

# Scientific Calculator with DevOps

Siva Hitesh

IMT2019041

[kartikeya.sivahitesh@iiitb.ac.in](mailto:kartikeya.sivahitesh@iiitb.ac.in)

## 1. Problem Statement

We need to create a scientific calculator program with user menu driven operations such as

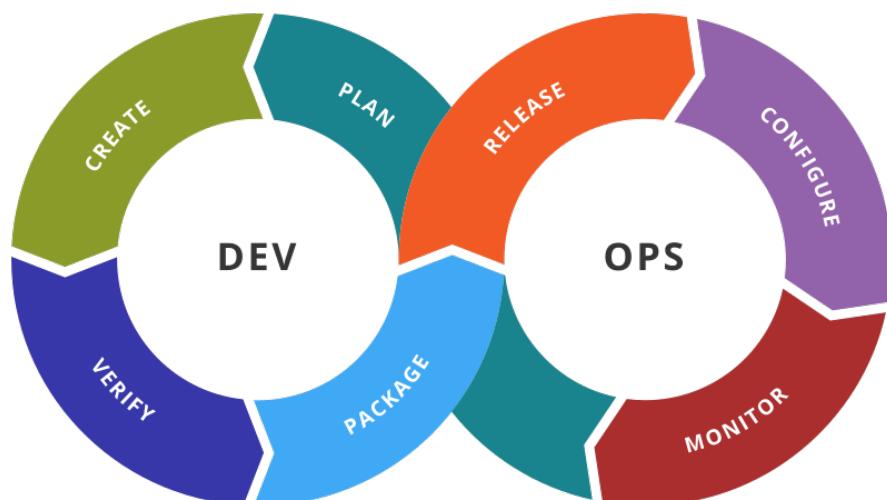
- Square root function -  $\sqrt{x}$
- Factorial function -  $x!$
- Natural logarithm (base e) -  $\ln(x)$
- Power function -  $x^y$

We need to build the calculator with DevOps principles using any set of DevOps tool chain, but the pipeline should be same.

## 2. DevOps

### 2.1. What is DevOps?

DevOps is a collaborative method for creating and delivering software that unites teams from the development and operations departments to increase the efficiency and effectiveness of software releases. The software development lifecycle is streamlined using automation, continuous integration, and delivery (CI/CD), and agile approaches. The DevOps methodology places a strong emphasis on sharing responsibility, communication, and collaboration among all parties involved, including business teams, operations personnel, and developers. DevOps aims to produce high-quality software more quickly and reliably while cutting costs and risks.



## 2.2. Why DevOps?

Due to the increased demand for quicker, more reliable software delivery, DevOps has grown in importance in recent years. Long development cycles, manual testing, and siloed teams that operate independently from one another are frequent features of traditional software development processes. Delays, mistakes, and a lack of accountability may result from this.

By dismantling silos and promoting cooperation between the development and operations teams, DevOps overcomes these issues. DevOps enables teams to create software faster, more reliably, and with fewer errors by automating repetitive operations and optimizing the software development lifecycle. As a result, businesses can adapt to shifting market demands more rapidly, spend less money, and satisfy customers better.

