Mini Project Report
Scientific Calculator with DevOps

Student Name: Aryan Vaghasiya

Roll Number: IMT2022046

Project Title: Scientific Calculator with DevOps

October 10, 2025

Contents

1 Introduction

1.1 Problem Statement

This project develops a scientific calculator implementing:

- Square root function: $\sqrt{x}$
- Factorial function: $x!$
- Natural logarithm (base e): $\ln(x)$
- Power function: x^b

1.2 Project Goal

The project implements a complete DevOps pipeline demonstrating CI/CD practices, automated testing, containerization, and configuration management.

2 What and Why of DevOps?

2.1 What is DevOps?

DevOps unifies software development (Dev) and IT operations (Ops) to deliver applications faster and more reliably. It emphasizes:

- **Collaboration:** Breaking down barriers between development and operations teams
- **Automation:** Automating repetitive tasks throughout the software lifecycle
- **Continuous Feedback:** Establishing feedback loops for rapid iteration
- **Monitoring:** Continuous observation of application performance

Key components include Continuous Integration (CI), Continuous Delivery/Deployment (CD), Infrastructure as Code (IaC), automated testing, and monitoring.

2.2 Why DevOps?

DevOps adoption provides:

1. **Faster Release Cycles:** Automation enables rapid deployment
2. **Improved Quality:** Automated testing catches bugs early
3. **Enhanced Scalability:** Infrastructure as code facilitates scaling
4. **Reduced Deployment Risk:** Automated deployments with rollback capabilities
5. **Better Collaboration:** Unified tooling improves team communication
6. **Cost Efficiency:** Automation reduces manual effort

2.3 DevOps Lifecycle

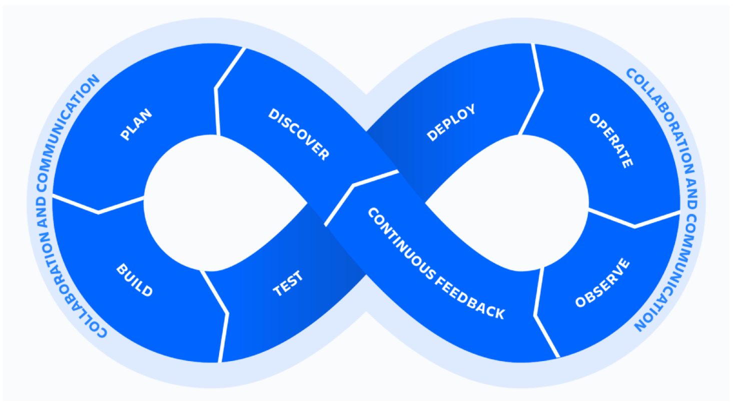

COLLABORATION AND COMMUNICATION
PLAN
DISCOVER
DEPLOY
OPERATE
COLLABORATION AND COMMUNICATION
BUILD
TEST
CONTINUOUS FEEDBACK
OBSERVE

Figure 1: DevOps Lifecycle

3 Tools Used

Table 1: Tools Used in the DevOps Pipeline

Pipeline Stage	Tool	Purpose
Source Control	Git + GitHub	Version control and collaboration
Automated Testing	JUnit 5	Unit testing framework
Build Management	Maven	Build automation and dependency management
Continuous Integration	Jenkins	CI/CD pipeline orchestration
Webhook Tunneling	ngrok	GitHub-Jenkins webhook connectivity
Containerization	Docker	Application containerization
Container Registry	Docker Hub	Image storage and distribution
Configuration Management	Ansible	Automated deployment

4 Pipeline Architecture

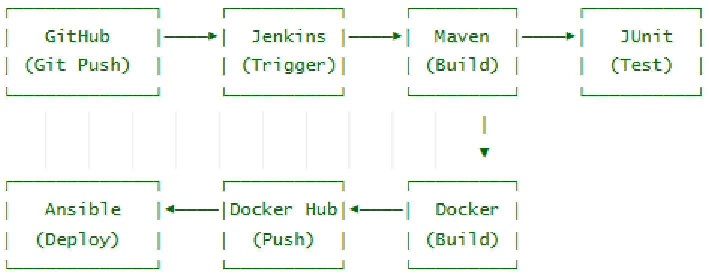


Figure 2: Pipeline Architecture Overview

Pipeline Workflow:

1. Developer pushes code to GitHub repository
2. GitHub webhook triggers Jenkins via ngrok tunnel

3. Jenkins pipeline executes via Jenkinsfile (provides stage view)
4. Jenkins pulls latest code from GitHub repository
5. Maven runs automated test cases using JUnit
6. Docker builds container image from Dockerfile
7. Jenkins logs into Docker Hub using credentials
8. Docker image is pushed to Docker Hub registry
9. Ansible deploys container on local system
10. Email notification sent about pipeline success/failure

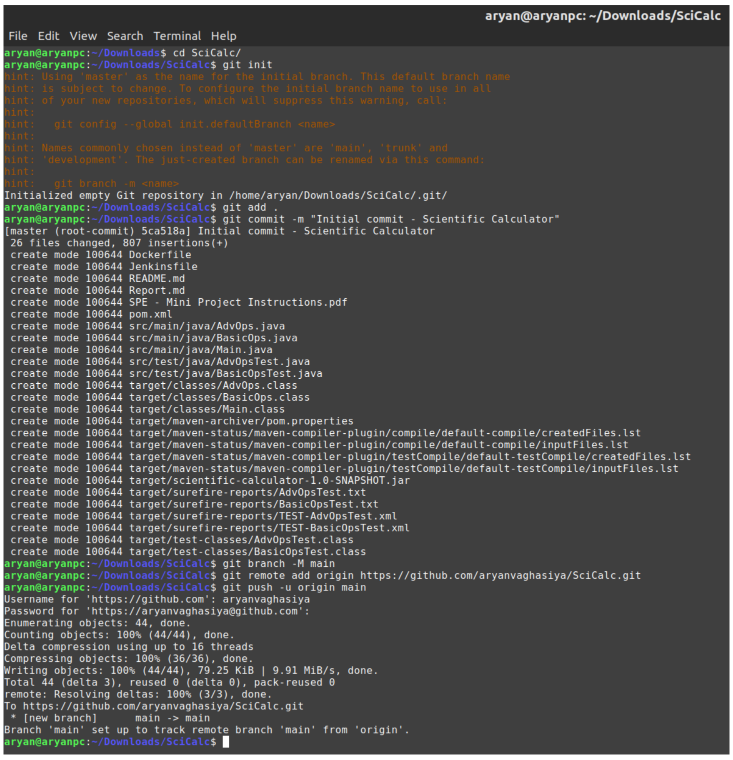
5 Implementation Details

5.1 Source Control Management (Git + GitHub)

5.1.1 Setup

```
# Initialize local Git repository
git init

# Add remote repository
git remote add origin https://github.com/aryanvaghasiya/SciCalc.git

# Stage and commit files
git add .
git commit -m "Initial commit: SciCalc project uploaded"

# Push to remote
git push -u origin main
```

