

Spe Mini Project - Mini Calculator

Ellanti Bharath Sai

IMT2018022

Introduction	1
What is DevOps	2
Why DevOps?	3
DevOps Tool Chain	3
Git and Github - Source Code Management	4
Maven - Build	8
Junit - Testing	17
Jenkins - Continuous Integration	20
Docker	25
Ansible - Deployment	30
EIK Stack - Monitoring	35
Github Repo : https://github.com/Bharath14/MiniProject	40

Introduction

This is a simple calculator program developed in java. It has the following four functionalities.

1. Square root of a number.
2. Factorial of a number.
3. Logarithm of a number($\ln(x)$).
4. Power function.

```
1. Square Root
2. Factorial
3. Logarithm
4. Power
5. Exit
Select From the above Menu
1
Enter number
4
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
15:18:37.917 [Calculator.java] INFO calculator.Calculator - Square root of 4 2.0
Square root of 4 is: 2.0
1. Square Root
2. Factorial
3. Logarithm
4. Power
5. Exit
Select From the above Menu
```

We can select from the given options and perform our required operation. The program will run until 5 is pressed which will terminate the program.

We have different DevOps tools for different stages of a project like building, testing, deploying, and monitoring.

What is DevOps

DevOps is a project execution methodology where the development team and an operation team work together in the cycle of development and deployment of the product.

Dev- People developing the software

Ops - System Engineers, Operations staff, Security Professionals, etc

We have studied agile methodology which breaks the wall between the business team and the development team. It collaborates with customers, product management, developers, and QA to fill in the gaps and rapidly iterate towards a better product. DevOps is an extension of agile principles by including the operations team and going beyond “the code”.

Goal of DevOps

1. Less time to reach the market.
2. Deployment frequency will be high.
3. The failure rate will be less for the new releases.
4. Shortened lead time between fixes.

DevOps focuses on the idea of sharing ideas, issues, knowledge, and goals between individuals. DevOps is also characterized by operations staff making use of many of the same techniques/tools as developers for their systems work.

DevOps principles involve CALMS:

1. Culture - people -> process -> tools
2. Automation - Infrastructure as code
3. Lean – Small batch sizes focus on value for end-user
4. Measure – Measure everything; Show improvement
5. Sharing - Collaboration between Dev and Ops.

DevOps practice involves:

1. Version control for all.
2. Automated Testing.
3. Proactive monitoring and metrics.
4. Configuration Management.
5. Continuous Integration/ deployment/delivery.
6. virtualization/cloud/containers .
7. Toolchain approach.

Why DevOps?

Business Benefits:

1. Improving customer service and increasing the client's satisfaction. One of the main tasks of DevOps is to develop better software and deliver it faster to end-users (which increases the loyalty of the latter)
2. IT and business integration. Another goal pursued by DevOps is to promote mutual understanding between departments within the company
3. Innovations and digital transformation. Companies tend to keep up with innovations (such as the Internet of Things), and DevOps can be a good helper in supporting digital services offered to the market. Thus, DevOps effectively frees up resources for innovation.
4. flexibility: A flexible approach to enterprise-wide development is more efficient because closer cooperation leads to the generation (as well as implementation) of better ideas.

DevOps Tool Chain

1. Source Code Management - Github.
2. Testing - JUnit

3. Build - Maven
4. Continuous Integration - Jenkins
5. Containerization - Docker Hub
6. Deployment - Ansible
7. Monitoring - ELK Stack

We push our code from the local machine to Github. Jenkins server checks the GitHub periodically and triggered a building job when there is any new commit in the GitHub. Maven builds the code and also completes the testing using JUnit. A jar file with all the dependencies is created. Then we create a Docker image using this jar file. We then push this docker image to the Docker hub. Ansible pulls the image from the Docker hub and places it on the server we specified. We then run this docker image on the server using the command:

```
docker run -i <Image name>.
```

We then send the log files to ELK Stack and monitor the deployment.

Git and Github - Source Code Management

Git is a distributed version control system, it is a tool to manage project source code history. Using git we can retrieve the previous versions of a project. Git stores and tracks the different changes of the project.

Installation and Configuration of Git on Ubuntu:

1. \$ sudo apt update
2. \$ sudo apt install git
3. \$ git config --global user.name "Your name"
4. \$ git config --global user.email "Your email"
5. \$ git --version

Github is a web application that provides the features of git. We can store our projects on GitHub and share them with others. We can also clone repositories from GitHub and make changes to them in our repository, commit the changes using git and then push the committed code to the Github. Github provides the option of multiple people working on the same project. We can pull the changes of others into our local system. Merge different worker's codes, create different branches, etc.

Configuration:

1. Signup on the Github page.
2. Create a repository.

3. Enter the project name, make the project public/private, add some ReadMe files (optional) and click create

For this project, we are using IntelliJ as an IDE.

Configure Github on IntelliJ:

In IntelliJ create a maven project and select the java version as per requirements. In the menu bar of the project select VCS and follow the instructions. It would ask the URL of the repo to be linked and git credentials. After providing the correct details our IntelliJ project is linked with the repo we created on GitHub.

After adding the required code to our local repository click on Git->commit

Commit to master



✓ **Changes** 11 files

- ✓ calculator.CalculatorTest.txt ~/IdeaProjects/MiniProject/target/surefire-report
- ✓ Calculator.class ~/IdeaProjects/MiniProject/target/classes/calculator
- ✓ Calculator.java ~/IdeaProjects/MiniProject/src/main/java/calculator
- ✓ CalculatorTest.class ~/IdeaProjects/MiniProject/target/test-classes/calculator
- ✓ Main.class ~/IdeaProjects/MiniProject/target/classes
- ✓ Main.java ~/IdeaProjects/MiniProject/src/main/java
- ✓ MiniProject.log ~/IdeaProjects/MiniProject
- ✓ MiniProject-1.0-SNAPSHOT.jar ~/IdeaProjects/MiniProject/target
- ✓ MiniProject-1.0-SNAPSHOT-jar-with-dependencies.jar ~/IdeaProjects/MiniProject
- ✓ pom.properties ~/IdeaProjects/MiniProject/target/maven-archiver
- ✓ TEST-calculator.CalculatorTest.xml ~/IdeaProjects/MiniProject/target/surefire-

