Software Production Engineering

Mini Project 1

Name: Sparsh Salodkar

Roll Number: IMT2022113

Scientific Calculator

CI/CD Pipeline using Docker, Jenkins, and Ansible

Introduction

This report walks through a simple, reproducible CI/CD setup for a **Scientific Calculator** (CLI) application. The pipeline uses GitHub for source control, Jenkins for automation, Docker for packaging, and Ansible for local deployment. The idea is straightforward: *push code* $\rightarrow$ *Jenkins builds & tests* $\rightarrow$ *image pushed to Docker Hub* $\rightarrow$ *Ansible pulls the latest image and deploys*.

What and Why of DevOps

DevOps combines development and operations to shorten feedback loops and make releases swift. In this project:

- Every push runs the same build and tests on Jenkins.
- The app is containerized, so it runs the same everywhere.
- Deployment is scripted with Ansible, removing manual steps.

Tools Used

GitHub (SCM & webhooks), **Jenkins** (CI/CD), **Docker** (container image), **Ansible** (pull & run latest image), **Ngrok** (expose local Jenkins to GitHub for webhooks).

1. Source Control: Git & GitHub

The repository holds the Java source, tests, `Dockerfile`, `Jenkinsfile`, `ansible/` playbook, and a small `RUN.sh` helper to run the image locally.

Webhook (GitHub $\rightarrow$Jenkins)

In the repository settings, a webhook points to Jenkins' GitHub endpoint (via your active Ngrok URL). On every push, GitHub pings Jenkins, which triggers the pipeline on the latest commit.

Screenshots

Figure 1: Repository creation and initial push.

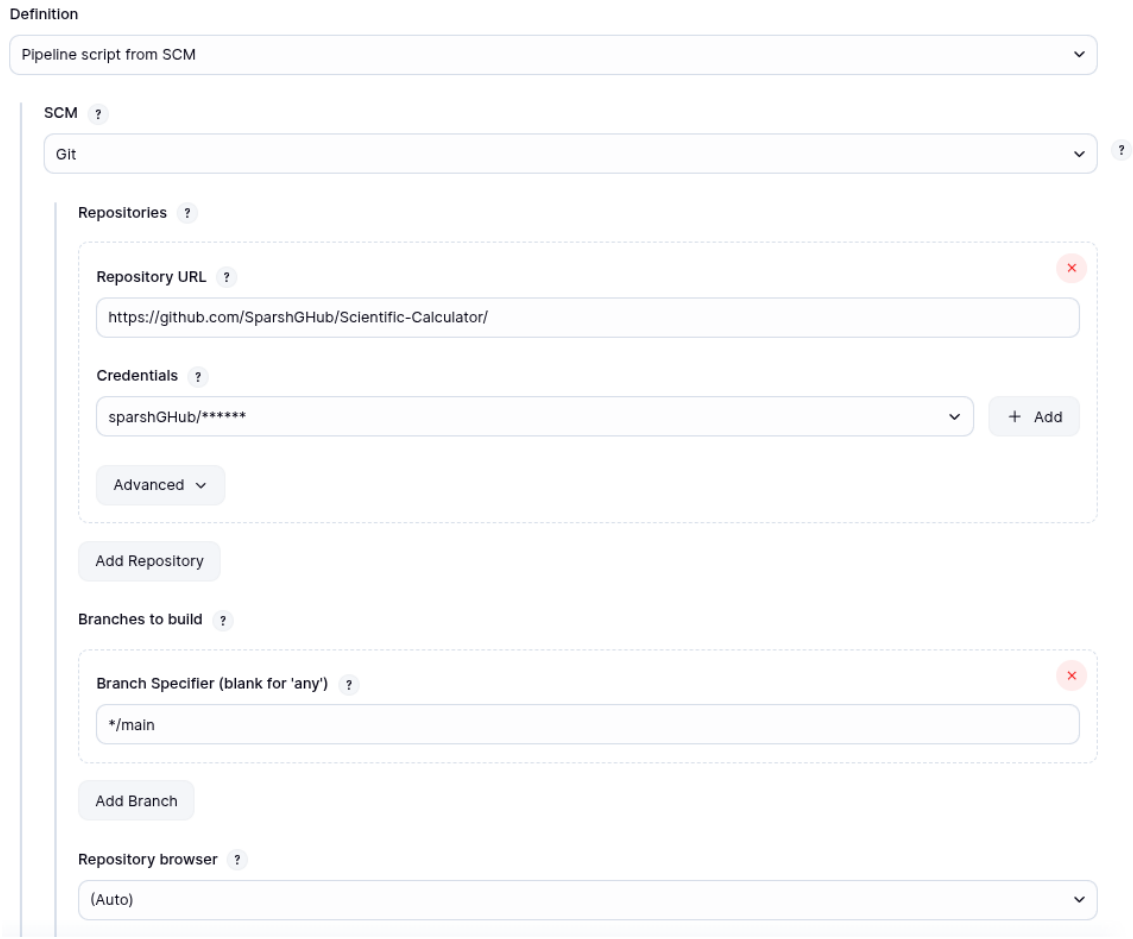

Figure 2: Jenkins job connected to the GitHub repo (credentials and URL).

