



# Scientific Calculator with DevOps

**Submitted By**

**Priyanshu Vaish, MT2023089**

**INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY,  
BANGALORE.**

26/C, Hosur Rd, Electronics City Phase 1, Electronic City, Bengaluru, Karnataka 560100,  
Academic Year 2023-24

GitHub Link

<https://github.com/Priyansuvaish/project>

DockerHub Link

[docker pull priyanshugupta753/calculator](https://hub.docker.com/r/priyanshugupta753/calculator)

# Table of Contents

## → Introduction

- ◆ What is DevOps?
- ◆ Why DevOps?

## → Tools Used in DevOps

## → Steps Explanation and Tool Configuration

### ◆ Source Control Management: Git and GitHub

- Brief Overview
- Setup and Configurations
- Commands Used
- Screenshots

### ◆ Building the code with Maven and Testing with JUnit

- Brief Overview
- Setup and Configurations
- Update the pom file
- Screenshots of the dependencies

### ◆ Continuous Integration/Continuous Deployment (CI/CD): Jenkins

- Brief Overview
- Setup and Configurations
- Screenshots of the Jenkins Pipeline Script

### ◆ Containerization: Docker

- Brief Overview
- Setup and Configurations
- Dockerfile Screenshots

### ◆ Configuration Management: Ansible

- Brief Overview
- Setup and Configurations
- Ansible Playbook Screenshots

## → Application working ScreenShot

# Introduction

## What is DevOps?

DevOps is like a bridge that connects the world of software development (where applications and systems are created) with IT operations (where those applications are deployed and maintained in the real world). This connection is not just about making sure that the two sides talk to each other but also ensuring they work closely together throughout the entire lifecycle of a software system. By doing so, DevOps aims to make the process of developing, testing, deploying, and maintaining software much smoother and faster.

In other words, DevOps is all about making the process of delivering software better, faster, and more efficient. It's like turning a bumpy, winding dirt road into a smooth, straight highway, allowing for quicker and safer delivery of the end product to the customer.

## Why DevOps?

There are many benefits of using DevOps some of them are as follows:

- With DevOps, companies can release new features, updates, and fixes more quickly and frequently. This rapid pace allows businesses to respond to market changes better and enhance customer satisfaction.
- Closer collaboration between teams means problems can be detected and resolved faster. This teamwork approach leads to a better understanding of the software and infrastructure, reducing downtime and improving service reliability.
- By automating repetitive tasks and streamlining processes, teams have more time to focus on innovation and improvement. DevOps practices like continuous integration and continuous deployment automate the software release process, from build to deployment.

- DevOps encourages the use of practices such as infrastructure as code (IaC), which allows the management of infrastructure using configuration files. This approach leads to more efficient use of resources, reduces errors, and enhances reproducibility.

## **Tools Used in DevOps**

- Source Control Management (SCM): GitHub
- Build the code: Maven
- Testing: JUnit
- Continuous Integration/Continuous Deployment (CI/CD): Jenkins
- Containerization: Docker
- For Deployment: Ansible

# Steps Explanation and Tool Configuration

## Source Control Management with GitHub

SCM is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management. GitHub, a cloud-based hosting service, lets you manage Git repositories.

### Setup and Configurations:

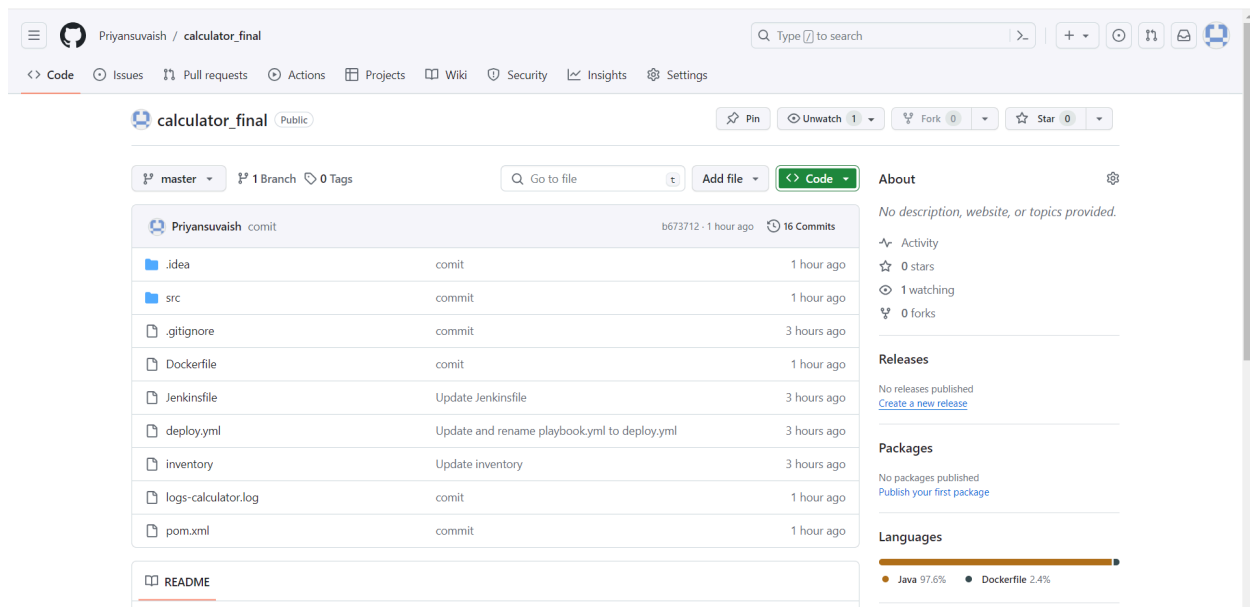
- Create a GitHub account and repository.
- Initialize Git in your project directory: `git init`

### Commands Used:

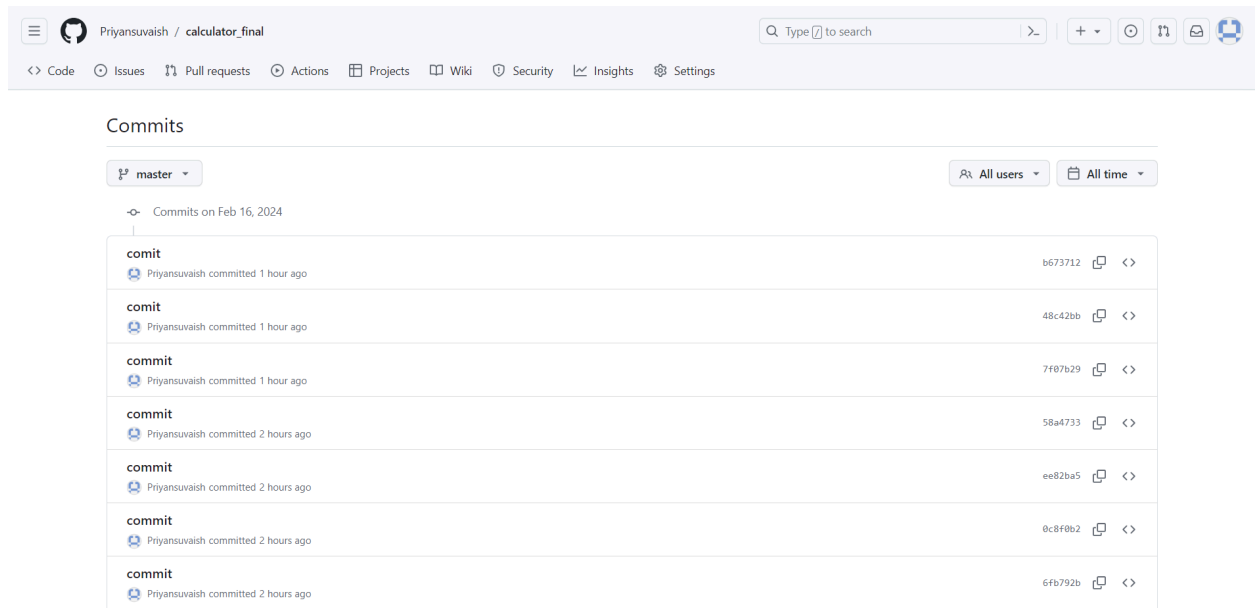
- `git add.`
- `git commit -m "<commit-message>"`
- `git push origin master`
- `git clone "https://github.com/Priyansuvaish/calculator\_final.git"`

