# International Institute of Information Technology, Banglore

Software Production Engineering Mini Project Report

Scientific Calculator

Submitted by

Mohit Gupta (MT2024049)

In the guidance of Prof. B. Thangaraju

## Problem Statement:

The goal of the project is to create a Scientific Calculator with the following user menu driven options :

1. Square Root Function - ($\sqrt{x}$)

2. Factorial Function - ($x!$)

3. Natural Logarithm (base $e$) - ($\ln x$)

4. Power Function - ($ab$)

## Abstract:

The Scientific Calculator project is a software application designed to carry out a range of mathematical and scientific functions, including square roots, logarithms, powers, and factorials. The project utilizes DevOps methodologies to optimize the development, testing, and deployment processes. Through the use of continuous integration and deployment (CI/CD) pipelines, automated testing, and containerization, the project ensures high reliability and scalability. It incorporates DevOps tools like GitHub Actions, Docker, Jenkins, and Ansible to showcase modern software engineering techniques that improve automation, maintainability, and team collaboration.

This Scientific Calculator is a compact application that performs a variety of mathematical and scientific calculations. Developed with contemporary software development practices, it integrates DevOps tools to streamline automation, testing, and deployment.

## DevOps Introduction:

What is DevOps: DevOps is a software development approach that unites development and operations teams, enabling them to collaborate throughout the entire application lifecycle. The primary objective of DevOps is to accelerate the delivery of applications and services.

Why DevOps: DevOps enhances software delivery by automating and optimizing workflows, enabling businesses to deliver software more quickly and effectively. It also fosters improved collaboration between teams.

Benefits of DevOps:

- Faster delivery: DevOps allows businesses to release software updates more frequently and rapidly.
- Improved quality: By reducing errors, DevOps contributes to enhanced product quality.
- Enhanced collaboration: DevOps encourages breaking down silos between development and operations teams, promoting better teamwork.
- Increased security: DevOps integrates security practices into the development lifecycle, improving overall security.
- Scalability: DevOps supports the management of scalable solutions through automation and optimized workflows.
- Better customer experience: By delivering high-quality software, DevOps ensures that customer needs are met effectively.
- Cost savings: DevOps reduces the costs associated with slow and inefficient software delivery processes.

## Tools and Technologies Used:

Programming Language: Java

Version Control: Git & GitHub

Build Automation: Maven

CI/CD Pipeline: GitHub Actions / Jenkins

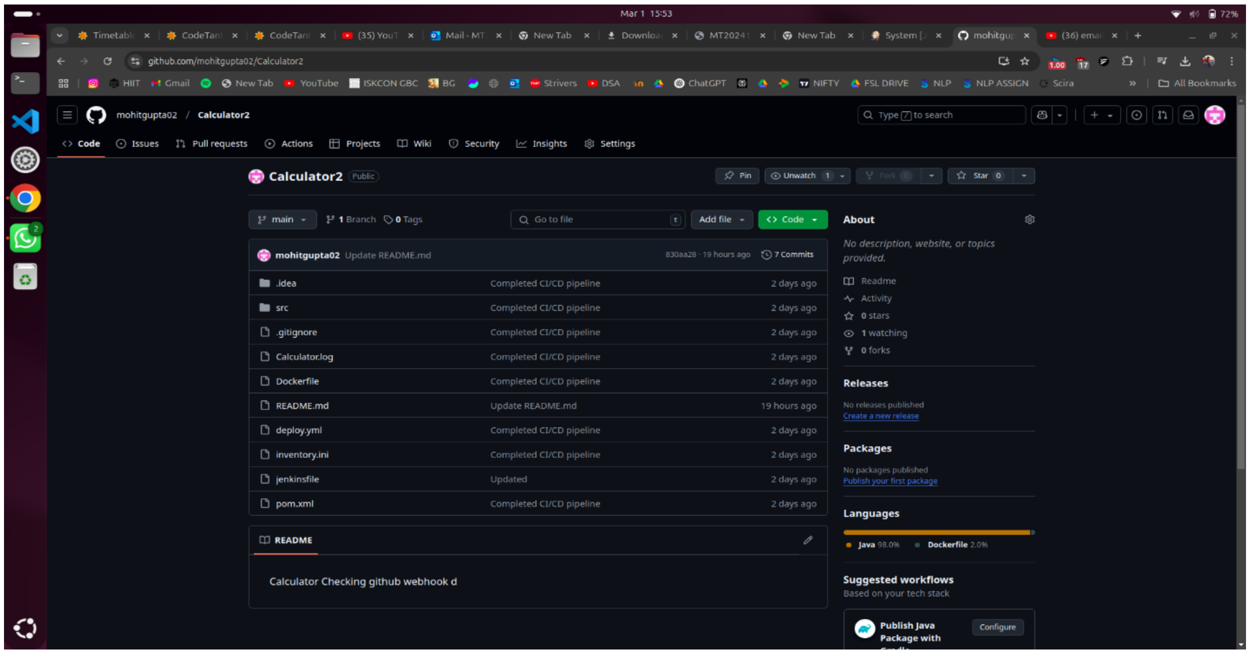
Containerization: Docker

## Source Code Management

**SCM (Source Code Management)** is a system used to track and manage changes made to a source code repository. It monitors modifications to the codebase and assists in resolving conflicts when merging updates from multiple contributors. SCM is often used interchangeably with Version Control.

The project is initially created on a local machine (local repository), and later pushed to a remote repository on platforms like GitHub. To push the local repository to the remote repository, a series of commands are executed, as outlined below:

- git init — Initializes a new Git repository in the local project directory.
- git status — Shows the current status of changes in the working directory.
- git add — Stages the changes to be committed.
- git commit -m “Commit Message” — Commits the staged changes with a descriptive message.
- git remote add origin https://github.com/mohitgupta02/Calculator2.git — Links the local repository to the remote GitHub repository.
- git push -u origin main — Pushes the committed changes to the remote repository's master branch.



mohitgupta02 / Calculator2

Calculator2 Public

main · 1 Branch · 0 Tags

mohitgupta02 Update README.md · 830aa28 · 19 hours ago · 7 Commits

| | | |
|---|---|---|
| .idea | Completed CI/CD pipeline | 2 days ago |
| src | Completed CI/CD pipeline | 2 days ago |
| .gitignore | Completed CI/CD pipeline | 2 days ago |
| Calculator.log | Completed CI/CD pipeline | 2 days ago |
| Dockerfile | Completed CI/CD pipeline | 2 days ago |
| README.md | Update README.md | 19 hours ago |
| deploy.yml | Completed CI/CD pipeline | 2 days ago |
| inventory.ini | Completed CI/CD pipeline | 2 days ago |
| jenkinsfile | Updated | 2 days ago |
| pom.xml | Completed CI/CD pipeline | 2 days ago |

README

Calculator Checking github webhook d

About

No description, website, or topics provided.

Readme · Activity · 0 stars · 1 watching · 0 forks

Releases: No releases published

Packages: No packages published

Languages: Java 98.0% · Dockerfile 2.0%

Source Code Management

## Docker:

Docker is a tool for containerization that allows packaging software into a standardized unit for development, shipment, and deployment. It leverages kernel-level virtualization to create isolated environments known as containers. In this case, Docker is used to generate an image that runs the JAR file produced by the Maven build process. This image can then be uploaded to Docker Hub, where other machines can pull the image and instantiate containers to run the JAR file.

To build a Docker image, you can use the following command:

```
docker build -t <USERNAME>/<IMAGE_NAME>:<TAG>
```

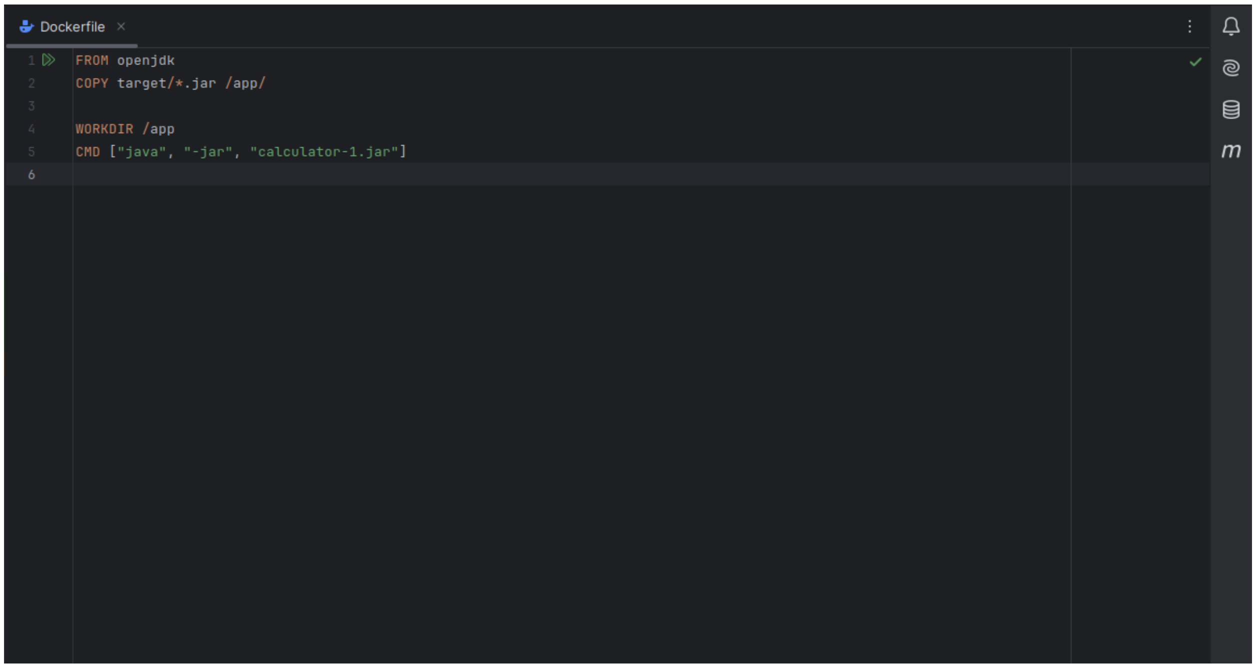
To push the Docker image to Docker Hub, use:

```
docker push <USERNAME>/<REPOSITORY_NAME>:<TAG>
```

