go-to the newly created project and select “Configure”**

Scroll down to “Advanced Pipeline Section”

Pipeline > Definitions > “Pipeline script from SCM” > SCM > “git” > Repositories >

add your github url and select “github” credentials configured in previous step.

-

Scroll down and specify the branch for the project and location of Jenkinsfile

(This Jenkinsfile is present in github repo)

=======================================================================

**Step7: Install below required plugins in Jenkins**

~ Ansible Plugin

(Invoke [Ansible](http://www.ansible.com/) Ad-Hoc commands and playbooks)

~ Pipeline: GitHub Groovy Libraries Version 3

(Allows Pipeline Groovy libraries to be loaded on the fly from GitHub)

~ Docker Plugin

(This plugin integrates Jenkins with [Docker](http://docker.io/))

~ Junit

(Allows JUnit-format test results to be published. )

~ Pipeline

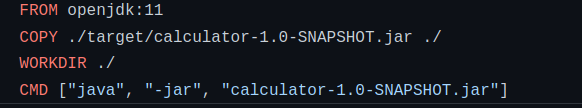
(A suite of plugins that lets you orchestrate automation, simple or complex)

~ SSH server

(Adds SSH server functionality to Jenkins, exposing CLI commands through it)

=======================================================================

**Step8: Create “Dockerfile” in root directory. It define show to build a docker image.**

 *Replace the XXX.jar with your jar file name.(you can find .jar file in target directory)*

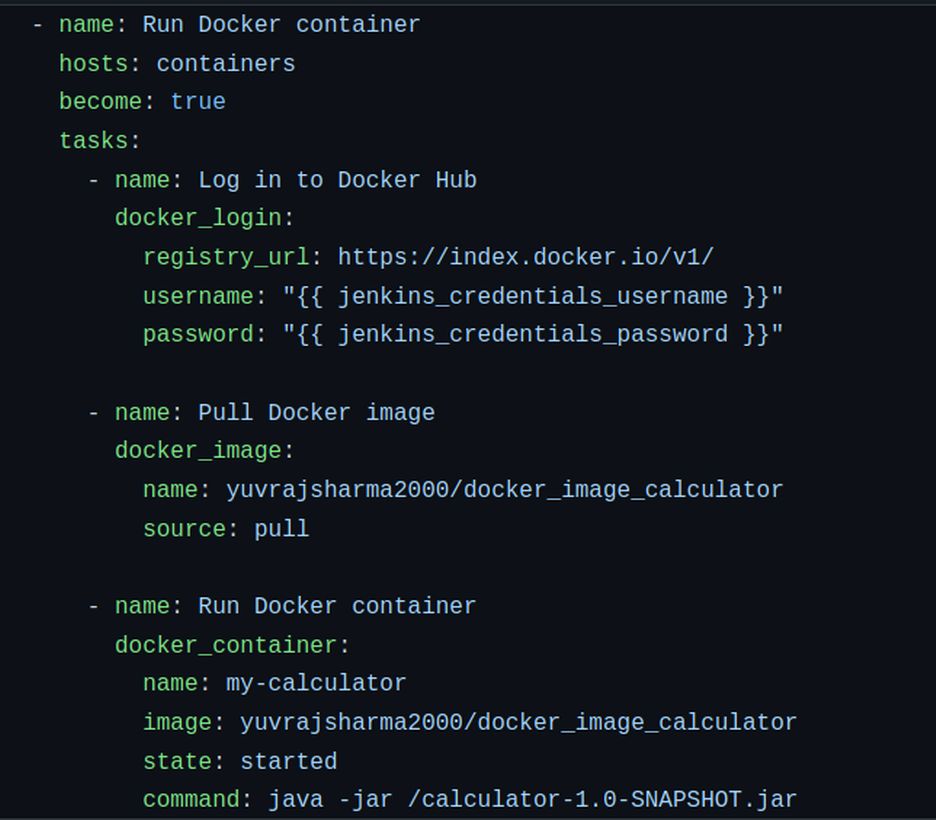
=======================================================================

**Step9: Create “inventory” file in root directory. It contains list of hosts that ansible will manage.**

*Add the host’s IPA,username and password you want to manage*.

