

CS 816 - Software Production Engineering

Mini Project - Scientific Calculator with DevOps

Sivani Channamsetty
IMT2019020

CONTENTS

1. Problem Statement
2. Introduction(What and Why of Devops)
3. Setting up tools
 - a. Java
 - b. Git and Github
 - c. Maven
 - d. Intellij
 - e. Jenkins
 - f. Docker
 - g. Ansible
 - h. Ngrok
4. Project Workflow
 - a. Git and Github
 - b. Intellij, Maven and JUnit
 - c. Docker
 - d. Ansible
 - e. Ngrok and Github WebHooks
 - f. Jenkins
 - g. ELK-Stack
5. Challenges
6. Links

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

Introduction(What and Why of Devops)

DevOps is a combination of culture, methods, and tools that improves an organization's capacity to release applications more quickly. This model breaks the wall between the development team and the operation team. The project's objective is to develop a calculator using deployment and integration automation tools. Source code management (SCM), continuous integration, continuous deployment, and configuration management for the entire source code are all included in the automation. This is achieved by putting into practise some of the devops tools that assist in the automation of the mentioned processes. The project entails developing a scientific calculator program that contains mathematical operations like square root calculation, factorial, natural logarithm, and power operations utilising devops tools like github, maven, junit, jenkins, ansible, and docker. The main objective of the project is to build a Jenkins pipeline while learning the CI/CD/CM devops concepts.

Frequently, there is friction between developers and operations teams because of the inherent differences in their roles. While developers aim to introduce new features to the application through changes, operations teams are cautious of any changes that could potentially harm the system, leading to application downtime. Organizations that embrace DevOps can enhance and expedite the evolution of software products more effectively than those using traditional software development approaches. DevOps serves as a bridge between development and operations within a company with the ultimate objective of boosting overall productivity. The DevOps team plays a critical role in bridging the gap between development and operations, thereby enabling testers and operators to work more efficiently.

Advantages of devops

- Improved deployment frequency
- Faster time to market
- Lower failure rate of new release
- Shortened lead time between fixes
- faster mean time to recovery in the event of a new release crashing

Tools used for

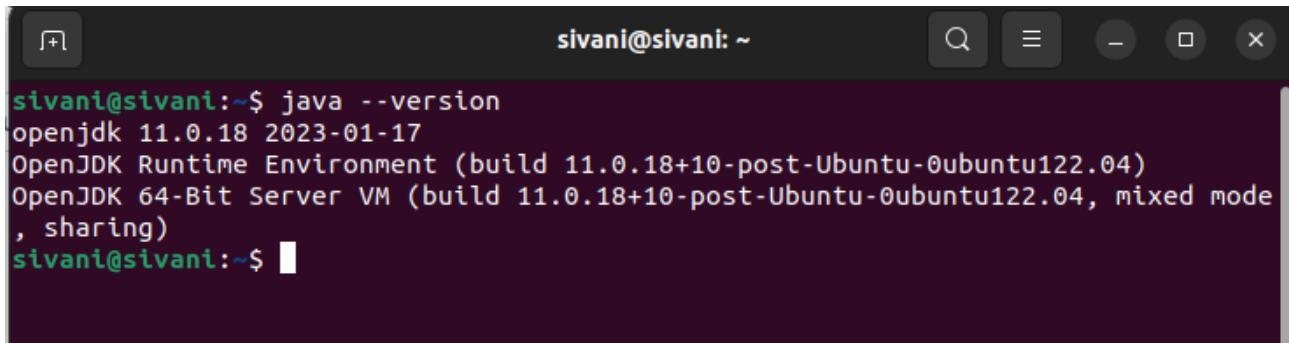
- *Source control management* : Git and GitHub
- *To perform WebHooks* : Ngrok
- *Automate the build process* : Github WebHooks
- *Code editor* : IntelliJ
- *Building and packaging the code* : Maven
- *Continuous Integration* : Jenkins

- *Containerising the code*: Docker
- *Configuration Management and Continuous Deployment* : Ansible
- *Monitoring* : ELK-stack(Elasticsearch)

Setting up tools

Java :

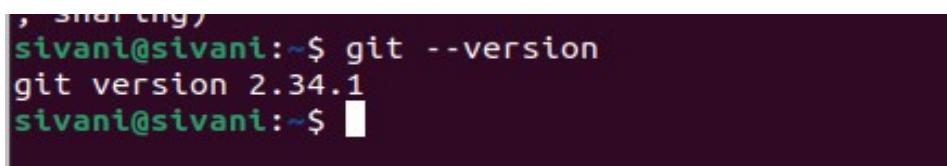
- *sudo apt-get update*
- *sudo apt install openjdk-11-jre-headless*
- *java --version*



```
sivani@sivani:~$ java --version
openjdk 11.0.18 2023-01-17
OpenJDK Runtime Environment (build 11.0.18+10-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.18+10-post-Ubuntu-0ubuntu122.04, mixed mode
, sharing)
sivani@sivani:~$
```

Git and Github:

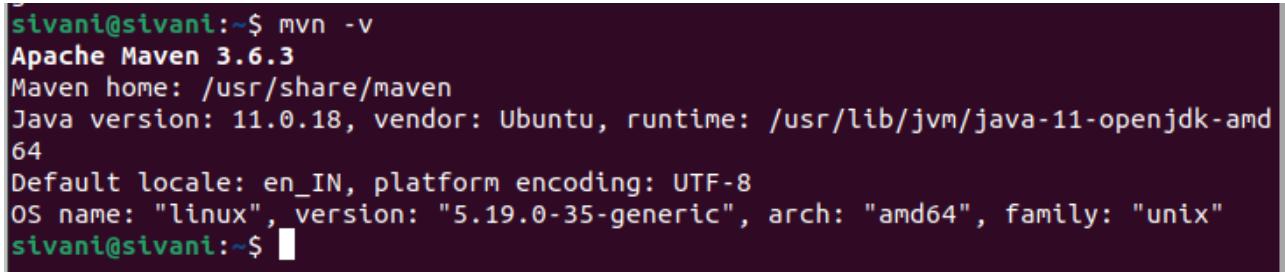
- *sudo apt-get update*
- *sudo apt-get install git*
- *git --version*



```
, sharing,
sivani@sivani:~$ git --version
git version 2.34.1
sivani@sivani:~$
```

Maven :

- *sudo apt-get update*
- *sudo apt-get install maven*
- *mvn -v*



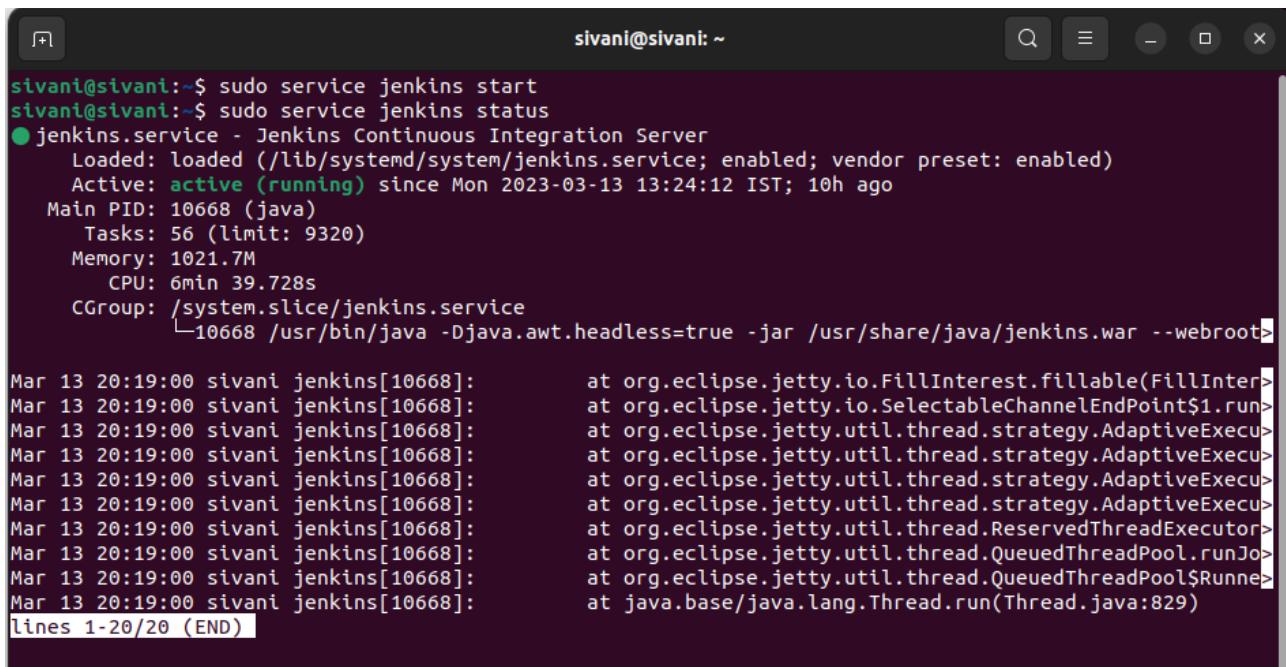
```
sivani@sivani:~$ mvn -v
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.18, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd
64
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "5.19.0-35-generic", arch: "amd64", family: "unix"
sivani@sivani:~$
```

IntelliJ :

- For stable : `sudo snap install intellij-idea-ultimate --classic`
- For EAP builds : `sudo snap install intellij-idea-community --classic --edge`

Jenkins :

- `wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -`
- `sudo sh -c 'echo deb https://pkg.jenkins.io/debian binary/ >/etc/apt/sources.list.d/jenkins.list'`
- `sudo apt install ca-certificates`
- `sudo apt-get update`
- `sudo apt-get install jenkins`
- `sudo service jenkins start`
- `sudo service jenkins status`
- `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` (copy jenkins password)
- Go to `http://localhost:8080/` to use jenkins
- Paste the password there and start jenkins



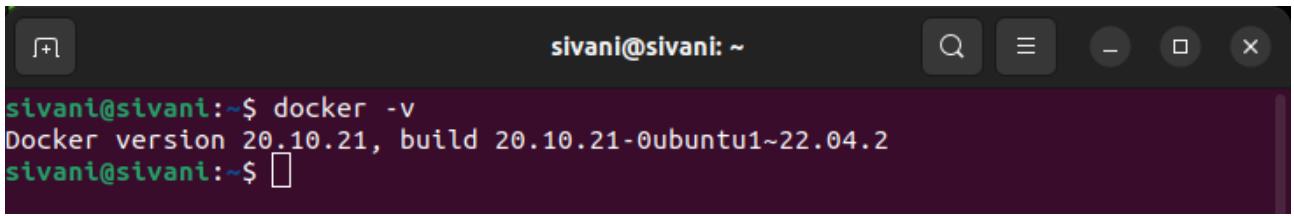
A screenshot of a terminal window titled "sivani@sivani: ~". The terminal shows the following command output:

```
sivani@sivani:~$ sudo service jenkins start
sivani@sivani:~$ sudo service jenkins status
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2023-03-13 13:24:12 IST; 10h ago
    Main PID: 10668 (java)
      Tasks: 56 (limit: 9320)
     Memory: 1021.7M
        CPU: 6min 39.728s
       CGroup: /system.slice/jenkins.service
               └─10668 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot>

Mar 13 20:19:00 sivani jenkins[10668]:          at org.eclipse.jetty.io.FillInterest.fillable(FillInter>
Mar 13 20:19:00 sivani jenkins[10668]:          at org.eclipse.jetty.io.SelectableEndPoint$1.run>
Mar 13 20:19:00 sivani jenkins[10668]:          at org.eclipse.jetty.util.thread.strategy.AdaptiveExecu>
Mar 13 20:19:00 sivani jenkins[10668]:          at org.eclipse.jetty.util.thread.ReservedThreadExecutor>
Mar 13 20:19:00 sivani jenkins[10668]:          at org.eclipse.jetty.util.thread.QueuedThreadPool.runJo>
Mar 13 20:19:00 sivani jenkins[10668]:          at org.eclipse.jetty.util.thread.QueuedThreadPool$Runne>
Mar 13 20:19:00 sivani jenkins[10668]:          at java.base/java.lang.Thread.run(Thread.java:829)
lines 1-20/20 (END)
```