The image is pushed from Jenkins after the build process and pulled during the Ansible job. You can run the Docker image with this command:

```
docker run -i -t <IMAGE_NAME>
```

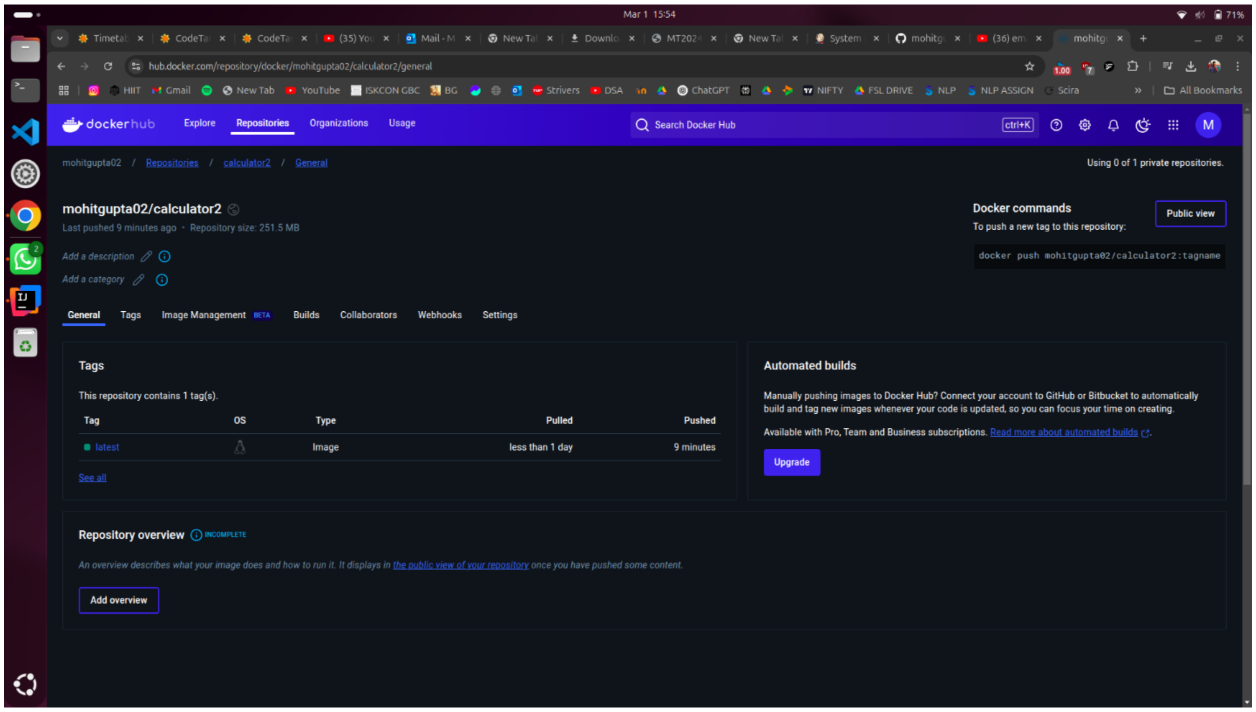
Once the Docker image is pushed to Docker Hub, you can verify it by checking your Docker Hub profile.



```
FROM openjdk
COPY target/*.jar /app/

WORKDIR /app
CMD ["java", "-jar", "calculator-1.jar"]
```

DockerFile

To create the Docker image and upload it to a remote repository on Docker Hub, we will define the steps in a Jenkins pipeline script. The Docker image is generated and uploaded to the user's Docker Hub repository. After the image is successfully uploaded to Docker Hub, the local image on the machine is removed.



hub.docker.com/repository/docker/mohitgupta02/calculator2/general

mohitgupta02 / Repositories / calculator2 / General

Using 0 of 1 private repositories.

**mohitgupta02/calculator2**

Last pushed 9 minutes ago · Repository size: 251.5 MB

Docker commands

To push a new tag to this repository:

```
docker push mohitgupta02/calculator2:tagname
```

**Tags**

This repository contains 1 tag(s).

| Tag | OS | Type | Pulled | Pushed |
|---|---|---|---|---|
| latest | | Image | less than 1 day | 9 minutes |

**Automated builds**

Manually pushing images to Docker Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about automated builds.

**Repository overview** INCOMPLETE

*An overview describes what your image does and how to run it. It displays in the public view of your repository once you have pushed some content.*

Docker Hub Repository

## Jenkins:

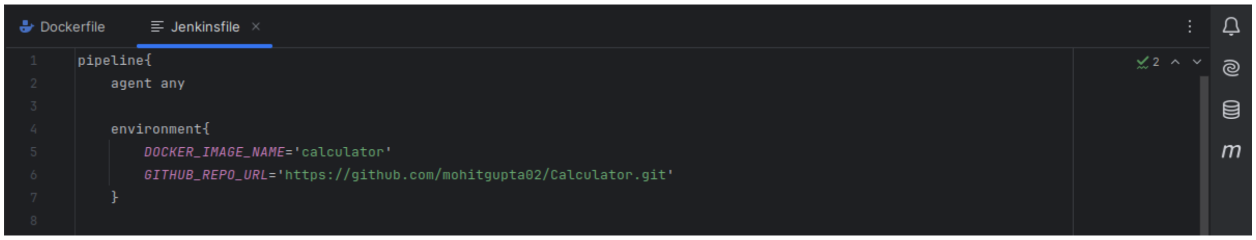
Jenkins is a robust tool that facilitates continuous integration and continuous delivery (CI/CD) for projects, regardless of the platform in use. It is an open-source application capable of handling all kinds of builds and CI processes. Jenkins can be integrated with a variety of testing and deployment tools.

The pipeline is set up so that whenever a new commit is made to the GitHub repository, the pipeline is triggered, executing all steps in sequence. A Jenkins pipeline is an automated workflow that outlines the process of retrieving software from version control. Each change to the software undergoes a series of intricate processes before being released. This includes developing the software in a repeatable, reliable manner, and progressing through multiple stages of testing and deployment.

All tasks are interconnected through the Jenkins pipeline. The pipeline pulls changes from GitHub, builds the project using Maven, creates the Docker image, pushes the image to Docker Hub, and then runs the Ansible playbook.

### Create CI Pipeline

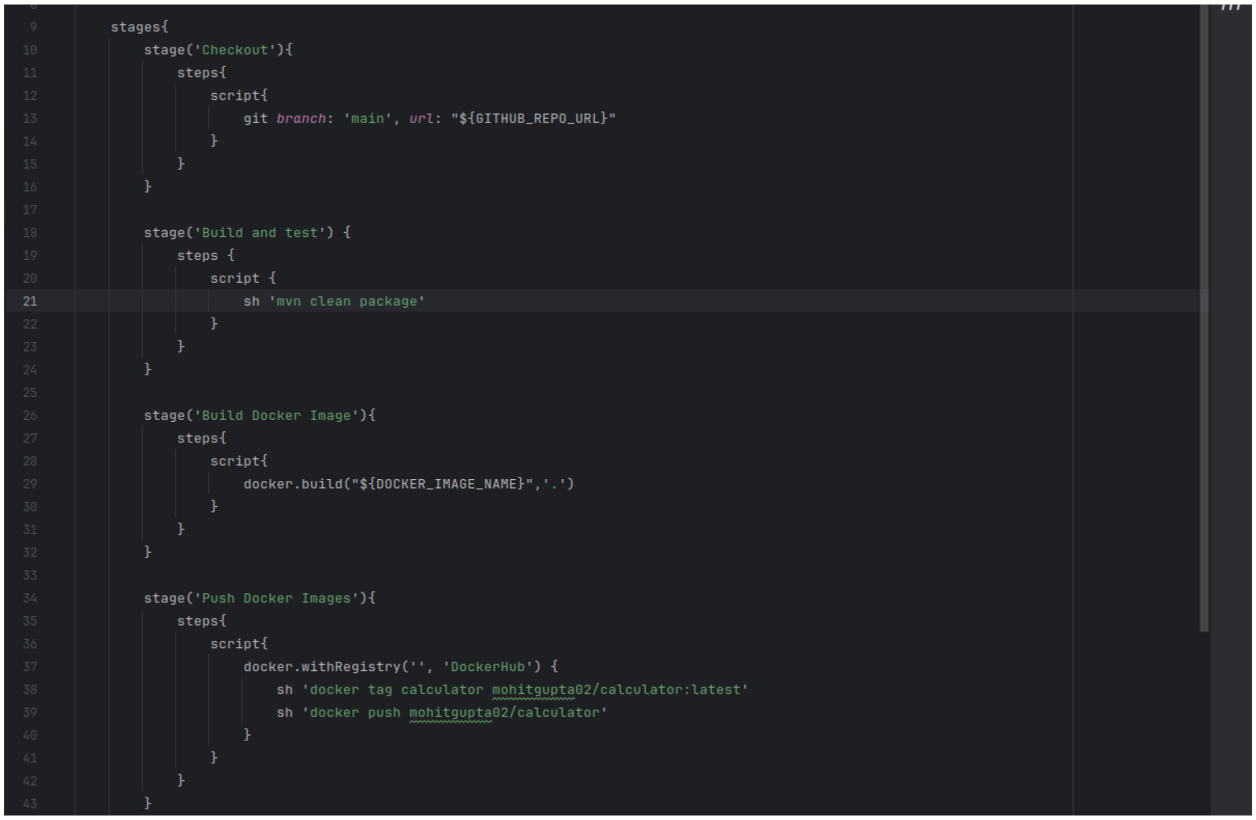
A Jenkins pipeline is an automated representation of the steps involved in obtaining software from version control. Every change made to the software goes through a series of detailed processes before being released. This includes developing the software in a consistent and dependable way, as well as advancing the software through several stages of testing and deployment.



