Software Production Engineering Mini-Project Report

Vikas Kalyanapuram
IMT2021040

September 26, 2024

1 Links and References

1. **GitHub Repo**: https://github.com/LieutPaul/SPE-MiniProject
2. **Dockerhub**: https://hub.docker.com/u/vikaskaly
3. Fixing **docker not found** error in Jenkins on Mac: Here

2 Introduction

2.1 Description of the Application

The aim of the project was to create a simple Command line menu driven calculator program that does the following operations:

- Square root function ($\sqrt{x}$)
- Factorial function ($x!$)
- Natural log function ($ln(x)$)
- Power function (a^b)

2.2 Objective

The project was more focused on the DevOps tools that are required to develop and deploy the application. The following were the functionalities that needed to be implemented:

1. To create the calculator application.
2. To build and test the application.
3. To create a local version control repository and push the local repository to a remote version control repository.

4. To create a pipeline that, everytime there is a change in the remote repository, automates the building and testing of the application.

5. To automatically containerise the project and push the image to a remote container repository.

6. To deploy the container onto a target machine.

2.3 Workflow and tools Used

1. **Code Development**: The process of writing the code to achieve the required functionality of the application (**Java**)

2. **Testing**: Writing Unit Tests to test the functionality of the application (**JUnit**)

3. **Version Control**: To manage different versions of the application to track changes. (**Git and GitHub**)

4. **Building**: Compiling the code, managing dependencies, and packaging the application for deployment (**Maven: mvn**)

5. **Continuous Integration and Continuous Deployment (CI/CD)**: Automating the process of building, testing, and deploying software, so that changes can be quickly and easily integrated into the production environment (**Jenkins**). For Git SCM Polling and Build Automation, we use **Ngrok and GitHub webhooks**.

6. **Containerisation**: Building an image out of an application to ensure smooth running of the application across platforms and devices (**Docker**)

7. **Deployment**: Pulling the docker image whenever a new image is pushed to the remote container registry and deploying the container (**Ansible**)

3 Installing the tools

1. **Java**: Java for Macs can be installed from The Official Website.

2. **GitHub**: Visit www.github.com to create an account.

3. **Maven**: To install Maven on Mac, we use the following command;

   ```
   brew install maven
   ```

   To verify if Maven is installed and know the version we use the following.

   ```
   mvn --version
   ```

4. **Jenkins**: Note that Jenkins requires Java to be installed on the system. To install Jenkins, we use the following command:

```
brew install jenkins-lts
```

After installing, you can start the jenkins by executing the following command:

```
brew services start jenkins
```

Once this is done, the Jenkins web interface can be accessed at http://localhost:8080/

5. **Docker**: Docker can be installed from docker's official website: Official Website. To verify the docker installation, we can run:

   ```
   docker --version
   ```

6. **Ansible**: To install ansible, we run:

   ```
   pip install ansible
   ```

   Note that if your machine has python virtual environments, source to that venv and then run the above command.

7. **Ngrok**: First visit ngrok's official website to sign up and get your auth token. Then, follow the steps outlined Here.

4 DevOps

4.1 What is DevOps?

DevOps is a set of practices and tools to integrate and automate the work of software development (Dev) and IT operations (Ops) as a means for improving and shortening the systems development life cycle. The DevOps approach was designed to ensure that high-quality updated software gets into the hands of users quickly while ensuring minimal human interaction in the building and deployment phases by automating the pipeline. The different tools used in DevOps can be found in fig 1.

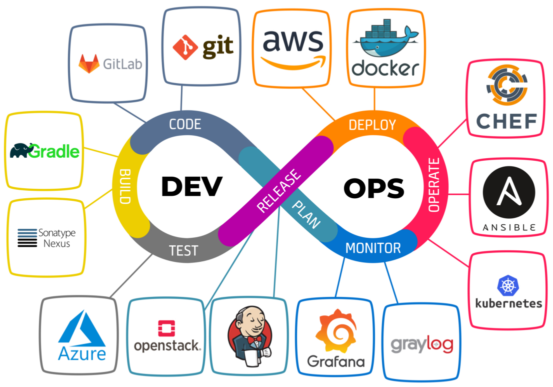


Figure 1: DevOps

4.2 Why do we need DevOps?

DevOps is essential because it bridges the gap between development and operations teams, improving collaboration and communication. Traditionally, software development and IT operations were siloed, which often led to bottlenecks, slower delivery times, and more errors during deployment. By adopting a DevOps approach, organizations can:

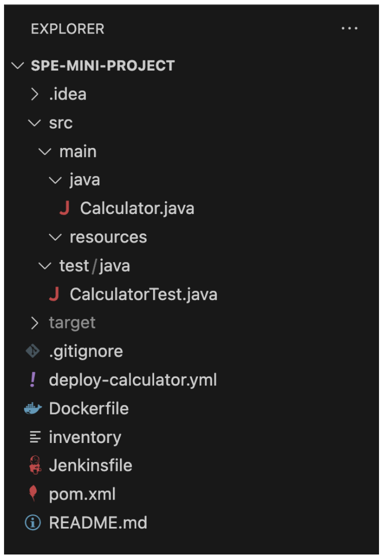
- **Accelerate time to market:** Automation of the build, test, and deployment pipelines reduces the time needed to bring new features or updates to production, allowing companies to release software more frequently and reliably.
- **Improve collaboration:** DevOps fosters a culture of shared responsibility and transparency between teams, enabling better coordination and fewer misunderstandings.
- **Enhance product quality:** Continuous integration and continuous delivery (CI/CD) pipelines allow developers to detect and fix bugs early in the development cycle, improving software stability and quality.
- **Increase efficiency:** Automation of repetitive tasks such as testing, deployment, and infrastructure management reduces manual work, freeing teams to focus on higher-value tasks.
- **Enable scalability:** DevOps practices help manage infrastructure more effectively, allowing organizations to scale their services and infrastructure quickly in response to growing demand.

4.3 Tools used in DevOps

- **Version Control:** Tools like *Git*, *GitHub*, and *GitLab* are used for version control, enabling teams to manage and track code changes collaboratively. It ensures that developers can work on different parts of the codebase simultaneously.
- **Continuous Integration/Continuous Deployment (CI/CD):** *Jenkins* is a popular tool for automating the testing and deployment of code. These tools help integrate code changes frequently and automatically deploy updates to production environments.
- **Configuration Management:** Tools like *Ansible* help automate the management of system configurations. These tools ensure that environments are consistent across development, testing, and production.
- **Containerization and Orchestration:** *Docker* is widely used for containerization, which allows developers to package applications with their dependencies in isolated containers. For managing and orchestrating containers at scale, *Kubernetes* is commonly used to automate container deployment, scaling, and operations.
- **Monitoring and Logging:** *Prometheus*, *Nagios*, *ELK Stack (Elasticsearch, Logstash, Kibana)*, and *Grafana* are widely used to monitor system performance and analyze logs, providing insight into the health of applications and infrastructure in real time.

5 Code Development and Testing

The project code mainly involved the code for the calculator, the tests, pom.xml and the various files to configure the CI/CD pipeline. The project strucutre is shown in fig 2. **src** and **target** are the 2 main folders in the project. **src** is where all the code resides and **target** is where all the outputs and build files are stored. Inside the src directory, there **main/java** where the project code resides and **test/java** where the tests code resides.

