

# Calculator with Devops

## 1.Problem Statement:

Create a scientific calculator program with user menu driven operations

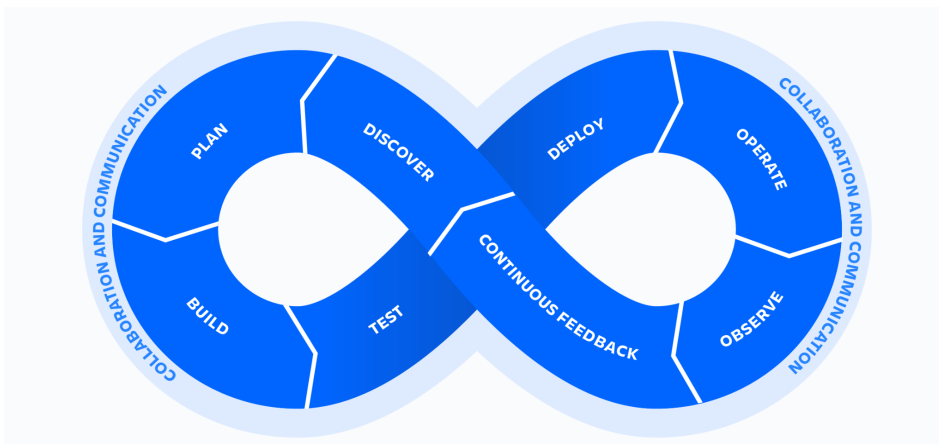
- Square root function -  $\sqrt{x}$
- Factorial function -  $x!$
- Natural logarithm (base e) -  $\ln(x)$
- Power function -  $x^b$

## 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 .

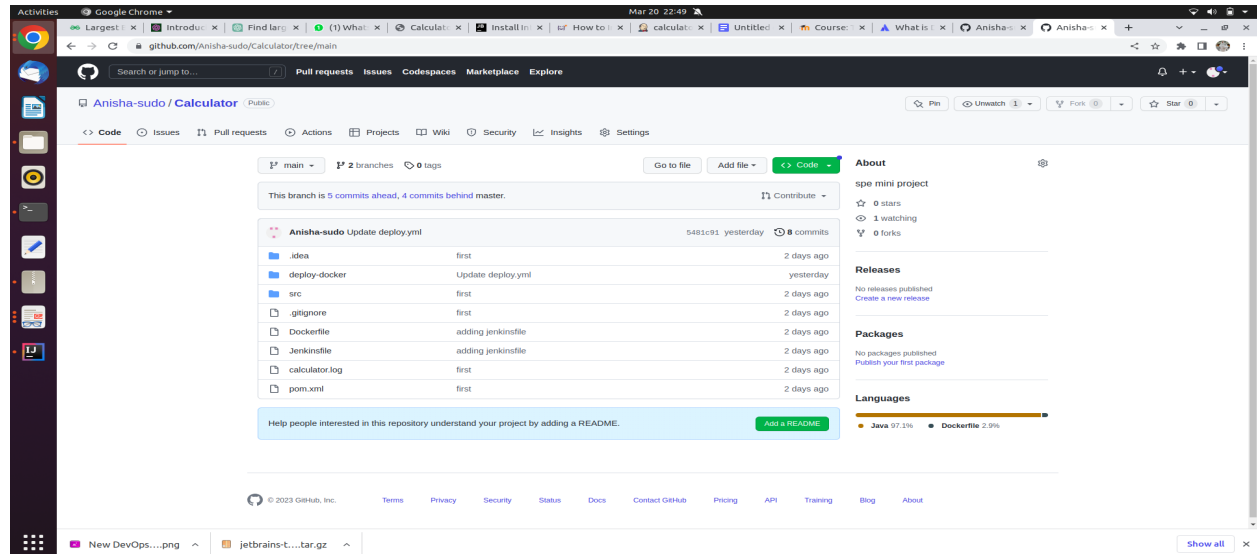
Once this is created you can add and commit the changes to your git and push it to github using the following command:

->*git add - .* // will stage all the updated files .

->*git commit -m "tag"* //will commit the staged files.

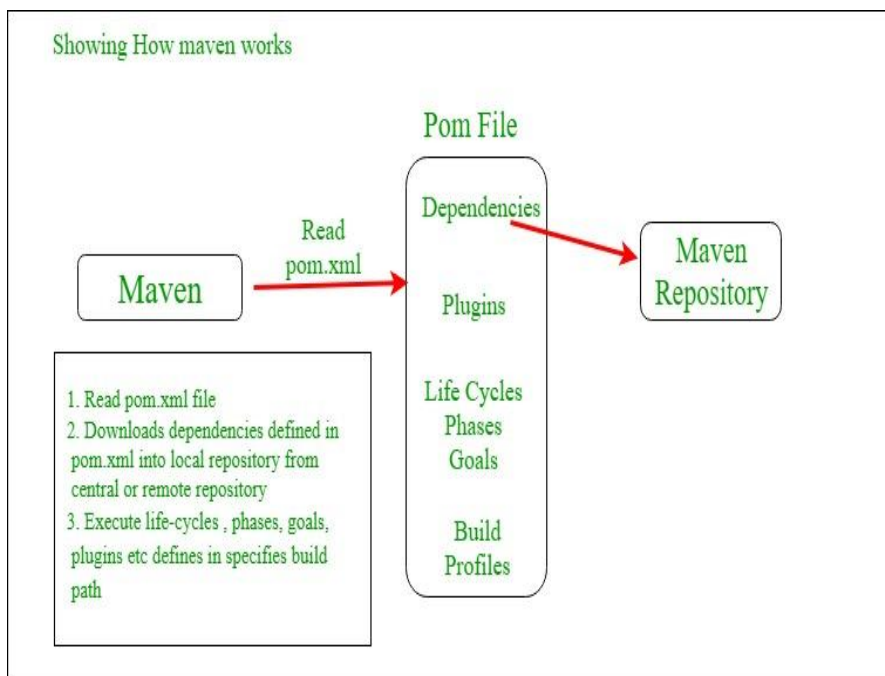
->*git push -u origin main* // will push the commits to github repository