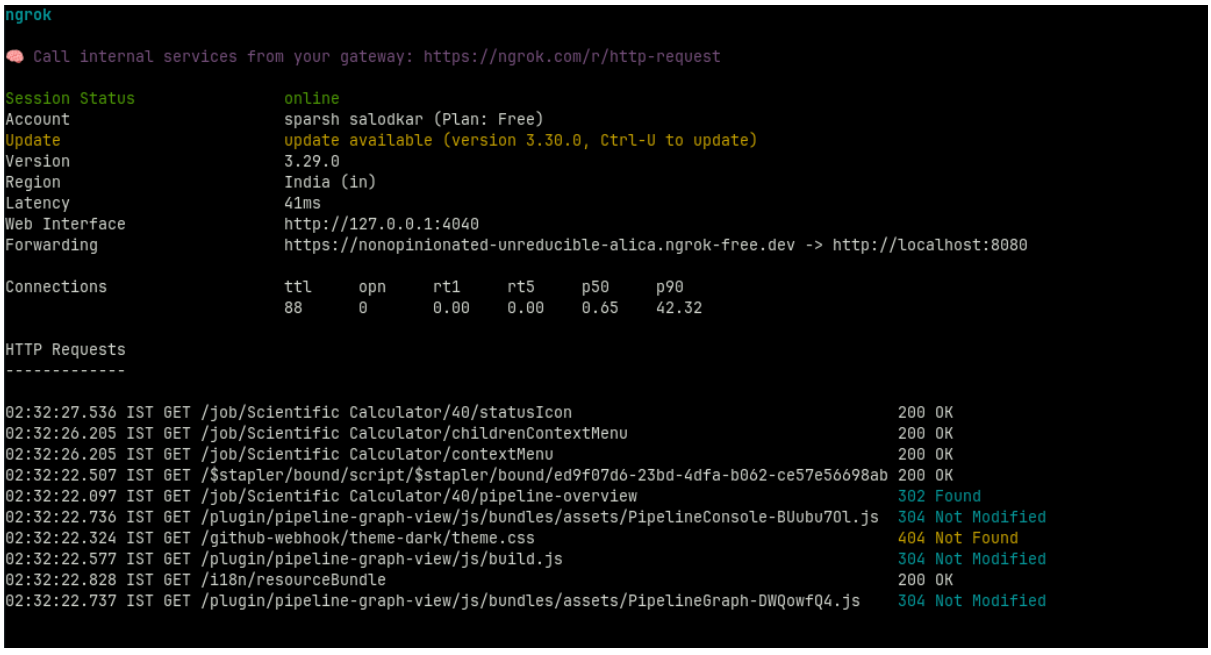

Figure 3: Ngrok exposing local Jenkins for GitHub webhooks.