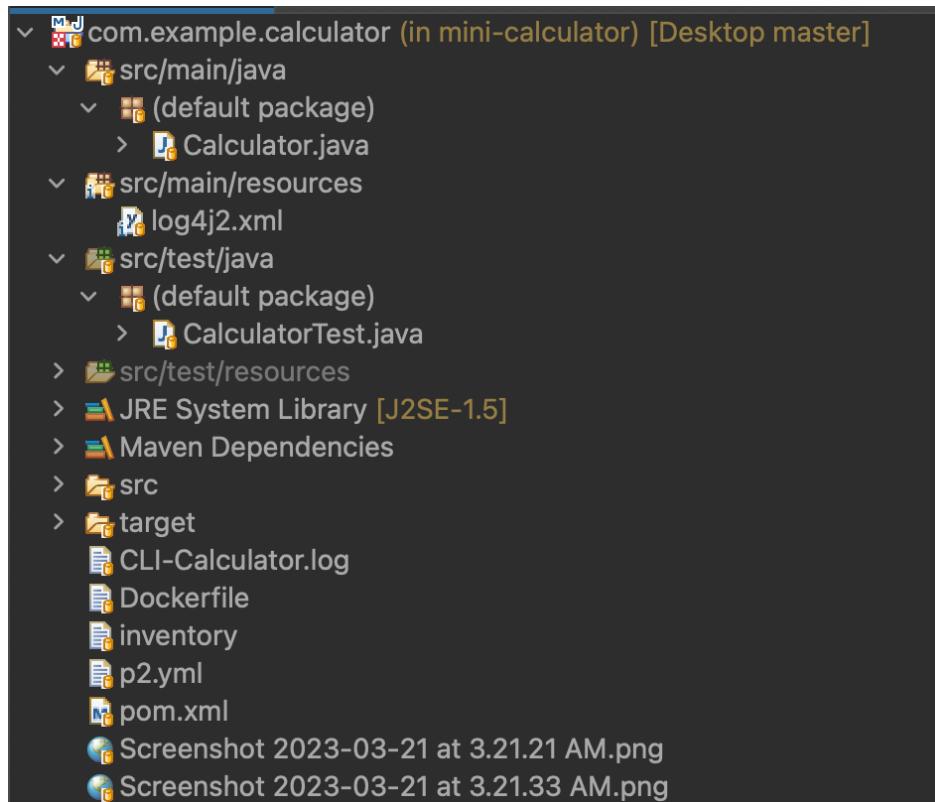
## 3. Tools Used

- **Git:** Used for version control.
- **GitHub:** Helps in source control management.
- **JUnit:** Used to test the code.
- **Maven:** Used to add dependencies and build the code as a jar file.
- **Jenkins:** Used for Continuous integration.
- **Docker:** Used for containerizing the code.
- **Docker Hub:** Used to host the docker images of the code.
- **Ansible:** Used to automate configuration management and deployment.
- **Elastic stack:** Used to monitoring. We generate a log file using log4j2 and use it for monitoring.
- **Ngrok and GitHub WebHooks:** Used to start the build process automatically whenever there is a new commit to the GitHub.

## 4. Steps

### 4.1. Developing, building, and testing the Calculator code

Firstly, install Java SE 8, maven, and Eclipse IDE and create a Maven project in the IDE. Add the dependencies JUnit, and log4j2 to the *pom.xml* file as we need them for testing and logging. Also add the compiler version to the *pom.xml* file to avoid errors in future steps. The images below show the same.



The project directory structure

```
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

Compiler version

```

<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.14.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.14.0</version>
    </dependency>
</dependencies>

```

Dependencies

```

<build>
    <plugins>
        <plugin>
            <!-- Build an executable JAR --> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-jar-plugin</artifactId>
            <configuration>
                <archive> <manifest>
                    <mainClass>Calculator</mainClass> </manifest>
                </archive> </configuration>
            </plugin>

            <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>calculator.Calculator</mainClass>
                                </manifest>
                            </archive>
                            <descriptorRefs>
                                <descriptorRef>jar-with-dependencies</descriptorRef>
                            </descriptorRefs>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
    </plugins>
</build>

```

To build the project as a jar file

The *Calculator.java* file contains the main functionality code. It also contains the logging statements which are used while generating log file which in turn is used for monitoring.

```

public double factorial(double val)
{
    double ans = 1;

    if(val <0)
    {
        logger.error("[FACTORIAL] - " + val);
        logger.error("[RESULT - FACTORIAL] - " + "null");
        ans = -1;
    }

    else
    {
        for(int i=1; i<=val; i++)
            ans *= i;
        logger.info("[FACTORIAL] - " + val);
        logger.info("[RESULT - FACTORIAL] - " + ans);
    }

    return ans;
}

public double logarithm(double val)
{
    double ans;
    if(val <=0) {
        ans = -1;
        logger.error("[LOGARITHM] - " + val);
        logger.error("[RESULT - LOGARITHM] - " + "null");
    }

    else
    {
        ans = Math.log(val);
        logger.info("[LOGARITHM] - " + val);
        logger.info("[RESULT - LOGARITHM] - " + ans);
    }

    return ans;
}

```

The functions for factorial and natural logarithm

The *CalculatorTest.java* contains the test cases used for testing purposes. It has both TruePositive and FalsePositive test cases for all the 4 functions. The *log4j2.xml* is used for generating the log file.