☐ Amend

11 modified

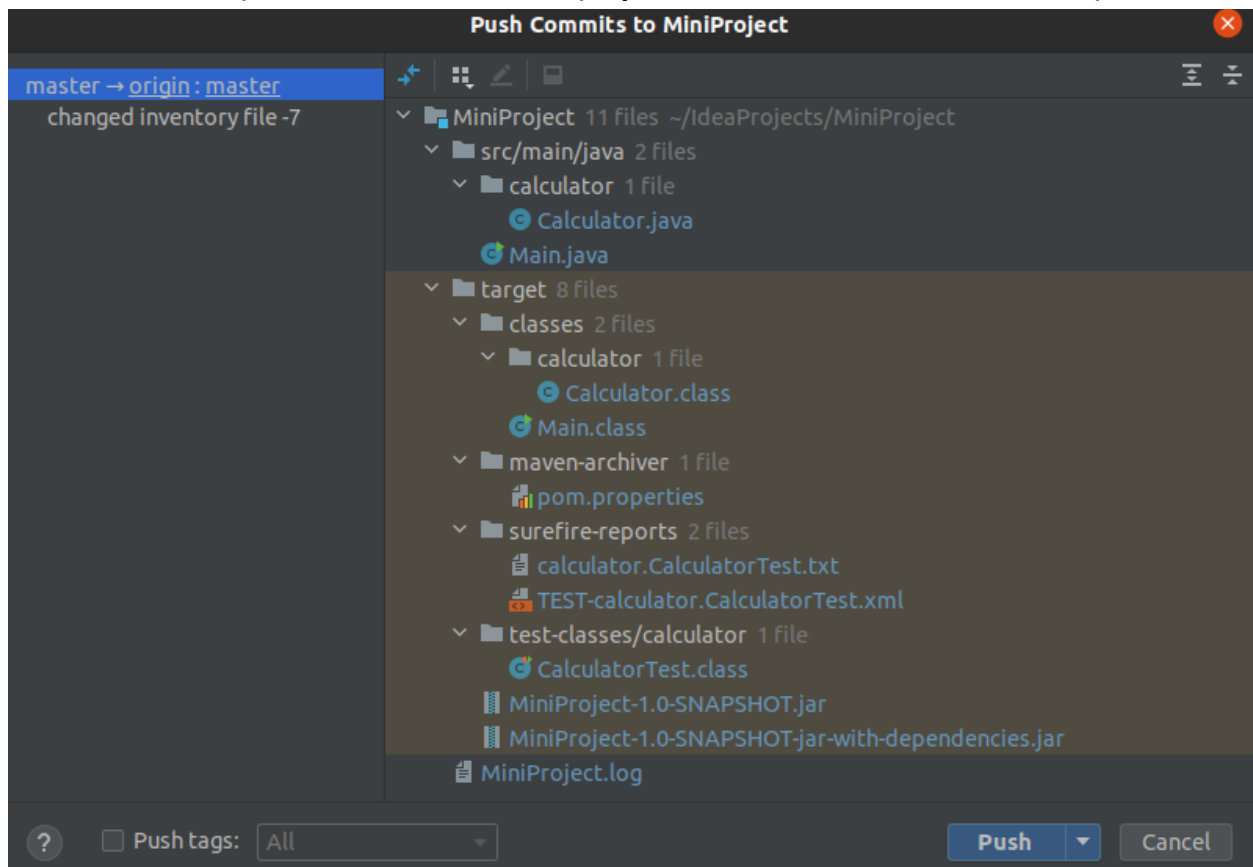
changed inventory file -7|

Commit

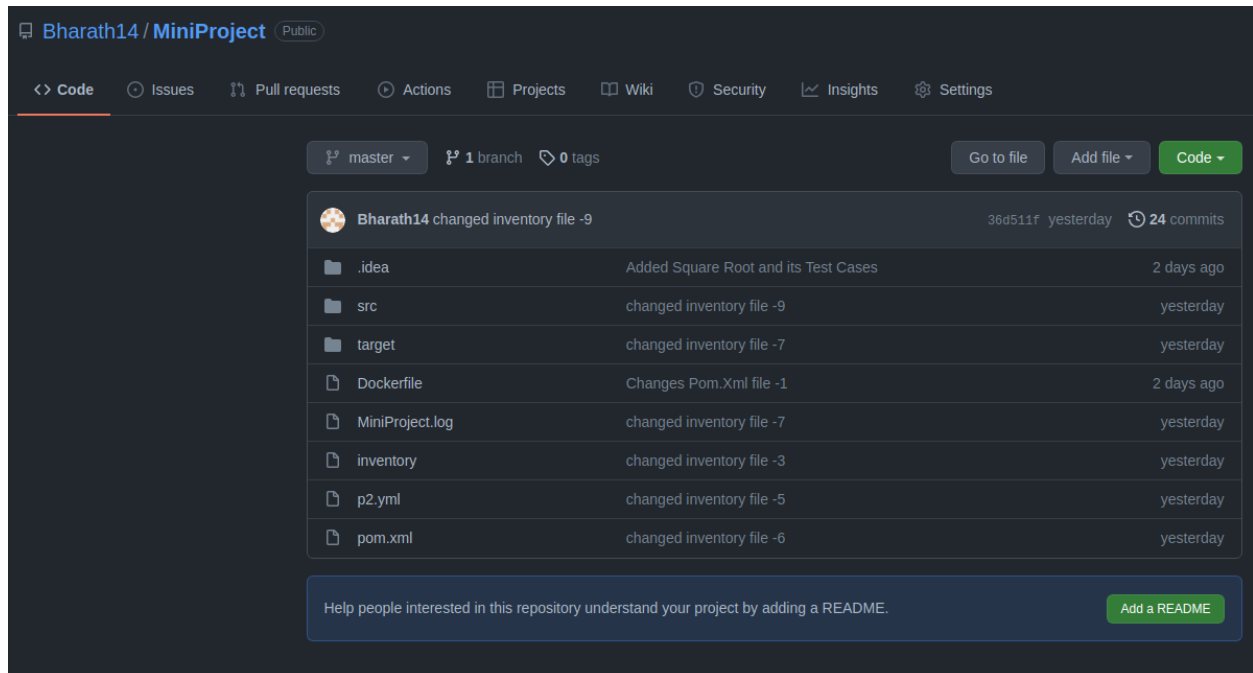
Commit and Push...

