SPE MINI Project

- *Chaithanya*
- *IMT2020054*

Github Link: [chaithanya99/Calculator (github.com)]

Introduction:

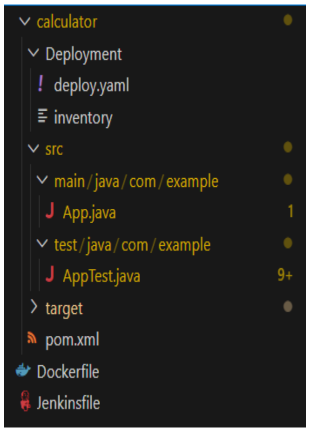
The aim of this project is to learn about the DevOps tools and CI/CD pipeline. I developed the calculator program to learn these tools. The calculator program takes two input values and 4 operations can be performed (addition, subtraction, multiplication, and division).

Tools and Languages Used:

1. Java: The Calculator program was written in Java language.
2. Maven: Used Maven to build the Java programs and to do the unit testing automatically. Maven can be used to generate the jar file.
3. Git: It's a version control tool used to manage the code in a local/remote repository efficiently.
4. Jenkins: It is used for the creation of pipeline and automation.
5. Docker: Used docker for containerization and docker hub to remotely store our images.
6. Ansible: It's an IT automation software application that can be used to configure systems, deploy software and more. In this project, I used this to run the docker container.
7. Ngrok: It's used to expose the local development server to the internet. In this project I used it to expose the jenkins server.

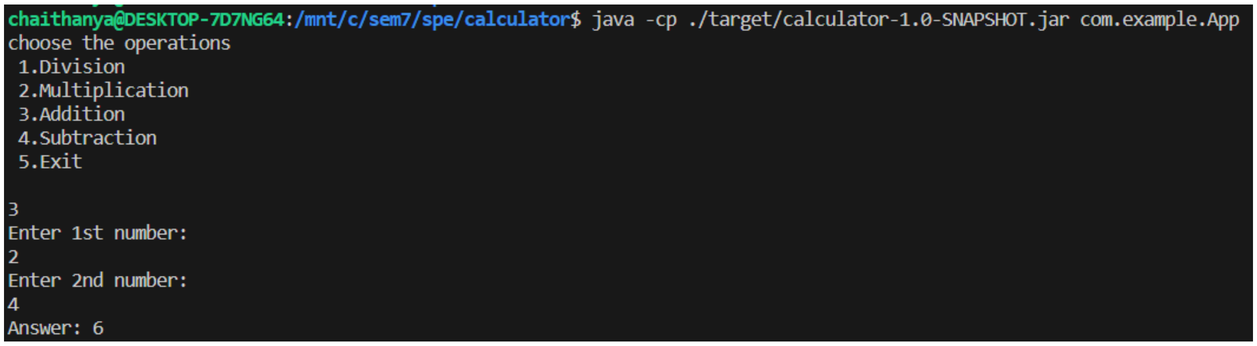
Steps:

1. Project Setup: I used VS Code for this project and used Maven to create project setup. Maven provides variety of project types I selected the quick start.



```
calculator
  Deployment
    deploy.yaml
    inventory
  src
    main/java/com/example
      App.java                 1
    test/java/com/example
      AppTest.java             9+
  target
  pom.xml
Dockerfile
Jenkinsfile
```