### ScreenShot:

### Repository image:



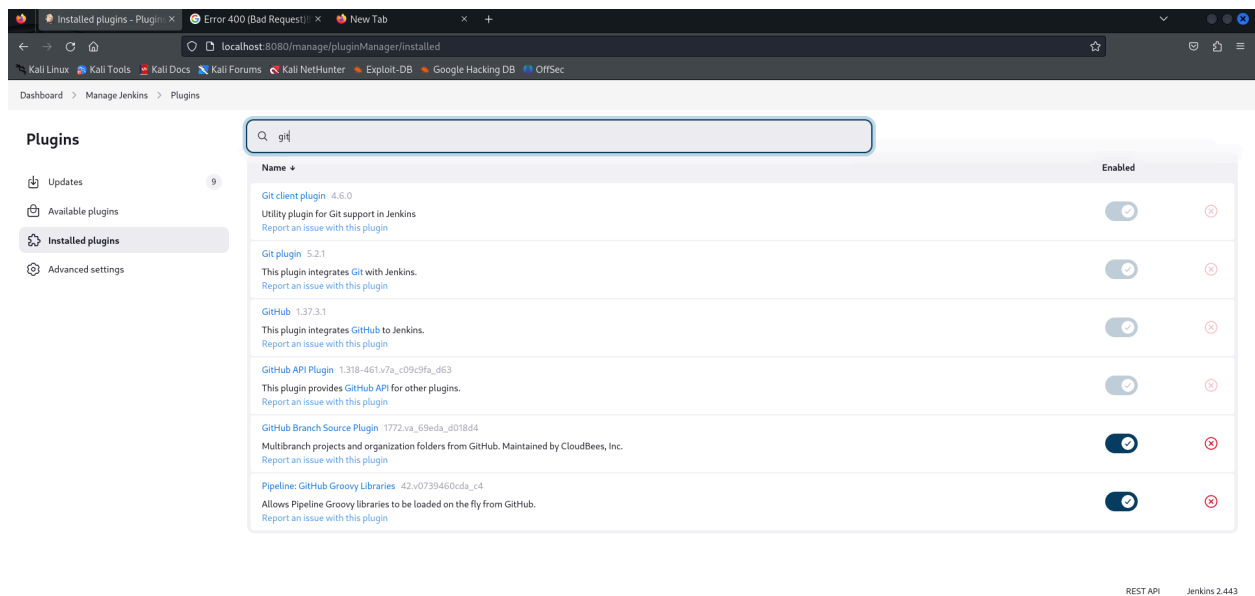
## Commit image:



The screenshot shows a GitHub repository page for 'calculator\_final' by user 'Priyansuvaish'. The 'Commits' tab is selected, showing a list of commits on the 'master' branch. The commits are sorted by 'All users' and 'All time'. The list shows seven commits, all by 'Priyansuvaish', with commit hashes and timestamps.

| Commit Hash | Author        | Time Ago    |
|-------------|---------------|-------------|
| b673712     | Priyansuvaish | 1 hour ago  |
| 48c42bb     | Priyansuvaish | 1 hour ago  |
| 7f07b29     | Priyansuvaish | 1 hour ago  |
| 58a4733     | Priyansuvaish | 2 hours ago |
| ee82ba5     | Priyansuvaish | 2 hours ago |
| 0c8f0b2     | Priyansuvaish | 2 hours ago |
| 6fb792b     | Priyansuvaish | 2 hours ago |

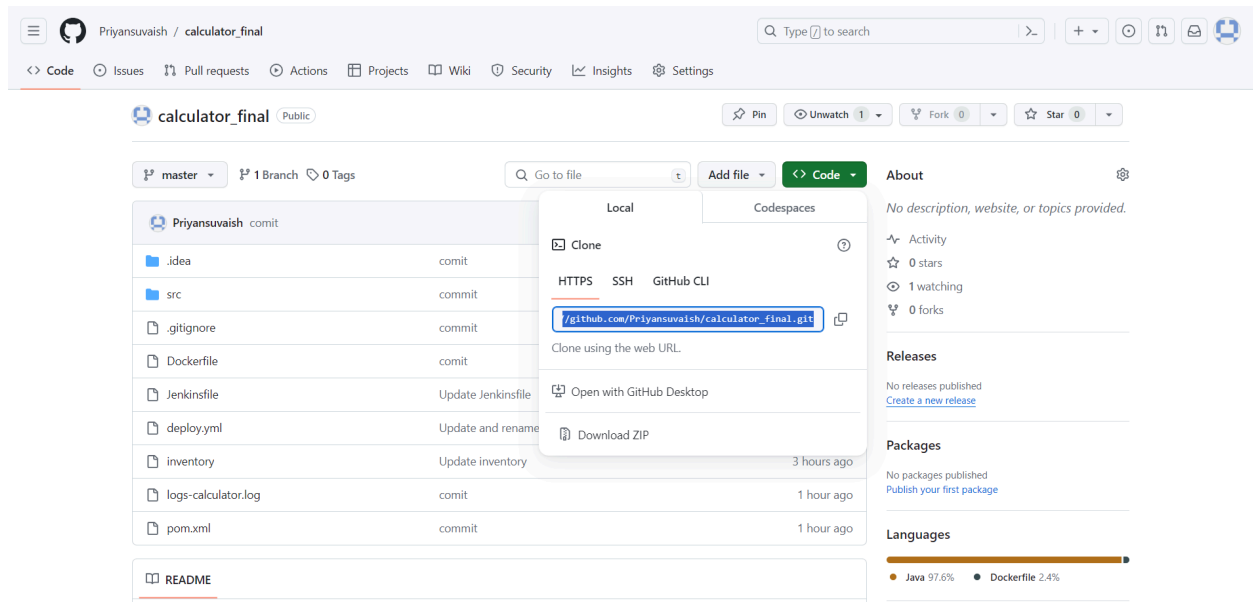
## Install the GitHub plugins:



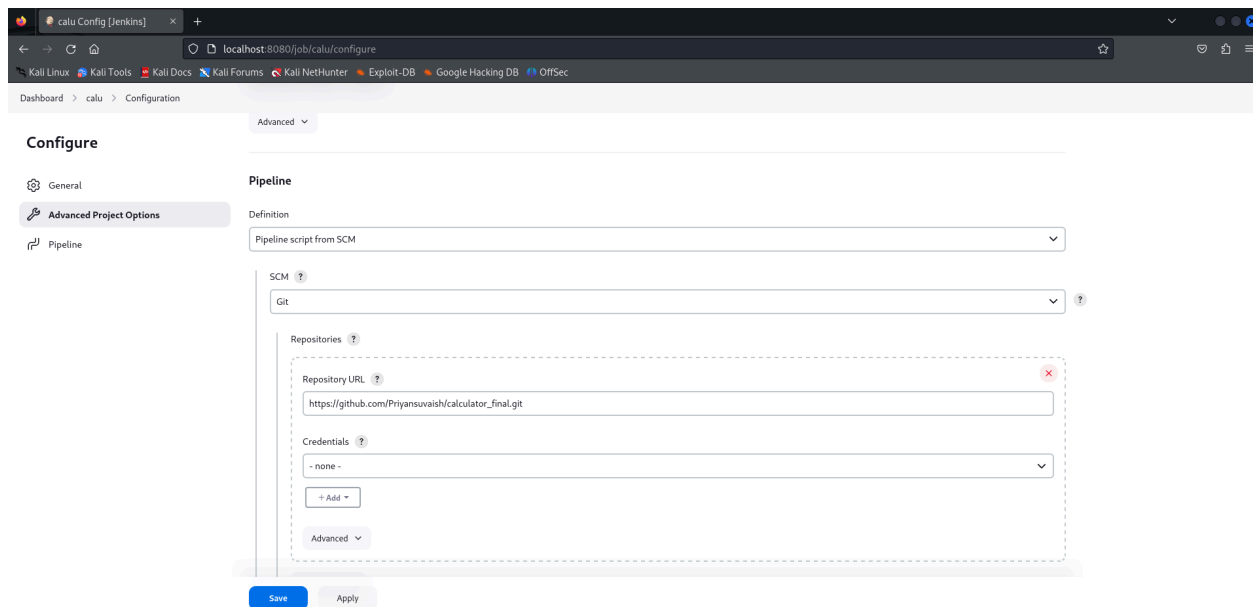
The screenshot shows the 'Manage Jenkins' page in a web browser. The 'Plugins' tab is selected, displaying a list of installed plugins. The search bar contains 'git'. The list includes plugins like 'Git client plugin', 'Git plugin', 'GitHub', 'GitHub API Plugin', 'GitHub Branch Source Plugin', and 'Pipeline: GitHub Groovy Libraries'. Each plugin has a description, version, and a status indicator (enabled/disabled).

| Name                              | Version                   | Enabled |
|-----------------------------------|---------------------------|---------|
| Git client plugin                 | 4.6.0                     | Enabled |
| Git plugin                        | 5.2.1                     | Enabled |
| GitHub                            | 1.37.3.1                  | Enabled |
| GitHub API Plugin                 | 1.318-461.v7a_c09c9fa_d63 | Enabled |
| GitHub Branch Source Plugin       | 1772.va_69eda_4018d4      | Enabled |
| Pipeline: GitHub Groovy Libraries | 42.v0739460cda_c4         | Enabled |

Copy the GitHub repository URL:



Paste the URL in the Jenkins project:



# Building the code with Maven and Testing with JUnit

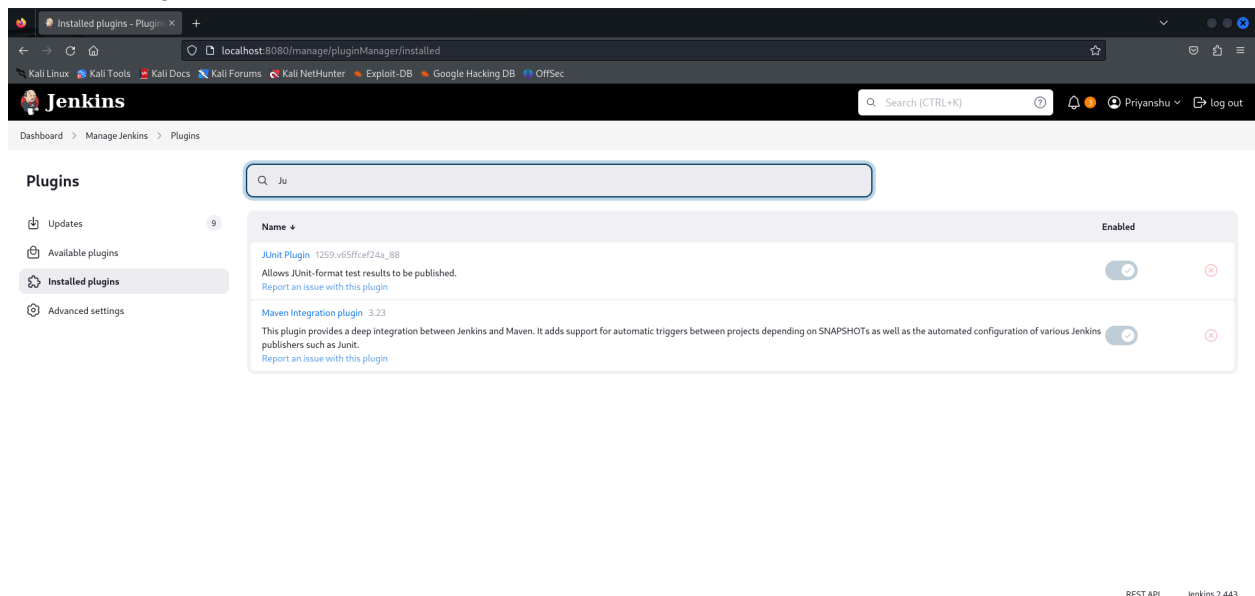
Maven is a powerful build tool for Java projects, simplifying the process of project building, dependency management, and more.

JUnit is a popular framework for testing Java applications, allowing you to write and run repeatable tests.

## Setting Up Maven and JUnit

IntelliJ IDEA comes with Maven support out of the box. To configure Maven for a new or existing project, simply open the project in IntelliJ. IntelliJ should automatically recognize the pom.xml file for existing Maven projects and import the necessary configurations.

Download the maven integration plugin, JUnit plugin and Oracle Java SE Development Kit Installer Plugin in Jenkins:



## Update the pom file for Building and Testing

In the pom.xml file, we define the project's dependencies, plugins, and other configurations. Here, we specify the version of JUnit that we use for testing, among other things.