```

@Test
public void factorialTruePositive(){
    assertEquals("Finding factorial of a number for True positive",2,calc.factorial(2), delta);
    assertEquals("Finding factorial of a number for True positive",120,calc.factorial(5), delta);
    assertEquals("Finding factorial of a number for True positive",40320,calc.factorial(8), delta);
    assertEquals("Finding factorial of a number for True positive",720,calc.factorial(6), delta);
}

@Test
public void factorialFalsePositive(){
    assertNotEquals("Finding factorial of a number for False Positive",100,calc.factorial(5),delta);
    assertNotEquals("Finding factorial of a number for False positive",47,calc.factorial(4),delta);
    assertNotEquals("Finding factorial of a number for False positive",18,calc.factorial(6),delta);
}

@Test
public void logarithmTruePositive(){
    assertEquals("Finding Natural Logarithm of a number for True positive",Math.log(7),calc.logarithm(7), delta);
    assertEquals("Finding Natural Logarithm of a number for True positive",Math.log(4.7),calc.logarithm(4.7), delta);
    assertEquals("Finding Natural Logarithm of a number for True positive",Math.log(16),calc.logarithm(16), delta);
}

@Test
public void logarithmFalsePositive(){
    assertNotEquals("Finding Natural Logarithm of a number for False positive",5,calc.logarithm(7), delta);
    assertNotEquals("Finding Natural Logarithm of a number for False positive",3,calc.logarithm(5), delta);
    assertNotEquals("Finding Natural Logarithm of a number for False positive",2,calc.logarithm(2), delta);
}

```

Test cases for the 4 functions.

After writing the code, use `mvn clean install` command to build the code. It also tests the code using the written test cases. This will create a new folder “target” and .jar file inside it.

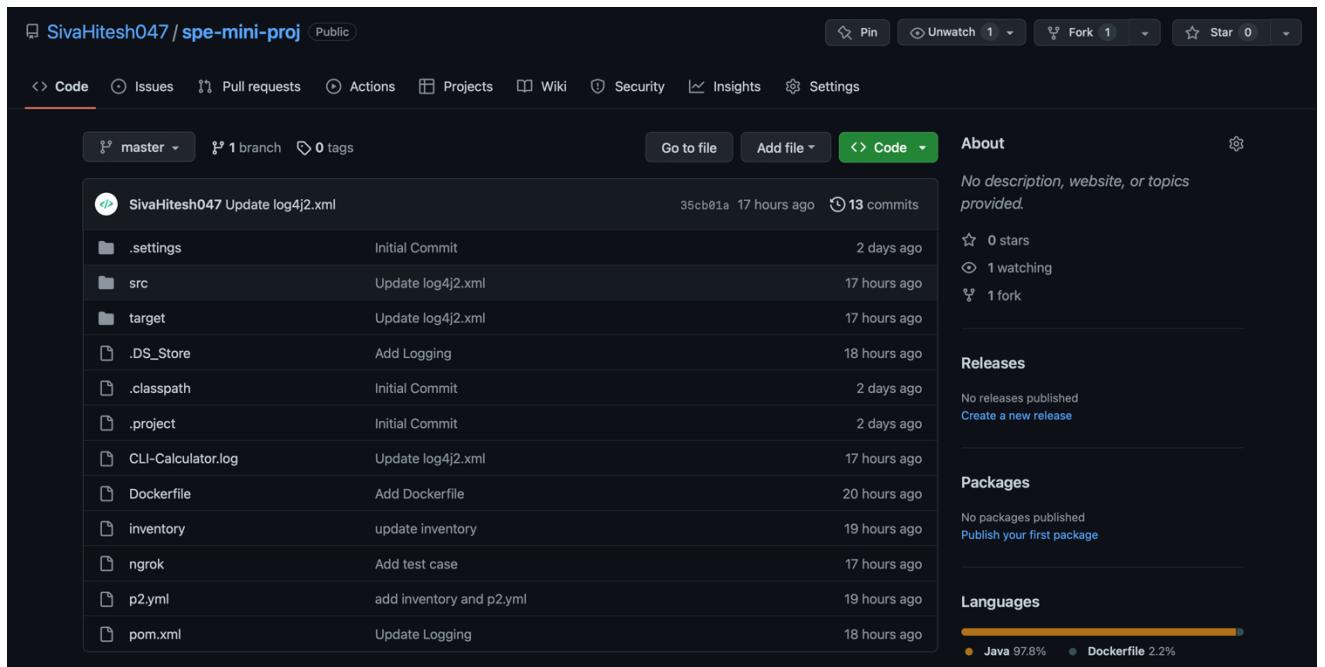
```
[INFO] [INFO] --- maven-install-plugin:2.4:install (default-install) @ com.example.calculator ---
[INFO] Installing /Users/sivahitesh/Desktop/mini-calculator/target/com.example.calculator-0.0.1-SNAPSHOT.jar to /Users/sivahitesh/.m2/repository/com/example/calculator/com.example.calculator/0.0.1-SNAPSHOT/com.example.calculator-0.0.1-SNAPSHOT.jar
[INFO] Installing /Users/sivahitesh/Desktop/mini-calculator/pom.xml to /Users/sivahitesh/.m2/repository/com/example/calculator/com.example.calculator/0.0.1-SNAPSHOT/com.example.calculator-0.0.1-SNAPSHOT.pom
[INFO] Installing /Users/sivahitesh/Desktop/mini-calculator/target/com.example.calculator-0.0.1-SNAPSHOT-jar-with-dependencies.jar to /Users/sivahitesh/.m2/repository/com/example/calculator/com.example.calculator/0.0.1-SNAPSHOT/com.example.calculator-0.0.1-SNAPSHOT-jar-with-dependencies.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.121 s
[INFO] Finished at: 2023-03-21T20:13:14+05:30
[INFO] -----
```

*mvn clean install output*

## 4.2. Pushing the code to GitHub

Create a Personal access token in GitHub from the *Developer Settings*. Then, create a repository in GitHub and push the code to that repository from the local folder using the following commands:

- \$ Git init
- \$ git add .
- \$ git commit -m “Commit Message.”
- \$ git push  
[https://\[USERNAME\]:\[ACCESTOKEN\]@github.com/\[USERNAME\]/\[REPO\].git](https://[USERNAME]:[ACCESTOKEN]@github.com/[USERNAME]/[REPO].git)



*GitHub Repository*

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

---

**Owner \***      **Repository name \***

SivaHitesh047 / spe-mini-project ✓

Great repository names are simple! spe-mini-project is available. Inspiration? How about [curly-octo-invention](#)?

**Description (optional)**

---

**Public**  
Anyone on the internet can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

**Add a README file**  
This is where you can write a long description for your project. [Learn more](#).

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: None ▾

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more](#).

License: None ▾

---

ⓘ You are creating a public repository in your personal account.

**Create repository**

## Creating a Repo

master			
-o- Commits on Mar 21, 2023			