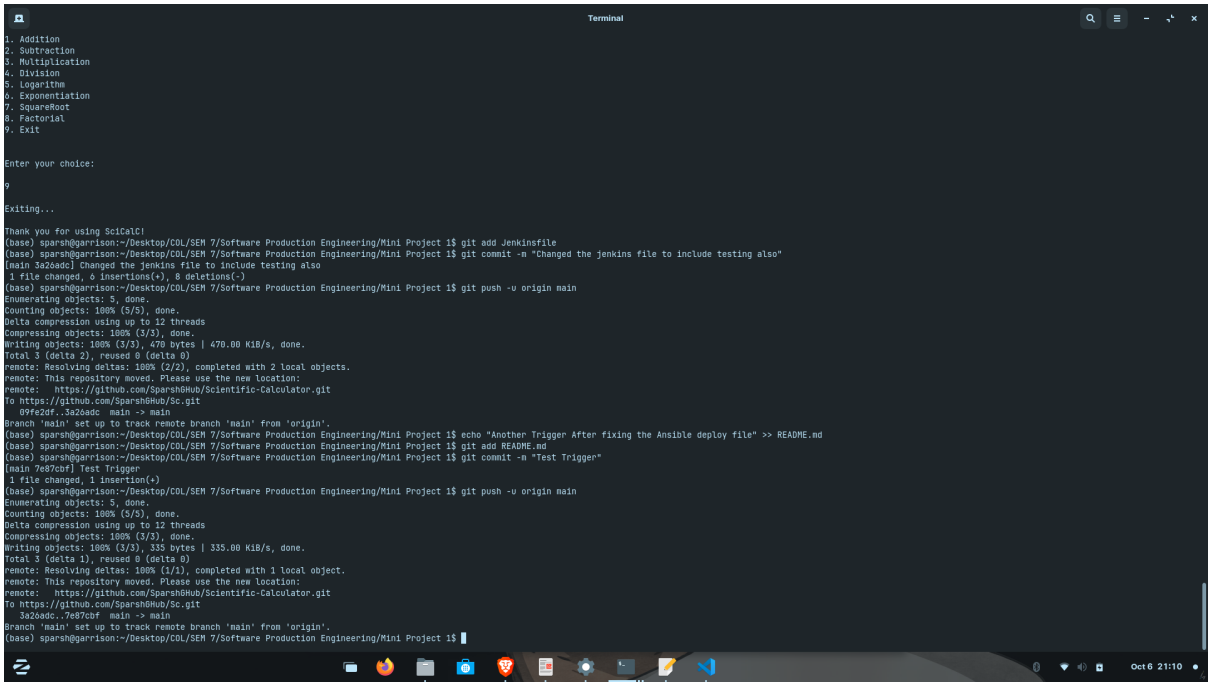

Figure 4: Test Trigger

2. Jenkins CI Pipeline (Stage-by-stage)

The pipeline below lives at `SciCalC/thescicalc/Jenkinsfile`. It runs on any agent with Java, Maven, Docker, and Ansible available.

Pipeline (final version)

```
pipeline {
  agent any
  options { timestamps(); timeout(time: 60, unit: 'MINUTES') }
  environment {
    DOCKER_IMAGE  = "sparshdockerman/scicalc"
    CREDENTIALS_ID = "dockerHubCreds"
  }
  stages {
    stage('Checkout') { steps { checkout scm } }
    stage('Verify Tools') {
      steps {
        sh 'java -version || true'
        sh 'mvn -v || true'
        sh 'docker --version || true'
        sh 'ansible --version || true'
      }
    }
    stage('Test') {
```

```
      steps { dir('SciCalC/thescicalc') { sh 'mvn -B test' } }
      post { always { junit 'SciCalC/thescicalc/target/surefire-reports
          /*.xml' } }
    }
    stage('Package') {
      steps { dir('SciCalC/thescicalc') { sh 'mvn -B package -
          DSkipTests' } }
      post { success { archiveArtifacts artifacts: 'SciCalC/thescicalc/
          target/*.jar', fingerprint: true } }
    }
    stage('Docker Build & Push') {
      steps {
        script {
          def sha = env.GIT_COMMIT ?: sh(script: "git rev-parse --short
              HEAD", returnStdout: true).trim()
          def imgLatest = "${DOCKER_IMAGE}:latest"
          def imgSha    = "${DOCKER_IMAGE}:${sha}"
          sh "docker build -f SciCalC/thescicalc/Dockerfile -t ${
              imgLatest} -t ${imgSha} SciCalC/thescicalc"
          withCredentials([usernamePassword(credentialsId: env.
              CREDENTIALS_ID,
                                            usernameVariable: '
                                                DOCKERHUB_USER',
                                            passwordVariable: '
                                                DOCKERHUB_TOKEN')]) {
            sh '''
              echo "$DOCKERHUB_TOKEN" | docker login -u "
                  $DOCKERHUB_USER" --password-stdin
              docker push ${DOCKER_IMAGE}:latest
              docker logout
            '''
          }
        }
      }
    }
    stage('Deploy with Ansible') {
      steps {
        dir('SciCalC/thescicalc/ansible') {
          withCredentials([usernamePassword(credentialsId: env.
              CREDENTIALS_ID,
                                            usernameVariable: '
                                                DOCKERHUB_USER',
                                            passwordVariable: '
                                                DOCKERHUB_TOKEN')]) {
            sh '''
              ansible-playbook -i inventory deploy.yml \
```

```
                --extra-vars "docker_image=${DOCKER_IMAGE}:latest
                    docker_user=$DOCKERHUB_USER docker_token=
                    $DOCKERHUB_TOKEN"
            '''
          }
        }
      }
    }
  }
  post {
    success { echo 'Build and deployment successful.' }
    failure { echo 'Build or deployment failed. Check console output.'
        }
    always  { echo 'Pipeline completed.' }
  }
}
```

What each stage does

Checkout pulls the exact commit that triggered the job.

