Calculator with Devops

1.Problem Statement:

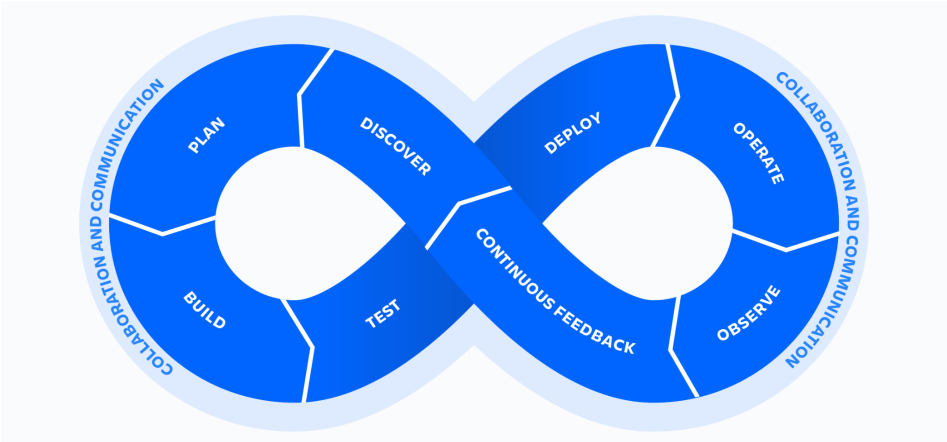
Create a scientific calculator program with user menu driven operations

- Square root function - √ x
- Factorial function - x !
- Natural logarithm (base e) - l n(x)
- Power function - x^b

2.DevOps:

2.1 What is DevOps?

DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams.The DevOps lifecycle consists of eight phases representing the processes, capabilities, and tools needed for development (on the left side of the loop) and operations (on the right side of the loop). Throughout each phase, teams collaborate and communicate to maintain alignment, velocity, and quality.



2.2 Why Devops:

- **Speed:** Teams that practice DevOps release deliverables more frequently, with higher quality and stability. Continuous delivery allows teams to build, test, and deliver software with automated tools.
- **Improved collaboration:** The foundation of DevOps is a culture of collaboration between developers and operations teams, who share responsibilities and combine work. This makes teams more efficient and saves time related to work handoffs and creating code that is designed for the environment where it runs.

- **Rapid deployment:**By increasing the frequency and velocity of releases, DevOps teams improve products rapidly. A competitive advantage can be gained by quickly releasing new features and repairing bugs.

- **Quality and reliability:**Practices like continuous integration and continuous delivery ensure changes are functional and safe, which improves the quality of a software product. Monitoring helps teams keep informed of performance in real-time.

- **Security:** By integrating security into a continuous integration, continuous delivery, and continuous deployment pipeline, DevSecOps is an active, integrated part of the development process. Security is built into the product by integrating active security audits and security testing into agile development and DevOps workflows.

3.Tools Used:

- Continuous Development (SCM): GitHub
- Continuous Build: Maven
- Continuous Testing: Junit
- Continuous Integration: Jenkins
- Containerization: Dockerfile
- Continuous Deployment: Ansible
- Continuous Monitoring: Elasticsearch-Logstash-Kibana

4.Steps involved:

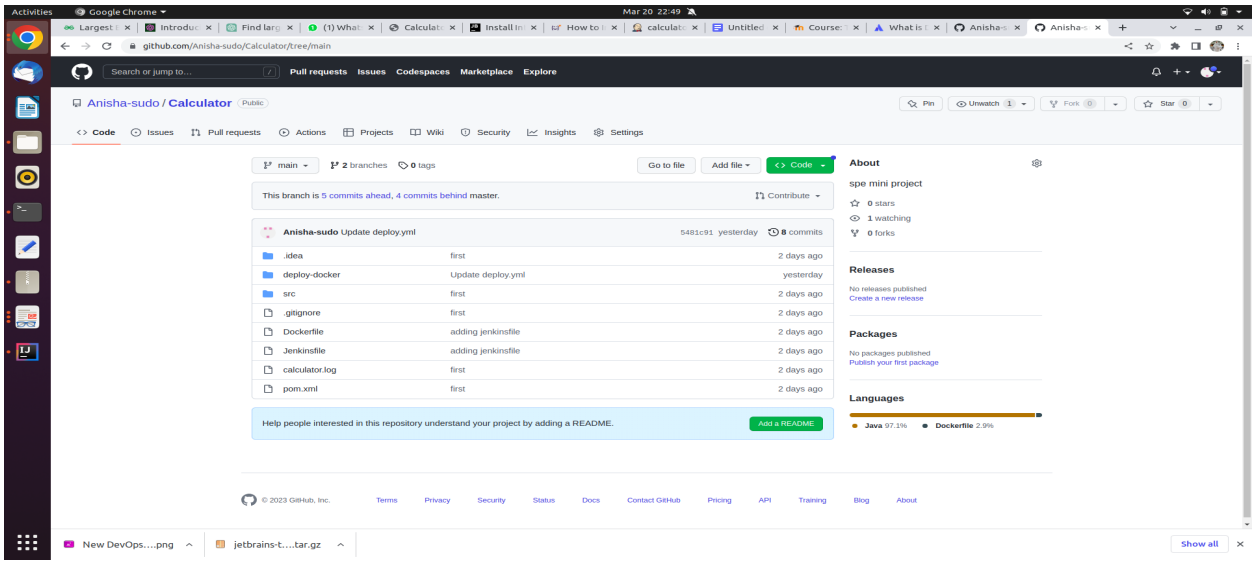
The first step involves creating a remote named origin for the repository we wish to push our git folder to.

4.1 Source Code Management:GIT

Commands to integrate your local source :
->git init //making your current local directory as git repository.
->git remote add origin https://github.com/Anisha-sudo/Calculator.git
The above command will create a remote to connect to your remote github repository.Make sure you already have a remote github repository whose link i am providing above .

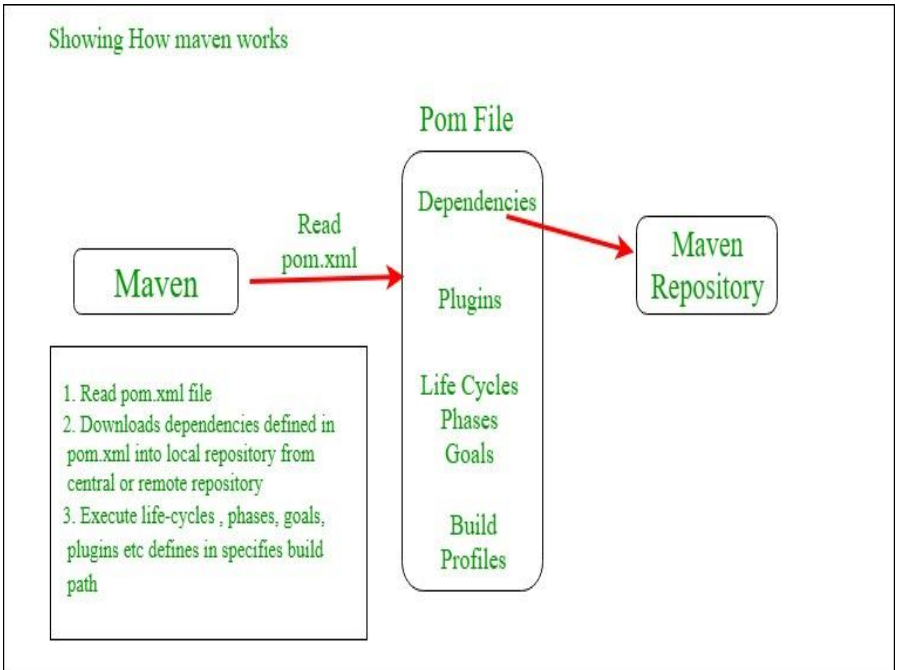
Once this is created you can add and commit the changes to your git and push it to github using the following command:
->git add - - a // will stage all the updated files .
->git commit -m "tag" //will commit the staged files.
->git push -u origin main // will push the commits to github repository

Provide your github id and token to push your git folder to the branch created(here main).