Docker :

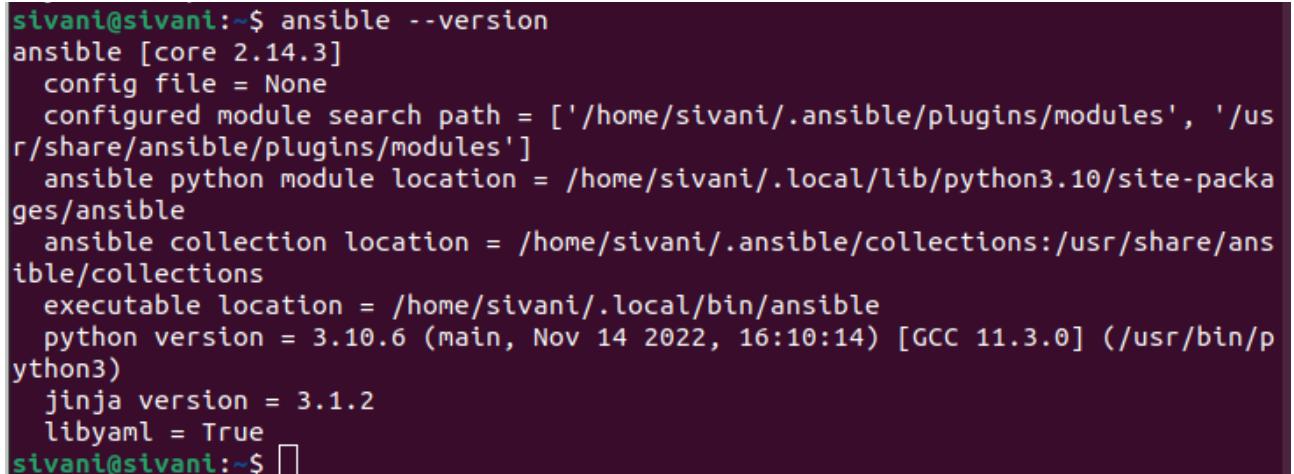
- `sudo apt-get install docker.io`
- `sudo apt install apt-transport-https ca-certificates curl`
- `sudo apt-get install software-properties-common`
- `curl -fsSL https://get.docker.com -o get-docker.sh`
- `bash get-docker.sh`
- `docker -v`



```
sivani@sivani:~$ docker -v
Docker version 20.10.21, build 20.10.21-0ubuntu1~22.04.2
sivani@sivani:~$ 
```

Ansible :

- *sudo apt-get update*
- *sudo apt install openssh-server*
- *ssh-keygen -t rsa*
- *ssh-copy-id <username>@<localhost IP>*
- *sudo apt-get install ansible*
- *pip3 install docker*
- *pip3 install ansible*
- *ansible --version*



```
sivani@sivani:~$ ansible --version
ansible [core 2.14.3]
  config file = None
  configured module search path = ['/home/sivani/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/sivani/.local/lib/python3.10/site-packages/ansible
  ansible collection location = /home/sivani/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/sivani/.local/bin/ansible
  python version = 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
sivani@sivani:~$ 
```

Ngrok :

- Sign up in <https://ngrok.com/>
- Download ngrok from: <https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz>
 - Then extract ngrok from the terminal: *sudo tar xvzf ~/Downloads/ngrok-stable-linux-amd64.tgz -C /usr/local/bin*
 - Copy Authtoken from: <https://dashboard.ngrok.com/get-started/your-authtoken>
 - Add Authtoken: *\$ngrok authtoken <token>* in terminal
 - Execute *\$ngrok http 8080* in terminal; copy the public ip address for your local host.

Project Workflow :

I am working this project on local repository(Calculator) directory. So no need to add paths in commands. All the commands are without path of the directory.

Git and Github :

For version control system we use git and github. Github helps in automation through jenkins integration. Git is an invaluable tool for implementing version control systems, enabling the preservation of all code versions and facilitating easy access to the required version when necessary. To leverage git in any project folder, the folder must first be initialized as a git repository. While git can be used locally, we push the code to a cloud-based management platform such as Github. This platform allows anyone with access, subject to the repository's privacy settings, to make modifications and contribute to the code. First create a github repository and clone that repository to local repository. We add code and make changes to the code in local repository. For this we use *IntelliJ* code editor. After that we add code into the staging area and then commit changes. Finally we push the code to the remote repository.

- To initialize repository in local : `git init`
- To see status of the repository : `git status`
- Clone remote repository into local : `git clone <remote repository url> <where to clone>`
- After adding files in local repository, we add to the staging area : `git add -A`
- Commit the changes : `git commit -m "msg"`
- To check log like who made changes and in which branch changes are made : `git log`
- Push to the remote repository : `git push`
- Add the link to the current local repository with the use of git remote add – `git remote add origin <link to github repo>`

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 ***

 sivanich9  / 

Great repository names are short and memorable. Need inspiration? How about **ubiquitous-doodle**?

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](#).

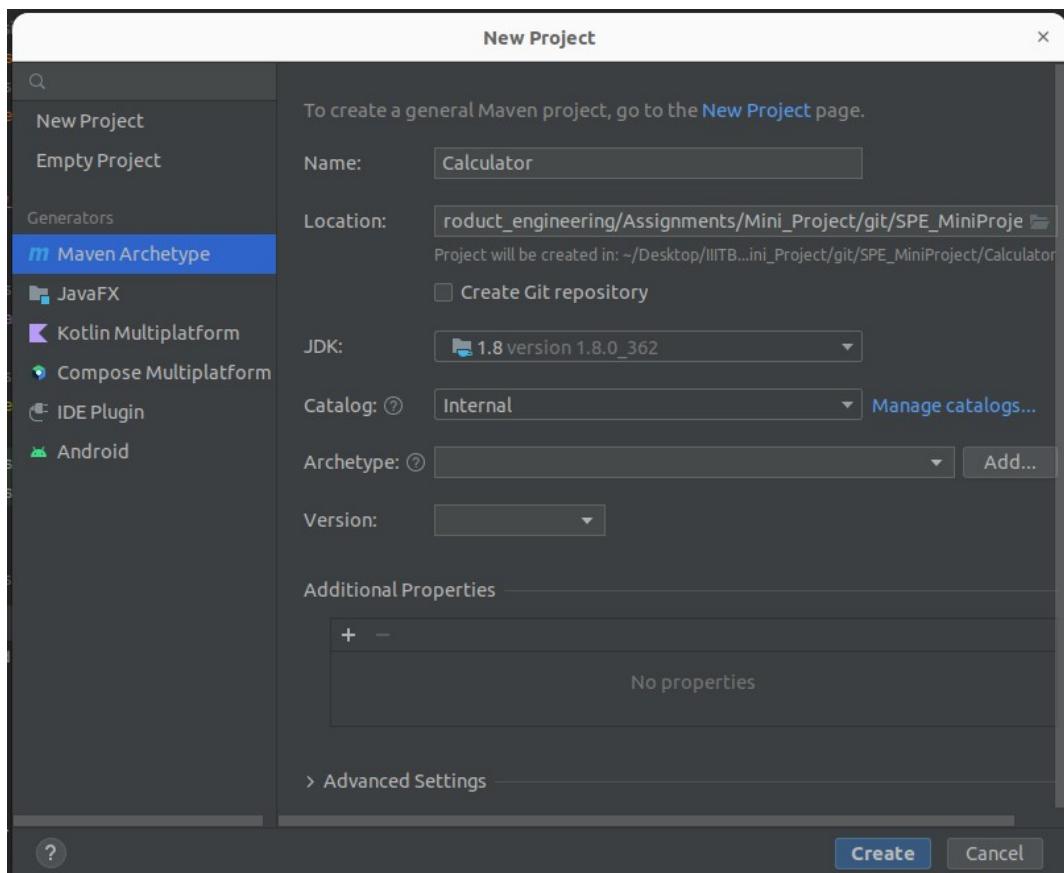
IntelliJ , Maven and JUnit :

We use java to build our project as it is system independent and we use IDE IntelliJ. Maven is a popular build automation tool that's mostly used with Java projects. This tool is based on Java and is used for developing applications. It allows us to include dependencies and create a jar file, which is a snapshot of our project that can be executed on any machine. By controlling project dependencies, handling compilation, packaging the code into a distributable format, and producing documentation, it simplifies the building process. Maven defines the project structure, dependencies, and build process using a Project Object Model (POM). The project's name, version, description, and dependencies are among the details found in the POM file (POM.xml). It is simple to obtain and manage project dependencies, including third-party libraries, using Maven's dependency management tool. We add dependencies of our project in pom.xml file. Maven simplifies the management of dependencies and assists users in transforming a project into a JAR file that can be easily transported and executed on different systems. Developers can carry out a variety of operations with the help of Maven's plugins, including deploying the program, creating code documentation, and performing tests.

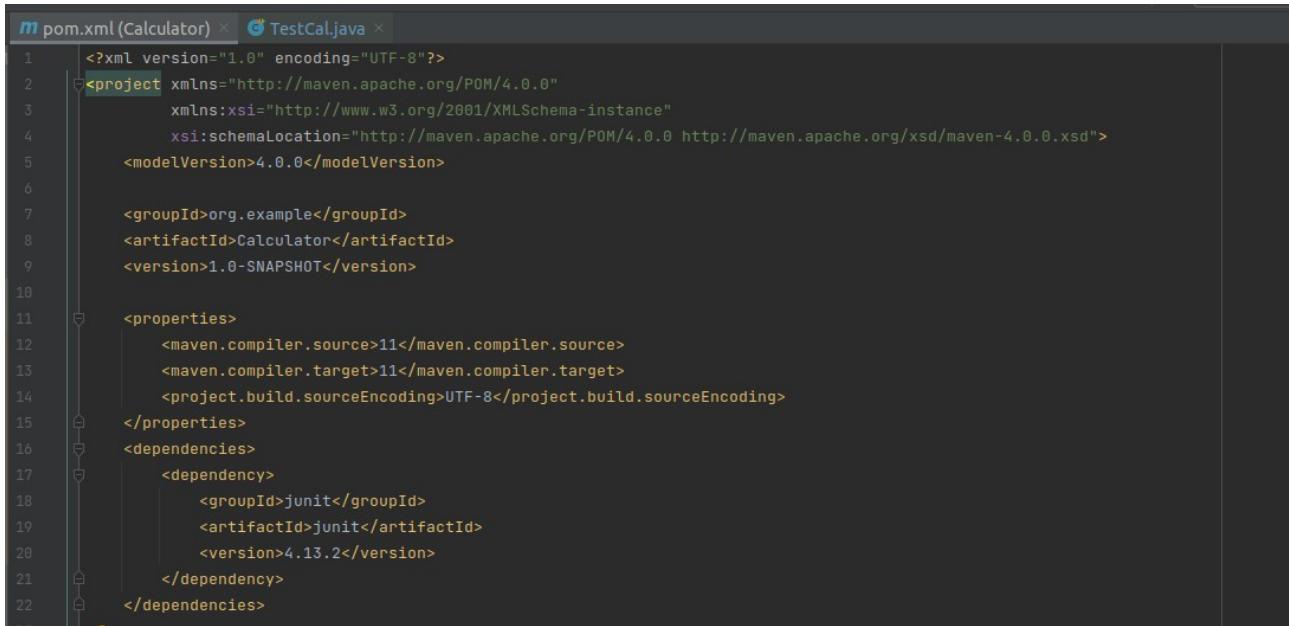
We test our code using JUnit. The project uses JUnit, a user-friendly unit testing tool. In my project, I've defined three tags: @before, @after, and @test. The @test tag is used for the annotation to specify that a method is a test method. Every time one of the @Test methods is called, @Before is done first. Following each @Test method, an @After annotation is executed. With this, a Test setup may be repeatedly set up and cleaned up after each test. To determine whether the expected value and the actual value are the same, we utilise assertEquals for true positive test cases. If they are not same, we have messages in assertEquals statement which gives messages that will be displayed if the test fails. Such messages help in identifying and resolving the failures quickly. Similarly we use assertNotEquals for false positive test cases.