Listing 1: Git Setup Commands


aryan@aryanpc: ~/Downloads/SciCalc
File Edit View Search Terminal Help

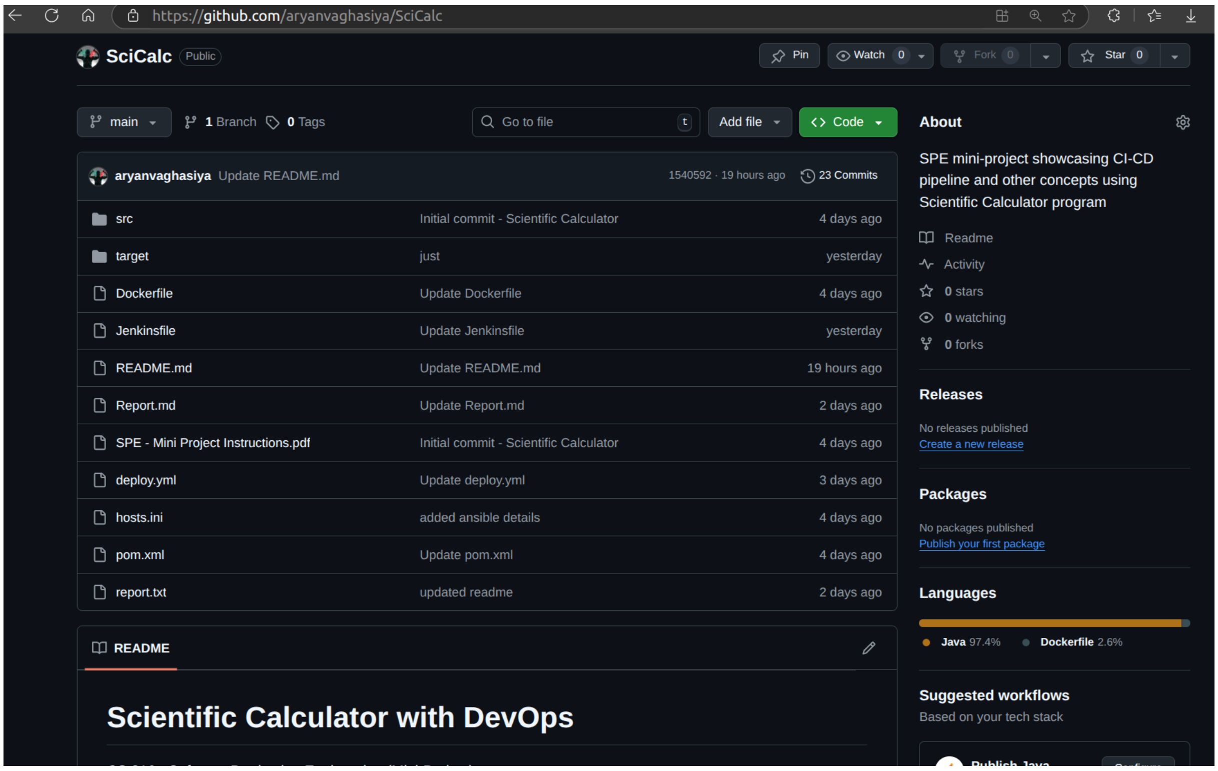
```
aryan@aryanpc:~/Downloads$ cd SciCalc/
aryan@aryanpc:~/Downloads/SciCalc$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/aryan/Downloads/SciCalc/.git/
aryan@aryanpc:~/Downloads/SciCalc$ git add .
aryan@aryanpc:~/Downloads/SciCalc$ git commit -m "Initial commit - Scientific Calculator"
[master (root-commit) 5ca518a] Initial commit - Scientific Calculator
 26 files changed, 807 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 Jenkinsfile
 create mode 100644 README.md
 create mode 100644 Report.md
 create mode 100644 SPE - Mini Project Instructions.pdf
 create mode 100644 pom.xml
 create mode 100644 src/main/java/AdvOps.java
 create mode 100644 src/main/java/BasicOps.java
 create mode 100644 src/main/java/Main.java
 create mode 100644 src/test/java/AdvOpsTest.java
 create mode 100644 src/test/java/BasicOpsTest.java
 create mode 100644 target/classes/AdvOps.class
 create mode 100644 target/classes/BasicOps.class
 create mode 100644 target/classes/Main.class
 create mode 100644 target/maven-archiver/pom.properties
 create mode 100644 target/maven-status/maven-compiler-plugin/compile/default-compile/createdFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/createdFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/inputFiles.lst
 create mode 100644 target/scientific-calculator-1.0-SNAPSHOT.jar
 create mode 100644 target/surefire-reports/AdvOpsTest.txt
 create mode 100644 target/surefire-reports/BasicOpsTest.txt
 create mode 100644 target/surefire-reports/TEST-AdvOpsTest.xml
 create mode 100644 target/surefire-reports/TEST-BasicOpsTest.xml
 create mode 100644 target/test-classes/AdvOpsTest.class
 create mode 100644 target/test-classes/BasicOpsTest.class
aryan@aryanpc:~/Downloads/SciCalc$ git branch -M main
aryan@aryanpc:~/Downloads/SciCalc$ git remote add origin https://github.com/aryanvaghasiya/SciCalc.git
aryan@aryanpc:~/Downloads/SciCalc$ git push -u origin main
Username for 'https://github.com': aryanvaghasiya
Password for 'https://aryanvaghasiya@github.com':
Enumerating objects: 44, done.
Counting objects: 100% (44/44), done.
Delta compression using up to 16 threads
Compressing objects: 100% (36/36), done.
Writing objects: 100% (44/44), 79.25 KiB | 9.91 MiB/s, done.
Total 44 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/aryanvaghasiya/SciCalc.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
aryan@aryanpc:~/Downloads/SciCalc$
```

Figure 3: Git commands

5.1.2 Repository Structure

```
Scientific-Calculator-SPE/
|-- src/
|   |-- main/
|   |   |-- java/
|   |       |-- calculator/
|   |           |-- BasicOps.java
|   |           |-- AdvOps.java
|   |-- test/
|       |-- java/
|           |-- calculator/
|               |-- BasicOpsTest.java
|               |-- AdvOpsTest.java
|-- target/               (Maven build directory)
|   |-- ...
|-- pom.xml
|-- Dockerfile
|-- Jenkinsfile
|-- deploy.yml
|-- hosts.ini
|-- README.md
```

Listing 2: Project Repository Structure


https://github.com/aryanvaghasiya/SciCalc

SciCalc Public

Pin | Watch 0 | Fork 0 | Star 0

main | 1 Branch | 0 Tags | Go to file | Add file | Code

aryanvaghasiya Update README.md 1540592 · 19 hours ago 23 Commits

src	Initial commit - Scientific Calculator	4 days ago
target	just	yesterday
Dockerfile	Update Dockerfile	4 days ago
Jenkinsfile	Update Jenkinsfile	yesterday
README.md	Update README.md	19 hours ago
Report.md	Update Report.md	2 days ago
SPE - Mini Project Instructions.pdf	Initial commit - Scientific Calculator	4 days ago
deploy.yml	Update deploy.yml	3 days ago
hosts.ini	added ansible details	4 days ago
pom.xml	Update pom.xml	4 days ago
report.txt	updated readme	2 days ago

README

Scientific Calculator with DevOps

About

SPE mini-project showcasing CI-CD pipeline and other concepts using Scientific Calculator program

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Java 97.4% Dockerfile 2.6%

Suggested workflows
Based on your tech stack

Figure 4: Project structure in GitHub repository

Github Repository Link: SciCalc

5.2 Testing (JUnit 5)

JUnit 5 provides a robust framework for automated testing of Java applications.

