CS 816 - Software Production Engineering

Mini Project - Scientific Calculator with DevOps

Submitted By : Sreenidhi KR : MT2022115
Github : https://github.com/Sreenidhi-KR/SpeMiniProject
DockerHub : sreenidhikr/calculator

Problem Statement

Create a scientific calculator program with user menu driven operations

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

Overview of DevOps

DevOps is an approach to software development that emphasizes collaboration and communication between software developers and IT operations teams. The goal of DevOps is to make it easier and faster to deliver software applications and services to users.

In traditional software development, developers write code and then hand it off to the operations team to deploy and manage. But in DevOps, developers and operations work together from the beginning of the software development process to create software that is easier to deploy and manage.

To achieve this, DevOps uses a variety of tools and techniques, such as automation, continuous integration and delivery, and infrastructure as code. By automating many of the steps involved in software development and deployment, DevOps enables organizations to deliver software more quickly, reliably, and with fewer errors.

Overall, DevOps is all about making software development and deployment more efficient and effective, so organizations can create better software and deliver it to users more quickly.

Tools Used

- Git: A popular version control system used for tracking changes in source code and collaborating with other developers on software projects.

- Jenkins: An open-source automation server that provides a platform for continuous integration and continuous delivery/deployment (CI/CD) of software applications, helping to streamline the software development process.

- Docker: Docker is a containerization platform that allows developers to package and deploy applications in a portable, efficient, and secure manner

- Ansible: An open-source automation tool used for configuration management, application deployment, and infrastructure orchestration, helping to automate IT processes and make them more efficient.

- Maven: A build automation tool used primarily for Java projects to manage project dependencies, build processes, and deployment.

- JUnit: Testing framework for Java

- Ngrok : To expose our private IP to the Internet

- ELK: Elasticsearch, Logstash, and Kibana, which are three open-source tools used together for centralized logging and data analysis, providing real-time insights into system logs and performance metrics.

Overview of the development process

- Install and setup Docker , Jenkins , Ansible , Java , ..etc in your system

- Write the first module along with Test cases of the Calculator app and push it Github

- Write pipeline scripts in Jenkins
    a. Git Pull
    b. Maven build
    c. Docker Image creation
    d. Pushing Image to Docker Hub
    e. Ansible Deploy

- Monitor Logs using ELK

Installing Required Software

- Here I will be focusing on running the project on a **MAC M1 (ARM)** system (Server).
- Client nodes will be replicated by using a Ubuntu VM (Multipass)
- All commands provided will will only be applicable of MacOS

Jenkins

Installing Jenkins

1. First, ensure that you have Java 8 or later installed on your Mac. You can check by opening the Terminal app and entering the command "java -version". If Java is not installed, you can download and install it from the official Java website.

2. Use the command *brew install jenkins* to install Jenkins.

3. After the installation is complete, open a web browser and go to http://localhost:8080/ to access the Jenkins web interface.

4. When you first access Jenkins, you will be prompted to enter an initial admin password. The password can be found in the file /Users/<username>/.jenkins/secrets/initialAdminPassword. Copy the password and paste it into the Jenkins web interface.

5. After entering the initial admin password, Jenkins will prompt you to install recommended plugins. You can choose to install all of the recommended plugins or select only the ones you need.

Docker

Installing docker

1. Install rosetta - 2 : recommended by docker
   softwareupdate --install-rosetta

2. Download Docker from https://desktop.docker.com/mac/main/arm64/Docker.dmg?utm_source=docker&utm_medium=webreferral&utm_campaign=docs-driven-download-mac-arm64

3. Install interactively
   Double-click Docker.dmg to open the installer, then drag the Docker icon to the Applications folder.
   Double-click Docker.app in the Applications folder to start Docker.
   Select Accept to continue. Docker Desktop starts after you accept the terms.

Ansible

1. Install using homebrew : *brew install ansible*
2. Verify Version : *ansible -- version*

Setting Up Client (Ubuntu VM using Multipass)

- Install Multipass

  Download and install Multipass from the official website: https://multipass.run/
  Once installed, run the following command to launch a new instance:
  multipass launch --name testvm

- Generate SSH Key pair

  Run the following command to generate a new SSH key pair:
  ssh-keygen -t rsa
  Press Enter to accept the default location (*~/.ssh*) for the key file.
  When prompted, enter a passphrase for the key. You can leave this blank if you don't want to use a passphrase.
  Once the key pair has been generated, you can view the public key by running the following command:
  cat ~/.ssh/id_rsa.pub

  DO NOT MODIFY ANYTHING IN THE KEY's EVEN ADDING OR REMOVING A EMPTY LINE WILL CAUSE KEY TO CORRUPT

- Run the following command to get the IP address of your Multipass instance:
  multipass list

- Configure SSH access to Multipass instances

  Run the following command to add your public SSH key to the VM instance:

  multipass exec <instance-name> -- sudo sh -c "mkdir -p /root/.ssh && chmod 700 /root/.ssh && echo '<your-public-ssh-key>' >> /root/.ssh/authorized_keys && chmod 600 /root/.ssh/authorized_keys"

  Replace <instance-name> with the name of your Multipass instance and <your-public-ssh-key> with the content of your public SSH key. You can find your public SSH key by running the following command on your Mac: cat ~/.ssh/id_rsa.pub

  The above command is similar to this *: ssh-copy-id ubuntu@196.168.9.0*

- Configure Ansible

  Create a new directory on your Mac to store the Ansible configuration files:
  mkdir ansible && cd ansible

  Create a new file called hosts and add the following content:

  [multipass]
  <instance-ip-address>

  Replace <instance-ip-address> with the IP address of your Multipass instance

  Create a new file called ansible.cfg and add the following content:

  [defaults]
  host_key_checking = False
  remote_user = root

  This configuration file tells Ansible to disable host key checking and use the root user to connect to the remote host

- Test the Ansible configuration

  Run the following command to test the Ansible configuration:
  ansible -i ansible/hosts multipass -m ping --user root

  If everything is set up correctly, you should see the following output:
  Javascript

  ```
  <instance-ip-address> | SUCCESS => {
      "changed": false,
      "ping": "pong"
  }
  ```

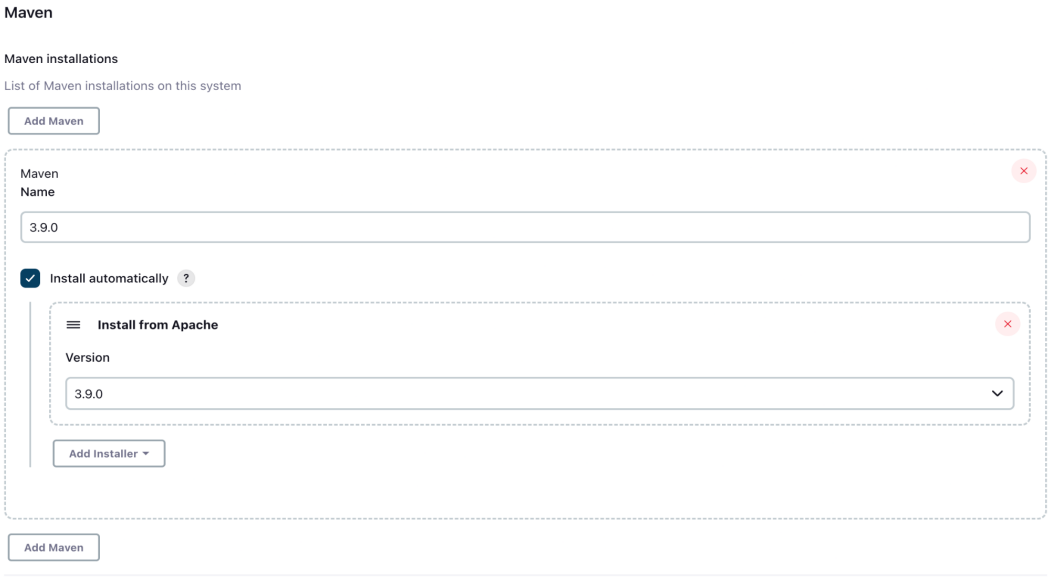
  This means that Ansible was able to connect to your Multipass instance using SSH

- Install Docker on the Client VM :
  https://docs.docker.com/engine/install/ubuntu/
  To run docker without root privileges https://docs.docker.com/engine/install/linux-postinstall/

Installing Required Plugins on Jenkins

1. Pipeline Maven Integration Plugin

   After Installing the plugin go to Global tools Configuration and provide the executable path of maven in your local system , or let it install automatically while running, here i have set to up to install automatically

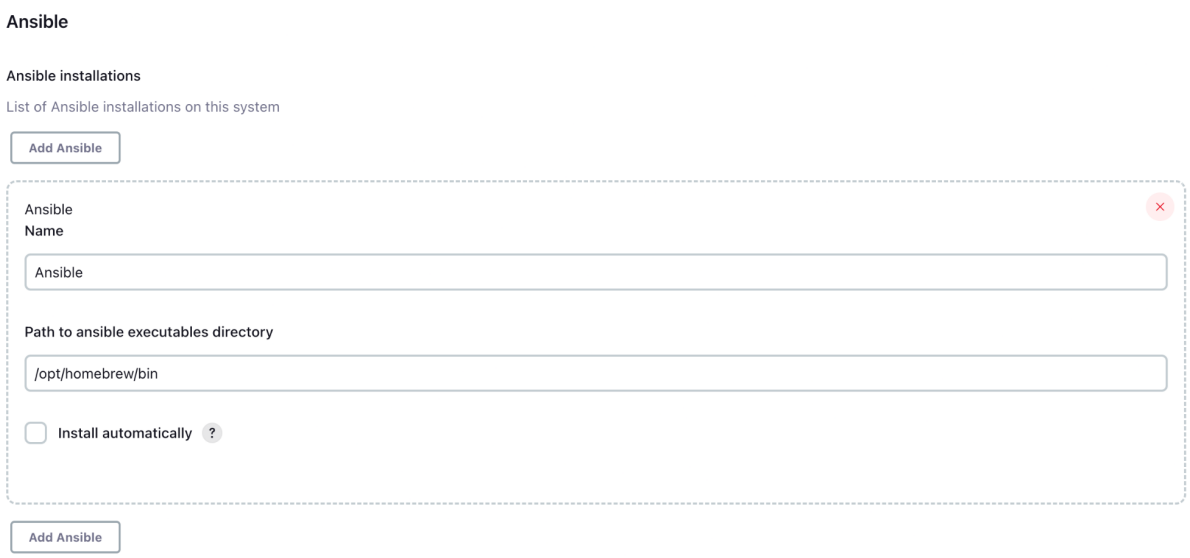


2. Docker Pipeline

   After Installation create an account in docker hub and add the credentials in Jenkins