- *mvn clean install* – To build the project
- The build project is stored in target folder of maven project
- *java -cp <jar-name> <class-path>* - To build the jar file
- *mvn clean test* (or) *maven test* - Maven runs the JUnit tests

Create new maven project in IntelliJ



Add JUnit dependencies in pom.xml file



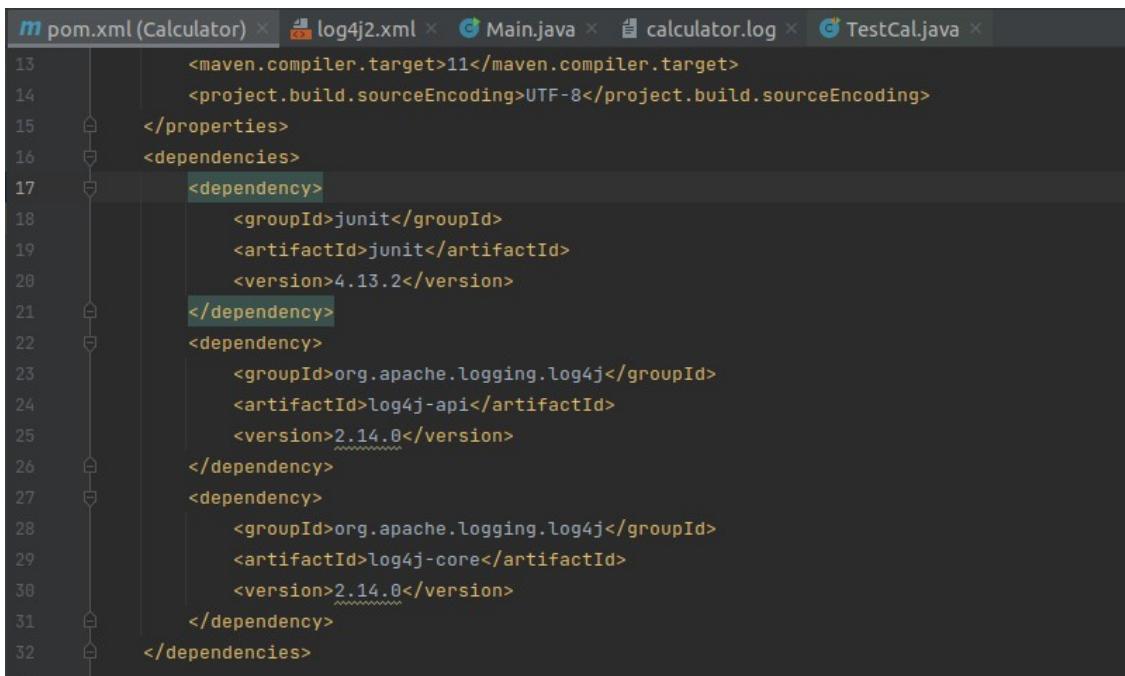
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>Calculator</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.13.2</version>
    </dependency>
  </dependencies>
```

Add log4j dependencies in pom.xml file

logging is very important as using logs we can see if any any undesirable behaviour is occurring, for backup purposes and lot more. For this feature we will be using Log4J with our code. Log4j is a Java-based logging utility that is widely used in software development to log information at various levels of severity. In Maven, Log4j can be included as a dependency in the project's pom.xml file, which allows developers to easily configure logging settings and output locations. By using Log4j with Maven, developers can efficiently debug and troubleshoot their applications, as well as monitor performance and diagnose errors.



```
<maven.compiler.target>11</maven.compiler.target>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
  </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>
```

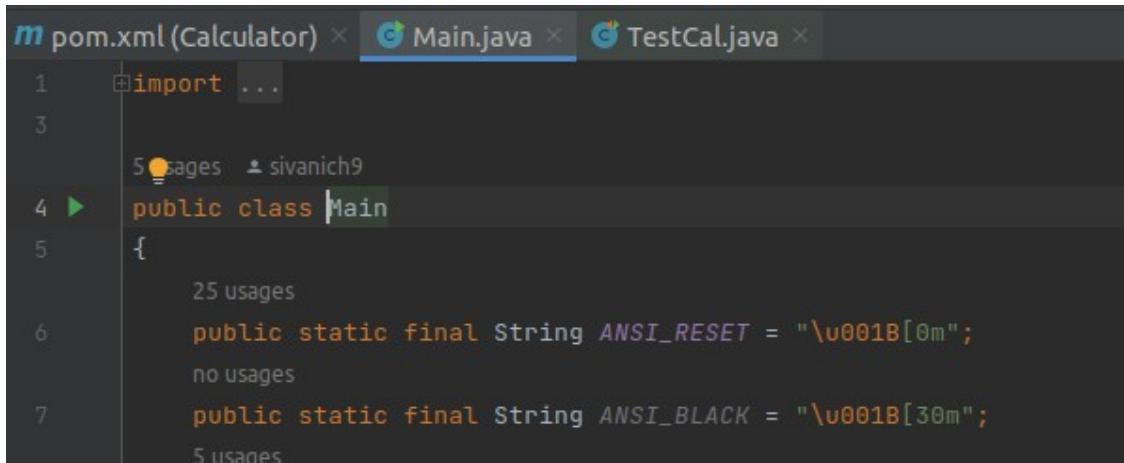
In essence, in the following step we are instructing the process to include all the dependencies within the jar file during its creation, eliminating the need to separately install these dependencies

on the system where the jar file will be executed. We are accomplishing this by utilizing the Maven Assembly Plugin.



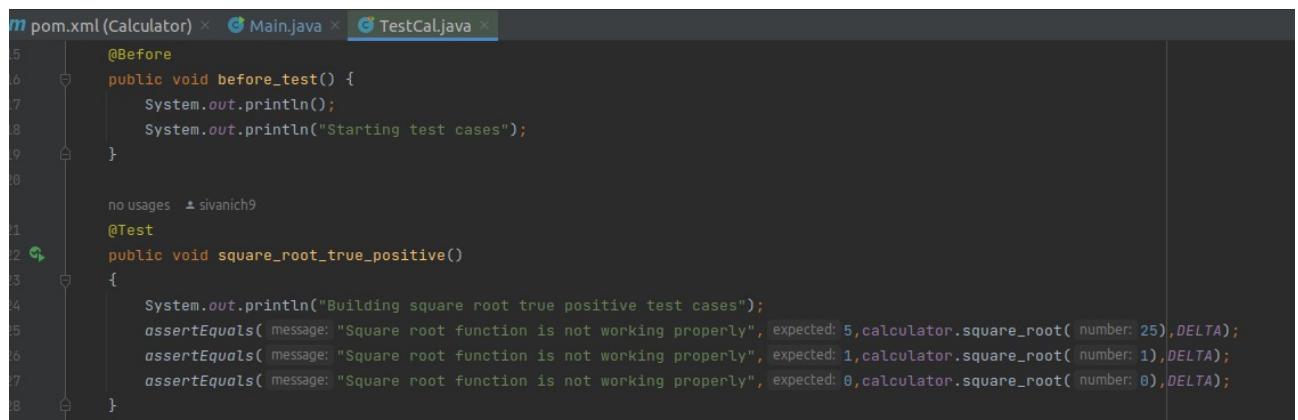
```
37 <groupId>org.apache.maven.plugins</groupId>
38 <artifactId>maven-assembly-plugin</artifactId>
39 <executions>
40   <execution>
41     <phase>package</phase>
42     <goals>
43       <goal>single</goal>
44     </goals>
45     <configuration>
46       <archive>
47         <manifest>
48           <mainClass>Main</mainClass>
49         </manifest>
50       </archive>
51     <descriptorRefs>
52       <descriptorRef>jar-with-dependencies</descriptorRef>
53     </descriptorRefs>
54   </configuration>
55 </execution>
56 </executions>
57 </plugin>
58 </plugins>
59 </build>
```

Code in main.java file



```
1 import ...
3
5 usages  sivanich9
4 ► public class Main
5 {
6
7   25 usages
8   public static final String ANSI_RESET = "\u001B[0m";
9   no usages
10  public static final String ANSI_BLACK = "\u001B[30m";
11  5 usages
```

Code in the test.java file



```
5 @Before
6 public void before_test() {
7   System.out.println();
8   System.out.println("Starting test cases");
9 }
10
11 no usages  sivanich9
12 @Test
13 public void square_root_true_positive()
14 {
15   System.out.println("Building square root true positive test cases");
16   assertEquals( message: "Square root function is not working properly", expected: 5,calculator.square_root( number: 25),DELTA);
17   assertEquals( message: "Square root function is not working properly", expected: 1,calculator.square_root( number: 1),DELTA);
18   assertEquals( message: "Square root function is not working properly", expected: 0,calculator.square_root( number: 0),DELTA);
19 }
```

we have to create “log4j.xml” in “<project-folder>/src/main/resources/log4j2.xml”.

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

In *Appenders*, in *PatternLayout* we have format like `%d{dd/MM/yyyy:HH:mm:ss SSS}{IST} [%F] [%level] %logger{36} %msg%n`

Denoting

1. First part is date
2. [%F] denotes the class which calls it
3. "%level" denotes type of log like INFO, ERROR
4. "%logger{36}" means in which class the logger object was created
5. "%msg" is the message passed to the logger while logging.

Add logger method in main.java and logger.info in methods of main.java

```
17
18     12 usages
19     private static final Logger logger = LogManager.getLogger(Main.class);
```

```
    }
    7 usages sivanich9
    public double square_root(double number)
    {
        try
        {
            logger.info("[SQRT] - " + number);
            if(number < 0)
            {
                logger.error("[ERROR - SQRT] - Square root function is given -ve number");
                throw new IllegalArgumentException(ANSI_RED+"Cannot calculate square root of negative number"+ANSI_RESET);
            }
            logger.info("[RESULT - SQRT] - " + Math.sqrt(number));
            return Math.sqrt(number);
        }
        catch (IllegalArgumentException e)
        {
            System.err.println(e.getMessage());
            return Double.NaN;
        }
    }
```

Run test cases using *mvn test*

```
Terminal: Local + ▾
-----
Running TestCal

Starting test cases
Building natural logarithm true positive test cases
Ending test cases

Starting test cases
Building square root false positive test cases
Ending test cases
```

```
Terminal: Local + ▾
Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.048 sec

Results :

Tests run: 8, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  0.863 s
[INFO] Finished at: 2023-03-14T21:52:25+05:30
[INFO] -----
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$
```

Build the project using *mvn clean install*