<a href="#">Update log4j2.xml</a> SivaHitesh047 committed 17 hours ago			
	35cb01a	🔗	📋
<a href="#">Update log4j2.xml</a> SivaHitesh047 committed 17 hours ago			
	125fdcc	🔗	📋
<a href="#">Add test case</a> SivaHitesh047 committed 17 hours ago			
	4c6233c	🔗	📋
<a href="#">Add test case</a> SivaHitesh047 committed 17 hours ago			
	92b80c3	🔗	📋
<a href="#">Add Logging</a> SivaHitesh047 committed 18 hours ago			
	2a423a6	🔗	📋
<a href="#">Merge branch 'master' of https://github.com/SivaHitesh047/spe-mini-proj</a> SivaHitesh047 committed 18 hours ago			
	05f941e	🔗	📋
<a href="#">Update Logging</a> SivaHitesh047 committed 18 hours ago			
	f877bea	🔗	📋
<a href="#">update inventory</a> SivaHitesh047 committed 19 hours ago			
	Verified	2b6bf63	🔗
<a href="#">add inventory and p2.yml</a> SivaHitesh047 committed 19 hours ago			
	c9c12c9	🔗	📋
<a href="#">Add Dockerfile</a> SivaHitesh047 committed 20 hours ago			
	e251236	🔗	📋
-o- Commits on Mar 20, 2023			
<a href="#">Update pom.xml</a> SivaHitesh047 committed yesterday			
	95cd876	🔗	📋
<a href="#">Update pom.xml</a> SivaHitesh047 committed yesterday			
	d9d9dd0	🔗	📋
-o- Commits on Mar 19, 2023			

## The Commit messages.

## 4.3. Creating the pipeline in Jenkins

Jenkins is a continuous integration/continuous delivery and deployment (CI/CD) automation software. It is a very useful DevOps tool and is written in Java language. Install the Jenkins via *homebrew* package using the following commands:

- Install the latest LTS version: brew install jenkins-lts
- Install a specific LTS version: brew install jenkins-lts@YOUR\_VERSION
- Start the Jenkins service: brew services start jenkins-lts
- Restart the Jenkins service: brew services restart jenkins-lts
- Update the Jenkins version: brew upgrade jenkins-lts

After starting the Jenkins service, go to <http://localhost:8080> and finish the setup. Install the following plugins that are required for the project in **Manage Plugins**:

- Git
- Maven
- Docker
- Ansible

We need to configure these in **Global tool Configuration**. The path given should be the same as the path where the tool is located in the local system. It can be found using `$whereis mvn` command.



Docker

Docker installations ▾ Edited

Docker installations

List of Docker installations on this system

Add Docker

Docker Name	<input type="text" value="Docker"/>	
Installation root ?	<input type="text" value="/usr/local/bin/"/>	
<input type="checkbox"/> Install automatically ?		

Add Docker

For Jenkins to access the repositories, we give the credentials in **Manage Credentials**.

#### Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	95750270-14c3-4f0d-b450-f7b091875fa9	shoyohinata2141@gmail.com/*****
		System	(global)	2c26fb82-0374-43b3-b44a-25a47f02920a	SivaHitesh047/*****
		System	(global)	docker_credentials	sivahitesh/***** (credentials for docker)

#### Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Icon: S M L

Now, Create a pipeline project from the Jenkins dashboard.

Dashboard >

+ New Item Add description

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Lockable Resources

**Build Queue** No builds in the queue.

**Build Executor Status** 1 Idle 2 Idle

S	W	Name ↓	Last Success	Last Failure	Last Duration
		disk-information	2 mo 2 days #1	N/A	1 sec
		Jenkins-demo	1 yr 3 mo #16	1 yr 3 mo #11	10 sec
		labdemo	N/A	N/A	N/A
		maven-jenkins	N/A	N/A	N/A
		mini-calculator	9 hr 19 min #23	19 hr #19	1 min 9 sec
		ram-information	2 mo 2 days #3	2 mo 2 days #2	0.74 sec
		spe-test	2 mo 2 days #2	N/A	0.65 sec
		test_proj	1 yr 4 mo #9	N/A	1.9 sec

Icon: S M L

Icon legend Atom feed for all Atom feed for failures Atom feed for just latest builds

Jenkins Dashboard

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 for something other than software build.
-  **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/or organizing complex activities that do not easily fit in free-style job type.
-  **Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

## Creating Pipeline Project

### 4.3.1. The Pipeline

- **Git Pull:** This stage pulls the repository from GitHub.