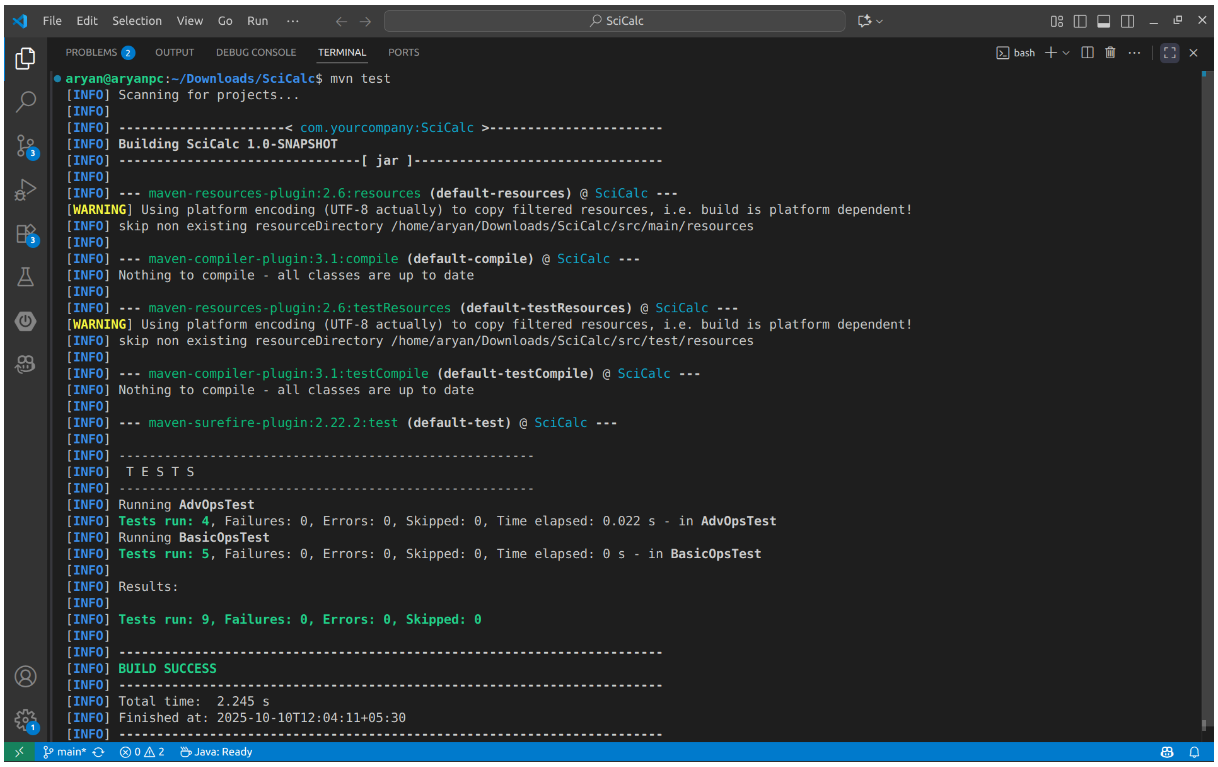
5.2.1 Test Classes

Test classes were created in the `src/test/java` directory to verify the correctness of all calculator operations including square root, factorial, logarithm, and power functions.

5.2.2 Test Execution

```
mvn test
```

Listing 3: Maven Test Command



```
aryan@aryanpc:~/Downloads/SciCalc$ mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< com.yourcompany:SciCalc >---------------------
[INFO] Building SciCalc 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ SciCalc ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/aryan/Downloads/SciCalc/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ SciCalc ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ SciCalc ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/aryan/Downloads/SciCalc/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ SciCalc ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ SciCalc ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running AdvOpsTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 s - in AdvOpsTest
[INFO] Running BasicOpsTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in BasicOpsTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.245 s
[INFO] Finished at: 2025-10-10T12:04:11+05:30
[INFO] ------------------------------------------------------------------------
```

Figure 5: Successful test execution

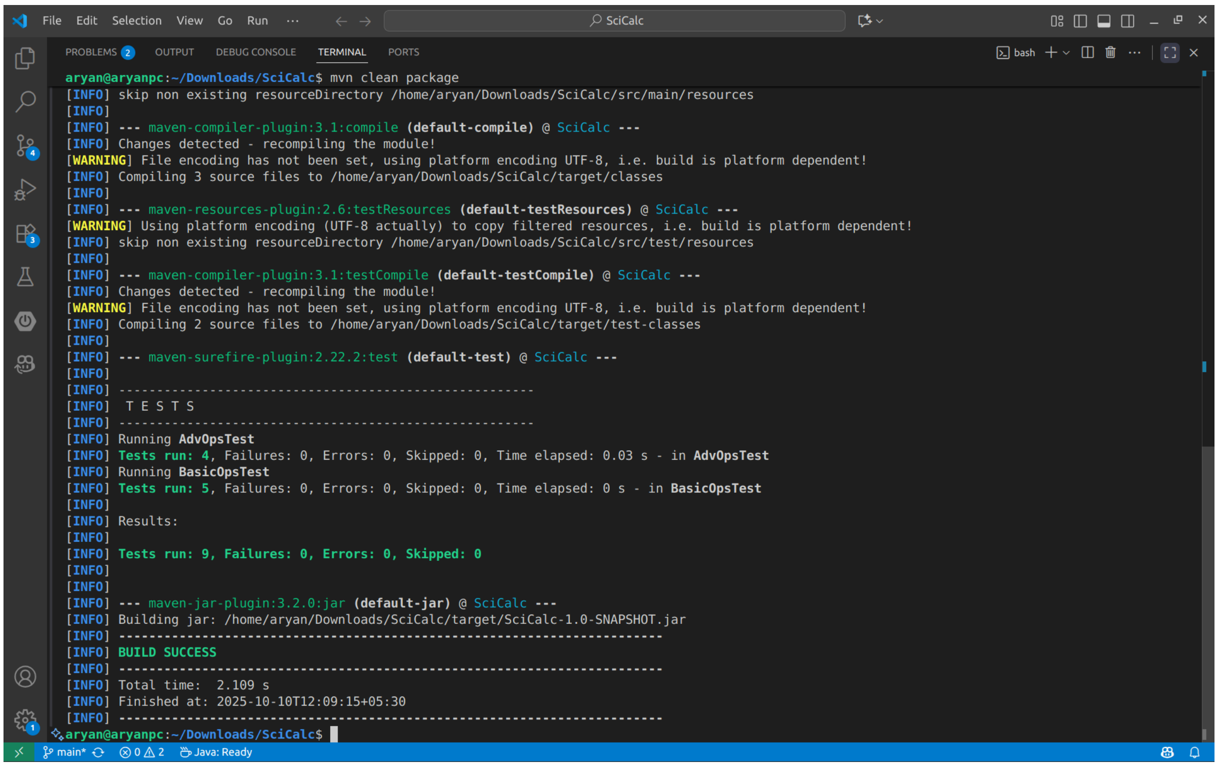
5.3 Build Tool (Maven)

Apache Maven is a powerful project management and build automation tool based on the Project Object Model (POM). Maven simplifies the build process, manages dependencies, and creates executable artifacts.