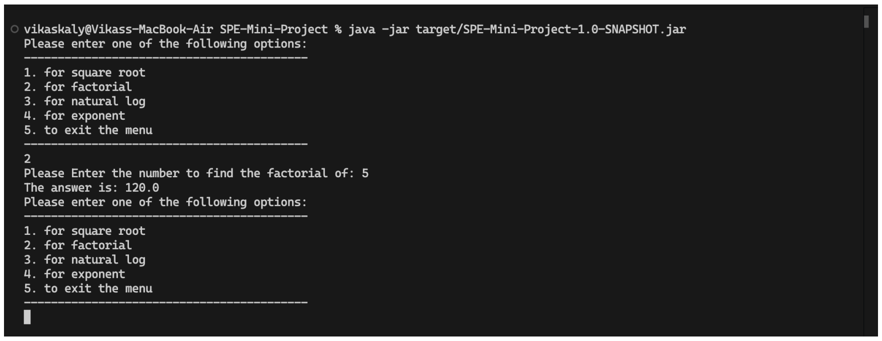


```
EXPLORER                        ...
∨ SPE-MINI-PROJECT
  > .idea
  ∨ src
    ∨ main
      ∨ java
         Calculator.java
      ∨ resources
    ∨ test/java
       CalculatorTest.java
  > target
  .gitignore
  deploy-calculator.yml
  Dockerfile
  inventory
  Jenkinsfile
  pom.xml
  README.md
```

Figure 2: Code Structure

5.1 Calculator.java

The calculator application is a menu driven program that asks the user the operation to perform and the number(s) to perform the operations on. The menu keeps running until the user exits.
The output of the calculator program can be seen in fig 3.



```
vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % java -jar target/SPE-Mini-Project-1.0-SNAPSHOT.jar
Please enter one of the following options:
------------------------------------------
1. for square root
2. for factorial
3. for natural log
4. for exponent
5. to exit the menu
------------------------------------------
2
Please Enter the number to find the factorial of: 5
The answer is: 120.0
Please enter one of the following options:
------------------------------------------
1. for square root
2. for factorial
3. for natural log
4. for exponent
5. to exit the menu
------------------------------------------
```

Figure 3: Calculator Menu

5.2 CalculatorTest.java

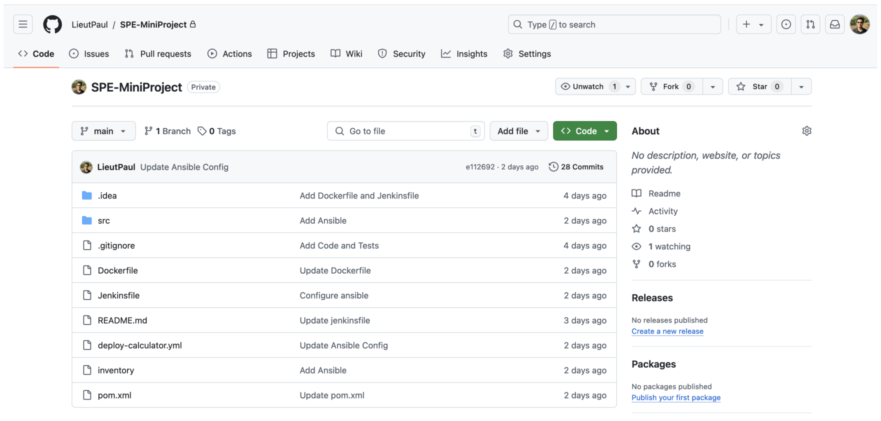
This file contains all the unit tests to test the functionality of the Calculator.java code. The tests were written in JUnit.

6 Version Control - Git and GitHub

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. These systems provide a structured and collaborative environment, allowing multiple developers to work on the same project, avoid conflicts, and maintain a history of the changes made.
Git is a distributed version control system that tracks versions of files. It is widely used in software development for managing source code and is popular due to its efficiency, speed, and flexibility. Unlike centralized version control systems, Git allows every developer to have a full copy of the repository on their machine, including the entire history of changes.
GitHub is a cloud-based hosting service that lets you manage Git repositories. It allows developers to store, share, and collaborate on Git repositories over the internet. GitHub provides features like pull requests, issues, and project management tools that make collaborative development easier.
Some commonly used git commands are:

1. **git init**: Initializes a new Git repository in the current directory. This command creates a .git subdirectory where Git stores all the metadata for version control.
2. **git clone**: Creates a local copy of a remote repository. You can clone a repository from GitHub or any other Git hosting service.
3. **git status**: Shows the current state of the working directory and staging area. It lets you see which files have been modified, staged, or untracked.
4. **git add**: Stages changes by adding files or updates to the staging area in preparation for a commit.
5. **git commit**: Saves the staged changes as a new commit with a descriptive message. The commit acts as a snapshot of your project at a specific point in time.
6. **git push**: Sends local commits to a remote repository, such as GitHub. You can push to the main branch or any other branch you are working on.
7. **git pull**: Fetches updates from a remote repository and merges them into your local repository. This is useful for keeping your local repository up-to-date.

Screenshot of my GitHub repo for this project can be found in fig 4.



No description, website, or topics provided.

No releases published

No packages published

LieutPaul Update Ansible Config e112692 · 2 days ago 28 Commits

.idea	Add Dockerfile and Jenkinsfile	4 days ago
src	Add Ansible	2 days ago
.gitignore	Add Code and Tests	4 days ago
Dockerfile	Update Dockerfile	2 days ago
Jenkinsfile	Configure ansible	2 days ago
README.md	Update jenkinsfile	3 days ago
deploy-calculator.yml	Update Ansible Config	2 days ago
inventory	Add Ansible	2 days ago
pom.xml	Update pom.xml	2 days ago

Figure 4: My GitHub repo

7 Build Tool - Maven

Maven is a powerful build automation tool primarily used for Java projects. It was developed by the Apache Software Foundation and helps manage a project's build process, dependencies, documentation, and reporting in a uniform way.

7.1 What is Maven Used For

Maven simplifies the process of building and managing projects by automating repetitive tasks, such as compiling code, packaging binaries, running tests, generating documentation, and managing dependencies. Here are some common use cases:

1. **Project Build Automation**: Maven automates the entire build process, including compiling source code, running tests, packaging the application into a JAR file.
2. **Dependency Management**: One of Maven's core features is its ability to manage project dependencies which are specified in a *pom.xml* file. It allows you to declare libraries your project relies on, and Maven automatically downloads them from a central repository. This eliminates the need to manually search for and add JAR files.

7.2 Running the project using Maven

1.
   ```
   mvn clean install
   ```

   This command does the following:

   (a) Cleans up previous build artifacts.

   (b) Downloads all project dependencies defined in the pom.xml file from the central Maven repository (or any other specified repository) if they are not already available in the local repository.