Git - 15 TODO - Problems - Profiler - Terminal - Dependencies

Select the files to be pushed into the GitHub. Then add some commit message and then click commit. Now we have stored our versions of the project in our local system. Now we have to push this version of the project into Github. So we click Git->push



Click on push and refresh the Github repository to see the new changes.



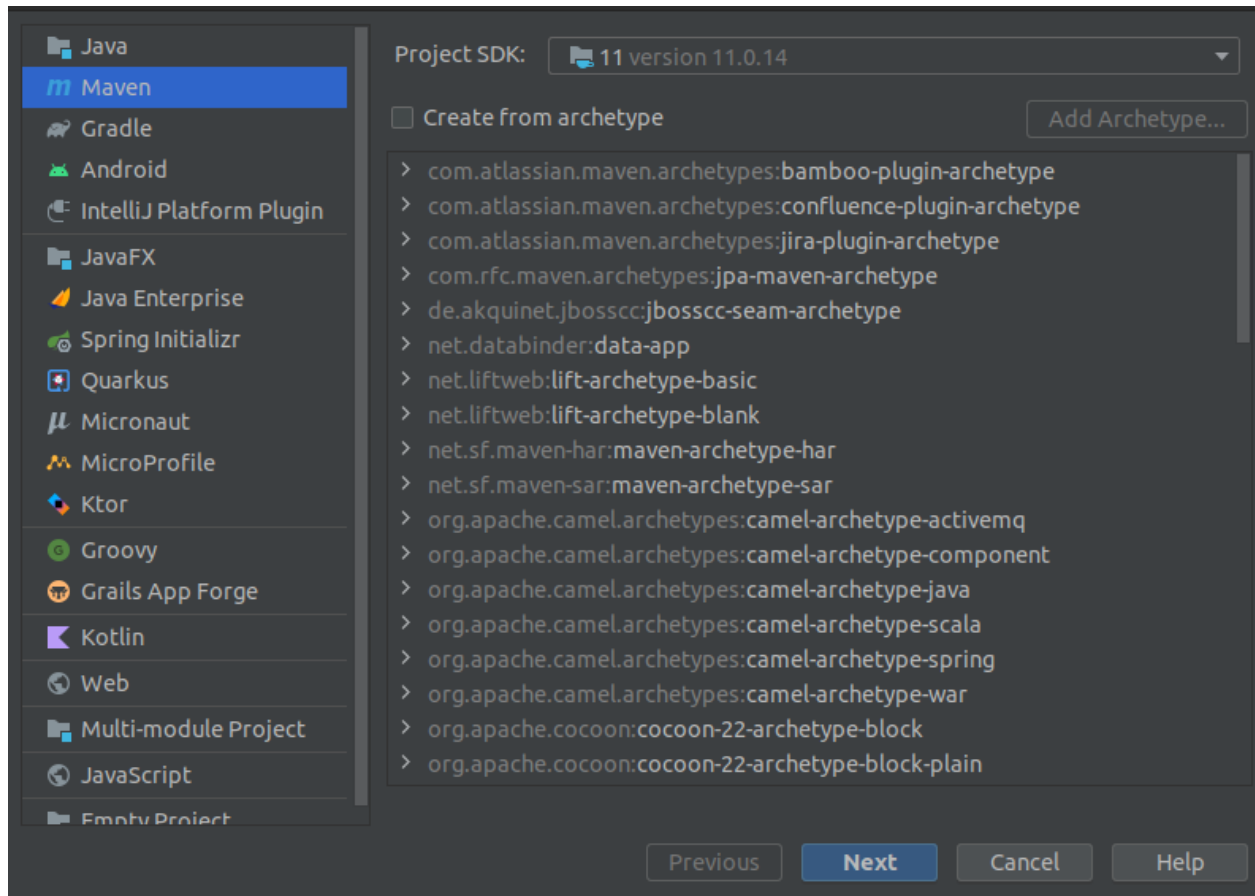
Maven - Build

Maven is a build automation tool for Java programs. It is used for building management and dependencies. We need to specify our dependencies and configuration in a config file called pom.xml. When we build a project using maven first it will search for the dependencies in the local repository, if it doesn't find then it will try to fetch the dependency jar files from the remote repository. Using maven we can also complete the unit testing of the application.

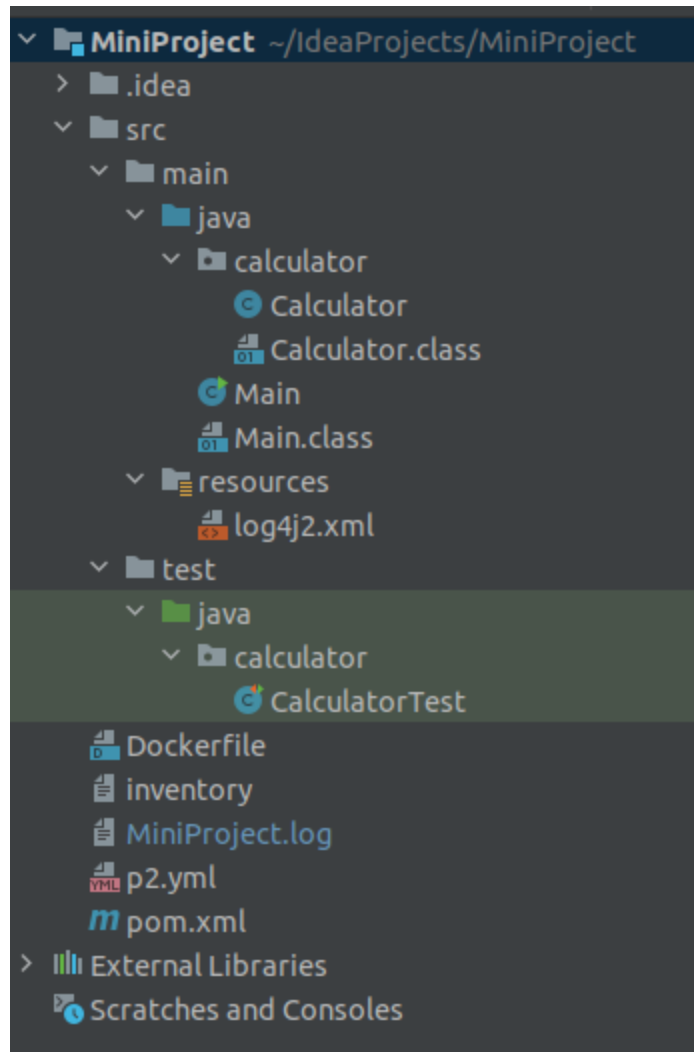
To install maven in ubuntu run the following command:

```
$sudo apt update  
$sudo apt-get install maven
```

We use IntelliJ for this project as an IDE. Create a new project in IntelliJ by clicking File->New->Project



Select Maven, java version, and click next. Create the project. Add our code in the src/main/java folder and test cases in the src/test/java folder. The final project structure would be.