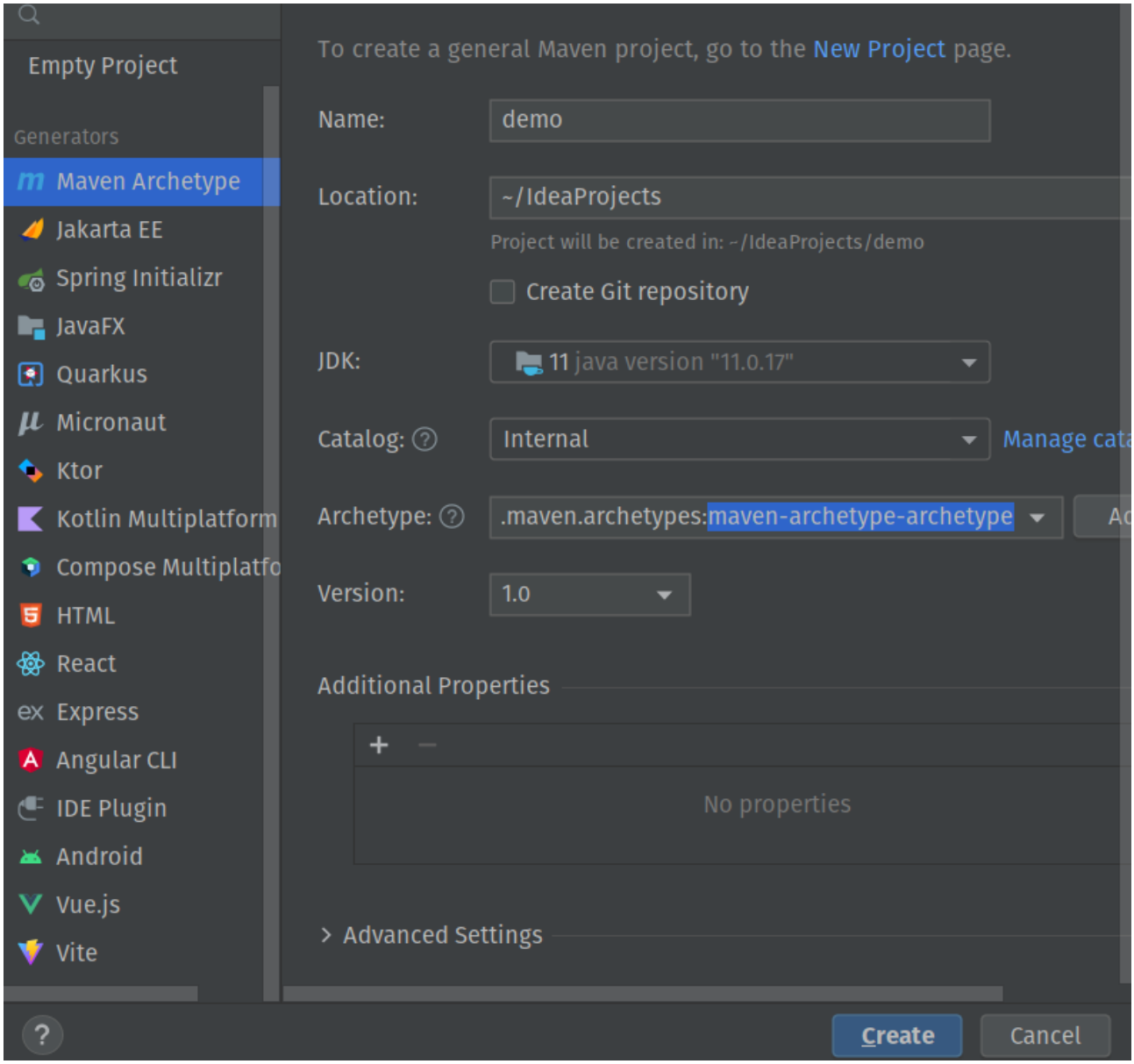


4.2 Continuous Building:Maven

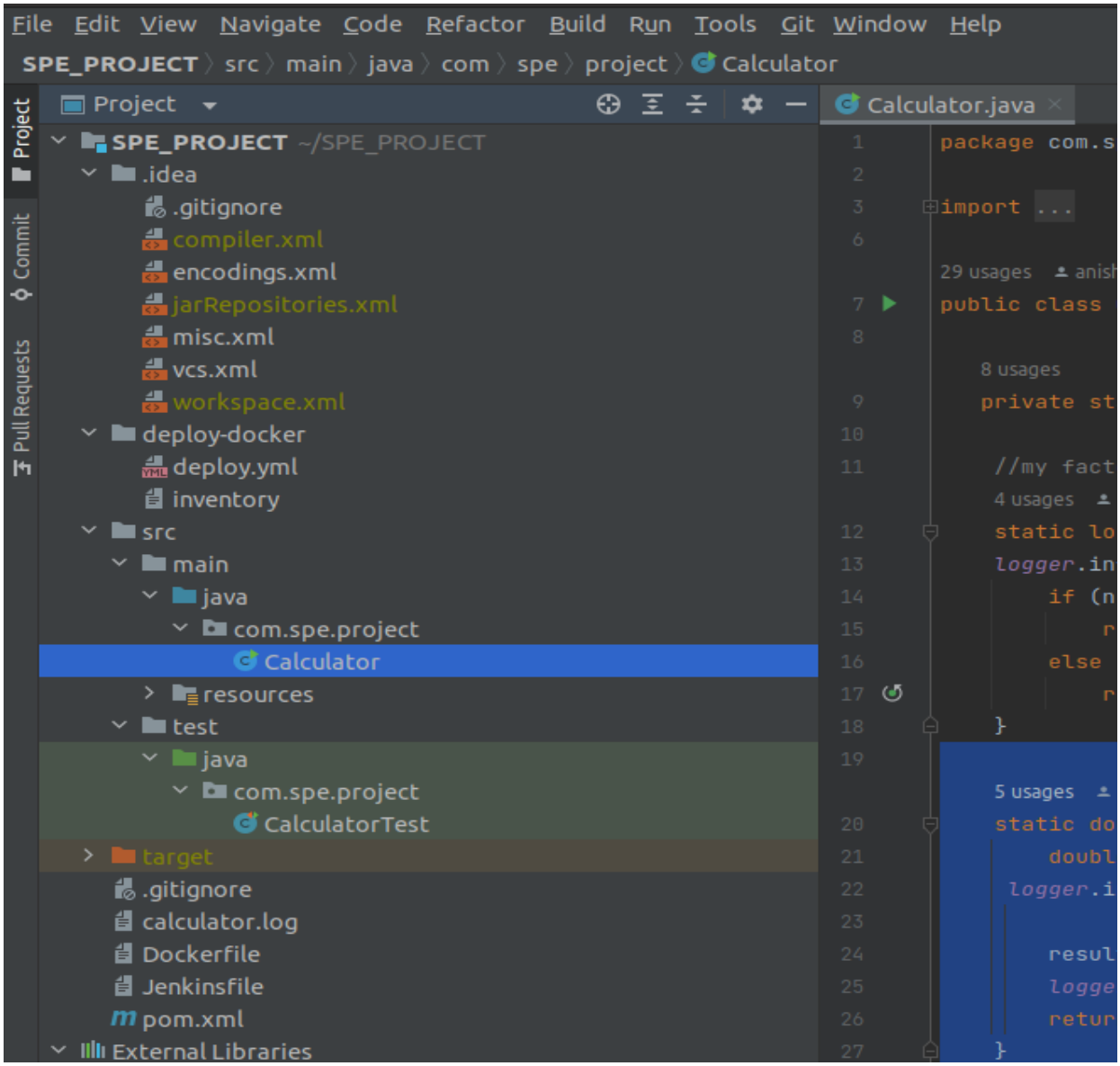
Maven is a popular open-source build tool Maven focuses on the simplification and standardization of the building process, taking care of builds ,documentation, dependencies, reports, SCMs,distribution,releases,mailing list etc.The image below describes the working of maven. It begins with reading the pom.xml file ,which mentions the required dependencies and



builds the project by making the dependencies available in case it’s not present in the local repository.
The image below depicts creation of maven project:



The image below depicts the tree structure of the project. Here we have created the java file under com.spe.project package.This file contains the java code for implementation of calculator.
The test folder contains the java file which contains the test cases to be run.We will look into it further in the continuous testing section.