```
stage('Git Pull stage') {
    steps {
        // Get some code from a GitHub repository
        git 'https://github.com/SivaHitesh047/spe-mini-proj.git'
    }
}
```

- **Maven Build:** This stage builds the project again and generates a new jar file deleting the existing one.

```
environment{
    PATH = "/Users/sivahitesh/Downloads/miscellaneous/apache-maven-3.8.6/bin:$PATH"
}
agent any

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

- **Docker Build Image:** This stage is used to build the docker image of the project.

```
stage('Docker Build Image')
{
    steps{
        script{
            imageName = docker.build "sivahitesh/scientific-calculator:latest"
        }
    }
}
```

- **Push Docker Image:** In this stage, we deploy the existing docker image to the DockerHub repository.

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

- **Ansible Deployment:** This is used to pull the docker image from the DockerHub and deploy it on every machine mentioned in Inventory file.

```
stage('Ansible pull docker image')
{
    steps{
        sh "/usr/bin/pip3 install docker"
        sh "/usr/bin/pip3 install requests"
        sh "ansible-playbook p2.yml -i inventory"
    }
}
```



## 4.4. Containerization

We use docker for Containerization. Docker is a platform for developing, packaging, and deploying applications in a containerized environment. It packages software and all its dependencies into a portable unit, making it easy to move between different environments.

We need to create a docker image and push it to the DockerHub. Create a file named Dockerfile. In this, we'll use the jar file to create the image.

```
FROM openjdk:8
COPY ./target/com.example.calculator-0.0.1-SNAPSHOT.jar .
WORKDIR .
CMD ["java", "-cp", "com.example.calculator-0.0.1-SNAPSHOT.jar", "Calculator"]
```

- FROM imports the base image of jdk8 for creating a new image.
- COPY copies the jar file into the image in its root directory.
- WORKDIR changes the current working directory.
- CMD runs the command inside the image.