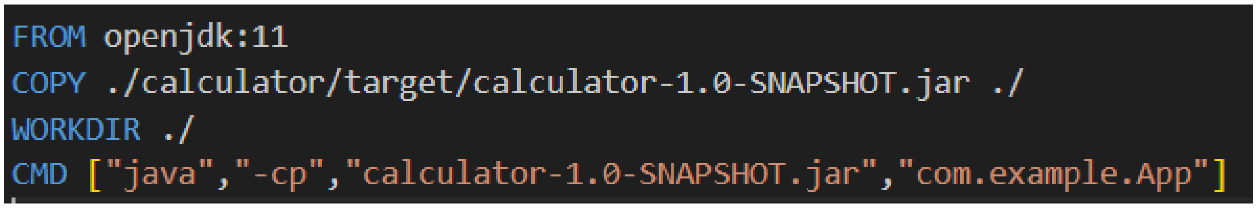
This is the file structure of the project. Calculator program is written in the App.java file. I wrote four functions each one for each operation so that it will be helpful while testing the program. I added test cases in the AppTest.java. Maven will automatically test the project while building the project.



```
chaithanya@DESKTOP-7D7NG64:/mnt/c/sem7/spe/calculator$ java -cp ./target/calculator-1.0-SNAPSHOT.jar com.example.App
choose the operations
 1.Division
 2.Multiplication
 3.Addition
 4.Subtraction
 5.Exit

3
Enter 1st number:
2
Enter 2nd number:
4
Answer: 6
```

In the above image, we can see how the program works. It asks us to choose the operation first then it asks for two inputs and then it prints the answer. There is an option to exit the program. We can use the commands mvn clean, mvn compile and mvn install to clean, to compile project and test, and to generate the jar file. We can see the jar file in the target directory.

2. Git: I used the ‘git init’ command to integrate the project with the git. Thereafter I used ‘git add .’ and ‘git commit -m message’ to keep track of the files and save the changes in the files. I created a remote repository in GitHub. I connected the local repository to the remote repository using the ‘git remote add origin URL” command. Then I pushed the whole repository into GitHub.
3. Docker: Installed the docker in the machine and created an account in the docker hub so that we can push images into the docker hub and connect to the docker hub in the terminal. I created a docker file so that we can build the image.

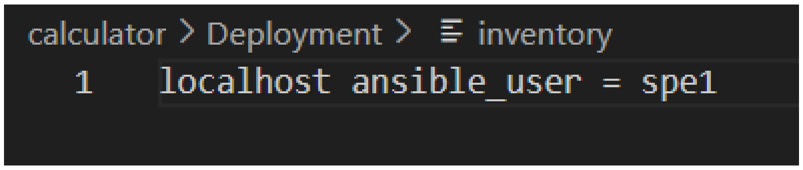


```
FROM openjdk:11
COPY ./calculator/target/calculator-1.0-SNAPSHOT.jar ./
WORKDIR ./
CMD ["java","-cp","calculator-1.0-SNAPSHOT.jar","com.example.App"]
```

We can see the content of the docker file. I am creating the image on top of openjdk:11 because we need Java to run our program. I copied the jar file into the docker image. When the docker container runs it will automatically run the jar file. The command in the CMD will execute when the container starts running. Now we can build an image and push it to the docker hub which I did using Jenkins so it will automatically create an image and push it to the docker hub.

4. Ansible: We created the docker image and pushed it to the docker hub using Jenkins. We can use Ansible to deploy the container on multiple servers but for this project, we need to deploy the container only local machine so, the Ansible will pull the image from the docker hub and it will run the image. I created the new directory “Deployment” There