5.3.1 Build Process

```
mvn clean package
```

Listing 4: Maven Clean and Package Command



```
aryan@aryanpc:~/Downloads/SciCalc$ mvn clean package
[INFO] skip non existing resourceDirectory /home/aryan/Downloads/SciCalc/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ SciCalc ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 3 source files to /home/aryan/Downloads/SciCalc/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ SciCalc ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/aryan/Downloads/SciCalc/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ SciCalc ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 2 source files to /home/aryan/Downloads/SciCalc/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ SciCalc ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running AdvOpsTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.03 s - in AdvOpsTest
[INFO] Running BasicOpsTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in BasicOpsTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ SciCalc ---
[INFO] Building jar: /home/aryan/Downloads/SciCalc/target/SciCalc-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.109 s
[INFO] Finished at: 2025-10-10T12:09:15+05:30
[INFO] ------------------------------------------------------------------------
aryan@aryanpc:~/Downloads/SciCalc$
```

Figure 6: Maven build process - packaging and artifact creation

5.3.2 GitHub Webhook Configuration

Setting up ngrok Tunnel:

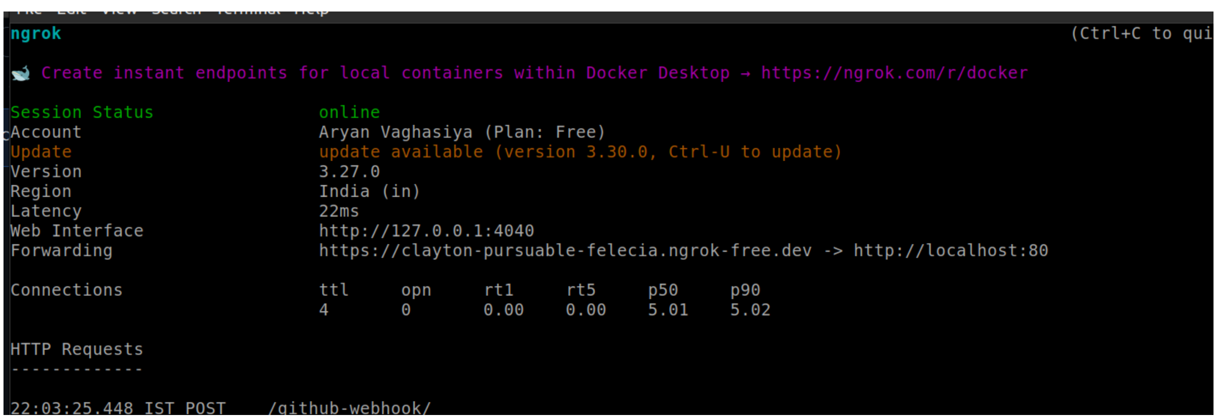
Since Jenkins is running locally, ngrok creates a secure tunnel to expose the local Jenkins instance to the internet, allowing GitHub to send webhook notifications.

```
# Start ngrok tunnel on port 8080 (Jenkins default port)
ngrok http 8080
# for static
ngrok http --url:"https://clayton-pursuable-felecia.ngrok-free.dev/"
    8080
```

Listing 5: Starting ngrok Tunnel

Configuring GitHub Webhook:

1. Navigate to your GitHub repository: `SciCalc`
2. Go to **Settings** → **Webhooks** → **Add webhook**
3. Configure the webhook:
   - **Payload URL:** `https://clayton-pursuable-felecia.ngrok-free.dev/github-webhook`
   - **Content type:** `application/json`
   - **Which events:** Select "Just the push event"
   - **Active:** Check this option
4. Click **Add webhook**

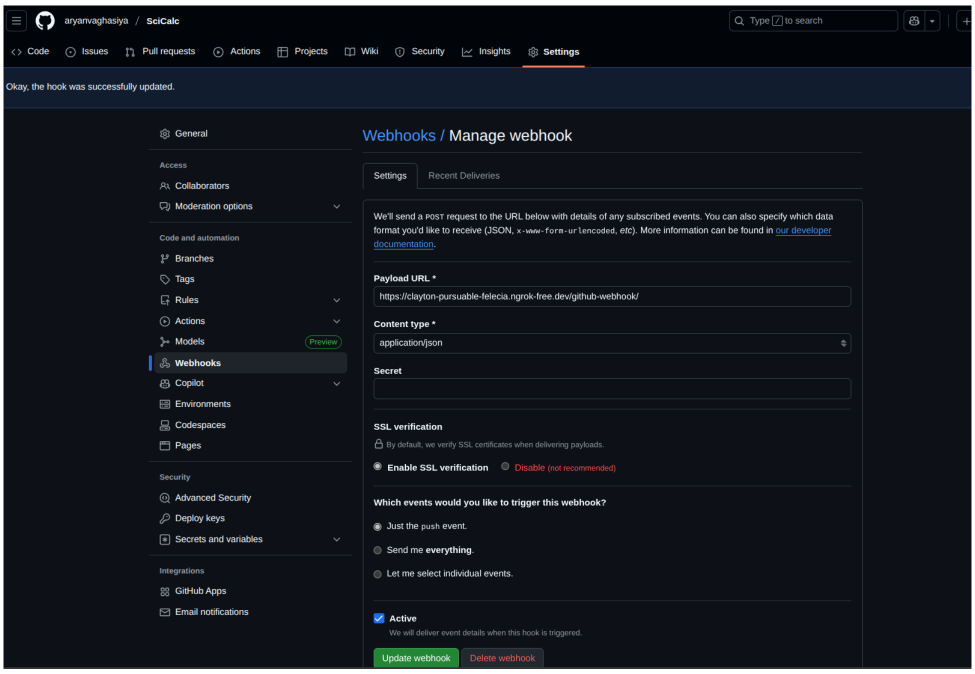


```
ngrok                                                                                  (Ctrl+C to qui

🐳 Create instant endpoints for local containers within Docker Desktop → https://ngrok.com/r/docker

Session Status                online
Account                       Aryan Vaghasiya (Plan: Free)
Update                        update available (version 3.30.0, Ctrl-U to update)
Version                       3.27.0
Region                        India (in)
Latency                       22ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://clayton-pursuable-felecia.ngrok-free.dev -> http://localhost:80

Connections                   ttl     opn     rt1     rt5     p50     p90
                              4       0       0.00    0.00    5.01    5.02

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

22:03:25.448 IST POST    /github-webhook/
```

Figure 7: ngrok

Pipeline Flow: How It Works:

1. Developer pushes code to GitHub repository
2. GitHub sends POST request to the ngrok webhook URL
3. ngrok tunnels the request to local Jenkins instance (port 8080)
4. Jenkins receives webhook notification and triggers the pipeline defined in Jenkinsfile
5. Pipeline stages are displayed in Jenkins Stage View UI
6. Pipeline executes the stages sequentially.
7. Post-build action sends email notification about success or failure



Okay, the hook was successfully updated.

Webhooks / Manage webhook

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://clayton-pursuable-felecia.ngrok-free.dev/github-webhook/

Content type *
application/json

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 8: Github Webhook

Important Notes:

- The ngrok URL changes with each session unless using a static url domains
- Ensure ngrok is running whenever automated builds are required
- Update GitHub webhook URL if ngrok URL changes
- For production environments, Jenkins should be hosted on a public server with a static URL

5.4 Continuous Integration (Jenkins)

Jenkins is an open-source automation server that enables continuous integration and continuous delivery. It automates the entire build, test, and deployment pipeline, triggered by code changes.