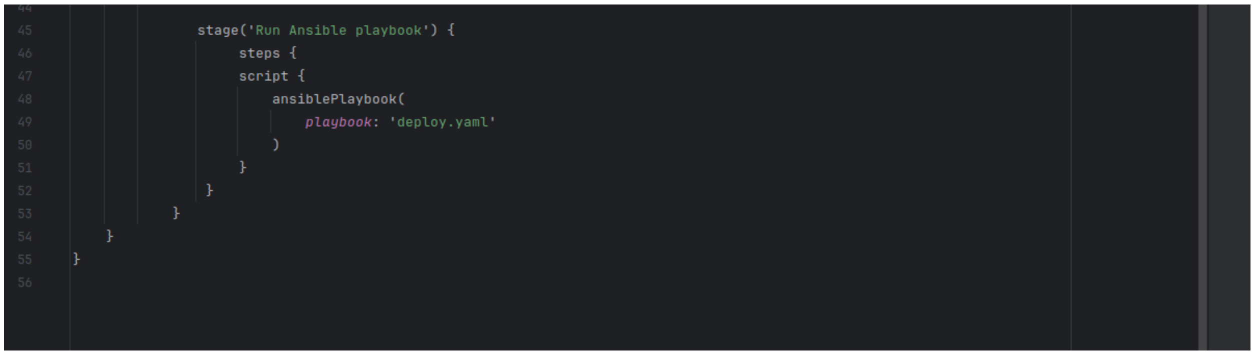
```
pipeline{
    agent any

    environment{
        DOCKER_IMAGE_NAME='calculator'
        GITHUB_REPO_URL='https://github.com/mohitgupta02/Calculator.git'
    }

```

Jenkins Pipeline

There are various stages in Jenkins Pipeline including Checkout, Build Docker Image, Push Docker Image and Run ansible playbook then there is post action to send email for acknowledge the status of pipeline.



```
    stages{
        stage('Checkout'){
            steps{
                script{
                    git branch: 'main', url: "${GITHUB_REPO_URL}"
                }
            }
        }

        stage('Build and test') {
            steps {
                script {
                    sh 'mvn clean package'
                }
            }
        }

        stage('Build Docker Image'){
            steps{
                script{
                    docker.build("${DOCKER_IMAGE_NAME}",'.')
                }
            }
        }

        stage('Push Docker Images'){
            steps{
                script{
                    docker.withRegistry('', 'DockerHub') {
                        sh 'docker tag calculator mohitgupta02/calculator:latest'
                        sh 'docker push mohitgupta02/calculator'
                    }
                }
            }
        }
```



```
        stage('Run Ansible playbook') {
            steps {
            script {
                ansiblePlaybook(
                    playbook: 'deploy.yaml'
                )
            }
        }
    }
}
}
```

Stages of Pipeline

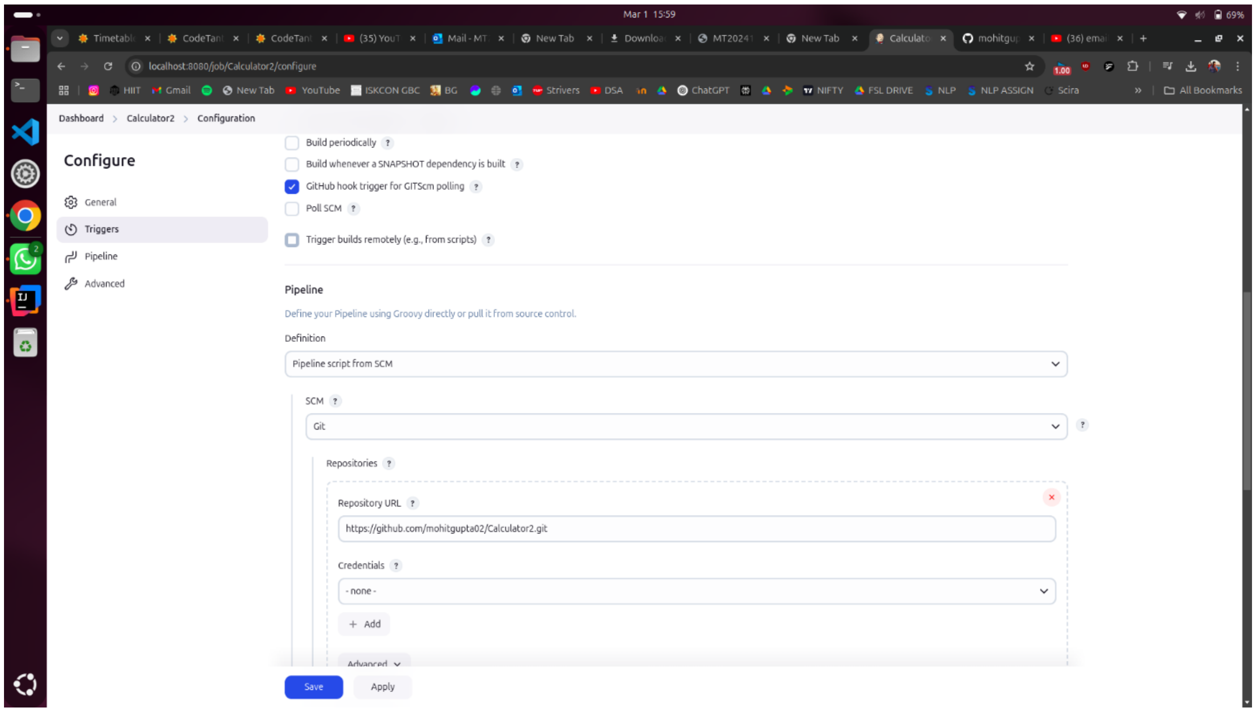
## Configuration of Project in Jenkins

To run Jenkins in local machine there is a command to run it

sudo systemctl start jenkins

To check status of Jenkins

sudo systemctl status jenkins


Mar 1 15:59
localhost:8080/job/Calculator2/configure
Dashboard > Calculator2 > Configuration
Configure
General
Triggers
Pipeline
Advanced
Save
Apply

- [ ] Build periodically
- [ ] Build whenever a SNAPSHOT dependency is built
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM
- [ ] Trigger builds remotely (e.g., from scripts)

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition: Pipeline script from SCM

SCM: Git

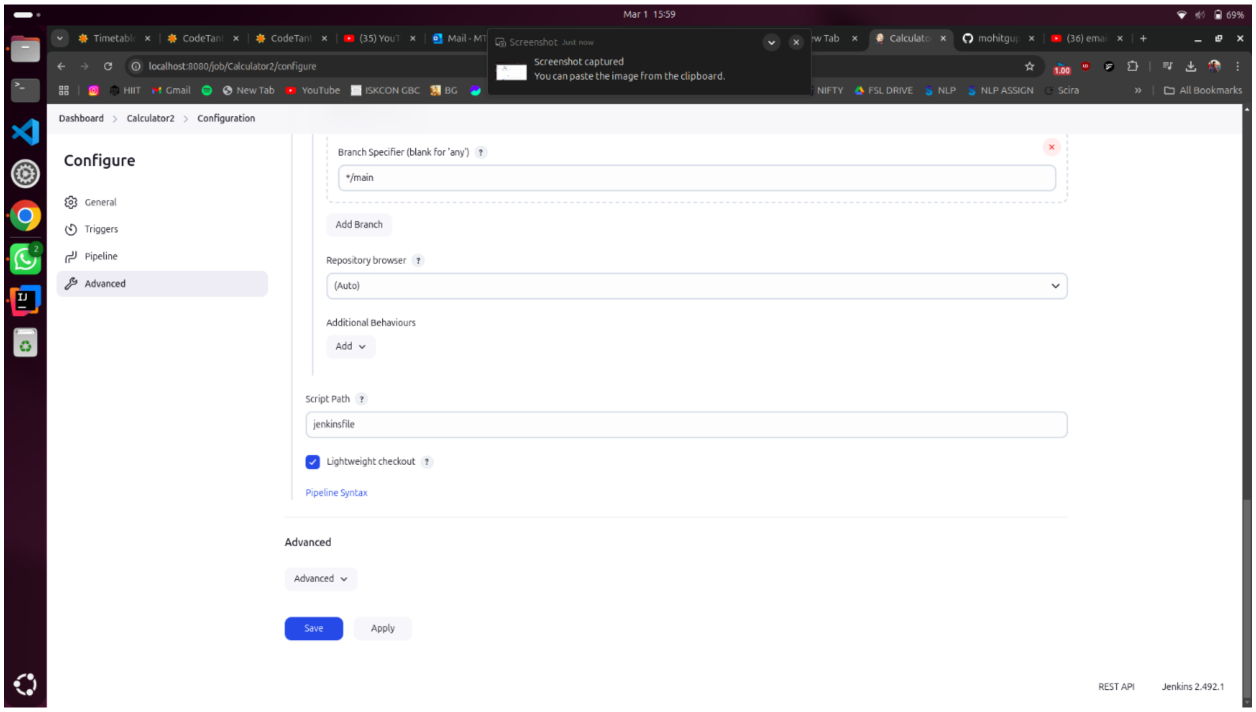
Repositories

Repository URL: https://github.com/mohitgupta02/Calculator2.git

Credentials: - none -

+ Add

Advanced


Dashboard > Calculator2 > Configuration

Configure

Branch Specifier (blank for 'any'): */main

Repository browser: (Auto)