4.3 Continuous Testing:Junit

JUnit is a unit testing framework for the Java programming language. It plays a crucial role in test-driven development. It provides annotations to identify test methods and provides assertions for testing expected results.

```java
no usages  anisha
public class CalculatorTest {

    10 usages
    private Calculator objCalcUnderTest;
    9 usages
    private static final double DELTA = 1e-15;
    no usages  anisha
    @Before
    public void setUp() {
        //Arrange
        objCalcUnderTest = new Calculator();
    }

    no usages  anisha
    @Test
    public void PowerTruePositive() {
        int x = 2; int b = 3;
        double expectedResult = 8.0;
        //Act
        double result = objCalcUnderTest.x_power_b(x, b);
        //Assert
        assertEquals( message: "Finding factorial for true positive", expectedResult, result, DELTA);
    }
    no usages  anisha
    @Test
    public void SquareTruePositive(){
        assertEquals( message: "Finding square root of a number",  expected: 4, objCalcUnderTest.square_root( n: 16), DELTA);
        assertEquals( message: "Finding square root of a number",  expected: 6, objCalcUnderTest.square_root( n: 36), DELTA);
        assertEquals( message: "Finding square root of a number",  expected: 8, objCalcUnderTest.square_root( n: 64), DELTA);
    }
```

```xml
    <version>2.20.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.20.0</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
```

The pom.xml file contains the dependencies for Junit.

Once this is done we will package our Maven project into an executable Jar file.Here we are using Apache Maven Assembly Plugin(you can also use other plugins as well like apache maven shade plugin etc).Mention the package name and file in the main class and the name of jar file you wish to create.

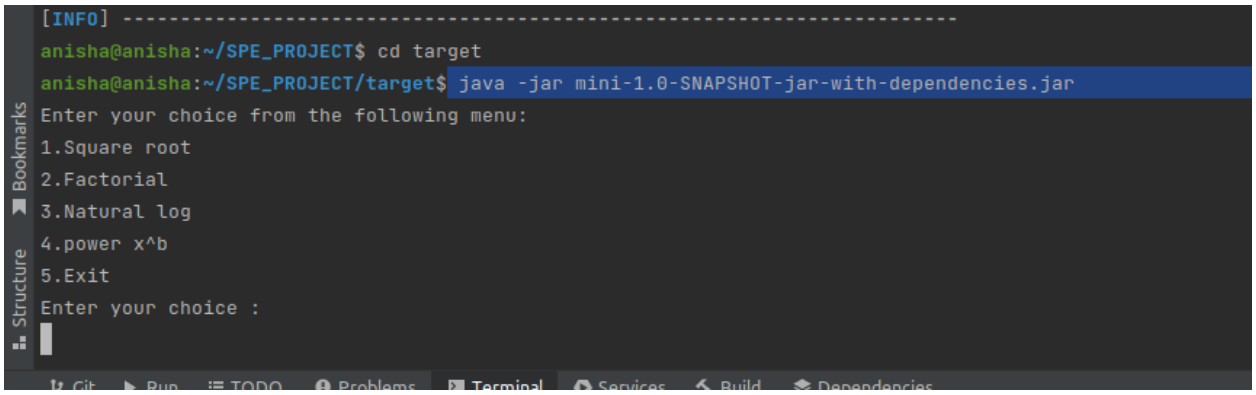
```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-assembly-plugin</artifactId>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
      <configuration>
        <archive>
          <manifest>
            <mainClass>com.spe.project.Calculator</mainClass>
          </manifest>
        </archive>
        <descriptorRefs>
          <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
      </configuration>
    </execution>
  </executions>
```

Execute the below command to build and check all test cases

→$ *mvn clean install*

A temporary folder named “target” will be generated in which mini-1.0-SNAPSHOT-jar-with-dependencies.jar will be generated. (it can always be removed using *mvn clean* and recreated by build or *mvn clean install*.Target folder is basically where the files are generated during the build from the sources, serving a single purpose: create the artifacts of the project.).

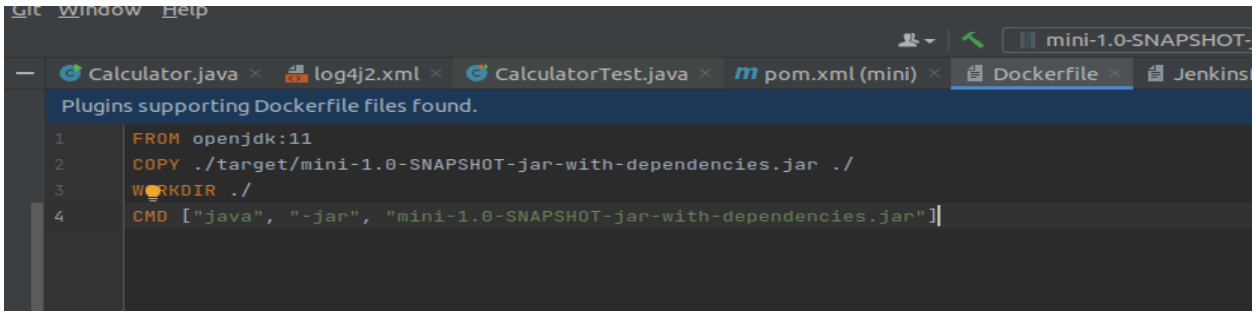
Once we get the jar file we can run our project by using the below command:



4.4 Containerization

4.4.1 Creating docker file

After creation of the jar file we will create a Dockerfile which will create an image for the build we have created using the jar file.The snap for dockerfile is provided below:



Here the jar file is copied to the root of the container and the command is mentioned which has to be executed while we run the container.

We can test the working of the dockerfile by the following command:

docker build -t democalci—-->this builds the image with name democalci

docker run -i democalci—-->will create a container

```
ubuntu                          latest    08d22c0ceb15   13 days ago          77.8MB
hello-world                     latest    feb5d9fea6a5   18 months ago        13.3kB
anisha@anisha:~/SPE_PROJECT$ docker build -t democalci .
[+] Building 2.6s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 194B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/openjdk:11
 => [auth] library/openjdk:pull token for registry-1.docker.io
 => [internal] load build context
 => => transferring context: 2.15MB
 => CACHED [1/3] FROM docker.io/library/openjdk:11@sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e5
 => [2/3] COPY ./target/mini-1.0-SNAPSHOT-jar-with-dependencies.jar ./
 => exporting to image
 => => exporting layers
 => => writing image sha256:1de105b56d9bad28316219c3f659f80db66e45b302093aae1d32de67c29ce9ab
 => => naming to docker.io/library/democalci
anisha@anisha:~/SPE_PROJECT$ docker images
REPOSITORY                      TAG       IMAGE ID       CREATED              SIZE
democalci                       latest    1de105b56d9b   10 seconds ago       656MB
```

However in our project we are implementing this using jenkins pipeline and the image created will be named as anisha0987/spe_mini_project as mentioned in the script for the ‘docker build to image’ stage.