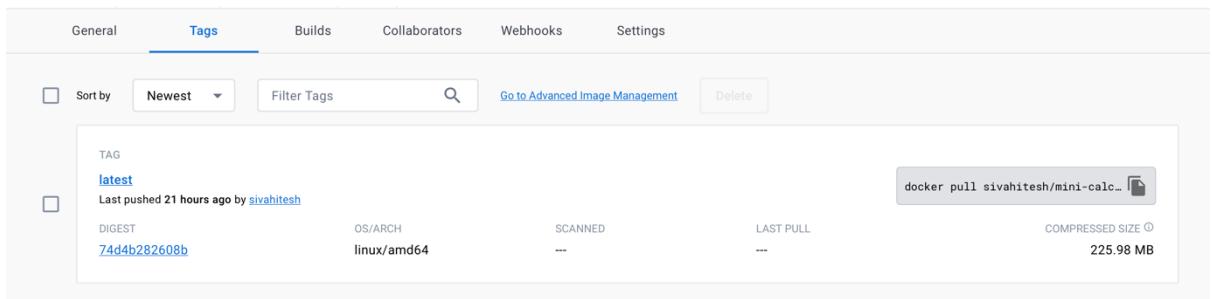


Image in the DockerHub

DockerHub Repository

## 4.5. Deployment

We use Ansible tool for deployment. Ansible is an automation tool used for configuration management, application deployment, and task automation. It uses a simple syntax called YAML to define tasks and playbooks that can be executed on multiple servers simultaneously.

Install ansible using command `$brew install ansible`.

```
> sivahitesh > Desktop > mini-calculator > inventory
localhost ansible_user=sivahitesh ansible_connection=local ansible_python_interpreter=/usr/bin/python3
```

Inventory file

```

---  

- name: pull mini_project image  

  hosts: all  

  tasks:  

    - name: Pull devops image  

      docker_image:  

        name: sivahitesh/mini-calculator  

        source: pull  

    - name: creating updated container  

      shell: docker run -id sivahitesh/mini-calculator

```

The Playbook p2.yml.

## 4.6. Monitoring using ELK stack

Monitoring should be done once the deployment. It must be done to measure the performance of the various components of a software system to ensure that it is functioning correctly and efficiently.