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



## 4.2 Continuous Building:Maven

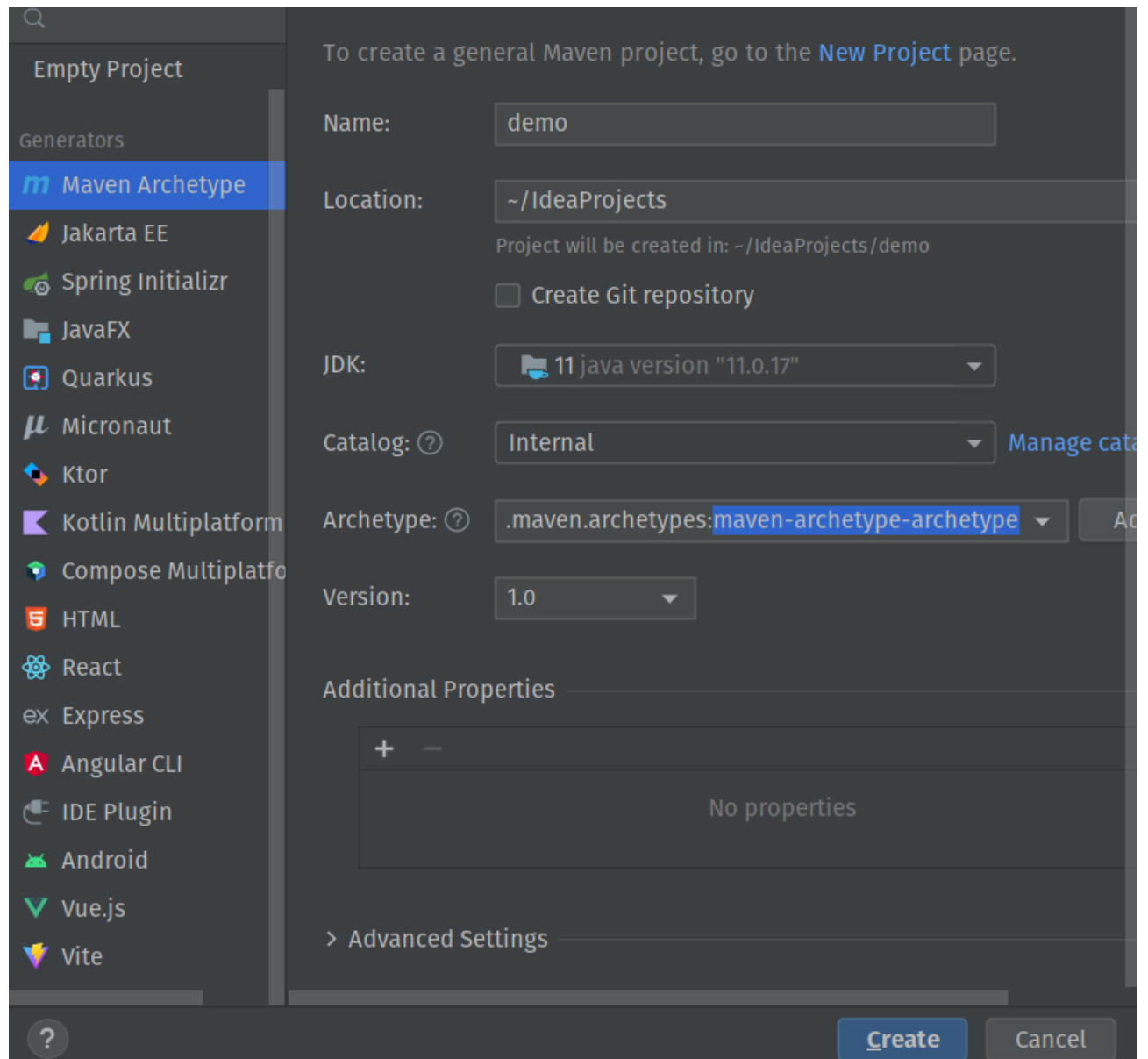
Maven is a popular open-source build tool. Maven focuses on the simplification and standardization of the building process, taking care of builds, documentation, dependencies, reports, SCMs, distribution, releases, mailing list etc. The image below