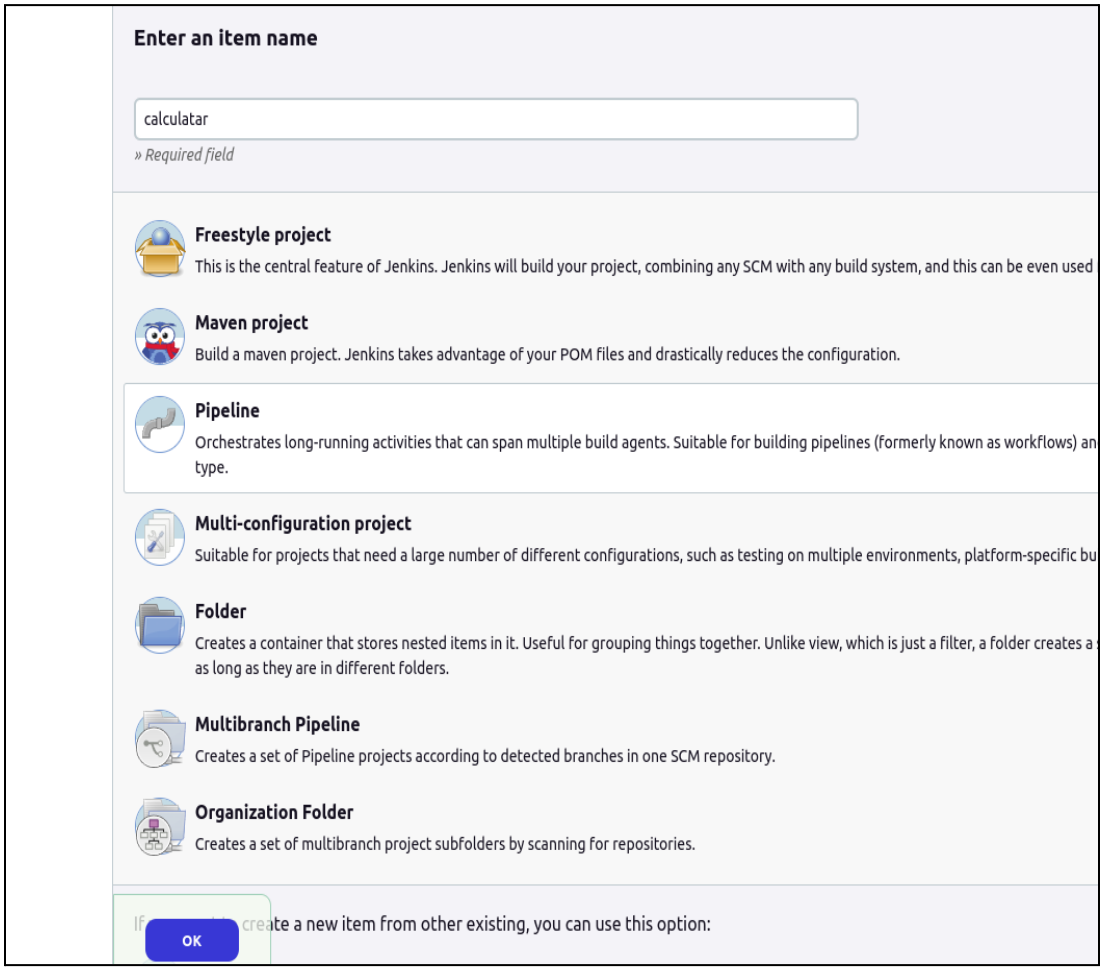
```
        }
    }
    stage('Docker Build to Image') {
        steps {
            script{
                imageName=docker.build "anisha0987/spe_mini_project"
            }
        }
    }

    stage('Push Docker Image') {
        steps {
            script{
                docker.withRegistry('','docker-jenkins'){
                    imageName.push()
                }
            }
        }
    }
    stage('Deploy') {
        steps {
            ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyCheckin
             playbook: 'deploy-docker/deploy.yml', sudoUser: null, extras: '-e "imag
```

4.5 Continuous Integration

4.5.1 Creating the jenkins pipeline

Jenkins Pipeline basically comprises of Sequence of stages to perform the given tasks such as pulling code from the Git repository, static code analysis, building project, executing the unit tests, automated tests,performance tests, and deploying application
Steps:

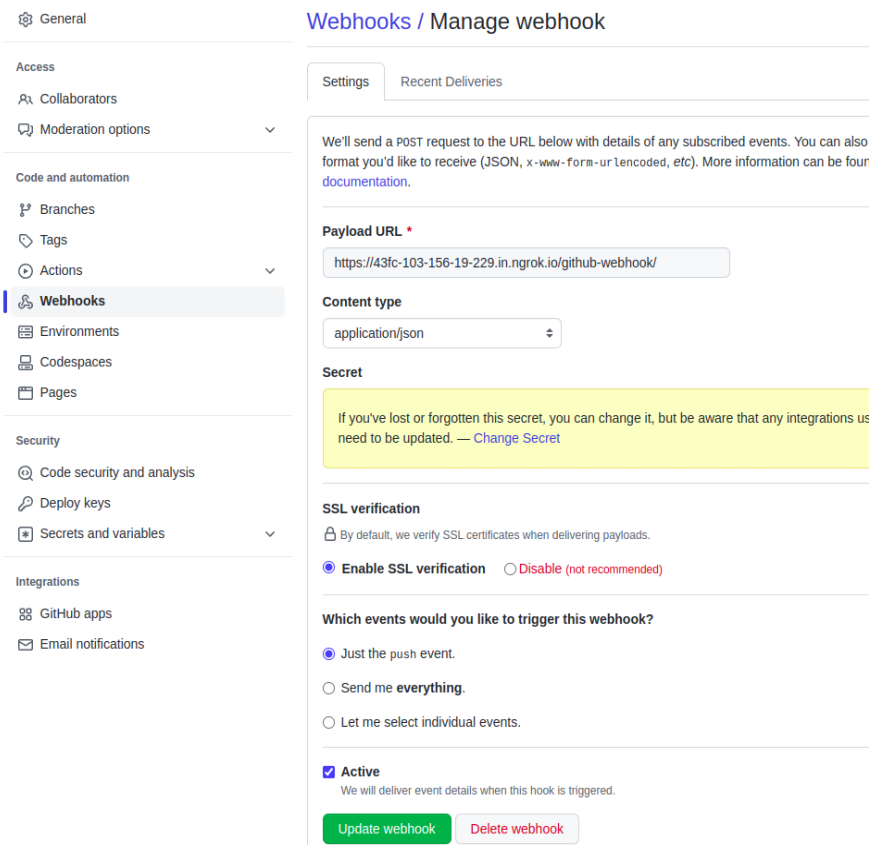
1.Create new jenkins project : Click new -> enter project name ->
2.select pipeline project -> click OK



3.Mention Jenkinsfile(script file) in the script path option in the configurations.
4.And follow the below steps for git scm polling:

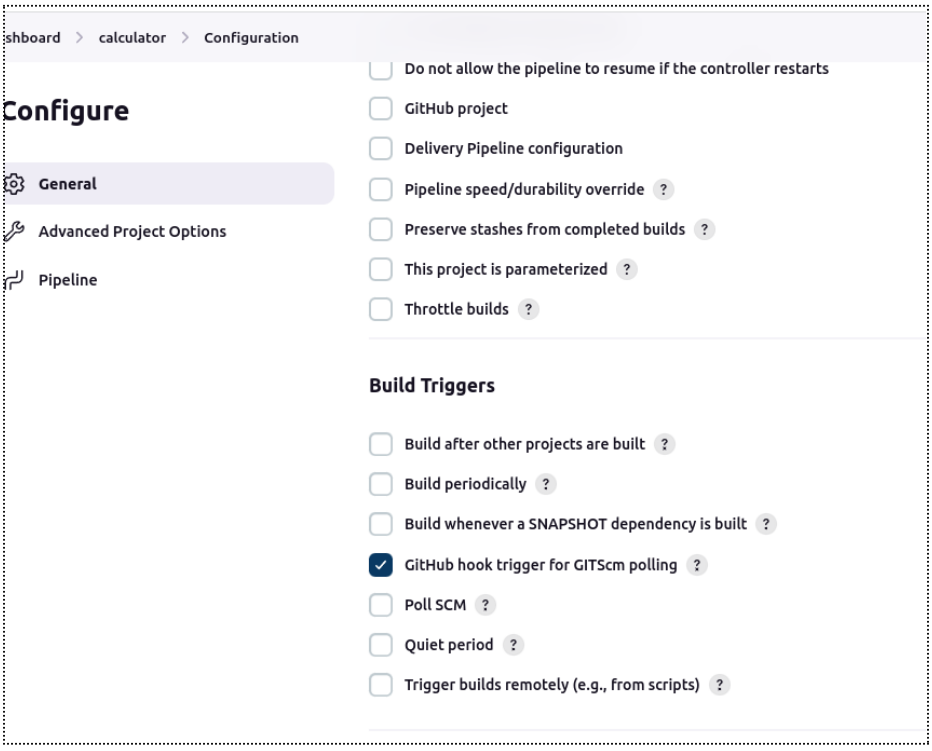
The first thing we would be doing is to expose our localhost to public ip via **NGROK and configure jenkins for github hook trigger for GITscm polling.**