3. Ansible

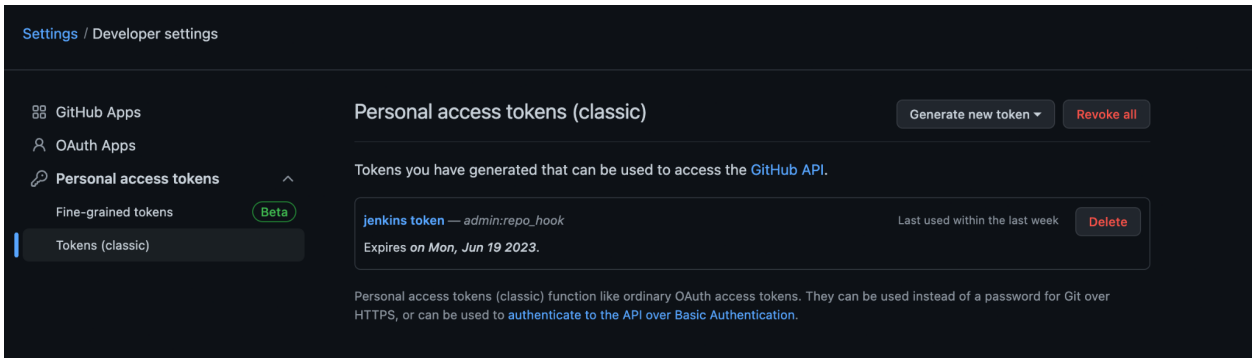
   After installing the plugin Add path to the ansible binaries which are present in your system



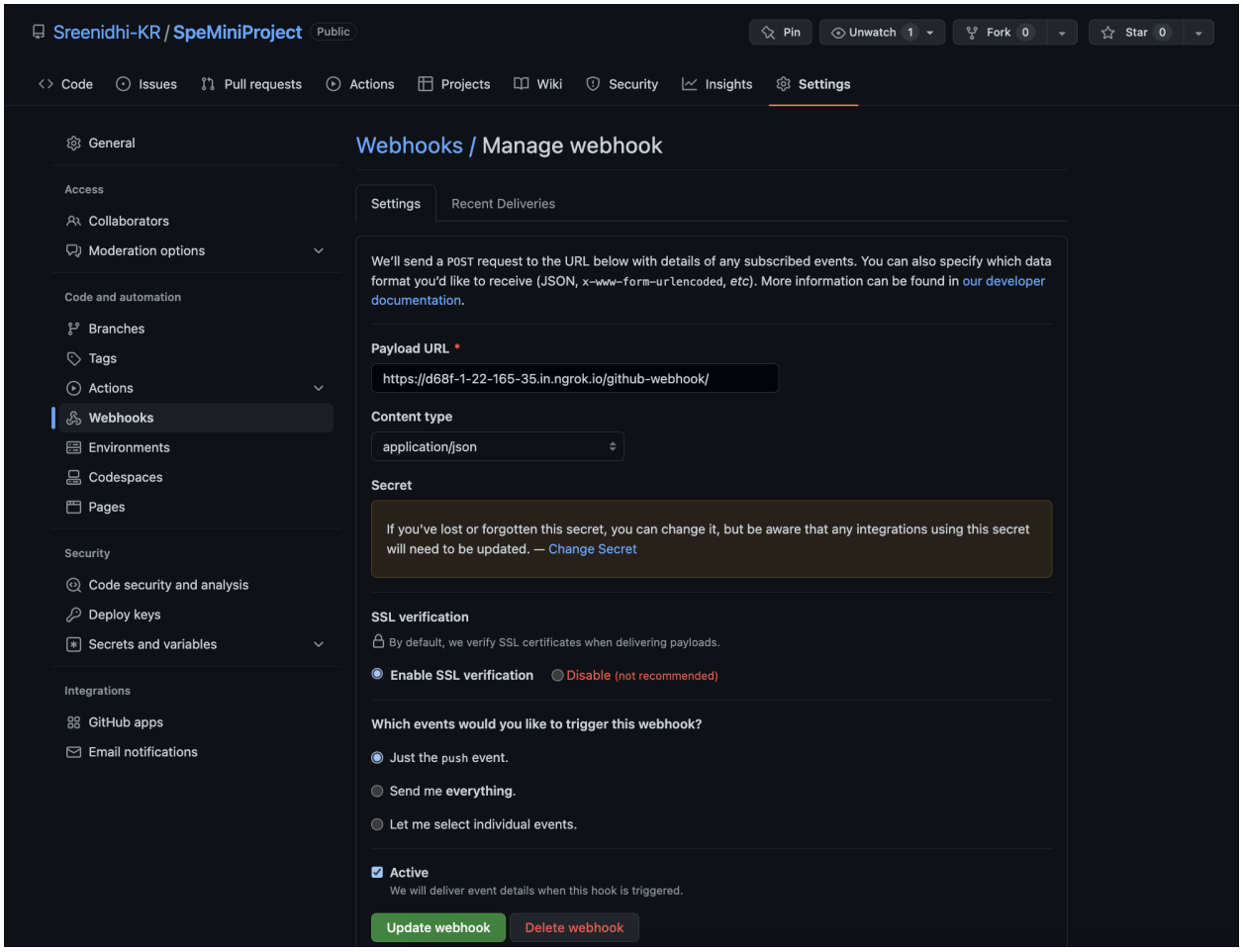
4. Github

   Install the github plugin if not already installed , if you want to integrate github webhooks into the project copy paste the secret key into jenkins credentials . If your github repository is private you need to add credentials of your github account into jenkins as well.
   Setup ngrok on port 8080 (where jenkins is running) using command *ngrok http 8080*