=======================================================================

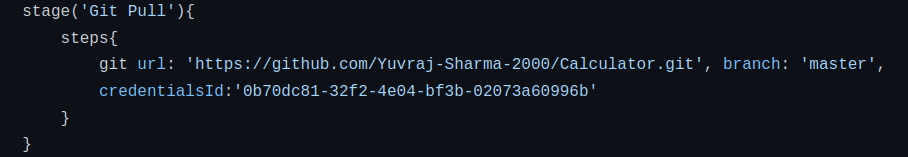
**Step10: Create playbook.yml file in root directory. It define sthe tasks to run on ansible hosts.**



=======================================================================

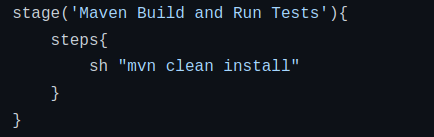
**Step11: Create a file named “Jenkinsfile” in the root directory of project.**

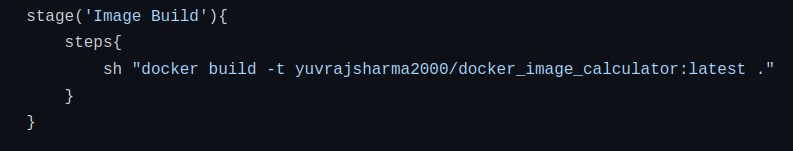
Add below pipeline stages-

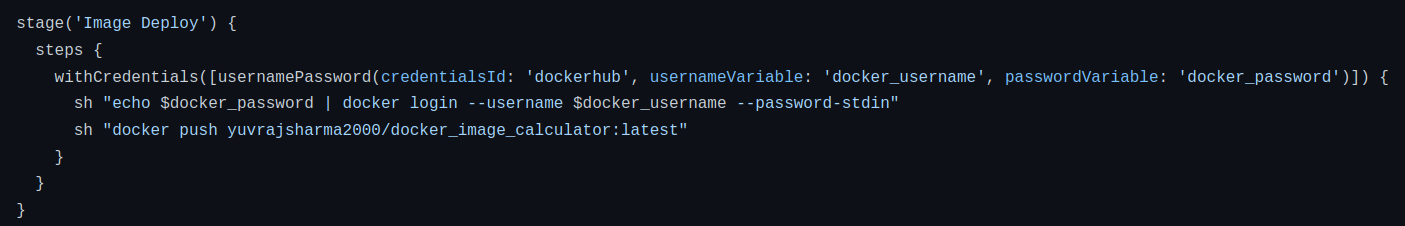


-

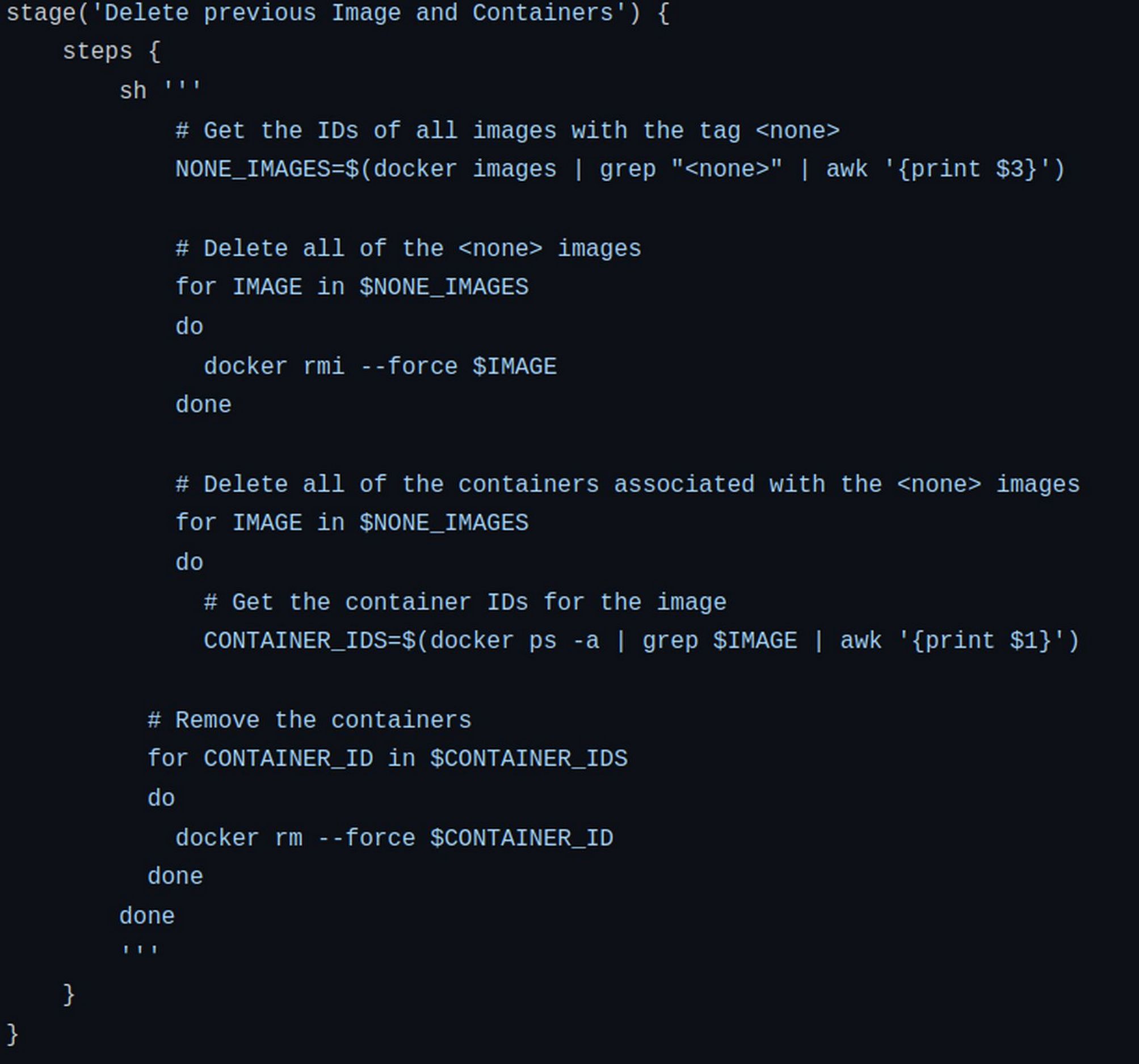
*Change git url, credentialsId and branch as selected in job configure section.*

