



Scientific Calculator with DevOps

Software Production Engineering Mini Project

Student: Divyam Agrawal

Roll number: IMT2019028

Course: CS 816

Report

Contents

1	Problem Statement	1
2	Implementation	1
3	DevOps	1
3.1	What is DevOps?	1
3.2	Why DevOps?	2
4	Tools Utilized	3
5	Development and Deployment Process	3
5.1	Code, Build and Test	4
5.2	Source Code Management	7
5.3	Continuous Integration / Continuous Deployment	9
5.3.1	Jenkins Plugins	9
5.3.2	Jenkins Pipeline	12
5.3.3	GitHub Webhook	17
5.4	Containerization	18
5.5	Deployment	20
5.6	Monitoring	22
5.7	Scientific Calculator Outputs	25

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. Implementation

- GitHub Repository: https://github.com/divyamagwl/Calculator_DevOps
- DockerHub: <https://hub.docker.com/repository/docker/divyamagwl/calculator>

3. DevOps

3.1 What is DevOps?

- DevOps is a cultural philosophy that emphasizes on team empowerment, cross-team communication and collaboration, and technology automation.
- It is a set of methods and tools that automate and integrate processes between software development team and operation teams, allowing enterprises to produce applications and services at high velocity and improve products faster. This allows companies to better serve their customers and compete in the market.
- DevOps is built around four core principles that govern the efficacy and efficiency of application development and deployment -
 1. **Automation of the software development lifecycle:** This involves automating testing, builds, releases, provisioning of development environments, and other manual operations that can slow down or bring human error into the software delivery process.
 2. **Collaboration and communication:** Collaboration and communication are essential to facilitate the rapid and continuous delivery of software, and to ensure that everyone is working towards the same goals.
 3. **Continuous improvement and minimization of waste:** DevOps teams are constantly looking for ways to enhance their processes, from automating repetitive operations to monitoring performance data for ways to minimise release delays or mean-time-to-recovery.
 4. **Hyperfocus on user needs with short feedback loops:** DevOps teams may take a step back and focus on what real users want and how to offer it to them through automation, greater communication and cooperation, and continuous improvement.

3.2 Why DevOps?

- **Maximizes Efficiency with Automation:** DevOps maximizes efficiency and streamlines software development and deployment processes through automation. This reduces human error and accelerates software delivery. Automation allows DevOps teams to focus on high-level tasks like improving software quality, collaborating with stakeholders, and continuous improvement.
- **Optimizes the Entire Business:** DevOps compels organizations to prioritize the optimization of the entire system, rather than just individual IT silos, in order to enhance the overall business. This involves being more adaptive and utilizing data-driven approaches to align with the needs of customers and the business as a whole.
- **Improves Speed and Stability of Software Development and Deployment:** By breaking down the traditional silos between software development and IT operations, DevOps teams can work collaboratively to streamline the software delivery pipeline. This leads to faster and more frequent releases, allowing organizations to respond quickly to changing market conditions and customer demands.

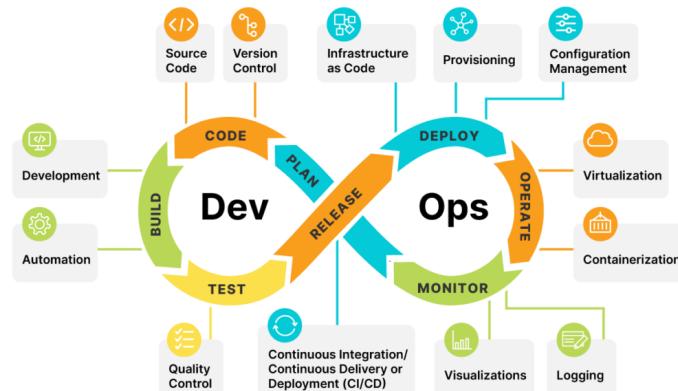


Figure 1: DevOps Cycle

Periodic Table of DevOps Tools	Element	Symbol	Type	Description
1 En O	En	O	Database	Oracle
3 Os My	Os	My	SCM	Mercurial
12c Gt	12c	Gt	Build	Git
11 En Mq	En	Mq	Deployment	Apache Message Queue (Apache MQ)
MSSQL Subversion			Release Mgmt	Subversion
19 Os Pg	Os	Pg	Testing	PostgreSQL
20 Fm Gh	Fm	Gh	Containerization	GitHub
21 Os Mv	Os	Mv	Collaboration	Maven
22 Os Gr	Os	Gr	Cloud/ IaaS / Pass	Gradle
23 En Mr	En	Mr	Security	Meister
24 En Jn	En	Jn	Monitoring	Jenkins
25 Fd Ba	Fd	Ba	Archiving	Bamboo
26 Os Tr	Os	Tr	Deployment	Apache Maven
27 Fr Ar	Fr	Ar	Testing	Archiva
28 Os Fn	Os	Fn	Containerization	FitNesse
29 Fr Se	Fr	Se	Testing	Selenium
30 Os Gn	Os	Gn	Deployment	Gatling
31 Pd Gd	Pd	Gd	Deployment	Deployment
32 Os Sf	Os	Sf	Deployment	Blade Logic
33 Os Cb	Os	Cb	Deployment	Smart Frog
34 Os Bc	Os	Bc	Deployment	PostgreSQL
35 Os Kb	Os	Kb	Deployment	Bcfg2
36 En Rs	En	Rs	Deployment	Kubernetes
37 Os Mg	Os	Mg	Deployment	Rackspace
38 Fm Bb	Fm	Bb	Deployment	Heroku
39 Os Br	Os	Br	Deployment	Hawkular
40 Os At	Os	At	Deployment	Redis
41 Fm Bm	Fm	Bm	Deployment	MongoDB
42 Fm Cs	Fm	Cs	Deployment	Bluemix
43 Fm Sn	Fm	Sn	Deployment	Bluemix
44 Fm Cr	Fm	Cr	Deployment	Redis
45 Os Nx	Os	Nx	Deployment	MongoDB
46 Fr Cu	Fr	Cu	Deployment	CFEngine
47 Os Cj	Os	Cj	Deployment	Packer
48 Fr Qu	Fr	Qu	Deployment	MongoDB
49 Fr Cp	Fr	Cp	Deployment	Bluemix
50 Fr Ju	Fr	Ju	Deployment	Redis
51 Os RD	Os	RD	Deployment	Redis
52 Os Cf	Os	Cf	Deployment	Redis
53 Fr Pk	Fr	Pk	Deployment	Redis
54 Fm Bx	Fm	Bx	Deployment	Redis
55 En Db	En	Db	Deployment	Redis
56 Os Hg	Os	Hg	Deployment	Redis
57 Fm Qb	Fm	Qb	Deployment	Redis
58 Os Ub	Os	Ub	Deployment	Redis
59 Fd Ta	Fd	Ta	Deployment	Redis
60 Fm Tc	Fm	Tc	Deployment	Redis
61 Fm Sh	Fm	Sh	Deployment	Redis
62 Os Cc	Os	Cc	Deployment	Redis
63 Fm Ay	Fm	Ay	Deployment	Redis
64 Fr Jt	Fr	Jt	Deployment	Redis
65 Fr Jm	Fr	Jm	Deployment	Redis
66 Fr Tn	Fr	Tn	Deployment	Redis
67 En Ry	En	Ry	Deployment	Redis
68 Fr Cy	Fr	Cy	Deployment	Redis
69 En Oc	En	Oc	Deployment	Redis
70 En No	En	No	Deployment	Redis
71 En Eb	En	Eb	Deployment	Redis
72 En Ad	En	Ad	Deployment	Redis
73 Fr Cs	Fr	Cs	Deployment	Redis
74 En Hx	En	Hx	Deployment	Redis
75 Os Mb	Os	Mb	Deployment	Redis
76 Os Rk	Os	Rk	Deployment	Redis
77 Os Lb	Os	Lb	Deployment	Redis
78 Os Co	Os	Co	Deployment	Redis
79 Os Gu	Os	Gu	Deployment	Redis
80 Os Gu	Os	Gu	Deployment	Redis
81 Os Ng	Os	Ng	Deployment	Redis
82 Os Ap	Os	Ap	Deployment	Redis
83 En Appium	En	Appium	Deployment	Redis
84 Os Xltv	Os	Xltv	Deployment	Redis
85 Os Tc	Os	Tc	Deployment	Redis
86 Fr Go	Fr	Go	Deployment	Redis
87 Fr Ef	Fr	Ef	Deployment	Redis
88 Fr Xld	Fr	Xld	Deployment	Redis
89 Os Eb	Os	Eb	Deployment	Redis
90 En Mo	En	Mo	Deployment	Redis
91 Mesos		Mesos	Deployment	Redis
92 OpenShift		OpenShift	Deployment	Redis

Figure 2: DevOps Tools Periodic Table

4. Tools Utilized

- **Git:** a *source code management* tool that allows developers to collaborate on code and track changes.
- **JUnit:** an open-source *testing framework* for Java that enables developers to write and run unit tests.
- **Maven:** a *build automation* tool that helps manage dependencies and build Java-based projects.
- **Jenkins:** a *continuous integration and continuous delivery (CI/CD)* tool that automates the build, test, and deployment processes.
- **GitHub Webhooks:** a tool that *trigger automated actions* when specific events occur in a GitHub repository.
- **Docker:** a *containerization platform* that enables developers to package applications and dependencies into portable, lightweight containers.
- **Ansible:** a *configuration management* tool that automates the deployment and management of infrastructure and applications.
- **ELK:** an acronym for Elasticsearch, Logstash, and Kibana, which are open-source tools used for *centralized logging and log analysis*.
- **ngrok:** a tool that creates *secure tunnels* to expose local servers to the public internet, useful for testing and development purposes.