```
Terminal: Local + ▾
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ Calculator ---
[INFO] Installing /home/sivani/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator/Calculator-1.0-SNAPSHOT/Calculator-1.0-SNAPSHOT.jar
[INFO] Installing /home/sivani/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator/Calculator-1.0-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  6.094 s
[INFO] Finished at: 2023-03-14T21:47:17+05:30
[INFO] -----
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$
```

Run the jar file

```
Terminal: Local + ▾
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ java -cp ./target/Calculator-1.0-SNAPSHOT.jar Main

Welcome to the Scientific Calculator

Choose one of the following operations

1. Square root function - √x
2. Factorial function - x!
3. Natural logarithm (base e) - ln(x)
4. Power function - x^b
5. Quit
```

Add changes to the github repository

```
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ git add -A
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ git commit -m "added changes"
[main 085322] added changes
 7 files changed, 10 insertions(+), 8 deletions(-)
 rewrite target/Calculator-1.0-SNAPSHOT.jar (83%)
 rewrite target/classes/Main.class (85%)
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ git push
Username for 'https://github.com': sivanich9
Password for 'https://sivanich9@github.com':
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (17/17), 6.15 KiB | 899.00 KiB/s, done.
Total 17 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 7 local objects.
To https://github.com/sivanich9/Calculator.git
 ee012eb..0853222 main -> main
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$
```

Docker :

Docker is a popular open-source containerization platform that allows developers to package an application and its dependencies into a single, portable container that can run consistently across different computing environments.(e.g., development, testing, and production). Because of the many dependencies and versions of dependencies that must be supported, there will be a number of issues to be resolved when the product is delivered. The clients will find it very difficult to resolve. Docker is helpful in this situation since it enables us to correctly install all of the dependencies and then compress them into an image. As the image was uploaded to DockerHub for a similar reason as github, anyone can now download the image and use it for their own projects. Docker is used to make images through containerization.

First we sign in to the dockerhub and create one repository named calculator

The screenshot shows the Docker Hub interface for the repository 'sivani4/calculator'. At the top, there's a purple banner with the text 'Wasm is a fast, light alternative to Linux containers – try it out today with the Docker+Wasm Beta.' Below the banner, the header includes the Docker Hub logo, a search bar, navigation links for 'Explore', 'Repositories', 'Organizations', 'Help', and an 'Upgrade' button. On the right, there's a user profile icon for 'sivani4'. The main content area shows the repository details: 'sivani4' (owner), 'calculator' (name), and 'General' (tab selected). Below this, there are tabs for 'General', 'Tags', 'Builds', 'Collaborators', 'Webhooks', and 'Settings'. The 'General' tab section contains a description ('Scientific Calculator'), a timestamp ('Last pushed: a few seconds ago'), and a 'Docker commands' section with the command 'docker push sivani4/calculator:tagname'. There's also a 'Public View' button and a link to 'Using 0 of 1 private repositories'.

This is the docker file. On building this docker file, we will get docker image. Building this file, it helps us to install all dependencies and compress them into an image.

The following is the meaning of the Dockerfile commands:

- **FROM:** Imports the base image openjdk11 to create a new image.
- **COPY:** Copies a file from the current directory of the Dockerfile into the root directory of the image.
- **WORKDIR:** Changes the current working directory within the image.

- CMD: Runs a command inside the image.

```

1 FROM openjdk:11
2 COPY ./target/Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar .
3 WORKDIR /
4 CMD ["java", "-cp", "Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar", "Main"]

```

We build docker file to get a docker image. Command to build is : *sudo docker build -t <docker-username/reponame>* .

```

sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ sudo docker build -t sivani4/calculator .
Sending build context to Docker daemon 337.4kB
Step 1/4 : FROM openjdk:11
--> 47a932d998b7
Step 2/4 : COPY ./target/Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar .
--> Using cache
--> 6260bf113dc4
Step 3/4 : WORKDIR /
--> Using cache
--> 75f0084dec0d5
Step 4/4 : CMD ["java", "-cp", "Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar", "Main"]
--> Using cache
--> 812ed1e5214d
Successfully built 812ed1e5214d
Successfully tagged sivani4/calculator:latest

```

By using *sudo docker images*, we can see docker image sivani4/calculator is created.

```

sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ sudo docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
calculator      latest   812ed1e5214d  5 minutes ago  654MB
sivani4/calculator  latest   812ed1e5214d  5 minutes ago  654MB
my-java-app     latest   b884a62f7595  12 hours ago  654MB
<none>          <none>  3010c52d172d  12 hours ago  526MB
<none>          <none>  fdf8d3e36db8  13 hours ago  526MB
openjdk          8       b273004037cc  7 months ago  526MB
openjdk          11      47a932d998b7  7 months ago  654MB
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ 

```

To push our docker image to dockerhub, we first login into our dockerhub using *sudo docker login <docker-username/reponame>*

```

sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: sivani4
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ 

```

Then we push our docker image to dockerhub using *sudo docker push <docker-username/reponame:latest>*

```

sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ sudo docker push sivani4/calculator:latest
The push refers to repository [docker.io/sivani4/calculator]
8bf315119cd9: Pushed
7b7f3078e1db: Mounted from library/openjdk
826c3ddb29c: Mounted from library/openjdk
b626401ef603: Mounted from library/openjdk
9b5515abf26: Mounted from library/openjdk
293d5db30c9f: Mounted from library/openjdk
03127cd479b: Mounted from library/openjdk
9c742cd6c7a5: Mounted from library/openjdk
latest: digest: sha256:852ec9a6bb49241971a34ea2816c10d860f1b0023f6d2fe9f0cafcb88aeb69824 size: 2003
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ 

```

sivani4 / calculator

Description
Scientific Calculator

Last pushed: 13 hours ago

Docker commands

To push a new tag to this repository,

```
docker push sivani4/calculator:tagname
```

Tags

Tag	OS	Type	Pulled	Pushed
latest		Image	--	13 hours ago

See all [Go to Advanced Image Management](#)

IMAGE INSIGHTS INACTIVE [Activate](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds.](#)

Upgrade

We run the container using command `docker run -it <docker-username/repo-name:tagname>`

```
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ sudo docker run -it sivani4/calculator
Welcome to the Scientific Calculator
Choose one of the following operations
1. Square root function - √x
2. Factorial function - x!
3. Natural logarithm (base e) - ln(x)
4. Power function - x^b
5. Quit
5
You have chosen the operation 'Quit'
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$
```

Ansible :

Continuous deployment is a technique for automatically distributing software to lots of client computers. Ansible is a configuration management tool. However anything may be sent to a lot of "controlled hosts" using this method. On the control node, Ansible is installed and maintained. All copies of your Ansible project files and configuration data are kept on a control node. Managed hosts are computers that deliver the application or infrastructure as code. Ansible serves as a central distributor to all managed hosts from the control node. It is possible to view a list of managed hosts in the inventory file. Here, I'm using my localhost as the host rather than another VM or machine.

We create p2.yml. It is the playbook that contains the configuration and infrastructure as code.

Inventory file has the list of target servers we want to manage. I am using my localhost only.

```

p2.yml                               inventory
1 ---                                 x
2 - name: Copy jar file to inventory i.e. Pull docker image
3   hosts: all
4   tasks:
5     - name: Pull junit devops image
6       docker_image:
7         name: sivani4/calculator
8         source: pull
9
10
11   |
12   - name: running container
13     shell: docker run -it -d sivani4/calculator /bin/bash

```

```

inventory                               p2.yml
1 localhost ansible_user=sivani ansible_connection=local

```

The command "`ansible-playbook p2.yml -i inventory`" will run an Ansible playbook called "p2.yml" using an inventory file called "inventory".

- "ansible-playbook" is the command that tells Ansible to execute a playbook.
- "p2.yml" is the name of the playbook file that will be executed.
- "-i inventory" specifies the inventory file that Ansible should use to determine which hosts to run the playbook on. The inventory file contains a list of hosts and their associated configuration information, such as IP addresses and connection details.

```

sivani@sivani: ~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assign...
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ ansible-playbook p2.yml -i inventory

PLAY [Copy jar file to inventory i.e. Pull docker image] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Pull junit devops image] ****
ok: [localhost]

TASK [running container] ****
changed: [localhost]

PLAY RECAP ****
localhost                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
                           ignored=0

sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ 

```

```
sivani@sivani: ~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/
Calculator$ sudo docker images
[sudo] password for sivani:
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
sivani4/calculator    latest    85b972bfc419  3 days ago   654MB
sivani4/calculator    <none>   1c0f930a1959  3 days ago   654MB
sivani4/calculator    <none>   a628d3ae4891  3 days ago   654MB
sivani4/calculator    <none>   85592cffa270  3 days ago   654MB
sivani4/calculator    <none>   7f863246f43e  3 days ago   654MB
sivani4/calculator    <none>   4be5f1bd139c  3 days ago   654MB
sivani4/calculator    <none>   48ad9e433946  3 days ago   654MB
sivani4/calculator    <none>   ba075d4cac56  3 days ago   654MB
sivani4/calculator    <none>   6a85ed2ba139  3 days ago   654MB
sivani4/calculator    <none>   8821aa6f7dc4  3 days ago   654MB
sivani4/calculator    <none>   b919567395e8  3 days ago   654MB
sivani4/calculator    <none>   377adebf58a5  5 days ago   654MB
calculator          latest    812ed1e5214d  6 days ago   654MB
sivani4/calculator    <none>   812ed1e5214d  6 days ago   654MB
my-java-app          latest    b884a62f7595  7 days ago   654MB
<none>              <none>   3010c52d172d  7 days ago   526MB
<none>              <none>   fdf8d3e36db8  7 days ago   526MB
openjdk              8        b273004037cc  7 months ago  526MB
openjdk              11       47a932d998b7  7 months ago  654MB
sivani@sivani: ~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/
/Calculator$
```

Ngrok and Github WebHooks :

Whenever the developer builds the code and commit it and push that to Github, webhooks automatically build the code in Jenkins. To perform a webhook, the private IP address of the local machine needs to be converted to a public IP address.

Automated messages known as "webhooks" are triggered when changes occur. Specifically, in our situation, updates made to the GitHub repository will activate the Jenkins pipeline through the webhook.

Ngrok is a tool that establishes secure tunnels to connect local servers located behind Network Address Translation (NAT) and firewalls to the internet. It features a web interface that provides real-time monitoring of any HTTP traffic that passes through the tunnels. This tool enables users to access the internet via a web server running on their local system, with the port on which the web server is listening to being specified to ngrok.

To start ngrok, use command : `ngrok http://jenkins-server:8080`

```
sivani@sivani: ~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/
sivani@sivani: ~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/
Calculator$ ngrok http http://172.16.144.241:8080/
```

```
sivani@sivani: ~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assgnm... (Ctrl+C to quit)
ngrok
Add OAuth and webhook security to your ngrok (its free!): https://ngrok.com/free