*write this code as it is.*

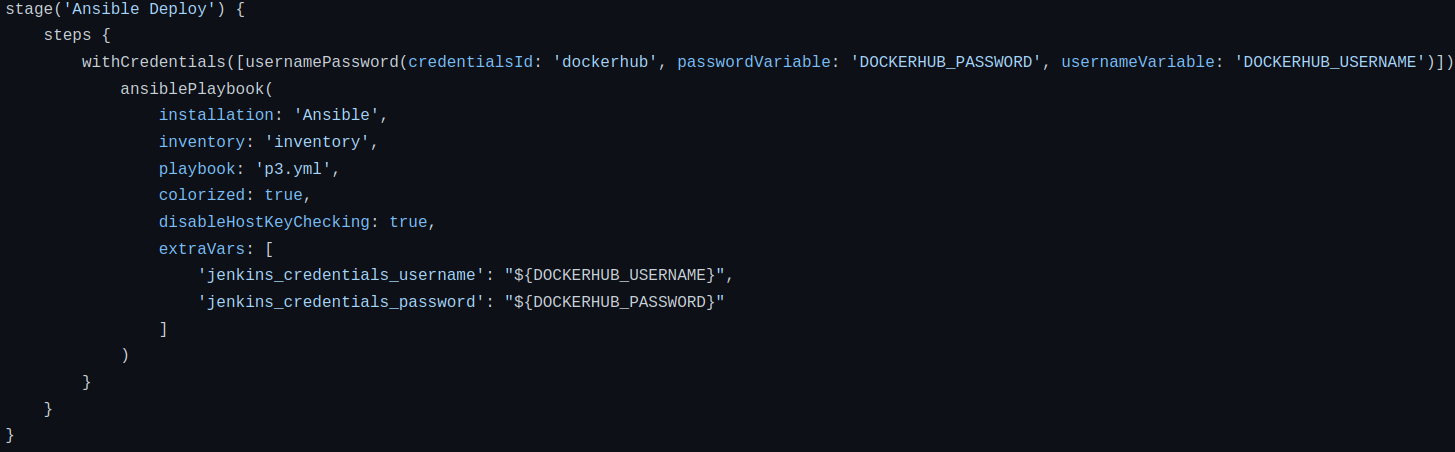
*Change dockerhub image name as per your desire.*

*Change credentialsId to your jenkins credentials dockerhub.*

*Pushing image created in previous image to dockerhub*

*Remove previous builds (to free up space and reduce manual work) using tag <none>.*