Screenshot of the Dependencies:

```
<?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>1.8</maven.compiler.
source> <!-- Updated to 1.8 as 21 is not a standard
version as of my last update -->
    <maven.compiler.target>1.8</maven.compiler.
target> <!-- Updated to 1.8 as well -->
    <project.build.sourceEncoding>UTF-8</project.
build.sourceEncoding>
    <junit.jupiter.version>5.7.0</junit.jupiter.
version>
    <log4j2.version>2.22.1</log4j2.version>
  </properties>

  <dependencies>
    <!-- JUnit Jupiter (JUnit 5) dependencies for
testing -->
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>${junit.jupiter.version}</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</
artifactId>
      <version>${junit.jupiter.version}</version>
      <scope>test</scope>
    </dependency>
```

```
<!-- Log4j2 dependencies for logging -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
  <version>${log4j2.version}</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>${log4j2.version}</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.1</version>
  <scope>test</scope>
</dependency>
</dependencies>

<build>
  <plugins>
    <!-- Maven Compiler Plugin to ensure Java
version compatibility -->
    <plugin>
      <groupId>org.apache.maven.plugins</
groupId>
      <artifactId>maven-compiler-plugin</
artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>${maven.compiler.source}</
source>
        <target>${maven.compiler.target}</
target>
      </configuration>
    </plugin>
    <!-- Maven Assembly Plugin for creating a
jar with dependencies -->
    <plugin>
      <groupId>org.apache.maven.plugins</
groupId>
```

```

        <artifactId>maven-assembly-plugin</
artifactId>
        <version>3.3.0</version> <!-- Update to
the latest version available -->
        <executions>
            <execution>
                <phase>package</phase>
                <goals>
                    <goal>single</goal>
                </goals>
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>org.
example.calculator</mainClass>
                        </manifest>
                    </archive>
                    <descriptorRefs>
                        <descriptorRef>jar-with
-dependencies</descriptorRef>
                    </descriptorRefs>
                </configuration>
            </execution>
        </executions>
    </plugin>
</plugins>
</build>
</project>

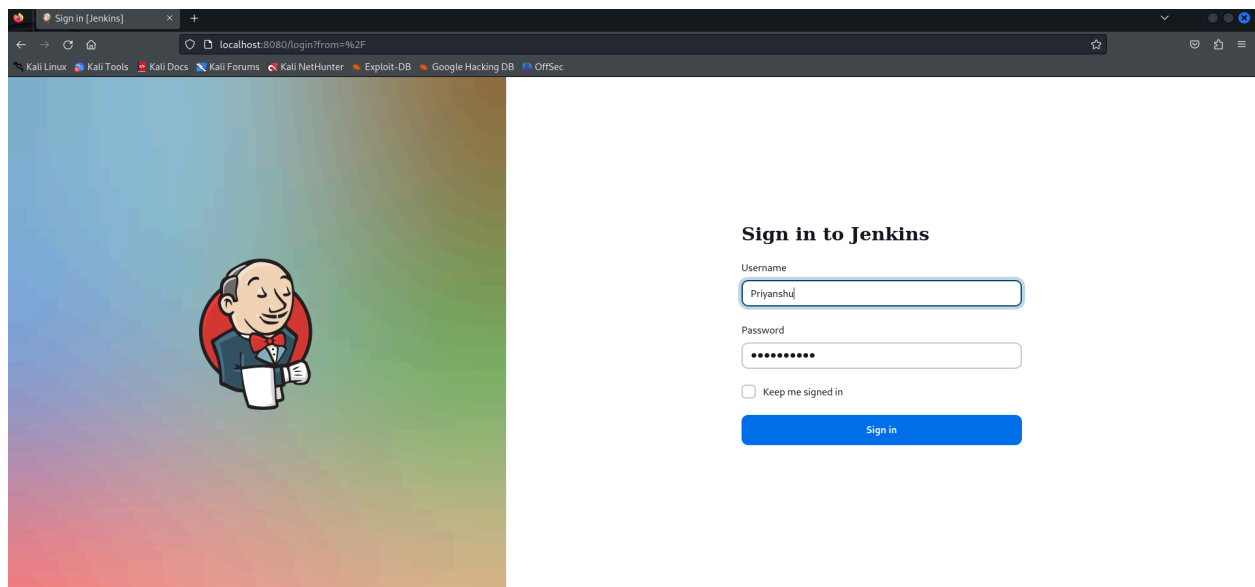
```

# Continuous Integration/Continuous Deployment (CI/CD): Jenkins

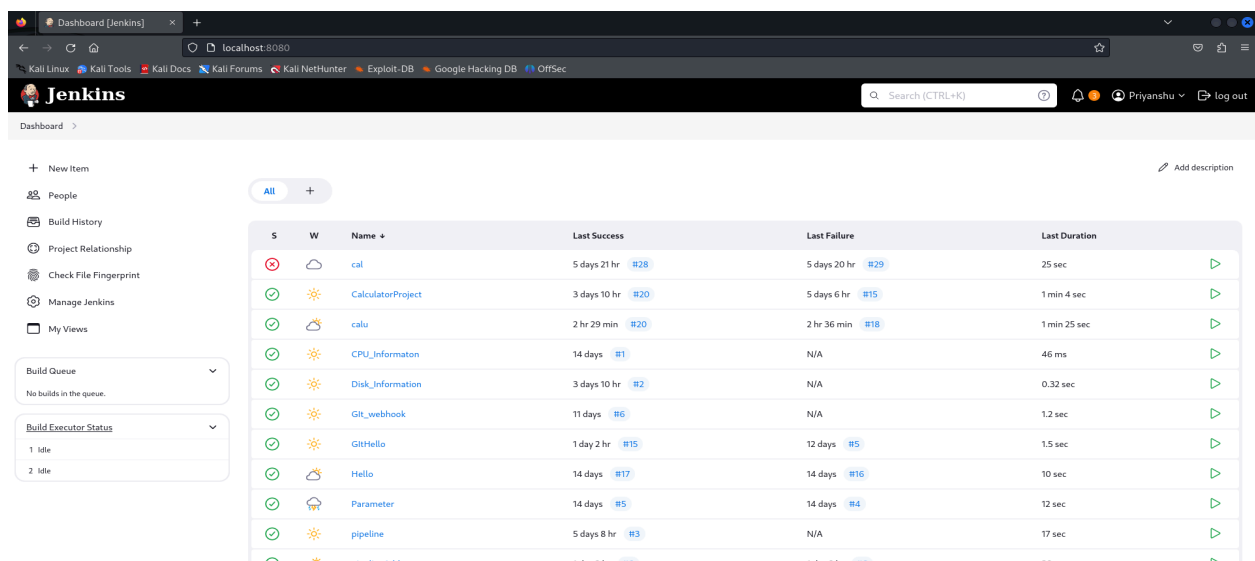
Jenkins automates the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

## Setup and Configurations:

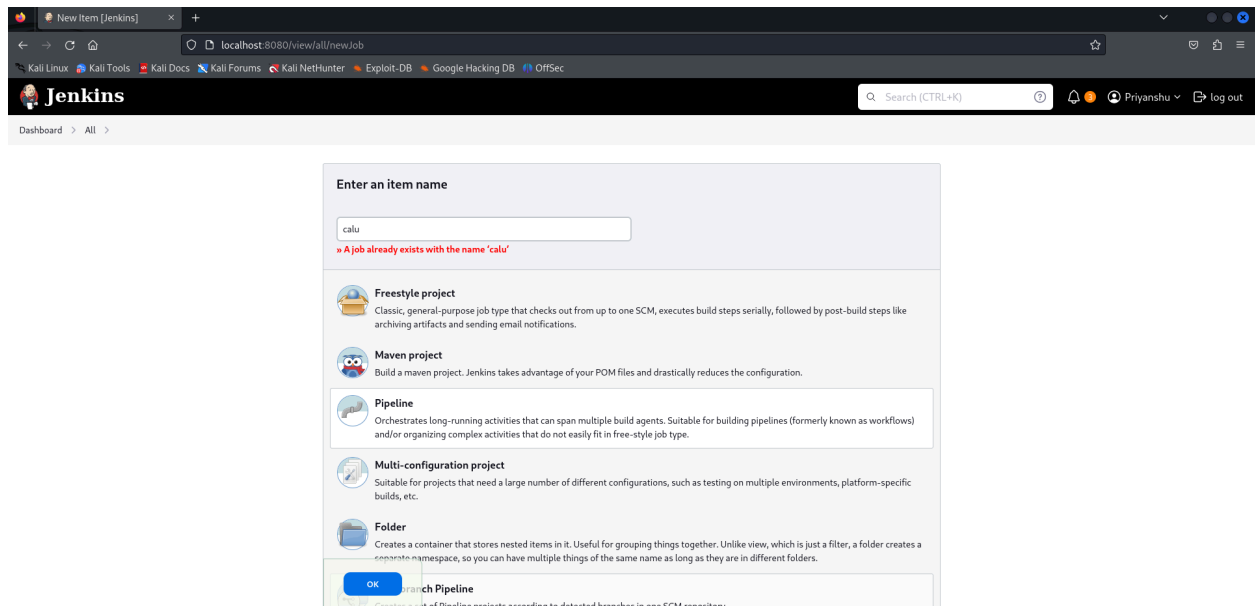
- Install Jenkins on the local machine.
- Login the Jenkins using the credentials.