Additional Behaviours

Script Path: jenkinsfile

☑ Lightweight checkout

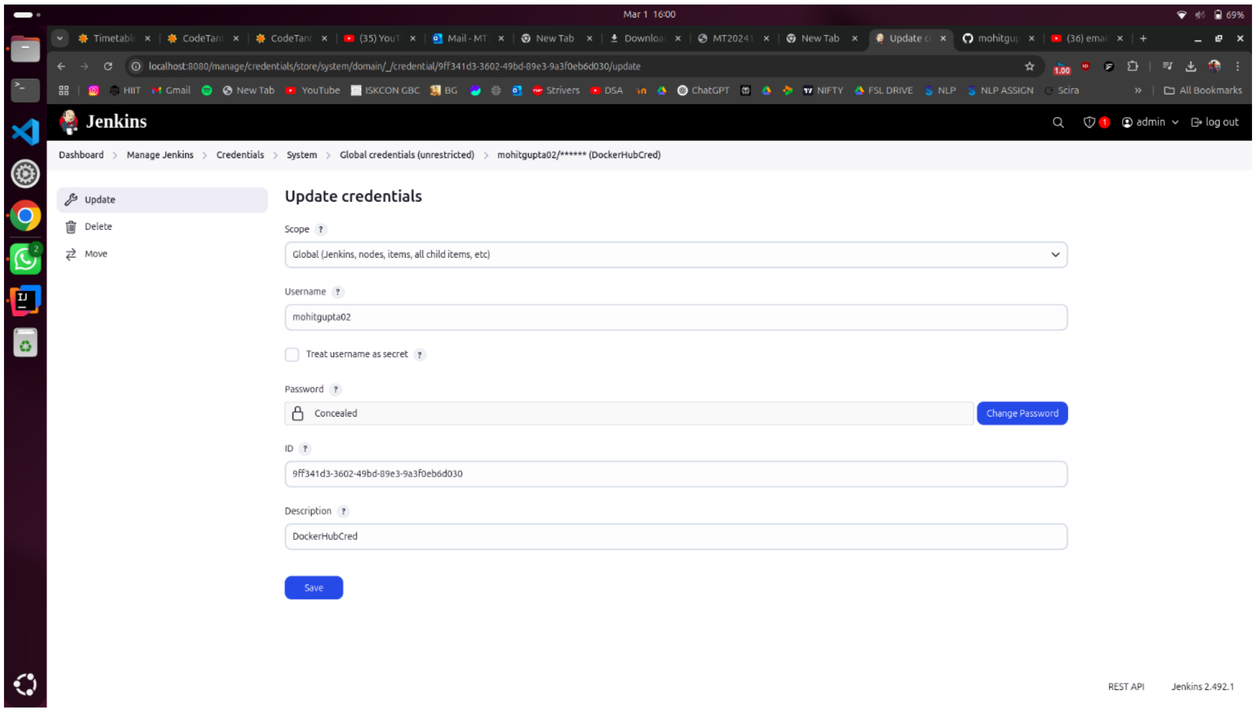
Advanced

Configuration of Project

In this step, we configure a GitHub project in Jenkins by providing the URL of our GitHub repository and specifying the path to the Jenkins file in the configuration.

We then set up system credentials for both Docker Hub and the local machine for the Ansible playbook. Since the project runs on the local machine and pulls the image from Docker Hub, it's necessary to define these system configurations in Jenkins' system credentials. Two credentials need to be added: one for the Docker Hub repository and another for the local machine.

To configure Docker Hub credentials, we must specify the Docker Hub username from which the image will be pulled.


localhost:8080/manage/credentials/store/system/domain/_/credential/9ff341d3-3602-49bd-89e3-9a3f0eb6d030/update

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > mohitgupta02/****** (DockerHubCred)

Update credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: mohitgupta02

Treat username as secret

Password: Concealed

ID: 9ff341d3-3602-49bd-89e3-9a3f0eb6d030

Description: DockerHubCred

Adding DockerHub Credentials

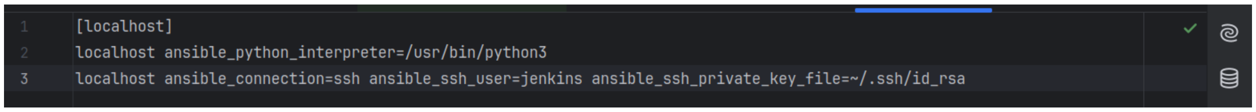
## Ansible:

Ansible is a continuous deployment tool. In which there is one control node and the nodes on which the deliverables are deployed are called managed nodes.

Ansible need to be installed only on the control node. Ansible connects with the managed nodes using ssh. In Ansible we provide a set of commands that need to be executed on the managed node in the form of yml file. It is called a playbook. Along with the playbook we also provide information about the managed nodes. Which called inventory.

To execute this playbook we run the following command:

ansible-playbook <playbook> -i <inventory>



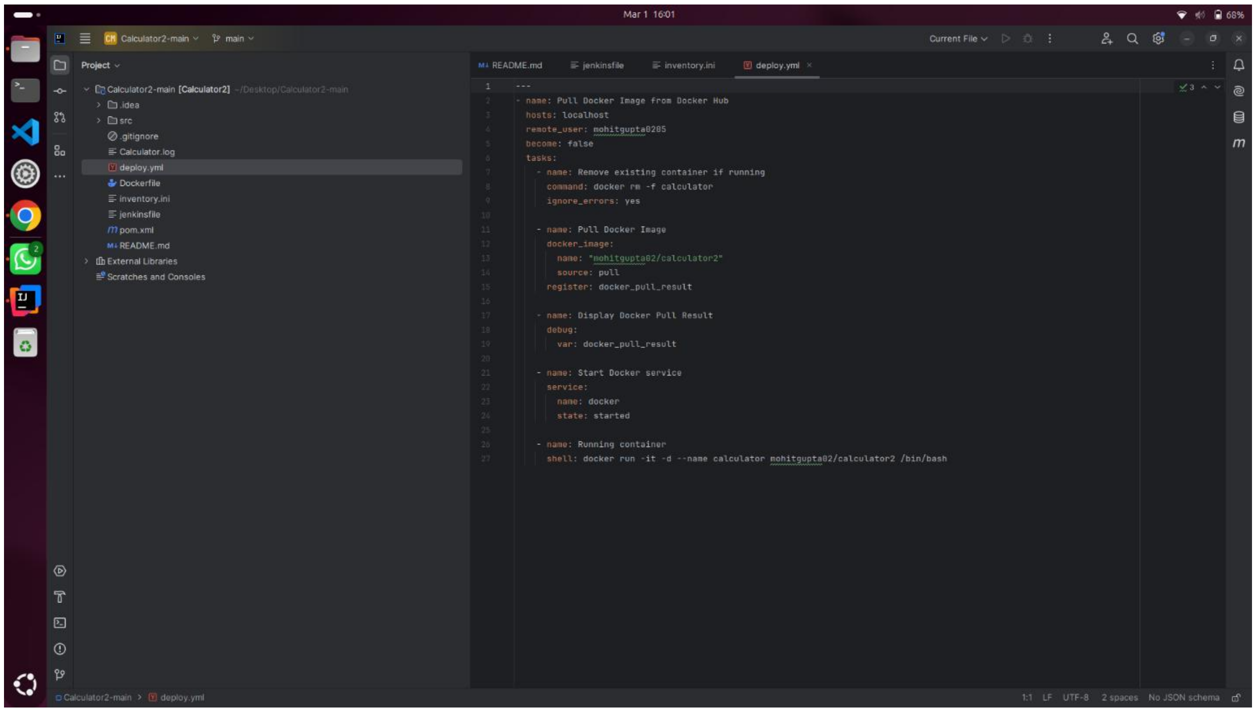
```
[localhost]
localhost ansible_python_interpreter=/usr/bin/python3
localhost ansible_connection=ssh ansible_ssh_user=jenkins ansible_ssh_private_key_file=~/.ssh/id_rsa
```

Inventory File

The Ansible inventory file specifies the hosts and host groups on which tasks, modules, and commands within a playbook are executed. The format of the inventory file can vary based on the Ansible environment and the plugins in use.

Ansible playbooks are used to document and carry out configuration, deployment, and orchestration tasks.

To run the application’s Docker image on the host, Python's pip and Docker must be installed on the host machine. The Ansible playbook pulls the Docker image from Docker Hub and launches the container on the specified hosts. This entire process is automated through the playbook.