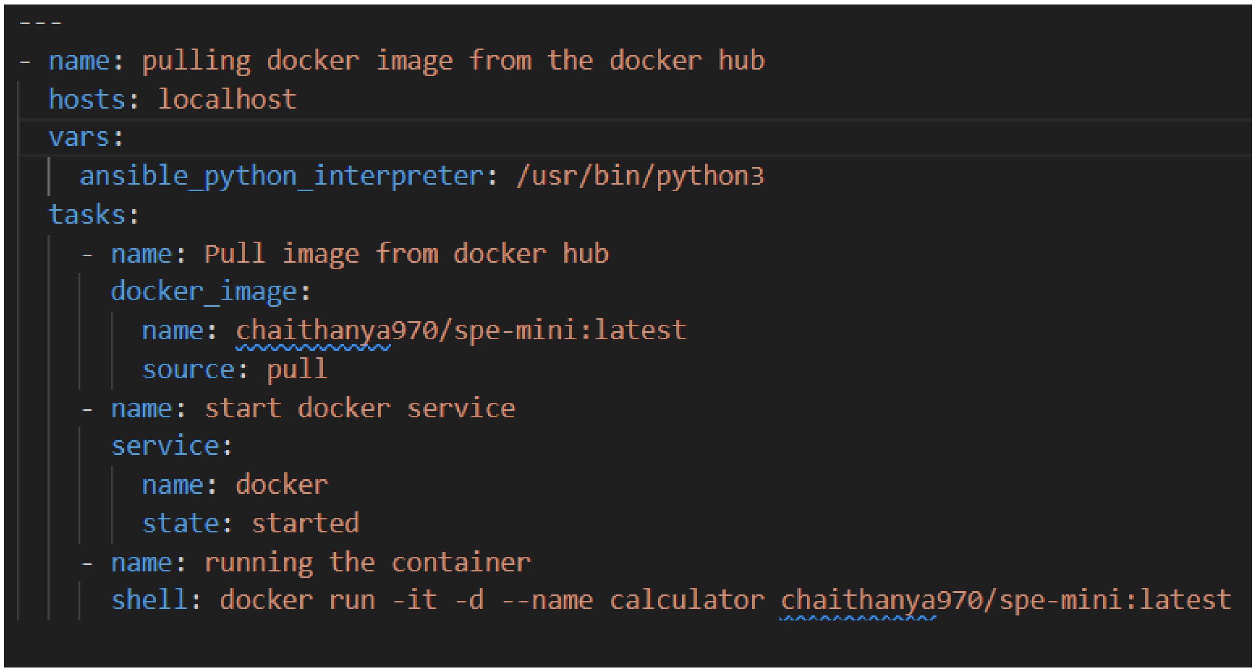
are two files in the folder “inventory file” and “deploy.yaml”. We use an inventory file to store the IP address of the server for the deployment stage.



calculator > Deployment > inventory

```
localhost ansible_user = spe1
```

The inventory file looks like this because we are deploying on the local machine only. Spe1 is the user in the machine.