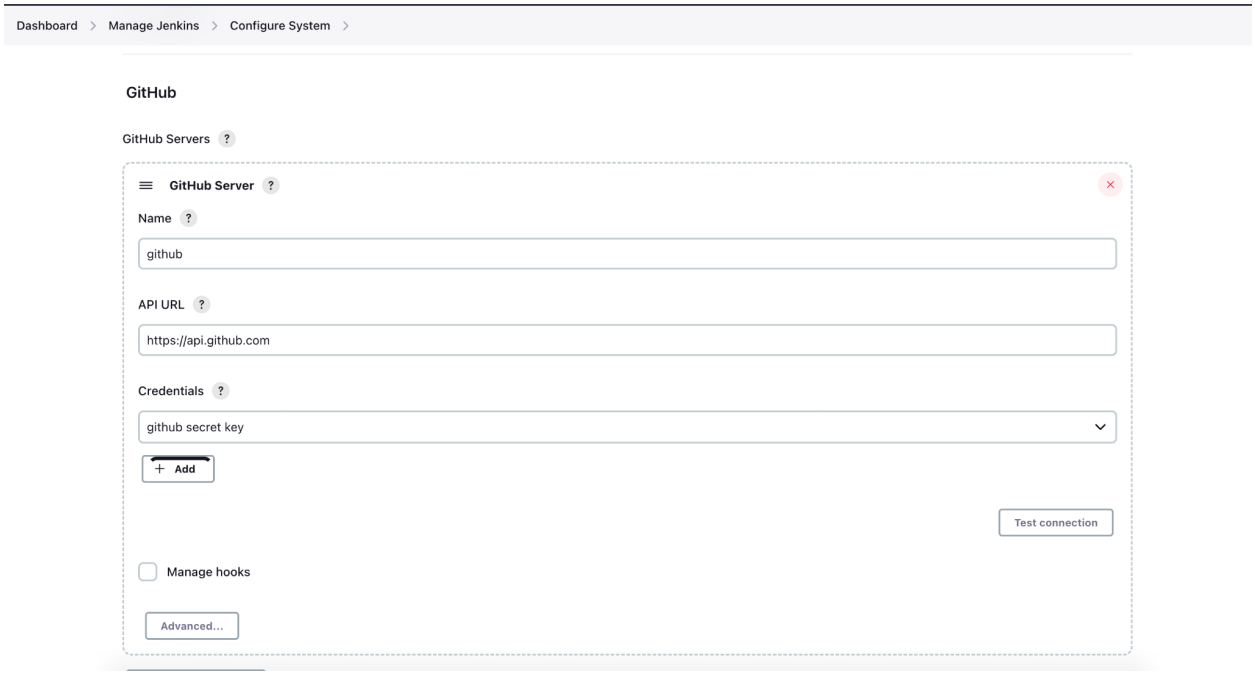
Goto github > settings > developer settings and generate secret text