5.4.1 Benefits of Jenkins

- **Automation:** Streamlines repetitive tasks like building, testing, and deploying code
- **Extensibility:** Supports a wide range of plugins to integrate with tools like Git, Maven, and Docker
- **Flexibility:** Configurable pipelines allow customization of CI/CD workflows
- **Scalability:** Supports distributed builds across multiple nodes for faster processing
- **Community Support:** Large open-source community provides extensive resources and plugins

Access Jenkins at `http://localhost:8080`.

5.4.2 Install Required Plugins

Navigate to **Manage Jenkins** → **Manage Plugins** and install:

- Git Plugin
- Maven Integration Plugin
- Docker Pipeline Plugin
- Ansible Plugin
- GitHub Integration Plugin
- Email Extension Plugin (for email notifications)

5.4.3 Configure Global Tools

Navigate to **Manage Jenkins** → **Global Tool Configuration** and configure:

- **JDK:** Specify Java installation path (JDK 17)
- **Maven:** Add Maven installation (name: `Maven_3.6.3`, version: 3.6.3)
- **Git:** Verify Git executable path
- **Docker:** Configure Docker installation
- **Ansible:** Specify Ansible executable path

5.4.4 Configure Email Notifications

Navigate to **Manage Jenkins** → **Configure System** and configure Extended E-mail Notification and E-mail Notification with the following:

- **SMTP Server:** smtp.gmail.com (for Gmail)
- **SMTP Port:** 465
- **Use SSL:** checked
- **Use TLS:**
- **Credentials:** Add Gmail credentials (email and app password)
- **Default Recipients:** aryanvaghasia12345@gmail.com

Note: For Gmail, you need to generate an App Password from Google Account settings if 2-factor authentication is enabled.

5.4.5 Configure Credentials

Navigate to **Manage Jenkins** → **Manage Credentials** and add:

- GitHub Webhook credentials (secret text: personal access token with ID: `webhook`)
- Docker Hub credentials (username/password with ID: `dockerhub-credentials`)

5.4.6 Jenkins URL Configuration

Navigate to **Manage Jenkins** → **Configure System** and set:

- **Jenkins URL:** `https://clayton-pursuable-felecia.ngrok-free.dev/`

This static ngrok URL enables GitHub to communicate with the locally hosted Jenkins instance through a secure tunnel.

5.4.7 Create Jenkins Pipeline Job

1. Click **New Item** in Jenkins dashboard
2. Enter job name: `SciCalc`
3. Select **Pipeline** and click **OK**
4. In job configuration:
   - Check **GitHub project** and enter repository URL
   - Under **Build Triggers**, check **GitHub hook trigger for GITScm polling**
   - Under **Pipeline**, select **Pipeline script from SCM**
   - Select **Git** as SCM
   - Enter repository URL and credentials
   - Specify branch: `*/main`
   - Script Path: `Jenkinsfile`
5. Click **Save**

5.4.8 Jenkinsfile

```
pipeline {
    agent any   // Run on any available Jenkins agent

    // triggers {
    //     githubPush()
    // }

    tools {
        maven 'Maven_3.6.3'   // Must match Global Tool Config name in Jenkins
        jdk 'JDK17'
    }

    environment {
        DOCKER_IMAGE = "aryanvaghasiya/scicalc-app"
        // Define the location of the Ansible inventory file (e.g., hosts.ini)
        ANSIBLE_INVENTORY = "hosts.ini"
        // Define the playbook file (assuming it's in the root of the repo)
        ANSIBLE_PLAYBOOK = "deploy.yml"
    }

    stages {
        stage('Checkout') {
            steps {
                // Pull code from GitHub
                git branch: 'main', url: 'https://github.com/aryanvaghasiya/SciCalc'
            }
        }

        stage('Build') {
            steps {
                // clean and compile
                sh 'mvn clean compile'
            }
        }

        stage('Test') {
            steps {
                // run unit tests
                sh 'mvn test'
            }
        }

        stage('Package') {
            steps {
                // create the jar file
                sh 'mvn package'
            }
        }

        stage('Archive Artifact') {
            steps {
                // Save JAR file inside Jenkins
                archiveArtifacts artifacts: 'target/*.jar', fingerprint: true
            }
        }

        stage('Docker Build & Push') {
            steps {
                script {
                    docker.withRegistry('https://index.docker.io/v1/', 'dockerhub-credentials') {
                        // Build Docker image with version tag
                        def app = docker.build("${DOCKER_IMAGE}:${env.BUILD_NUMBER}")
                        // Push with build number tag
                        app.push()
                        // Also push as latest
                        app.push("latest")
                    }
                }
            }
        }

        stage('Ansible Deploy') {
            steps {
                echo "Starting Ansible deployment for image ${DOCKER_IMAGE}:${env.BUILD_NUMBER}"

                sh "ansible-playbook -i ${ANSIBLE_INVENTORY} ${ANSIBLE_PLAYBOOK} -e \"docker_image_tag=${DOCKER_IMAGE}:${env.BUILD_NUMBER}\""
            }
        }
    }

    post {
        success {
            echo "Build #${env.BUILD_NUMBER} completed successfully!"
            emailext(
                to: 'aryanvaghasia12345@gmail.com',
                subject: "Jenkins Build SUCCESS: ${env.JOB_NAME} #${env.BUILD_NUMBER}",
                body: """
                Hi Team,

                The Jenkins pipeline for *${env.JOB_NAME}* completed successfully.
                Docker Image pushed: ${DOCKER_IMAGE}:${env.BUILD_NUMBER}
                Deployment initiated via Ansible.

                Regards,
                Jenkins CI/CD(Aryan Vaghasiya)
                """
            )
        }
        failure {
            echo "Build #${env.BUILD_NUMBER} failed!"
            emailext(
                to: 'aryanvaghasia12345@gmail.com',
                subject: "Jenkins Build FAILURE: ${env.JOB_NAME} #${env.BUILD_NUMBER}",
                body: """
                Hi Team,

                The Jenkins pipeline for *${env.JOB_NAME}* has FAILED.
                Please check Jenkins logs for details.

                Regards,
                Jenkins CI/CD(Aryan Vaghasiya)
                """
            )
        }
    }
}
```

Listing 6: Jenkins Pipeline Script (`Jenkinsfile`)

5.4.9 Jenkinsfile Explanation

The Jenkinsfile defines a declarative pipeline that automates the CI/CD process. It consists of configuration blocks and sequential stages.