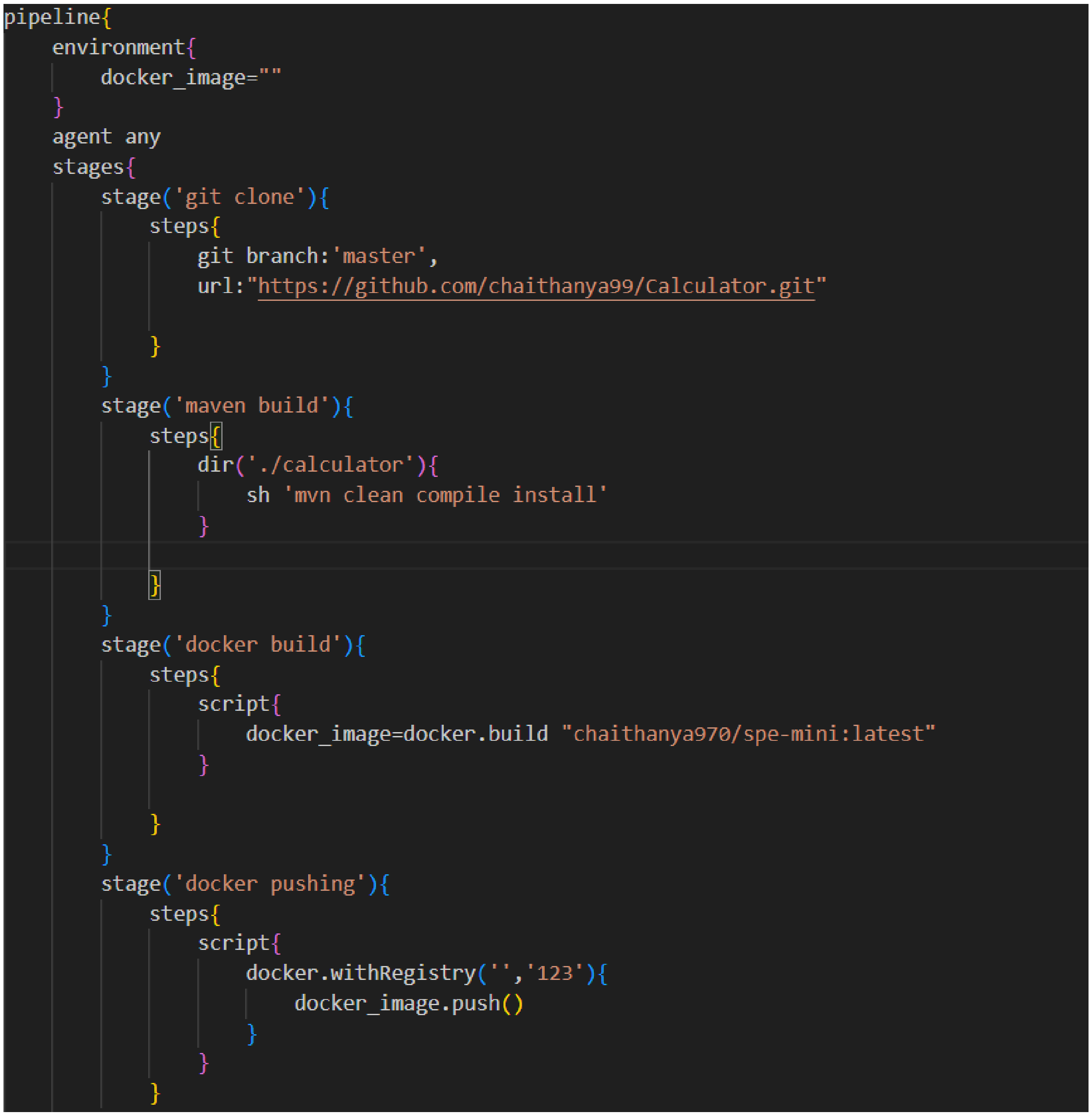


```yaml
---
- name: pulling docker image from the docker hub
  hosts: localhost
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Pull image from docker hub
      docker_image:
        name: chaithanya970/spe-mini:latest
        source: pull
    - name: start docker service
      service:
        name: docker
        state: started
    - name: running the container
      shell: docker run -it -d --name calculator chaithanya970/spe-mini:latest
```

The above image is the Ansible playbook file which contains the steps that need to be followed for the deployment.

First, we are pulling the image from the docker hub and we will check whether the docker service is active or not on the server if it's not active it will start the docker service next it will run the container.