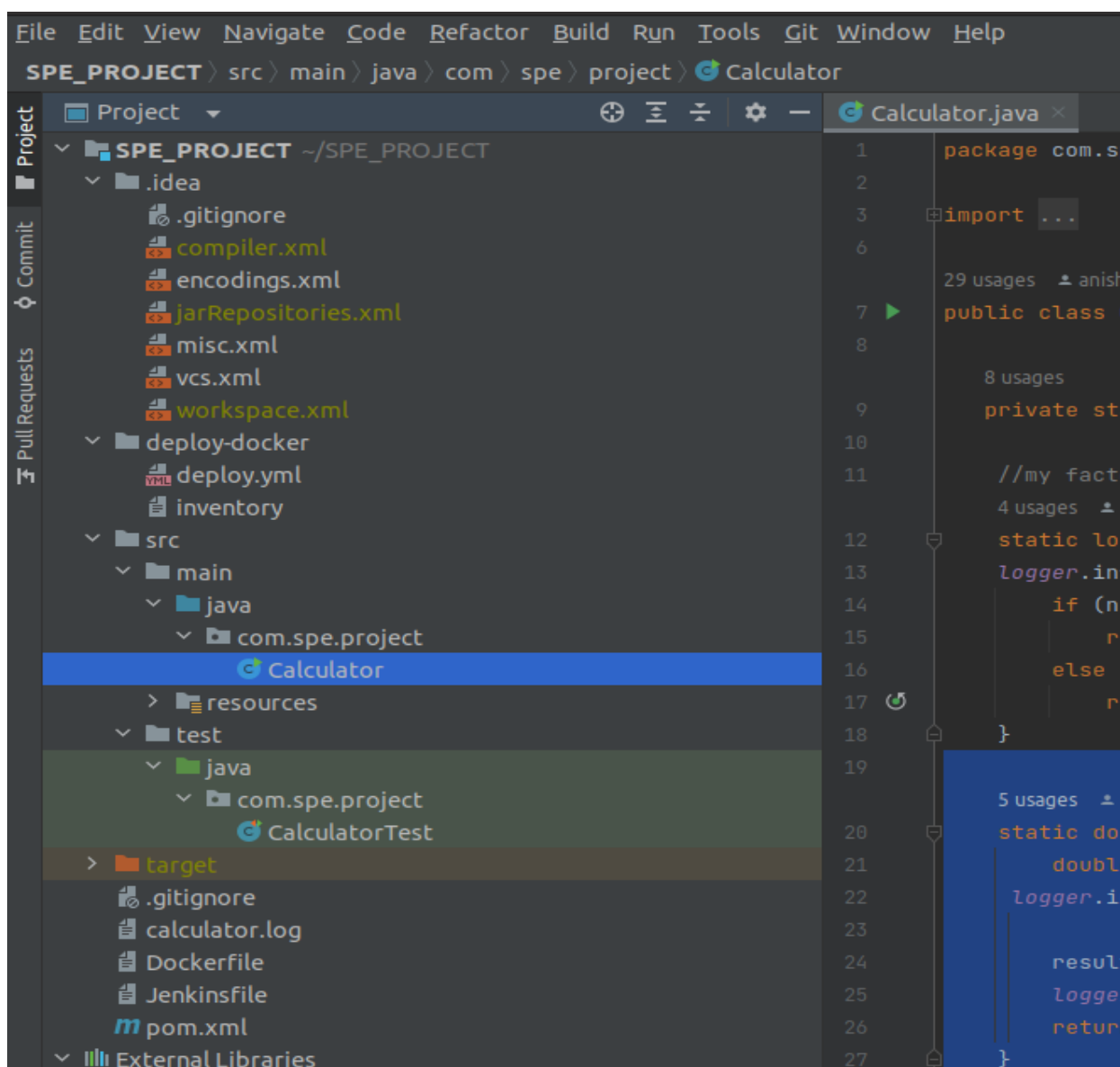
describes the working of maven. It begins with reading the pom.xml file, which mentions the required dependencies and