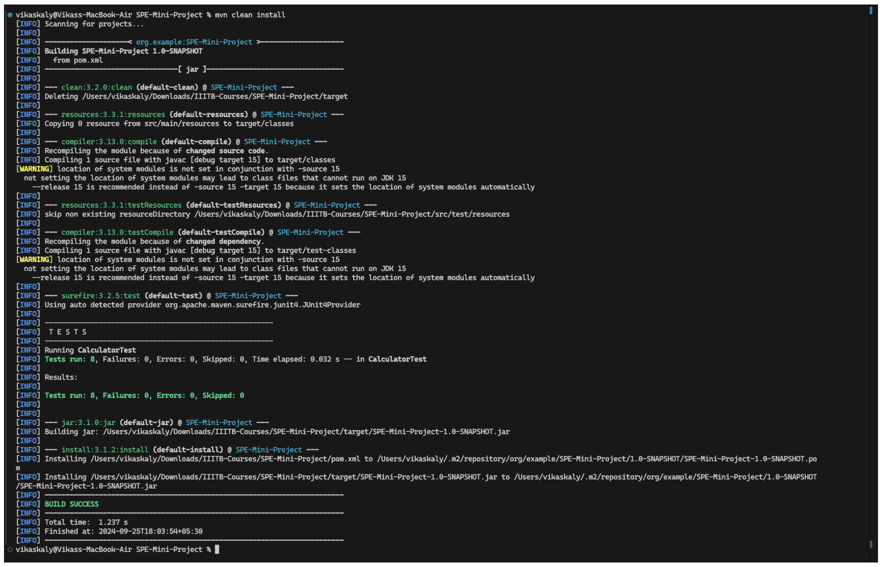
   (c) Compiles the source code of the project.

(d) Runs all unit tests to verify the correctness of the code.

(e) Packages the project (e.g., into a JAR file).

(f) Places the packaged JAR file into the *target/* directory

The output of this command can be seen in fig 5

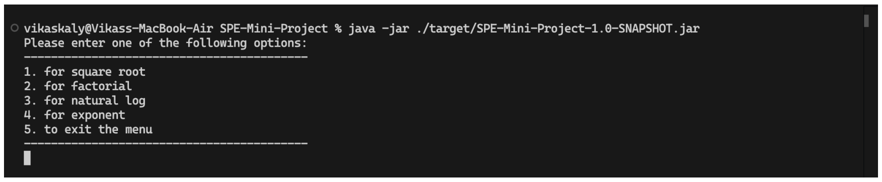


```
vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------< org.example:SPE-Mini-Project >-------------------
[INFO] Building SPE-Mini-Project 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ SPE-Mini-Project ---
[INFO] Deleting /Users/vikaskaly/Downloads/IIITB-Courses/SPE-Mini-Project/target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ SPE-Mini-Project ---
[INFO] Copying 0 resource from src/main/resources to target/classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ SPE-Mini-Project ---
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 1 source file with javac [debug target 15] to target/classes
[WARNING] location of system modules is not set in conjunction with -source 15
  not setting the location of system modules may lead to class files that cannot run on JDK 15
    --release 15 is recommended instead of -source 15 -target 15 because it sets the location of system modules automatically
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ SPE-Mini-Project ---
[INFO] skip non existing resourceDirectory /Users/vikaskaly/Downloads/IIITB-Courses/SPE-Mini-Project/src/test/resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ SPE-Mini-Project ---
[INFO] Recompiling the module because of changed dependency.
[INFO] Compiling 1 source file with javac [debug target 15] to target/test-classes
[WARNING] location of system modules is not set in conjunction with -source 15
  not setting the location of system modules may lead to class files that cannot run on JDK 15
    --release 15 is recommended instead of -source 15 -target 15 because it sets the location of system modules automatically
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ SPE-Mini-Project ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit4.JUnit4Provider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running CalculatorTest
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.032 s -- in CalculatorTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.1.0:jar (default-jar) @ SPE-Mini-Project ---
[INFO] Building jar: /Users/vikaskaly/Downloads/IIITB-Courses/SPE-Mini-Project/target/SPE-Mini-Project-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- install:3.1.2:install (default-install) @ SPE-Mini-Project ---
[INFO] Installing /Users/vikaskaly/Downloads/IIITB-Courses/SPE-Mini-Project/pom.xml to /Users/vikaskaly/.m2/repository/org/example/SPE-Mini-Project/1.0-SNAPSHOT/SPE-Mini-Project-1.0-SNAPSHOT.po
m
[INFO] Installing /Users/vikaskaly/Downloads/IIITB-Courses/SPE-Mini-Project/target/SPE-Mini-Project-1.0-SNAPSHOT.jar to /Users/vikaskaly/.m2/repository/org/example/SPE-Mini-Project/1.0-SNAPSHOT
/SPE-Mini-Project-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.237 s
[INFO] Finished at: 2024-09-25T18:03:54+05:30
[INFO] ------------------------------------------------------------------------
vikaskaly@Vikass-MacBook-Air SPE-Mini-Project %
```

Figure 5: mvn clean install

2.
```
java -jar ./target/<target-jar-name>.jar
```

This command is used to run the packaged Java application (the JAR file) that was built by Maven. The output of this command can be found in fig 6



```
vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % java -jar ./target/SPE-Mini-Project-1.0-SNAPSHOT.jar
Please enter one of the following options:
-------------------------------------------
1. for square root
2. for factorial
3. for natural log
4. for exponent
5. to exit the menu
-------------------------------------------
```

Figure 6: java -jar output

8 Containerization

Containerization is a lightweight form of virtualization that allows applications to run in isolated environments called containers. A container bundles an application and its dependencies (such as libraries, frameworks, and configurations) into a single, portable unit that can run consistently across different computing environments. Unlike virtual machines (VMs), containers do not require a full operating system (OS) for each instance, making them more efficient in terms of resource usage and performance.

8.1 Benefits of Containerization

1. **Consistency Across Environments**: One of the biggest challenges in software development is ensuring that applications work the same way in different environments (development, staging, and production). Containers solve this problem by packaging the application with all its dependencies, making it portable and consistent across any environment.

2. **Isolation**: Containers isolate applications and their dependencies, ensuring that different applications running on the same host do not interfere with each other.

3. **Fast Deployment**: Containers start up almost instantly compared to virtual machines. This makes it easier to deploy and test applications rapidly, enhancing the development workflow.

4. **CI/CD Integration**: Docker integrates seamlessly with Continuous Integration/Continuous Deployment (CI/CD) pipelines, allowing developers to automate the testing and deployment of their containerized applications.

8.2 Docker

Docker is one of the most widely used platforms for containerization. It simplifies the creation, deployment, and management of containers. With Docker, developers can package their applications, along with all necessary dependencies, into a Docker image. This image can then be shared and deployed consistently across different environments, ensuring that the application runs exactly the same way regardless of where it's executed (development, testing, or production).
Key components of Docker are:

1. **Dockerfile**: A text file that contains the instructions for building a Docker image. It specifies the base image, dependencies, and the commands to run the application.

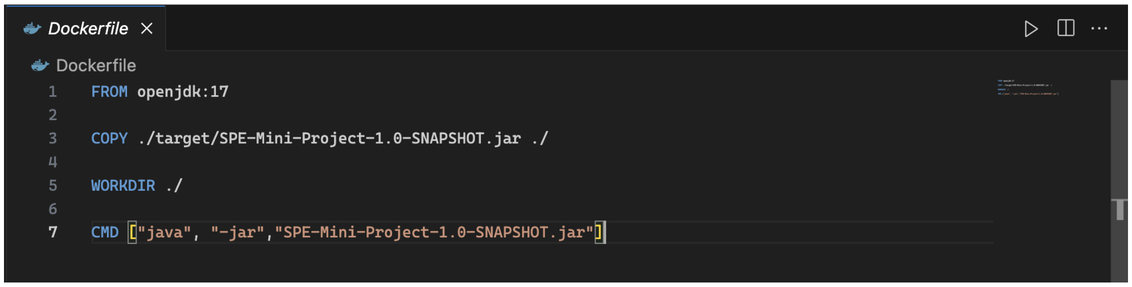
2. **Docker Image**: A read-only template that contains all the components needed to run an application, including the code, runtime, libraries, and environment variables. Images are stored in repositories such as Docker Hub.