*Also remove containers belonging to <none> images.*

*Use the inventory file to find groups to deploy image.*

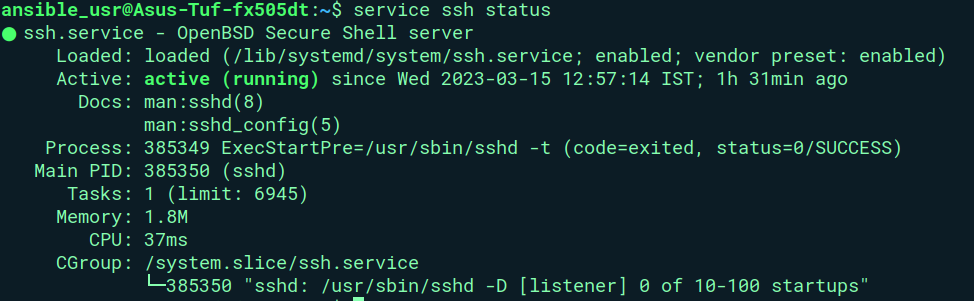
*Use p3.yml to describe the tasks you want to run after deploying.*

*Pass the credentials to the p3.yml as variables.*

Note-

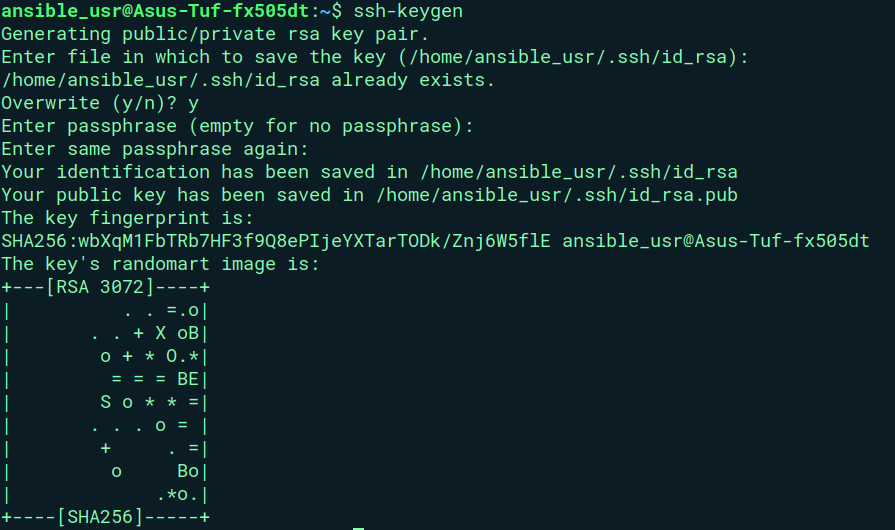
Before running the project make sure you have ssh installed, running and key generated in your local machine user.

You can check if ssh is running by-

-

*If ssh is not started then use command : “service ssh start”*

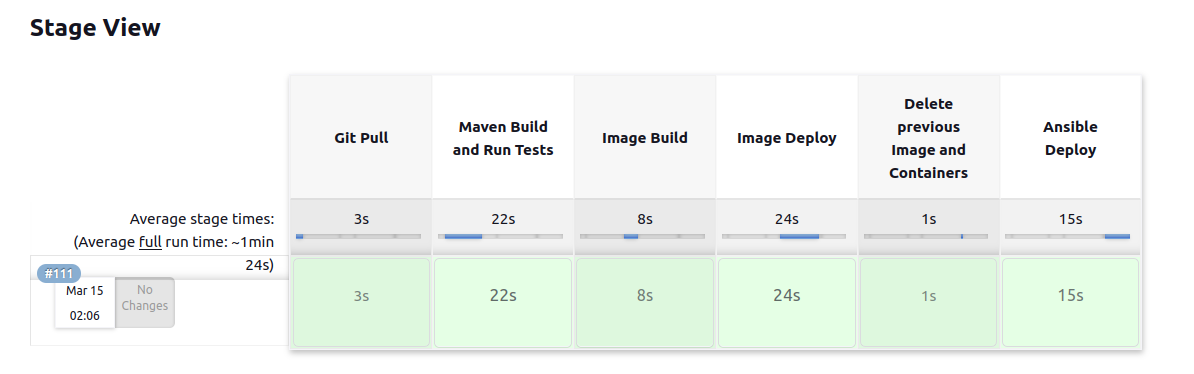
You can generate key by-

-

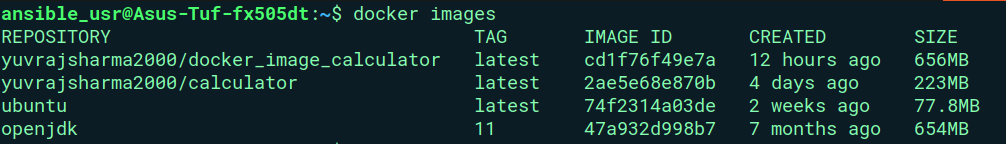
Now you are ready to run the project.

