International Institute of Information Technology
Bangalore

Software Production Engineering
CS 816

Mini Project Report

Karanjit Saha(IMT2020003)

Introduction

I have developed a simple calculator application which performs the following tasks:

1. Addition of 2 numbers
2. Subtraction of 2 numbers
3. Multiplication of 2 numbers
4. Division of 2 numbers

In the development of the Java project, the following tools were used:

1. Git: A distributed version control system for tracking changes to the code base and collaborating with other developers.
2. JUnit: A unit testing framework for Java for writing and running tests for the code.
3. Jenkins: A continuous integration and continuous delivery (CI/CD) tool for automating the building, testing, and deployment of the code.
4. Docker: A containerization platform for packaging the code and its dependencies into a single unit called a container for deployment and running Java applications.
5. Maven: A build automation tool for Java for automating the building, testing, and packaging of the code.
6. Ansible: A configuration management tool for automating the configuration of the infrastructure for Java applications.
7. GitHub WebHooks: A service for triggering events in external systems when certain events occur in the GitHub repository.
8. ngrok: A tunneling service for exposing local servers to the public internet over a secure tunnel for debugging and testing web applications.

Implementation

1. GitHub Repository
2. DockerHub Repository

About the code

Main.java

My calculator code has four basic operations which are:

1. int add(int num1, int num2): This function performs the addition of two integers.
2. int subtract(int num1, int num2): This function performs the subtraction of two integers.
3. int multiply(int num1, int num2): This function performs the multiplication of two integers.
4. int division(int num1, int num2): This function performs the division of num1 by num2.

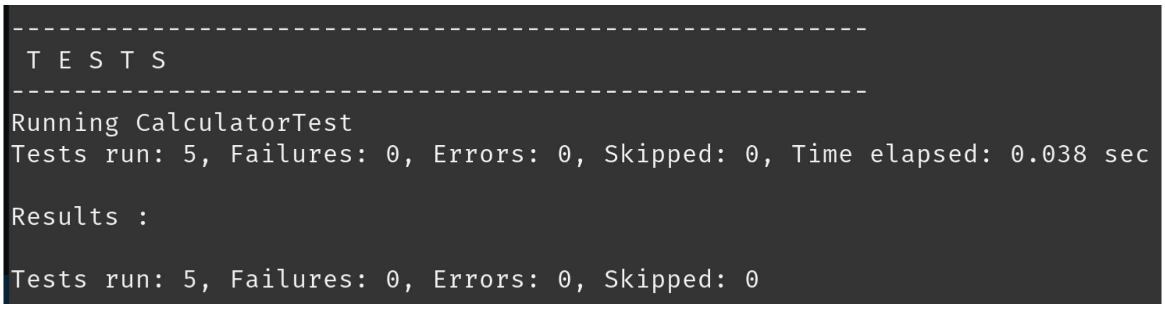
There is also a fifth option to exit the program.

CalculatorTest.java

In this file, I have tested the various functions created in Main.java. I have covered the following test cases:

1. public void test_add(): This is to check whether the addition function of the calculator gives an expected result.
2. public void test_add1(): This is to check whether the addition function of the calculator gives an expected result.
3. public void test_subtract(): This is to check whether the subtraction function of the calculator gives an expected result.
4. public void test_multiply(): This is to check whether the multiplication function of the calculator gives an expected result.
5. public void test_division(): This is to check whether the division function of the calculator gives an expected result.

On running "mvn clean install" the following output shows that all the test cases are passed.

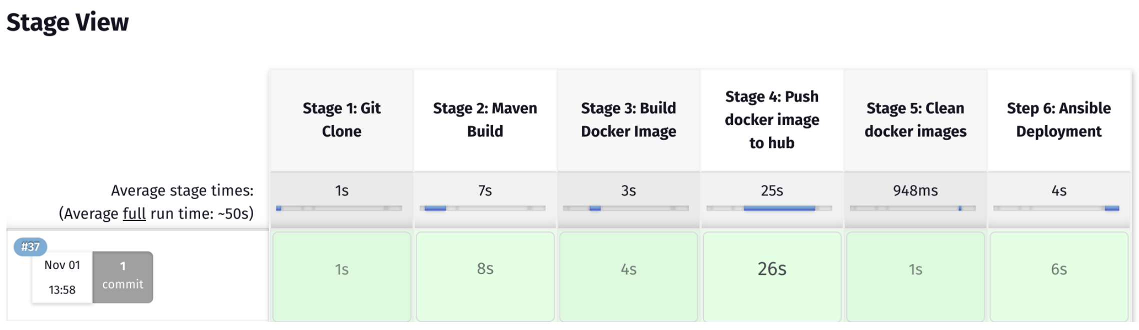


```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running CalculatorTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.038 sec

Results :

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
```

