International Institute of Information Technology, Bangalore

Software Production Engineering Mini Project Report

Scientific Calculator

Submitted by
Bhagya Shah (MT2024135)

Under the guidance of
Prof. B. Thangaraju

Contents

1 Abstract

The Scientific Calculator project is a software application that helps perform various mathematical and scientific operations such as square root, logarithms, exponentiation, and factorial calculations.

This project follows DevOps practices to make development, testing, and deployment easier and more efficient. By using continuous integration and deployment (CI/CD) pipelines, automated testing, and containerization, the project ensures reliability and scalability.

With the help of DevOps tools like GitHub Actions, Docker, Jenkins, and Ansible, the project automates processes, making it easier to maintain and collaborate.

In simple terms, the Scientific Calculator project is a small calculator application that performs mathematical operations and is developed using modern software practices to ensure automation, testing, and smooth deployment.

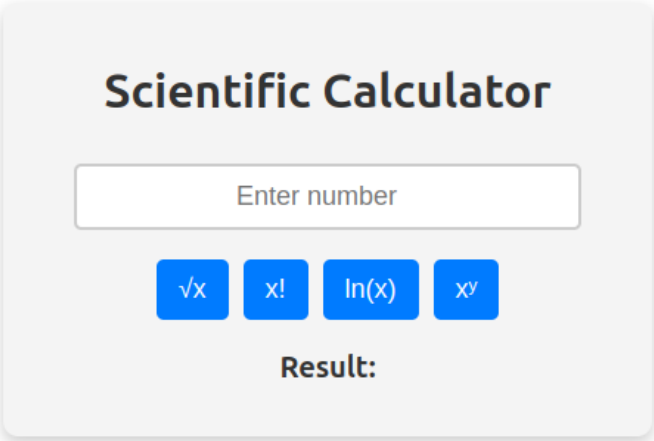

Figure 1: UI

2 DevOps

2.1 What is DevOps

DevOps is a software development practice that combines development and operations teams to work together throughout the application lifecycle. The goal of DevOps is to speed up the delivery of applications and services.

2.2 Why DevOps

DevOps improves software delivery by automating and streamlining workflows, which helps businesses deliver software faster and more efficiently. DevOps also helps improve collaboration between teams.

2.3 Benefits of DevOps

- **Faster delivery:** DevOps can help businesses release software updates more frequently.
- **Better quality:** DevOps can help reduce errors and improve product quality.
- **Better collaboration:** DevOps can help break down silos between development and operations teams.
- **Better security:** DevOps can help integrate security practices into the development process.
- **Better scalability:** DevOps can help manage scalable solutions through automation and improved workflows.
- **Better customer experience:** DevOps can help deliver high-quality software that meets customer needs.
- **Cost reduction:** DevOps can help reduce the costs associated with slow and inefficient software delivery processes.

3 Tools and Technologies Used

- **Programming Language:** Java, Javascript
- **Framework:** SpringBoot , React
- **Version Control:** Git & GitHub
- **Build Automation:** Maven
- **CI/CD Pipeline:** GitHub Actions / Jenkins
- **Containerization:** Docker
- **Testing Framework:** JUnit

4 Source Code Management

SCM is used to track modifications to a source code repository. SCM keeps track of changes to a code base and helps resolve conflicts when merging updates from multiple contributors. SCM is also synonymous with Version Control.

The entire project is first created on a local machine (repository). This project is then pushed to a remote repository on GitHub.

The series of commands are executed to push the local repository to the remote repository.

- `git init`
- `git status`
- `git add .`
- `git commit -m "Commit Message"`
- `git remote add origin https://github.com/Bhagyashah05/calc-sci-mini.git`
- `git push -u origin main`