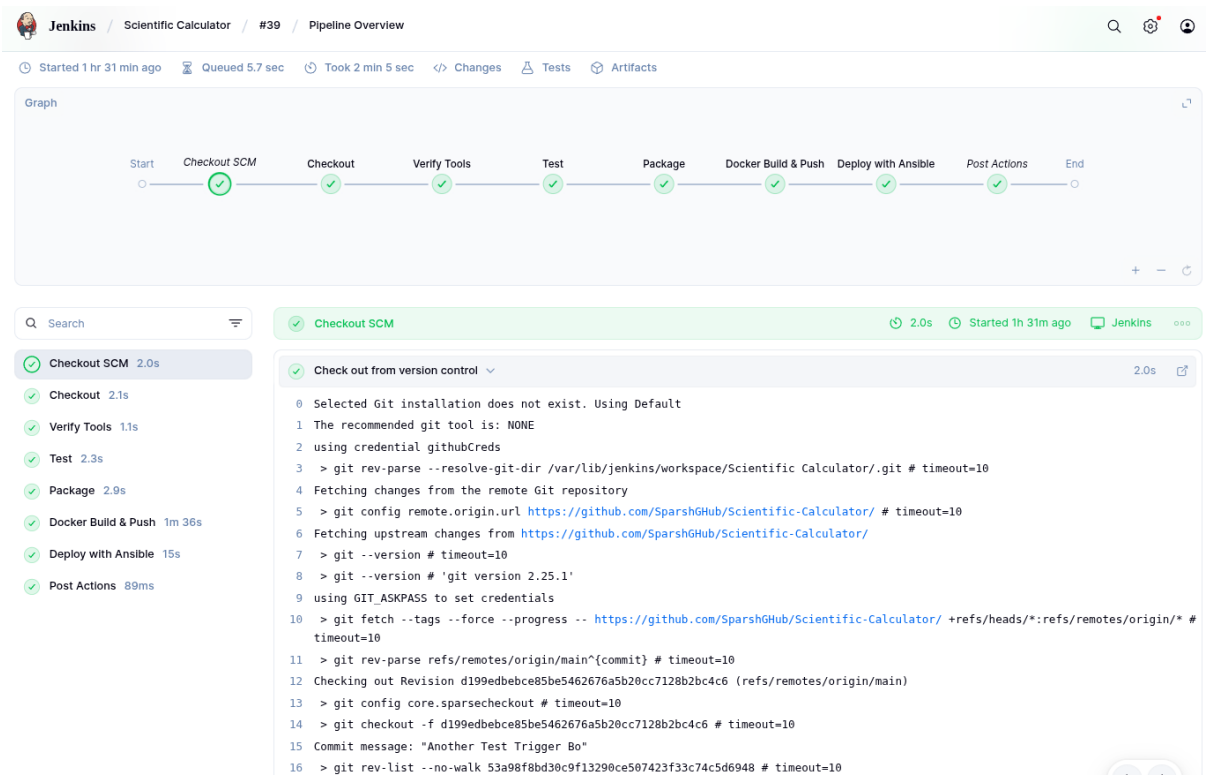

Figure 5: Checkout SCM

Verify Tools prints versions so you immediately see if a tool is missing or misconfig-

ured.

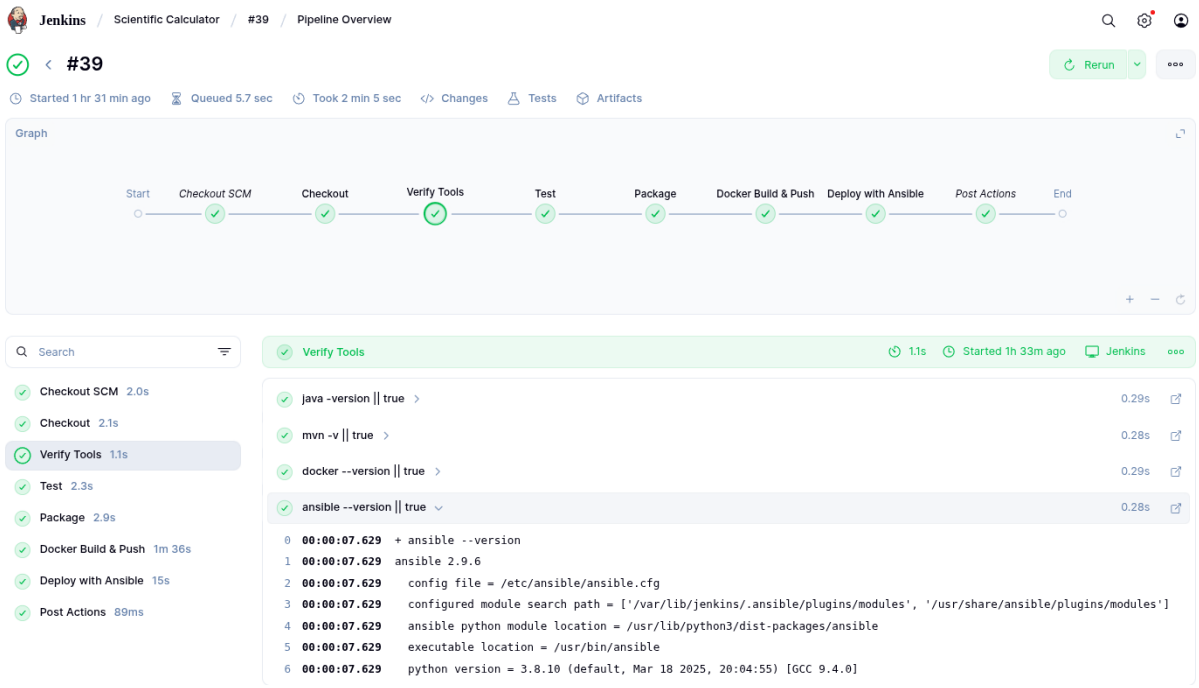

Figure 6: Verify Tools

Test runs all JUnit tests (Jenkins always publishes results from `target/surefire-reports`).