Add the dependencies in the 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>MiniProject</artifactId>
  <version>1.0-SNAPSHOT</version>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <executions>
          <execution>
            <phase>package</phase>
            <goals>
              <goal>
                single
              </goal>
            </goals>
            <configuration>
              <archive>
                <manifest>
                  <mainClass>Main</mainClass>
                </manifest>
              </archive>
              <descriptorRefs>
                <descriptorRef>
                  jar-with-dependencies
                </descriptorRef>
              </descriptorRefs>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
```

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>3.2.2</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>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.11.1</version>
  </dependency>
</dependencies>
```

Source Code:
Calculator.Java

```

package calculator;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import java.lang.Math;
public class Calculator {
    private static final Logger logger = LogManager.getLogger(Calculator.class);
    public double SquareRoot(double x)
    {
        logger.info("Square Root");
        return Math.sqrt(x);
    }
    public double factorial(double x)
    {
        logger.info("Factorial");
        double fac = 1;
        for(int i =2;i<=x;i++)
        {
            fac = fac*i;
        }
        return fac;
    }

    public double log(double x)
    {
        logger.info("Logarithm");
        return Math.log(x);
    }

    public double power(double x , int y)
    {
        logger.info("Power");
        return Math.pow(x, y);
    }
}

```

This file contains all the required functions of the calculator.

Main.java

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args)
    {
        Calculator calculator = new Calculator();
        Scanner scan = new Scanner(System.in);
        System.out.println("1. Square Root");
        System.out.println("2. Factorial");
        System.out.println("3. Logarithm");
        System.out.println("4. Power");
        System.out.println("5. Exit");
        System.out.println("Select From the above Menu");

        int menu = scan.nextInt();
        while(menu != 5)
        {
            if(menu == 1)
            {
                System.out.println("Enter number");
                double x = scan.nextDouble();
                System.out.println("Square root of " + x + " is: " + calculator.SquareRoot(x));
            }
            else if(menu == 2)
            {
                System.out.println("Enter number:");
                double x = scan.nextDouble();
                System.out.println("Factorial of " + x + " is: " + calculator.factorial(x));
            }
            else if(menu == 3)
            {
                System.out.println("Enter number:");
                double x = scan.nextDouble();
                System.out.println("Logarithm of " + x + " is: " + calculator.log(x));
            }
            else if(menu == 4)
            {
                System.out.println("Enter base");
                double x = scan.nextDouble();
                System.out.println("Enter power");
                int y = scan.nextInt();
            }
        }
    }
}
```

```

    }
    else if(menu == 4)
    {
        System.out.println("Enter base");
        double x = scan.nextDouble();
        System.out.println("Enter power");
        int y = scan.nextInt();
        System.out.println(x + " Power of " + y + " is: " + calculator.power(x, y));
    }
    System.out.println("1. Square Root");
    System.out.println("2. Factorial");
    System.out.println("3. Logarithm");
    System.out.println("4. Power");
    System.out.println("5. Exit");
    System.out.println("Select From the above Menu");
    menu = scan.nextInt();
}
}

```

This function is the main function that takes input from the user and calls the respective function.

Commands to build, and test projects using maven. We need to first write our pom.xml file in the project repo.

\$mvn clean test

This command will first clean the target folder which was created by earlier builds, then build the project and then run the test cases we mentioned in the project.

\$mvn clean install

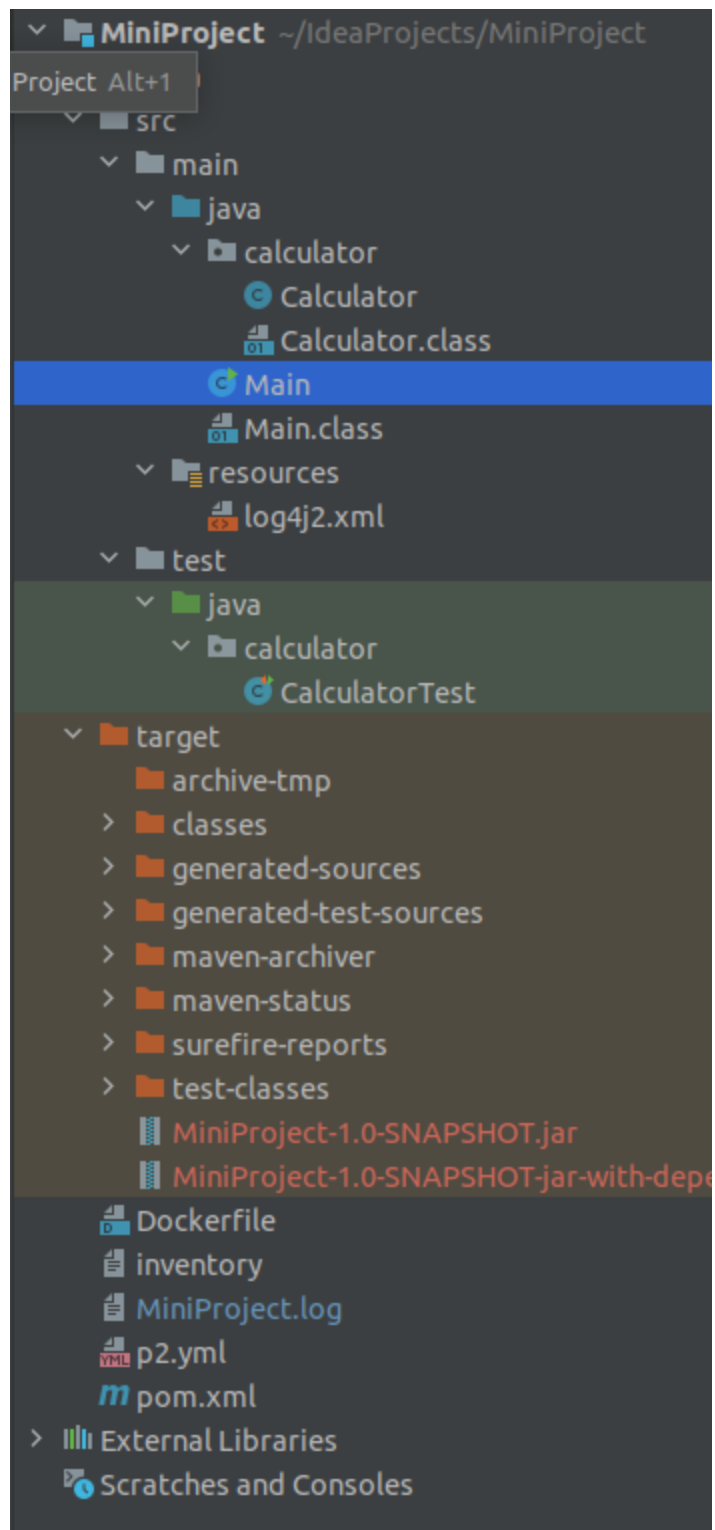
Unlike the test command, this will build the project, run the test cases and also create a jar file with code and all the dependencies. We can then use this JAR file to run the application.