**RUN PROJECT**

On clicking “Build Now” you should see below pipeline-

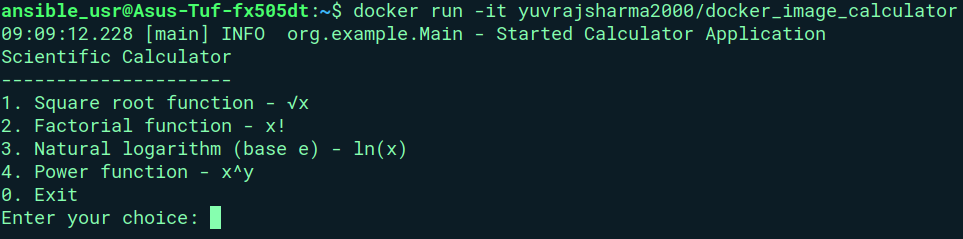
-

To check if image is deployed correctly, you can login to the user specified in inventory file and check images.

*Here we can see “yuvrajsharma2000/docker\_image\_calculator” image is build.*

=======================================================================

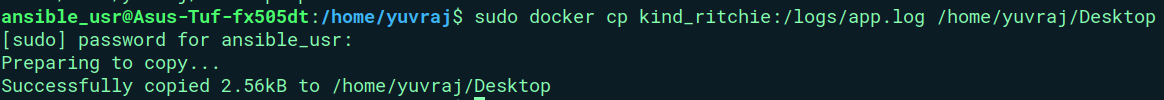
By running the image, we can see if everything worked right-

*As we can see the program is running perfectly.*

**Monitor in ELK**

Step1 : Install ELK

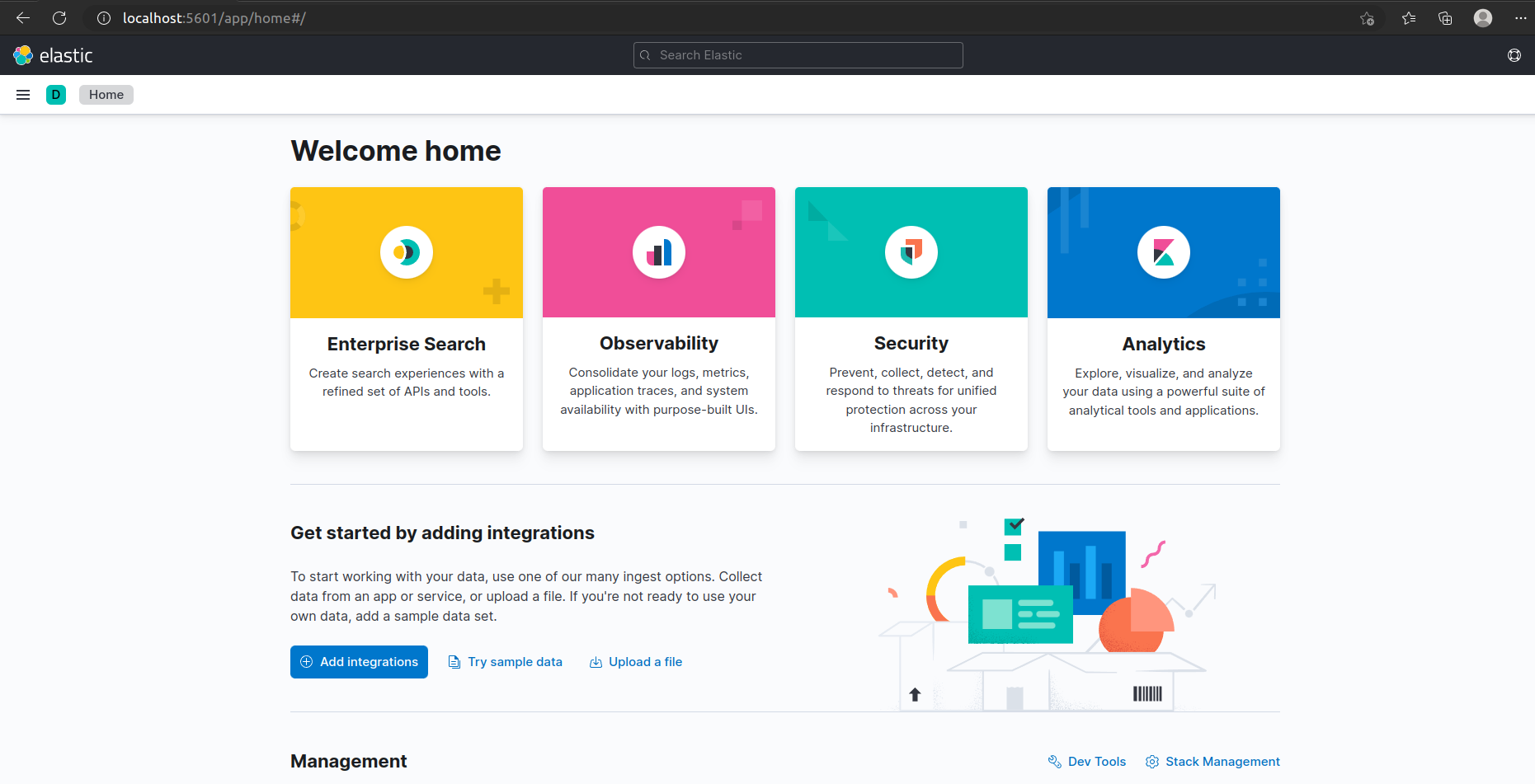
Step2: Copy your log file from the container to local file system (Desktop in our case)