Pipeline Configuration:

- **agent any:** Allows pipeline to run on any available Jenkins node
- **tools:** Specifies Maven 3.6.3 and JDK 17 configured in Jenkins Global Tool Configuration
- **environment:** Defines variables accessible across all stages:
  - `DOCKER_IMAGE`: Docker Hub repository name

  - `ANSIBLE_INVENTORY`: Path to hosts.ini file
  - `ANSIBLE_PLAYBOOK`: Path to deploy.yml file

Pipeline Stages:

1. **Checkout:** Clones the GitHub repository and checks out the main branch. This provides Jenkins with the latest source code for building and testing.
2. **Build:** Executes `mvn clean compile` to remove previous build artifacts and compile Java source code into `.class` files. Validates code syntax and dependency resolution.
3. **Test:** Runs `mvn test` to execute all JUnit test cases from `src/test/java`. Pipeline fails if any test fails, ensuring code quality before deployment.
4. **Package:** Executes `mvn package` to create an executable JAR file (`SciCalc-1.0-SNAPSHOT.jar`) in the target directory. This JAR contains all compiled classes and dependencies.
5. **Archive Artifact:** Stores the JAR file permanently in Jenkins with MD5 fingerprinting. Enables artifact versioning and provides rollback capability.
6. **Build Docker Image:** Creates two Docker images using the Dockerfile:
   - Tagged with build number: `aryanvaghasiya/scicalc-app:${BUILD_NUMBER}`
   - Tagged as latest: `aryanvaghasiya/scicalc-app:latest`

   Both images package the application with OpenJDK 17 environment.
7. **Push to Docker Hub:** Authenticates with Docker Hub using stored credentials (`dockerhub-credentials`) and pushes both image tags to the registry. Makes images accessible for deployment from any environment.
8. **Deploy with Ansible:** Executes the Ansible playbook with the command:

   ```
   ansible-playbook -i hosts.ini deploy.yml
     -e "docker_image_tag=aryanvaghasiya/scicalc-app:BUILD_NUMBER"
   ```

   The playbook pulls the Docker image, stops any existing container, and starts a new container with the updated image on port 9000.

Post-Build Actions:

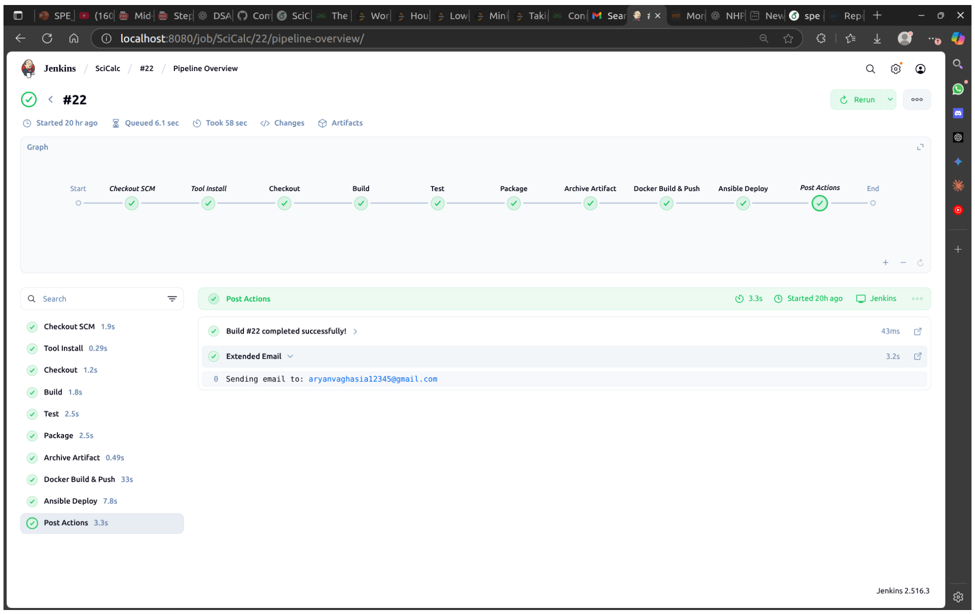
- **Success:** Sends HTML email notification with job name, build number, Docker image details, and success confirmation when all stages complete successfully.
- **Failure:** Sends HTML email alert with job name, build number, and instructions to check Jenkins logs when any stage fails. Enables rapid response to issues.

Stage Dependencies: Each stage depends on the previous stage's success. If any stage fails, the pipeline stops immediately and triggers the failure notification.

Key Variables Used:

- `${env.BUILD_NUMBER}`: Auto-incremented build number (e.g., 123)

- `${env.JOB_NAME}`: Name of the Jenkins job
- `${DOCKER_IMAGE}`: Docker repository name from environment variables



Checkout SCM	1.9s
Tool Install	0.29s
Checkout	1.2s
Build	1.8s
Test	2.5s
Package	2.5s
Archive Artifact	0.49s
Docker Build & Push	33s
Ansible Deploy	7.8s
Post Actions	3.3s

Post Actions 3.3s Started 20h ago Jenkins

```
Build #22 completed successfully!    43ms
Extended Email                       3.2s
0 Sending email to: aryanvaghasia12345@gmail.com
```

Figure 9: Jenkins pipeline execution showing all stages in Stage View

Jenkins Stage View: Visualization of the pipeline stages in Jenkins UI, allowing developers to monitor the progress of each stage. Each stage is color-coded (green for success, red for failure) and displays execution time.

5.5 Containerization (Docker)

Docker is a platform for developing, shipping, and running applications in containers. Containers package an application with all its dependencies, ensuring consistent behavior across different environments.

5.5.1 Benefits of Containerization

- **Portability:** Run anywhere Docker is installed
- **Isolation:** Applications run in isolated environments
- **Efficiency:** Lightweight compared to virtual machines
- **Scalability:** Easy to scale horizontally
- **Version Control:** Images are versioned and reproducible
- **Consistency:** Eliminates "works on my machine" problems

5.5.2 Dockerfile

```
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY target/SciCalc-1.0-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Listing 7: Dockerfile for Scientific Calculator

Dockerfile Explanation:

- `FROM openjdk:17-jdk-slim`: Uses lightweight OpenJDK 17 base image
- `WORKDIR /app`: Sets working directory inside container
- `COPY`: Copies JAR file from Maven target directory
- `ENTRYPOINT`: Defines command to run the application

5.5.3 Build and Run Locally

```
# Build image
docker build -t aryanvaghasiya/scicalc-app:latest .

# Run container interactively
docker run -it --rm aryanvaghasiya/scicalc-app:latest
```

Listing 8: Docker Build and Run Commands

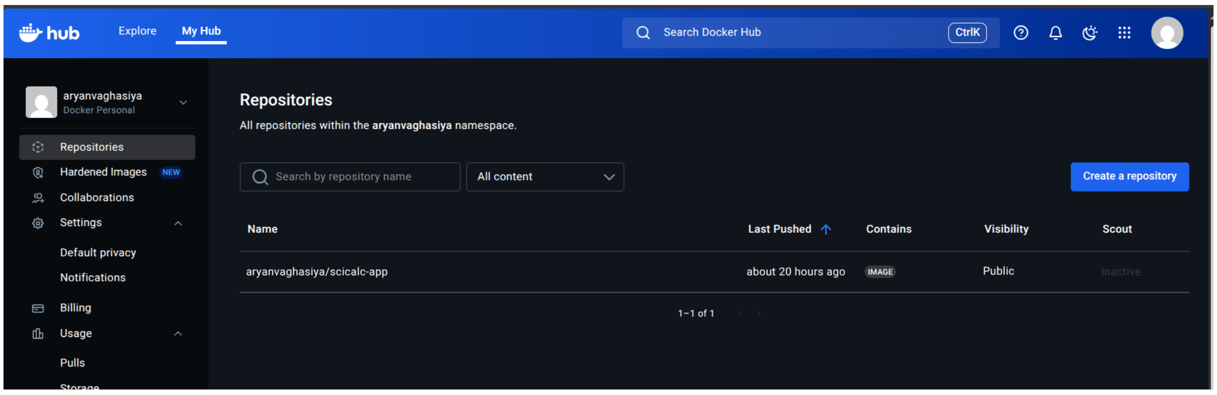
5.6 Docker Hub Repository

Docker Hub serves as the central registry for storing and distributing Docker images.

5.6.1 Manual Push to Docker Hub