Session Status          online
Account                 Sivani Channamsetty (Plan: Free)
Version                3.2.1
Region                 India (in)
Latency
Web Interface          http://127.0.0.1:4040
Forwarding             https://3a59-103-156-19-229.in.ngrok.io -> http://172.16.144.241:8080/
Connections
ttl      opn      rt1      rt5      p50      p90
0       0       0.00    0.00    0.00    0.00
```

Setting up WebHooks :

Navigate to the settings section of the Github repository, then proceed to the Webhooks tab. Add the ngrok address in payload URL and the personal access token in secret

The screenshot shows the 'General' tab selected in the GitHub repository settings. The left sidebar lists various repository settings: Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Actions), Webhooks (selected), Environments, Codespaces, Pages, Security, and Code security and analysis. The main content area displays the 'General' configuration. It includes fields for Repository name (Calculator) and a 'Rename' button. There are two unchecked checkboxes: 'Template repository' (described as letting users generate new repositories with the same directory structure and files) and 'Require contributors to sign off on web-based commits' (described as requiring contributors to sign off on commits made through GitHub's web interface). Below these is the 'Social Preview' section, which instructs users to upload an image to customize their repository's social media preview. A note states that images should be at least 640x320px (1280x640px for best display) and provides a 'Download template' link.

General

Webhooks

Add webhook

Access

Collaborators

Moderation options

Code and automation

- Branches
- Tags
- Actions
- Webhooks**
- Environments
- Codespaces
- Pages

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://9893-119-161-98-68.in.ngrok... (push)

Edit Delete

Run ngrok in terminal and copy the ngrok public ip address

```
sivani@sivani: ~/Desktop/IITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assgnm...
ngrok
Add OAuth and webhook security to your ngrok (its free!): https://ngrok.com/free

Session Status      online
Account            Sivani Channamsetty (Plan: Free)
Version            3.2.1
Region             India (in)
Latency            29ms
Web Interface     http://127.0.0.1:4040
Forwarding         https://fa0f-119-161-98-68.in.ngrok.io -> http://172.16.144.241:8080/
Connections        ttl     opn     rt1     rt5     p50     p90
                  0       0     0.00    0.00    0.00    0.00
```

General

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

SSL verification

By default, we verify SSL certificates when delivering payloads.

Enable SSL verification **Disable (not recommended)**

Which events would you like to trigger this webhook?

Just the push event.

Send me **everything**.

Let me select individual events.

Active

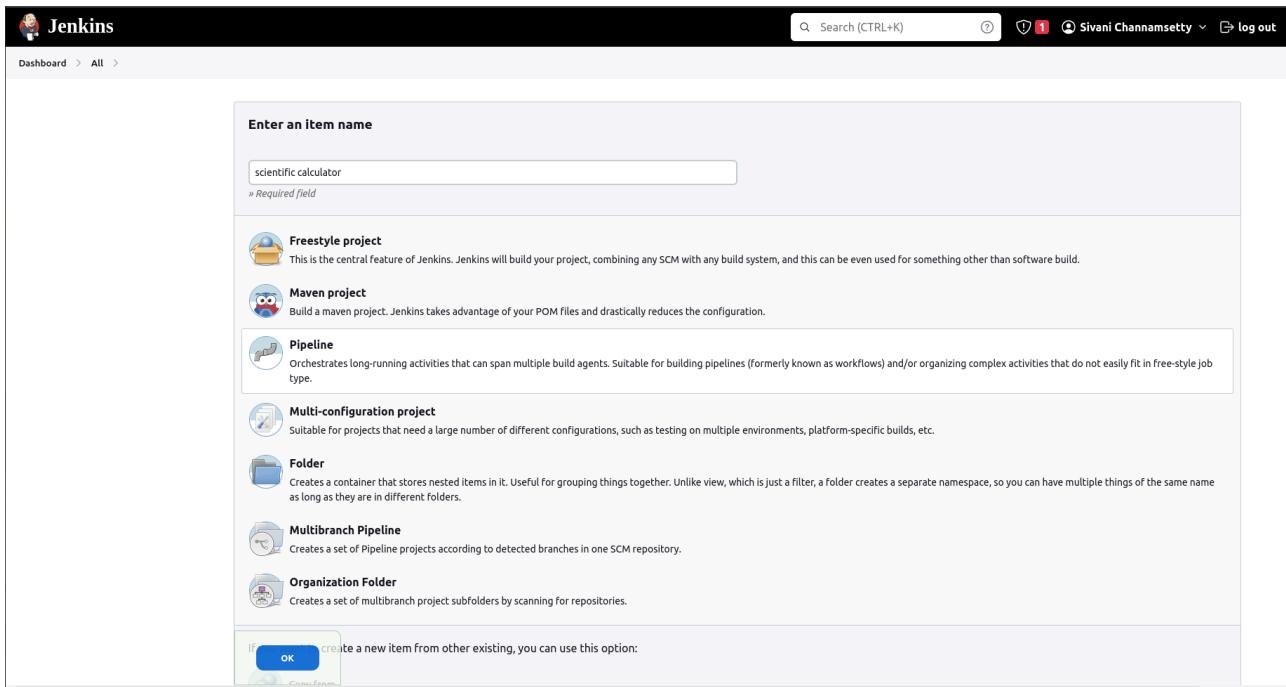
We will deliver event details when this hook is triggered.

Add webhook

Jenkins :

An open-source automation server called Jenkins helps in automating a few steps in the software development process. It's a popular tool that development teams use to automate their build, test, and deployment processes as part of continuous integration and continuous delivery (CI/CD). Developers can use Jenkins to automatically build, test, and deploy their code to several environments, including development, staging, and production. To build a full CI/CD pipeline, Jenkins may also integrate with a variety of different tools and technologies. Jenkins can integrate with a variety of technologies, including version control systems, testing frameworks, and deployment tools, and employs plugins to expand its capability. It is compatible with a variety of operating systems, including Windows, Linux, and macOS. We need docker, github, maven, ansible and build pipeline plugins. So we download all plugins in jenkins.

First create jenkins project and select pipeline



The screenshot shows the Jenkins dashboard with a search bar containing "scientific calculator". Below the search bar, there is a list of project types:

- 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.
- 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 separate namespace, so you can have multiple things of the same name 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.

At the bottom of the dialog, there is a note: "If you want to create a new item from other existing, you can use this option:" followed by an "OK" button.

Add dockerhub credentials in jenkins

Go to Manage Jenkins > Manage Credentials > System > Global credentials (unrestricted) > Add Credentials

Choose Kind as Username with password and Scope as Global; provide your docker username and password with a unique ID of your own. This ID is used for referring the username with a password later in your pipelines.

Jenkins

Dashboard >

+ New Item

All +

People Build History Manage Jenkins My Views

Build Queue Build Executor Status

No builds in the queue.

1 Idle 2 Idle

Last Success Last Failure Last Duration

Icon: S W Name: Calculator Last Success: 1 day 6 hr #6 Last Failure: 17 hr #9 Last Duration: 18 sec

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

REST API Jenkins 2.387.1

172.16.144.241:8080/manage

Dashboard > Manage Jenkins

People Build History Manage Jenkins My Views

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

Set up agent Set up cloud Dismiss

Build Queue Build Executor Status

No builds in the queue.

1 Idle 2 Idle

System Configuration

Configure System Global Tool Configuration Manage Plugins Manage Nodes and Clouds

Configure global settings and paths. Configure tools, their locations and automatic installers. Add, remove, disable or enable plugins that can extend the functionality of Jenkins. Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

Configure Global Security Manage Credentials Configure Credential Providers Manage Users

Secure Jenkins; define who is allowed to access/use the system. Configure credentials Configure the credential providers and types Create/delete/modify users that can log in to this Jenkins.

Status Information

System Information System Log Load Statistics About Jenkins

Displays various environmental information to assist troubleshooting. System log captures output from java.util.logging output related to Jenkins. Check your resource utilization and see if you need more computers for your builds. See the version and license information.

Troubleshooting

Manage Old Data

Scrub configuration files to remove remnants from old plugins and earlier versions.

172.16.144.241:8080/manage/credentials

Jenkins

Dashboard > Manage Jenkins > Credentials > System >

System

+ Add domain

Domain	Description
Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching.

Icon: S M L

172.16.144.241:8080/manage/credentials/store/system/domain/ REST API Jenkins 2.387.1

Jenkins

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			

Icon: S M L

172.16.144.241:8080/manage/credentials/store/system/domain/_/newCredentials REST API Jenkins 2.387.1

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: sivani4

Treat username as secret:

Password: *****

ID: dockercred

Description:

Create

REST API Jenkins 2.387.1

Install the following plugins : Maven, Git, Ansible and Docker.

Go to *Dashboard* and the *Manage Jenkins* and then *Manage Plugins*

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

System Configuration

- Configure System: Configure global settings and paths.
- Global Tool Configuration: Configure tools, their locations and automatic installers.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Set up agent
- Set up cloud
- Dismiss

Security

- Configure Global Security: Secure Jenkins; define who is allowed to access/use the system.
- Manage Credentials: Configure credentials.
- Configure Credential Providers: Configure the credential providers and types.
- Manage Users: Create/delete/modify users that can log in to this Jenkins.

Status Information

- System Information: Displays various environmental information to assist troubleshooting.
- System Log: System log captures output from java.util.logging output related to Jenkins.
- Load Statistics: Check your resource utilization and see if you need more computers for your builds.
- About Jenkins: See the version and license information.

Troubleshooting

Install the plugins required

Name	Enabled
Ansible plugin 148.v6b_13c6de3a_47 Invoke Ansible Ad-Hoc commands and playbooks. Report an issue with this plugin	<input checked="" type="checkbox"/> <input type="checkbox"/>
Ansible Tower Plugin 0.16.0 This plugin connects Jenkins with Ansible Tower Report an issue with this plugin	<input checked="" type="checkbox"/> <input type="checkbox"/>
Ant Plugin 481.v7b_09e53bfcca Adds Apache Ant support to Jenkins Report an issue with this plugin	<input checked="" type="checkbox"/> <input type="checkbox"/>
Apache HttpClient 4.x API Plugin 4.5.14-150.v7a_b_9d17134a_5 Bundles Apache HttpClient 4.x and allows it to be used by Jenkins plugins. Report an issue with this plugin	<input checked="" type="checkbox"/> <input type="checkbox"/>
Authentication Tokens API Plugin 1.4 This plugin provides an API for converting credentials into authentication tokens in Jenkins. Report an issue with this plugin	<input checked="" type="checkbox"/> <input type="checkbox"/>
Bootstrap 5 API Plugin 5.2.2-1 Provides Bootstrap 5 for Jenkins Plugins. Bootstrap is (according to their self-perception) the world's most popular front-end component library to build responsive, mobile-first projects on the web. Report an issue with this plugin	<input checked="" type="checkbox"/> <input type="checkbox"/>
bouncycastle API Plugin 2.27 This plugin provides a stable API to Bouncy Castle related tasks. Report an issue with this plugin	<input checked="" type="checkbox"/> <input type="checkbox"/>

Select *Github hook trigger for GITScm polling*

Configure

General

- Discard old builds ?
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts
- GitHub project
- Pipeline speed/durability override ?
- Preserve stashes from completed builds ?
- This project is parameterized ?
- Throttle builds ?

Build Triggers

- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?
- Quiet period ?
- Trigger builds remotely (e.g., from scripts) ?

Advanced Project Options

Advanced ▾

Pipeline

Save **Apply**

Add pipeline script for stages *Git pull, Maven test, Maven build, Docker image creation , Deploying/Pushing docker image and Ansible deploy.*

Jenkins pipeline :