 change the name of container

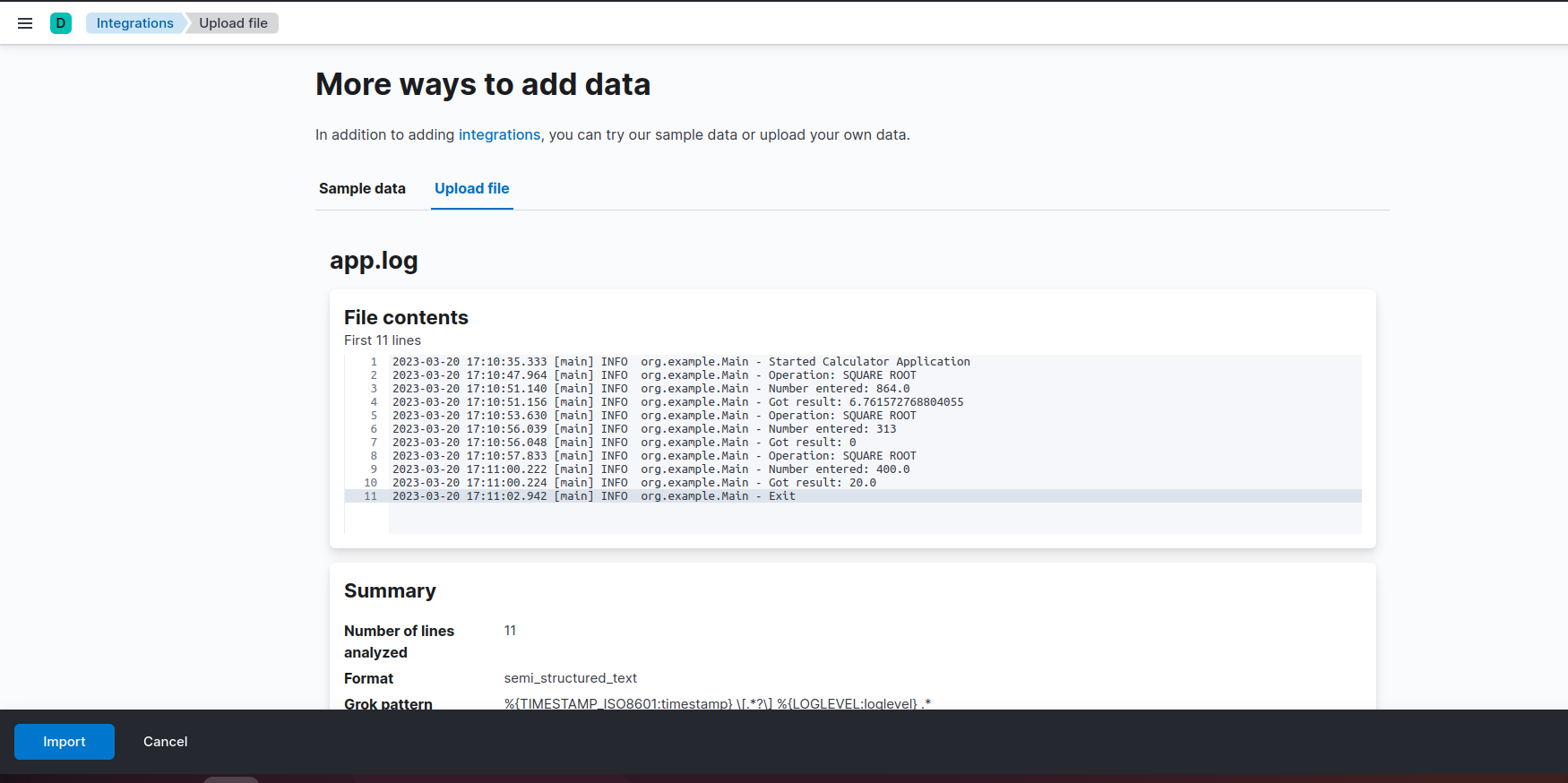


Here you can see we have app.log file copied on Desktop