5. Development and Deployment Process

Before starting, **install** the following tools:

- Git : <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
- Java: <https://www.oracle.com/in/java/technologies/downloads/#java11>
- IntelliJ IDE: <https://www.jetbrains.com/idea/download/#section=mac>
- Maven: <https://maven.apache.org/install.html>
- Jenkins: <https://www.jenkins.io/doc/book/installing/>
- Docker: <https://docs.docker.com/engine/install/>
- Ansible: https://docs.ansible.com/ansible/latest/installation_guide/index.html
- ngrok: <https://ngrok.com/download>

Create repositories on these platforms:

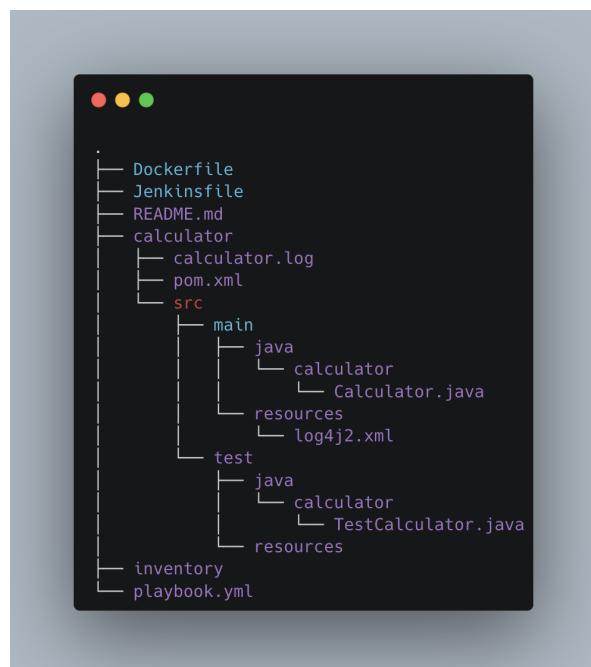
- GitHub: <https://github.com/new>
- DockerHub: <https://hub.docker.com/repository/create>

We will cover the following steps in detail in the upcoming sections:

1. Writing Java code for a Scientific Calculator program.
2. Writing unit tests using JUnit.
3. Committing and pushing changes to your GitHub repository.
4. Writing a pipeline script in Jenkins for
 - (a) Git Pull
 - (b) Maven Build
 - (c) Docker Image Creation
 - (d) Pushing Image to DockerHub
 - (e) Ansible Deployment
5. Attaching the container to your terminal.
6. Uploading the log file to ELK.

5.1 Code, Build and Test

- Tools used -
 - Programming Language - Java (version - 11)
 - Logging - Log4j
 - Testing - JUnit
 - Build - Maven
- Directory structure -



- Description of files -

- **Calculator.java** - Contains the main function including the menu and the following functions -
 - * Square root function - \sqrt{x}
 - * Factorial function - $x!$
 - * Natural logarithm (base e) - $\ln(x)$
 - * Power function - x^b
- **TestCalculator.java** - Contains JUnit test cases for the functions described above. Each function has two types of test cases: true positive and false positive.

```

Scientific_Calculator - TestCalculator.java

18  @Test
19  public void squareRootTruePositive()
20  {
21      assertEquals("Square Root Test 1", 2, cal.mySqrt(4), DELTA);
22      assertEquals("Square Root Test 2", 3, cal.mySqrt(9), DELTA);
23      assertEquals("Square Root Test 3", 9, cal.mySqrt(81), DELTA);
24      assertEquals("Square Root Test 4", 1.4142135623730951, cal.mySqrt(2), DELTA);
25  }
26
27  @Test
28  public void squareRootFalsePositive()
29  {
30      assertNotEquals("Square Root Test 1", 1, cal.mySqrt(4), DELTA);
31      assertNotEquals("Square Root Test 2", 2, cal.mySqrt(9), DELTA);
32      assertNotEquals("Square Root Test 3", 2, cal.mySqrt(15), DELTA);
33      assertNotEquals("Square Root Test 4", 2, cal.mySqrt(20), DELTA);
34  }

```

Figure 3: Example Test case

- **log4j2.xml** - Includes basic configurations for log4j, such as which appender to use (e.g., console appender, file appender) and other settings like pattern and log level.

```

Scientific_Calculator - log4j2.xml

1  <?xml version="1.0" encoding="UTF-8"?>
2  <Configuration status="INFO">
3      <Appenders>
4          <Console name="ConsoleAppender" target="SYSTEM_OUT">
5              <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
6          </Console>
7          <File name="FileAppender" fileName="calculator.log" immediateFlush="false" append="true">
8              <PatternLayout pattern="%d{yy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
9          </File>
10     </Appenders>
11     <Loggers>
12         <Root level="debug">
13             <AppenderRef ref="ConsoleAppender" />
14             <AppenderRef ref="FileAppender"/>
15         </Root>
16     </Loggers>
17 </Configuration>

```

Figure 4: Log4j configurations

- **pom.xml** - Contains information about the project and configuration details used by Maven to build the project.

```

Scientific_Calculator - pom.xml

39 <dependencies>
40   <dependency>
41     <groupId>junit</groupId>
42     <artifactId>junit</artifactId>
43     <version>4.13.2</version>
44     <scope>test</scope>
45   </dependency>
46   <dependency>
47     <groupId>org.apache.logging.log4j</groupId>
48     <artifactId>log4j-api</artifactId>
49     <version>2.20.0</version>
50   </dependency>
51   <dependency>
52     <groupId>org.apache.logging.log4j</groupId>
53     <artifactId>log4j-core</artifactId>
54     <version>2.20.0</version>
55   </dependency>
56 </dependencies>
57

```

Figure 5: Added Dependencies for JUnit and Log4j

```

Scientific_Calculator - pom.xml

11 <build>
12   <plugins>
13     <plugin>
14       <groupId>org.apache.maven.plugins</groupId>
15       <artifactId>maven-assembly-plugin</artifactId>
16       <version>3.3.0</version>
17       <executions>
18         <execution>
19           <phase>package</phase>
20           <goals>
21             <goal>single</goal>
22           </goals>
23           <configuration>
24             <archive>
25               <manifest>
26                 <mainClass>calculator.Calculator</mainClass>
27               </manifest>
28             </archive>
29             <descriptorRefs>
30               <descriptorRef>jar-with-dependencies</descriptorRef>
31             </descriptorRefs>
32           </configuration>
33         </execution>
34       </executions>
35     </plugin>
36   </plugins>
37 </build>
38

```

Figure 6: Maven Assembly Plugin to generate jar file with dependencies

- To generate a JAR file with dependencies, run the following command

```
$ mvn clean install
```

Maven will build the project and check all test cases. Once completed, a "target" directory will be created in the current directory, which will contain the JAR file.

```

divyam@Divyams-MacBook-Pro:~/Semester 8/SPE/MiniProject/Scientific_Calculator/calculator
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.582 s - in calculator.TestCalculator
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ calculator ---
[INFO] Building jar: /Users/divyam/Semester 8/SPE/MiniProject/Scientific_Calculator/calculator/target/calculator-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- assembly:3.3.0:single (default) @ calculator ---
[INFO] Building jar: /Users/divyam/Semester 8/SPE/MiniProject/Scientific_Calculator/calculator/target/calculator-0.0.1-SNAPSHOT-jar-with-dependencies.jar
[INFO]
[INFO] --- install:3.1.0:install (default-install) @ calculator ---
[INFO] Installing /Users/divyam/Semester 8/SPE/MiniProject/Scientific_Calculator/calculator/pom.xml to /Users/divyam/.m2/repository/org/calculator/0.0.1-SNAPSHOT/calculator-0.0.1-SNAPSHOT.pom
[INFO] Installing /Users/divyam/Semester 8/SPE/MiniProject/Scientific_Calculator/calculator/target/calculator-0.0.1-SNAPSHOT.jar to /Users/divyam/.m2/repository/org/calculator/0.0.1-SNAPSHOT/calculator-0.0.1-SNAPSHOT.jar
[INFO] Installing /Users/divyam/Semester 8/SPE/MiniProject/Scientific_Calculator/calculator/target/calculator-0.0.1-SNAPSHOT-jar-with-dependencies.jar to /Users/divyam/.m2/repository/org/calculator/0.0.1-SNAPSHOT/calculator-0.0.1-SNAPSHOT-jar-with-dependencies.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.975 s
[INFO] Finished at: 2023-03-19T19:40:06+05:30
[INFO] -----
divyam .../calculator > main ? >

```

Figure 7: Console Output

5.2 Source Code Management

- SCM tools allow developers to collaborate on code, track changes and maintain version control of their codebase.
- SCM tools also provide features such as branching and merging, which allow developers to work on different versions of the codebase and merge changes together.
- SCM Tool Used - Git and GitHub
- Create a new GitHub repository
- To push changes to the repository, use the following commands

```

$ git init
$ git add .
$ git commit -m "Message"
$ git branch -M main
$ git remote add origin <GitHub repo URL>
$ git push origin master