```

[INFO] --- maven-install-plugin:2.4:install (default-install) @ MiniProject ---
[INFO] Installing /home/bharath/IdeaProjects/MiniProject/target/MiniProject-1.0-SNAPSHOT.jar to /home/bharath/.m2/repository/org/example/MiniProject/1.0-SNAPSHOT/MiniProject-1.0-SNAPSHOT.jar
[INFO] Installing /home/bharath/IdeaProjects/MiniProject/pom.xml to /home/bharath/.m2/repository/org/example/MiniProject/1.0-SNAPSHOT/MiniProject-1.0-SNAPSHOT.pom
[INFO] Installing /home/bharath/IdeaProjects/MiniProject/target/MiniProject-1.0-SNAPSHOT-jar-with-dependencies.jar to /home/bharath/.m2/repository/org/example/MiniProject/1.0-SNAPSHOT/MiniProject-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.929 s
[INFO] Finished at: 2022-04-18T13:37:33+05:30
[INFO] -----

```

After this command a target folder will be created and we can see a jar file in the target folder.



We can then run this JAR file to run the application

```
$java -cp ./target/MiniProject-1.0-SNAPSHOT-jar-with-dependencies.jar Main
```



```
bharath@bharath-VirtualBox:~/IdeaProjects/MiniProject$ java -cp ./target/MiniProject-1.0-SNAPSHOT-jar-with-dependencies.jar Main
1. Square Root
2. Factorial
3. Logarithm
4. Power
5. Exit
Select From the above Menu

```

JUnit - Testing

JUnit is a unit testing tool for Java programs. JUnit features include:

1. Assertions for testing expected results
2. Test fixtures for sharing common test data
3. Test runners for running tests

Setup:

A Pom.xml file will be created when we created the project in IntelliJ IDEA. We need to add the JUnit dependency in that file.

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
  <scope>test</scope>
</dependency>
```

Configure this JUnit file. We have to now write the unit test cases for each function.

1. True Positive: Case when the code is perfect and the test cases pass.
2. True Negative: Case when the code is perfect but still some cases don't pass.
3. False Positive: Case there are errors which the test cases can't find and they pass.
4. False Negative: When the code is broken and the tests don't pass.

As our application is simple enough we just write test cases for True Positive and False positive for each function.

To create a test file we need to create a calculator package in src/test/java. In the calculator package create a CalculatorTest.java file and add our test suites.

```
import static org.junit.Assert.*;
import org.junit.Test;
public class CalculatorTest {
    private static final double DELTA = 1e-15;
    Calculator calc = new Calculator();

    @Test
    public void squarerootTruePositive()
    {
        assertEquals ( message: "Square root of an Int - True Positive", expected: 2, calc.SquareRoot( 4), DELTA);
        assertEquals ( message: "Square root of an Int - True Positive", expected: 4, calc.SquareRoot( 16), DELTA);
    }

    @Test
    public void squarerootFalsePositive()
    {
        assertEquals ( message: "Square root of an Int - True Positive", unexpected: 3, calc.SquareRoot( 7), DELTA);
        assertEquals ( message: "Square root of an Int - True Positive", unexpected: 5, calc.SquareRoot( 49), DELTA);
    }

    @Test
    public void factorialTruePositive()
    {
        assertEquals ( message: "Square root of an Int - True Positive", expected: 1, calc.factorial( 1), DELTA);
        assertEquals ( message: "Square root of an Int - True Positive", expected: 120, calc.factorial( 5), DELTA);
    }

    @Test
    public void factorialFalsePositive()
    {
        assertEquals ( message: "Square root of an Int - True Positive", unexpected: 0, calc.factorial( 0), DELTA);
        assertEquals ( message: "Square root of an Int - True Positive", unexpected: 12, calc.factorial( 4), DELTA);
    }
}
```

```

@Test
public void logTruePositive()
{
    assertEquals ( message: "Square root of an Int - True Positive", expected: 0, calc.log( x: 1), DELTA);
    assertEquals ( message: "Square root of an Int - True Positive", expected: 2.302585092994046, calc.log( x: 10), DELTA);
}

@Test
public void logFalsePositive()
{
    assertEquals ( message: "Square root of an Int - True Positive", unexpected: 0, calc.log( x: 2), DELTA);
    assertEquals ( message: "Square root of an Int - True Positive", unexpected: 12, calc.log( x: 4), DELTA);
}

@Test
public void powerTruePositive()
{
    assertEquals ( message: "Square root of an Int - True Positive", expected: 0, calc.power( x: 0, y: 1), DELTA);
    assertEquals ( message: "Square root of an Int - True Positive", expected: 100, calc.power( x: 10, y: 2), DELTA);
}

@Test
public void powerFalsePositive()
{
    assertEquals ( message: "Square root of an Int - True Positive", unexpected: 0, calc.power( x: 2, y: 0), DELTA);
    assertEquals ( message: "Square root of an Int - True Positive", unexpected: 12, calc.power( x: 4, y: 3), DELTA);
}