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.

```
8
no usages  ▲ anisha
9 ► public class CalculatorTest {
10
11     10 usages
    private Calculator objCalcUnderTest;
12     9 usages
    private static final double DELTA = 1e-15;
13     no usages  ▲ anisha
    @Before
14     public void setUp() {
15         //Arrange
16         objCalcUnderTest = new Calculator();
17     }
18
19     no usages  ▲ anisha
    @Test
20     public void PowerTruePositive() {
21         int x = 2; int b = 3;
22         double expectedResult = 8.0;
23         //Act
24         double result = objCalcUnderTest.x_power_b(x, b);
25         //Assert
26         assertEquals( message: "Finding factorial for true positive", expectedResult, result, DELTA);
27     }
28     no usages  ▲ anisha
    @Test
29     public void SquareTruePositive(){
30         assertEquals( message: "Finding square root of a number", expected: 4, objCalcUnderTest.square_root( n: 16), DELTA);
31         assertEquals( message: "Finding square root of a number", expected: 6, objCalcUnderTest.square_root( n: 36), DELTA);
32         assertEquals( message: "Finding square root of a number", expected: 8, objCalcUnderTest.square_root( n: 64), DELTA);
33     }
34 }
```

```
27     <version>2.20.0</version>
28 </dependency>
29 <dependency>
30     <groupId>org.apache.logging.log4j</groupId>
31     <artifactId>log4j-api</artifactId>
32     <version>2.20.0</version>
33 </dependency>
34 <dependency>
35     <groupId>junit</groupId>
36     <artifactId>junit</artifactId>
37     <version>4.13.1</version>
38     <scope>test</scope>
39 </dependency>
40 </dependencies>
41
42 <build>
43     <plugins>
44         <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.

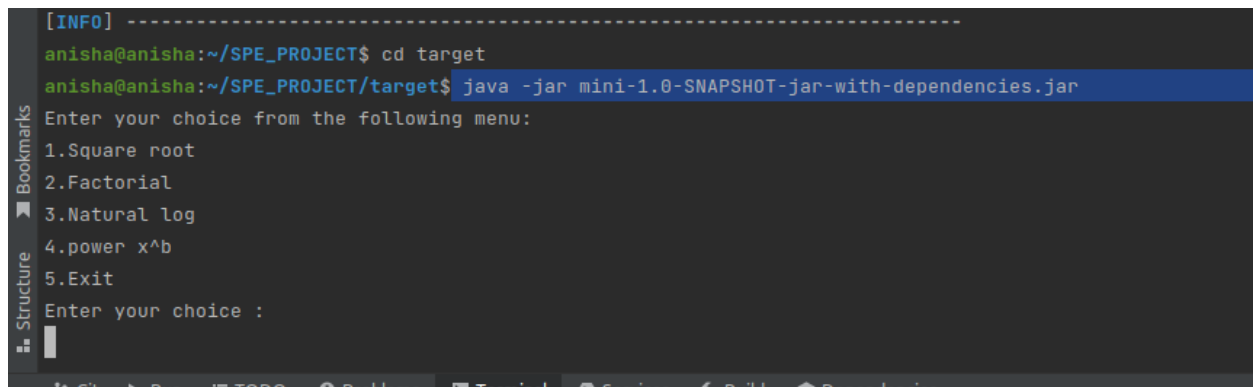
```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-assembly-plugin</artifactId>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
      <configuration>
        <archive>
          <manifest>
            <mainClass>com.spe.project.Calculator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

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:

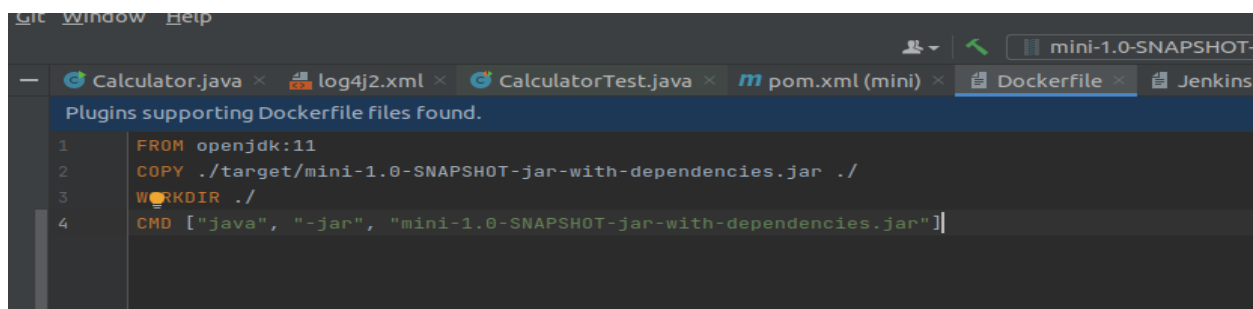


```
[INFO] -----
anisha@anisha:~/SPE_PROJECT$ cd target
anisha@anisha:~/SPE_PROJECT/target$ java -jar mini-1.0-SNAPSHOT-jar-with-dependencies.jar
Enter your choice from the following menu:
1.Square root
2.Factorial
3.Natural log
4.power x^b
5.Exit
Enter your choice :
```