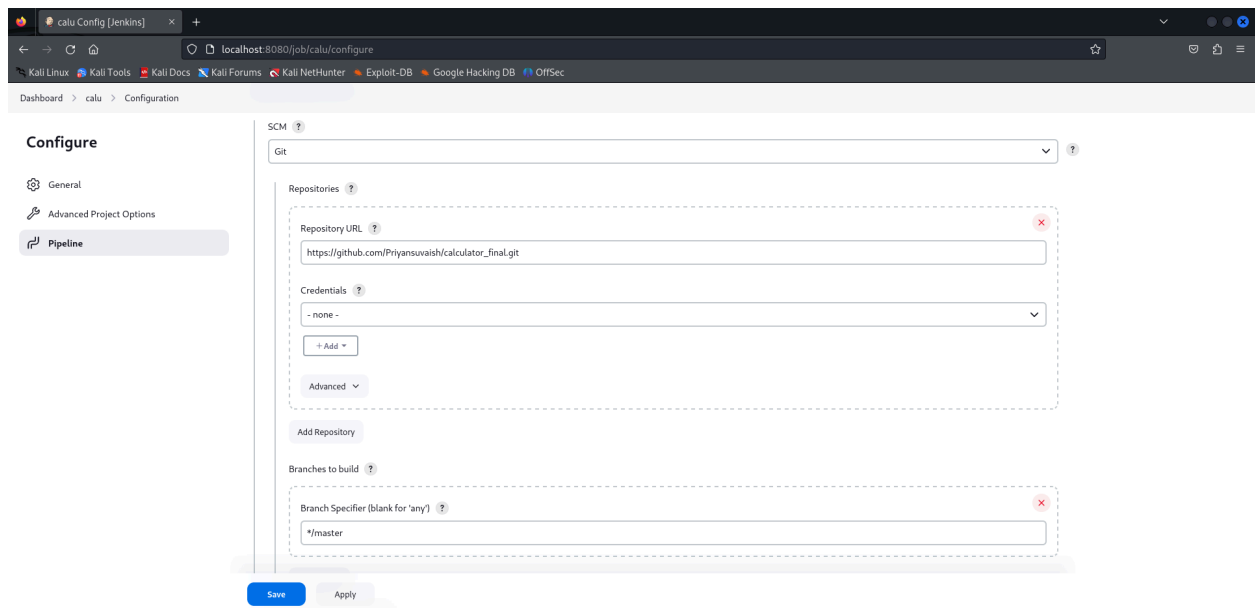
- After login, the welcome screen or the previous jobs dashboard comes.



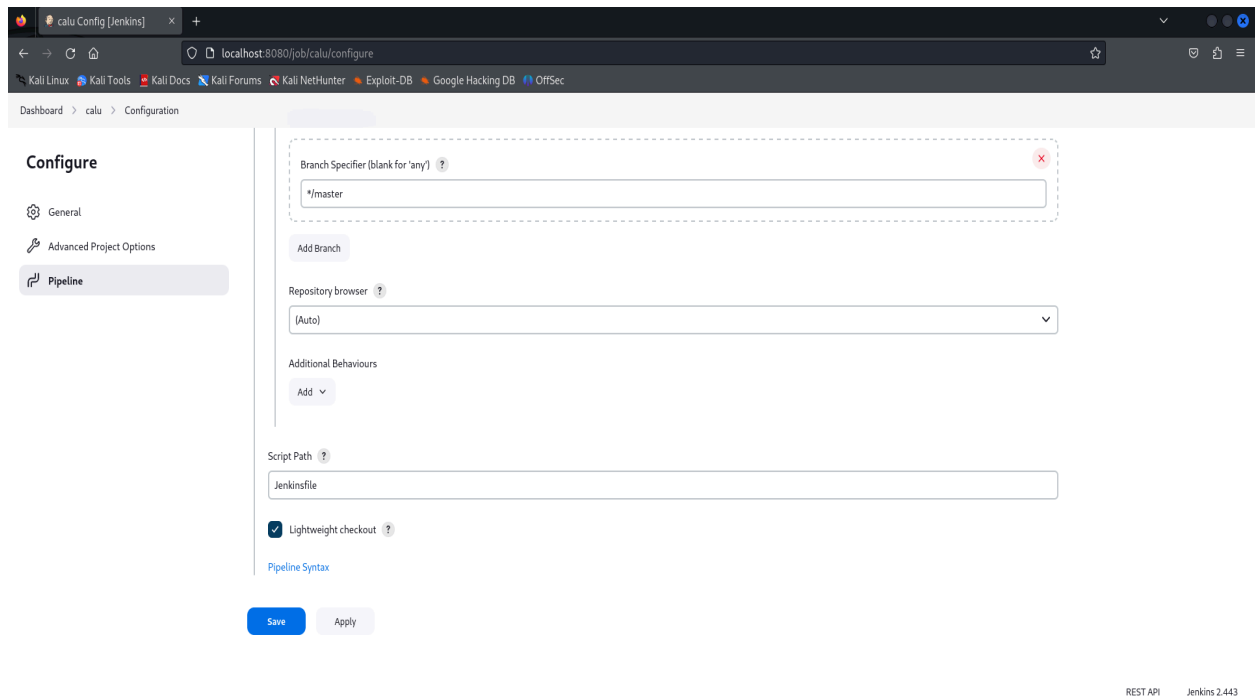
- Create the new job as the pipeline.



- Set up the git SCM and give the branch of the project in GitHub.



- Give the Jenkinsfile name in the Script Path.



## Screenshots of the Jenkins Pipeline Script

```

pipeline {
    agent any

    tools {
        maven "Maven"
        jdk "jdk"
    }
    environment {
        DOCKER_IMAGE_NAME = 'calculator'
        GITHUB_REPO_URL = 'https://github.com/Priyansuvaish/calculator_final.git'
    }
}

```

In this, we set up the tools and environment variables that we need in different stages.

Now the stages of the Jenkins file are as follows:

```
stages {
    stage('Checkout') {
        steps {
            script {
                // Checkout the code from the GitHub repository
                git branch: 'master', url: "${GITHUB_REPO_URL}"
            }
        }
    }

    stage('Build') {
        steps {
            // Run Maven build
            script {
                mvnHome = tool 'Maven' // Retrieves the Maven installation configured in Jenkins global tools
                sh "${mvnHome}/bin/mvn clean package"
            }
        }
    }

    stage('Test') {
        steps {
            // Run tests
            script {
                sh "${mvnHome}/bin/mvn test"
            }
        }
    }

    stage('Archive') {
        steps {
            // Archive the built artifacts
            archiveArtifacts artifacts: '**/target/*.jar', fingerprint: true
        }
    }

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

---

```

    stage('Push Docker Images') {
        steps {
            script{
                docker.withRegistry('', 'de42dccd-1664-4127-a754-d681873bedec') {
                    sh 'docker tag calculator priyanshugupta753/calculator:latest'
                    sh 'docker push priyanshugupta753/calculator'
                }
            }
        }
    }

    stage('Run Ansible Playbook') {
        steps {
            script {
                ansiblePlaybook(
                    playbook: 'deploy.yml',
                    inventory: 'inventory'
                )
            }
        }
    }
}