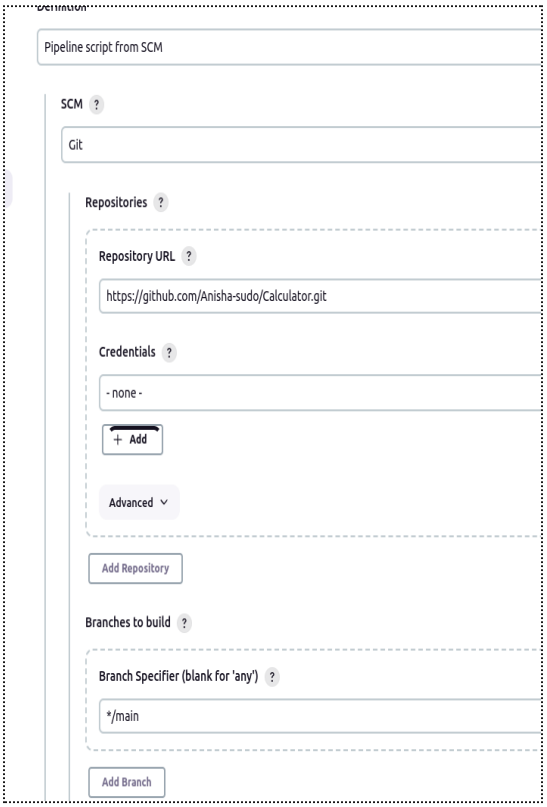
1.install ngrok by signing up at https://ngrok.com/
2.run the command to expose the port (here 8080)
—-->ngrok http 8080
3.copy the public id.
4.Create a webhook for the repository you wish to trigger build.
5.Mention the public id provided by ngrok and the secret key.



(we can generate the secret key from github→developer setting->personal access token)

6.Now we need to configure the jenkins accordingly to trigger the build.





7.provide the details of git repository along with the secret token here as shown in the image below:

Enter the repository and branch details.

8.Go to dashboard->manage jenkins->configure systems:

Provide the ngrok public url in the jenkins url option.

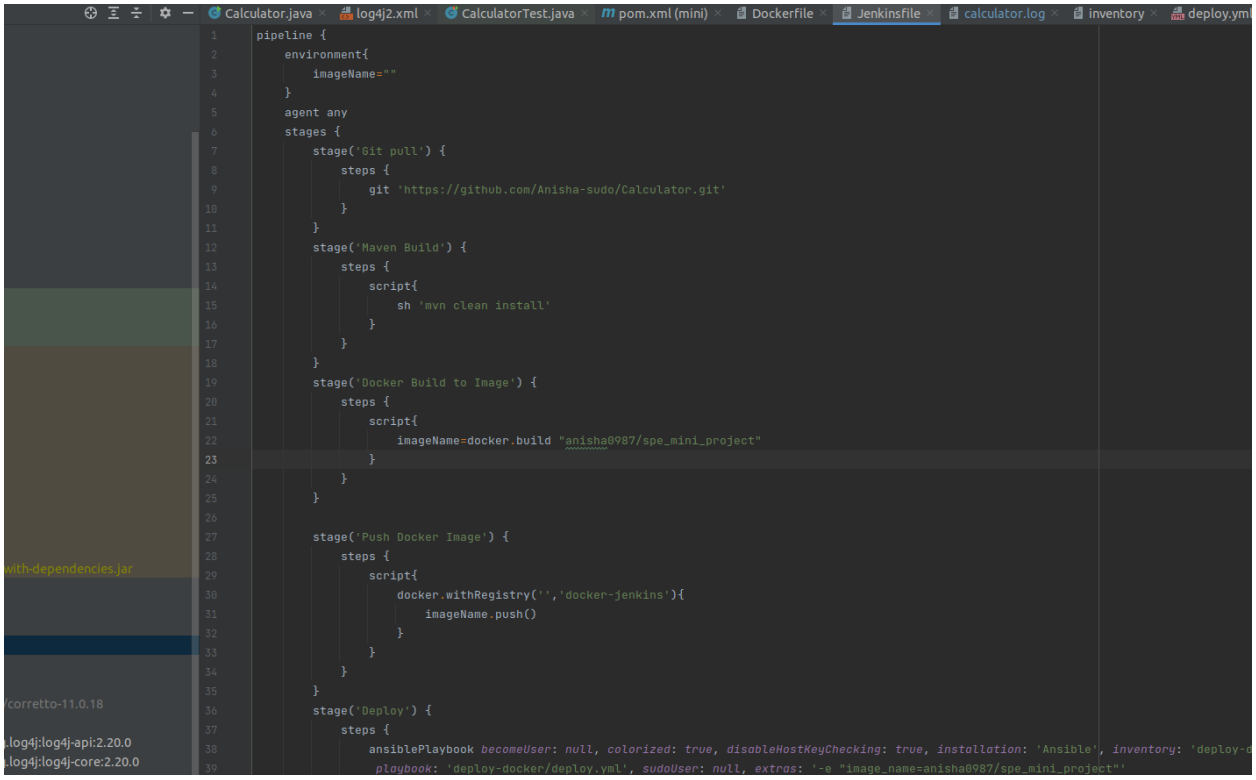
Now moving forward ,we will install the required plugins in jenkins:
Install the following plugins from the plugin manager

- Maven
- Git
- Docker
- Ansible
  Enter the path details for the plug ins in global tool configurations:

The above picture depicts the path provided for the installed plugin ANSIBLE.

Jenkinsfile details:

The Jenkinsfile is a script which is used to integrate various stages in a pipeline.This comprises SCM,build,creation of docker image,pushing image to dockerhub,and configuring the infrastructure.



BUILD:

1.Git pull:This stage pulls the repository from github

```
agent any
stages {
    stage('Git pull') {
        steps {
            git 'https://github.com/Anisha-sudo/Calculator.git'
        }
    }
```

2.Maven build:This stage generates a jar file that contains our source code as well as any dependencies. The existing target folder with old dependencies are deleted,and a new target folder with the new jar file will be created.

```
stage('Maven Build') {
    steps {
        script{
            sh 'mvn clean install'
        }
    }
}
```

3.Docker Image Creation: It is used to produce images on our local system that are then posted to our Docker hub, allowing us to pull the image and run the application on other servers.Provide the docker hub repository name as the image name here.

```
stage('Docker Build to Image') {
    steps {
        script{
            imageName=docker.build "anisha0987/spe_mini_project"
        }
    }
}
```