```

Now run the following command in the terminal to see the test logs.

```
$mvn clean test
```

```

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ MiniProject ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/bharath/IdeaProjects/MiniProject/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ MiniProject ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to /home/bharath/IdeaProjects/MiniProject/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ MiniProject ---
[INFO] Surefire report directory: /home/bharath/IdeaProjects/MiniProject/target/surefire-reports

-----
T E S T S
-----
Running calculator.CalculatorTest
2:46:51.728 [Calculator.java] INFO calculator.Calculator - Factorial of 0.0 1.0
2:46:51.740 [Calculator.java] INFO calculator.Calculator - Factorial of 4.0 24.0
2:46:51.742 [Calculator.java] INFO calculator.Calculator - Square root of 4.0 2.0
2:46:51.743 [Calculator.java] INFO calculator.Calculator - Square root of 16.0 4.0
2:46:51.745 [Calculator.java] INFO calculator.Calculator - Square root of 7.0 2.6457513110645907
2:46:51.746 [Calculator.java] INFO calculator.Calculator - Square root of 49.0 7.0
2:46:51.748 [Calculator.java] INFO calculator.Calculator - Logarithm of 1.0 0.0
2:46:51.749 [Calculator.java] INFO calculator.Calculator - Logarithm of 10.0 2.302585092994046
2:46:51.750 [Calculator.java] INFO calculator.Calculator - Factorial of 1.0 1.0
2:46:51.751 [Calculator.java] INFO calculator.Calculator - Factorial of 5.0 120.0
2:46:51.761 [Calculator.java] INFO calculator.Calculator - 2.0Power0 1.0
2:46:51.762 [Calculator.java] INFO calculator.Calculator - 4.0Power3 64.0
2:46:51.763 [Calculator.java] INFO calculator.Calculator - 0.0Power1 0.0
2:46:51.763 [Calculator.java] INFO calculator.Calculator - 10.0Power2 100.0
2:46:51.764 [Calculator.java] INFO calculator.Calculator - Logarithm of 2.0 0.6931471805599453
2:46:51.765 [Calculator.java] INFO calculator.Calculator - Logarithm of 4.0 1.3862943611198906
Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.896 sec

Results :

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

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.361 s
[INFO] Finished at: 2022-04-18T02:46:51+05:30
[INFO] -----

```

Jenkins - Continuous Integration

Continuous Integration is a process of integrating a developer's code into a shared repository in a version control system frequently. CI continuously monitors the version-control system for any new changes. When there are changes CI starts building the code and tests the program. If there are any errors then developers are notified about the error immediately. It will then allow you to deploy on a server automatically or as a one-click process.

Jenkins is an open-source CI/CD server capable of orchestrating a chain of actions that helps to achieve the Continuous Integration and Continuous Deployment process in an automated way.

Jenkins has various plugins which help us to clone, build, and test the code. We can also Containerize, and deploy code on servers using Jenkins. It provides the status of each stage and notifies about the failures if any.

The screenshot shows the Jenkins Dashboard. The top navigation bar includes the Jenkins logo, a search bar, and user information (E Bharath Sai). The left sidebar contains various navigation links: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Lockable Resources, and New View. The main content area displays a table of build items with columns for status, name, last success, last failure, and last duration. Below the table, there are links for 'Icon Legend', 'Atom Feed for all', 'Atom Feed for Failures', and 'Atom Feed for just latest builds'. The bottom right corner shows 'REST API' and 'Jenkins 2.332.2'.

S	W	Name	Last Success	Last Failure	Last Duration
		demo	N/A	N/A	N/A
		MiniProject	2 days 1 hr #4	N/A	8.1 sec
		MiniProject Pipeline	18 hr #39	18 hr #36	49 sec

Installation of Jenkins:

To install Jenkins we need to have java installed in our system. Then follow the following commands to install Jenkins.

1. `wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -`
2. `sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`
3. `sudo apt install ca-certificates`
4. `sudo apt-get update`
5. `sudo apt-get install jenkins`

To start jenkins run `sudo service jenkins start`

Open Jenkins in the localhost using port 8080. Follow the onboard instructions and set up Jenkins. To copy the admin password run `sudo cat /var/lib/Jenkins/secrets/initialAdminPassword` in the terminal.

To install any plugins in Jenkins go to dashboard Manage Jenkins -> Manage plugins and install the required plugins.

Plugin Manager

[Updates](#)[Available](#)[Installed](#)[Advanced](#)**Install** **Name****Released** **Installed****Credentials** 1118.v320cd028cb_a_0 **Unavailable**

Library plugins (for use by other plugins)

This plugin allows you to store credentials in Jenkins.

This version of the plugin exists but it is not being offered for installation, so the latest bug fixes or features are not available to you. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. If you are using the latest version of Jenkins offered to you, newer plugin releases may not be available to your release line yet. See the [plugin documentation](#) for information about its requirements.

1087.1089.v2f1b_9a_b_040e4

Update information obtained: 1 day 7 hr ago

[Check now](#)Select: [All](#), [Compatible](#), [None](#)

This page lists updates to the plugins you currently use.

After setting up username and password in Jenkins. We create a new project by selecting on new item in the top left corner of the Jenkins dashboard.

Enter an item name

» Required field

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**

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

**Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**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.

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

[OK](#)[Cancel](#)

Type a name for the project and select any type of project we want. For this project, we will be creating a pipeline project. After clicking OK we would be prompted to configure our project.

General
Build Triggers
Advanced Project Options
Pipeline

Description

[Plain text] Preview

☐ 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 ?
Schedule ?

⚠ Do you really mean "every minute" when you say "*****"? Perhaps you meant "H *****" to poll once per hour
Would last have run at Sunday, 17 April, 2022 at 7:13:14 PM India Standard Time; would next run at Sunday, 17 April, 2022 at 7:13:14 PM India Standard Time.

We will select Poll SCM under Build triggers which will check the Github for every minute periodically and triggers the pipeline whenever there is a new commit in the Github.

Then scroll down to see the heading pipeline. Under pipeline add the following script.

```


pipeline {
    environment{
        registry = "ellanti14/miniproject"
        registryCredential = '1'
        dockerimage = ''
    }
    agent any

    stages {
        stage('Step 1: Git clone') {
            steps {
                git 'https://github.com/Bharath14/MiniProject.git'
            }
        }
        stage('Step2 : Maven Build') {
            steps {
                sh 'mvn clean install'
            }
        }
    }
}


```


This script specifies cloning the project from Github with the given URL in the first stage. Then in the second stage, we specify to build and test the project using maven. Ignore the environment section for now. After adding this script click on Apply and save. Then we will be taken to the following page.