Figure 1: Testcases for Calculator project

Pipeline Stages and Tools Used

```
pipeline {
  environment {
    docker_image = ''
  }
  agent any

  stages {
    stage('Stage 1: Git Clone') {
      steps {
        git branch: 'master', url: 'https://github.com/KaranjitSaha/IMT2020003_Calculator.git'
      }
    }

    stage('Stage 2: Maven Build') {
      steps {
        sh 'mvn clean install'
      }
    }

    stage('Stage 3: Build Docker Image') {
      steps {
        script {
          docker_image = docker.build('karanjit708/calculator:latest')
        }
      }
    }

    stage('Stage 4: Push docker image to hub') {
      steps {
        script {
          docker.withRegistry('', 'DockerCred') {
            docker_image.push()
          }
        }
      }
    }

    stage('Stage 5: Clean docker images') {
      steps {
        script {
          sh 'docker stop calculator'
          sh 'docker rm calculator'
          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
            )
        }
    }
  }
}
```



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: ~50s)	1s	7s	3s	25s	948ms	4s
#37 Nov 01 13:58 1 commit	1s	8s	4s	26s	1s	6s

Figure 2: Jenkins Pipeline for Calculator project

Following is the description of each of the stages in detail:

1. **Stage 1 - Git Clone** :In this stage, the project code is cloned from the GitHub Repository. This is to ensure that any new changes pushed to the repository are cloned before the build starts. Using Git in the project enables version control and collaboration during development.

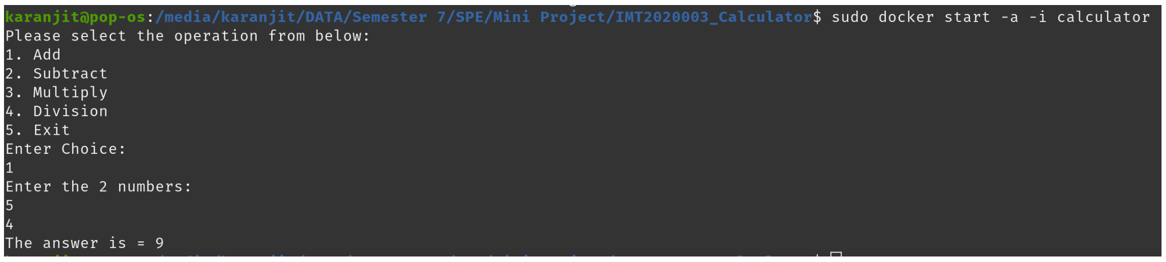
2. **Stage 2 - Maven Build**: This stage involves the use of Maven to build the project and run test cases. Maven is used to clean, compile, and install the dependencies required for the project. With the help of JUnit, it can also perform testing of the code. This creates a packaged JAR file of the project along with the dependencies.

3. **Stage 3 - Build Docker Image**: In this stage, a Docker image of the application is created using the built JAR file from the previous stage. Docker builds a container image using the Dockerfile which copies

the compiled JAR file into it. This container contains the application and its dependencies.

4. **Stage 4 - Push Docker Image to DockerHub**: The built Docker image is pushed Docker Hub container registry. Docker Hub is used to store the built Docker image remotely. This allows other stages or systems to access the image.
5. **Stage 5 - Clean Docker Image**: In this stage, unused Docker images are cleaned up from the local system. This removes dangling/unused images and containers with the same container name freeing up disk space and avoiding errors when running containers from old images.
6. **Stage 6 - Ansible Deployment**: Ansible playbook to deploy the application by pulling the Docker image and running it. Ansible executes the playbook tasks (which are mentioned in the deploy.yml file) to pull the latest image from Docker Hub, start the Docker service, and create the calculator container (on the localhost device in this case).