3. **Docker Container**: A running instance of a Docker image. Containers are lightweight and portable, providing an isolated environment where the application runs.

4. **Dockerhub**: A cloud-based registry where Docker images are stored and shared. It acts as a repository where developers can pull and push images.

8.3 Using Docker in this Project

The Dockerfile for this project was as shown in the fig 7. Here is the explanation for the dockerfile:



```
Dockerfile
FROM openjdk:17

COPY ./target/SPE-Mini-Project-1.0-SNAPSHOT.jar ./

WORKDIR ./

CMD ["java", "-jar","SPE-Mini-Project-1.0-SNAPSHOT.jar"]
```

Figure 7: Dockerfile for the Calculator

1. **FROM openjdk:17**: This line specifies the base image for the container. In this case, the base image is openjdk:17, which provides the Java Runtime Environment (JRE) version 17. This ensures that the container will have the necessary environment to run a Java application.
2. **COPY ./target/SPE-Mini-Project-1.0-SNAPSHOT.jar ./**: This command copies the JAR file (SPE-Mini-Project-1.0-SNAPSHOT.jar) from the local target directory into the root directory of the container. The JAR file is the compiled output of the Maven build process, which contains the application and its dependencies.
3. **WORKDIR ./**: This sets the working directory inside the container to the root directory (./). Any commands executed after this line will be run relative to this directory.
4. **CMD [”java”, ”-jar”, ”SPE-Mini-Project-1.0-SNAPSHOT.jar”]**: This defines the default command that will be executed when the container starts. The java -jar SPE-Mini-Project-1.0-SNAPSHOT.jar command tells the container to execute the JAR file using Java. This starts the application inside the container.

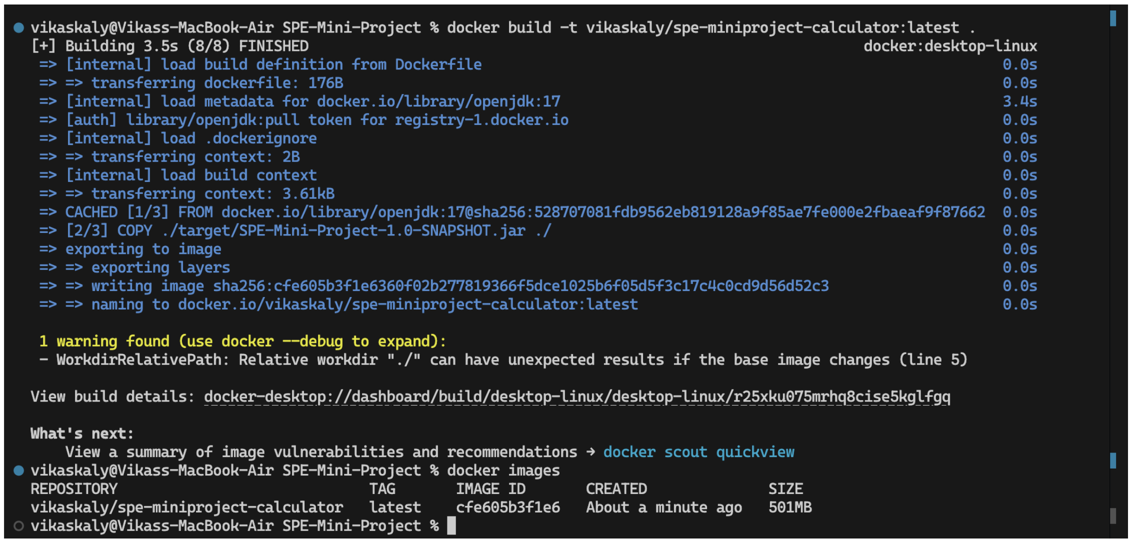
8.4 Building the image and running the container

1. We can build the image and list images using:

```
docker build -t vikaskaly/spe-miniproject-calculator:latest .

docker images
```

The first command builds a Docker image from the Dockerfile in the current directory (.). The -t flag tags the image with a specific name, in this case vikaskaly/spe-miniproject-calculator, and gives it the latest tag. The tag helps identify different versions of the image. After the build is complete, the image will be stored locally. The second command lists all the Docker images available in the local Docker repository. It shows details such as the image ID, repository name, tag, size, and creation date, helping you manage the images on your system. The output of these commands can be seen in fig 8

2. We can run the container from the image and look at the local containers using:



```
● vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % docker build -t vikaskaly/spe-miniproject-calculator:latest .
  [+] Building 3.5s (8/8) FINISHED                                                             docker:desktop-linux
   => [internal] load build definition from Dockerfile                                                         0.0s
   => => transferring dockerfile: 176B                                                                         0.0s
   => [internal] load metadata for docker.io/library/openjdk:17                                                3.4s
   => [auth] library/openjdk:pull token for registry-1.docker.io                                               0.0s
   => [internal] load .dockerignore                                                                            0.0s
   => => transferring context: 2B                                                                              0.0s
   => [internal] load build context                                                                            0.0s
   => => transferring context: 3.61kB                                                                          0.0s
   => CACHED [1/3] FROM docker.io/library/openjdk:17@sha256:528707081fdb9562eb819128a9f85ae7fe000e2fbaeaf9f87662  0.0s
   => [2/3] COPY ./target/SPE-Mini-Project-1.0-SNAPSHOT.jar ./                                                 0.0s
   => exporting to image                                                                                       0.0s
   => => exporting layers                                                                                      0.0s
   => => writing image sha256:cfe605b3f1e6360f02b277819366f5dce1025b6f05d5f3c17c4c0cd9d56d52c3                 0.0s
   => => naming to docker.io/vikaskaly/spe-miniproject-calculator:latest                                       0.0s

   1 warning found (use docker --debug to expand):
   - WorkdirRelativePath: Relative workdir "./" can have unexpected results if the base image changes (line 5)

  View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/r25xku075mrhq8cise5kglfgq

  What's next:
      View a summary of image vulnerabilities and recommendations → docker scout quickview
● vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % docker images
  REPOSITORY                              TAG       IMAGE ID       CREATED              SIZE
  vikaskaly/spe-miniproject-calculator   latest    cfe605b3f1e6   About a minute ago   501MB
○ vikaskaly@Vikass-MacBook-Air SPE-Mini-Project %
```

Figure 8: Building Image

```
docker run -it --name Calculator vikaskaly/spe-miniproject
    -calculator

docker ps -a
```

The first command is used to create and run the container in interactive mode (interact with it directly via the terminal) from the specified Docker image (vikaskaly/spe-miniproject-calculator). Here's a breakdown of the options used:

(a) **docker run**: This command creates and starts a new container from a specified image.

(b) **-it**: This flag combines two options:

  i. **-i**: Stands for interactive mode, which allows you to interact with the container's standard input (stdin).

  ii. **-t**: Allocates a pseudo-TTY (terminal) for the container, which makes it possible to run interactive applications.

(c) **–name Calculator**: This option assigns the name "Calculator" to the container, making it easier to reference later.