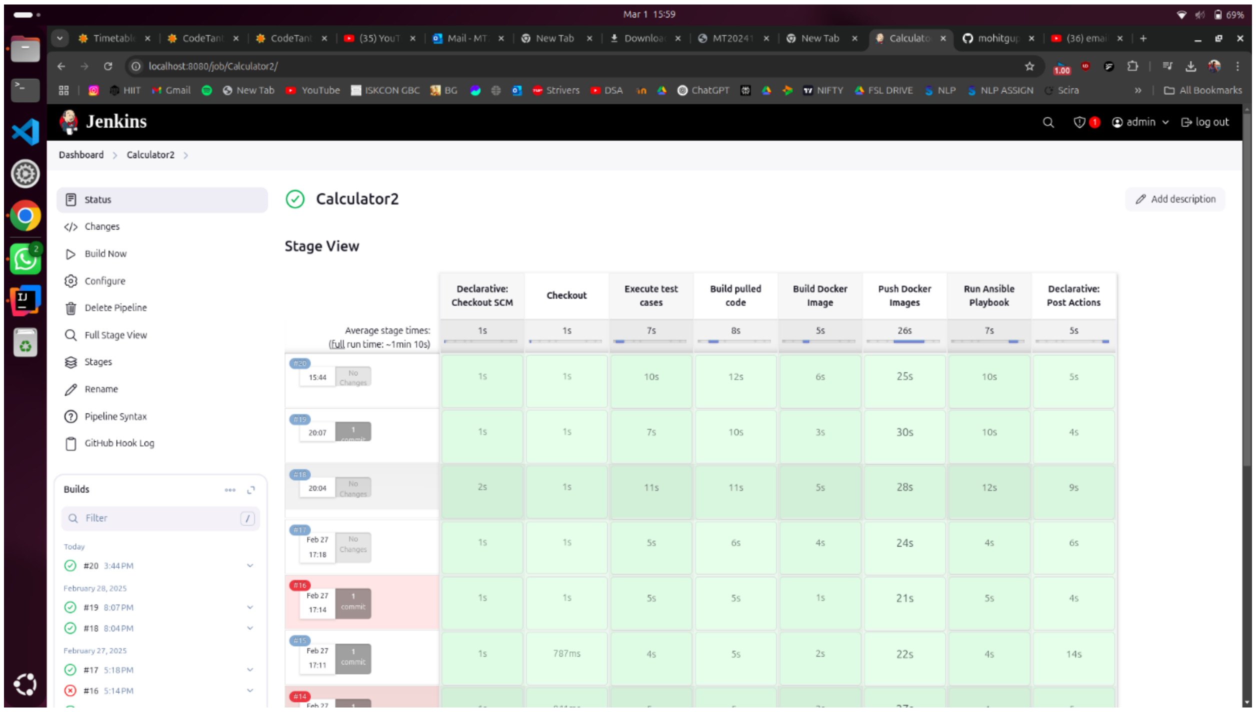


```yaml
---
- name: Pull Docker Image from Docker Hub
  hosts: localhost
  remote_user: mohitgupta0205
  become: false
  tasks:
    - name: Remove existing container if running
      command: docker rm -f calculator
      ignore_errors: yes

    - name: Pull Docker Image
      docker_image:
        name: "mohitgupta02/calculator2"
        source: pull
      register: docker_pull_result

    - name: Display Docker Pull Result
      debug:
        var: docker_pull_result

    - name: Start Docker service
      service:
        name: docker
        state: started

    - name: Running container
      shell: docker run -it -d --name calculator mohitgupta02/calculator2 /bin/bash
```

Playbook file

## Run Project as Pipeline in Jenkins

To run the project as pipeline in jenkins we have to click build now to see each stages of pipeline and show the output of each stages.



Calculator2

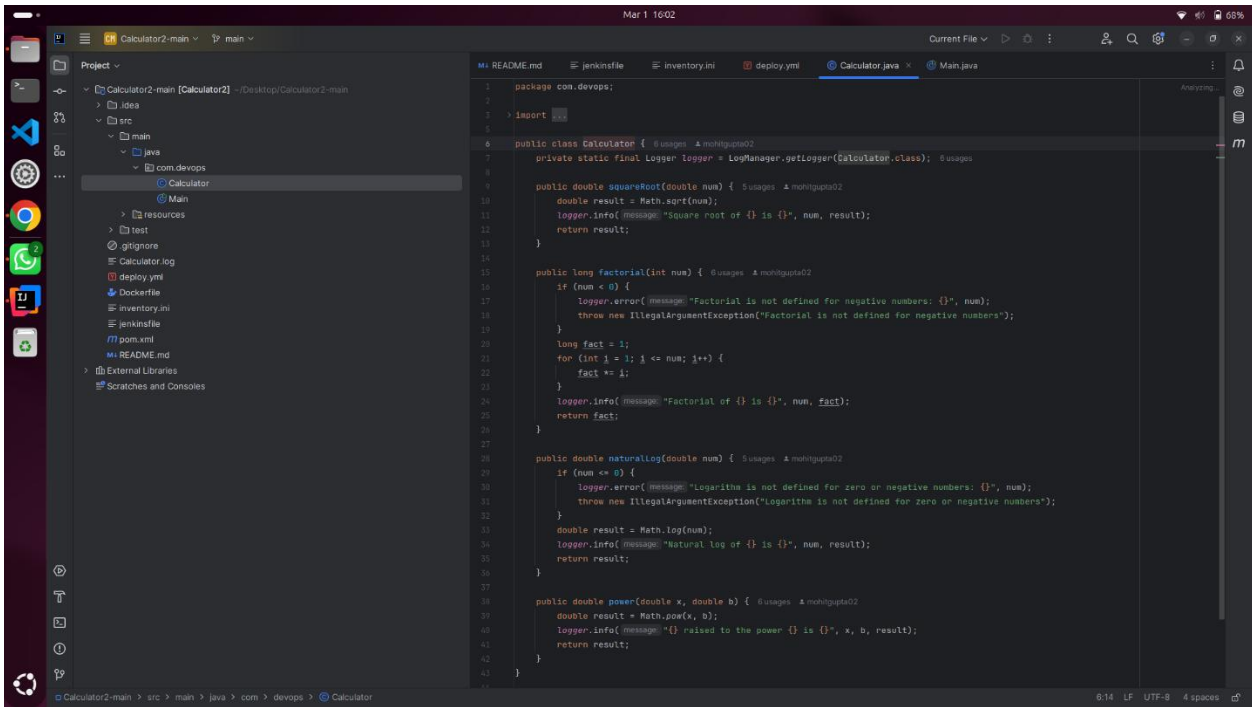
Stage View

| | Declarative: Checkout SCM | Checkout | Execute test cases | Build pulled code | Build Docker Image | Push Docker Images | Run Ansible Playbook | Declarative: Post Actions |
|---|---|---|---|---|---|---|---|---|
| Average stage times: (full run time: ~1min 10s) | 1s | 1s | 7s | 8s | 5s | 26s | 7s | 5s |
| #20 15:44 No Changes | 1s | 1s | 10s | 12s | 6s | 25s | 10s | 5s |
| #19 20:07 1 commit | 1s | 1s | 7s | 10s | 3s | 30s | 10s | 4s |
| #18 20:04 No Changes | 2s | 1s | 11s | 11s | 5s | 28s | 12s | 9s |
| #17 Feb 27 17:18 No Changes | 1s | 1s | 5s | 6s | 4s | 24s | 4s | 6s |
| #16 Feb 27 17:14 1 commit | 1s | 1s | 5s | 5s | 1s | 21s | 5s | 4s |
| #15 Feb 27 17:11 1 commit | 1s | 787ms | 4s | 5s | 2s | 22s | 4s | 14s |