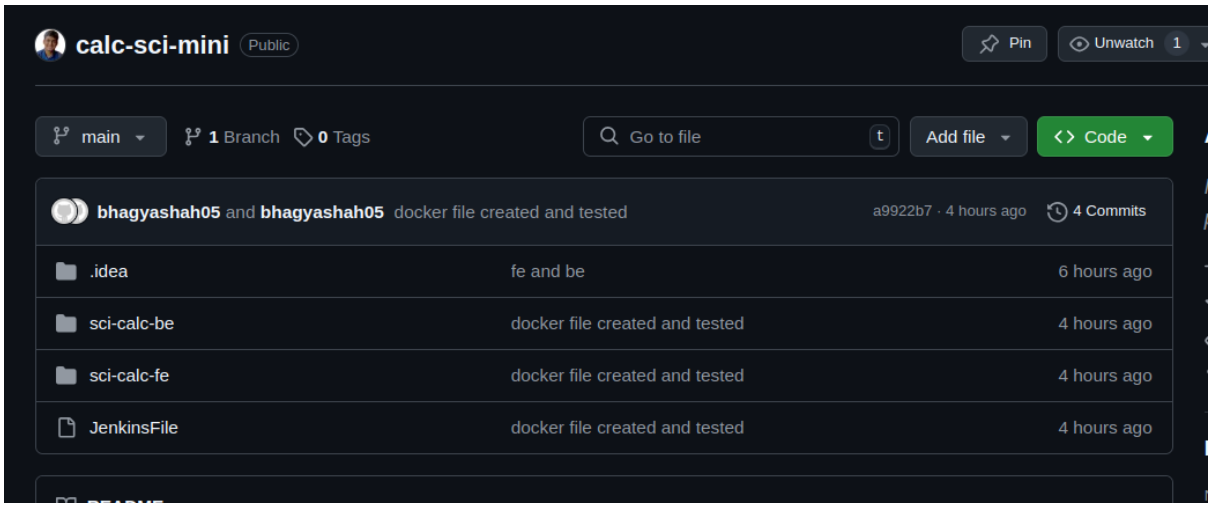

Figure 2: Git Repo

5 Docker

Docker is a tool that helps package software and run it in isolated environments called containers. It allows us to create an image that includes everything needed to run the scientific calculator application, including the JAR file built using Maven.

Once the Docker image is created, it can be uploaded to Docker Hub. Other machines can then download this image and create containers to run the application without needing to set up the environment manually.

To build the docker image and push it to Docker Hub remote repository, we will specify steps in Jenkins pipeline script. The Docker image is built and pushed on the Docker Hub repository of the user. The local image on the machine is deleted after the image is pushed on the Docker Hub.

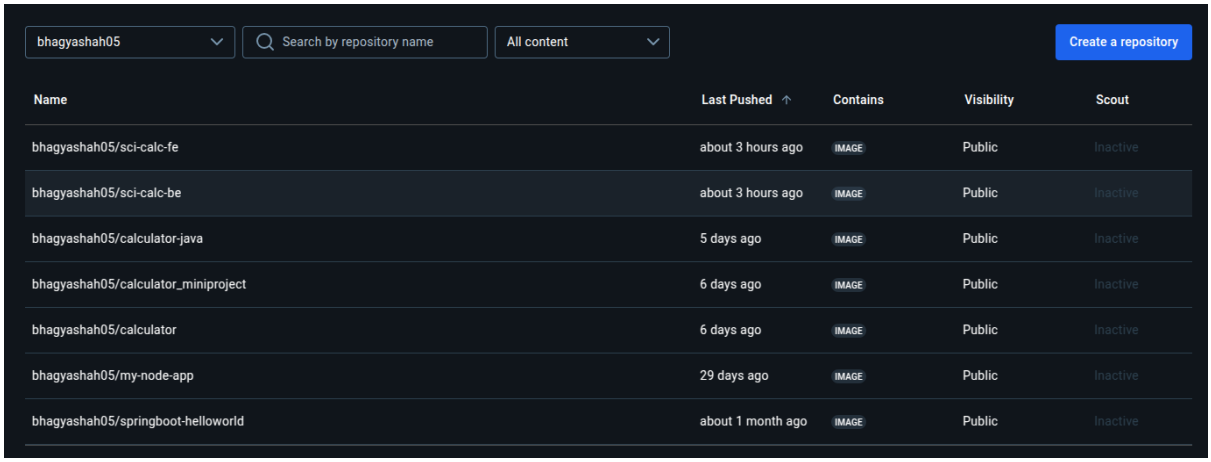

Figure 3: Docker Hub

5.1 Steps to Work with Docker

- **Build the Docker Image:**
  Run the following command to create a Docker image:

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

- **Push the Image to Docker Hub:**
  After creating the image, upload it to Docker Hub using:

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

- **Run the Docker Container:**
  To run the application using the Docker image, use the command:

  ```
  docker run -it <IMAGE_NAME>
  ```

- **Automating with Jenkins and Ansible:**

  - Jenkins builds the project and creates the Docker image.
  - The image is then pushed to Docker Hub automatically.
  - Ansible pulls the image from Docker Hub and deploys it on the server.