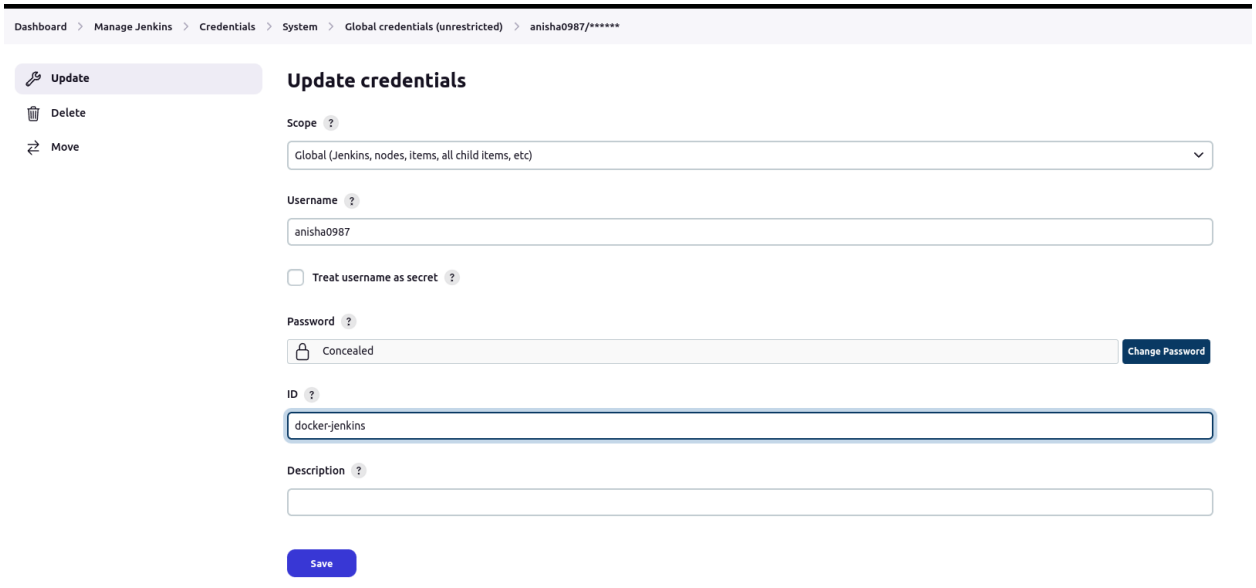
4.Pushing docker image to dockerhub:
Here we are deploying the image into DockerHub so that anyone can pull the image.In this case, the "docker.withRegistry" step is used to configure the Docker registry to be used for pushing the image. The first argument is an empty string, which indicates that the default Docker registry will be used. The second parameter(docker-jenkins) is the id which informs jenkins about the docker hub id and password and allows it to push the image to it.

The "imageName.push()" step is used to push the Docker image to the configured registry. "imageName" is a variable that contains the

name of the Docker image that was built in an earlier stage of the pipeline.(i.e.anisha0987/spe_mini_project)

```
stage('Push Docker Image') {
    steps {
        script{
            docker.withRegistry('','docker-jenkins'){
                imageName.push()
            }
        }
    }
}
```

Also in order to access the docker hub we have to enter the credentials in jenkins as provided in the image below:
Go to Dashboard->Manage Jenkins->Credentials->System ->Global credentials (unrestricted)
Click on “add credentials ” on the top right corner and then enter the Docker Hub id and password along with the id (“docker-jenkins”)

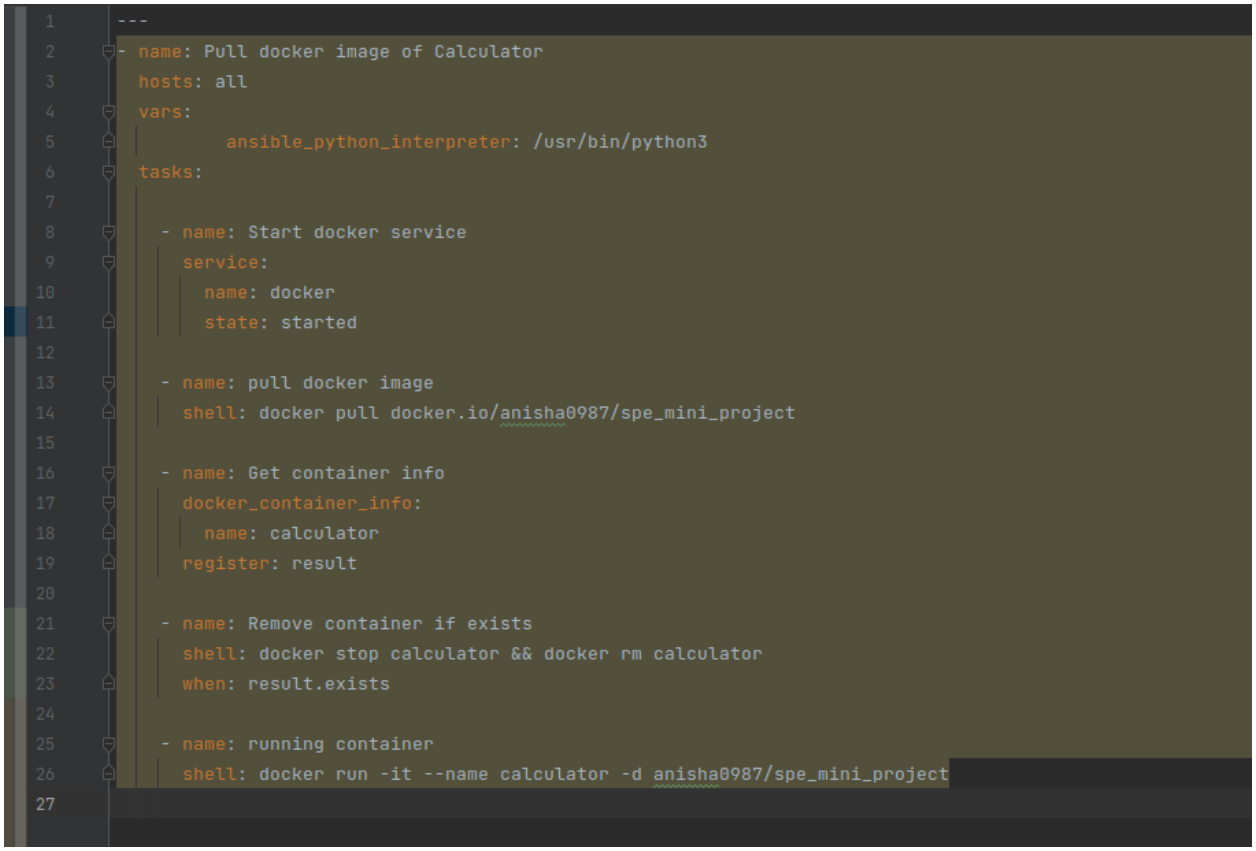


4.6 Continuous Deployment:

Deployment stage:In this stage, we run our ansible playbook. The ansible playbook contains all the commands to be executed on a machine (called ansible node) and the ansible_node details are mentioned in the inventory file.(here we are deploying in our local machine).

```
    }
    stage('Deploy') {
        steps {
            ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'deploy-docker/inventory',
             playbook: 'deploy-docker/deploy.yml', sudoUser: null, extras: '-e "image_name=anisha0987/spe_mini_project"'
        }
    }
```

The playbook for our project is shown in the image below:



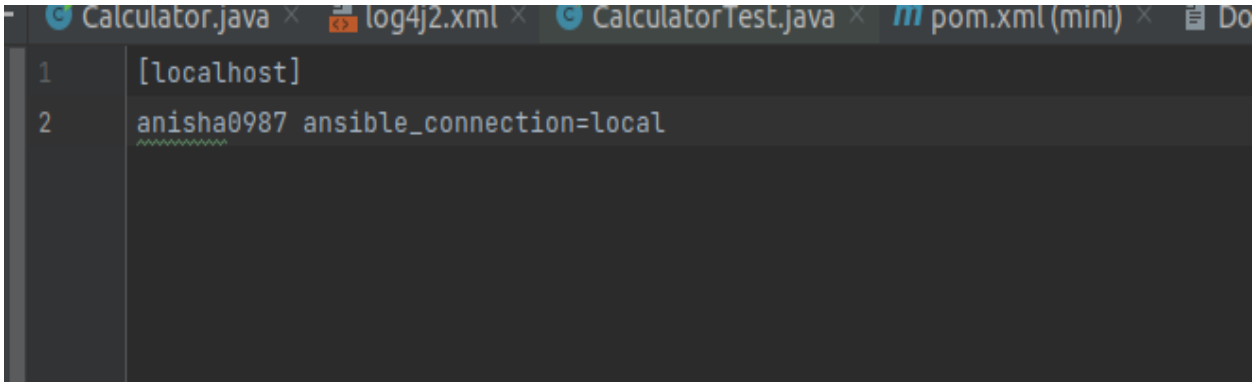
```
---
- name: Pull docker image of Calculator
  hosts: all
  vars:
        ansible_python_interpreter: /usr/bin/python3
  tasks:

    - name: Start docker service
      service:
        name: docker
        state: started

    - name: pull docker image
      shell: docker pull docker.io/anisha0987/spe_mini_project

    - name: Get container info
      docker_container_info:
        name: calculator
      register: result

    - name: Remove container if exists
      shell: docker stop calculator && docker rm calculator
      when: result.exists

    - name: running container
      shell: docker run -it --name calculator -d anisha0987/spe_mini_project