```
1 pipeline {
2     agent any
3
4     stages {
5         stage('Git Pull stage') {
6             steps {
7                 git url: 'https://github.com/sivanich9/Calculator',
8                     branch: 'main'
9
10            }
11        }
12        stage('Maven test'){
13            steps{
14                script{
15                    sh 'mvn clean test'
16                }
17            }
18        }
19        stage('Maven Build'){
20            steps{
21                script{
22                    sh 'mvn clean install'
23                }
24            }
25        }
26        stage('Docker Build Image'){
27            steps{
28                script{
29                    sh 'docker build -t sivani4/calculator:latest .'
30                }
31            }
32        }
33        stage('Push Docker Image'){
34            steps{
35                script{
36                    withDockerRegistry([ credentialsId: "dockercred", url: "" ])
37                    {
38                        sh 'docker push sivani4/calculator:latest'
39                    }
40                }
41            }
42        }
43        stage("Ansible deploy") {
44            steps {
45                script [
46                    // sh "ssh-keygen -t rsa"
47                    // sh "ssh-copy-id sivani@localhost"
48                    // ansiblePlaybook disableHostKeyChecking: true, installation: 'ansible_1', inventory: 'inventory', playbook: 'p2.yml'
49                    sh "/usr/bin/pip3 install docker"
50                    sh "ansible-playbook p2.yml -i inventory"
51                ]
52            }
53        }
54    }
55}
```

Code :

```
pipeline {

    agent any

    stages {
        stage('Git Pull stage') {
            steps {
                git url: 'https://github.com/sivanich9/Calculator',
                    branch: 'main'
            }
        }
        stage('Maven test'){
            steps{
                script{
                    sh 'mvn clean test'
                }
            }
        }
        stage('Maven Build'){
            steps{
                script{
                    sh 'mvn clean install'
                }
            }
        }
        stage('Docker Build Image'){
            steps{
                script{
                    sh 'docker build -t sivani4/calculator:latest .'
                }
            }
        }
        stage('Push Docker Image'){
            steps{
                script{
                    withDockerRegistry([ credentialsId: "dockercred", url: "" ])
                    {
                        sh 'docker push sivani4/calculator:latest'
                    }
                }
            }
        }
        stage("Ansible deploy") {
            steps {
                script {
                    // sh "ssh-keygen -t rsa"
                    // sh "ssh-copy-id sivani@localhost"
                    // ansiblePlaybook disableHostKeyChecking: true, installation: 'ansible_1', inventory:
                    'inventory', playbook: 'p2.yml'
                    sh "/usr/bin/pip3 install docker"
                }
            }
        }
    }
}
```

```
        sh "ansible-playbook p2.yml -i inventory"
    }
}
}
}
```

Dashboard > Calculator > Configuration

Configure

Advanced Project Options

General Advanced Project Options Pipeline

Pipeline

Definition

Pipeline script

```
Script ?  
1< pipeline {  
2   agent any  
3  
4   stages {  
5     stage('Git Pull stage') {  
6       steps {  
7         git url: 'https://github.com/sivanich9/Calculator',  
8             branch: 'main'  
9       }  
10    }  
11    stage('Maven test'){  
12      steps {  
13        script{  
14          sh "mvn clean test"  
15        }  
16      }  
17    }  
18    stage('Maven Build'){  
19      steps {  
20        script{  
21          sh "mvn clean install"  
22        }  
23      }  
24    }  
25  }  
26}>
```

Use Groovy Sandbox ?

[Pipeline Syntax](#)

Save Apply

I added code to Github. As we set Github WebHooks, jenkins automatically build the code. Note that we have to run ngrok in our terminal. As you come out of ngrok, it doesn't build. You have to again run ngrok in our terminal then the address and accordingly we have to update the webhook in Github.

Jenkins

Dashboard > Calculator >

Status Changes Build Now Configure Delete Pipeline Full Stage View Rename Pipeline Syntax GitHub Hook Log Build History trend Filter builds... #27 Mar 21, 2023, 6:00 PM #26 Mar 21, 2023, 5:57 PM #25 Mar 21, 2023, 3:45 PM #24 Mar 17, 2023, 11:33 PM #23 Mar 17, 2023, 11:32 PM

Pipeline Calculator

Add description Disable Project

Stage View

	Git Pull stage	Maven test	Maven Build	Docker Build Image	Push Docker Image	Ansible deploy
#27 Mar 21, 18:00 1 commit	2s	7s	9s	9s	23s	12s
#26 Mar 21, 17:57 1 commit	3s	8s	22s	14s	40s	11s
#25 Mar 21, 15:45 3 commits	3s	10s	7s	29s	30s	27s
#24 Mar 21, 23:33 No Changes	5s	10s	11s	14s	23s	37s
#23 Mar 17, 23:32 1 commit	1s	6s	9s	5s	21s	18s
	2s	5s	7s	5s	19s	7s failed

Jenkins

Dashboard > Calculator >

Pipeline Calculator

- </> Changes
- ▷ Build Now
- ⚙ Configure
- trash Delete Pipeline
- 🔍 Full Stage View
- edit Rename
- info Pipeline Syntax
- git GitHub Hook Log

Stage View

	Git Pull stage	Maven test	Maven Build	Docker Build Image	Push Docker Image	Ansible deploy
Average stage times: (Average full run time: ~1min 6s)	1s	6s	8s	5s	20s	6s
#26 Mar 17 23:33	1s	6s	9s	5s	21s	18s
#23 Mar 17 23:32 1 commit	2s	5s	7s	5s	19s	7s failed
#22 Mar 17 23:30 No changes	1s	5s	7s	5s	20s	2s failed
#21 Mar 17 23:28 No changes	1s	6s	8s	5s	20s	2s failed
#20 Mar 17 23:23 No changes	1s	6s	8s	5s	20s	8s failed
#19 Mar 17 No	1s	7s	8s	7s	21s	3s

Add description | Disable Project

Jenkins

Dashboard > Calculator >

Pipeline Calculator

- </> Changes
- ▷ Build Now
- ⚙ Configure
- trash Delete Pipeline
- 🔍 Full Stage View
- edit Rename
- info Pipeline Syntax
- git GitHub Hook Log

Stage View

	Git Pull stage	Maven test	Maven Build	Docker Build Image	Push Docker Image	Ansible deploy
Average stage times: (Average full run time: ~1min 34s)	2s	7s	8s	6s	20s	9s
#25 Mar 21 15:45 3 commits	5s	16s	11s	14s	23s	37s
#24 Mar 17 23:33 No changes	1s	6s	9s	5s	21s	18s
#23 Mar 17 23:32 1 commit	2s	5s	7s	5s	19s	7s failed
#22 Mar 17 23:30 No changes	1s	5s	7s	5s	20s	2s failed
#21 Mar 17 23:28 No changes	1s	6s	8s	5s	20s	2s failed
#20 Mar 17 No	1s	6s	8s	5s	20s	8s

Add description | Disable Project

Now go to terminal and find docker image and run it. We will be able to run our project.

```
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ sudo docker images
[sudo] password for sivani:
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
sivani4/calculator    latest   91e8ad54dbce  2 minutes ago  654MB
sivani4/calculator    <none>  85b972bfc419  3 days ago   654MB
sivani4/calculator    <none>  1c0f930a1959  3 days ago   654MB
sivani4/calculator    <none>  a628d3ae4891  3 days ago   654MB
sivani4/calculator    <none>  85592cffa270  3 days ago   654MB
sivani4/calculator    <none>  7f863246f43e  3 days ago   654MB
sivani4/calculator    <none>  4be5f1bd139c  3 days ago   654MB
sivani4/calculator    <none>  48ad9e433946  3 days ago   654MB
sivani4/calculator    <none>  ba075d4cac56  3 days ago   654MB
sivani4/calculator    <none>  6a85ed2ba139  3 days ago   654MB
sivani4/calculator    <none>  8821aa6f7dc4  3 days ago   654MB
sivani4/calculator    <none>  b919567395e8  3 days ago   654MB
sivani4/calculator    <none>  377adefbf58a5  5 days ago   654MB
calculator           latest   812ed1e5214d  6 days ago   654MB
sivani4/calculator    <none>  812ed1e5214d  6 days ago   654MB
my-java-app          latest   b884a62f7595  7 days ago   654MB
<none>                <none>  3010c52d172d  7 days ago   526MB
<none>                <none>  fdf8d3e36db8  7 days ago   526MB
openjdk              8        b273004037cc  7 months ago  526MB
openjdk              11       47a932d998b7  7 months ago  654MB
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ 
```

```
sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ sudo docker run -it sivani4/calculator:latest
Welcome to the Scientific Calculator

Choose one of the following operations

1. Square root function - √x
2. Factorial function - x!
3. Natural logarithm (base e) - ln(x)
4. Power function - x^b
5. Quit

5

You have chosen the operation 'Quit'

sivani@sivani:~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ 
```

ELK-Stack :

Once you finish deploying your software, the next crucial step is to continuously monitor its performance. This involves verifying whether the software is functioning according to its intended design. The ELK stack offers a monitoring tool for any deployed software, which involves analyzing its logs. This analysis can then be accessed and visualized on the Kibana dashboard.

Go to <https://cloud.elastic.co/login> and create a deployment

The screenshot shows the Elasticsearch Cloud Service dashboard. At the top, there's a navigation bar with the Elastic logo, a 'Cloud' button, and a 'Trial - 9 days left' message. Below the navigation is a table for managing deployments:

Deployment name	Status	Version	Cloud region	Manage deployment
calculator	Healthy	8.6.2	GCP - Iowa (us-central1)	[Edit]

On the left side, there are sections for 'Documentation' (with links to 'Elastic documentation', 'Indexing data into Elasticsearch', and 'Elasticsearch REST API'), 'Support' (with a 'Contact support' button), and 'Community' (with links to 'Join an ElasticON event', 'Introduction to Vector Search and Modern NLP', and 'Kibana Workshop'). On the right side, there are sections for 'Cloud status' (showing 'All systems operational'), 'News' (with articles like 'Insights into chess game trends' and 'Importing 4 billion chess games'), and 'Training' (with a 'Get started with our free training' section and a 'Elastic Learning Portal' button). A search bar at the top is also visible.

And go to that deployment and then upload a file

The screenshot shows the deployment interface for the 'calculator' deployment. At the top, it says 'Welcome home'. Below that are four main service icons: Enterprise Search (yellow), Observability (pink), Security (teal), and Analytics (blue). Each icon has a brief description. Below the icons is a section titled 'Get started by adding integrations' with three buttons: 'Add integrations', 'Try sample data', and 'Upload a file'. To the right of this is a decorative graphic of a house with various charts and graphs. At the bottom, there are several management links: 'Management' (with 'Manage permissions', 'Monitor the stack', 'Back up and restore', and 'Manage index lifecycles'), 'Dev Tools', and 'Stack Management'.

The screenshot shows the Elasticsearch interface with the title "More ways to add data". Below it, a sub-section titled "Visualize data from a log file" is displayed. This section includes instructions to "Upload your file, analyze its data, and optionally import the data into an Elasticsearch index." It lists supported file formats: Delimited text files (CSV, TSV), Newline-delimited JSON, and Log files with timestamp. A note states "You can upload files up to 100 MB." Below these instructions is a file upload area with a "Select or drag and drop a file" placeholder and a "No file chosen" button.

The screenshot shows a "Open File" dialog box. The left sidebar lists recent locations: Home, Desktop, Documents, Downloads, Music, Pictures, Videos, and exams. The main area shows a file tree under "Documents" with a list of files. The file "calculator.log" is highlighted with a red selection bar. The file list includes:

Name	Type	Modified
calculator.log	Text	17:54
Dockerfile	Text	17:54
inventory	Text	10:44
Jenkinsfile	Text	17:59
p2.yml	Text	18:00
pom.xml	Text	Wed
README.md	Text	10:56

At the bottom of the dialog, there is a checkbox for "Open files read-only".

Calcurtor.log file contains logs