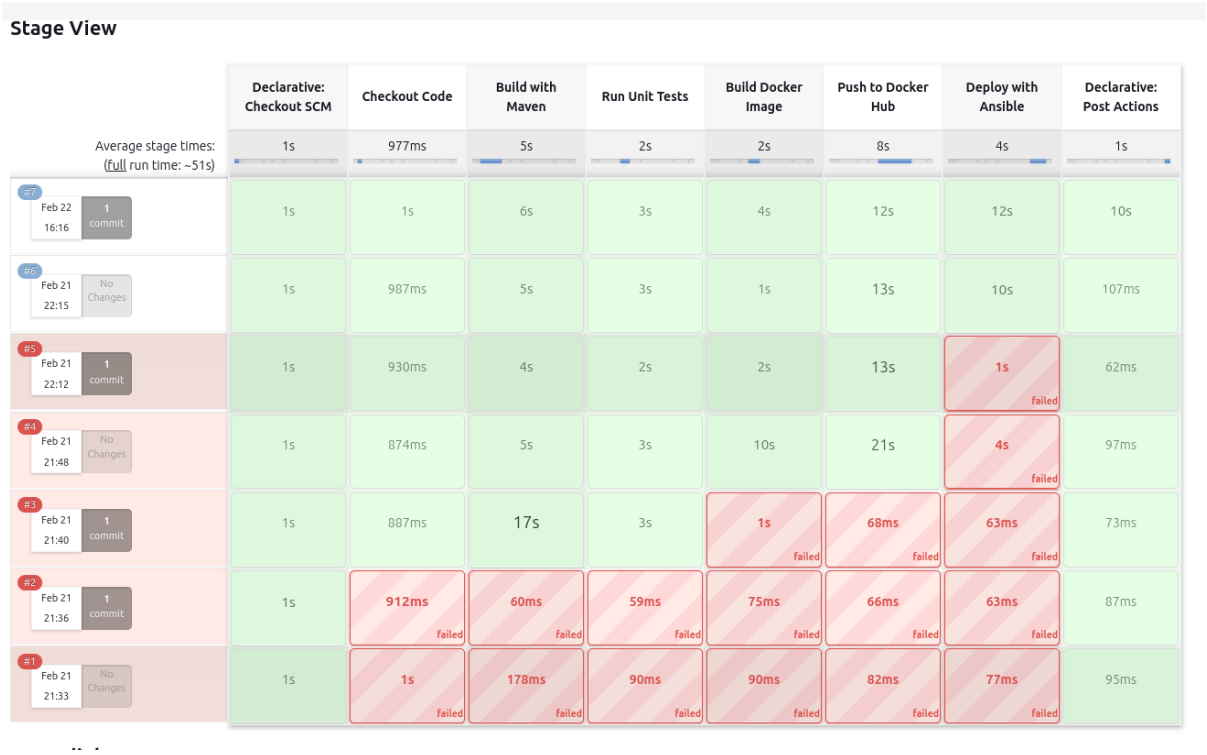

Figure 4: Stage View

6 Jenkins and CI/CD Pipeline

6.1 What is Jenkins?

Jenkins is an open-source automation server that helps automate the process of building, testing, and deploying applications. It supports continuous integration (CI) and continuous deployment (CD), ensuring that software is built and tested automatically before being deployed.

6.2 Why Use Jenkins?

- **Automation:** Reduces manual effort in software development.
- **Continuous Integration:** Automatically tests and builds code after every change.
- **Continuous Deployment:** Ensures that applications are deployed efficiently.
- **Scalability:** Can be integrated with various tools like Docker, Ansible, and GitHub.
- **Error Detection:** Quickly identifies and notifies developers about build failures.

6.3 CI/CD Pipeline in Jenkins

The pipeline automates the entire software deployment process, ensuring that each step runs smoothly. Below is the breakdown of the pipeline:

- **Agent Definition:** The pipeline runs on any available Jenkins agent.
- **Environment Variables:** Defines variables for Docker Hub user and image names.