```

Here we are starting the docker service and then pulling the image

anisha0987/spe_mini_project from docker hub and creating the container named calculator and in case there already exists a docker we will stop and remove it and run the docker.

For Ansible to work make sure you have python
and ssh server installed on your controller and managed nodes.

The inventory file:



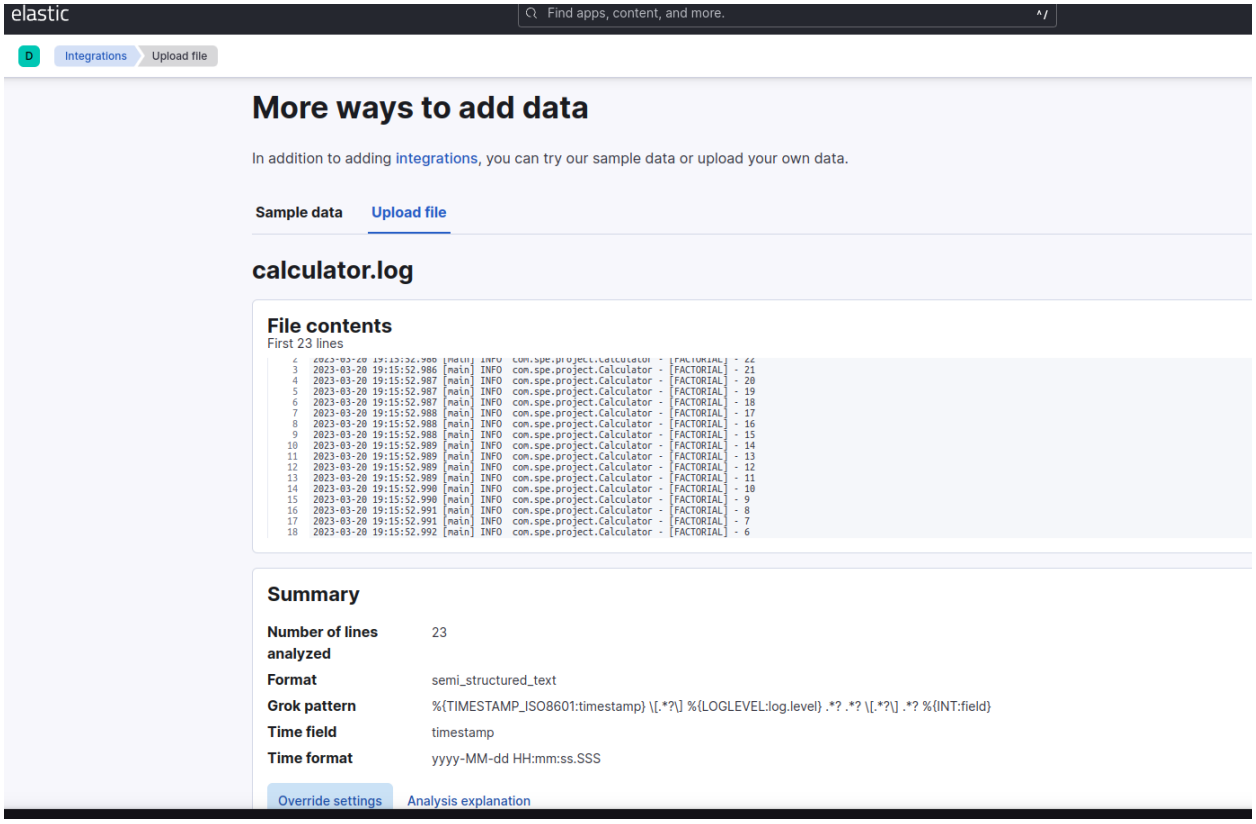
An inventory defines a collection of hosts that ansible will manage. These hosts can also be assigned to groups, which can be managed collectively. Groups can contain child groups, and hosts can be members of multiple groups.

4.6 Continuous Monitoring:

Tool: ELK - ElasticSearch, Logstash, Kibana

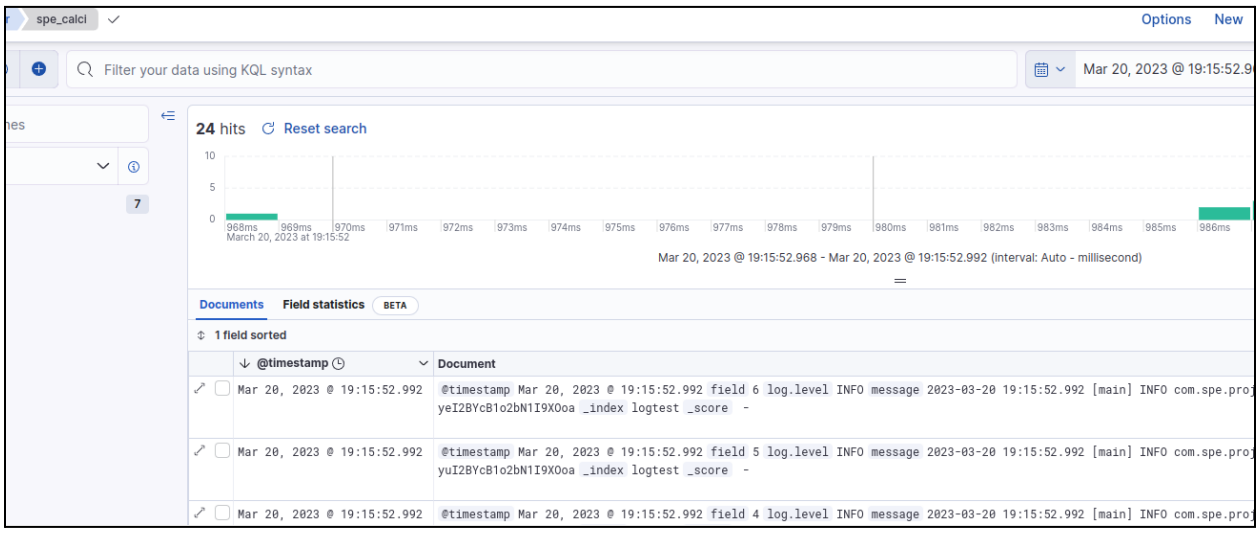
The ELK Stack helps by providing users with a powerful platform that collects and processes data from multiple data sources, stores that data in one centralized data store that can scale as data grows, and that provides a set of tools to analyze the data.

After performing the operations in the docker container Calculator.log is created. Copy that log file into your local machine.Add the log file into the ELK stack and record the stats.
Upload the log file in elastic cloud.