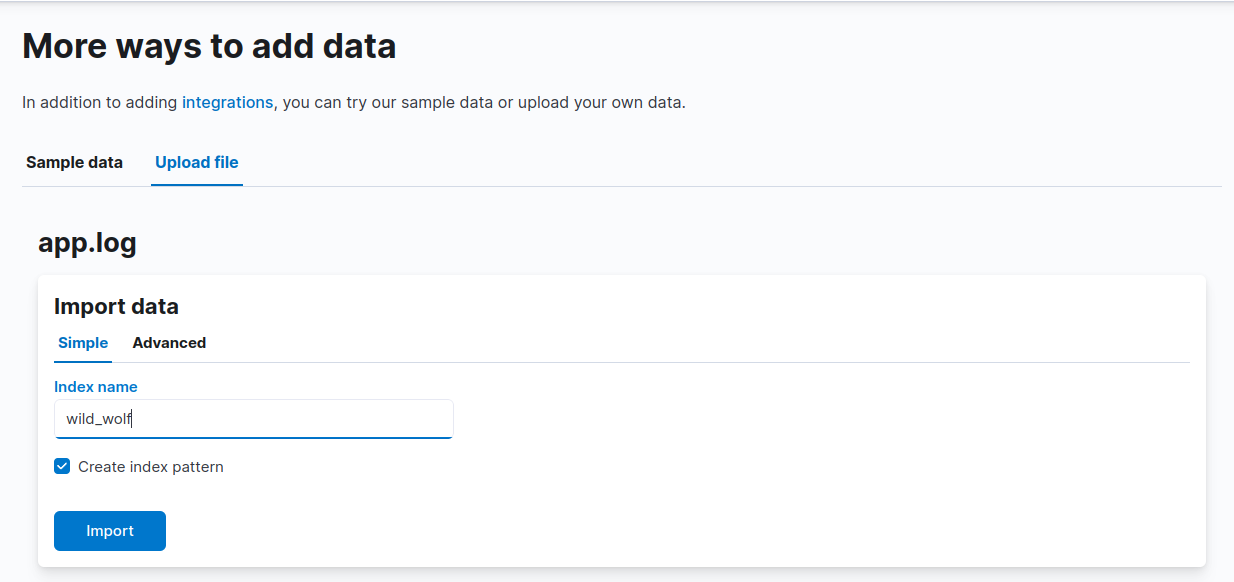
Step3: Upload the file on ELK stash



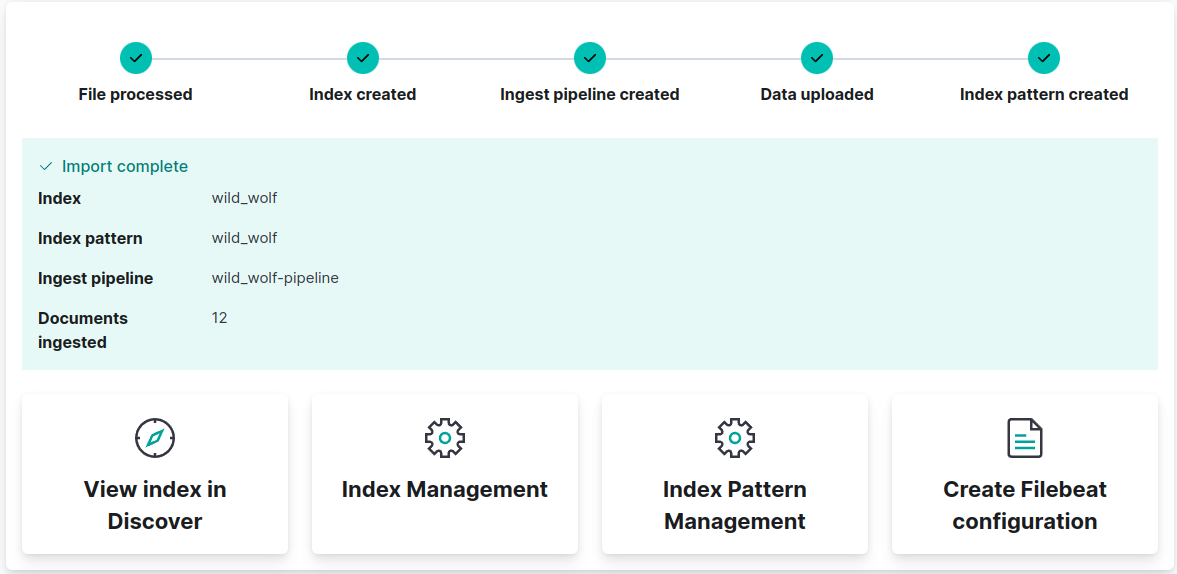
Step4: Select app.log from desktop and upload

