SPE MINI PROJECT

Calculator Program

Akhil Tavva
IMT2020124

Github repo link:
https://github.com/Akhil-Tavva/Calculator.git

Docker Hub link:
https://hub.docker.com/repository/docker/akhiltavva/calculator/general

Objective

The goal of the mini project is not to focus on the calculator program but to run it by using DevOps tools and workflows.In this project on DevOps parts, we learn how to build a pipeline and learn how CI/CD works. We will also create a jenkins pipeline to build,test, deploy the code. In this calculator program the operations that are possible are:

- Addition
- Subtraction
- Multiplication
- Division

DevOps Implementation

The tools we need for implementation of DevOps are:

- **IDE:** We use Intellij IDE in this project which supports the integration of Maven.
- **Maven**: This is a versatile build tool that can be seamlessly integrated into your IDE as a plugin or imported as a project dependency folder. It enables your Maven project to run in any

environment, provided the necessary dependencies can be obtained by connecting to the appropriate server for downloads.

- **Git(VCS)**: Git is used by developers which allows them to roll back any changes or to work on multiple versions of the project. Git and GitHub are different.
- **Github:** It is one of the tools for Remote VCS.
- **Jenkins:** We use CI/CD pipeline which is a feature of Jenkins. And it can also use along with GitHub, Docker etc.
- **Docker:** With the help of Docker, we can package the whole project or product we're working on into an executable container that requires little setup or configuration at the deployment locations and can be exported and imported as a single file. Similar to virtual machines, but considerably smaller, containers are very helpful to deploy instead of requiring a virtual machine to be set up at each client.
- **Docker Hub:** It is an extension of Docker where all the containers we create are stored in a remote repository, similar to the one we have for Git.
- **Ansible:** The deployment tool Ansible is the last phase in this little project. Now that the operations and development teams are working together, we also need to stop assigning blame and pointing fingers whenever a bug or crash occurs.
- **Ngrok:** Ngrok is a popular tunneling and reverse proxy service that allows you to expose a web server, service, or application running on your local machine to the public internet. It creates a secure and temporary tunnel to a publicly accessible URL that you can share with others, making your local web server or application accessible from anywhere in the world.

<u>Step by Step documentation</u>

Step1: Create a Maven Project in Intellij IDE

Create a new empty project with the Build system as Maven and language as Java. Once created the project, install maven in the command line with command sudo apt install maven.

```
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ mvn clean
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for org.example:Calculator:jar:1.0-SNAPSHOT
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)' must be unique: junit:junit:jar -> duplicate declaration of version
4.13.2 @ line 34, column 21
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----------------------< org.example:Calculator >-----------------------
[INFO] Building Calculator 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ Calculator ---
[INFO] Deleting /home/akhil/Desktop/Sem-7/Software Production Enginerring/Calculator/target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.144 s
[INFO] Finished at: 2023-11-07T14:14:06+05:30
[INFO] ------------------------------------------------------------------------
```

Executing maven clean

```
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ mvn compile
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for org.example:Calculator:jar:1.0-SNAPSHOT
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)' must be unique: junit:junit:jar -> duplicate declaration of version
4.13.2 @ line 34, column 21
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----------------------< org.example:Calculator >-----------------------
[INFO] Building Calculator 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Calculator ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Calculator ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/akhil/Desktop/Sem-7/Software Production Enginerring/Calculator/target/classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.712 s
[INFO] Finished at: 2023-11-07T14:17:43+05:30
[INFO] ------------------------------------------------------------------------
```

Executing maven compile

```
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ mvn install
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for org.example:Calculator:jar:1.0-SNAPSHOT
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)' must be unique: junit:junit:jar -> duplicate declaration of version
4.13.2 @ line 34, column 21
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----------------------< org.example:Calculator >-----------------------
[INFO] Building Calculator 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Calculator ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Calculator ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ Calculator ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/akhil/Desktop/Sem-7/Software Production Enginerring/Calculator/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ Calculator ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/akhil/Desktop/Sem-7/Software Production Enginerring/Calculator/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ Calculator ---
[INFO] Surefire report directory: /home/akhil/Desktop/Sem-7/Software Production Enginerring/Calculator/target/surefire-reports
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.12.4/surefire-junit4-2.12.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.12.4/surefire-junit4-2.12.4.pom (2.
4 kB at 4.3 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-providers/2.12.4/surefire-providers-2.12.4
```

Executing maven install

Now that we compiled the code, we get the JAR file in the target folder of the project. To make a difference between the JAR that we are generating we add some code in the pom.xml file(which will be available from github) which now creates a new JAR file.