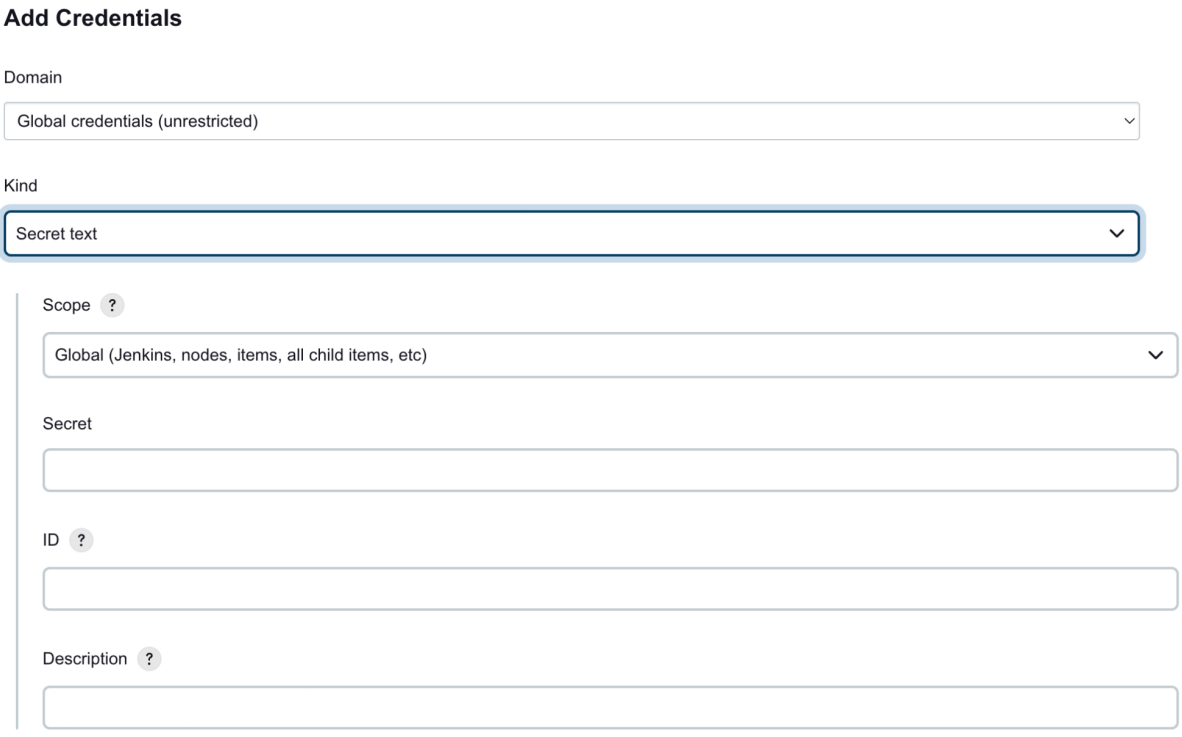


Configure Webhook



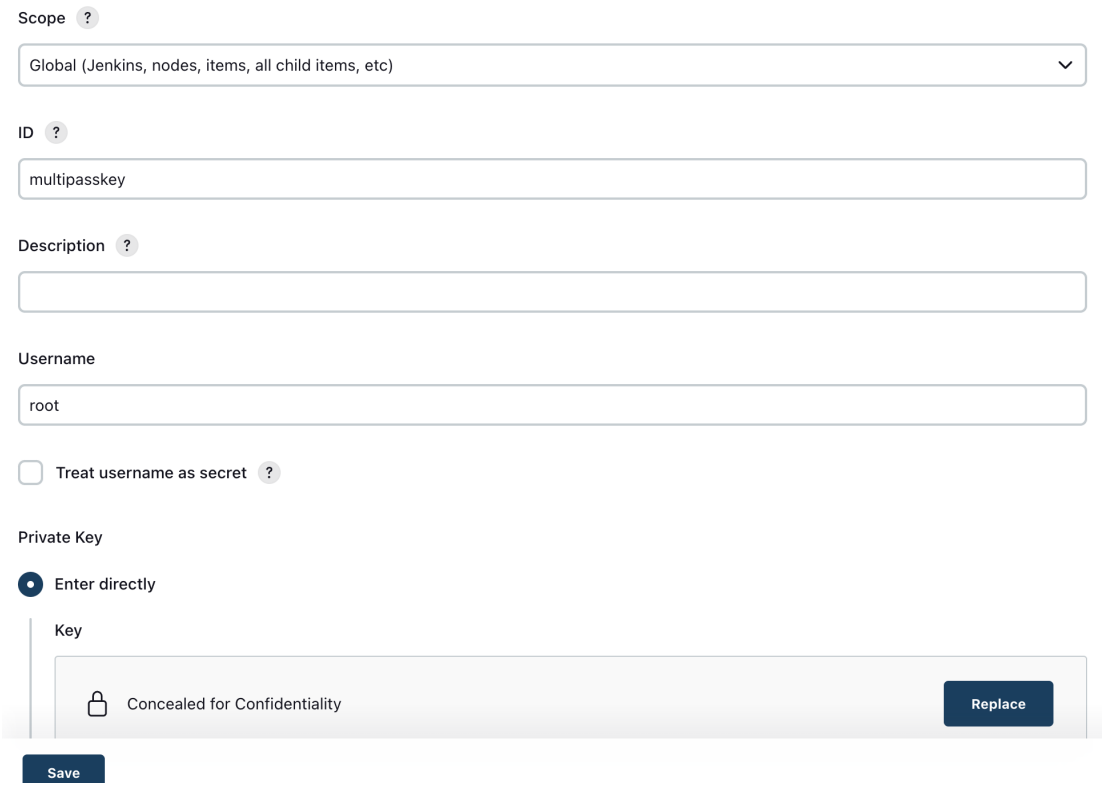
Add Credentials in Jenkins





5. Add SSH Private Key to Jenkins

   Copy the private key generated using ssh keygen command , by default it's placed in /.ssh directory
   cat ~/.ssh/id_rsa



Create a Maven Project in Intellij Idea

Write a basic implementation of 1 module (factorial) along with its test cases , other modules can be pushed in subsequent commits.

```
public class Main {
   public static double factorial(double num) {
       double fact = 1;
       for(int i = 1; i <= num; i++)
       {
           fact *= i;
       }
       return  fact;
   }
   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       double number1, number2;
       do {
           System.out.println("Scientific Calculator using DevOps. \n Choose");
           System.out.print("""
                   1. Factorial
                   2. Square root
                   3. Power
                   4. Natural Logarithm
                   5. Exit
                   Enter your choice:\s""");
           int choice;
           try {
               choice = scanner.nextInt();
           } catch (InputMismatchException error) {
               return;
           }

           switch (choice) {
               case 1 -> {
                   // Factorial
                   System.out.print("Enter a number : ");
                   number1 = scanner.nextDouble();
                               System.out.println("Factorial of " + number1 + " is : " +
factorial(number1));
                   System.out.println("\n");
               }
               default -> {
                   System.out.println("Exiting....");
                   return;
               }
           }
       } while (true);
   }
}
```

JUnit Test case

Add JUnit dependency to your project by adding the below code in pom.xml file

```
<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
</dependency>
```

```
public class MainTest {
   private static final double DELTA = 1e-15;


   @Test
   public void factorialTruePositive(){
       assertEquals("Finding factorial of a number for True Positive", 720, Main.factorial(6), DELTA);
       assertEquals("Finding factorial of a number for True Positive", 6, Main.factorial(3), DELTA);
   }

   @Test
   public void factorialFalsePositive(){
          assertNotEquals("Finding factorial of a number for False Positive", 113, Main.factorial(6),
DELTA);
           assertNotEquals("Finding factorial of a number for False Positive", 10, Main.factorial(6),
DELTA);
   }
}
```

Dockerfile

This file will help build the docker image , add this in the root directory of your project.

```
FROM openjdk:17
COPY ./target/SpeMiniProject-1.0-SNAPSHOT.jar ./
WORKDIR ./
CMD ["java" , "-cp","SpeMiniProject-1.0-SNAPSHOT.jar" , "org.example.Main"]
```

Ansible Inventory

This file contains the list of client node on which you want to deploy the docker image

```
[ubuntu18]
192.168.64.9
192.168.64.10
```