The ELK stack is an open-source toolset for collecting, analyzing, and visualizing large volumes of data in real-time. It consists of three main components: Elasticsearch, Logstash, and Kibana. Together, these tools provide a powerful solution for managing and analyzing log data.

```
(base) sivahitesh@Siva-Hiteshs-MacBook-Pro mini-calculator % cat CLI-Calculator.log
21/03/2023:03:13:47 716 [Calculator.java] [INFO] Calculator [FACTORIAL] - 5.0
21/03/2023:03:13:47 721 [Calculator.java] [INFO] Calculator [RESULT - FACTORIAL] - 120.0
21/03/2023:03:13:47 722 [Calculator.java] [INFO] Calculator [FACTORIAL] - 4.0
21/03/2023:03:13:47 722 [Calculator.java] [INFO] Calculator [RESULT - FACTORIAL] - 24.0
21/03/2023:03:13:47 722 [Calculator.java] [INFO] Calculator [FACTORIAL] - 6.0
21/03/2023:03:13:47 723 [Calculator.java] [INFO] Calculator [RESULT - FACTORIAL] - 720.0
21/03/2023:03:13:47 723 [Calculator.java] [INFO] Calculator [SQUARE ROOT] - 25.0
21/03/2023:03:13:47 724 [Calculator.java] [INFO] Calculator [RESULT - SQUARE ROOT] - 5.0
21/03/2023:03:13:47 724 [Calculator.java] [INFO] Calculator [SQUARE ROOT] - 16.0
21/03/2023:03:13:47 724 [Calculator.java] [INFO] Calculator [RESULT - SQUARE ROOT] - 4.0
21/03/2023:03:13:47 725 [Calculator.java] [INFO] Calculator [SQUARE ROOT] - 169.0
21/03/2023:03:13:47 725 [Calculator.java] [INFO] Calculator [RESULT - SQUARE ROOT] - 13.0
21/03/2023:03:13:47 725 [Calculator.java] [INFO] Calculator [SQUARE ROOT] - 36.0
21/03/2023:03:13:47 725 [Calculator.java] [INFO] Calculator [RESULT - SQUARE ROOT] - 6.0
21/03/2023:03:13:47 726 [Calculator.java] [INFO] Calculator [LOGARITHM] - 7.0
21/03/2023:03:13:47 726 [Calculator.java] [INFO] Calculator [RESULT - LOGARITHM] - 1.9459101490553132
21/03/2023:03:13:47 727 [Calculator.java] [INFO] Calculator [LOGARITHM] - 4.7
21/03/2023:03:13:47 727 [Calculator.java] [INFO] Calculator [RESULT - LOGARITHM] - 1.547562508716013
21/03/2023:03:13:47 727 [Calculator.java] [INFO] Calculator [LOGARITHM] - 16.0
21/03/2023:03:13:47 727 [Calculator.java] [INFO] Calculator [RESULT - LOGARITHM] - 2.772588722239781
21/03/2023:03:13:47 728 [Calculator.java] [INFO] Calculator [SQUARE ROOT] - 10.0
21/03/2023:03:13:47 728 [Calculator.java] [INFO] Calculator [RESULT - SQUARE ROOT] - 3.1622776601683795
21/03/2023:03:13:47 729 [Calculator.java] [INFO] Calculator [SQUARE ROOT] - 25.0
21/03/2023:03:13:47 729 [Calculator.java] [INFO] Calculator [RESULT - SQUARE ROOT] - 5.0
21/03/2023:03:13:47 729 [Calculator.java] [INFO] Calculator [SQUARE ROOT] - 36.0
21/03/2023:03:13:47 729 [Calculator.java] [INFO] Calculator [RESULT - SQUARE ROOT] - 6.0
21/03/2023:03:13:47 730 [Calculator.java] [INFO] Calculator [FACTORIAL] - 2.0

```

Log file

elastic

Cloud

### Elasticsearch Service

Deployment name	Status	Version	Cloud region	Manage deployment
spe-mini-proj	Healthy	8.6.2	GCP - Iowa (us-central1)	

### Documentation

Help me find...

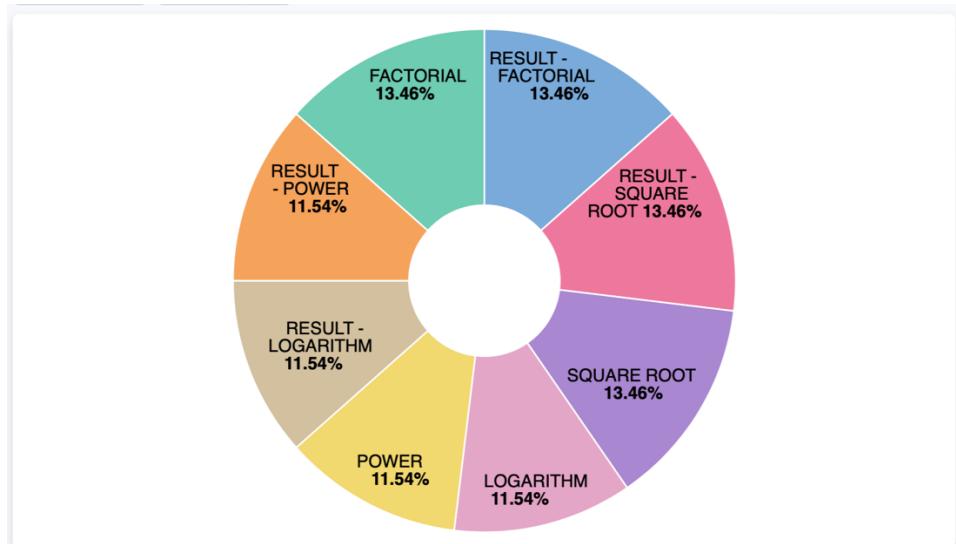
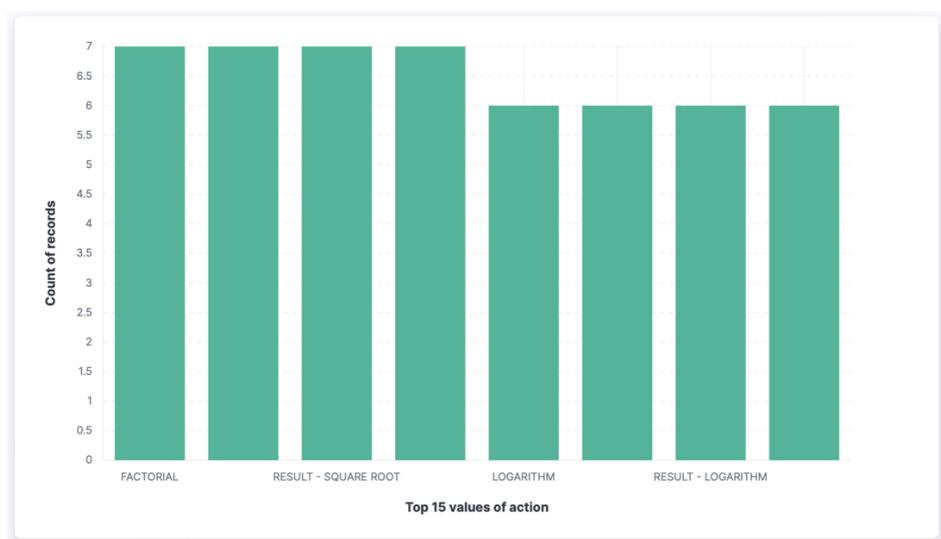
Elastic documentation  
Indexing data into Elasticsearch  
Elasticsearch REST API