Running the deployed container

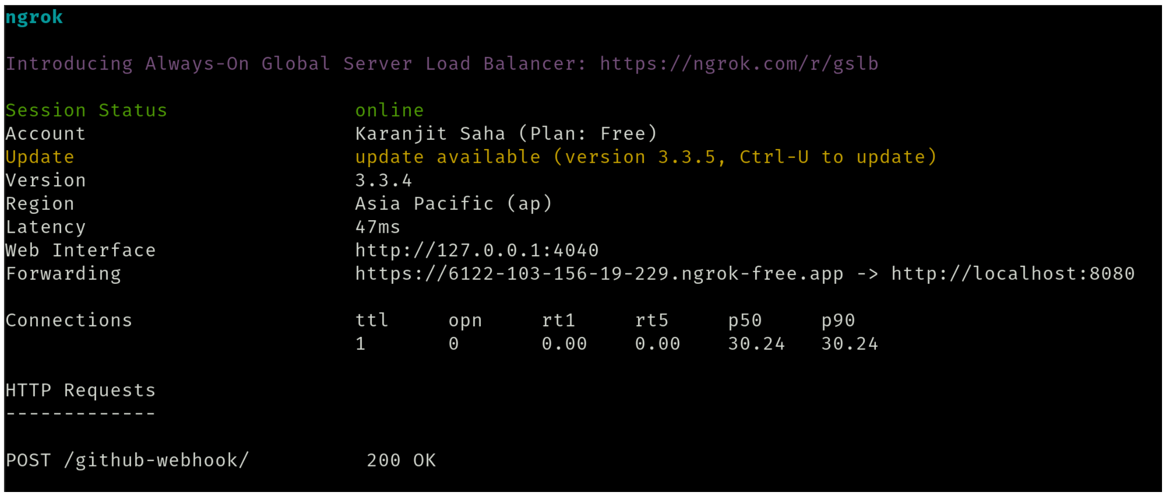


```
karanjit@pop-os:/media/karanjit/DATA/Semester 7/SPE/Mini Project/IMT2020003_Calculator$ sudo docker start -a -i calculator
Please select the operation from below:
1. Add
2. Subtract
3. Multiply
4. Division
5. Exit
Enter Choice:
1
Enter the 2 numbers:
5
4
The answer is = 9
```

Figure 3: Running the container.