```

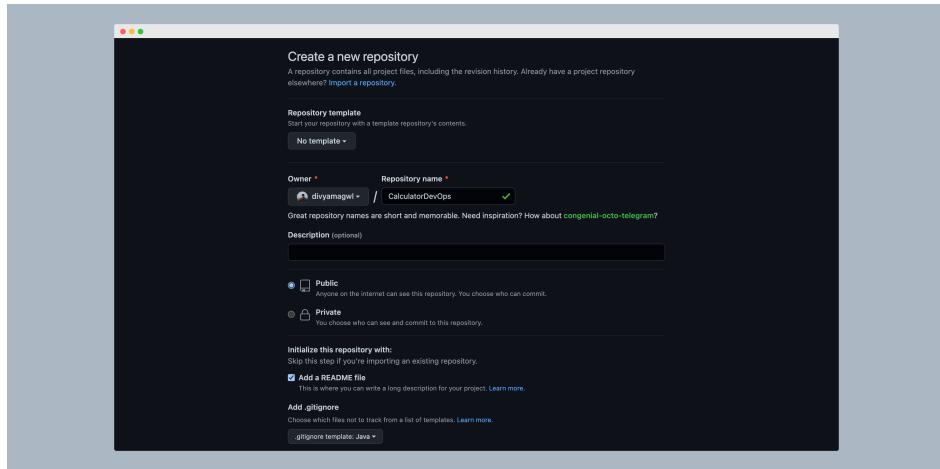


Figure 8: Creating a new repository on GitHub

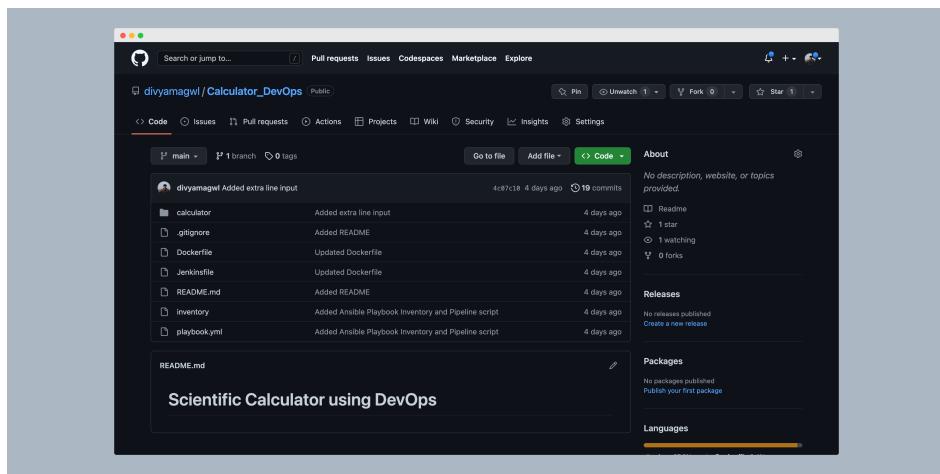


Figure 9: My GitHub repository

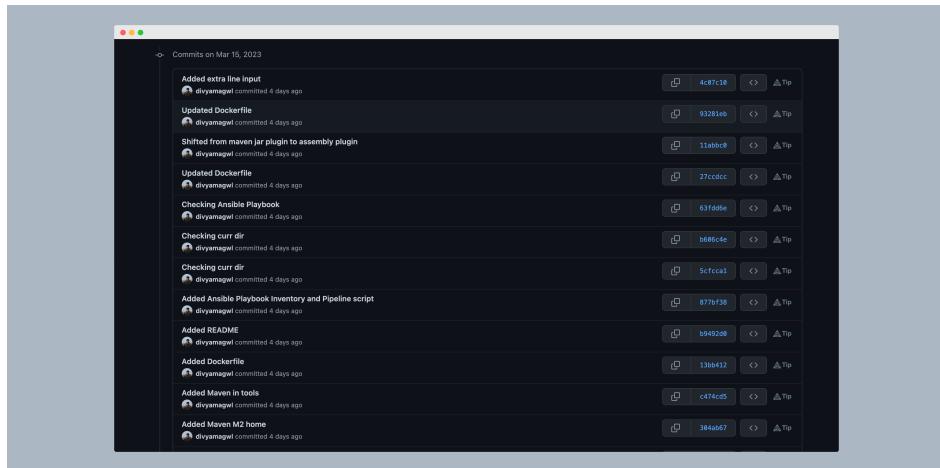


Figure 10: My GitHub commits

5.3 Continuous Integration / Continuous Deployment

- Jenkins is an open-source automation tool written in Java with plugins built for continuous integration.
- Jenkins is utilized to continuously build and test software projects, simplifying the process for developers to integrate changes, and allowing users to obtain up-to-date builds with ease.
- Additionally, Jenkins enables continuous delivery of software by integrating with a wide range of testing and deployment technologies.
- We utilize Jenkins Pipeline for our project. Jenkins Pipeline is a suite of plugins that allows developers to create and manage continuous delivery pipelines as code. It enables developers to define the entire software delivery process in a single pipeline script.
- After Jenkins is installed, browse to <http://localhost:8080> (or whichever port you configured for Jenkins when installing it)

5.3.1 Jenkins Plugins

Browse to [Manage Jenkins → Plugin Manager → Available Plugins](#) and install the following plugins -

- Git plugins & GitHub plugins
- Maven Integration
- Docker plugin & Docker pipeline
- Ansible plugin

Browse to [Manage Jenkins → Global Tool Configuration](#) and use the following configurations -



Figure 11: Git Installations config

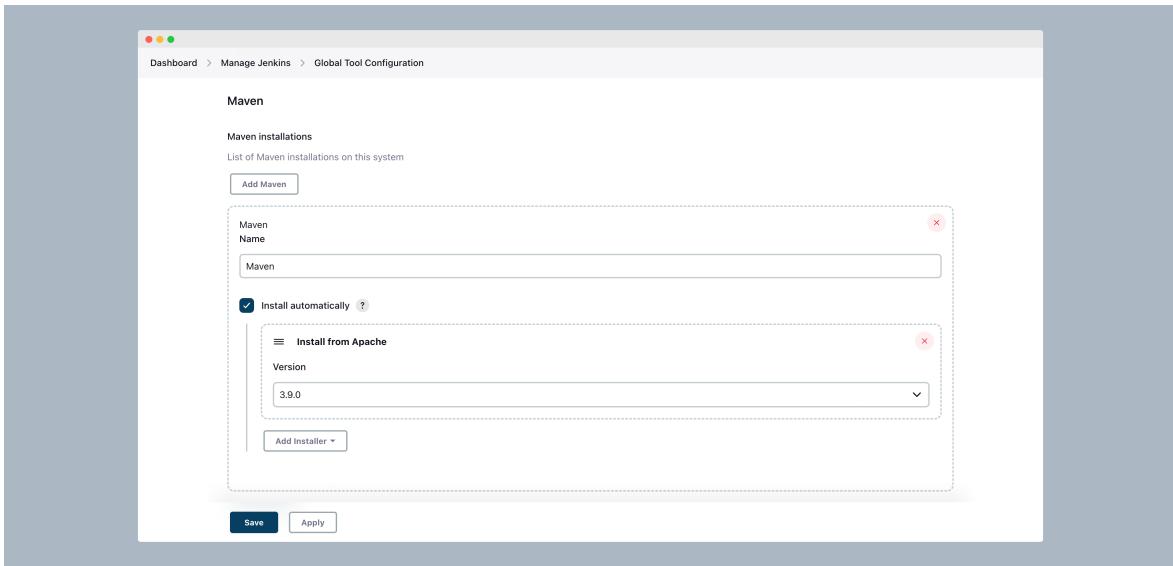


Figure 12: Maven Installations config

MacOS users will need to follow these specific steps in order to use Docker and Ansible with Jenkins. This involves adding the local installation paths for both tools to Jenkins. Please refer to the following [Stack Overflow question](#) for more details.

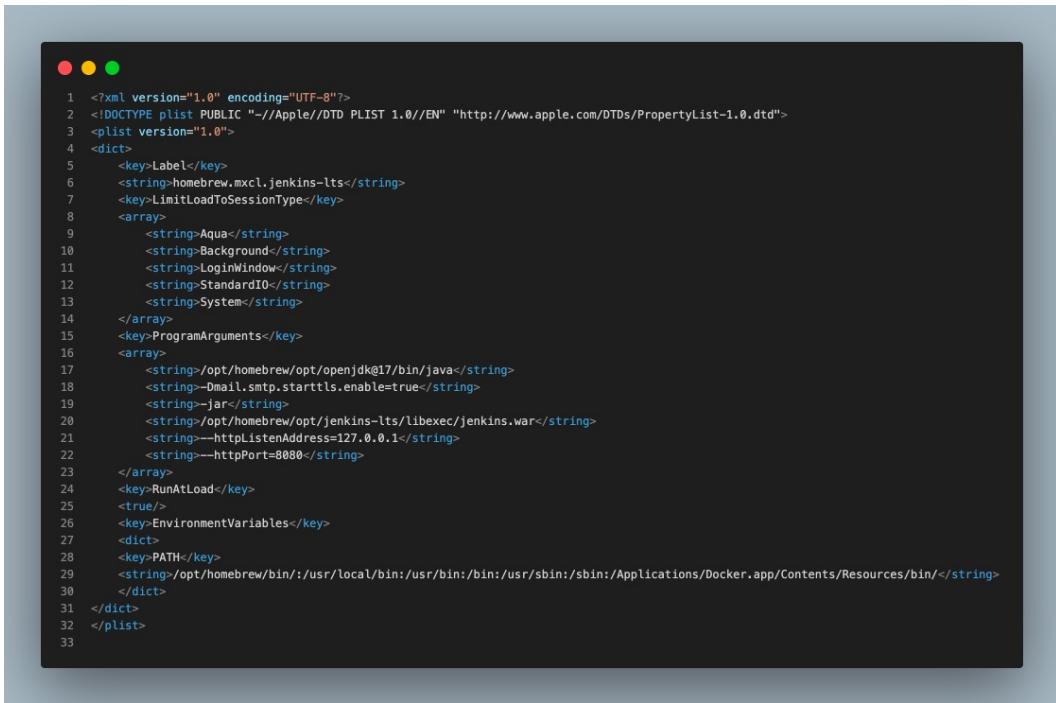
- Go to the following directory

```
$ cd /opt/homebrew/Cellar/jenkins-lts/<jenkins-version>
```