```
# Login to Docker Hub
docker login

# Push image
docker push aryanvaghasiya/scicalc-app:latest
```

Listing 9: Docker Login and Push Commands



Repositories

All repositories within the **aryanvaghasiya** namespace.

Name	Last Pushed	Contains	Visibility	Scout
aryanvaghasiya/scicalc-app	about 20 hours ago	IMAGE	Public	Inactive

1–1 of 1

Figure 10: Docker Hub repository showing multiple image versions

Repository Link: https://hub.docker.com/r/aryanvaghasiya/scicalc-app

5.7 Continuous Deployment (Ansible)

Ansible is an open-source automation tool for configuration management, application deployment, and task automation. It uses YAML-based playbooks.

5.7.1 Ansible Project Structure

```
Scientific-Calculator-SPE/
|-- ...other files...
|-- hosts.ini       (Ansible inventory file)
|-- deploy.yml          (Ansible playbook)
```

Listing 10: Ansible Project Structure

5.7.2 Deployment Playbook

```
---
- name: Deploy Scientific Calculator Docker container
  hosts: local
  become: no
  tasks:
    - name: Ensure Python 'docker' package is installed
      ansible.builtin.pip:
        name: docker
        state: present

    - name: Deploy and manage SciCalc container from Docker Hub
      community.docker.docker_container:
        name: calc-app2
        image: areen9295/calc-app2:latest
        state: started
        pull: yes
        recreate: yes
        ports:
          - "9000:80"
        restart_policy: always
```

Listing 11: Ansible Deployment Playbook (`deploy.yml`)

Playbook Explanation:

- **Ensure Python 'docker' package:** Installs Python Docker library required by Ansible Docker modules
- **Deploy container:** Pulls latest image from Docker Hub, recreates container if exists, maps port 9000 on host to port 80 in container
- **Restart policy:** Ensures container automatically restarts on failure

5.7.3 Execution

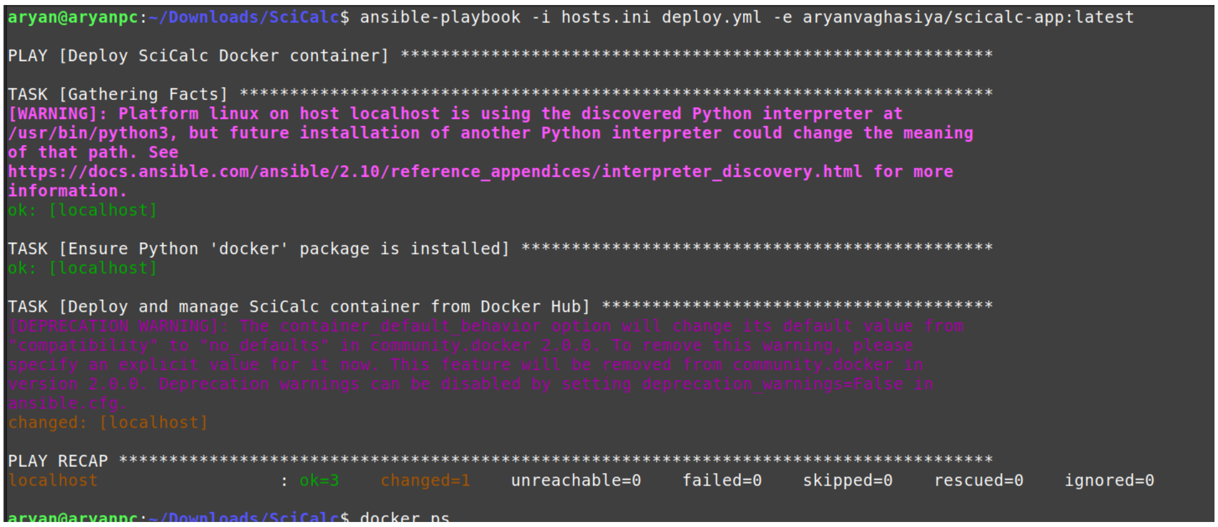
```
# Deploy application
ansible-playbook -i hosts.ini deploy.yml

# Deploy with specific image tag
ansible-playbook -i hosts.ini deploy.yml -e docker_image_tag=latest

#you can interact with the application with docker afterwards
docker ps #use the running container
docker run -it --rm aryanvaghasiya/scicalc-app:latest
```

Listing 12: Ansible Playbook Execution

5.7.4 Ansible Deployment Output

When executed successfully, Ansible displays:

- PLAY: Shows which playbook is being executed
- TASK: Lists each task being performed
- PLAY RECAP: Summary of successful, failed, and changed tasks

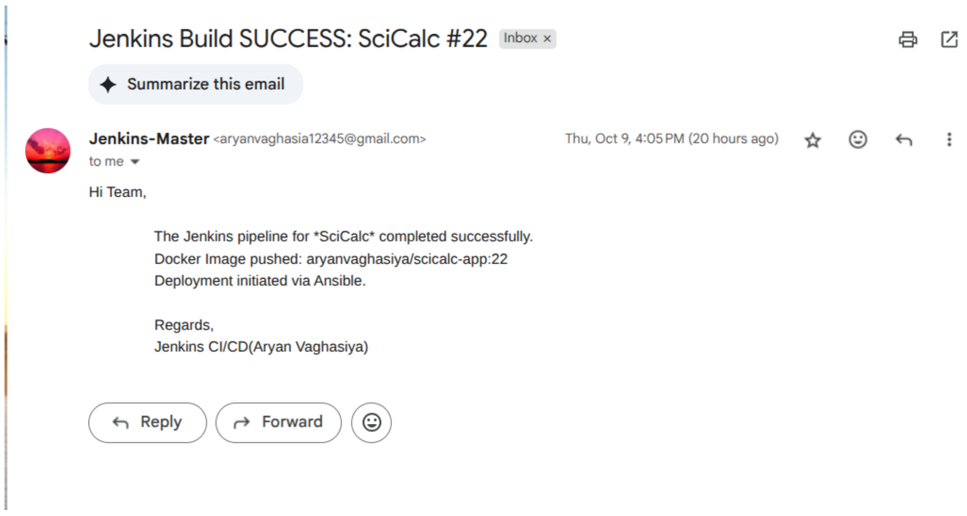


```
aryan@aryanpc:~/Downloads/SciCalc$ ansible-playbook -i hosts.ini deploy.yml -e aryanvaghasiya/scicalc-app:latest

PLAY [Deploy SciCalc Docker container] **************************************************************

TASK [Gathering Facts] ******************************************************************************
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at
/usr/bin/python3, but future installation of another Python interpreter could change the meaning
of that path. See
https://docs.ansible.com/ansible/2.10/reference_appendices/interpreter_discovery.html for more
information.
ok: [localhost]

TASK [Ensure Python 'docker' package is installed] **************************************************
ok: [localhost]