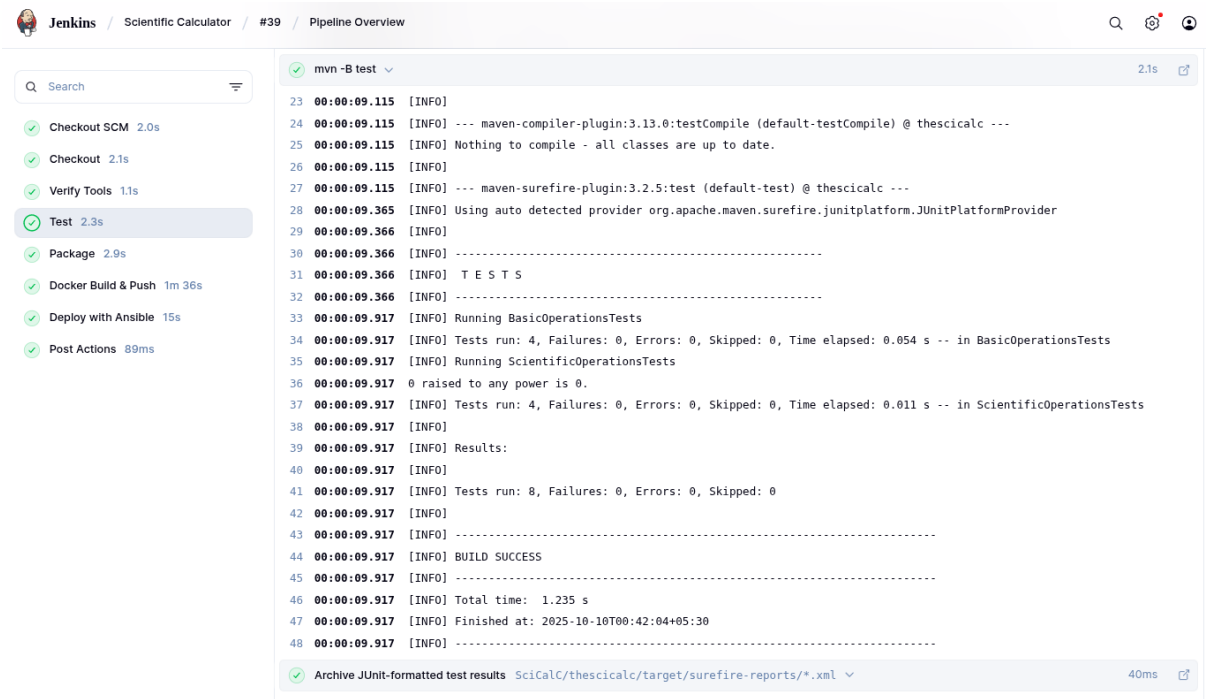

Figure 7: Test

Package builds the JAR (tests skipped here), and archives it as a build artifact.