 [Back to Dashboard](#)

 [Status](#)


 [Changes](#)

 [Build Now](#)

 [Configure](#)

 [Delete Pipeline](#)

 [Full Stage View](#)

 [Rename](#)

 [Pipeline Syntax](#)

 [Git Polling Log](#)

Click on build now to manually trigger the project or do a new commit in GitHub and wait for a minute to make Jenkins automatically trigger the pipeline. After completion of the process, we will be shown as follows.



If there is an error in any stage that block will be shown in red color. We can hover on that block to see logs of the error.

Till now we have completed building and testing a project automatically using maven in Jenkins.

Docker

Docker is a tool used to create, deploy, and run applications in an easier way using containers. A container contains software code and its necessary components like libraries, frameworks, and other dependencies. Containers are isolated from each other. This makes the software to be move and run on any machine independent of the operating system. These containers are light compared to a Virtual Machine. Virtual Machine splits on top of the hardware. Container splits on top of OS. Docker allows programs to operate on the same Linux kernel as the host system and only needs applications to be packaged with things that aren't already running on the host computer. This results in a large performance gain while also reducing the application's size.

A Dockerfile is the foundation of every Docker container. A Dockerfile is a text file with instructions for building a Docker image. A Dockerfile describes the operating system that will be used by the container, as well as the languages, environmental variables, file locations, network ports, and other components it will require, as well as what the container will perform once it is started.

We need to first build a Docker Image from the code with all the dependencies and then we push this to Docker Hub. Docker image is the portable file with the specifications for which software components the container will run. Docker hub is a web application similar to GitHub which stores the Docker Images. We first push our docker image to Docker Hub from our local machine and then pull the docker image from Docker Hub to the server/machine we intended to run.

Installation of Docker:

1. `sudo apt update`
2. `sudo apt install apt-transport-https ca-certificates curl software-properties-common`
3. `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
4. `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"`
5. `sudo apt install docker-ce`
6. `sudo service docker status`

Executing docker without Sudo:

1. `sudo groupadd docker`

2. `sudo usermod -aG docker ${USER}`
3. `newgrp docker`
4. Restart the OS

Check if docker runs without sudo

`docker run hello-world`

To build and run an image the following command is used:

`$ docker build -t <name>.`

`$ docker run <imagename>:<tagname>`

The first searches for a Dockerfile in the current directory and then builds the image according to the Dockerfile.

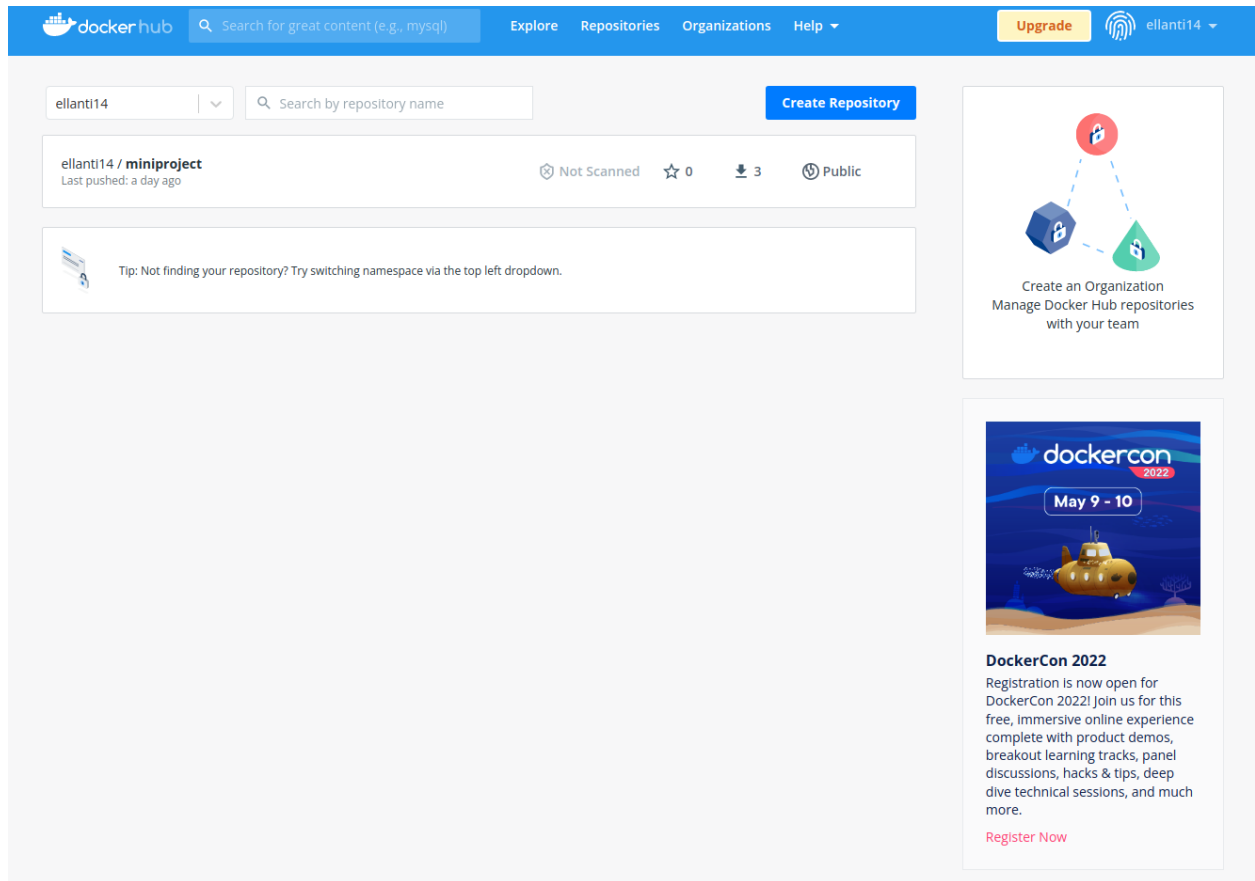
For our project, we first create a Dockerfile in our project repo

```
FROM openjdk:11
COPY ./target/MiniProject-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java", "-jar", "MiniProject-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

We have two ways to create a docker image. One gets the jar file from the target folder of the repo after building the project using maven or we can compile the src folder in the Docker container and then get the jar file and run that jar file. We have followed the first method. We first build and test the code using maven. A jar file with all the dependencies is created. We then use this jar file and create the docker image. We need to push this docker file to our repo on GitHub.

Docker in Jenkins:

We have to first sign up at <https://hub.docker.com/> and create our profile. Create a repository in the Docker hub.



Install Docker plugins in the Jenkins server.

Now we have to add these docker hub credentials to our Jenkins server. To do that we need to click manage Jenkins -> Manage Credentials->Jenkins(store)->Global Credentials->Add credential.

Scope ?
Global (Jenkins, nodes, items, all child items, etc) ▼

Username ?
ellanti14

☐ Treat username as secret ?

Password ?
Concealed Change Password

ID ?
1

Description ?
Docker Hub credentials

Save

We have to remember this ID field. We need to add the environment field at the start of the pipeline script.