### Community

Join an ElasticON event  
Hear success stories, lessons learned, tips, tricks, best practices, and funny anecdotes from Elastic...

Kibana Workshop  
MARCH 22, 14:30

Simplify multi-cloud monitoring and insights with Elastic Observability



## 4.7. WebHooks

Webhooks are automated messages that are sent from a web application to a designated URL when a specific event occurs. The Webhook will build the Jenkins pipeline automatically if we make any new commit to the GitHub repo.

Ngrok creates a secure tunnel between the developer's machine and the internet, which allows external users to access the web server through a unique URL generated by Ngrok. It is commonly used for testing webhooks, APIs, and other web services locally before deploying them to a production environment.

- Install ngrok using `$ brew install ngrok/ngrok/ngrok`
- Start it using `ngrok http 8080`;
- Go to the Webhooks in repository settings and add ngrok address in payload url and the personal access token in secret.

**Webhooks / Add webhook**

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 \***

**Content type**

**Secret**

SivaHitesh047 / spe-mini-proj Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

General Webhooks Add webhook

Access Collaborators Moderation options

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

https://3245-119-161-98-68.in.n... (push) Edit Delete

Code and automation

Branches Tags Actions Webhooks Environments Codespaces Pages

```

ngrok                                     (Ctrl+C to quit)

Add OAuth and webhook security to your ngrok (its free!): https://ngrok.com/free

Session Status      online
Account            Siva Hitesh (Plan: Free)
Update             update available (version 3.2.1, Ctrl-U to update)
Version            3.1.1
Region             India (in)
Latency            26ms
Web Interface     http://127.0.0.1:4040
Forwarding         https://3245-119-161-98-68.in.ngrok.io -> http://localhost:8080

Connections        ttl     opn     rt1     rt5     p50     p90
                    2       0       0.02   0.01   30.23  30.25

HTTP Requests
-----
POST /github-webhook/          200 OK
POST /github-webhook/          200 OK

```

## 5. Challenges

I got docker command not found while building the Jenkins pipeline and I had to add the following to /usr/local/Cellar/jenkins-lts/2.176.3/homebrew.mxcl.jenkins-lts.plist file:

```

<key>EnvironmentVariables</key>
<dict>
  <key>PATH</key>
  <string>/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Applications/Docker.app/Contents/Resources/bin:/Users/Kh0a/Library/Group\Containers/group.com.docker/Applications/Docker.app/Contents/Resources/bin</string>
</dict>

```

I also got mvn command not found and had to add Path to the environment in the pipeline script. Alongside doing these I needed to set the path in Global Tool Configuration for both Maven and docker.

## 6. Links

Link to the GitHub Repo:

<https://github.com/SivaHitesh047/spe-mini-proj.git>

Link to the DockerHub Repo:

<https://hub.docker.com/repository/docker/sivahitesh/scientific-calculator/general>