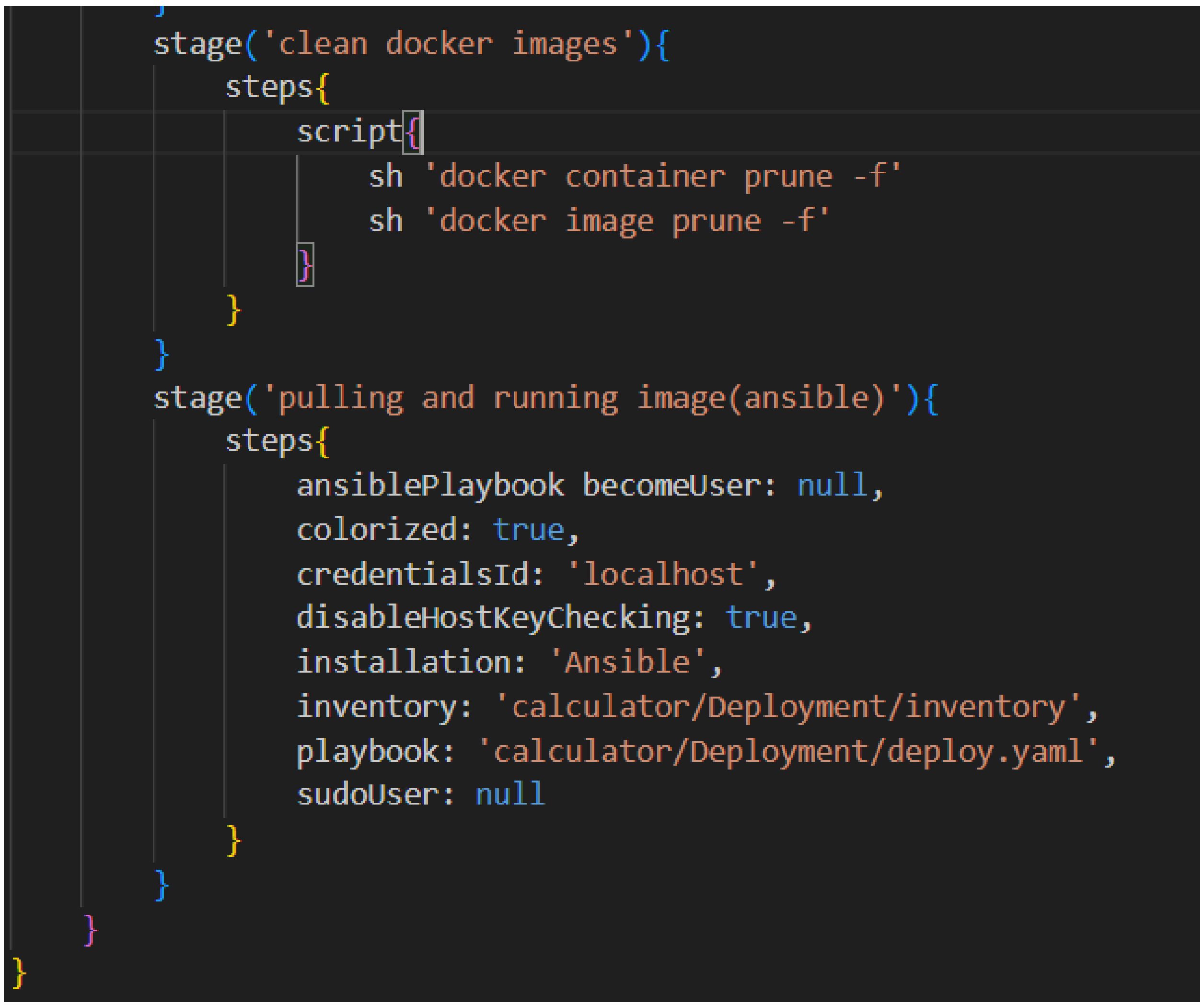
5. Jenkins: Installed Jenkins and required plugins like git, docker, etc. I added the docker credentials to Jenkins so that we can automate the process of pushing the docker to the docker hub. I created the project in Jenkins and chose the pipeline script and I created a Jenkins file in the project.



```
pipeline{
    environment{
        docker_image=""
    }
    agent any
    stages{
        stage('git clone'){
            steps{
                git branch:'master',
                url:"https://github.com/chaithanya99/Calculator.git"

            }
        }
        stage('maven build'){
            steps{
                dir('./calculator'){
                    sh 'mvn clean compile install'
                }

            }
        }
        stage('docker build'){
            steps{
                script{
                    docker_image=docker.build "chaithanya970/spe-mini:latest"
                }

            }
        }
        stage('docker pushing'){
            steps{
                script{
                    docker.withRegistry('','123'){
                        docker_image.push()
                    }
                }
            }
```

We can see the stages in the pipeline.