post {
    // Define actions to take in case of build failure or success
    success {
        echo 'Build succeeded.'
    }
    failure {
        echo 'Build failed.'
    }
}
}

```

---

## Containerization with Docker

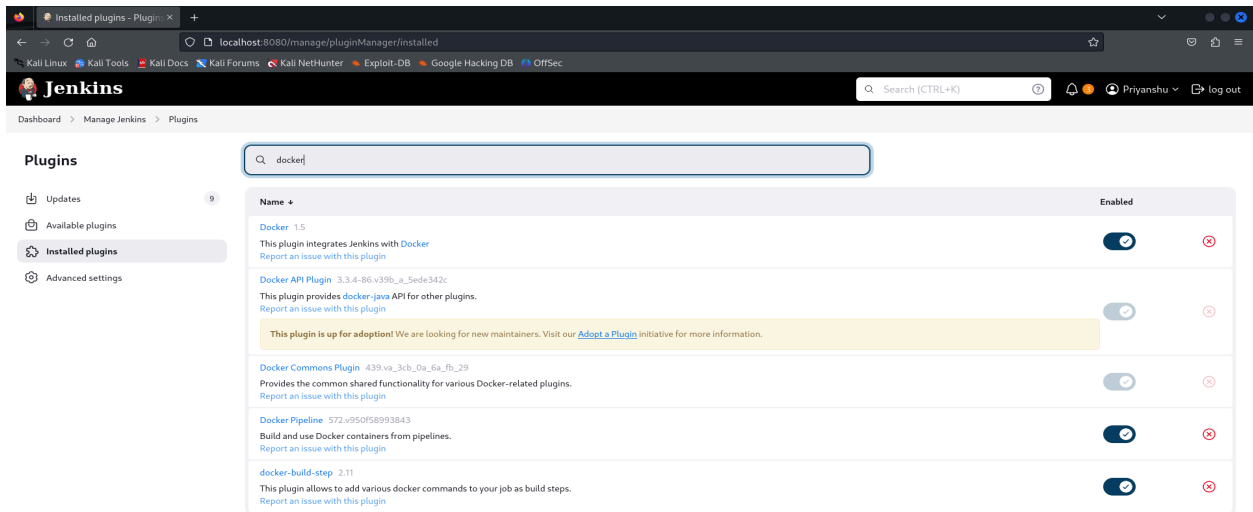
Docker is a set of platform-as-a-service products that use OS-level virtualization to deliver software in packages called containers.

### Setup and Configurations:

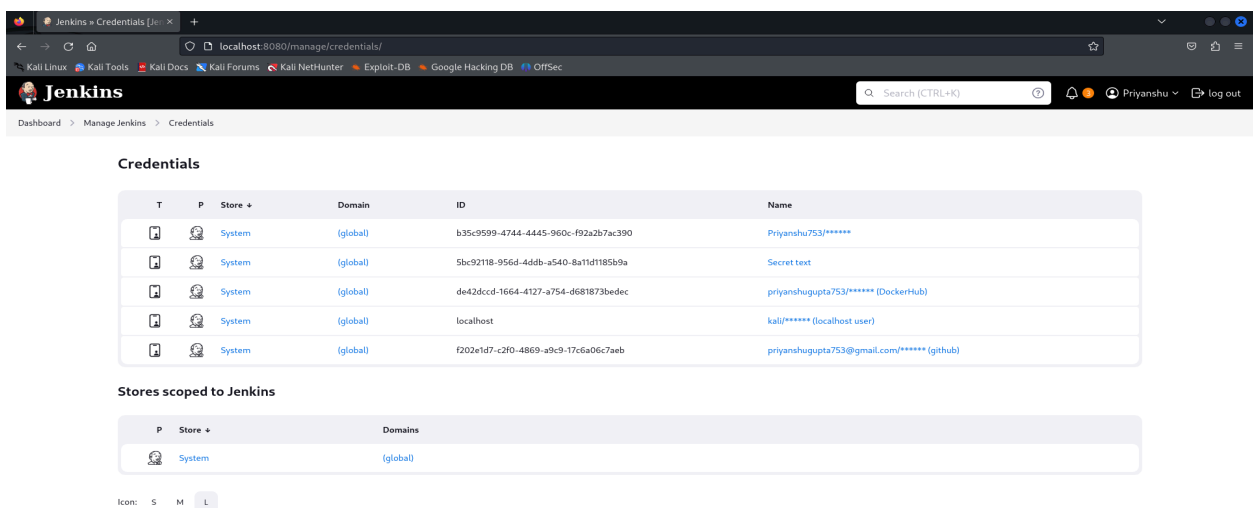
- Install Docker on the local machine.



- Install the plugins in Jenkins



- In the credential give the docker username and password.



- Copy the ID that you give in the credential and put it in the Jenkins file in the command:

```
stage('Push Docker Images') {
    steps {
        script{
            docker.withRegistry('', 'de42dcd-1664-4127-a754-d681873bedec') {
                sh 'docker tag calculator priyanshugupta753/calculator:latest'
                sh 'docker push priyanshugupta753/calculator'
            }
        }
    }
}
```

## ScreenShot of Docker file script

```
FROM openjdk:8
COPY ./target/Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java", "-jar", "Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

## Deployment using Ansible

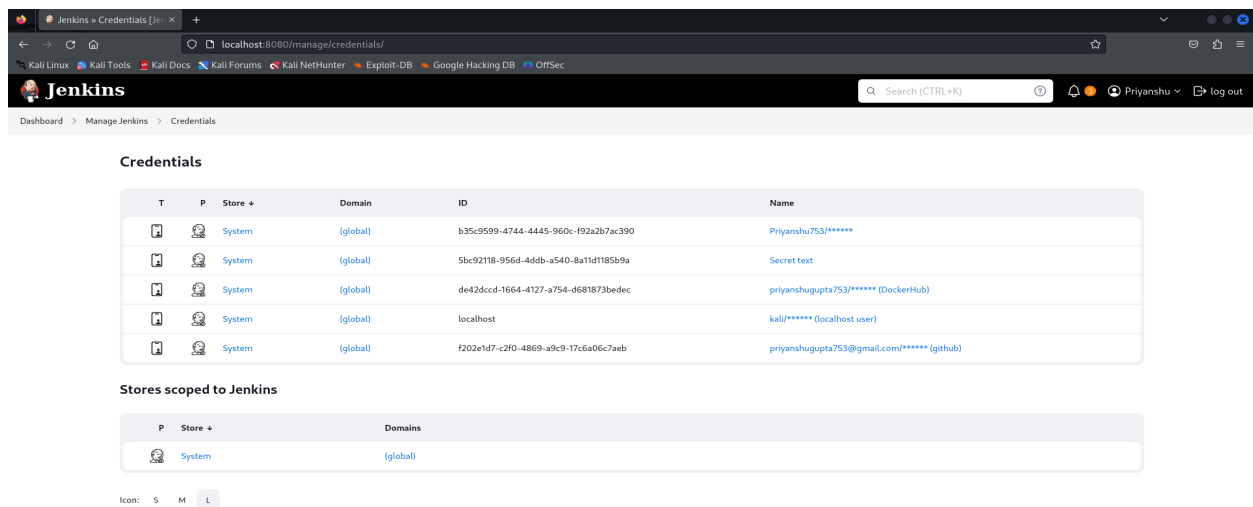
Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code.

Setup and Configurations:

- Install Ansible on the local machine.
- Configure inventory file is as follows:

```
ansible_host=localhost ansible_user=kali ansible_ssh_pass=kali
# ansible_ssh_common_args = '-o StrictHostKeyChecking=no'
```

- In the credential give the local machine username and password.



The screenshot shows the Jenkins web interface at localhost:8080/manage/credentials/. The page title is "Jenkins » Credentials [Jenkins]". The breadcrumb navigation is "Dashboard > Manage Jenkins > Credentials".