6.4 Stages of the Pipeline Used in Building Project

1. **Clone Repository:** Jenkins pulls the latest code from the GitHub repository using the command:

   ```
   git clone https://github.com/Bhagyashah05/calc-sci-mini.git
   ```

2. **Build Backend:** Inside the backend directory, Jenkins runs Maven to clean and package the application:

   ```
   mvn clean package
   ```

3. **Run Backend Tests:** Jenkins executes unit tests to verify that the backend code works correctly:

   ```
   mvn test
   ```

4. **Build Backend Docker Image:** Jenkins builds a Docker image for the backend service:

```
docker build -t bhagyashah05/sci-calc-be:latest .
```

5. **Build Frontend Docker Image:** Similar to the backend, Jenkins builds an image for the frontend:

```
docker build -t bhagyashah05/sci-calc-fe:latest .
```

6. **Push Backend Image to DockerHub:** The built backend image is pushed to DockerHub for deployment:

```
docker push bhagyashah05/sci-calc-be:latest
```

7. **Push Frontend Image to DockerHub:** The frontend image is also uploaded to DockerHub:

```
docker push bhagyashah05/sci-calc-fe:latest
```

8. **Run Ansible Playbook:** Ansible is used to deploy the application on the target server:

```
ansible-playbook -i inventory deploy.yml
```

6.5 Post-Processing

- **Success Notification:** Sends an email if the deployment is successful.
- **Failure Notification:** Sends an email if the build fails.
- **Cleanup:** Jenkins cleans up workspace files after execution.

This automated pipeline ensures smooth and reliable deployment, reducing manual intervention and improving software quality.

7 Setting Up Credentials in Jenkins

Jenkins requires credentials to authenticate with GitHub for fetching repositories and DockerHub for pushing images. This section outlines the steps to configure both.

7.1 Adding GitHub Credentials

To allow Jenkins to access your GitHub repository, follow these steps:

1. Open Jenkins and go to **Manage Jenkins** → **Manage Credentials**.
2. Click on **(global)** under **Stores scoped to Jenkins**.
3. Click on **Add Credentials**.
4. Select **Username with password** as the credential type.
5. Enter the following details:
   - **Username**: Your GitHub username.
   - **Password**: Your GitHub personal access token.
   - **ID**: Enter `github-credentials` (to reference in Jenkinsfile).
6. Click **OK** to save.

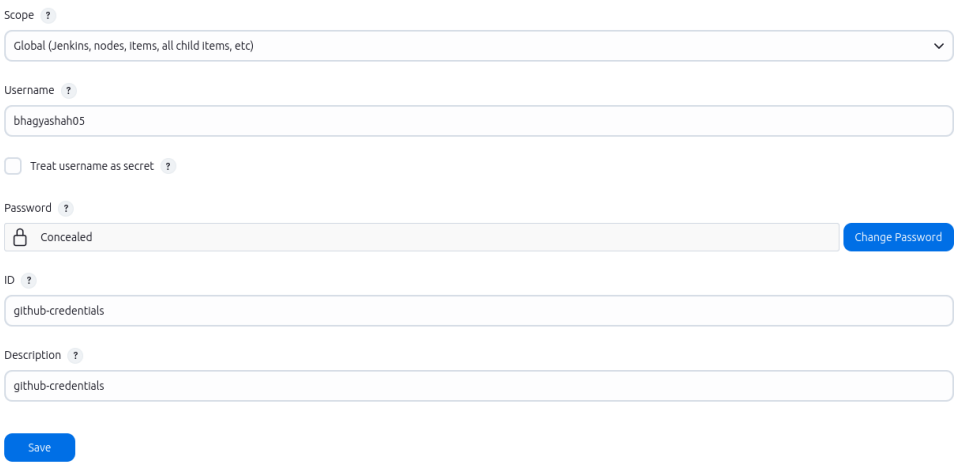

Figure 5: Adding GitHub Credentials in Jenkins

7.2 Adding DockerHub Credentials

To authenticate with DockerHub for pushing images, follow these steps:

1. Open Jenkins and navigate to **Manage Jenkins** → **Manage Credentials**.
2. Click on **(global)** under **Stores scoped to Jenkins**.
3. Click on **Add Credentials**.