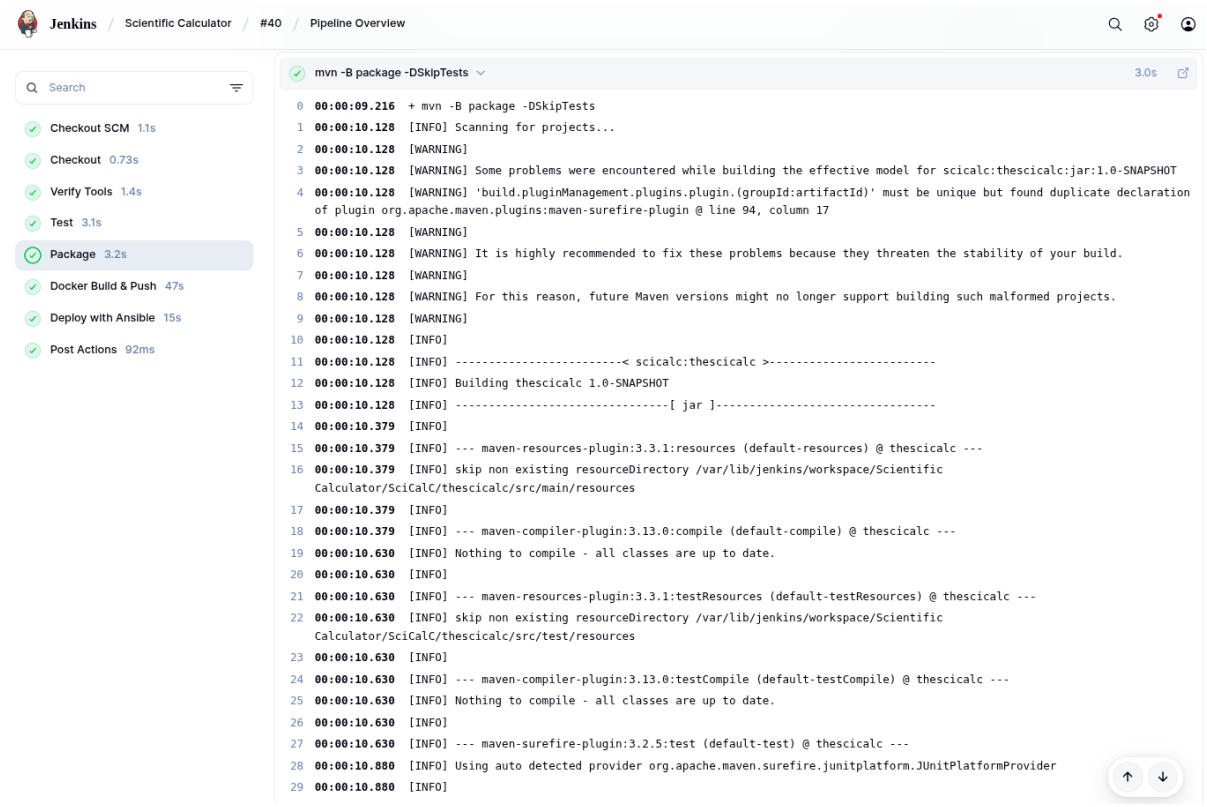

Figure 8: Package

Docker Build & Push builds a multi-tagged image (`:latest` and the commit SHA) and pushes `:latest` to Docker Hub using Jenkins credentials.

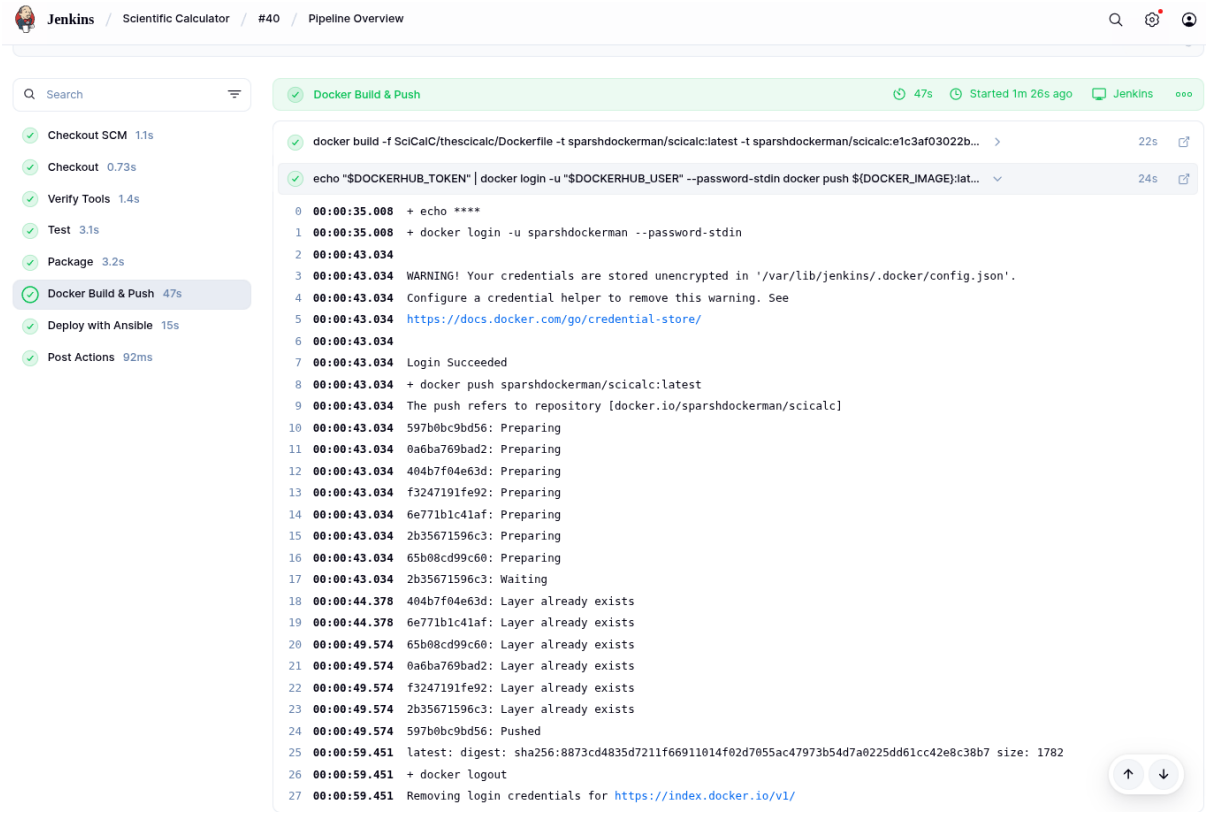

Figure 9: Docker Build and Push

Deploy with Ansible switches into the `ansible/` folder and runs the playbook that pulls `:latest` and restarts the local container.