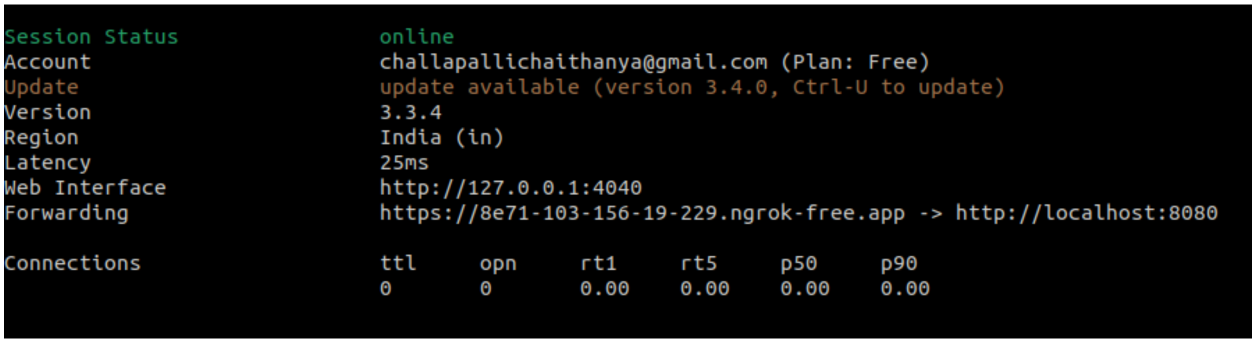
1. In the git clone stage, Jenkins will clone the git repository from the master branch.
2. It will test the code and build the jar file.
3. Now it will build the docker image
4. Now it will push the image to the docker hub since we have added the credentials in Jenkins.



```
        stage('clean docker images'){
            steps{
                script{
                    sh 'docker container prune -f'
                    sh 'docker image prune -f'
                }
            }
        }
        stage('pulling and running image(ansible)'){
            steps{
                ansiblePlaybook becomeUser: null,
                colorized: true,
                credentialsId: 'localhost',
                disableHostKeyChecking: true,
                installation: 'Ansible',
                inventory: 'calculator/Deployment/inventory',
                playbook: 'calculator/Deployment/deploy.yaml',
                sudoUser: null
            }
        }
    }
}
```

5. Now we clear the docker images and docker containers if they are not in running state.
6. It will run the ansible so it will pull images from the docker hub and spin the container on all servers.