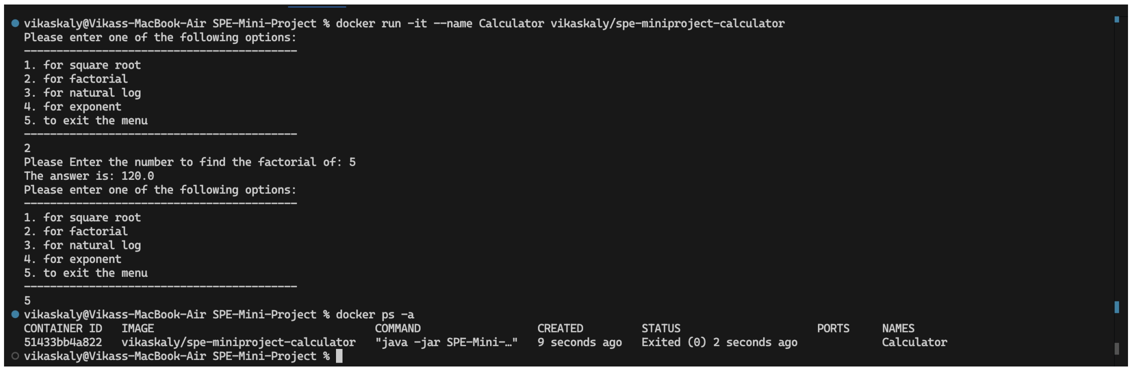
(d) **vikaskaly/spe-miniproject-calculator**: This specifies the image from which to create the container.

The second command lists all Docker containers on the system, both running and stopped. Here's what it does:

(a) **docker ps**: Displays the currently running containers by default.

(b) **-a**: The -a (or –all) flag extends the command to include all containers, regardless of their state (running, exited, or stopped).

The output of these commands can be seen in fig 9



```
vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % docker run -it --name Calculator vikaskaly/spe-miniproject-calculator
Please enter one of the following options:
-----------------------------------------
1. for square root
2. for factorial
3. for natural log
4. for exponent
5. to exit the menu
-----------------------------------------
2
Please Enter the number to find the factorial of: 5
The answer is: 120.0
Please enter one of the following options:
-----------------------------------------
1. for square root
2. for factorial
3. for natural log
4. for exponent
5. to exit the menu
-----------------------------------------
5
vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % docker ps -a
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS                      PORTS     NAMES
51433bb4a822   vikaskaly/spe-miniproject-calculator   "java -jar SPE-Mini-…"   9 seconds ago    Exited (0) 2 seconds ago              Calculator
vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % 
```

Figure 9: Running the container

8.5 Pushing Docker Container to dockerhub

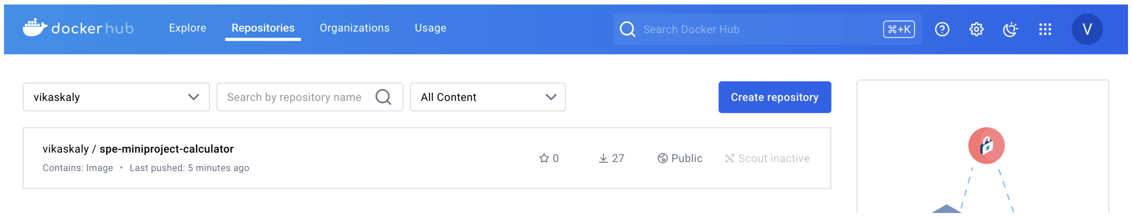
Dockerhub is a remote docker container repository where we can push our images to.

```
docker push <image-name>
```

The above command can be run to push a built image to Dockerhub. Note that you need to be logged in to docker locally using your credentials to publish the image.

```
docker pull <image-name>
```

The above command can be run to pull an image located in dockerhub and store it locally on our machine. Refer to fig 10 for how dockerhub looks after pushing an image to dockerhub.



vikaskaly / spe-miniproject-calculator
Contains: Image • Last pushed: 5 minutes ago
☆ 0 ⤓ 27 Public Scout inactive

Figure 10: Dockerhub

9 Ansible

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation. It enables users to automate the provisioning and management of systems in a simple and efficient manner. It can automate the entire deployment pipeline, ensuring that applications are set up correctly and quickly.

9.1 Key Concepts of Ansible

1. **Controller Node**: The controller node is the machine where Ansible is installed and run. It is responsible for executing automation tasks and managing configurations.

From the controller node, users can deploy updates, configure systems, and orchestrate tasks across multiple managed hosts.

2. **Managed Hosts**: Managed hosts (or nodes) are the target machines that Ansible manages. These can be physical servers, virtual machines, or cloud instances. Ansible communicates with the managed hosts over SSH (for Linux) or WinRM (for Windows) without needing to install any agent software on them, allowing for straightforward management.

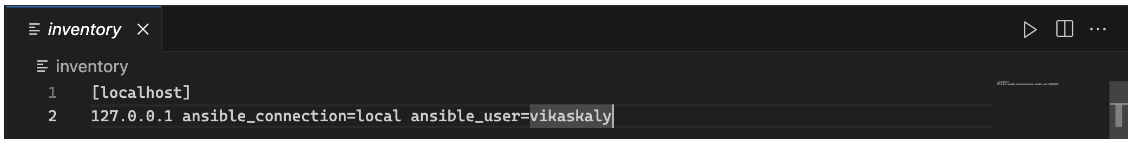
3. **Inventories**: An inventory is a file or a script that defines the managed hosts and their properties. It specifies the IP addresses or hostnames of the systems that Ansible will manage. This allows for flexible organization of hosts into groups for easier management and targeting.

4. **Playbooks**: Playbooks are YAML files that define a series of tasks to be executed on the managed hosts. They allow users to describe the desired state of the systems and the steps required to achieve that state. Playbooks can include various tasks such as installing packages, configuring services, and copying files. They also support conditional execution and looping, making them powerful tools for automation.

9.2 Ansible configurations in this project

9.2.1 Inventory

The screenshot of my inventory file can be found in fig 11.



```
[localhost]
127.0.0.1 ansible_connection=local ansible_user=vikaskaly
```

Figure 11: Ansible inventory

1. **[localhost]**: This line defines a group called localhost. The name localhost indicates that this group contains the local machine where Ansible is being executed.

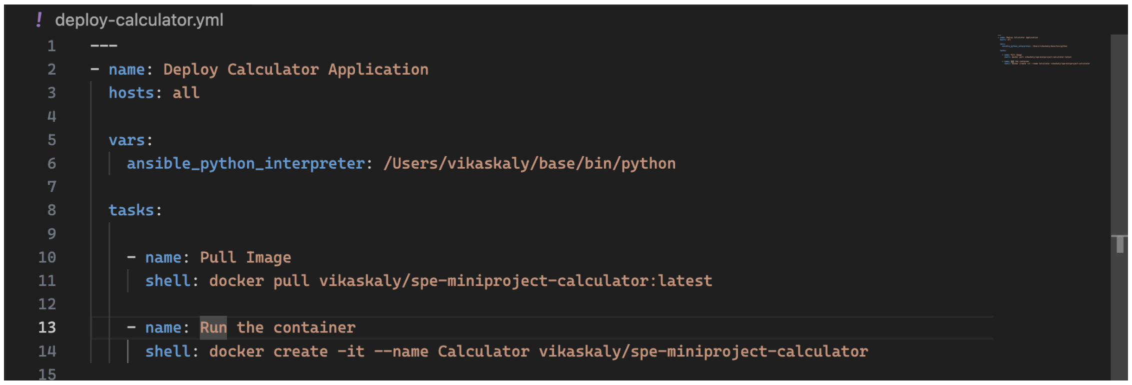
2. **127.0.0.1**: This is the IP address of the local machine, commonly referred to as "localhost." It points to the local system on which the Ansible playbook is being run. Ansible will use this address to perform tasks on the local machine.