TASK [Deploy and manage SciCalc container from Docker Hub] ******************************************
[DEPRECATION WARNING]: The container_default_behavior option will change its default value from
"compatibility" to "no_defaults" in community.docker 2.0.0. To remove this warning, please
specify an explicit value for it now. This feature will be removed from community.docker in
version 2.0.0. Deprecation warnings can be disabled by setting deprecation_warnings=False in
ansible.cfg.
changed: [localhost]

PLAY RECAP ******************************************************************************************
localhost                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

aryan@aryanpc:~/Downloads/SciCalc$ docker ps
```

Figure 11: Ansible Deployment

5.7.5 Email Notification

Both on success and failure, an email is sent to aryanvaghasia12345@gmail.com notifying the result:


Jenkins Build SUCCESS: SciCalc #22

Jenkins-Master <aryanvaghasia12345@gmail.com>
to me
Thu, Oct 9, 4:05 PM (20 hours ago)

Hi Team,

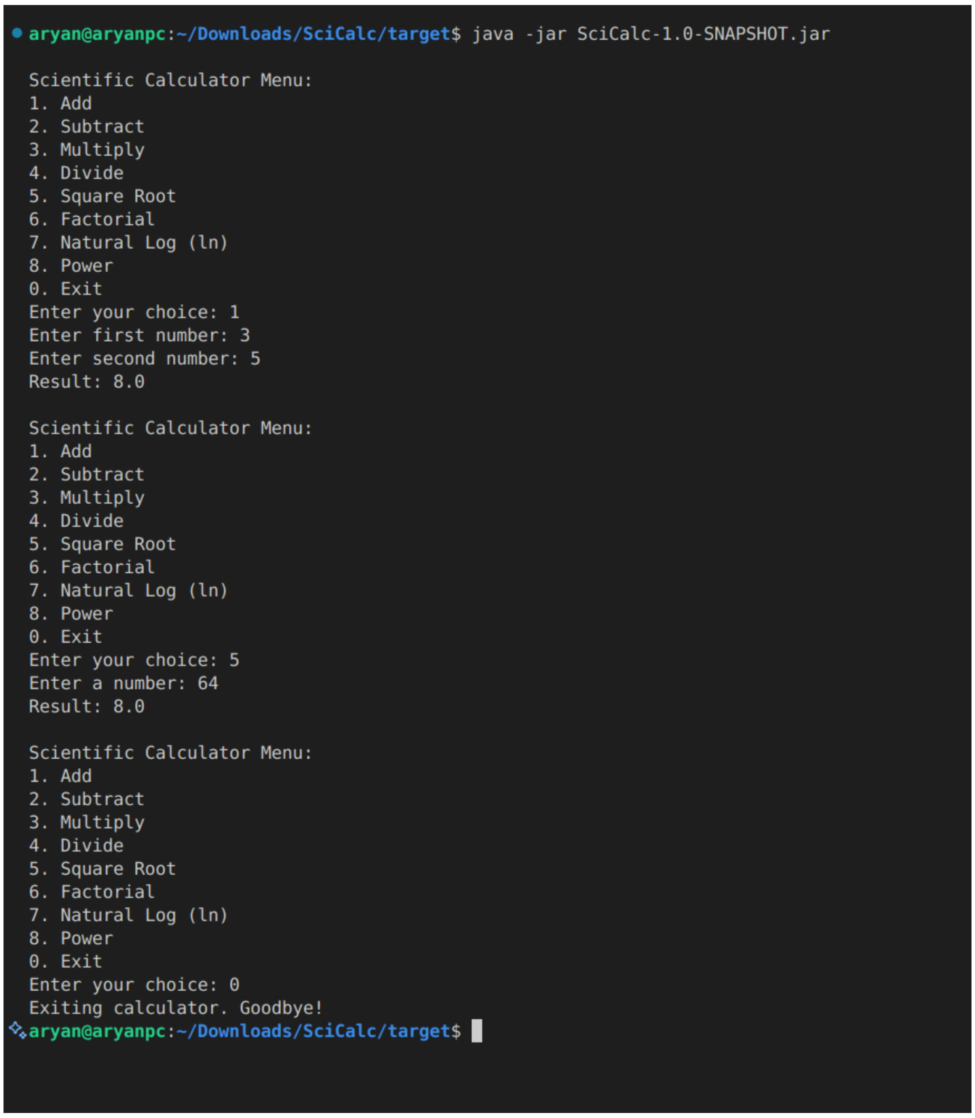
The Jenkins pipeline for *SciCalc* completed successfully.
Docker Image pushed: aryanvaghasiya/scicalc-app:22
Deployment initiated via Ansible.

Regards,
Jenkins CI/CD(Aryan Vaghasiya)

Figure 12: Email Notification

6 Application Output

The scientific calculator application runs inside a Docker container and provides an interactive command-line interface. Users can perform the following operations:



```
aryan@aryanpc:~/Downloads/SciCalc/target$ java -jar SciCalc-1.0-SNAPSHOT.jar

Scientific Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Square Root
6. Factorial
7. Natural Log (ln)
8. Power
0. Exit
Enter your choice: 1
Enter first number: 3
Enter second number: 5
Result: 8.0

Scientific Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Square Root
6. Factorial
7. Natural Log (ln)
8. Power
0. Exit
Enter your choice: 5
Enter a number: 64
Result: 8.0

Scientific Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Square Root
6. Factorial
7. Natural Log (ln)
8. Power
0. Exit
Enter your choice: 0
Exiting calculator. Goodbye!
aryan@aryanpc:~/Downloads/SciCalc/target$
```

Figure 13: Application output

7 Conclusion

7.1 Project Summary

This project successfully implemented a scientific calculator application with a complete end-to-end DevOps pipeline. The pipeline automates the entire software delivery process from code commit to deployment:

1. **Code Push:** Developer pushes code to GitHub
2. **Webhook Trigger:** GitHub webhook automatically notifies Jenkins via ngrok
3. **Pipeline Execution:** Jenkinsfile plays the entire pipeline with visual stage view
4. **Code Checkout:** Jenkins pulls latest code from GitHub repository
5. **Automated Testing:** JUnit test cases validate code functionality

6. **Image Building:** Docker creates containerized application image
7. **Registry Push:** Image is pushed to Docker Hub after successful authentication
8. **Deployment:** Ansible automates deployment on local system
9. **Notification:** Email alerts inform about pipeline success or failure

This implementation demonstrates industry-standard DevOps practices including source control, automated testing, containerization, and infrastructure as code.

7.2 Project Links

Github Repo: SciCalc App GitHub Repository
DockerHub Image: SciCalc App Docker Hub Image

8 References

1. **Git:** https://git-scm.com/doc
2. **JUnit 5:** https://junit.org/junit5/docs/current/user-guide/
3. **Maven:** https://maven.apache.org/guides/
4. **Jenkins:** https://www.jenkins.io/doc/
5. **Docker:** https://docs.docker.com
6. **Docker Hub:** https://docs.docker.com/docker-hub/
7. **Ansible:** https://docs.ansible.com
8. **ngrok:** https://ngrok.com/docs
9. **DevOps Overview:** https://aws.amazon.com/devops/what-is-devops/
10. **GitHub Webhooks:** https://docs.github.com/en/webhooks