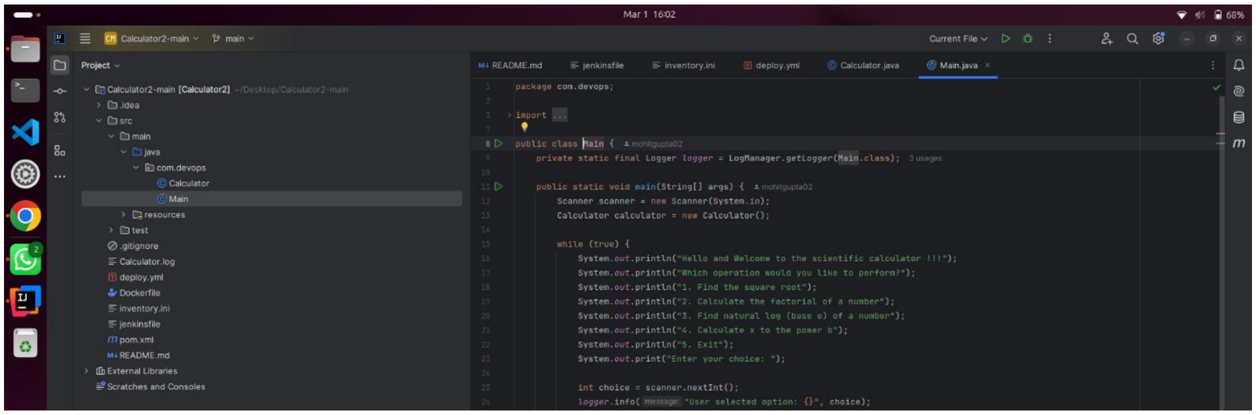
View of Pipeline Stages

## Scientific Calculator Source Code:

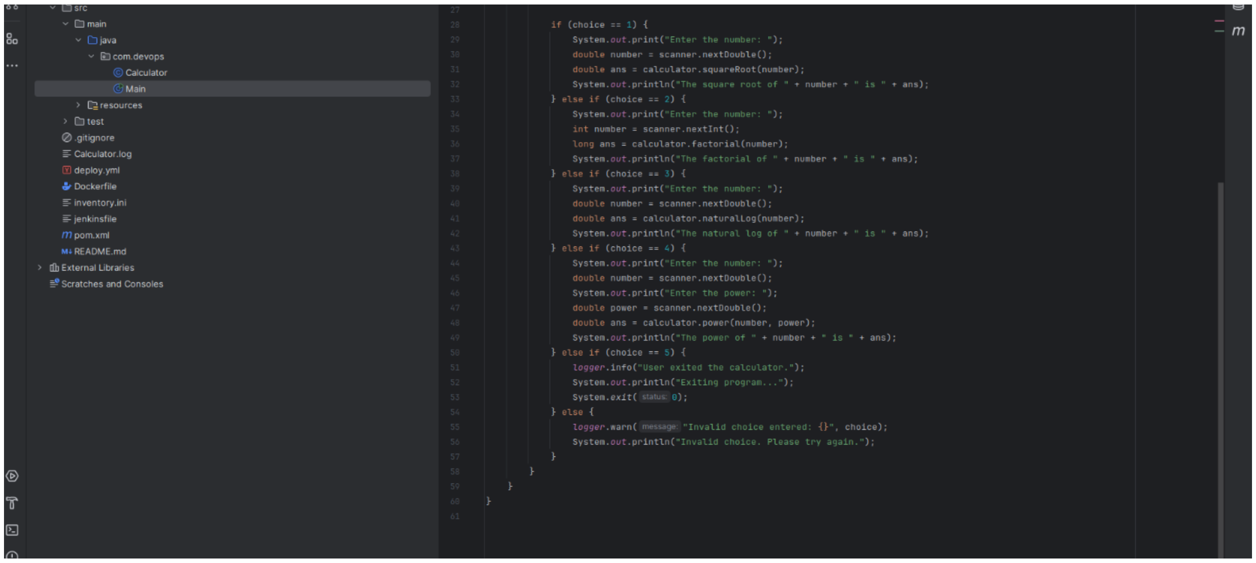
Below attached the source code of scientific calculator in java



```java
package com.devops;

import ...

public class Calculator {
    private static final Logger logger = LogManager.getLogger(Calculator.class);

    public double squareRoot(double num) {
        double result = Math.sqrt(num);
        logger.info("Square root of {} is {}", num, result);
        return result;
    }

    public long factorial(int num) {
        if (num < 0) {
            logger.error("Factorial is not defined for negative numbers: {}", num);
            throw new IllegalArgumentException("Factorial is not defined for negative numbers");
        }
        long fact = 1;
        for (int i = 1; i <= num; i++) {
            fact *= i;
        }
        logger.info("Factorial of {} is {}", num, fact);
        return fact;
    }

    public double naturalLog(double num) {
        if (num <= 0) {
            logger.error("Logarithm is not defined for zero or negative numbers: {}", num);
            throw new IllegalArgumentException("Logarithm is not defined for zero or negative numbers");
        }
        double result = Math.log(num);
        logger.info("Natural log of {} is {}", num, result);
        return result;
    }

    public double power(double x, double b) {
        double result = Math.pow(x, b);
        logger.info("{} raised to the power {} is {}", x, b, result);
        return result;
    }
}
```



```java
package com.devops;

import ...

public class Main {
    private static final Logger logger = LogManager.getLogger(Main.class);

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Calculator calculator = new Calculator();

        while (true) {
            System.out.println("Hello and Welcome to the scientific calculator !!!");
            System.out.println("Which operation would you like to perform?");
            System.out.println("1. Find the square root");
            System.out.println("2. Calculate the factorial of a number");
            System.out.println("3. Find natural log (base e) of a number");
            System.out.println("4. Calculate x to the power b");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();
            logger.info("User selected option: {}", choice);
```



```java

            if (choice == 1) {
                System.out.print("Enter the number: ");
                double number = scanner.nextDouble();
                double ans = calculator.squareRoot(number);
                System.out.println("The square root of " + number + " is " + ans);
            } else if (choice == 2) {
                System.out.print("Enter the number: ");
                int number = scanner.nextInt();
                long ans = calculator.factorial(number);
                System.out.println("The factorial of " + number + " is " + ans);
            } else if (choice == 3) {
                System.out.print("Enter the number: ");
                double number = scanner.nextDouble();
                double ans = calculator.naturalLog(number);
                System.out.println("The natural log of " + number + " is " + ans);
            } else if (choice == 4) {
                System.out.print("Enter the number: ");
                double number = scanner.nextDouble();
                System.out.print("Enter the power: ");
                double power = scanner.nextDouble();
                double ans = calculator.power(number, power);
                System.out.println("The power of " + number + " is " + ans);
            } else if (choice == 5) {
                logger.info("User exited the calculator.");
                System.out.println("Exiting program...");
                System.exit(0);
            } else {
                logger.warn("Invalid choice entered: {}", choice);
                System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```

## Adding Test Cases:

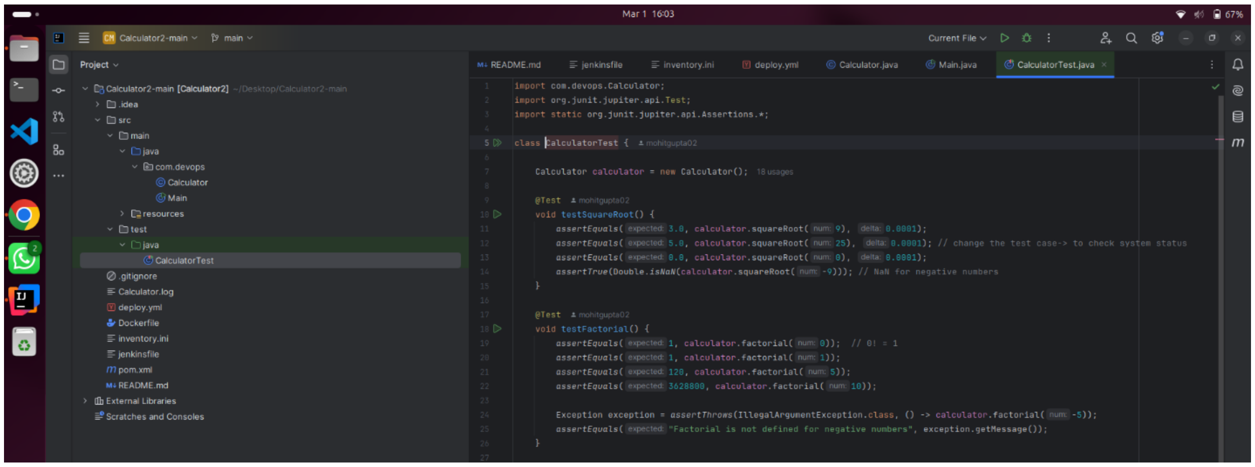
### What is Unit Testing?

Unit Testing is the process of testing individual components of code in isolation. A "unit" refers to the smallest functional part of a system, which could be a single line of code, a method, or an entire class. The smaller the unit being tested, the faster and more efficient the process becomes. This helps in gaining deeper insights into the code's functionality and overall performance.

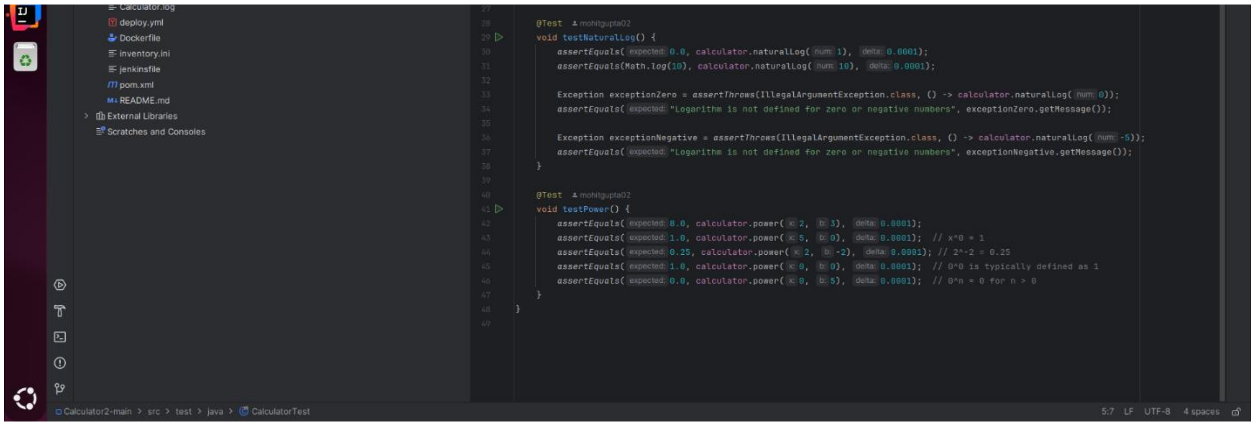
### JUnit: A Unit Testing Framework for Java

JUnit is an open-source framework designed for unit testing in Java. It allows developers to write and execute automated test cases efficiently. In Java development, test cases must be rerun whenever new code is introduced to ensure that existing functionality remains intact.

JUnit also provides visual feedback on test execution through graphical indicators. A successful test is shown in green, while a failed test appears in red. By leveraging JUnit, developers can create robust, error-free code with greater reliability.



```java
import com.devops.Calculator;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class CalculatorTest {

    Calculator calculator = new Calculator();

    @Test
    void testSquareRoot() {
        assertEquals(3.0, calculator.squareRoot(9), 0.0001);
        assertEquals(5.0, calculator.squareRoot(25), 0.0001); // change the test case-> to check system status
        assertEquals(0.0, calculator.squareRoot(0), 0.0001);
        assertTrue(Double.isNaN(calculator.squareRoot(-9))); // NaN for negative numbers
    }

    @Test
    void testFactorial() {
        assertEquals(1, calculator.factorial(0));  // 0! = 1
        assertEquals(1, calculator.factorial(1));
        assertEquals(120, calculator.factorial(5));
        assertEquals(3628800, calculator.factorial(10));