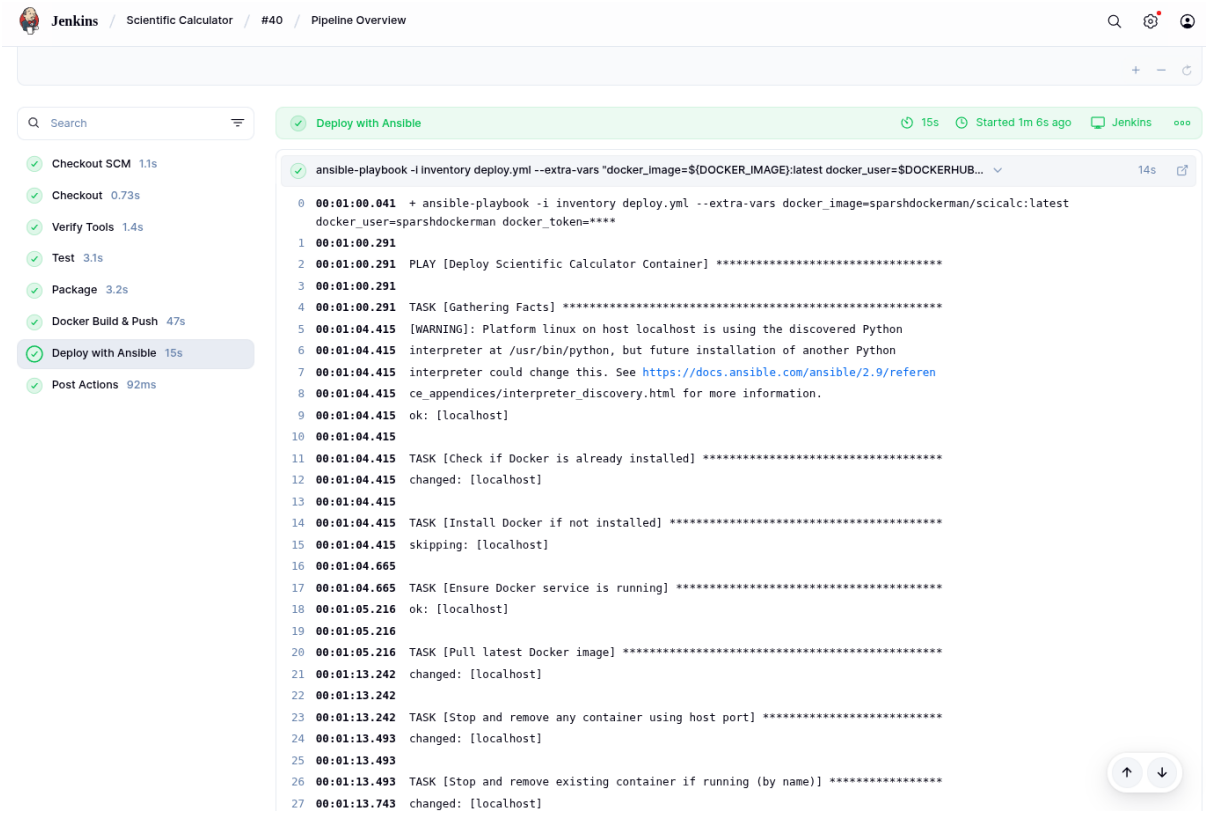

Figure 10: Deploy With Ansible

3. Docker Image (multi-stage)

Dockerfile (final)

```
# Stage 1: Build the JAR using Maven (Java 17)
FROM maven:3.9.6-eclipse-temurin-17 AS build
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests

# Stage 2: Run the application using OpenJDK (Java 17)
FROM eclipse-temurin:17-jre
WORKDIR /app
COPY --from=build /app/target/thescicalc-*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

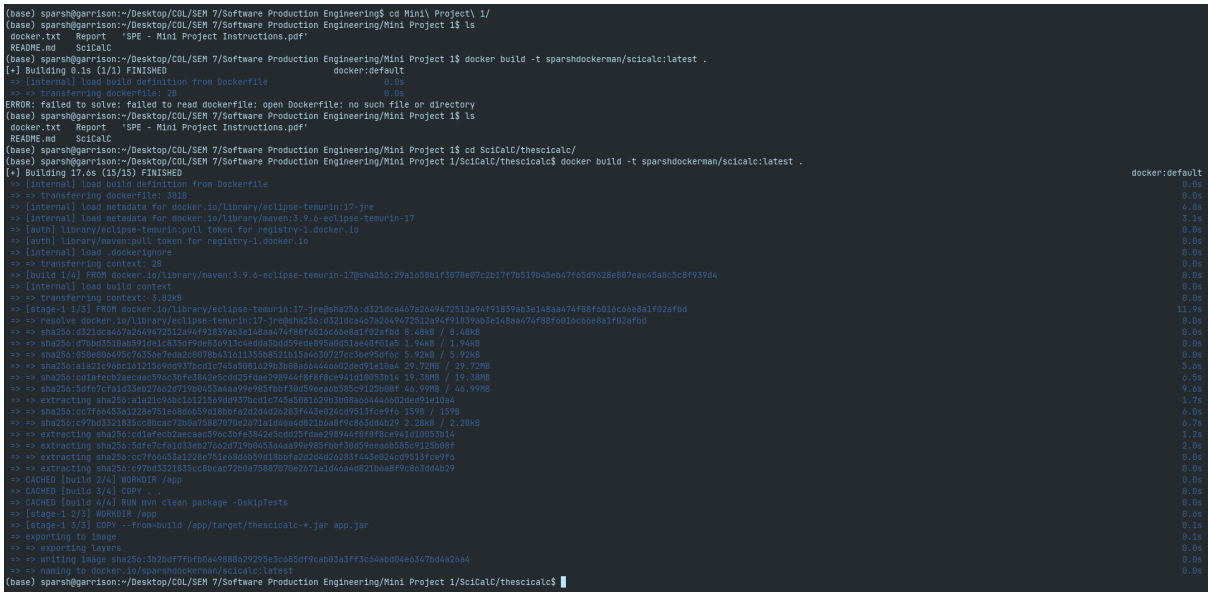

Figure 11: Docker Build

4. Ansible Deployment (local host)

The playbook below runs on `localhost`, pulls the latest image, stops any old container by name or port, and (for the CLI case) demonstrates an interactive run step in the pipeline context.

deploy.yml