- Open `homebrew.mxcl.jenkins-lts.plist` file.
- Add the following code snippet after RunAtLoad and `<true>` :

```
<key>EnvironmentVariables</key>
<dict>
<key>PATH</key>
<string>/opt/homebrew/bin/:/usr/local/bin:/usr/bin
:/bin:/usr/sbin:/sbin:/Applications/Docker.app/
Contents/Resources/bin/</string>
</dict>
```

- We have included `/opt/homebrew/bin/` as the installation path for Ansible.
- The remaining path `/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Applications/Docker.app/Contents/Resources/bin/` has been added as the installation path for Docker.



```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3  <plist version="1.0">
4  <dict>
5      <key>Label</key>
6      <string>homebrew.mxcl.jenkins-lts</string>
7      <key>LimitLoadToSessionType</key>
8      <array>
9          <string>Aqua</string>
10         <string>Background</string>
11         <string>LoginWindow</string>
12         <string>StandardIO</string>
13         <string>System</string>
14     </array>
15     <key>ProgramArguments</key>
16     <array>
17         <string>/opt/homebrew/opt/openjdk@17/bin/java</string>
18         <string>-Dmail.smtp.starttls.enable=true</string>
19         <string>--jar</string>
20         <string>/opt/homebrew/opt/jenkins-lts/libexec/jenkins.war</string>
21         <string>--httpListenAddress=127.0.0.1</string>
22         <string>--httpPort=8080</string>
23     </array>
24     <key>RunAtLoad</key>
25     <true/>
26     <key>EnvironmentVariables</key>
27     <dict>
28         <key>PATH</key>
29         <string>/opt/homebrew/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Applications/Docker.app/Contents/Resources/bin/</string>
30     </dict>
31 </dict>
32 </plist>
33

```

Figure 13: homebrew.mxcl.jenkins-lts.plist

Browse to [Manage Jenkins → Manage Credentials](#) and add your DockerHub credentials and GitHub Personal Access token.

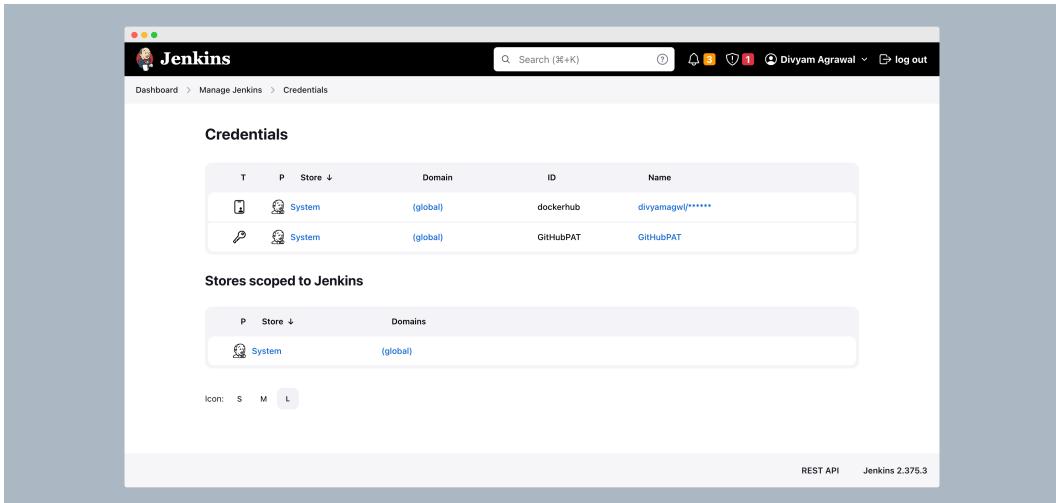


Figure 14: Credentials stored in Jenkins

Browse to [Manage Jenkins → Configure system](#) and add GitHub server with the following configuration.

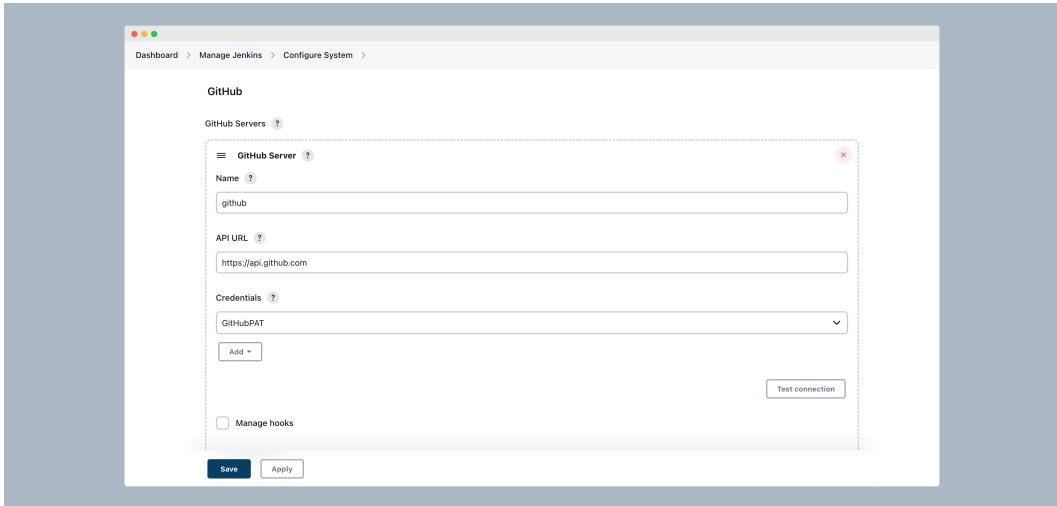


Figure 15: GitHub Server configuration in Jenkins

5.3.2 Jenkins Pipeline

- Create a Jenkinsfile which will contain the stages of the Jenkins pipeline -

- **Pull GitHub:**

```
stage('Pull GitHub') {  
    steps {  
        git branch: 'main', url: 'https://github.com/divyamagwl/Calculator_DevOps'  
    }  
}
```

Pulls the remote repository from GitHub using Jenkins.

- **Build Maven jar package:**

```
stage('Build Maven jar package') {  
    steps {  
        dir("calculator/") {  
            script{  
                sh 'mvn clean install'  
            }  
        }  
    }  
}
```

Generates a JAR file with all required dependencies, while also deleting the old target folder and installing all dependencies again for the new target folder.

- **Docker Image Build:**

```
environment
{
    registry = "divyamagwl/calculator"
    registryCredential = "dockerhub"
    dockerImage = ""
}

stage('Docker Image Build') {
    steps {
        script {
            dockerImage = docker.build(registry
                + ":latest")
        }
    }
}
```

The docker build command builds Docker Images from a Dockerfile and a “context”. A build’s context is the set of files located in the specified PATH or URL.

- **DockerHub Image Push:**

```
stage('DockerHub Image Push') {
    steps {
        script {
            docker.withRegistry('',
                registryCredential) {
                dockerImage.push()
            }
        }
    }
}
```

The docker push command shares your image to the DockerHub registry.

- **Removing Docker Image:**

```
stage('Cleaning Up') {
    steps {
        sh "docker rmi $registry:latest"
    }
}
```

The docker rmi command removes one or more images from the host node.

- Execute Ansible:

```
stage('Execute Ansible') {  
    steps {  
        ansiblePlaybook colorized: true,  
        installation: 'Ansible',  
        inventory: 'inventory',  
        playbook: 'playbook.yml'  
    }  
}
```

Runs Ansible Playbook.

- Browse to [Add Item](#) and select Pipeline project.

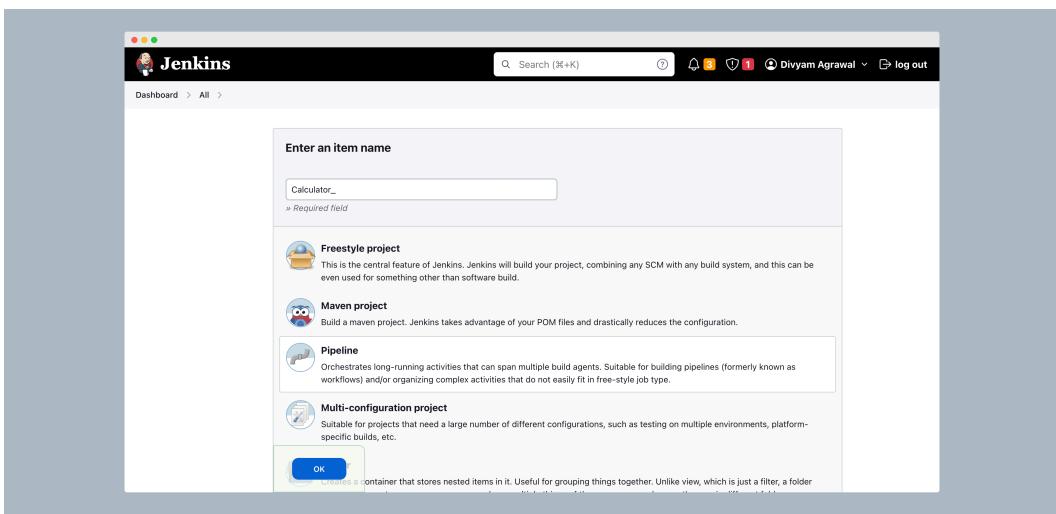


Figure 16: Creating a new Jenkins Project

- For the job to work with a GitHub repository, use the following configuration -