## 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:



```
Calculator.java x log4j2.xml x CalculatorTest.java x pom.xml (mini) x Dockerfile x Jenkins
Plugins supporting Dockerfile Files Found.
1 FROM openjdk:11
2 COPY ./target/mini-1.0-SNAPSHOT-jar-with-dependencies.jar ./
3 WORKDIR ./
4 CMD ["java", "-jar", "mini-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

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


**Enter an item name**


» Required field


**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used


**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and type.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific bu

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a as long as they are in different folders.

**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

**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.

The screenshot shows the GitHub 'Webhooks / Manage webhook' interface. On the left is a sidebar with navigation links: General, Access (Collaborators, Moderation options), Code and automation (Branches, Tags, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables), and Integrations (GitHub apps, Email notifications). The 'Webhooks' link is highlighted. The main content area has two tabs: 'Settings' (selected) and 'Recent Deliveries'. The 'Settings' tab contains the following fields: 'Payload URL' (https://43fc-103-156-19-229.in.ngrok.io/github-webhook/), 'Content type' (application/json), 'Secret' (a yellow box with a warning to change it if lost), 'SSL verification' (radio buttons for 'Enable SSL verification' (selected) and 'Disable (not recommended)'), and 'Which events would you like to trigger this webhook?' (radio buttons for 'Just the push event.' (selected), 'Send me everything.', and 'Let me select individual events.'). At the bottom, there is a checkbox for 'Active' (checked) and two buttons: 'Update webhook' (green) and 'Delete webhook' (red).

(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.

The screenshot shows the Jenkins Configuration page for a project named 'calculator'. The breadcrumb navigation is 'Dashboard > calculator > Configuration'. On the left, there is a 'Configure' section with three tabs: 'General' (selected), 'Advanced Project Options', and 'Pipeline'. The 'General' tab contains several checkboxes: 'Do not allow the pipeline to resume if the controller restarts', 'GitHub project', 'Delivery Pipeline configuration', 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'This project is parameterized', and 'Throttle builds'. Below these is the 'Build Triggers' section, which includes checkboxes for 'Build after other projects are built', 'Build periodically', 'Build whenever a SNAPSHOT dependency is built', 'GitHub hook trigger for GITScm polling' (which is checked), 'Poll SCM', 'Quiet period', and 'Trigger builds remotely (e.g., from scripts)'.

The screenshot shows the 'Add Repository' configuration page in Jenkins. It is titled 'Pipeline script from SCM'. Under the 'SCM' dropdown, 'Git' is selected. The 'Repositories' section contains a 'Repository URL' field with the value 'https://github.com/Anisha-sudo/calculator.git', a 'Credentials' dropdown set to '- none -', and an '+ Add' button. Below this is an 'Advanced' dropdown and an 'Add Repository' button. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field with the value '\*/main' and an 'Add Branch' button.

7. Provide the details of the 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:



**Jenkins Location**

Jenkins URL ?

System Admin e-mail address ?

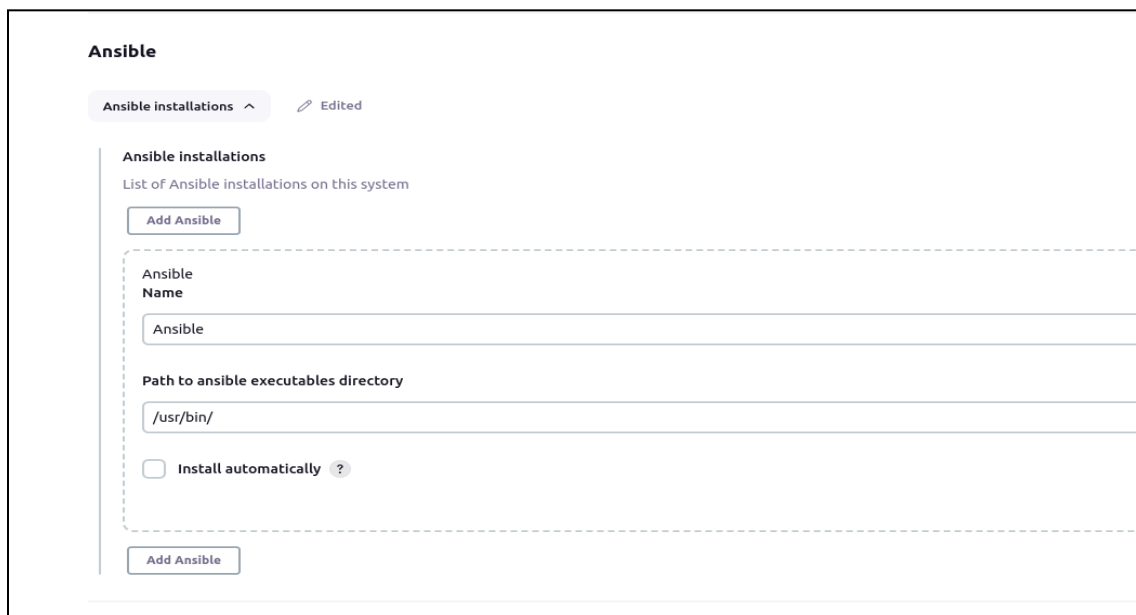
**Serve resource files from another domain**

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:



**Ansible**

Ansible installations ^ Edited

**Ansible installations**  
List of Ansible installations on this system

Ansible  
Name

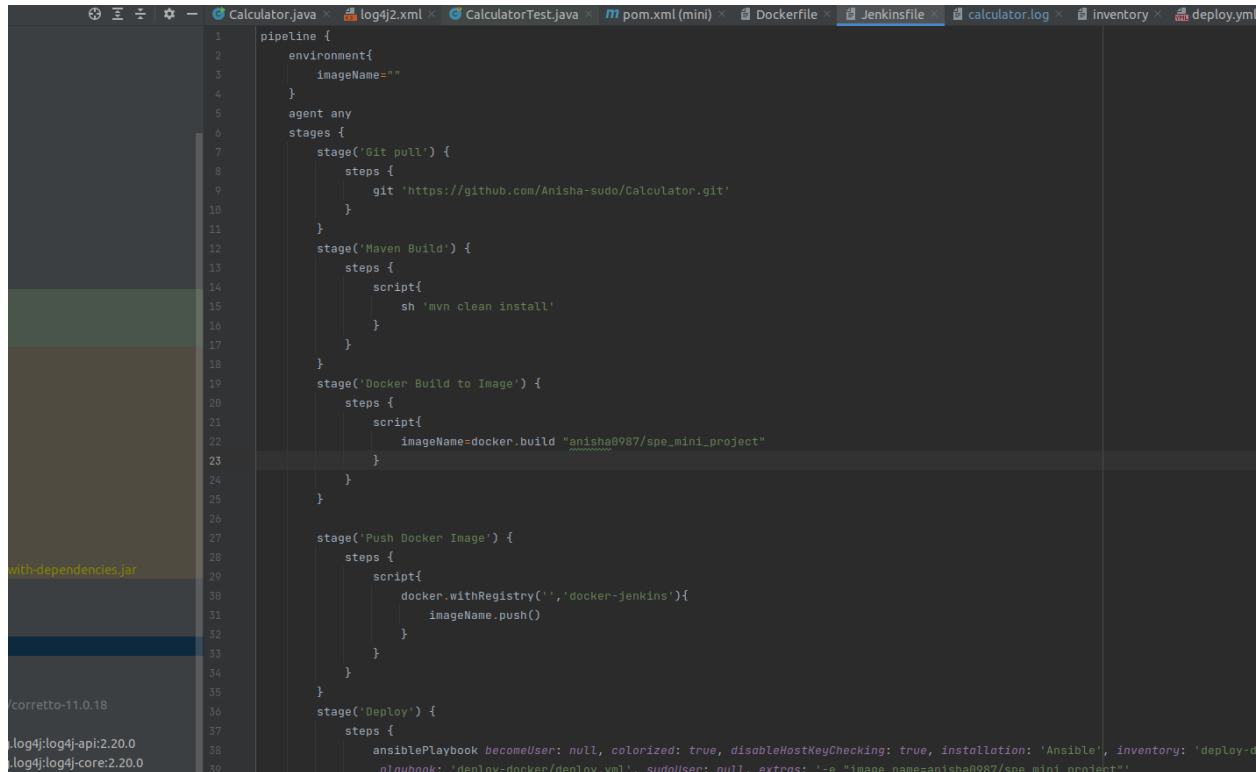
Path to ansible executables directory

☐ Install automatically ?

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.