The contents of pom.xml file are

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>Calculator</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.14.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.14.0</version>
```

Run the jar file to get the output from the Main class.

Step2: Initialize Git and Connect to Github

If git is not installed, install it by using sudo apt install git. Once done, then initiate it by command git init.

To add files into the current repository the command we use is: git add .To see the files that have been changed, we use this command: git status.

```
Everything up-to-date
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ git add .
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/main/java/org/example/Main.java

akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$
alculator > src > main > java > org > example > Main > main
```

Now to commit the changes, we will use the following command:
git commit -m <Your commit message here>

```
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ git commit -m "Final Code"
[master 4da6d80] Final Code
 1 file changed, 9 insertions(+), 9 deletions(-)
```

Now in order to push these changes into GitHub we need to create a personal access token. After creating it, copy the token such that we need to use it whenever to push. Now we have to add remote origin by using the command:
git remote add origin <github_repo_url>

To push into github, we use the following command:
git push <remote directory> <branch name>

```
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ git push
Username for 'https://github.com': takhil2020@gmail.com
Password for 'https://takhil2020@gmail.com@github.com':
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (8/8), 609 bytes | 609.00 KiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Akhil-Tavva/Calculator.git
   b7ae915..4da6d80  master -> master
```

In this command we have to enter username and Personal Access Token.

Step3: Install Jenkins and Create Pipeline

To install jenkins, we will follow the guide mentioned in reference and the basic command will be:

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |
sudo apt-key
add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >

/etc/apt/sources.list.d/jenkins.list'
sudo apt install ca-certificates
sudo apt update
sudo apt install jenkins

Once this is done, then Go to https://localhost:8080 and install the necessary packages. Now we create a new Jenkins pipeline and install the needed plugins that need to be present for this project such as Anisble, Git plugin, Junit, Github etc.

The Jenkins Pipeline has 6 stages and they are listed below:

Stage View

	Stage 1: Git Clone	Stage 2: Maven Build	Stage 3: Build Docker Image	Stage 4: Push docker image to hub	Stage 5: Clean docker images	Step 6: Ansible Deployment
Average stage times: (Average full run time: ~39s)	1s	2s	4s	24s	535ms	287ms
#27 Nov 06 21:56 1 commit	1s	2s	4s	25s	590ms	625ms

The pipeline script of this jenkins pipeline is:

Dashboard > Mini Project > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

```
}
agent any
stages{
    stage('Stage 1: Git Clone'){
        steps{
            git branch: 'master',
            url:'https://github.com/Akhil-Tavva/Calculator.git'
        }
    }
    stage('Stage 2: Maven Build'){
        steps{
            sh 'mvn clean install'
        }
    }
    stage('Stage 3: Build Docker Image'){
        steps{
            script{
                docker_image = docker.build "akhiltavva/calculator:latest"
            }
        }
    }
    stage('Stage 4: Push docker image to hub'){
        steps{
            script{
                docker.withRegistry('', 'DockerHubCred'){
                    docker_image.push()
                }
            }
        }
    }
    stage('Stage 5: Clean docker images'){
        steps{
            script{
                sh 'docker container prune -f'
                sh 'docker image prune -f'
            }
        }
    }
    stage('Step 6: Ansible Deployment'){
        steps{
            ansiblePlaybook becomeUser: null, colorized: true,
            credentialsId: 'localhost', disableHostKeyChecking: true, installation: 'Ansible',
            inventory: 'Deployment/inventory', playbook: 'Deployment/deploy.yml', sudoUser: null
        }
    }
}
```

Step4: Install Docker & create Docker Image

To install docker we follow these commands:
sudo apt install curl
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh
sudo -E sh -c docker version

We will create a docker container for our project. For this, we need a Dockerfile.