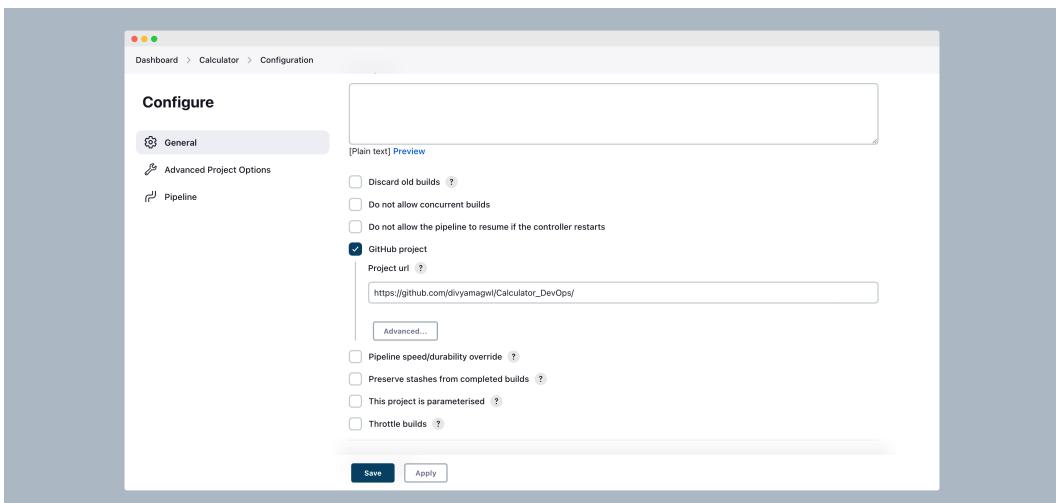


Figure 17: Selecting GitHub project

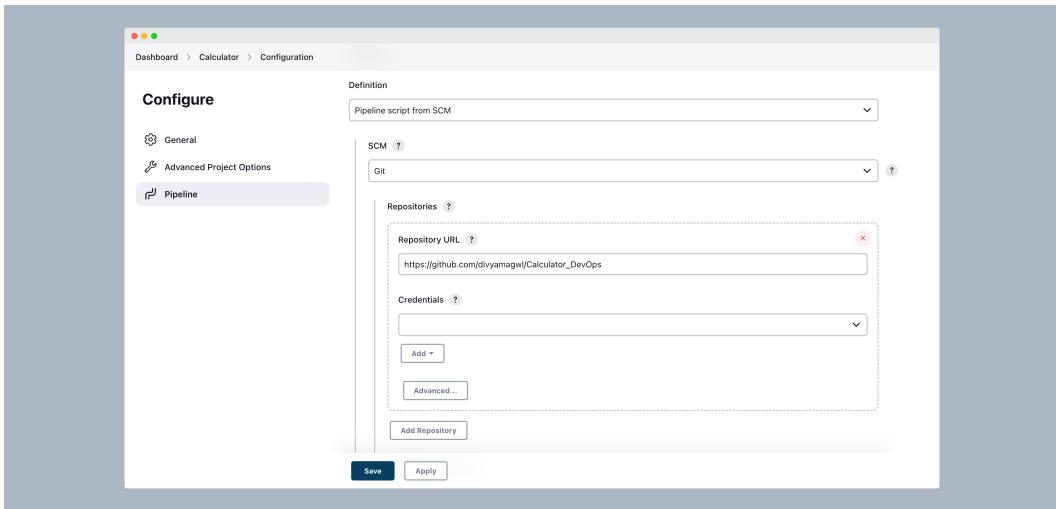


Figure 18: Pipeline script from SCM

- My Jenkins Project

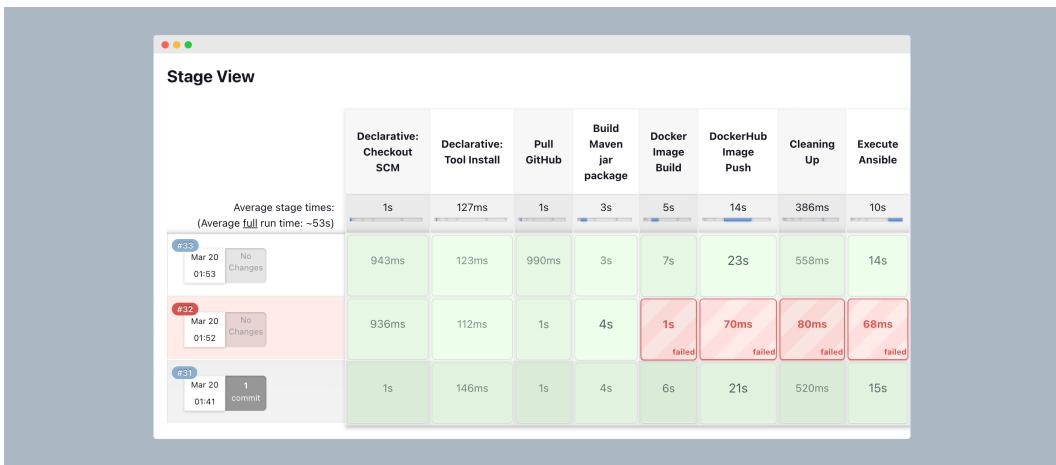


Figure 19: Pipeline Execution

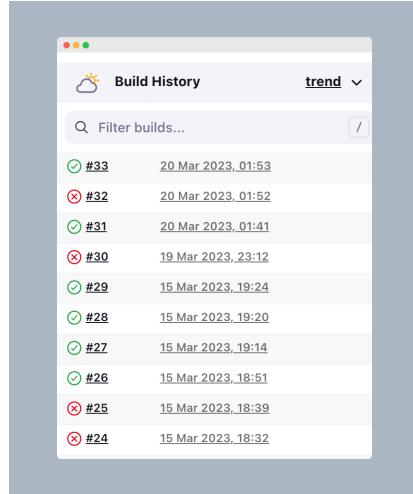


Figure 20: Build History

- Upon a successful build, the Docker Image and running container will be visible on the host node. We can attach the running container to our terminal using the `docker attach` command.

```

divyam ~/calculator > docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
divyamagw1/calculator    latest    8e40b75a8302  10 minutes ago  647MB
divyamagw1/calculator    <none>   da67e0eeeef1  21 minutes ago  647MB
divyam ~/calculator > docker ps -a
CONTAINER ID IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
f5c361752788  divyamagw1/calculator "java -jar /calculat..."  9 minutes ago  Up 9 minutes          calculator
divyam ~/calculator > docker attach calculator
Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: 1
Enter number: 4
20:34:02.632 [main] INFO calculator.Calculator - [SQUARE ROOT] [SUCCESS] 4.0
20:34:02.638 [main] INFO calculator.Calculator - [RESULT - SQUARE ROOT] 2.0
Square root of 4.0 is 2.0
Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: |

```

Figure 21: Docker attach running container

5.3.3 GitHub Webhook

- A Webhook is a mechanism to automatically trigger the build of a Jenkins project in response to a commit pushed to a Git repository.
- We will use Ngrok to expose Jenkins local web server running on port 8080 to the internet. Ngrok is a tool that enables developers to create secure tunnels from a public endpoint to a locally running web service.
- After installing Ngrok, execute the following command-

```
$ ngrok http 8080
```

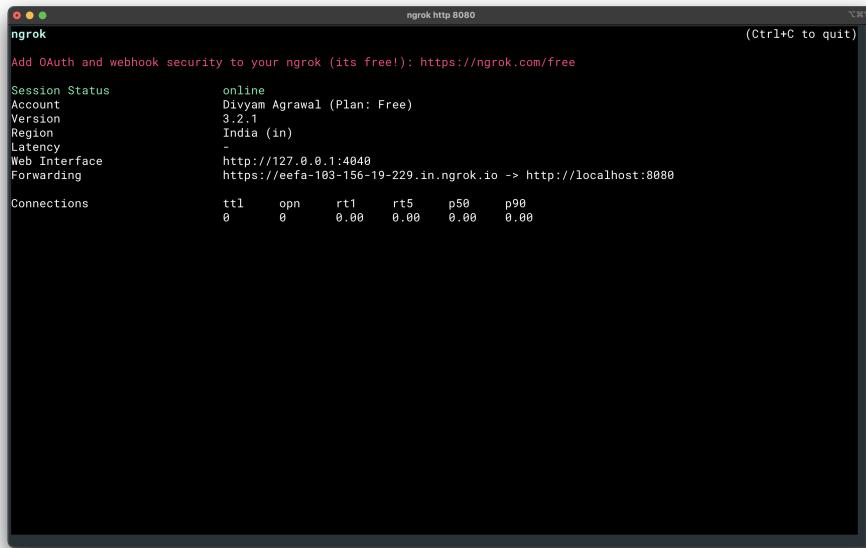


Figure 22: Ngrok secure tunnel

- Browse to [Manage Jenkins → Configure system](#) and add Ngrok Forwarding URL to Jenkins URL.



Figure 23: Add Ngrok URL in Jenkins URL

- Browse to [Your Jenkins Project → Configure → Build Triggers](#) and select [GitHub hook trigger for GITScm polling](#)

- In your GitHub repository, browse to [Settings → Webhooks](#) and add Ngrok Forwarding URL appended with ”github-webhook/” in payload URL and Personal Access token in Secret.