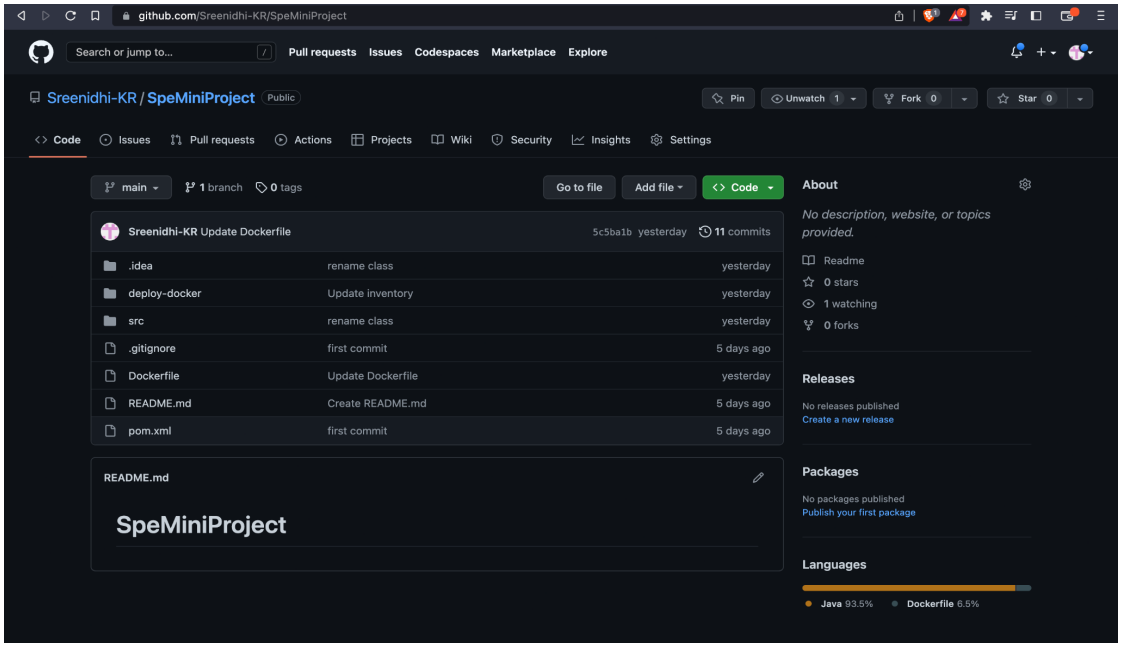
Ansible Playbook

This file contains plays which are set of instructions which has to be executed in the client nodes

```
---
- name: Pull docker image of speminiproject
 hosts: all
 tasks:
   - name: Pull speminiproject devops image
     docker_image:
       name: sreenidhikr/calculator
       force_source: true
       source: pull
```

Creating a Github Repository

Steps:
1. Create a public repository at https://github.com/
2. $ git init.
3. $ git add .
4. $ git remote add origin <github repo URL>.
5. $ git commit -m “Commit message”.
6. $ git push origin master.



Pipeline Script

Create a pipeline project in Jenkins and add the following script

```
pipeline {
   environment{
       imageName = ""
   }
   agent any
   tools {
       maven '3.9.0'
   }
   stages {
       stage('Git Pull') {
           steps {
               git branch: 'main',url:'https://github.com/Sreenidhi-KR/SpeMiniProject'
           }
       }
       stage('Maven build') {
           steps {
               script{
                   sh 'mvn clean install '
               }
           }
       }
       stage('Docker build') {
           steps {
               script{
                   imageName = docker.build "sreenidhikr/calculator:latest"
               }
           }
       }
       stage('Push Docker Image') {
           steps {
               script{
                   docker.withRegistry('' , 'docker-hub'){
                       imageName.push()
                   }
               }
           }
       }

       stage('Pull Ansible Docker Image') {
           steps {
               script{
                       ansiblePlaybook colorized: true, credentialsId: 'multipasskey',
disableHostKeyChecking:true,installation:'Ansible',inventory:'./deploy-docker/inventor
y', playbook: './deploy-docker/deploy-image.yml'
                   }
           }
       }
   }
}
```

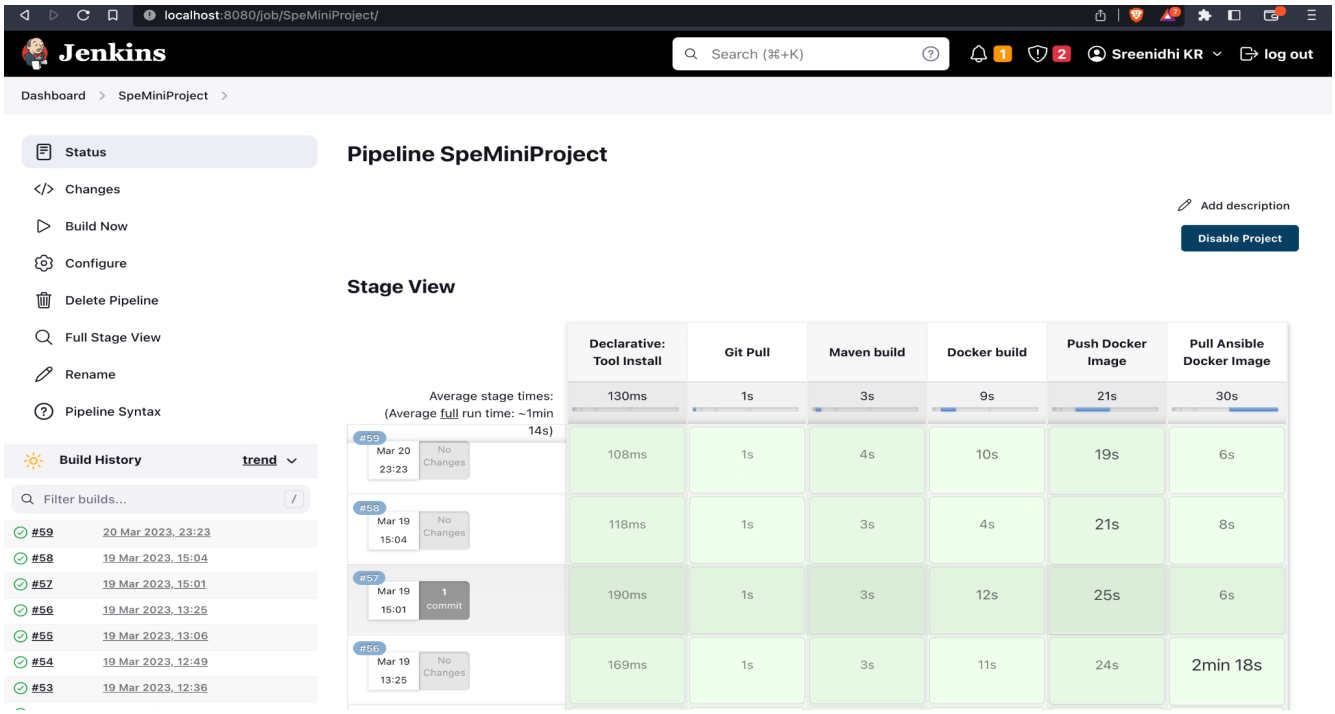
- "Git Pull" stage
  In this stage, the script checks out the code from a Git repository located at https://github.com/Sreenidhi-KR/SpeMiniProject. The code is checked out from the "main" branch of the repository.