```
FROM openjdk:11
COPY ./target/Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java","-cp","Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar","org.example.Main"]
```

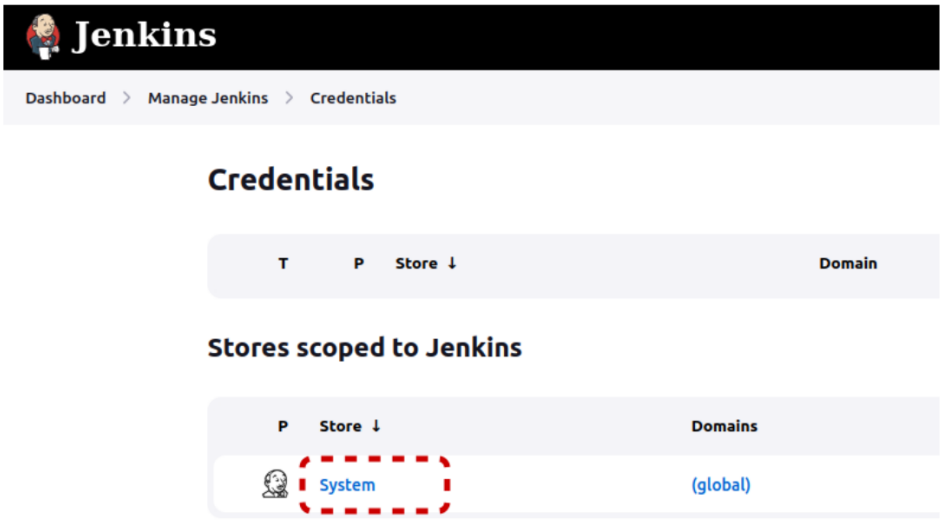
The above code is to generate a Docker container with OpenJDK version 11 which acts like a JVM on top of which we are going to run . The stage 3 of pipeline script builds the docker image and the below image shows that. We can verify by this cmd.

```
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Engineering/Calculator$ sudo docker images
[sudo] password for akhil:
REPOSITORY              TAG       IMAGE ID       CREATED        SIZE
akhiltavva/calculator   latest    425fd927ae03   18 hours ago   656MB
hello-world             latest    9c7a54a9a43c   6 months ago   13.3kB
```

Now we can see from jenkins that it is successfully created or not(docker images).

Step5: Pushing Docker image to Docker Hub

To push the Docker image to Docker Hub, begin by storing your Docker Hub credentials in Jenkins. To do this, go to Dashboard > Manage Jenkins > Credentials. Inside, choose 'System' under 'Stores scoped to Jenkins'.



System option in Credentials

In the system, click on Global Credentials and add credentials. In the New Credentials form, we are going to fill the following details:

KIND: Username with password
SCOPE: Global
USERNAME: <Your Docker Hub Username>
PASSWORD: <Your Docker Hub Password>
ID: DockerHubCred
DESCRIPTION: Docker Hub Credentials

With the credentials successfully created, the 4th stage of pipeline script will run successfully.

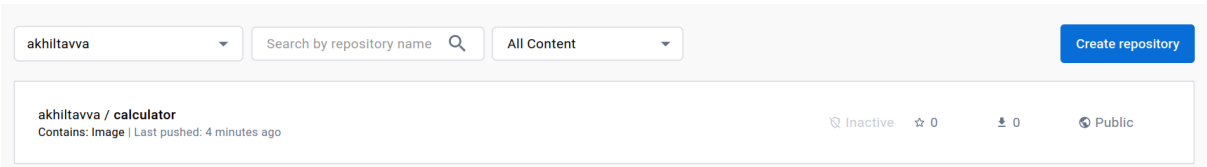
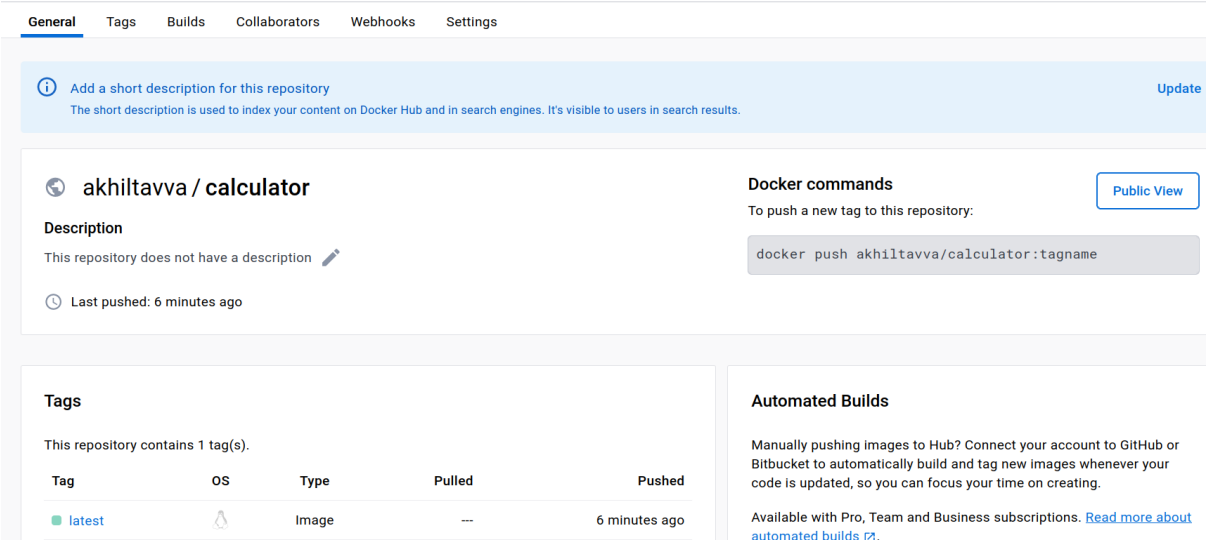


Image pushed to Docker Hub



Details about the Docker image that was pushed

Step6: Cleaning up Docker images

Instead of overwriting the previous Docker images, a new one is produced each time the pipeline is executed. The ones that are the newest will have tags, while the others won't. We will therefore add a pipeline stage to halt any running containers and delete all pictures that match the name of the current project (except from the one that has the tag) in order to guarantee that we don't use up all of our limited local secondary storage. Since the image has been successfully published to Docker hub and is no longer needed on our local system, it is OK to proceed in this manner. We can always pull from Docker Hub if necessary.

To remove the images except the most recent one, we execute the stage 5 code. After running sudo docker images in the terminal, then you will see only one image.