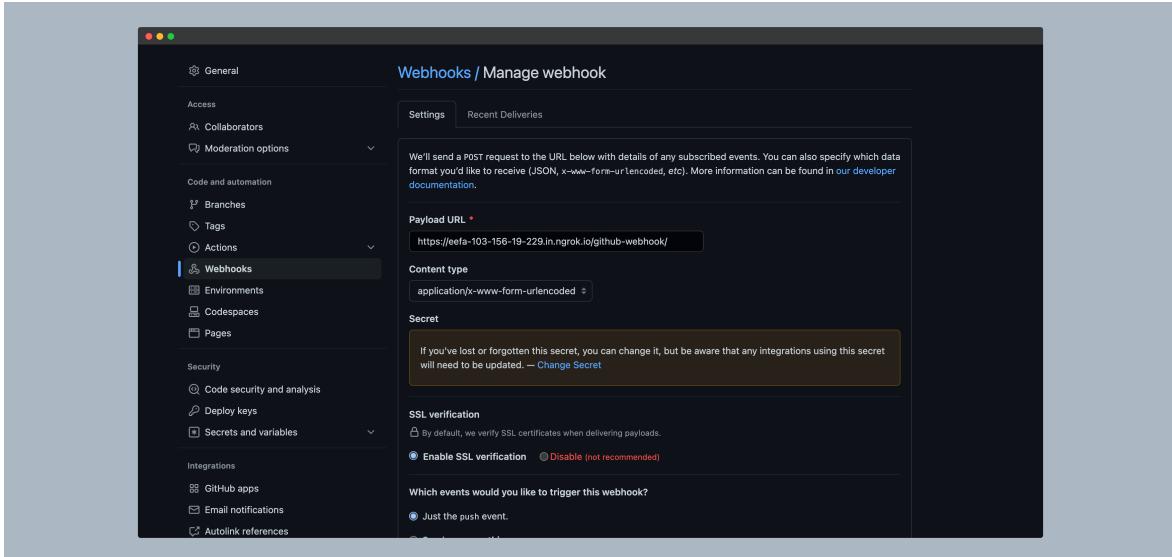


Figure 24: Add Payload URL and Secret

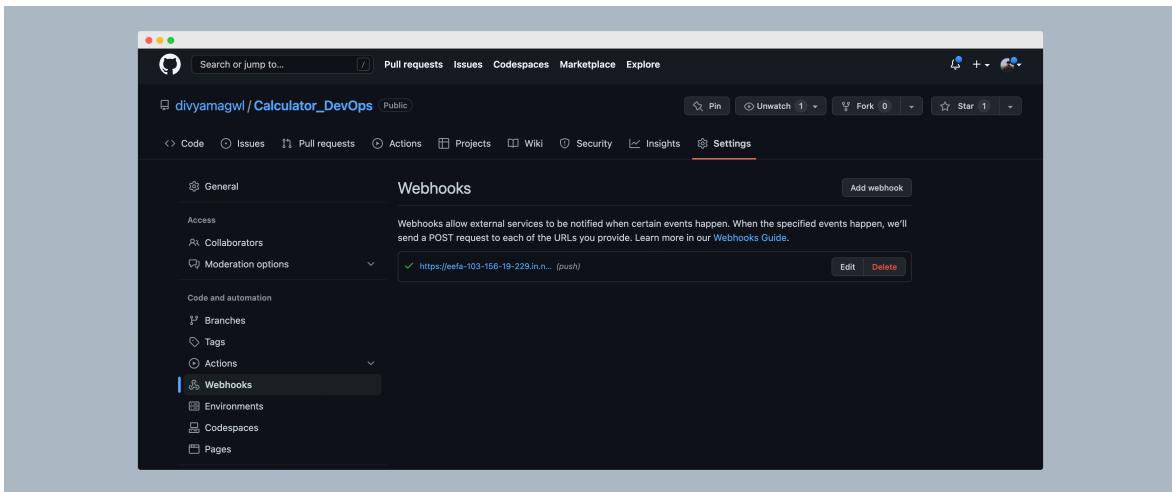


Figure 25: Successful Delivery

5.4 Containerization

- Containerization is a software deployment process that bundles an application’s code with all the files and libraries it needs to run on any infrastructure.
- Containers are lightweight, portable, and self-contained environments that enable developers to package an application with all its dependencies, libraries, and configuration files, ensuring that it runs consistently across different environments.

- We will use Docker to create containers. Docker is an open-source tool that enables developers to build, package, and deploy applications in a containerized environment.
- Create a Dockerfile

```

Scientific_Calculator - Dockerfile

1 FROM openjdk:11
2 ADD ./calculator/target/calculator-0.0.1-SNAPSHOT-jar-with-dependencies.jar ./
3 WORKDIR ./
4 ENTRYPOINT ["java","-jar","/calculator-0.0.1-SNAPSHOT-jar-with-dependencies.jar"]

```

Figure 26: Dockerfile

- **FROM:** We use the OpenJDK 11 base image to build our image.
- **ADD:** Copy jar file from source on the host machine into the container's file system.
- **WORKDIR:** Changes the current working directory.
- **ENTRYPOINT:** Specify the command that should be run when a container based on the image is started.
- The Jenkins Pipeline script has been configured to include the Dockerfile and Docker commands, automating the process of building and pushing the Docker Image to DockerHub.

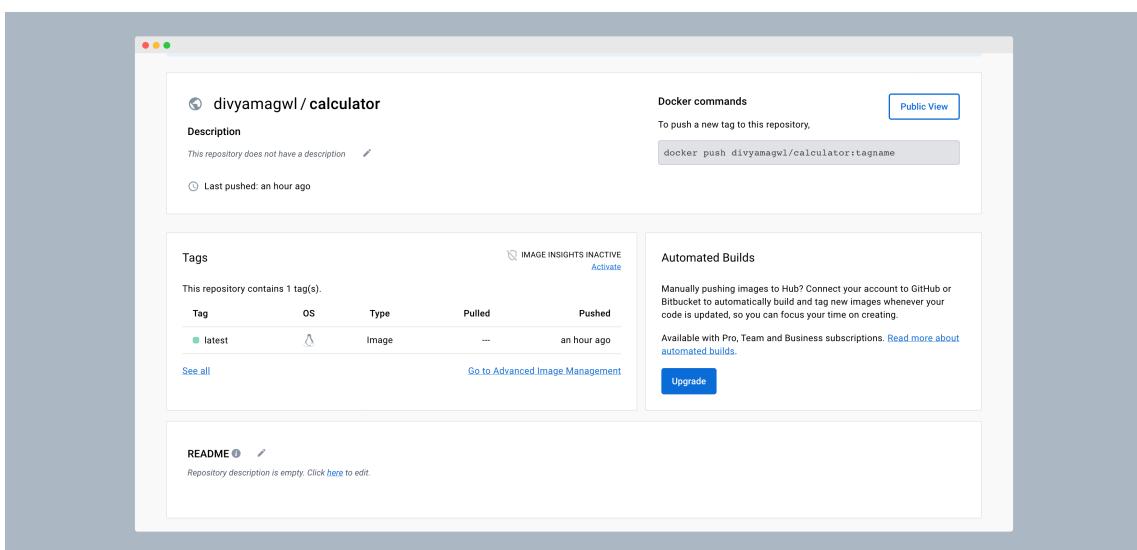
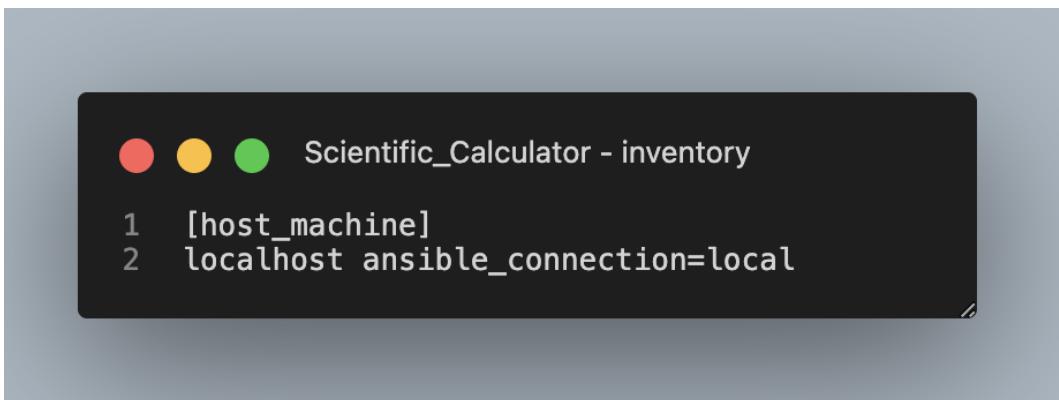