        Exception exception = assertThrows(IllegalArgumentException.class, () -> calculator.factorial(-5));
        assertEquals("Factorial is not defined for negative numbers", exception.getMessage());
    }

```

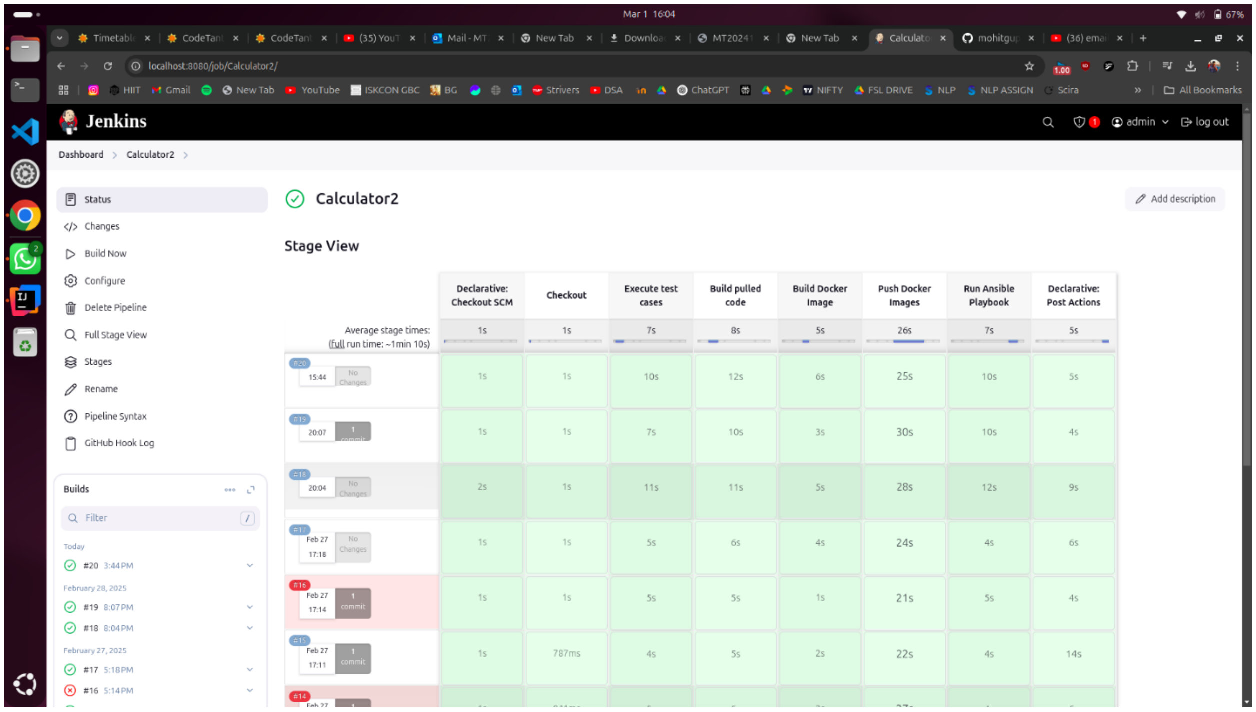


```java

    @Test
    void testNaturalLog() {
        assertEquals(0.0, calculator.naturalLog(1), 0.0001);
        assertEquals(Math.log(10), calculator.naturalLog(10), 0.0001);

        Exception exceptionZero = assertThrows(IllegalArgumentException.class, () -> calculator.naturalLog(0));
        assertEquals("Logarithm is not defined for zero or negative numbers", exceptionZero.getMessage());

        Exception exceptionNegative = assertThrows(IllegalArgumentException.class, () -> calculator.naturalLog(-5));
        assertEquals("Logarithm is not defined for zero or negative numbers", exceptionNegative.getMessage());
    }

    @Test
    void testPower() {
        assertEquals(8.0, calculator.power(2, 3), 0.0001);
        assertEquals(1.0, calculator.power(5, 0), 0.0001);  // x^0 = 1
        assertEquals(0.25, calculator.power(2, -2), 0.0001); // 2^-2 = 0.25
        assertEquals(1.0, calculator.power(0, 0), 0.0001);  // 0^0 is typically defined as 1
        assertEquals(0.0, calculator.power(0, 5), 0.0001);  // 0^n = 0 for n > 0
    }
}
```

Adding Test Cases Java File

**Run Project:**

To run project first run pipeline in Jenkins



Calculator2

Stage View

| | Declarative: Checkout SCM | Checkout | Execute test cases | Build pulled code | Build Docker Image | Push Docker Images | Run Ansible Playbook | Declarative: Post Actions |
|---|---|---|---|---|---|---|---|---|
| Average stage times: (full run time: ~1min 10s) | 1s | 1s | 7s | 8s | 5s | 26s | 7s | 5s |
| #20 15:44 No Changes | 1s | 1s | 10s | 12s | 6s | 25s | 10s | 5s |
| #19 20:07 1 commit | 1s | 1s | 7s | 10s | 3s | 30s | 10s | 4s |
| #18 20:04 No Changes | 2s | 1s | 11s | 11s | 5s | 28s | 12s | 9s |
| #17 Feb 27 17:18 No Changes | 1s | 1s | 5s | 6s | 4s | 24s | 4s | 6s |
| #16 Feb 27 17:14 1 commit | 1s | 1s | 5s | 5s | 1s | 21s | 5s | 4s |
| #15 Feb 27 17:11 1 commit | 1s | 787ms | 4s | 5s | 2s | 22s | 4s | 14s |

Pipeline View

**Webhooks:**

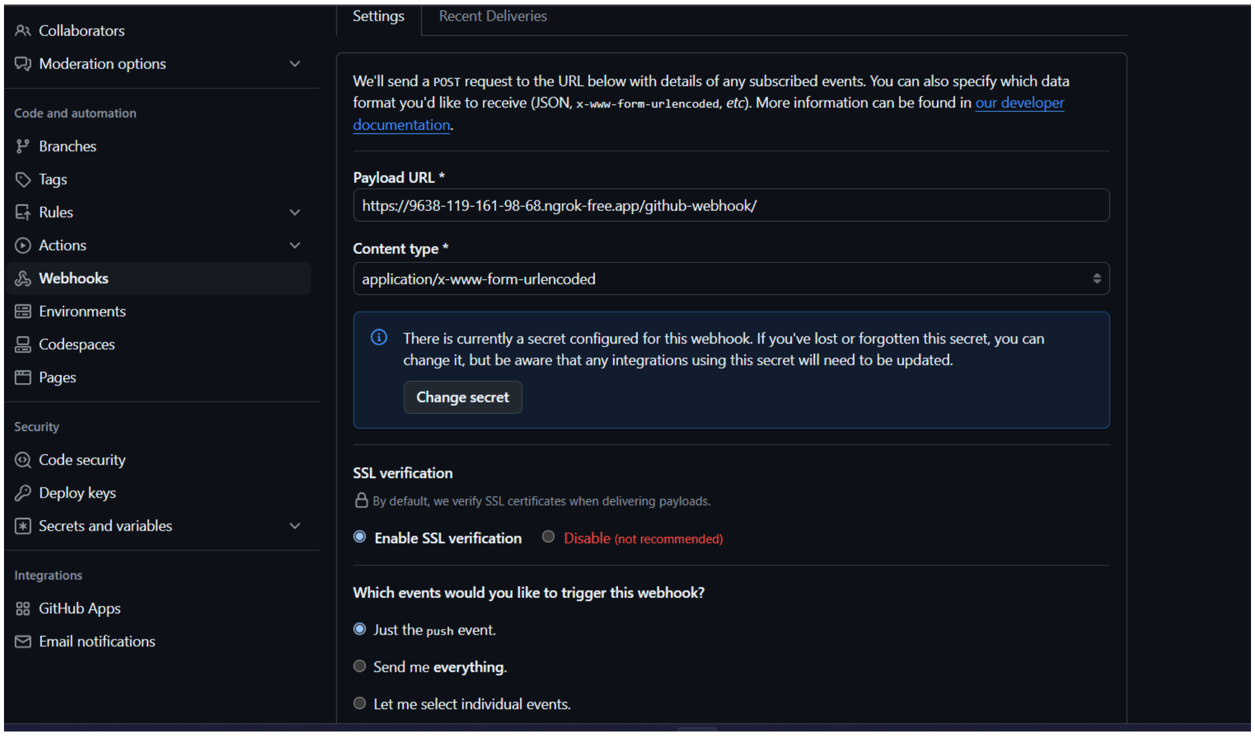
Webhooks are automated HTTP callbacks that trigger actions when an event occurs, such as a code push in a repository.

**Use of Webhooks in Jenkins with Ngrok**

Ngrok is used to expose Jenkins running on a local machine to the internet, enabling webhooks to communicate with it. This allows GitHub/GitLab webhooks to trigger Jenkins builds even when running on a local server.

**Key Benefits:**

- **Automatic Build Triggers:** Webhooks notify Jenkins of code changes.
- **Remote Accessibility:** Ngrok provides a public URL for Jenkins, making it accessible for webhook events.
- **Faster CI/CD:** Automates build and deployment without manual intervention.



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://9638-119-161-98-68.ngrok-free.app/github-webhook/

Content type *
application/x-www-form-urlencoded

There is currently a secret configured for this webhook. If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated.

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.

Webhooks Using Through the Git-hub

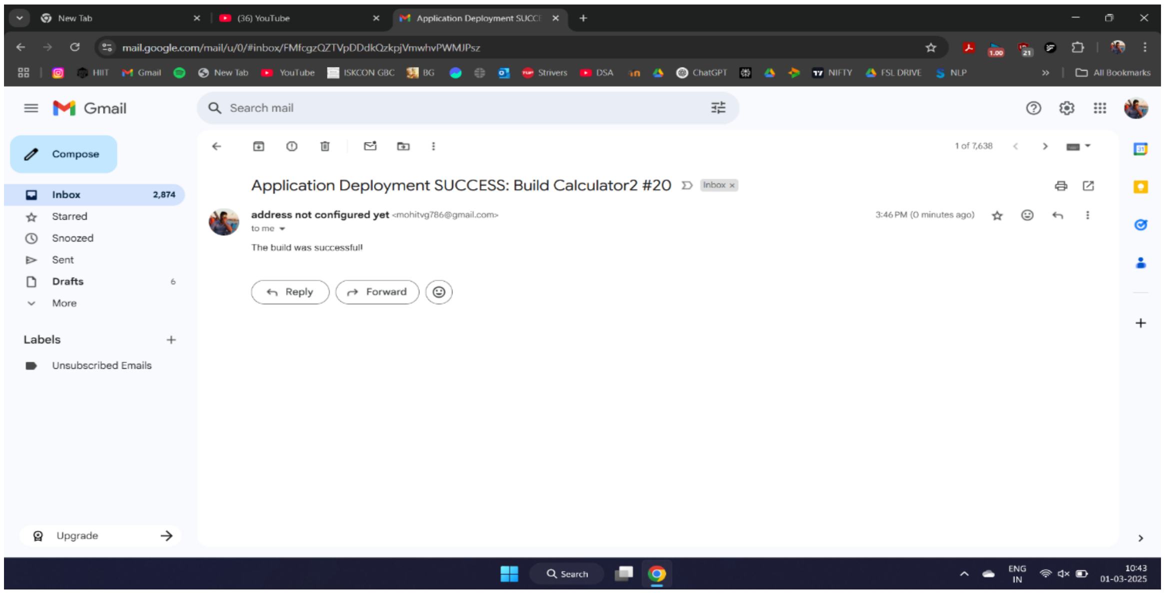
**Email Notification In Jenkins Pipeline project:**

What is SMTP in Jenkins?

SMTP (Simple Mail Transfer Protocol) allows Jenkins to send email notifications about build status, failures, or successes to developers and stakeholders. This ensures timely updates on project progress.

Use of SMTP in Jenkins

- Automated Email Alerts: Sends build status updates after each pipeline run.
- Failure Notifications: Alerts developers immediately if a build fails.
- Custom Messages: Can include logs, reports, and issue details in emails.



Application Deployment SUCCESS: Build Calculator2 #20

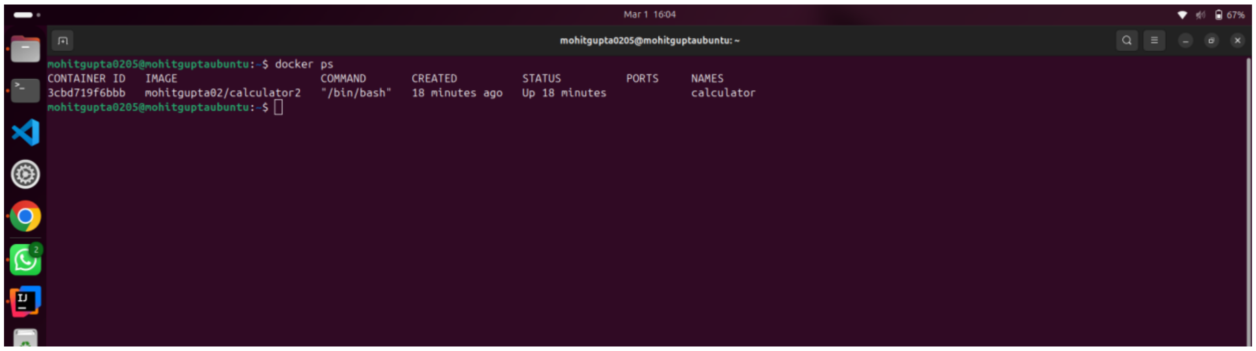
address not configured yet <mohitvg786@gmail.com>
to me

3:46 PM (0 minutes ago)

The build was successful!

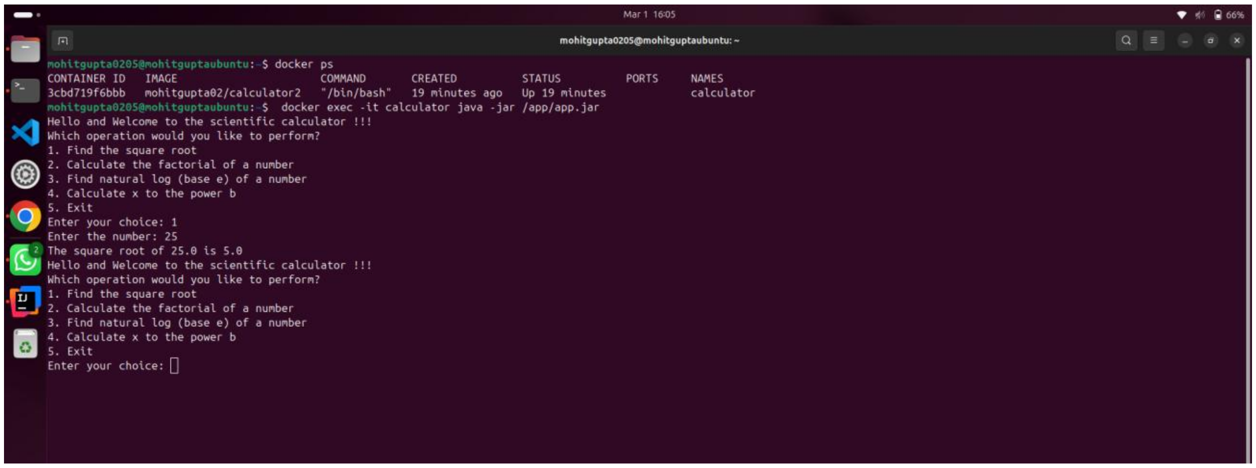
Email Photo Send by Jenkins after Deployment

Project Runs successfully and now our docker image is also pulled from docker hub. We Can run in local machine to check whether it runs or not by using below command.