```
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ sudo docker images
[sudo] password for akhil:
REPOSITORY              TAG       IMAGE ID       CREATED        SIZE
akhiltavva/calculator   latest    425fd927ae03   18 hours ago   656MB
hello-world             latest    9c7a54a9a43c   6 months ago   13.3kB
```

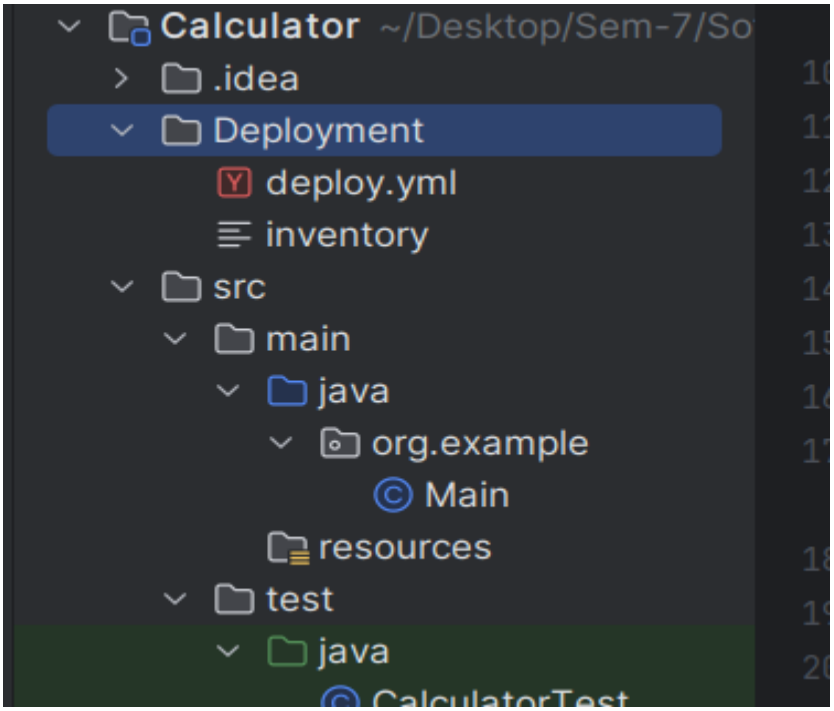
Step7: Pulling Docker images using Ansible

To install Ansible in the system, run the following commands:

sudo apt-add-repository ppa:ansible/ansible
sudo apt update
sudo apt install ansible

After installing ansible, Storing the instructions for deploying the Docker image is the first step towards deployment. We want to keep this in the project directory itself, under the Deployment subdirectory. There will be two files in the folder: deploy.yml and inventory.

The list of clients who will implement the project will be included in the inventory file. The image that is being retrieved and deployed has specs that are contained in the deploy yaml file.



In the inventory file, since we are deploying the project in user and pc, we are going to put the username into the file.