**Credentials**

| T | P | Store  | Domain   | ID                                   | Name                                       |
|---|---|--------|----------|--------------------------------------|--|
|   |   | System | (global) | b35c9599-4744-4445-960c-f92a2b7ac390 | Priyanshu753/*****                         |
|   |   | System | (global) | 5bc92118-956d-4ddb-a540-8a11d1185b9a | Secret text                                |
|   |   | System | (global) | de42dcd-1664-4127-a754-d681873bedec  | priyanshugupta753/***** (DockerHub)        |
|   |   | System | (global) | localhost                            | kali/***** (localhost user)                |
|   |   | System | (global) | f202e1d7-c2f0-4869-a9c9-17c6a06c7aeb | priyanshugupta753@gmail.com/***** (github) |

**Stores scoped to Jenkins**

| P | Store  | Domains  |
|---|--------|----------|
|   | System | (global) |

Icon: S M L

## ScreenShot of the playbook script

---

```
---
- name: Pull Docker Image from Docker Hub
  hosts: localhost
  remote_user: kali
  become: false
  tasks:
    - name: Pull Docker Image
      docker_image:
        name: "priyanshugupta753/calculator"
        source: pull
      register: docker_pull_result

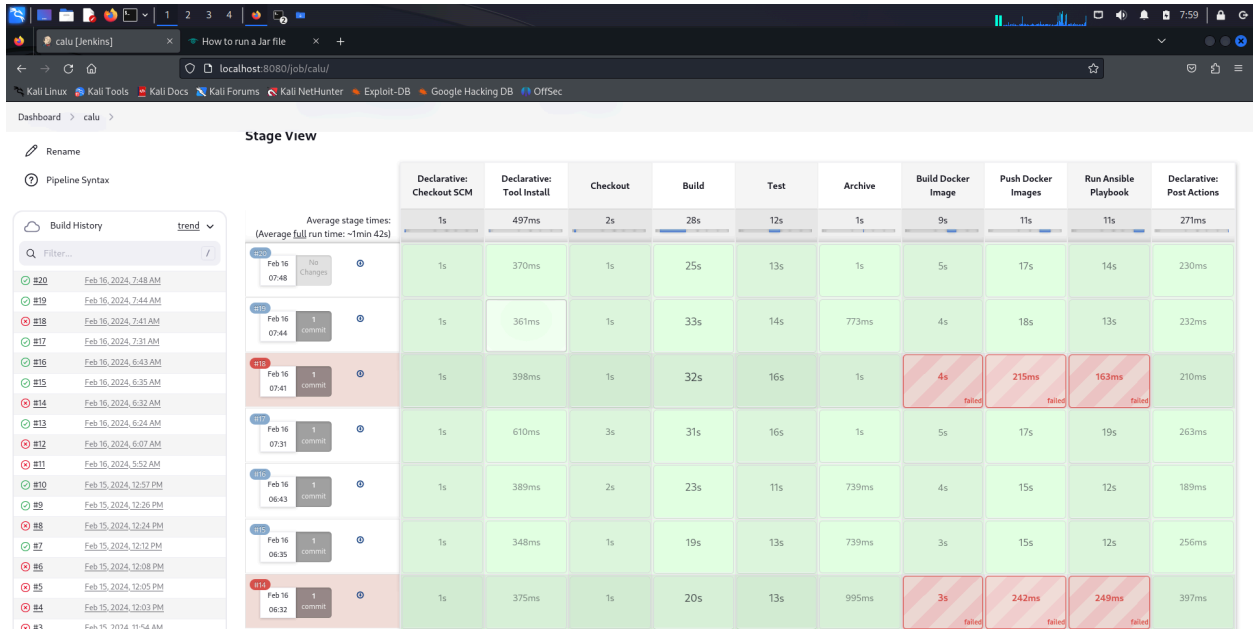
    - name: Display Docker Pull Result
      debug:
        var: docker_pull_result

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

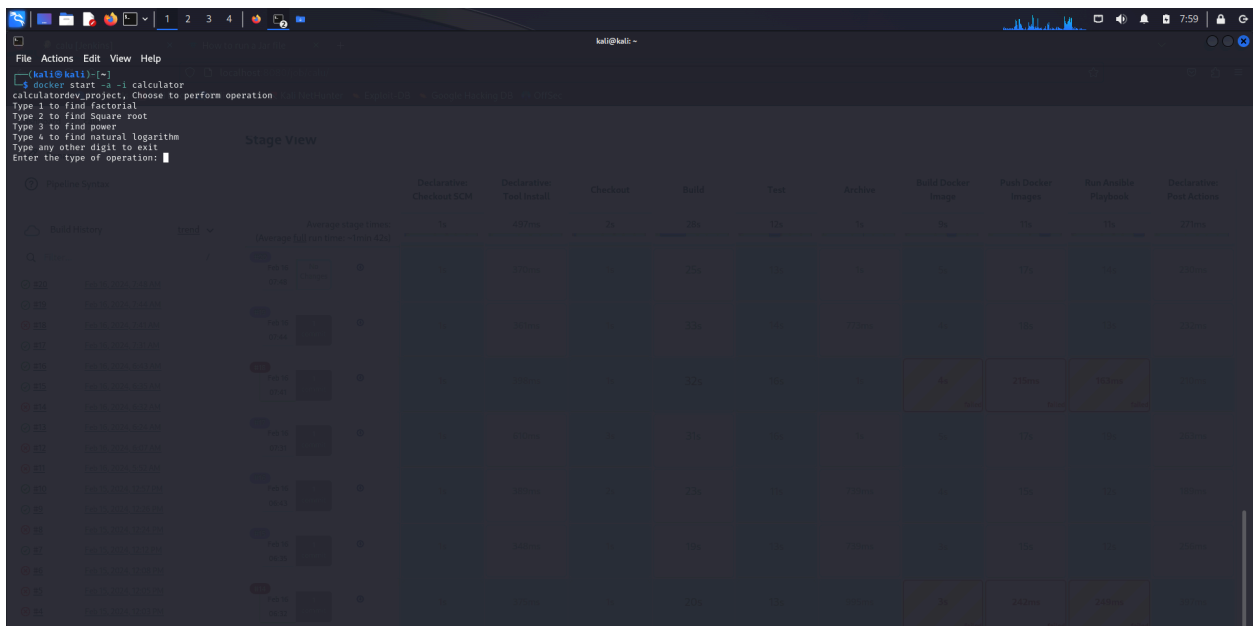
    - name: Running container
      shell: docker run -it -d --name calculator priyanshugupta753/calculator
```