- "Maven build" stage
  In this stage, the script builds the application using Maven. "mvn clean install" command, builds the application and produces an executable JAR file.

- "Docker build" stage
  In this stage, the script builds a Docker image of the application using the docker.build command .The name of the Docker image is "sreenidhikr/calculator:latest", the DockerFile contained in the repo specifies other details like default run command , work directory

- "Push Docker Image" stage:
  In this stage, the script pushes the Docker image to DockerHub

- "Pull Ansible Docker Image" stage:
  In this stage, the script pulls, Ansible deploys the Docker image in the client nodes using the ansiblePlaybook command. Steps and Configurations are given in the Ansible playbook named "deploy-image.yml" located in the "./deploy-docker" directory, we also provide the location of the inventory file.The credentialsId option specifies the ID of the private SSH Key to be used for authentication.



Once all steps in the pipeline are executed successfully , login to the client VM and check if the docker image is available using *docker images* command

```
ubuntu@testvm:~$ docker images
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
sreenidhikr/calculator  latest    67e72c099123   35 hours ago    501MB
hello-world             latest    46331d942d63   12 months ago   9.14kB
```

Run the container using *docker run -it sreenidhikr/calculator* command

```
ubuntu@testvm:~$ docker run -it sreenidhikr/calculator
Scientific Calculator using DevOps.
 Choose operation:
1. Factorial
2. Square root
3. Power
4. Natural Logarithm
5. Exit
Enter your choice:
```

Continuous Monitoring using ELK

The ELK Stack is a collection of open-source tools that are commonly used to collect, store, and analyze large amounts of data in real-time. The name ELK is an acronym that stands for:

- Elasticsearch: A distributed, RESTful search and analytics engine designed for horizontally scalable search and analysis of data.
- Logstash: A server-side data processing pipeline that allows you to collect, parse, and transform log data and other event data from a variety of sources.
- Kibana: A data visualization and exploration tool that allows you to interact with data stored in Elasticsearch and create real-time dashboards, charts, and graphs.

Together, these three tools form a powerful data processing and analysis platform that can be used to monitor and analyze data in a variety of contexts, including security, operations, and business analytics.

Installing ELK using docker

- git clone https://github.com/deviantony/docker-elk
- cd docker-elk
- docker-compose up -d
- start the elastic using localhost:5601, username and password for is *elastic* and *changeme*

We shall use log4j package for logging

- Add the log4j dependency to your project's pom.xml file:

```
<dependency>
   <groupId>log4j</groupId>
   <artifactId>log4j</artifactId>
   <version>1.2.17</version>
 </dependency>
```

- Add the following plugin in pom.xml

```
<build>
      <plugins>
          <plugin>
              <groupId>org.apache.maven.plugins</groupId>
              <artifactId>maven-shade-plugin</artifactId>
              <executions>
                  <execution>
                      <phase>package</phase>
                      <goals>
                          <goal>shade</goal>
                      </goals>
                      <configuration>
                          <transformers>
<transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                                  <mainClass>org.example.Main</mainClass>
                              </transformer>
                          </transformers>
                      </configuration>
                  </execution>
              </executions>
          </plugin>
      </plugins>
  </build>
```

- Save the pom.xml file and run the following command to update your project's dependencies: *mvn clean install*

- Once the dependencies are updated, you can start using log4j in your Java code by adding the following import statement:

  import org.apache.log4j.Logger;
  Logger logger = Logger.getLogger(MyClass.class);
  logger.debug("Debug message");