3. **ansible_connection=local**: This variable specifies the connection type Ansible should use for this host. In this case, local means that Ansible will execute tasks directly on the local machine without needing SSH or any remote connection. This is particularly useful for running playbooks or tasks that manage the local system.

4. **ansible_user=vikaskaly**: This variable defines the username that Ansible will use when connecting to the host. In this example, it specifies the user vikaskaly. This is important for permissions, especially if certain tasks require elevated privileges or specific user permissions.

9.2.2 Playbook

The screenshot of my inventory file can be found in fig 12.



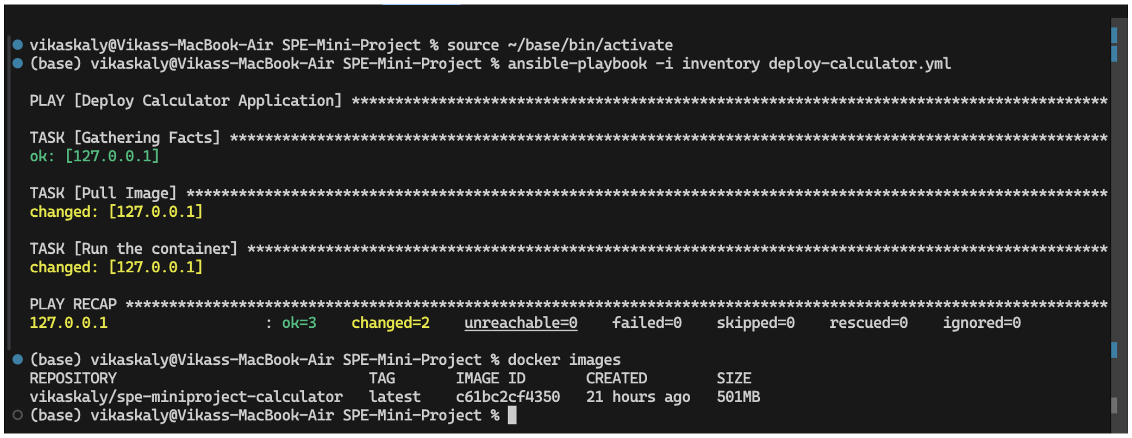
```yaml
deploy-calculator.yml
---
- name: Deploy Calculator Application
  hosts: all

  vars:
    ansible_python_interpreter: /Users/vikaskaly/base/bin/python

  tasks:

    - name: Pull Image
      shell: docker pull vikaskaly/spe-miniproject-calculator:latest

    - name: Run the container
      shell: docker create -it --name Calculator vikaskaly/spe-miniproject-calculator
```

Figure 12: Ansible playbook

1. **- name: Deploy Calculator Application**: This is a descriptive name for the playbook, providing context about what the playbook will accomplish.
2. **hosts: all**: This specifies that the playbook will be run on all hosts defined in the Ansible inventory.
3. **vars:** This section defines variables that can be used throughout the playbook. In this case, it specifies the Python interpreter to be used by Ansible.
   (a) **ansible_python_interpreter: /Users/vikaskaly/base/bin/python**: This sets the path to the Python interpreter that Ansible should use when executing tasks, particularly important if the system has multiple Python installations (venv's).
4. **Tasks**: The playbook contains a list of tasks that Ansible will execute on the specified hosts:
   (a) **shell: docker pull vikaskaly/spe-miniproject-calculator:latest**: This uses the shell module to execute a shell command on the target host. The command retrieves the latest version of the specified Docker image from the Docker registry. This ensures that the most up-to-date version of the application is available on the host.
   (b) **shell: docker create -it –name Calculator vikaskaly/spe-miniproject-calculator**: This command creates a new Docker container (interactive mode) from the specified image and names the container as *Calculator*.

9.2.3 Running the ansible playbook

Please refer to fig 13 to see the output on running the ansible playbook.



```
● vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % source ~/base/bin/activate
● (base) vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % ansible-playbook -i inventory deploy-calculator.yml

  PLAY [Deploy Calculator Application] *******************************************************************************

  TASK [Gathering Facts] *********************************************************************************************
  ok: [127.0.0.1]

  TASK [Pull Image] **************************************************************************************************
  changed: [127.0.0.1]

  TASK [Run the container] *******************************************************************************************
  changed: [127.0.0.1]

  PLAY RECAP *********************************************************************************************************
  127.0.0.1                  : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

● (base) vikaskaly@Vikass-MacBook-Air SPE-Mini-Project % docker images
  REPOSITORY                            TAG       IMAGE ID       CREATED        SIZE
  vikaskaly/spe-miniproject-calculator  latest    c61bc2cf4350   21 hours ago   501MB
○ (base) vikaskaly@Vikass-MacBook-Air SPE-Mini-Project %
```

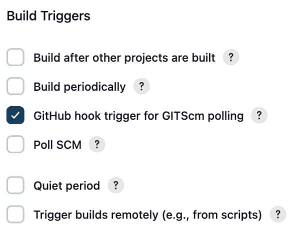
Figure 13: Running the Ansible playbook

10 Jenkins

Jenkins is an open-source automation server that facilitates Continuous Integration (CI) and Continuous Delivery (CD) in software development. It allows developers to automate the building, testing, and deployment of applications, ensuring that software changes can be integrated frequently and delivered rapidly and reliably.
For a pipeline style project, a **Jenkins Pipeline** defines a set of stages and steps that are executed sequentially or in parallel to build, test, and deploy code. A Jenkinsfile defines the pipeline steps and is stored in the root of your project's repository.

10.1 Jenkins project Configuration

Go to localhost:8080 to access jenkins and install all necessary plugins (GitHub, docker and ansible). Create a new pipeline project and refer to fig 14, 15 for the project configuration.



Build Triggers

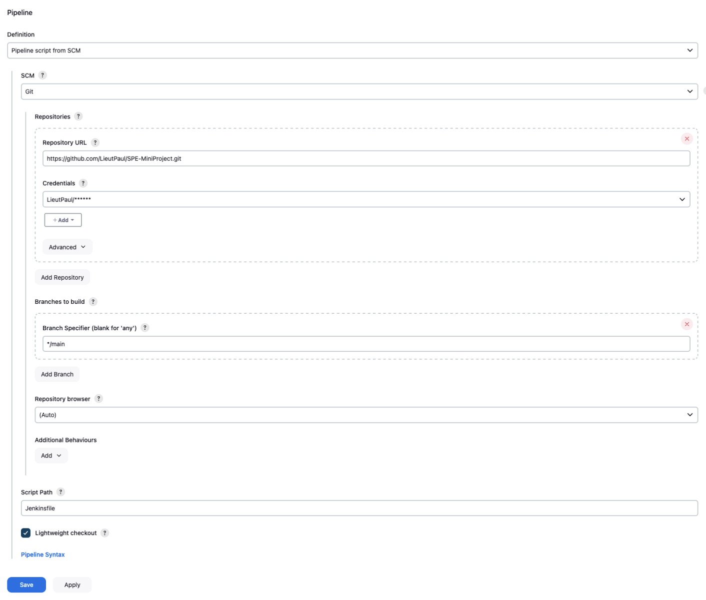
- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [x] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Quiet period ?
- [ ] Trigger builds remotely (e.g., from scripts) ?