```
1 pipeline {
2   environment{
3     imageName=""
4   }
5   agent any
6   stages {
7     stage('Git pull') {
8       steps {
9         git 'https://github.com/Anisha-sudo/Calculator.git'
10      }
11    }
12    stage('Maven Build') {
13      steps {
14        script{
15          sh 'mvn clean install'
16        }
17      }
18    }
19    stage('Docker Build to Image') {
20      steps {
21        script{
22          imageName=docker.build "anisha0987/spe_mini_project"
23        }
24      }
25    }
26
27    stage('Push Docker Image') {
28      steps {
29        script{
30          docker.withRegistry('', 'docker-jenkins'){
31            imageName.push()
32          }
33        }
34      }
35    }
36    stage('Deploy') {
37      steps {
38        ansiblePlaybook becomeUser: null, colonized: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'deploy-
39        playbook: 'deploy-docker/deploy.yml', sudoUser: null, extras: '-e "image_name=anisha0987/spe_mini_project"'
40      }
41    }
42  }
43 }
```

## 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”)

The screenshot shows the Jenkins 'Update credentials' page. The breadcrumb trail at the top is: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > anisha0987/\*\*\*\*\*. On the left, there are three buttons: 'Update' (with a key icon), 'Delete' (with a trash icon), and 'Move' (with a double arrow icon). The main form is titled 'Update credentials'. It contains the following fields: 'Scope' is a dropdown menu set to 'Global (Jenkins, nodes, items, all child items, etc)'; 'Username' is a text input field containing 'anisha0987'; there is an unchecked checkbox labeled 'Treat username as secret'; 'Password' is a text input field with a lock icon and the text 'Concealed', and a 'Change Password' button to its right; 'ID' is a text input field containing 'docker-jenkins'; and 'Description' is an empty text input field. At the bottom of the form is a blue 'Save' button.

## 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).

```
35     }
36     stage('Deploy') {
37         steps {
38             ansiblePlaybook becomeUser: null, colored: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'deploy-docker/inventory',
39                 playbook: 'deploy-docker/deploy.yml', sudoUser: null, extras: '-e "image_name=anisha0987/spe_mini_project"'
40         }
41     }
42 }
```

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

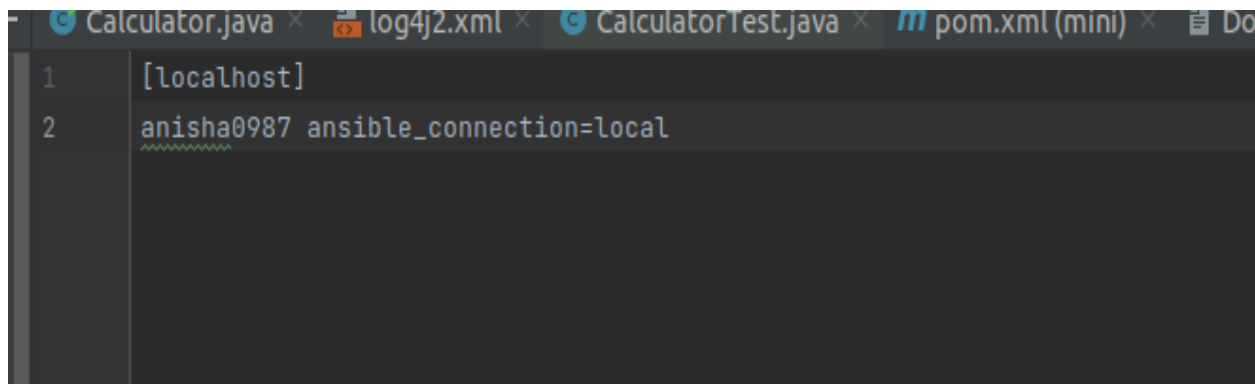
```
1  ---
2  - name: Pull docker image of Calculator
3    hosts: all
4    vars:
5      ansible_python_interpreter: /usr/bin/python3
6    tasks:
7
8      - name: Start docker service
9        service:
10         name: docker
11         state: started
12
13      - name: pull docker image
14        shell: docker pull docker.io/anisha0987/spe_mini_project
15
16      - name: Get container info
17        docker_container_info:
18         name: calculator
19        register: result
20
21      - name: Remove container if exists
22        shell: docker stop calculator && docker rm calculator
23        when: result.exists
24
25      - name: running container
26        shell: docker run -it --name calculator -d anisha0987/spe_mini_project
27
```

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:



```
1 [localhost]
2 anisha0987 ansible_connection=local
```

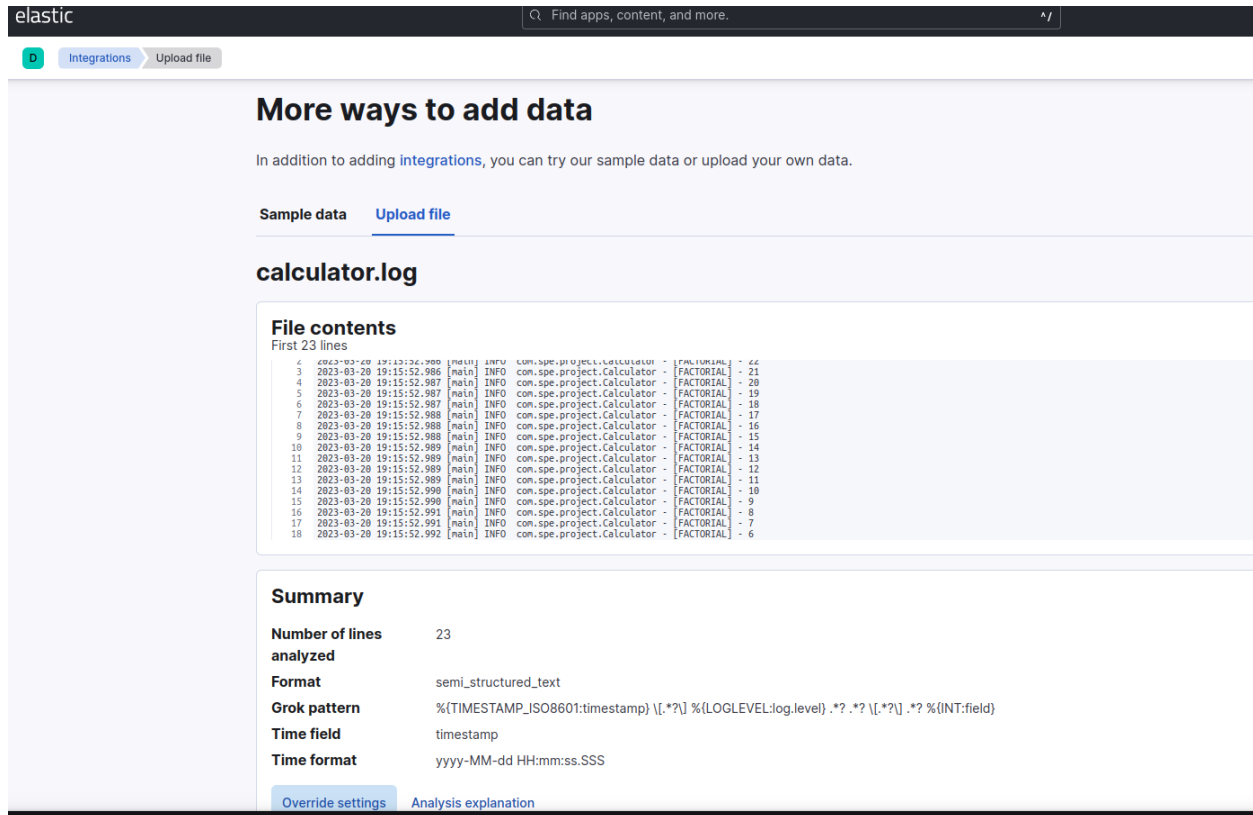
An inventory defines a collection of hosts that ansible will manage. These hosts can also be assigned to groups, which can be managed collectively. Groups can contain child groups, and hosts can be members of multiple groups.

## 4.6 Continuous Monitoring:

Tool: ELK - ElasticSearch, Logstash, Kibana

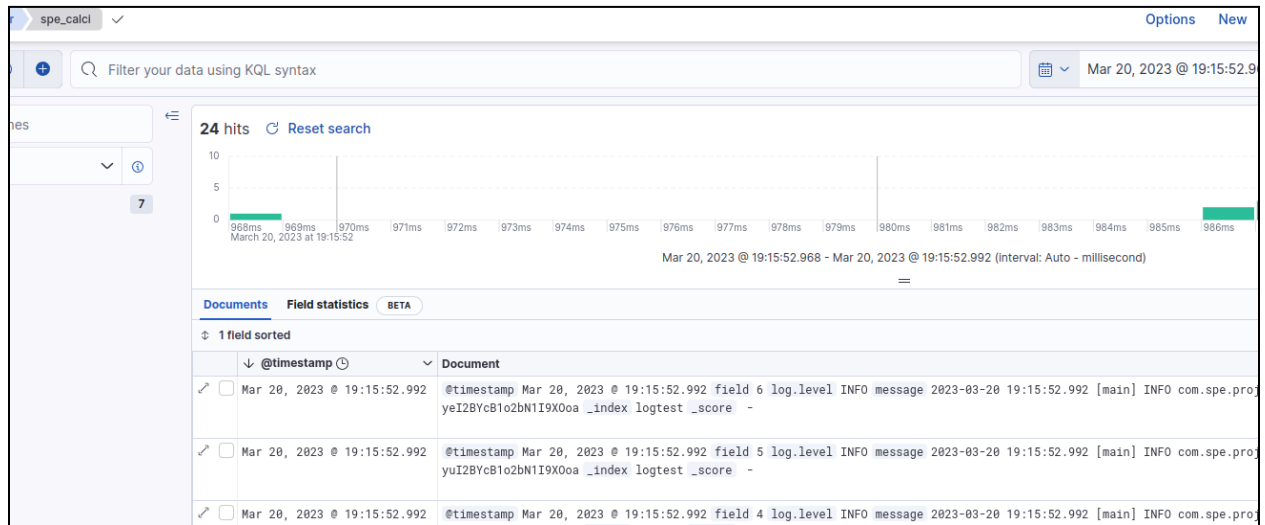
The ELK Stack helps by providing users with a powerful platform that collects and processes data from multiple data sources, stores that data in one centralized data store that can scale as data grows, and that provides a set of tools to analyze the data.

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:

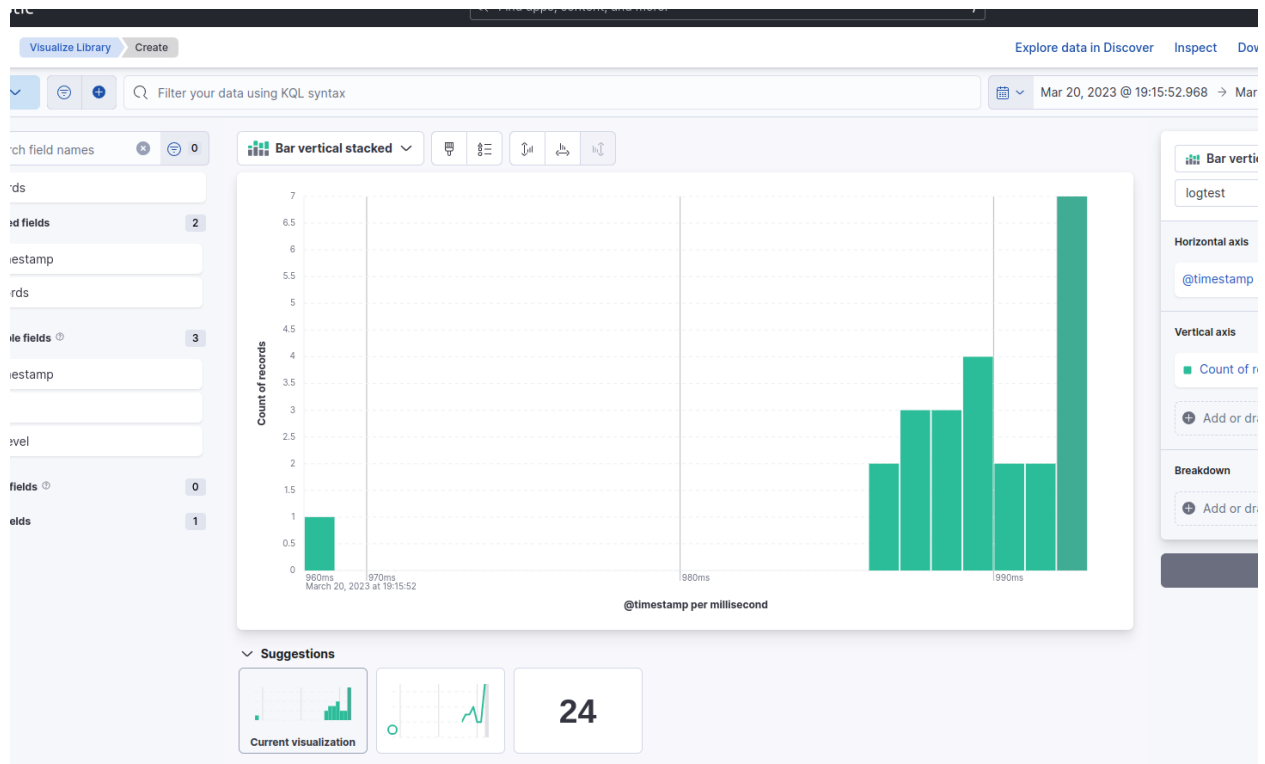


## 5.OUTPUT:

### 5.1 jenkins pipeline output:



## 5.2 Visualization output:



## 5.3 Calculator output:

```
anisha@anisha:~$ 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
```

```
Enter your choice from the following menu:
1.Square root
2.Factorial
3.Natural log
4.power x^b
5.Exit
Enter your choice :
4
Enter the first number->x
2
Enter the second number->b
4
17:59:38.808 [main] INFO    com.spe.project.Calculator - [POWER of2RAISED TO]4
17:59:38.808 [main] INFO    com.spe.project.Calculator - [RESULT - POWER] - 16.0
2 to the power 4 is 16.0
Do you want to continue?
1.Yes 2.No
```

**We have successfully implemented the calculator using  
DEVOPS.**

**Thankyou  
Anisha Rani  
MT2022153**