4. Select **Username with password** as the credential type.
5. Enter the following details:
    - **Username**: Your DockerHub username.
    - **Password**: Your DockerHub password.
    - **ID**: Enter `docker-credentials` (to reference in Jenkinsfile).
6. Click **OK** to save.

Figure 6: Adding DockerHub Credentials in Jenkins

7.3 Verifying Credentials

To ensure Jenkins can use the credentials:

- Go to **Manage Jenkins** → **Manage Credentials**.
- Locate `github-credentials` and `docker-credentials`.
- If they appear correctly, the setup is complete.

This setup ensures that Jenkins can securely interact with GitHub and DockerHub for seamless CI/CD.

8 Creating a New Pipeline Project in Jenkins for Scientific Calculator

To automate the deployment process, Jenkins allows users to create a pipeline project that integrates with GitHub and executes the pipeline defined in a `Jenkinsfile`. The following steps outline the setup process.

8.1 Step 1: Install Required Plugins

Before setting up a pipeline, ensure the following plugins are installed in Jenkins:

- **Pipeline Plugin** - Enables Jenkins pipeline functionality.
- **Git Plugin** - Allows integration with GitHub repositories.
- **Docker Plugin** - Required for building and pushing Docker images.
- **Ansible Plugin** - Needed if using Ansible for deployment.

8.2 Step 2: Create a New Pipeline Project

1. Open Jenkins and log in with administrator credentials.
2. Click on **New Item** from the Jenkins dashboard.
3. Enter a name for the pipeline project (e.g., `Sci-Calc-Pipeline`).
4. Select **Pipeline** and click **OK**.

8.3 Step 3: Configure GitHub Integration

1. In the project configuration, navigate to the **Pipeline** section.
2. Select **Pipeline script from SCM (Source Code Management)**.
3. Choose **Git** and enter the repository URL:

   ```
   https://github.com/Bhagyashah05/calc-sci-mini.git
   ```

4. Click on **Add** next to **Credentials** and select **Jenkins**.
5. Click on **Add Credentials** and choose:
   - **Kind**: Username with password
   - **Username**: Your GitHub username
   - **Password**: Your GitHub personal access token
   - **ID**: `github-credentials`
6. Click **OK**, then select `github-credentials` from the dropdown.
7. In the **Branch Specifier**, enter:

   ```
   main
   ```

8.4 Step 4: Define the Jenkins Pipeline

1. Ensure the repository contains a file named `Jenkinsfile`.
2. Inside Jenkins, under **Script Path**, enter:

   ```
   Jenkinsfile
   ```

3. Save the configuration.

8.5 Step 6: Build and Run the Pipeline

1. Go to the Jenkins pipeline project.
2. Click **Build Now** to trigger the pipeline manually.
3. View logs in **Console Output** to monitor progress.
4. Once the setup is complete, any code push to GitHub will trigger an automatic build.

8.6 Step 7: Verify Deployment

1. Check if the Docker images are pushed to DockerHub.
2. Verify the deployed application using the defined Ansible playbook.
3. If any stage fails, inspect the logs and re-run the pipeline after fixing issues.

This setup ensures that every code change is automatically built, tested, and deployed, improving the efficiency and reliability of software development.

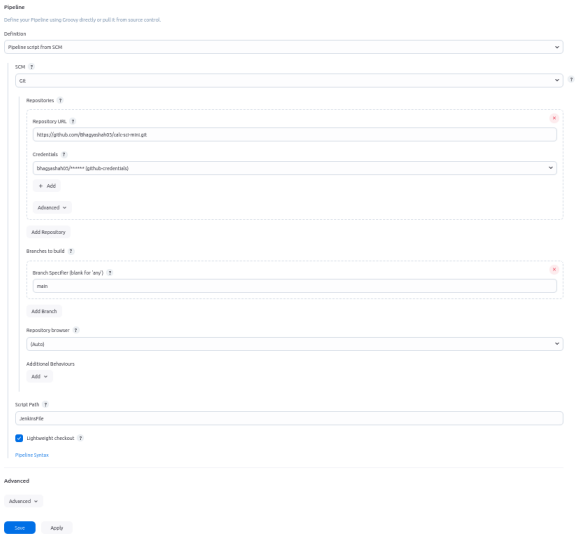

Figure 7: Creating Pipeline in Jenkins

9 Ansible and Its Role in Deployment

9.1 What is Ansible?

Ansible is an open-source automation tool that simplifies configuration management, application deployment, and task automation. It allows infrastructure as code (IaC), ensuring consistent and repeatable setups across multiple servers.