Figure 14: Jenkins Project Config 1

Also, add your GitHub and Dockerhub credentials to jenkins. Refer to fig 16

10.2 Ngrok and GitHub Webhook

Ngrok allows us to expose our localhost:8080 as a public url that allows GitHub to send triggers to jenkins. To do this run:


Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/LieutPaul/SPE-MiniProject.git

Credentials: LieutPaul/******

Branches to build

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

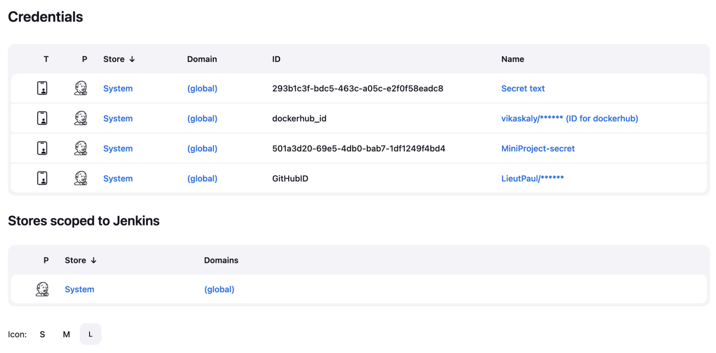
Repository browser: (Auto)

Additional Behaviours

Script Path: Jenkinsfile

Lightweight checkout: ✓

Figure 15: Jenkins Project Config 2


Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	293b1c3f-bdc5-463c-a05c-e2f0f58eadc8	Secret text
		System	(global)	dockerhub_id	vikaskaly/****** (ID for dockerhub)
		System	(global)	501a3d20-69e5-4db0-bab7-1df1249f4bd4	MiniProject-secret
		System	(global)	GitHubID	LieutPaul/******

Stores scoped to Jenkins

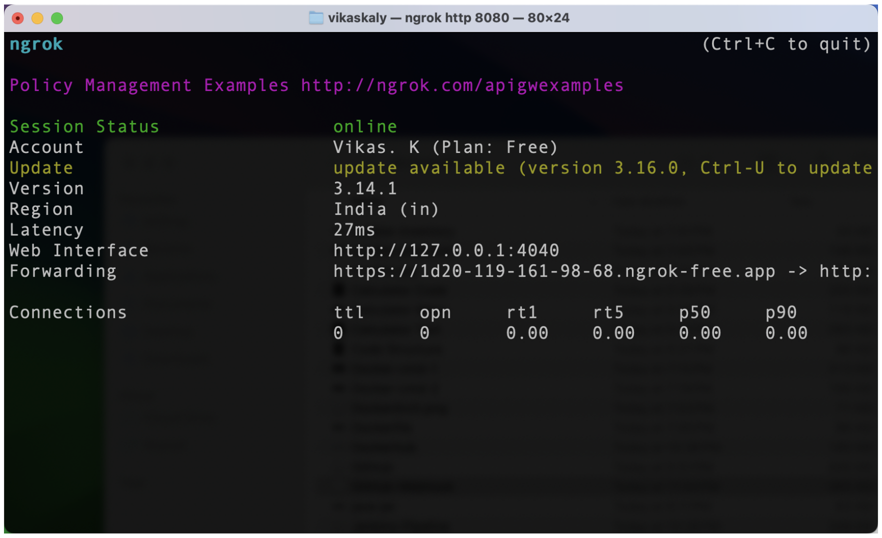
P	Store ↓	Domains
	System	(global)

Figure 16: Jenkins Credentials

```
ngrok http 8080
```

Refer to fig 17 to look at the output of this command.

Next, in the settings for the GitHub repo, add the ngrok url into the webhook followed by */gitub-webhook/* . Refer to fig 18 for this. **Now, whenever we push to GitHub, the change is detected and the script will be automatically run.**



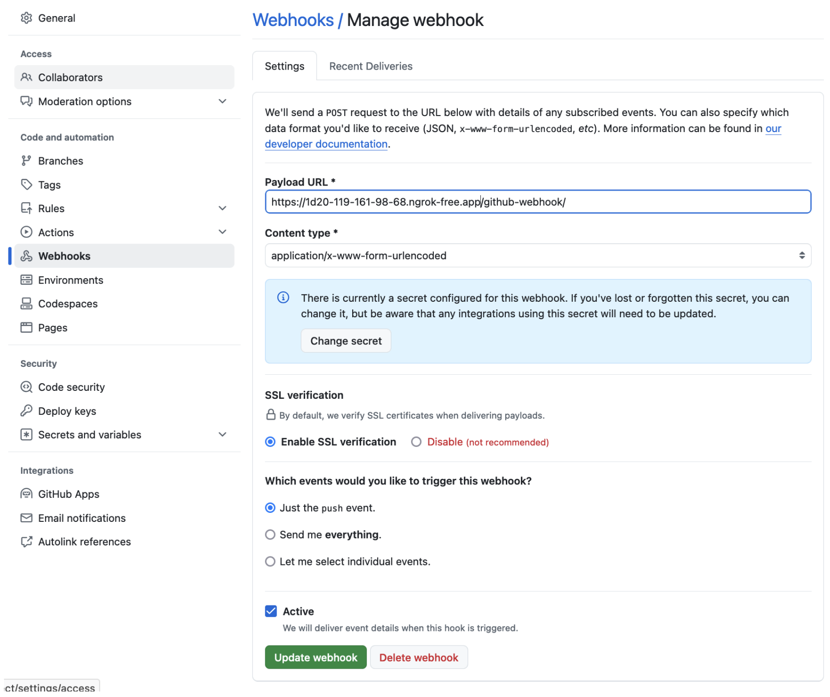
```
ngrok                                                            (Ctrl+C to quit)

Policy Management Examples http://ngrok.com/apigwexamples

Session Status                online
Account                       Vikas. K (Plan: Free)
Update                        update available (version 3.16.0, Ctrl-U to update
Version                       3.14.1
Region                        India (in)
Latency                       27ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://1d20-119-161-98-68.ngrok-free.app -> http:

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Figure 17: Ngrok

10.3 Jenkinsfile for my project

The following is the code for the Jenkinsfile:

```
pipeline{
    environment{
        DOCKERHUB_CRED = credentials("dockerhub_id")
        PATH = "/opt/homebrew/bin:$PATH"
    }
    agent any
    stages{

        stage("Stage 1: Maven Build and Test"){
            steps{
                sh 'mvn clean install'
            }
        }

        stage("Stage 2: Build Docker Image"){
            steps{
                sh "docker build -t vikaskaly/spe-miniproject-
                   calculator:latest ."
            }
        }

        stage("Stage 3: Push Docker Image to Dockerhub"){
            steps{
                sh 'echo $DOCKERHUB_CRED_PSW | docker login -u
                  $DOCKERHUB_CRED_USR --password-stdin'
```



Webhooks / Manage webhook

Settings | Recent Deliveries

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://1d20-119-161-98-68.ngrok-free.app/github-webhook/

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

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

Update webhook | Delete webhook

Figure 18: GitHub Webhooks