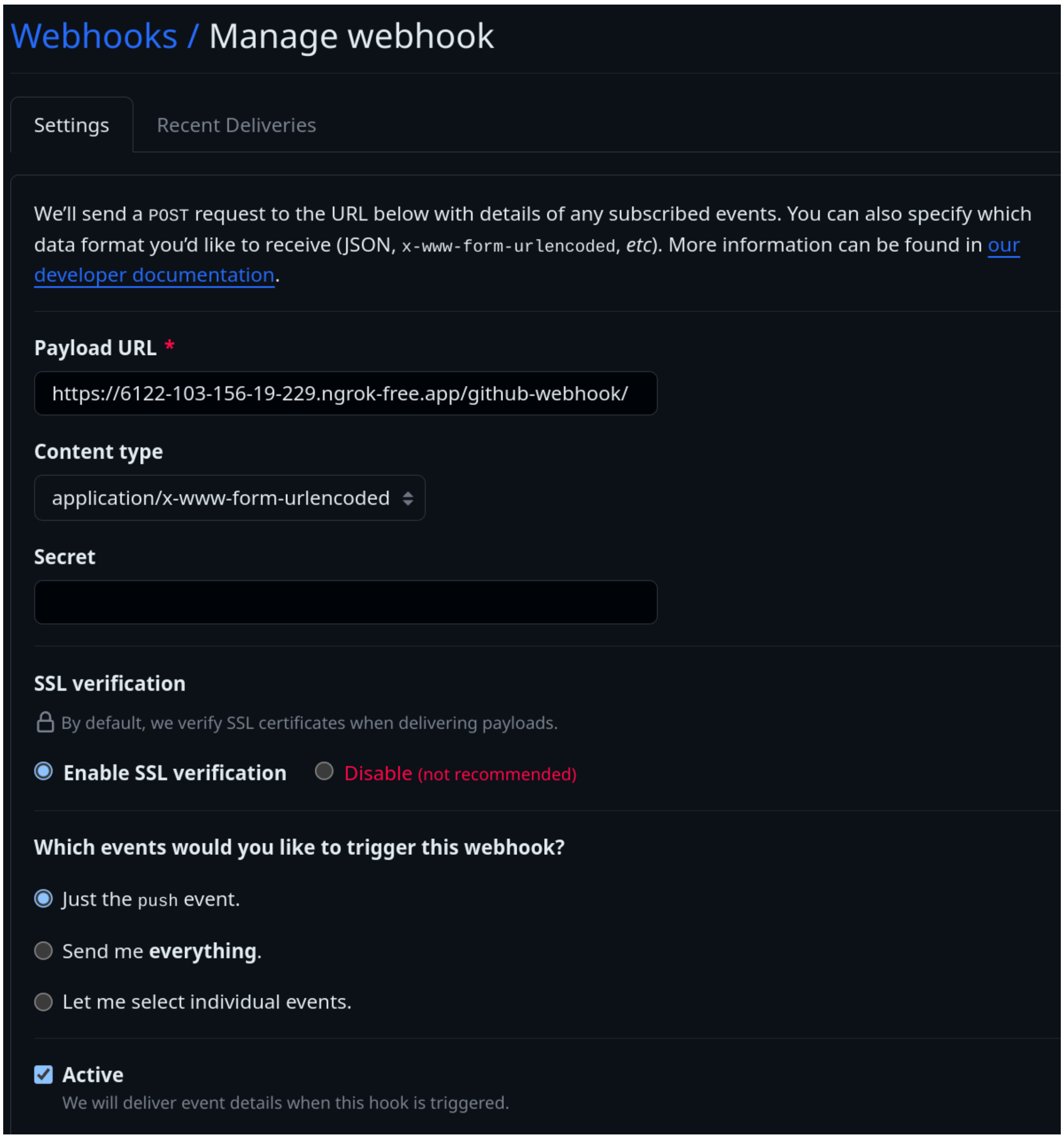
Setting up WebHook using ngrok

Here, the ngrok tool is used to expose a local development server to the Internet. The forwarding URL changes every time it is run. This URL is used to create a webhook on GitHub for our repository. The Github repository link is added to the pipeline configuration settings in Jenkins and GitHub Remote SCM is enabled for the same.



```
ngrok

Introducing Always-On Global Server Load Balancer: https://ngrok.com/r/gslb

Session Status                online
Account                       Karanjit Saha (Plan: Free)
Update                        update available (version 3.3.5, Ctrl-U to update)
Version                       3.3.4
Region                        Asia Pacific (ap)
Latency                       47ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://6122-103-156-19-229.ngrok-free.app -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              1       0       0.00    0.00    30.24   30.24

HTTP Requests
-------------

POST /github-webhook/          200 OK
```

Figure 4: Creating a ngrok server at port 8080.


Webhooks / Manage webhook

Settings | Recent Deliveries

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

https://6122-103-156-19-229.ngrok-free.app/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

[✓ Enable SSL verification] Disable (not recommended)

Which events would you like to trigger this webhook?

[✓ Just the `push` event.]

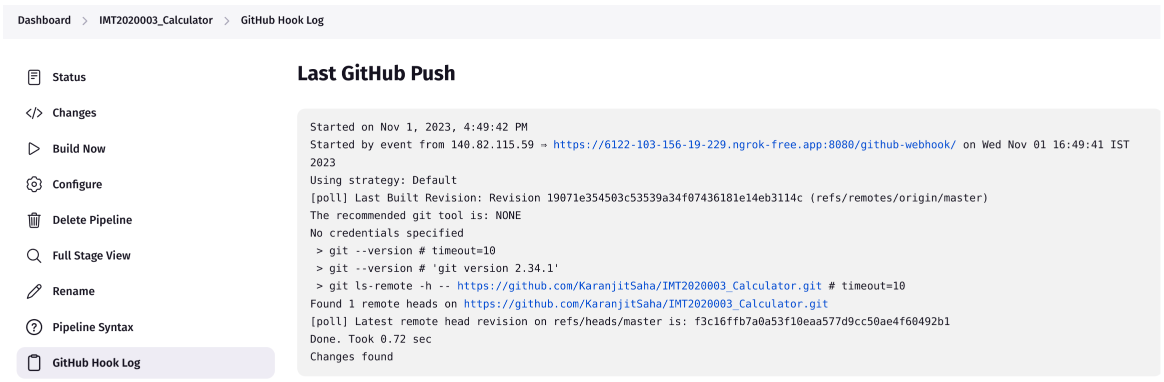
Send me **everything**.

Let me select individual events.

[✓] **Active**

We will deliver event details when this hook is triggered.

Figure 5: Addinig a WebHook in GitHub repository.



Last GitHub Push

```
Started on Nov 1, 2023, 4:49:42 PM
Started by event from 140.82.115.59 ⇒ https://6122-103-156-19-229.ngrok-free.app:8080/github-webhook/ on Wed Nov 01 16:49:41 IST
2023
Using strategy: Default
[poll] Last Built Revision: Revision 19071e354503c53539a34f07436181e14eb3114c (refs/remotes/origin/master)
The recommended git tool is: NONE
No credentials specified
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git ls-remote -h -- https://github.com/KaranjitSaha/IMT2020003_Calculator.git # timeout=10
Found 1 remote heads on https://github.com/KaranjitSaha/IMT2020003_Calculator.git
[poll] Latest remote head revision on refs/heads/master is: f3c16ffb7a0a53f10eaa577d9cc50ae4f60492b1
Done. Took 0.72 sec
Changes found
```

Figure 6: GitHub Hook Log showing.