9.2 Why Use Ansible for Deployment?

- **Agentless** - Unlike other configuration management tools, Ansible does not require an agent on managed servers.
- **Idempotency** - Ensures tasks are executed only when required, avoiding unnecessary changes.
- **Automation** - Automates deployment, reducing manual errors and saving time.
- **Scalability** - Easily manages and deploys applications on multiple servers.
- **Docker Integration** - Provides modules for managing Docker containers efficiently.

9.3 Ansible Playbook for Deploying Sci-Calc App

The following Ansible playbook automates the deployment of the Scientific Calculator application using Docker.

9.4 Explanation of Each Task

- **hosts: servers** - Specifies the target servers where the playbook will execute.
- **become: true** - Grants administrator privileges (sudo) for executing commands.
- **Pull Backend Docker Image**
  - Uses the `docker_image` module to pull the latest backend image.
  - Ensures the correct version of the backend application is retrieved from Docker Hub.
- **Pull Frontend Docker Image**
  - Similar to the backend task, this pulls the latest frontend image.
- **Create a Docker Network**
  - Uses the `docker_network` module to create an isolated network for inter-container communication.
  - Ensures seamless communication between the frontend and backend containers.
- **Run Backend Container**

- Uses the `docker_container` module to start the backend service.
- `name: backend` - Names the container as "backend".
- `image: bhagyashah05/sci-calc-be:latest` - Specifies the image to use.
- `state: started` - Ensures the container is running.
- `restart_policy: always` - Automatically restarts the container if it stops.
- `networks: sci-calc-network` - Connects the container to the created network.
- `ports: 8090:8090` - Maps port 8090 on the host to 8090 in the container.

- **Run Frontend Container**
  - Similar to the backend task, this starts the frontend service.
  - Uses port mapping `3000:80` to expose the frontend on port 3000.

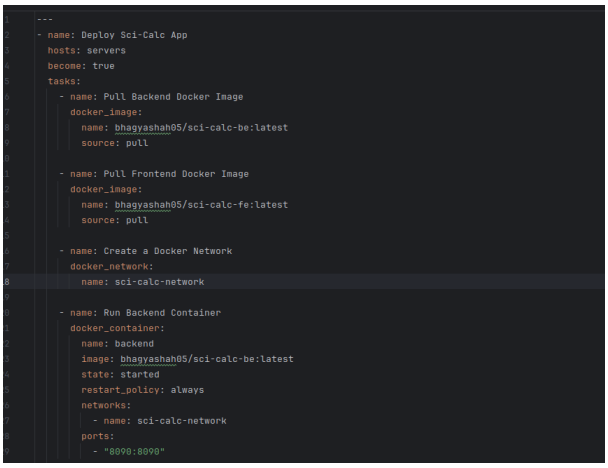

Figure 8: Deployment.yml file

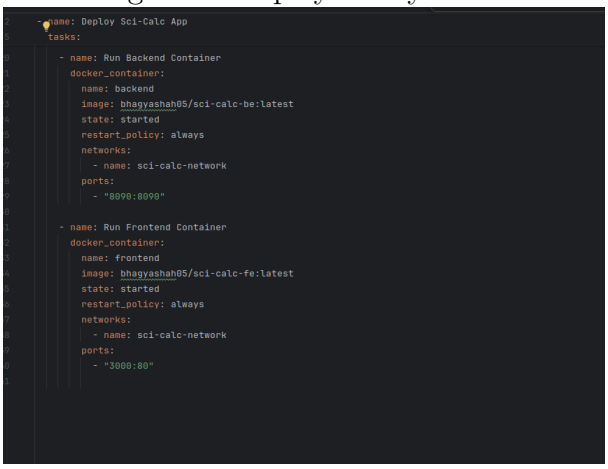

Figure 9: Deployment.yml file

9.5 Final Outcome

After running the playbook, the backend and frontend containers will be deployed, running inside the `sci-calc-network`. The frontend will be accessible at:

`http://localhost:3000`

and the backend at:

`http://localhost:8090`

This playbook ensures a fully automated deployment of the Sci-Calc app, improving efficiency and reducing deployment errors.