```
                sh "docker push vikaskaly/spe-miniproject-
                    calculator:latest"
            }
        }

        stage("Stage 4: Clean Unwanted Docker Images"){
            steps{
                sh "docker container prune -f"
                sh "docker image prune -a -f"
            }
        }

        stage('Stage 5: Ansible Deployment') {
            steps {
                sh "ansible-playbook -i inventory deploy-
                    calculator.yml"
            }
        }


    }
}
```

Its various components include:

1. **Environment**:

   (a) **DOCKERHUB_CRED = credentials("dockerhub_id")**: Fetches the DockerHub credentials (username and password) stored in Jenkins under the ID dockerhub_id. These are used for authenticating DockerHub.

   (b) **PATH = "/opt/homebrew/bin:$PATH"**: Adds /opt/homebrew/bin to the PATH environment variable. This is because mvn is located in this directory.

2. **Stage 1: Maven Buld and Test**: Runs the command *mvn clean install* to compile, test, and package the Maven project. This step ensures the code passes all tests before proceeding and then builds the jar file.

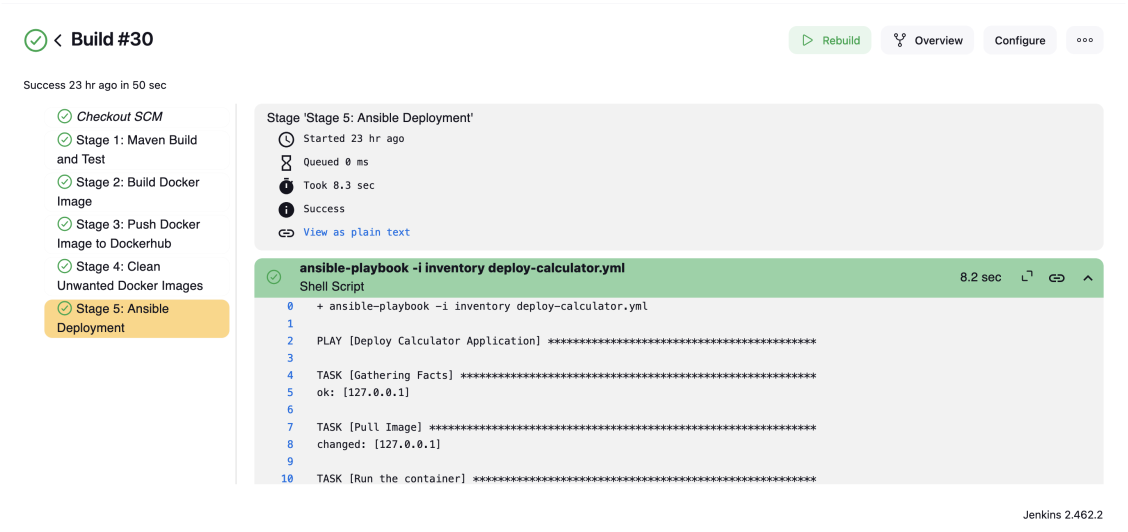
3. **Stage 2: Build Docker Image**: Builds a Docker image using the Dockerfile located in the project's root directory. The image is tagged as vikaskaly/spe-miniproject-calculator:latest.

4. **Stage 3: Push Docker image to Dockerhub**: Logs in to DockerHub using the credentials and pushes the built Docker image (vikaskaly/spe-miniproject-calculator:latest) to DockerHub.

5. **Stage 4: Clean unwanted docker Images**: Cleans up unnecessary Docker resources (images and containers) to free up space.

6. **Stage 5: Ansible Deployment**: Runs the ansible playbook with the inventory to deploy the application.

An image of the jenkins pipeline can be found in fig 19.



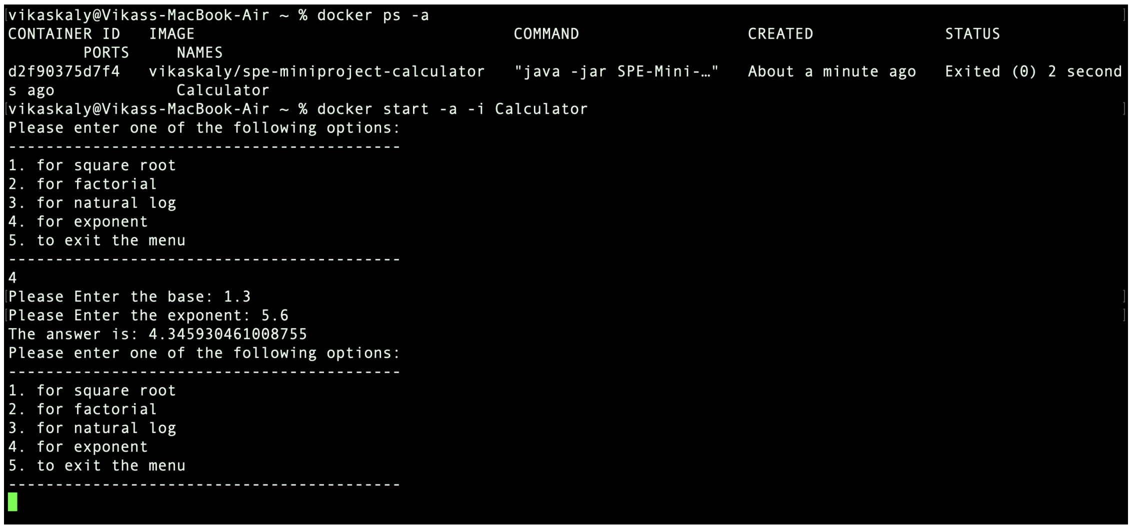
Build #30
Rebuild
Overview
Configure
Success 23 hr ago in 50 sec
Checkout SCM
Stage 1: Maven Build and Test
Stage 2: Build Docker Image
Stage 3: Push Docker Image to Dockerhub
Stage 4: Clean Unwanted Docker Images
Stage 5: Ansible Deployment
Stage 'Stage 5: Ansible Deployment'
Started 23 hr ago
Queued 0 ms
Took 8.3 sec
Success
View as plain text
ansible-playbook -i inventory deploy-calculator.yml
Shell Script
8.2 sec
Jenkins 2.462.2

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

PLAY [Deploy Calculator Application] ******************************************

TASK [Gathering Facts] *********************************************************
ok: [127.0.0.1]

TASK [Pull Image] **************************************************************
changed: [127.0.0.1]

TASK [Run the container] *******************************************************
```

Figure 19: Jenkins Pipeline

11 Running the container

After the jenkins pipeline finishes executing, the container that is pulled from dockerhub is on my machine. This container can be run locally. Refer to fig 20 for the output.



```
vikaskaly@Vikass-MacBook-Air ~ % docker ps -a
CONTAINER ID   IMAGE                                 COMMAND                  CREATED              STATUS
        PORTS     NAMES
d2f90375d7f4   vikaskaly/spe-miniproject-calculator   "java -jar SPE-Mini-…"   About a minute ago   Exited (0) 2 second
s ago            Calculator
vikaskaly@Vikass-MacBook-Air ~ % docker start -a -i Calculator
Please enter one of the following options:
------------------------------------------
1. for square root
2. for factorial
3. for natural log
4. for exponent
5. to exit the menu
------------------------------------------
4
Please Enter the base: 1.3
Please Enter the exponent: 5.6
The answer is: 4.345930461008755
Please enter one of the following options:
------------------------------------------
1. for square root
2. for factorial
3. for natural log
4. for exponent
5. to exit the menu
------------------------------------------
```

Figure 20: Running the docker container