```

sivant@sivant: ~/Desktop/IIITB/4_SEM_7_8/SEM_8/Software_product_engineering/Assignments/Mini_Project/git/Calculator$ cat calculator.log
16/03/2023:13:40:09 094 [Main.java] [INFO] Main [Ln] - 1.0
16/03/2023:13:40:09 120 [Main.java] [INFO] Main [RESULT - Ln] - 0.0
16/03/2023:13:40:09 122 [Main.java] [INFO] Main [SQRT] - 25.0
16/03/2023:13:40:09 123 [Main.java] [INFO] Main [RESULT - SQRT] - 5.0
16/03/2023:13:40:09 124 [Main.java] [INFO] Main [SQRT] - 1.0
16/03/2023:13:40:09 124 [Main.java] [INFO] Main [RESULT - SQRT] - 1.0
16/03/2023:13:40:09 124 [Main.java] [INFO] Main [SQRT] - 0.0
16/03/2023:13:40:09 125 [Main.java] [INFO] Main [RESULT - SQRT] - 0.0
16/03/2023:13:40:09 126 [Main.java] [INFO] Main [SQRT] - 25.0
16/03/2023:13:40:09 126 [Main.java] [INFO] Main [RESULT - SQRT] - 5.0
16/03/2023:13:40:09 127 [Main.java] [INFO] Main [SQRT] - 1.0
16/03/2023:13:40:09 127 [Main.java] [INFO] Main [RESULT - SQRT] - 1.0
16/03/2023:13:40:09 127 [Main.java] [INFO] Main [SQRT] - 0.0
16/03/2023:13:40:09 127 [Main.java] [INFO] Main [RESULT - SQRT] - 0.0
16/03/2023:13:40:09 128 [Main.java] [INFO] Main [FACT] - 8.0
16/03/2023:13:40:09 129 [Main.java] [INFO] Main [RESULT - FACT] - 40320.0
16/03/2023:13:40:09 129 [Main.java] [INFO] Main [FACT] - 3.0
16/03/2023:13:40:09 129 [Main.java] [INFO] Main [RESULT - FACT] - 6.0
16/03/2023:13:40:09 129 [Main.java] [INFO] Main [FACT] - 0.0
16/03/2023:13:40:09 130 [Main.java] [INFO] Main [RESULT - FACT] - 1.0
16/03/2023:13:40:09 137 [Main.java] [INFO] Main [POW] - 2.0 5.0
16/03/2023:13:40:09 138 [Main.java] [INFO] Main [RESULT - POW] - 32.0
16/03/2023:13:40:09 138 [Main.java] [INFO] Main [POW] - 9.0 0.0
16/03/2023:13:40:09 139 [Main.java] [INFO] Main [RESULT - POW] - 1.0
16/03/2023:13:40:09 139 [Main.java] [INFO] Main [POW] - 3.0 2.0
16/03/2023:13:40:09 139 [Main.java] [INFO] Main [RESULT - POW] - 9.0
16/03/2023:13:40:09 140 [Main.java] [INFO] Main [Ln] - 15.0
16/03/2023:13:40:09 140 [Main.java] [INFO] Main [RESULT - Ln] - 2.70805020110221
16/03/2023:13:40:09 141 [Main.java] [INFO] Main [RESULT - Ln] - 3.367295829986474
16/03/2023:13:40:09 141 [Main.java] [INFO] Main [Ln] - 1.0
16/03/2023:13:40:09 142 [Main.java] [INFO] Main [RESULT - Ln] - 0.0
16/03/2023:13:40:09 142 [Main.java] [INFO] Main [FACT] - 0.0

```

After that override settings to change grok pattern according to our log file

The screenshot shows the Elasticsearch interface with the file 'calculator.log' uploaded. The 'File contents' section displays the first 175 lines of the log file. The 'Summary' section provides an overview of the analyzed file, including the number of lines (175), format (semi_structured_text), and the current Grok pattern used for parsing the logs.

File contents
First 175 lines

```

1 16/03/2023:13:40:09 094 [Main.java] [INFO] Main [Ln] - 1.0
2 16/03/2023:13:40:09 120 [Main.java] [INFO] Main [RESULT - Ln] - 0.0
3 16/03/2023:13:40:09 122 [Main.java] [INFO] Main [SQRT] - 25.0
4 16/03/2023:13:40:09 123 [Main.java] [INFO] Main [RESULT - SQRT] - 5.0
5 16/03/2023:13:40:09 124 [Main.java] [INFO] Main [SQRT] - 1.0
6 16/03/2023:13:40:09 124 [Main.java] [INFO] Main [RESULT - SQRT] - 1.0
7 16/03/2023:13:40:09 124 [Main.java] [INFO] Main [SQRT] - 0.0
8 16/03/2023:13:40:09 125 [Main.java] [INFO] Main [RESULT - SQRT] - 0.0
9 16/03/2023:13:40:09 125 [Main.java] [INFO] Main [FACT] - 0.0
10 16/03/2023:13:40:09 126 [Main.java] [INFO] Main [RESULT - SQRT] - 5.0
11 16/03/2023:13:40:09 127 [Main.java] [INFO] Main [RESULT - SQRT] - 1.0
12 16/03/2023:13:40:09 127 [Main.java] [INFO] Main [RESULT - SQRT] - 0.0
13 16/03/2023:13:40:09 127 [Main.java] [INFO] Main [FACT] - 0.0
14 16/03/2023:13:40:09 128 [Main.java] [INFO] Main [FACT] - 8.0
15 16/03/2023:13:40:09 137 [Main.java] [INFO] Main [POW] - 2.0 5.0
16 16/03/2023:13:40:09 138 [Main.java] [INFO] Main [RESULT - POW] - 32.0
17 16/03/2023:13:40:09 138 [Main.java] [INFO] Main [POW] - 9.0 0.0

```

Summary

- Number of lines analyzed: 175
- Format: semi_structured_text
- Grok pattern: %{DATE:timestamp} %{TIME:time} %{INT:field} \[.*?\] \[%{LOGLEVEL:log.level}\] \[.*?\] \[.*?\] \[%{NUMBER:field2}:*
- Time field: timestamp
- Time format: dd/MM/yyyy

[Override settings](#) [Analysis explanation](#)

The screenshot shows the Elasticsearch interface with the file 'calculator.log' uploaded. The 'File contents' section displays the first 175 lines of the log file. The 'Summary' section provides an overview of the analyzed file, including the number of lines (175), format (semi_structured_text), and the current Grok pattern used for parsing the logs. The 'Override settings' panel is open, showing the modified Grok pattern.

File contents
First 175 lines

```

1 16/03/2023:13:40:09 094 [Main.java] [INFO] Main [Ln] - 1.0
2 16/03/2023:13:40:09 120 [Main.java] [INFO] Main [RESULT - Ln] - 0.0
3 16/03/2023:13:40:09 122 [Main.java] [INFO] Main [SQRT] - 25.0
4 16/03/2023:13:40:09 123 [Main.java] [INFO] Main [RESULT - SQRT] - 5.0
5 16/03/2023:13:40:09 124 [Main.java] [INFO] Main [SQRT] - 1.0
6 16/03/2023:13:40:09 124 [Main.java] [INFO] Main [RESULT - SQRT] - 1.0
7 16/03/2023:13:40:09 124 [Main.java] [INFO] Main [SQRT] - 0.0
8 16/03/2023:13:40:09 125 [Main.java] [INFO] Main [RESULT - SQRT] - 0.0
9 16/03/2023:13:40:09 125 [Main.java] [INFO] Main [FACT] - 0.0
10 16/03/2023:13:40:09 126 [Main.java] [INFO] Main [RESULT - SQRT] - 5.0
11 16/03/2023:13:40:09 127 [Main.java] [INFO] Main [RESULT - SQRT] - 1.0
12 16/03/2023:13:40:09 127 [Main.java] [INFO] Main [RESULT - SQRT] - 0.0
13 16/03/2023:13:40:09 127 [Main.java] [INFO] Main [FACT] - 0.0
14 16/03/2023:13:40:09 128 [Main.java] [INFO] Main [FACT] - 8.0
15 16/03/2023:13:40:09 137 [Main.java] [INFO] Main [POW] - 2.0 5.0
16 16/03/2023:13:40:09 138 [Main.java] [INFO] Main [RESULT - POW] - 32.0
17 16/03/2023:13:40:09 138 [Main.java] [INFO] Main [POW] - 9.0 0.0

```

Override settings

Number of lines to sample: 1000

Data format: semi_structured_text

Grok pattern:

```
%{DATE:timestamp} %{TIME:time} %{INT:field} \[.*?\] \[%{LOGLEVEL:log.level}\] \[.*?\] \[.*?\] \[%{NUMBER:field2}:*
```

Timestamp format: custom

Custom timestamp format: dd/MM/yyyy

Time field: timestamp

Edit field names

timestamp: time

[Override settings](#) [Analysis explanation](#)

More ways to add data

In addition to adding [integrations](#), you can try our sample data or upload your own data.

[Sample data](#) [Upload file](#)

calculator.log

File contents
First 175 lines