Figure 27: DockerHub public repository

5.5 Deployment

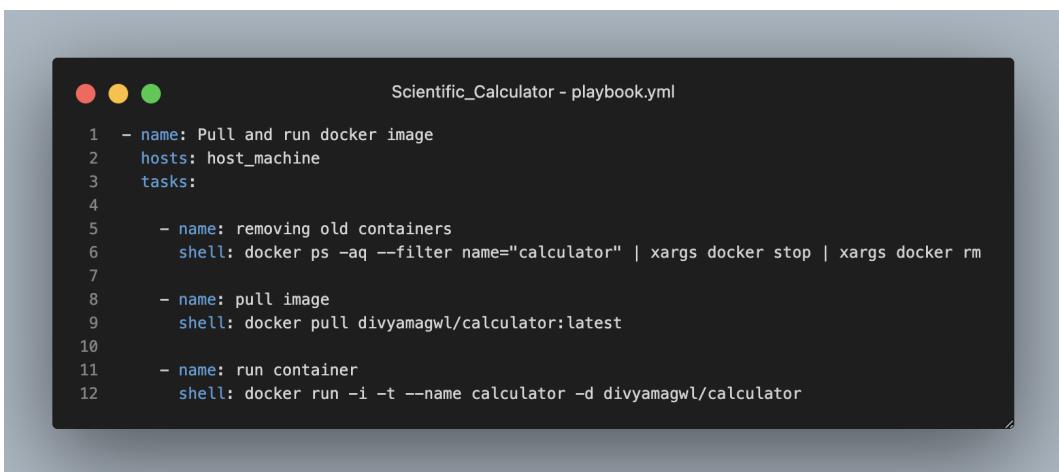
- We will use Ansible for local deployment. Ansible is a suite of software tools that enables infrastructure as code.
- In Ansible, managed hosts or servers which are controlled by the Ansible control node are defined in a host inventory file. The Ansible inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate.
- Create an inventory file where we explicitly define our localhost in the inventory file for local deployment.



```
Scientific_Calculator - inventory
1 [host_machine]
2 localhost ansible_connection=local
```

Figure 28: Ansible Inventory

- Ansible Playbooks offer a repeatable, re-usable, simple configuration management.
- Playbooks consist of one or more plays run in a particular order. A play is an ordered set of tasks run against hosts chosen from your inventory. Plays define the work to be done. Each play contains a set of hosts to configure, and a list of tasks to be executed.



```
Scientific_Calculator - playbook.yml
1 - name: Pull and run docker image
2   hosts: host_machine
3   tasks:
4
5     - name: removing old containers
6       shell: docker ps -aq --filter name="calculator" | xargs docker stop | xargs docker rm
7
8     - name: pull image
9       shell: docker pull divyamagwl/calculator:latest
10
11    - name: run container
12      shell: docker run -i -t --name calculator -d divyamagwl/calculator
```

Figure 29: Ansible Playbook

- Filtering out all the containers with the name "calculator", then stopping the running containers and removing them from the host machine.

```
- name: removing old containers
  shell: docker ps -aq --filter name="calculator"
         " | xargs docker stop | xargs docker rm
```

- Pulling our docker image from DockerHub to the host machine.

```
- name: pull image
  shell: docker pull divyamagwl/calculator:
         latest
```

- Creating a writeable container layer over our specified image and then starting it.

```
- name: run container
  shell: docker run -i -t --name calculator -d
         divyamagwl/calculator
```

The screenshot shows a terminal window on a Mac OS X system. The user is executing an Ansible playbook named 'playbook.yml' against a local inventory. The terminal output shows the following steps:

- Ansible ping:** The user runs 'ansible -i inventory host_machine -m ping'. The output shows facts about the Python interpreter installed on the host machine.
- Playbook execution:** The user runs 'ansible-playbook playbook.yml -i inventory'. This step includes pulling the Docker image and creating a new container named 'calculator'.
- Docker images:** The user runs 'docker images' to list the available Docker images. It shows one image: 'divyamagwl/calculator latest be07b2004337'.
- Docker ps:** The user runs 'docker ps -a' to list all Docker containers. It shows one container named 'calculator' with the ID '5c8aab92b15f'.

Figure 30: Pinging host machine and running playbook manually

- The Jenkins Pipeline script has been configured to execute the Ansible Playbook automatically.

5.6 Monitoring

- ELK stack is an open-source toolset that is used for log management and analysis. The ELK stack consists of three main components:
 - Elasticsearch: A distributed, RESTful search and analytics engine designed to store, search, and analyze large volumes of data in real-time.
 - Logstash: A data processing pipeline that collects, parses, and enriches log data from various sources and sends it to Elasticsearch for storage and analysis.
 - Kibana: A web-based user interface for visualizing and analyzing data stored in Elasticsearch, including metrics, logs, and application performance data.
- Go to <https://www.elastic.co/cloud/elasticsearch-service/signup> and sign up using a temporary email address.
- Follow the instructions provided by the platform to create a deployment and wait a few minutes for the deployment process to complete.

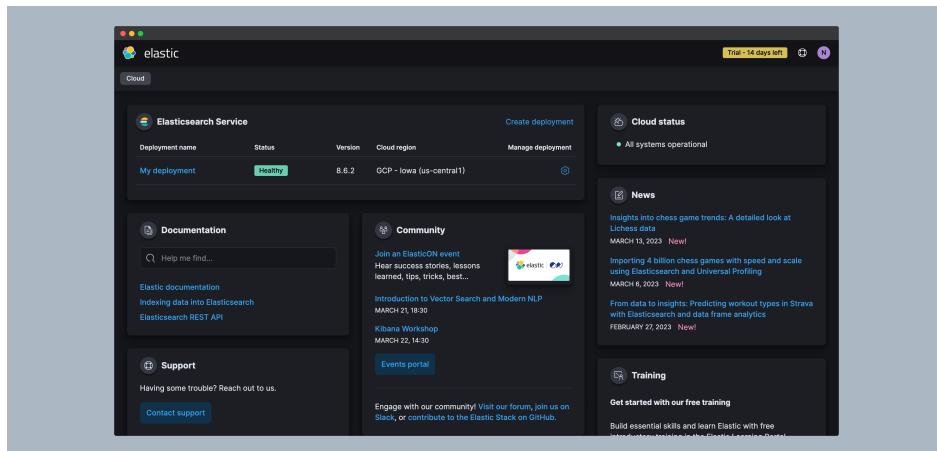


Figure 31: Elastic Cloud Home

- To obtain the log file from your Docker container, use the following command:

```
$ docker cp calculator:/calculator.log ../monitoring
```
- Now, navigate to **Your Deployment → Analytics → Machine Learning → Data Visualizer → File**
- Upload the log file to the dashboard.

- Click on "Override settings" and update the grok pattern with the following:

```
%{TIMESTAMP_ISO8601:timestamp} \[%{WORD:thread}\]
%{LOGLEVEL:log.level} .*? - \[%{WORD:function}\]
- %{NUMBER:value1}(\s%{NUMBER:value2})? - \[
RESULT\] - %{GREEDYDATA:result}
```

This grok pattern is based on the log outputs that were written during the code development.

```
divyam@Divyams-MacBook-Pro:~/Semester 8/SPE/MiniProject/Scientific_Calculator/calculator$ cat ..monitoring/calculator.log
2023-03-20 13:51:42.092 [main] INFO calculator.Calculator - [SQUARE_ROOT] - 4.0 - [RESULT] - 2.0
2023-03-20 13:51:46.217 [main] INFO calculator.Calculator - [SQUARE_ROOT] - 13.0 - [RESULT] - 3.605551275463989
2023-03-20 13:51:50.951 [main] ERROR calculator.Calculator - [SQUARE_ROOT] - 2.0 - [RESULT] - null
2023-03-20 13:51:54.230 [main] INFO calculator.Calculator - [SQUARE_ROOT] - 0.0 - [RESULT] - 0.0
2023-03-20 13:51:56.849 [main] INFO calculator.Calculator - [FACTORIAL] - 10.0 - [RESULT] - 3628800.0
2023-03-20 13:52:00.286 [main] INFO calculator.Calculator - [FACTORIAL] - 4.0 - [RESULT] - 24.0
2023-03-20 13:52:03.001 [main] INFO calculator.Calculator - [FACTORIAL] - 5.0 - [RESULT] - null
2023-03-20 13:52:05.487 [main] INFO calculator.Calculator - [FACTORIAL] - 1.0 - [RESULT] - 1.0
2023-03-20 13:52:08.145 [main] INFO calculator.Calculator - [LOGARITHM] - 100.0 - [RESULT] - 4.605170185988092
2023-03-20 13:52:10.511 [main] INFO calculator.Calculator - [LOGARITHM] - 10.0 - [RESULT] - 2.302585092994846
2023-03-20 13:52:12.193 [main] ERROR calculator.Calculator - [LOGARITHM] - 1.0 - [RESULT] - null
2023-03-20 13:52:15.181 [main] INFO calculator.Calculator - [LOGARITHM] - 0.0 - [RESULT] - null
2023-03-20 13:52:18.218 [main] INFO calculator.Calculator - [POWER] - 0.0 1.0 - [RESULT] - 0.0
2023-03-20 13:52:20.283 [main] INFO calculator.Calculator - [POWER] - 1.0 4.0 - [RESULT] - 1.0
2023-03-20 13:52:23.613 [main] INFO calculator.Calculator - [POWER] - 5.0 2.0 - [RESULT] - 25.0
2023-03-20 13:52:24.844 [main] INFO calculator.Calculator - [LOGARITHM] - 21.0 - [RESULT] - 3.04452437723423
2023-03-20 13:52:31.866 [main] INFO calculator.Calculator - [LOGARITHM] - 12.0 - [RESULT] - 2.484966497880004
2023-03-20 13:52:34.185 [main] INFO calculator.Calculator - [SQUARE_ROOT] - 24.0 - [RESULT] - 4.898979485566356
2023-03-20 13:52:35.766 [main] INFO calculator.Calculator - [SQUARE_ROOT] - 12.0 - [RESULT] - 3.4641016151377544
2023-03-20 13:52:44.777 [main] INFO calculator.Calculator - [FACTORIAL] - 3.0 - [RESULT] - 6.0
2023-03-20 13:52:50.357 [main] INFO calculator.Calculator - [LOGARITHM] - 10.0 - [RESULT] - 2.302585092994846
2023-03-20 13:52:53.813 [main] INFO calculator.Calculator - [LOGARITHM] - 23.0 - [RESULT] - 3.1354942159291497
divyam@Divyams-MacBook-Pro:~/Semester 8/SPE/MiniProject/Scientific_Calculator/calculator$
```

Figure 32: My Log file

- Finally, import the file and specify an index name.