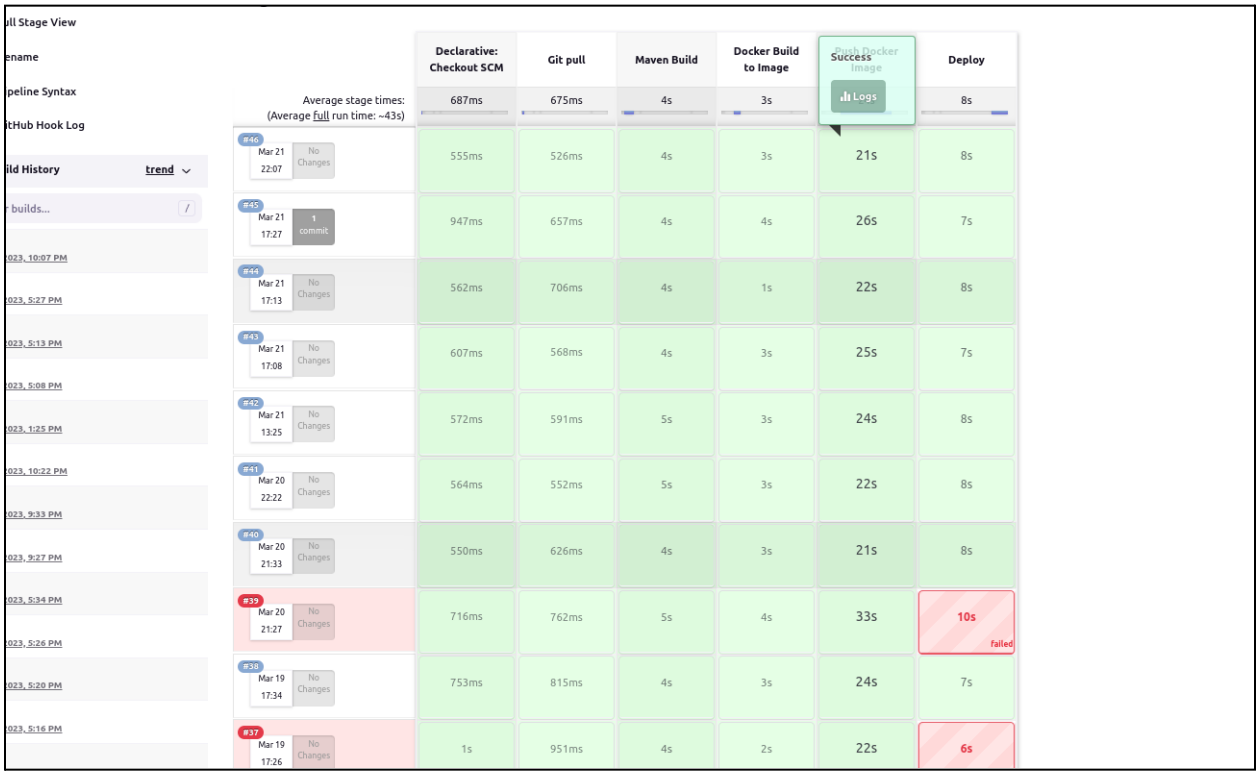
Click on the override setting and remove the constructed grok pattern and click on import.

The output:

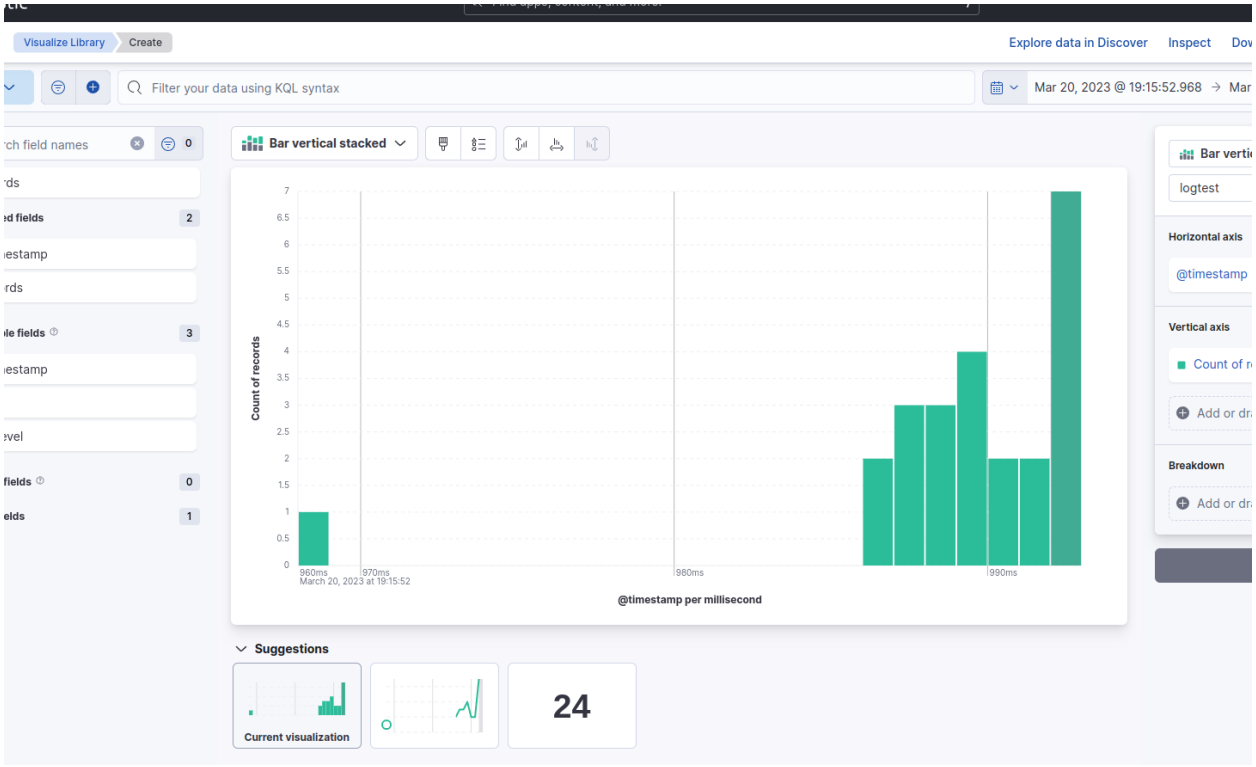


5.OUTPUT:

5.1 jenkins pipeline output:



5.2 Visualization output:



5.3 Calculator output:

```
antsha@antsha:~$ docker start -a -i a3
Enter your choice from the following menu:
1.Square root
2.Factorial
3.Natural log
4.power x^b
5.Exit
Enter your choice :
1
Enter the number
64
17:58:09.838 [main] INFO  com.spe.project.Calculator - [SQ ROOT] - 64
17:58:09.851 [main] INFO  com.spe.project.Calculator - [RESULT - SQ ROOT] - 8.0
Square root of 64 is 8.0
Do you want to continue?
 1.Yes 2.No
```

```
Enter your choice :
2
Enter the number
8
17:58:20.908 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 8
17:58:20.908 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 7
17:58:20.909 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 6
17:58:20.909 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 5
17:58:20.909 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 4
17:58:20.910 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 3
17:58:20.910 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 2
17:58:20.911 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 1
17:58:20.911 [main] INFO  com.spe.project.Calculator - [FACTORIAL] - 0
Factorial of 8 is 40320
```

```
Enter your choice from the following menu:
1.Square root
2.Factorial
3.Natural log
4.power x^b
5.Exit
Enter your choice :
3
Enter the number
10
17:59:26.162 [main] INFO  com.spe.project.Calculator
17:59:26.164 [main] INFO  com.spe.project.Calculator
Natural log (ln) of 10 is 2.302585092994046
Do you want to continue?
 1.Yes 2.No
```

```
nter your choice from the following menu:
.Square root
.Factorial
.Natural log
.power x^b
.Exit
nter your choice :
nter the first number->x
nter the second number->b
4
7:59:38.808 [main] INFO  com.spe.project.Calculator - [POWER of2RAISED TO]4
7:59:38.808 [main] INFO  com.spe.project.Calculator - [RESULT - POWER] - 16.0
 to the power 4 is 16.0
o you want to continue?
 1.Yes 2.No
```

We have successfully implemented the calculator using DEVOPS.

Thankyou
Anisha Rani
MT2022153