```
---
- name: Deploy Scientific Calculator Container
  hosts: localhost
  connection: local
  become: yes

  vars:
    app_name: scicalc
    image_name: sparshdockerman/scicalc:latest
    container_port: 8080
    host_port: 9090

  tasks:
    - name: Check if Docker is already installed
      shell: "docker --version"
      register: docker_check
      ignore_errors: yes

    - name: Ensure Docker service is running
```

```
    service:
      name: docker
      state: started
      enabled: true

  - name: Pull latest Docker image
    shell: |
      docker pull {{ image_name }}

  - name: Stop and remove any container using host port
    shell: |
      ids=$(docker ps --filter "publish={{ host_port }}" -q)
      if [ -n "$ids" ]; then
        docker stop $ids
        docker rm $ids
      fi

  - name: Stop and remove existing container if running (by name)
    shell: |
      if [ "$(docker ps -q -f name={{ app_name }})" ]; then
        docker stop {{ app_name }}
      fi
      if [ "$(docker ps -aq -f name={{ app_name }})" ]; then
        docker rm {{ app_name }}
      fi

  # For CLI demonstration (non-daemon)
  - name: Run new CLI container interactively (pipeline demo)
    shell: |
      docker run -it --name {{ app_name }} --rm {{ image_name }} <<'
         EOF'
      echo "Scientific Calculator CLI started"
      exit
      EOF
    register: run_output
    ignore_errors: yes

  - name: Verify running containers
    shell: "docker ps --filter name={{ app_name }}"
    register: container_status

  - name: Display container status and CLI output
    debug:
      msg:
        - "Container status:"
        - "{{ container_status.stdout_lines }}"
```

```
          - "CLI output:"
          - "{{ run_output.stdout_lines }}"
```

5. RUN.sh (local helper)

This is just to shorten the command for checking if any local container is already running and then running a new one. Make it executable and run:

```
chmod +x RUN.sh
./RUN.sh
```

6. Application Demonstration

Images demonstrating the working of the required functions.

```
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Logarithm
6. Exponentiation
7. SquareRoot
8. Factorial
9. Exit


Enter your choice:

7
Enter the number:

8

RESULT: 2.8284271247461903
```

Figure 12: Square root function - $\sqrt{x}$

```
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Logarithm
6. Exponentiation
7. SquareRoot
8. Factorial
9. Exit


Enter your choice:

8
Enter the number :

5

RESULT: 120
```

Figure 13: Factorial function - x!

```
(base) sparsh@garrison:~/Desktop/COL/SEM 7/Software Production Engineering/Mini Project 1$ ./RUN.sh
Starting Scientific Calculator container...
Welcome to SciCalC - Scientific Calculator

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Logarithm
6. Exponentiation
7. SquareRoot
8. Factorial
9. Exit


Enter your choice:

5
Enter the number and the base:

16
4

RESULT: 2.0
```

Figure 14: Logarithm function - $\log_b a$

```
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Logarithm
6. Exponentiation
7. SquareRoot
8. Factorial
9. Exit


Enter your choice:

6
Enter the base and the exponent:

3
5

RESULT: 243.0
```

Figure 15: Power function - a^b

```
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Logarithm
6. Exponentiation
7. SquareRoot
8. Factorial
9. Exit


Enter your choice:

7
Enter the number:

-23

SQUARE ROOT OF NEGATIVE NUMBERS IS NOT DEFINED.
```

Figure 16: Exception Handling 1

```
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Logarithm
6. Exponentiation
7. SquareRoot
8. Factorial
9. Exit


Enter your choice:

5
Enter the number and the base:

-12 2

LOGARITHM OF NON-POSITIVE NUMBERS IS NOT DEFINED.
```

Figure 17: Exception Handling 2

7. Repository Links

GitHub: https://github.com/SparshGHub/Scientific-Calculator
Docker Hub: https://hub.docker.com/r/sparshdockerman/scicalc