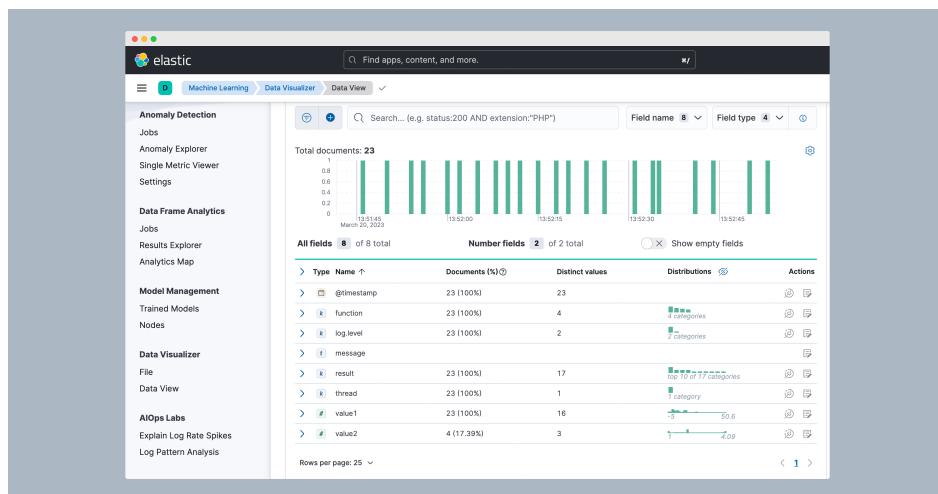


Figure 33: Data View

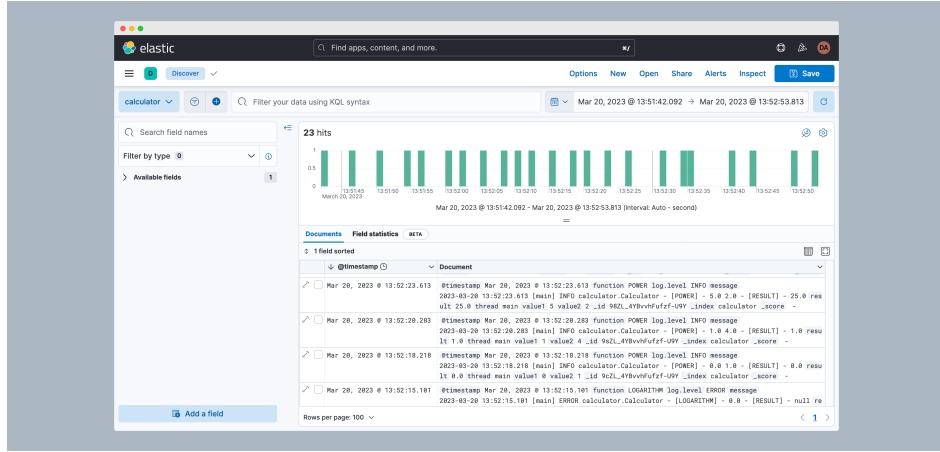


Figure 34: Discover View with documents from log

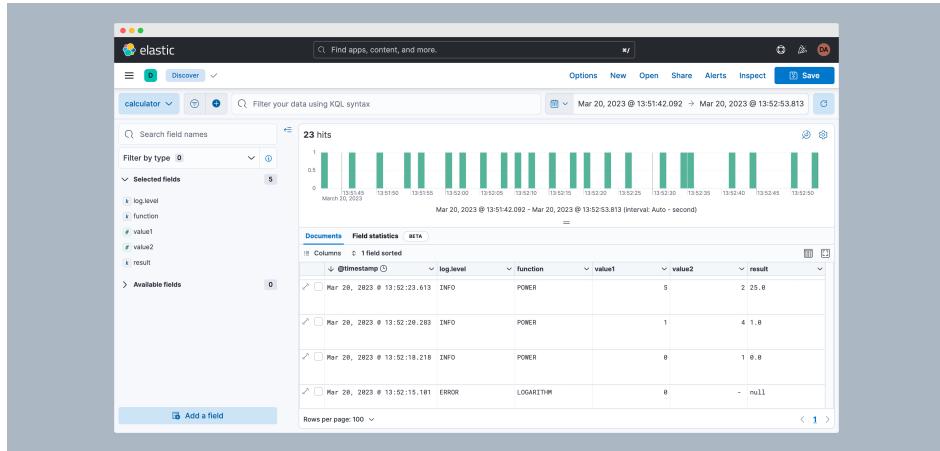


Figure 35: Discover View with selected fields from log

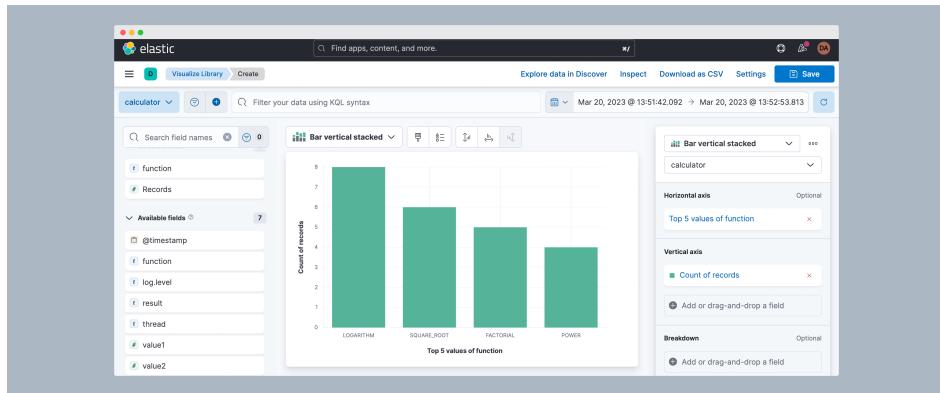


Figure 36: Function call count visualization

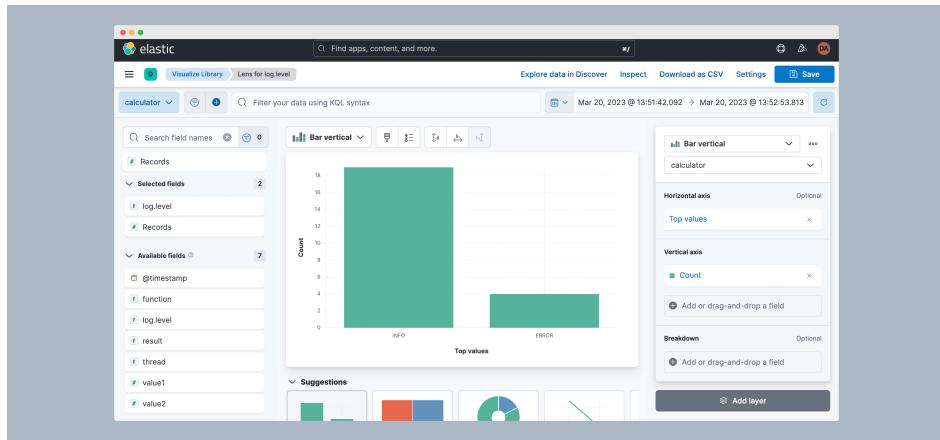


Figure 37: Log Level count visualization

5.7 Scientific Calculator Outputs

```
divyam >~/calculator> p main ?
>>> docker attach calculator
Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: 1
Enter number: 4
15:49:27.842 [main] INFO calculator.Calculator - [SQUARE_ROOT] - 4.0 - [RESULT] - 2.0
Square root of 4.0 is 2.0

Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: 2
Enter number: 3
15:49:30.697 [main] INFO calculator.Calculator - [FACTORIAL] - 3.0 - [RESULT] - 6.0
Factorial of 3.0 is 6.0

Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice:
```

Figure 38: Square root and Factorial functions

```
docker attach calculator
4. Power
0. Exit
Enter your choice: 3
Enter number: 2.71
15:49:48.335 [main] INFO calculator.Calculator - [LOGARITHM] - 2.71 - [RESULT] - 0.9969486348916095
Natural logarithm (base e) of 2.71 is 0.9969486348916095

Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: 4
Enter first number: 2
Enter second number: 3
15:49:59.860 [main] INFO calculator.Calculator - [POWER] - 2.0 3.0 - [RESULT] - 8.0
Power of 2.0 to 3.0 is 8.0

Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: 1
Enter number: -2
15:50:09.242 [main] ERROR calculator.Calculator - [SQUARE_ROOT] - -2.0 - [RESULT] - null
Invalid input

Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice:
```

Figure 39: Natural log and Power functions

```
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: 1
Enter number: -2
15:50:09.242 [main] ERROR calculator.Calculator - [SQUARE_ROOT] - -2.0 - [RESULT] - null
Invalid input

Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: 2
Enter number: -1
15:50:27.359 [main] ERROR calculator.Calculator - [FACTORIAL] - -1.0 - [RESULT] - null
Invalid input

Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: 3
Enter number: -10
15:50:38.659 [main] ERROR calculator.Calculator - [LOGARITHM] - -10.0 - [RESULT] - null
Invalid input

Scientific Calculator
-----
1. Square root
2. Factorial
3. Natural logarithm (base e)
4. Power
0. Exit
Enter your choice: |
```

Figure 40: Invalid inputs