- Add a log4j2.xml configuration file under src/resources

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
   <Appenders>
       <Console name="ConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} -
%msg%n" />
       </Console>
                        <File    name="FileAppender"    fileName="calculator.log"
immediateFlush="false" append="true">
              <PatternLayout pattern="%d{yyy-MM-dd HH:mm:ss.SSS} [%t] %-5level
%logger{36} - %msg%n"/>
       </File>
   </Appenders>
   <Loggers>
       <Root level="debug">
           <AppenderRef ref="ConsoleAppender" />
           <AppenderRef ref="FileAppender"/>
       </Root>
   </Loggers>
</Configuration>
```

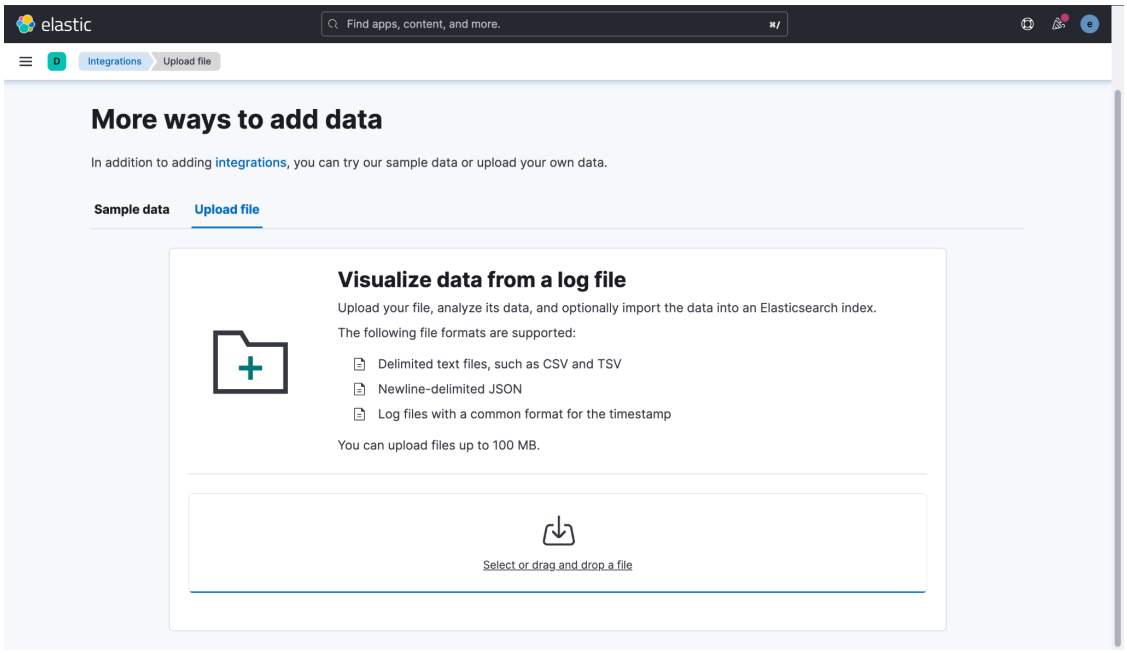
- Copy the log file from docker to . (current dir) in VM

  docker ps -all
  docker cp <container-id>:calculator.log .

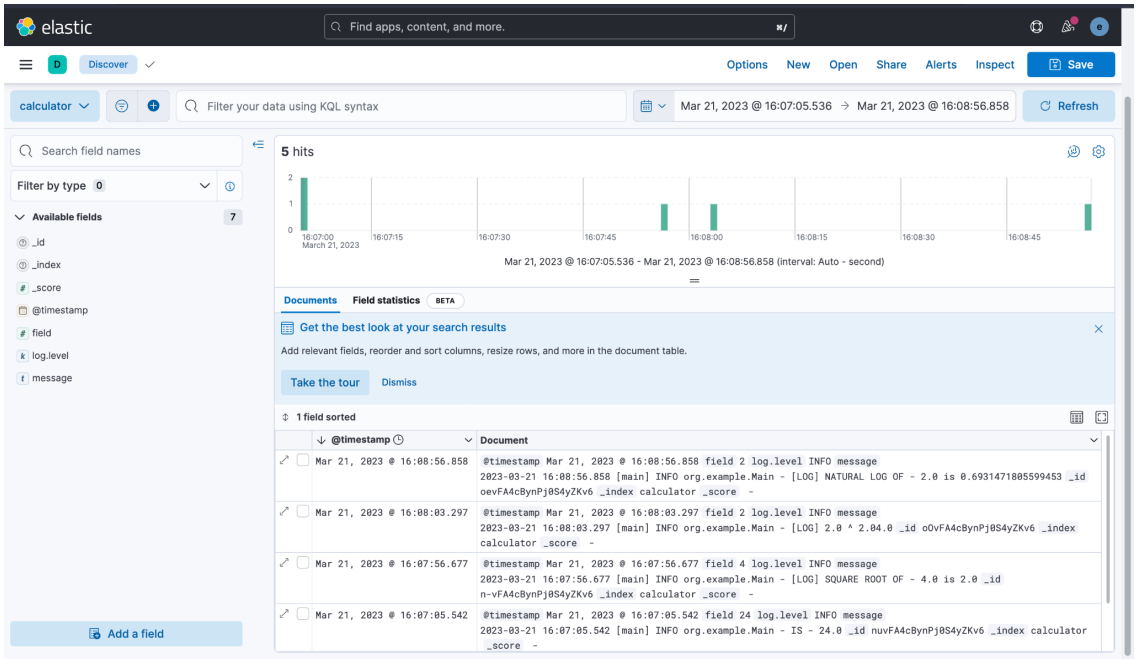
- Copy from Client VM to Server (Macbook)

  multipass transfer testvm:calculator.log .

- Upload the calculator.log file here and click on import



- After uploading we will see an import button, click on it and provide a name for visualizing the data. Then after all the processing we can see visualize the log file which we have uploaded



Conclusion

In conclusion, DevOps is a software development methodology that focuses on collaboration, communication, and automation between the software development and IT operations teams. It is designed to streamline the software development process, reduce time-to-market, and improve the overall quality of the software.

DevOps is not a one-size-fits-all approach and requires a cultural shift within organizations. It involves implementing best practices such as continuous integration, continuous delivery, and continuous testing. It also involves the use of tools such as configuration management, monitoring, and orchestration.

Implementing DevOps can have numerous benefits for organizations, including faster time-to-market, higher quality software, increased efficiency, and improved collaboration between teams. However, it requires a significant investment in terms of time, resources, and infrastructure.

Overall, DevOps is a powerful methodology that can help organizations to develop and deliver software more effectively, but it requires careful planning and implementation to achieve its full potential.