# Working of the application after the successfully deployed:

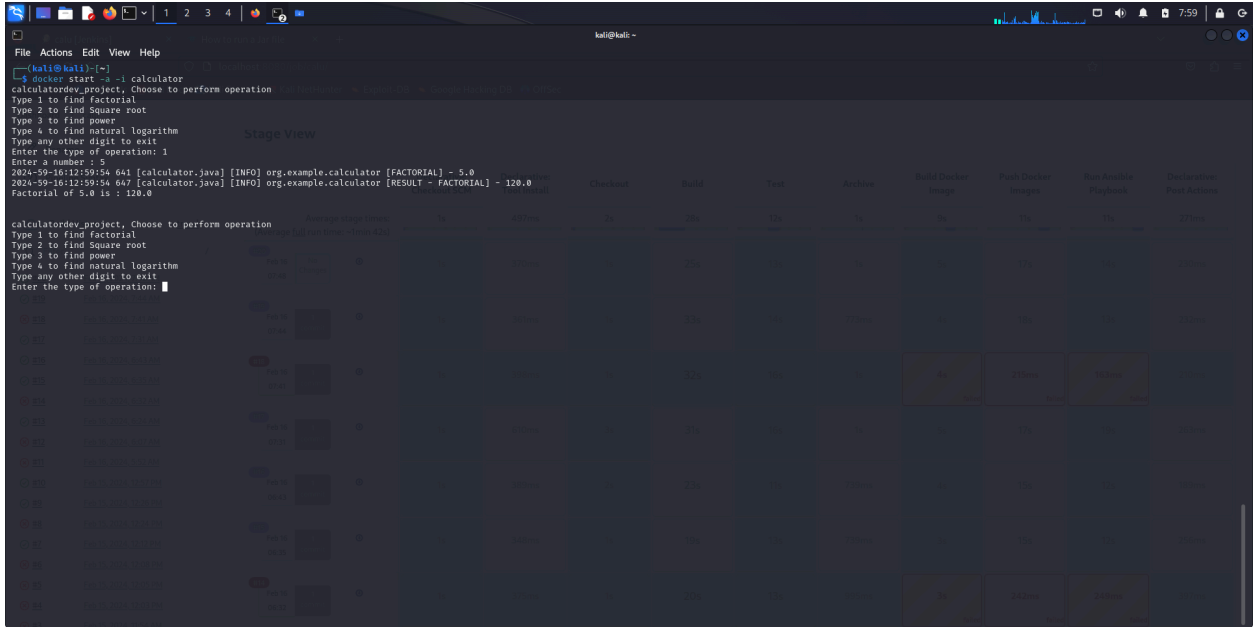
- All the stages are built successfully



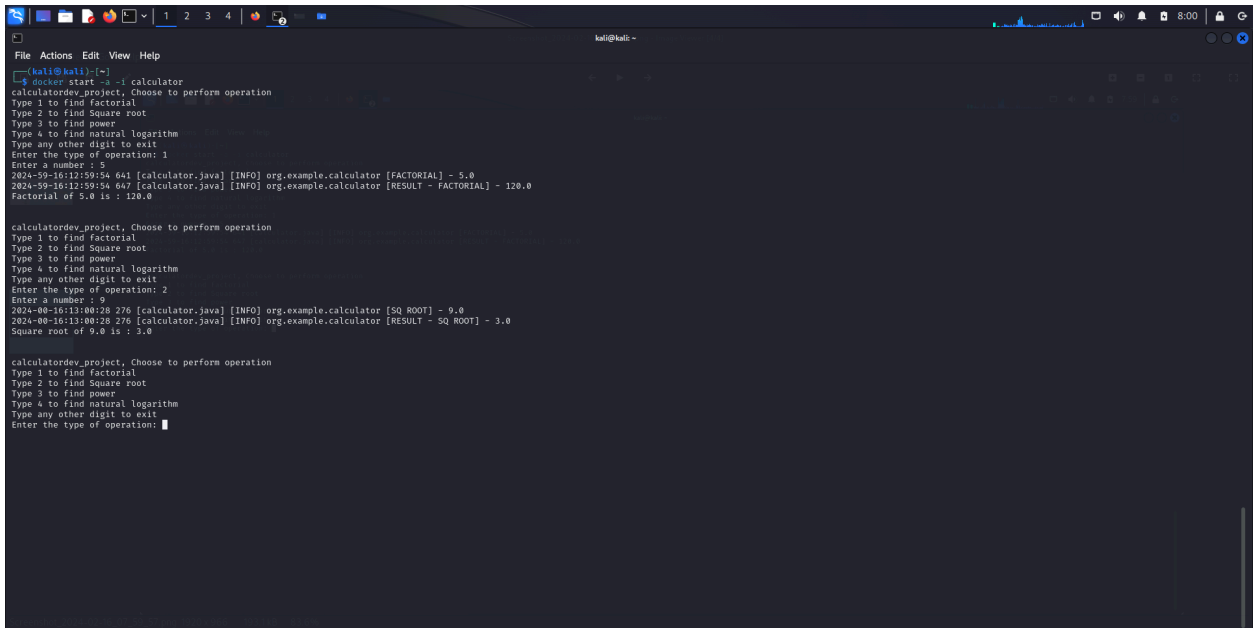
- Running the docker container:



- Finding the factorial of the number



- Finding the square root of the number



- Finding the power of the number

```
File Actions Edit View Help
--(kali@kali)-[~]
└─$ docker start -a -l calculator
calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 1
Enter a number : 5
2024-09-16:12:59:54 641 [calculator.java] [INFO] org.example.calculator [FACTORIAL] - 5.0
2024-09-16:12:59:54 647 [calculator.java] [INFO] org.example.calculator [RESULT - FACTORIAL] - 120.0
Factorial of 5.0 is : 120.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 2
Enter a number : 9
2024-09-16:13:00:28 276 [calculator.java] [INFO] org.example.calculator [SQ ROOT] - 9.0
2024-09-16:13:00:28 276 [calculator.java] [INFO] org.example.calculator [RESULT - SQ ROOT] - 3.0
Square root of 9.0 is : 3.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 3
Enter the first number : 3
Enter the second number : 3
2024-09-16:13:00:43 293 [calculator.java] [INFO] org.example.calculator [POWER - 3.0 RAISED TO] 3.0
2024-09-16:13:00:43 294 [calculator.java] [INFO] org.example.calculator [RESULT - POWER] - 27.0
3.0 raised to power 3.0 is : 27.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: █
```

- Finding the natural log of the number

```
File Actions Edit View Help
Enter a number : 5
2024-09-16:12:59:54 641 [calculator.java] [INFO] org.example.calculator [FACTORIAL] - 5.0
2024-09-16:12:59:54 647 [calculator.java] [INFO] org.example.calculator [RESULT - FACTORIAL] - 120.0
Factorial of 5.0 is : 120.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 2
Enter a number : 9
2024-09-16:13:00:28 276 [calculator.java] [INFO] org.example.calculator [SQ ROOT] - 9.0
2024-09-16:13:00:28 276 [calculator.java] [INFO] org.example.calculator [RESULT - SQ ROOT] - 3.0
Square root of 9.0 is : 3.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 3
Enter the first number : 3
Enter the second number : 3
2024-09-16:13:00:43 293 [calculator.java] [INFO] org.example.calculator [POWER - 3.0 RAISED TO] 3.0
2024-09-16:13:00:43 294 [calculator.java] [INFO] org.example.calculator [RESULT - POWER] - 27.0
3.0 raised to power 3.0 is : 27.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 4
Enter a number : 8
2024-09-16:13:00:57 607 [calculator.java] [INFO] org.example.calculator [NATURAL LOG] - 8.0
2024-09-16:13:00:57 607 [calculator.java] [INFO] org.example.calculator [RESULT - NATURAL LOG] - 2.0794415416798357
Natural log of 8.0 is : 2.0794415416798357

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: █
```

- Type any other digit to exit.

```
File Actions Edit View Help
Factorial of 5.0 is : 120.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 2
Enter a number : 9
2024-00-16:11:00:28 276 [calculator.java] [INFO] org.example.calculator [SQ ROOT] - 9.0
2024-00-16:11:00:28 276 [calculator.java] [INFO] org.example.calculator [RESULT - SQ ROOT] - 3.0
Square root of 9.0 is : 3.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 3
Enter the first number : 3
Enter the second number : 3
2024-00-16:11:00:43 292 [calculator.java] [INFO] org.example.calculator [POWER - 3.0 RAISED TO] 3.0
2024-00-16:11:00:43 294 [calculator.java] [INFO] org.example.calculator [RESULT - POWER] - 27.0
3.0 raised to power 3.0 is : 27.0

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: 4
Enter a number : 8
2024-00-16:11:00:57 607 [calculator.java] [INFO] org.example.calculator [NATURAL LOG] - 8.0
2024-00-16:11:00:57 607 [calculator.java] [INFO] org.example.calculator [RESULT - NATURAL LOG] - 2.0794415416798357
Natural log of 8.0 is : 2.0794415416798357

calculatordev_project, Choose to perform operation
Type 1 to find factorial
Type 2 to find Square root
Type 3 to find power
Type 4 to find natural logarithm
Type any other digit to exit
Enter the type of operation: e

kali@kali:~$
```

**Thank you**