```
localhost ansible_user = local
```

The deploy.yml file we add the code and the file is given below:

```
ain.java    CalculatorTest.java    inventory    deploy.yml ×
---
- name: Pull Docker Image of Calculator
  hosts: all
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Pull image
      docker_image:
        name: akhiltavva/calculator:latest
        source: pull
    - name: Start docker service
      service:
        name: docker
        state: started
    - name: Running container
      shell: docker run -it -d --name Calculator akhiltavva/calculator
```

We have to ensure that the <username>/<image name>:<tag> is the same as the one that you pushed to docker hub. Also ensure that the path of python3 is correct or not. The stage 6 is the code of this step in pipeline script(See the script).

Here, since we are deploying to our own system, we take care to provide credentialsId: 'localhost'. If not, an error about permission denied will appear. Next, in the same manner that we entered Docker Hub credentials, we must enter our localhost login and password in Jenkin's Global credentials. The Linux login credentials for the localhost user will be the username and password.

To see the list of containers,we can use the following command:

Sudo docker container ls -all

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ sudo docker container ls --all
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS     NAMES
d7b04eb723dc   akhiltavva/calculator    "java -cp Calculator…"   12 seconds ago   Up 11 seconds             Calculator
akhil@akhil-thinkpade14:~/Desktop/Sem-7/Software Production Enginerring/Calculator$ sudo docker start -a -i Calculator
```

We can see that the container status is Exited. To run the container, we use the following command:

sudo docker start -a -i Calculator

```
Operation Choices:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter a choice:
1
Enter the first number: 1
Enter the second number: 2
Result: 3
Operation Choices:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter a choice:
2
Enter the first number: 2
Enter the second number: 1
Result: 1
Operation Choices:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter a choice:
3
Enter the first number: 2
Enter the second number: 3
Result: 6
Operation Choices:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter a choice:
4
Enter the first number: 8
Enter the second number: 2
```

This is the output main.java of the mini project.

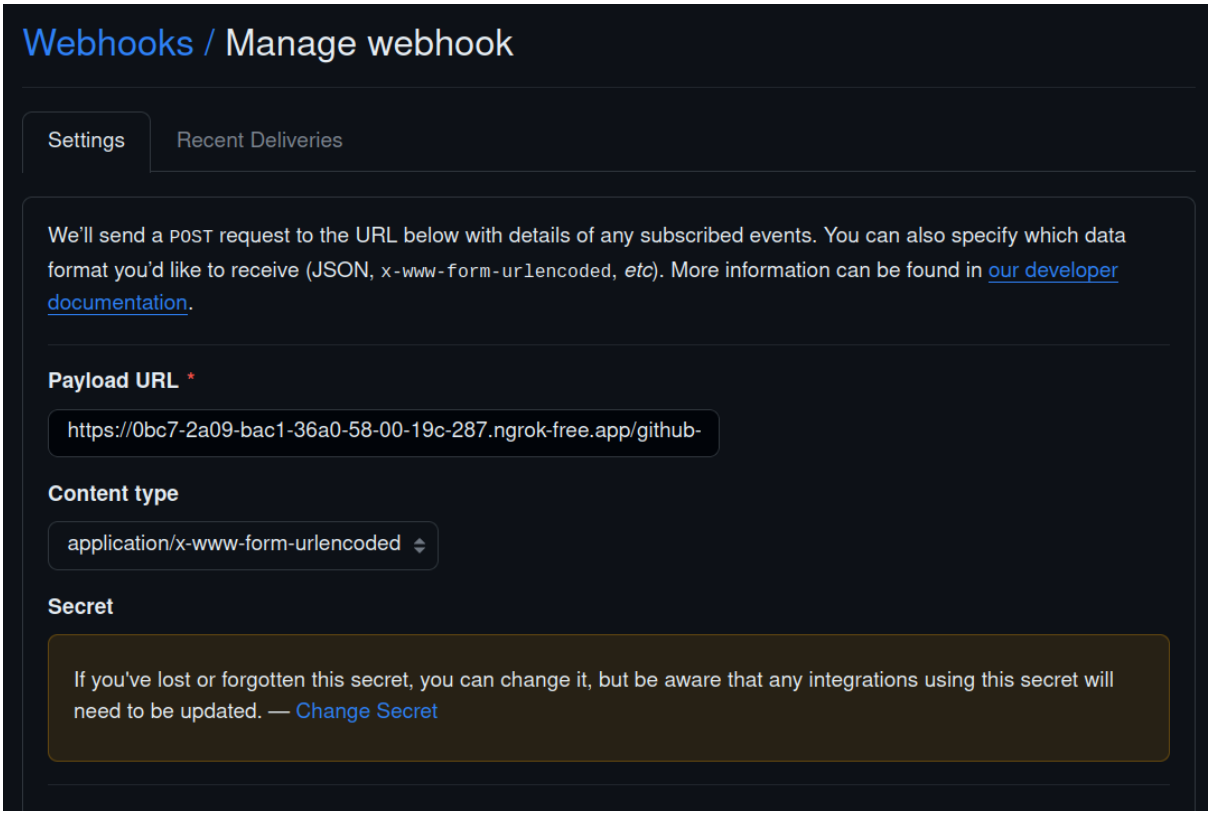
Step8: Setting up GitSCM polling in Jenkins