```
mohitgupta0205@mohitguptaubuntu:~$ docker ps
CONTAINER ID   IMAGE                    COMMAND       CREATED          STATUS          PORTS     NAMES
3cbd719f6bbb   mohitgupta02/calculator2   "/bin/bash"   18 minutes ago   Up 18 minutes             calculator
mohitgupta0205@mohitguptaubuntu:~$
```

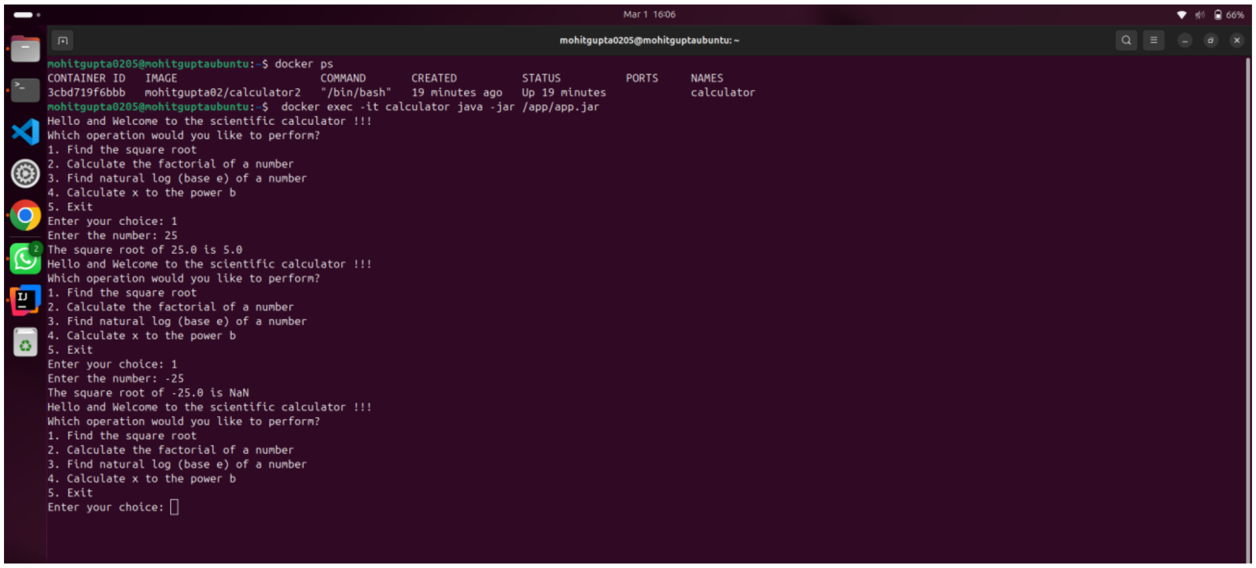
So, right now our container is running and to execute this code  we are using below command to enter the container and check whether our code is working or not.



```
mohitgupta0205@mohitguptaubuntu:~$ docker ps
CONTAINER ID   IMAGE                      COMMAND       CREATED          STATUS          PORTS     NAMES
3cbd719f6bbb   mohitgupta02/calculator2   "/bin/bash"   19 minutes ago   Up 19 minutes             calculator
mohitgupta0205@mohitguptaubuntu:~$  docker exec -it calculator java -jar /app/app.jar
Hello and Welcome to the scientific calculator !!!
Which operation would you like to perform?
1. Find the square root
2. Calculate the factorial of a number
3. Find natural log (base e) of a number
4. Calculate x to the power b
5. Exit
Enter your choice: 1
Enter the number: 25
The square root of 25.0 is 5.0
Hello and Welcome to the scientific calculator !!!
Which operation would you like to perform?
1. Find the square root
2. Calculate the factorial of a number
3. Find natural log (base e) of a number
4. Calculate x to the power b
5. Exit
Enter your choice: 
```

Execute Container

## Check for Some Test Cases:



```
mohitgupta0205@mohitguptaubuntu:~$ docker ps
CONTAINER ID   IMAGE                      COMMAND       CREATED          STATUS          PORTS     NAMES
3cbd719f6bbb   mohitgupta02/calculator2   "/bin/bash"   19 minutes ago   Up 19 minutes             calculator
mohitgupta0205@mohitguptaubuntu:~$  docker exec -it calculator java -jar /app/app.jar
Hello and Welcome to the scientific calculator !!!
Which operation would you like to perform?
1. Find the square root
2. Calculate the factorial of a number
3. Find natural log (base e) of a number
4. Calculate x to the power b
5. Exit
Enter your choice: 1
Enter the number: 25
The square root of 25.0 is 5.0
Hello and Welcome to the scientific calculator !!!
Which operation would you like to perform?
1. Find the square root
2. Calculate the factorial of a number
3. Find natural log (base e) of a number
4. Calculate x to the power b
5. Exit
Enter your choice: 1
Enter the number: -25
The square root of -25.0 is NaN
Hello and Welcome to the scientific calculator !!!
Which operation would you like to perform?
1. Find the square root
2. Calculate the factorial of a number
3. Find natural log (base e) of a number
4. Calculate x to the power b
5. Exit
Enter your choice: 
```

Checking for the Execution

GitHub Repo : https://github.com/mohitgupta02/Calculator2