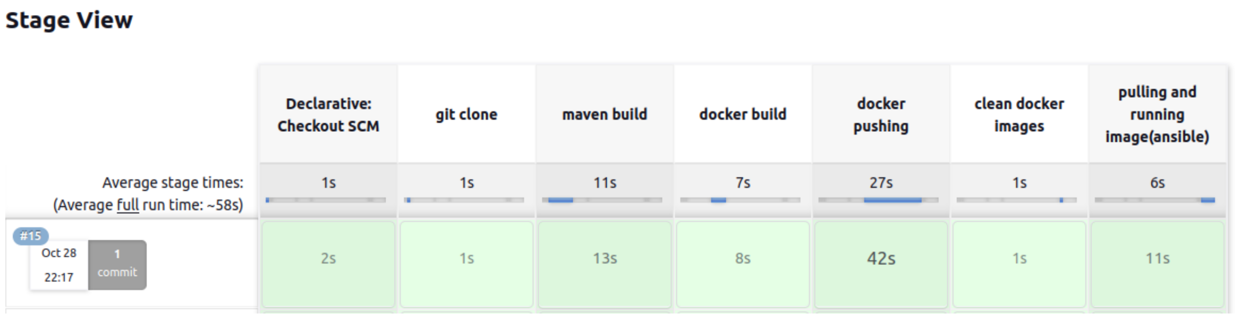
We can set the Jenkins build trigger to the GitHub hook trigger so that whenever we update something in the git it will automatically trigger the Jenkins and it will execute all these stages. So it create a new image and cleans all images and container and pushes to the docker hub and deploys container on all servers. I used ngrok to host jenkins online so that github can trigger the Jenkins when something is updated on the GitHub. We need to copy ngrok URL in the GitHub webhooks and in the Jenkins. Since we are using free version of ngrok everytime link will be updated so manually we need to change the link in the github and the jenkins.



```
Session Status                online
Account                       challapallichaithanya@gmail.com (Plan: Free)
Update                        update available (version 3.4.0, Ctrl-U to update)
Version                       3.3.4
Region                        India (in)
Latency                       25ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://8e71-103-156-19-229.ngrok-free.app -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

When we use the command “ngrok http 8080” we get this on the terminal we can see the URL we need to paste this in the Jenkins and GitHub webhook. 8080 because Jenkins is on port number 8080 so Ngork will forward that port.

In the below image, we can see the pipeline stages in Jenkins. We can check the logs of each stage. We can check errors and correct them if there is any problem at any stage. So from the start to the end, everything is automated whenever we update something on GitHub everything will be updated automatically.

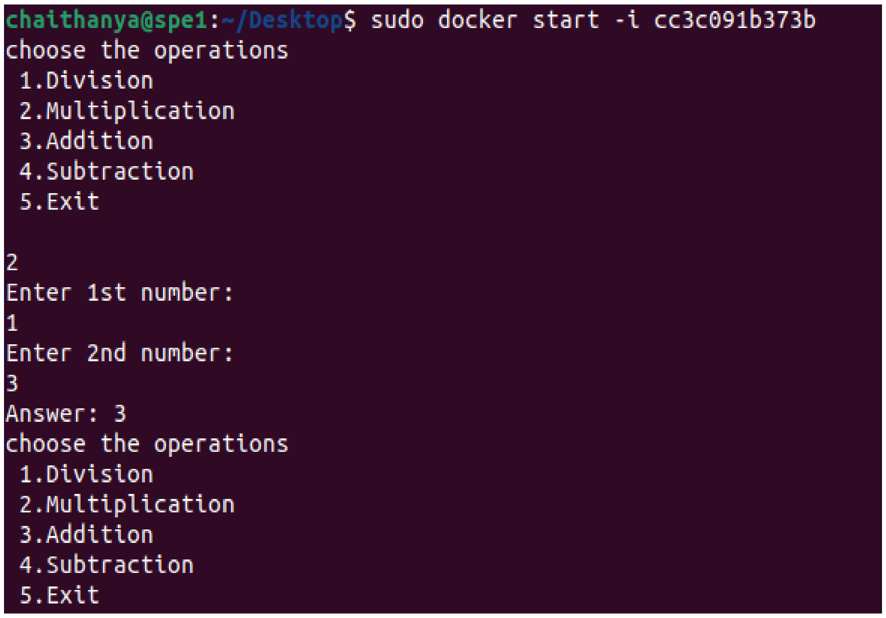


Stage View

	Declarative: Checkout SCM	git clone	maven build	docker build	docker pushing	clean docker images	pulling and running image(ansible)
Average stage times: (Average full run time: ~58s)	1s	1s	11s	7s	27s	1s	6s
#15 Oct 28 22:17 1 commit	2s	1s	13s	8s	42s	1s	11s

Running Docker Container

After all the stages we can see the docker image and container in the machine. We see the demo of the container in the image given below.



```
chaithanya@spe1:~/Desktop$ sudo docker start -i cc3c091b373b
choose the operations
 1.Division
 2.Multiplication
 3.Addition
 4.Subtraction
 5.Exit

2
Enter 1st number:
1
Enter 2nd number:
3
Answer: 3
choose the operations
 1.Division
 2.Multiplication
 3.Addition
 4.Subtraction
 5.Exit
```

I used this command “sudo docker start -i <container name/id>” -i is for interaction. Now it asks to choose an operation and then two numbers input. Then program prints the output. When input is 5 then it exits the program.