Sign up for ngrok at https://ngrok.com.To install ngrok follow the commands below:
Download ngrok package from their website
Extract the package using the following command:
sudo tar xvzf . /ngrok-stable-linux-amd64.tgz -C /usr/local/bin

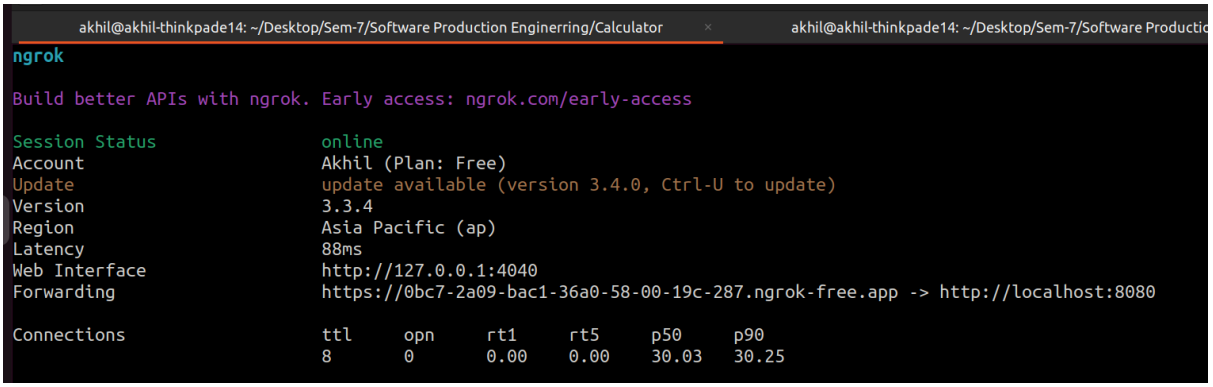
Then copy the authtoken from your ngrok profile dashboard

Add the authtoken using the following command: ngrok authtoken <token>

We then go to our repository in Github and open settings in the repository. After that go to Webhooks and do the same as below:



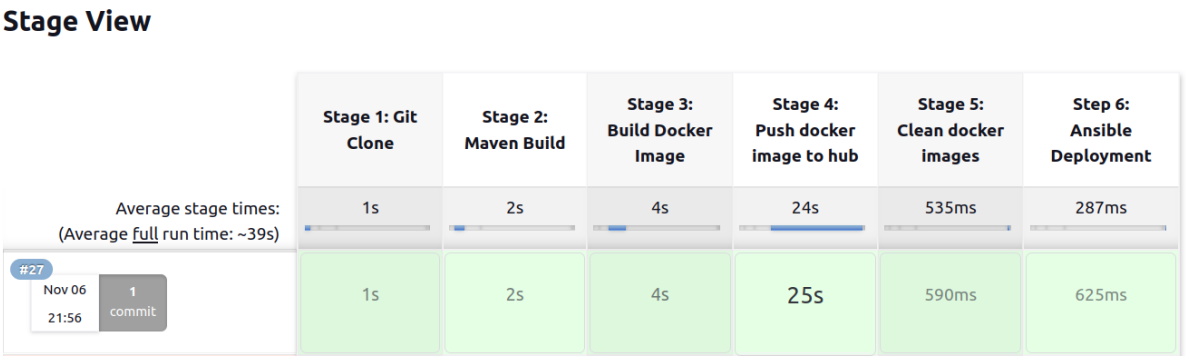
Now we have created a webhook.



Ngrok interface after running it

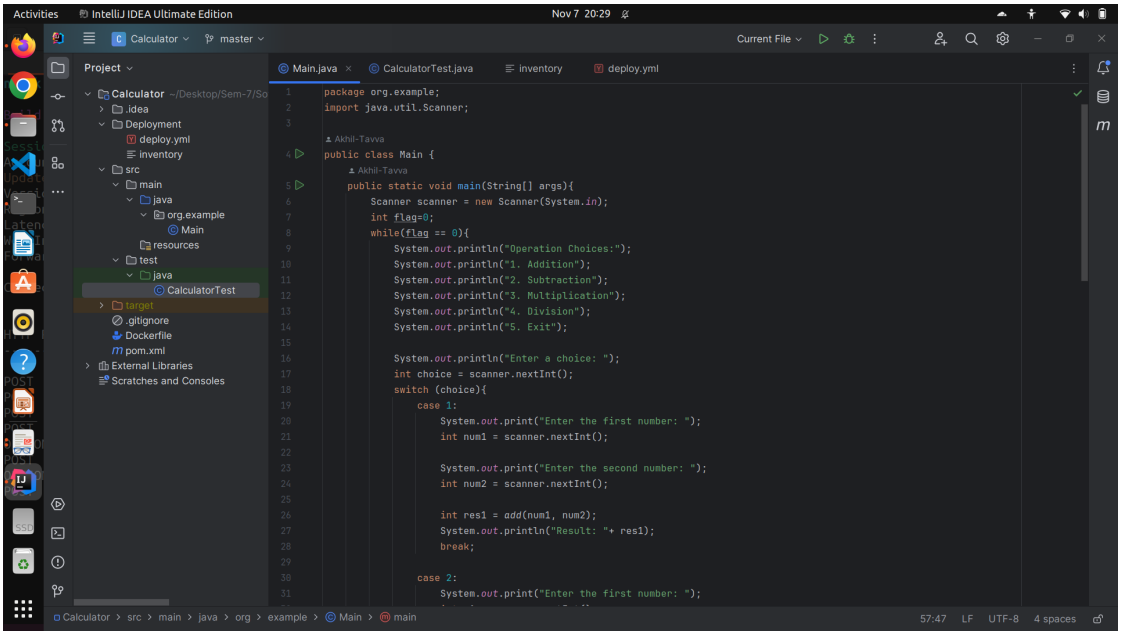
Now on the jenkins side of things, we go to our pipeline > configure and make sure that to tick GitScm polling build trigger and after that change the jenkins url. Now add GitHub API URL to accept Webhooks.