```
1: 16/03/2023:13:48:09 894 [Main.java] [INFO] Main [ln] - 1.0
2: 16/03/2023:13:48:09 120 [Main.java] [INFO] Main [RESULT] - 1.0
3: 16/03/2023:13:48:09 121 [Main.java] [INFO] Main [ln] - 2.0
4: 16/03/2023:13:48:09 123 [Main.java] [INFO] Main [RESULT] - 5.0
5: 16/03/2023:13:48:09 124 [Main.java] [INFO] Main [ln] - 3.0
6: 16/03/2023:13:48:09 124 [Main.java] [INFO] Main [RESULT] - 5.0
7: 16/03/2023:13:48:09 125 [Main.java] [INFO] Main [ln] - 4.0
8: 16/03/2023:13:48:09 125 [Main.java] [INFO] Main [RESULT] - 6.0
9: 16/03/2023:13:48:09 126 [Main.java] [INFO] Main [ln] - 5.0
10: 16/03/2023:13:48:09 126 [Main.java] [INFO] Main [RESULT] - 5.0
11: 16/03/2023:13:48:09 127 [Main.java] [INFO] Main [ln] - 6.0
12: 16/03/2023:13:48:09 127 [Main.java] [INFO] Main [RESULT] - 1.0
13: 16/03/2023:13:48:09 128 [Main.java] [INFO] Main [ln] - 7.0
14: 16/03/2023:13:48:09 128 [Main.java] [INFO] Main [RESULT] - 8.0
15: 16/03/2023:13:48:09 128 [Main.java] [INFO] Main [FACT] - 8.0
16: 16/03/2023:13:48:09 129 [Main.java] [INFO] Main [ln] - 8.0
17: 16/03/2023:13:48:09 129 [Main.java] [INFO] Main [RESULT] - 40320.0
```

Summary

Number of lines analyzed: 175

Format: semi_structured_text

Grok pattern: %(DATE:timestamp):%{TIME:time} %{INT:field} \(%{GREEDYDATA:thread}\) \(%{LOGLEVEL:level}\) %{GREEDYDATA:logger} \(%{GREEDYDATA:action}\) \-%{GREEDYDATA:line}

Time field: timestamp

Time format: dd/MM/yyyy

[Import](#) [Cancel](#)

More ways to add data

In addition to adding [integrations](#), you can try our sample data or upload your own data.

[Sample data](#) [Upload file](#)

calculator.log

Import data

[Simple](#) [Advanced](#)

Index name: calculatorlog

Create data view

[Import](#)

[Back](#) [Cancel](#)

More ways to add data

In addition to adding [integrations](#), you can try our sample data or upload your own data.

[Sample data](#) [Upload file](#)

calculator.log

Import data

[Simple](#) [Advanced](#)

Index name: calculatorlog

Create data view

[Reset](#)

File processed ✓ Index created ✓ Ingest pipeline created ✓ Data uploaded ✓ Data view created ✓

Import complete ✓

Index: calculatorlog
Data view: calculatorlog

[Back](#) [Cancel](#)

Then go to the dashboard to create a dashboard and then select the kibana

The image consists of three vertically stacked screenshots of the Elastic Stack interface.

Screenshot 1: Import Data Progress

This screenshot shows the "Import data" progress window. It displays a timeline with five steps: "File processed", "Index created", "Ingest pipeline created", "Data uploaded", and "Data view created". All steps are marked as completed with green checkmarks. Below the timeline, under "Import complete", the following details are listed:

- Index: calculatorlog
- Data view: calculatorlog
- Ingest pipeline: calculatorlog-pipeline
- Documents ingested: 176

At the bottom, there are four buttons: "View index in Discover", "Index Management", "Data View Management", and "Create Filebeat configuration".

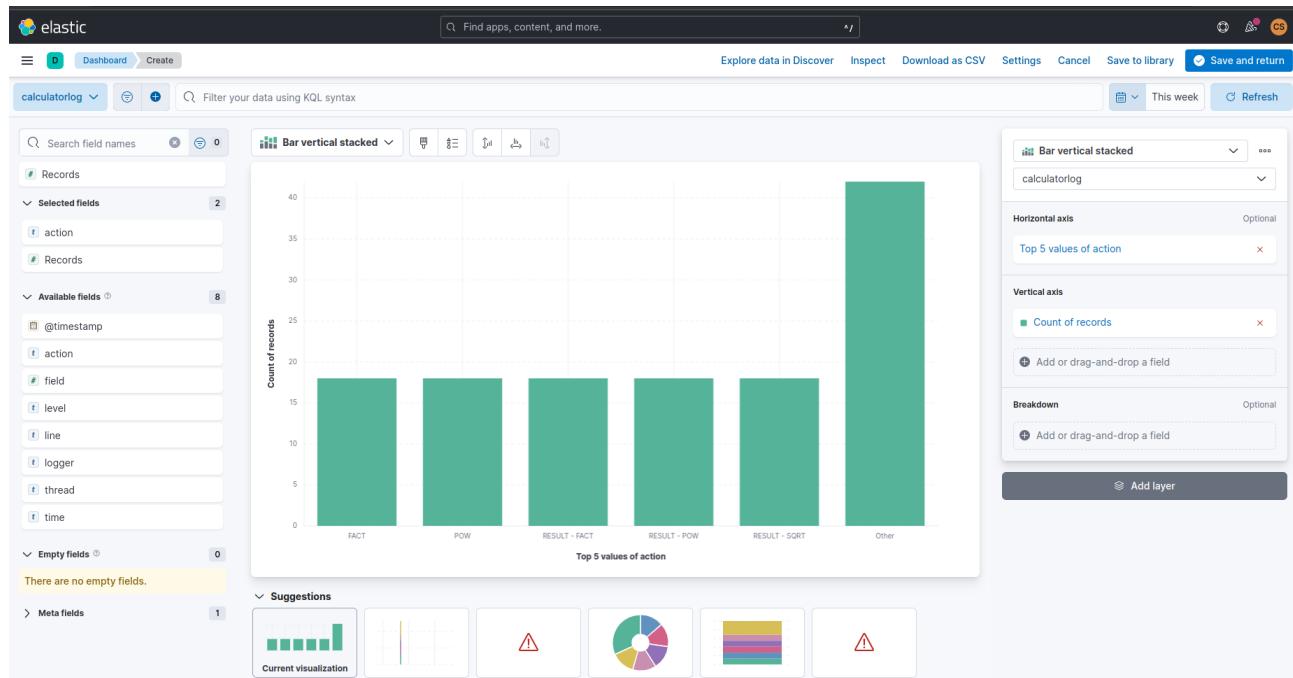
Screenshot 2: Create your first dashboard

This screenshot shows the "Create your first dashboard" page. It features a large central area with a "Create a dashboard" button. Above the button, there is a section titled "Create your first dashboard" with the sub-instruction: "Analyze all of your Elastic data in one place by creating a dashboard and adding visualizations." Below this, it says "New to Kibana? Add some sample data to take a test drive."

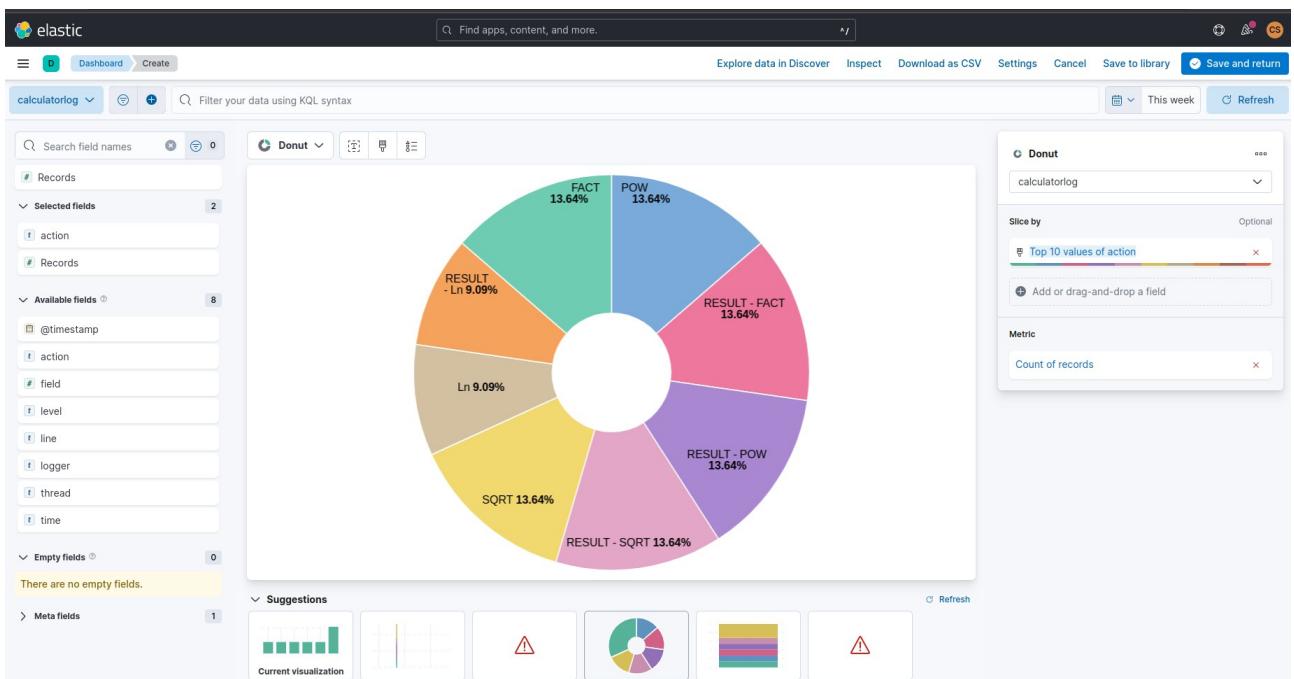
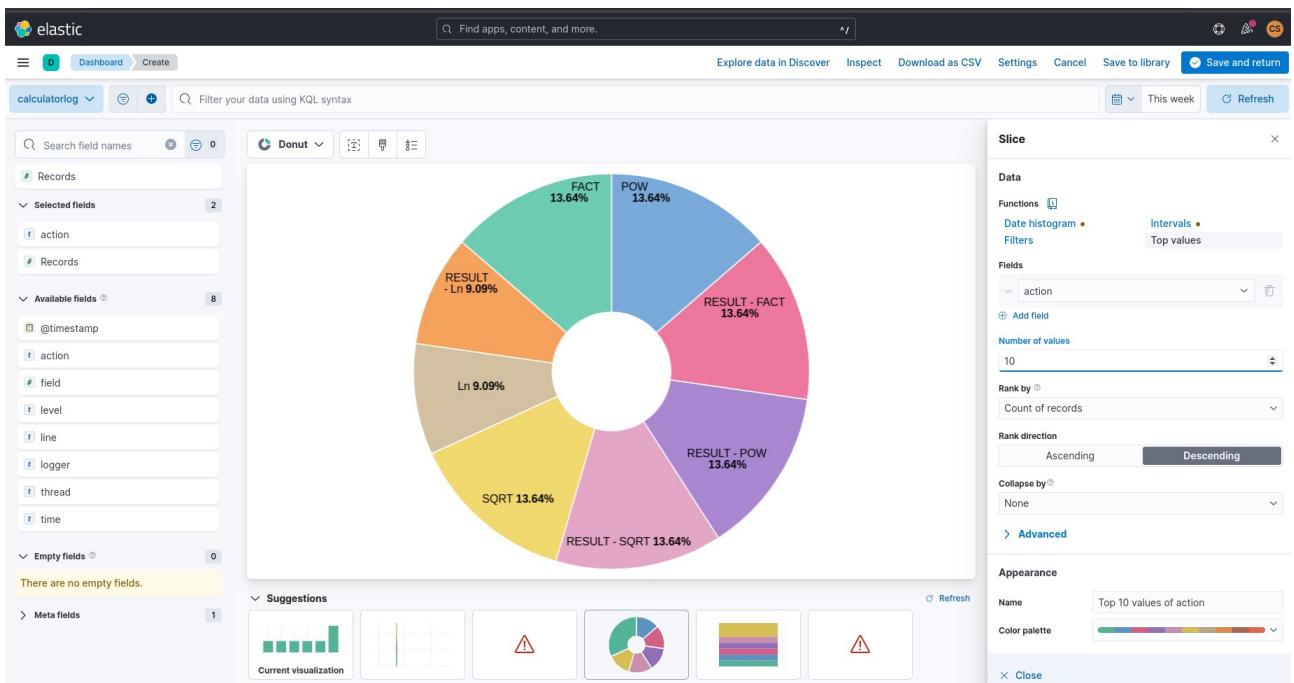
Screenshot 3: Visualization Configuration

This screenshot shows the configuration interface for a visualization. On the left, there is a sidebar with a "calculator_kibana" dropdown and various field selection options. The main area contains a "Bar vertical stacked" visualization. A large callout box in the center says "Drop some fields here to start". To the right, there are sections for "Horizontal axis", "Vertical axis", and "Breakdown", each with a "Add or drag-and-drop a field" input field. At the bottom right, there is a "Add layer" button.

Then drag the *action* to start. In this visualization, we can see how many logs are there for every function.



We can also change the value of records we wanted to see.



Challenges :

1. I got error when I am building docker in Jenkins

Solution : We have to login docker locally using `sudo docker login` and use `sudo usermod -a -G docker jenkins`

This command adds the user "jenkins" to the "docker" group on a Linux-based system.

Here is a breakdown of the command:

- "sudo" is a command used to run a command with administrative or "superuser" privileges.

- "usermod" is a command used to modify user account settings.
- "-a" specifies that the modification is an addition to the existing settings.
- "-G" specifies the group to which the user will be added.
- "docker" is the name of the group to which the user "jenkins" will be added.
- "jenkins" is the name of the user that will be added to the "docker" group.

2. I got errors with JDK while building maven project

Solution : I used one java version and set one version in maven. I changed that.

3. I got error in deployment in ansible step in jenkins

Solution : Use command `sh "/usr/bin/pip3 install docker"`. It installs python docker in jenkins.

Links :

Github link : <https://github.com/sivanich9/Calculator>

DockerHub link : <https://hub.docker.com/repository/docker/sivani4/calculator/general>