hit import

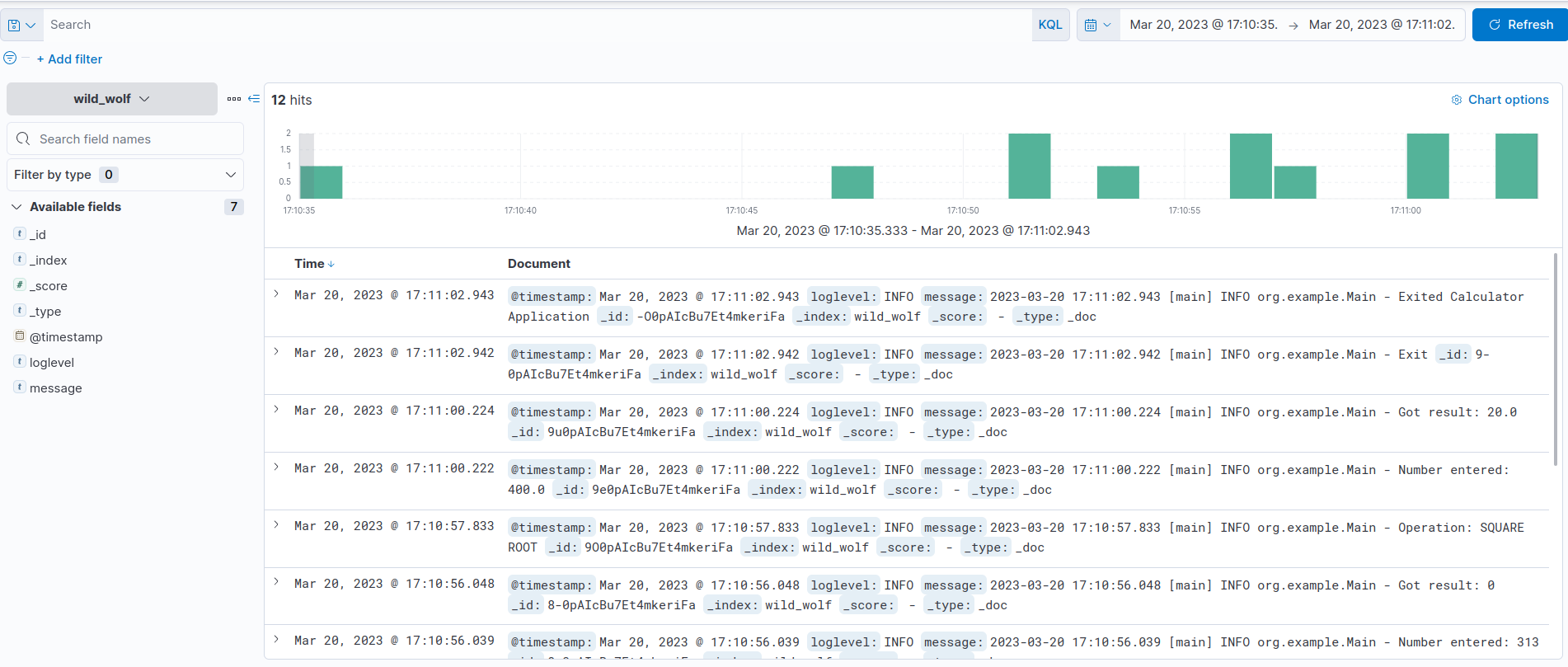
Step5: Enter any name you want to give to your pattern



Step6: Click on View Index in Discovery



Step7: Now you can view and search logs as shown below



*Links used in the project-*

Github: <https://github.com/Yuvraj-Sharma-2000/Calculator.git>

Docker hub: <https://hub.docker.com/repository/docker/yuvrajsharma2000/docker_image_calculator>