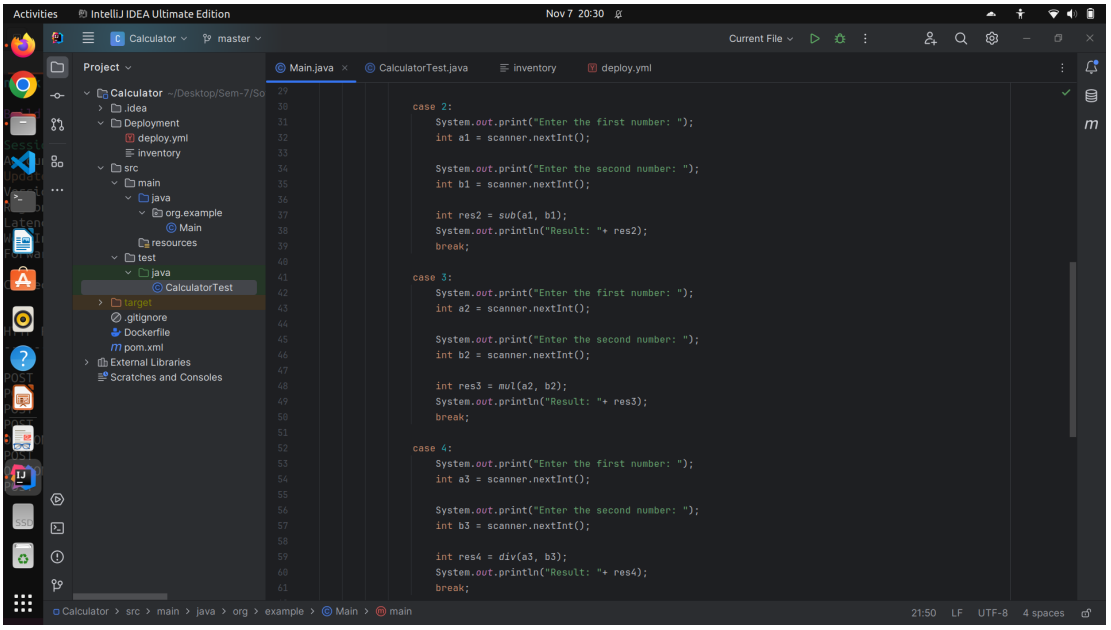
Now save the settings, and then when we push the changes into GitHub, we can see that Jenkins will build automatically and go to pipeline stages. Then we will get the pipeline output like this:



Step9: Implementation of Calculator program

Code for calculator program:




Activities
IntelliJ IDEA Ultimate Edition
Nov 7 20:30
Calculator
master
Current File
Project
Main.java
CalculatorTest.java
inventory
deploy.yml
Calculator ~/Desktop/Sem-7/So
.idea
Deployment
deploy.yml
inventory
src
main
java
org.example
Main
resources
test
java
CalculatorTest
target
.gitignore
Dockerfile
pom.xml
External Libraries
Scratches and Consoles
case 2:
System.out.print("Enter the first number: ");
int a1 = scanner.nextInt();
System.out.print("Enter the second number: ");
int b1 = scanner.nextInt();
int res2 = sub(a1, b1);
System.out.println("Result: "+ res2);
break;
case 3:
System.out.print("Enter the first number: ");
int a2 = scanner.nextInt();
System.out.print("Enter the second number: ");
int b2 = scanner.nextInt();
int res3 = mul(a2, b2);
System.out.println("Result: "+ res3);
break;
case 4:
System.out.print("Enter the first number: ");
int a3 = scanner.nextInt();
System.out.print("Enter the second number: ");
int b3 = scanner.nextInt();
int res4 = div(a3, b3);
System.out.println("Result: "+ res4);
break;
Calculator > src > main > java > org > example > Main > main
21:50
LF
UTF-8
4 spaces


Main.java
CalculatorTest.java
inventory
deploy.yml
60 System.out.println("Result: "+ res4);
61 break;
63 case 5:
64 flag = 1;
65 break;
67 default:
68 System.out.println("Invalid Input");
69 flag=1;
70 break;
71 }
72 }
73 }
74
3 usages Akhil-Tavva
75 public static int add(int a, int b) { return a+b; }
78
2 usages Akhil-Tavva
79 public static int sub(int a, int b) { return a-b; }
82
2 usages Akhil-Tavva
83 public static int mul(int a, int b) { return a*b; }
86
2 usages Akhil-Tavva
87 public static int div(int a, int b) { return a/b; }
90
91 }

Code for Tester of main.java

```
Akhil-Tavva
@Test
public void test1(){
    int a = 2;
    int b = 1;
    int expectedResult = 3;
    Assert.assertEquals(expectedResult, calculator.add(a, b));
}

Akhil-Tavva
@Test
public void test5(){
    int a = 2;
    int b = 1;
    int expectedResult = 5;
    Assert.assertNotEquals(expectedResult, calculator.add(a, b));
}

Akhil-Tavva
@Test
public void test2(){
    int a = 2;
    int b = 1;
    int expectedResult = 2;
    Assert.assertNotEquals(expectedResult, calculator.sub(a, b));
}

Akhil-Tavva
@Test
public void test3(){
    int a = 2;
    int b = 1;
    int expectedResult = 2;
```