```
environment{
    registry = "ellanti14/miniproject"
    registryCredential = '1'
    dockerimage = ''
}
```

In the registry field, we need to give the “username/repo name” of the docker hub. In the registryCredential field, we need to give the ID of Docker hub credentials that we mentioned in the Jenkins server.

We have to add the following pipeline script to our previous script. This added script specifies how to create a docker image from the jar file and then push this docker image to our Docker Hub repo.

```
stage('Steps : Build DockerImage') {
    steps {
        script{
            dockerimage = docker.build registry + ":latest"
        }
    }
}
stage('Push Image to Docker Hub') {
    steps {
        script {
            docker.withRegistry('', registryCredential){
                dockerimage.push()
            }
        }
    }
}
```

Now if we click build now after successful completion we can see the docker image in our docker hub repo.

Step 1: Git clone	Step2 : Maven Build	Step3 : Build DockerImage	Push Image to Docker Hub
1s	14s	1s	34s
734ms	10s	975ms	33s

We can pull this docker image from Docker Hub and run it in our local system.

```
$ docker run -i "username/reponame"
```

This command will first check if there is any docker image with the above name in our local system. If it doesn't find any such repo then it will pull this image from the Docker registry and run the docker image.

```
bharath@bharath-VirtualBox:~$ docker run -i ellanti14/miniproject
1. Square Root
2. Factorial
3. Logarithm
4. Power
5. Exit
Select From the above Menu
```

```
$ docker images
```

This command will show all the docker images that are residing in our local system.

```
bharath@bharath-VirtualBox:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ellanti14/miniproject	latest	4c5bce4d4368	23 hours ago	668MB
ellanti14/miniproject	<none>	fb9f3073d62	23 hours ago	668MB
ellanti14/miniproject	<none>	ae71098517fd	23 hours ago	668MB
ellanti14/miniproject	<none>	bfd0c4d4144d	23 hours ago	668MB
ellanti14/miniproject	<none>	828527ff9bb1	28 hours ago	668MB
ellanti14/miniproject	<none>	5f023aa3f21e	28 hours ago	668MB
ellanti14/miniproject	<none>	d54b464ba918	30 hours ago	662MB
ellanti14/miniproject	<none>	21b4d8b53789	30 hours ago	662MB
ellanti14/miniproject	31	d4dfec367a2e	30 hours ago	662MB
ellanti14/miniproject	30	1ff1614a08c2	30 hours ago	662MB
ellanti14/miniproject	29	e28cb017a76d	30 hours ago	662MB
ellanti14/miniproject	28	4e383cbd2e34	30 hours ago	662MB
ellanti14/miniproject	27	faa2687408fd	31 hours ago	662MB
ellanti14/miniproject	26	71fdbbe1117f5	31 hours ago	662MB
ellanti14/miniproject	25	3be57274ce97	31 hours ago	662MB
ellanti14/miniproject	21	fdef2c02a4de	31 hours ago	662MB
ellanti14/miniproject	20	072ab54f377b	32 hours ago	662MB
ellanti14/miniproject	19	f5ed398684c8	45 hours ago	662MB
ellanti14/miniproject	18	67a43dfac87e	46 hours ago	662MB
ellanti14/miniproject	16	bb65cc6c8306	2 days ago	662MB
ellanti14/miniproject	15	a8b60980e68e	2 days ago	660MB
ellanti14/miniproject	14	407404f6989d	2 days ago	660MB
ellanti14/miniproject	13	94a78b8d3bcd	2 days ago	660MB
openjdk	11	8c5fc4518cc2	2 weeks ago	660MB
hello-world	latest	feb5d9fea6a5	6 months ago	13.3kB

Ansible - Deployment

Ansible is open-source software that is used for Application deployment, configuration management, and continuous delivery. Ansible requires an inventory file and playbook. The inventory file contains all the information about different target hosts. The playbook contains information regarding what to do with a specific target node or set of target nodes. In this project, we use Ansible to pull the docker image from the docker hub and deploy that on the specified target host. Ansible is used to manage multiple machines using a control node. These target nodes are called agent nodes. These agent nodes are managed by the control node using ssh. Ansible doesn't depend on agent software and has no additional security infrastructure, so it's easy to deploy. That is we don't need the target nodes to be installed with ansible.

Ansible connects to network nodes (clients, servers, or whatever you're configuring) and then sends a little program known as an Ansible module to that node. Ansible runs

these modules through SSH and then deletes them after they're done. Only your Ansible control node needs login access to the managed nodes for this interaction to work. SSH keys are the most frequent method of authentication, however alternative methods are also allowed.

Inventory

```
[ubuntu18]  
172.16.134.184 ansible_user=bharath
```

First-line specifies some unique name for the machine. In the second line, we specify the IP address of the target machine and the user of the host machine.

PlayBook

```
---  
- name: Pull docker image from dockerHub  
  hosts: ubuntu18  
  remote_user : bharath  
  tasks:  
    - name: Pull calculator image  
      docker_image:  
        name: ellanti14/miniproject  
        source: pull
```

Here in the host's column, we specify the particular host on which we are going to deploy the docker image pulled from the docker hub. If we put hosts as all then all the target nodes specified in the inventory file will get the docker image. Add these files to our git repo.

To start using Ansible in our project we first need to install a ubuntu 18.04 server on a virtual machine. We use this server as our target node to deploy our calculator docker image.

We need to first install ansible on our local machine.

```
$sudo apt install ansible
```

We require ssh to connect our host machine to servers. Install OpenSSH on both host and remote machines.

```
$ sudo apt install openssh-server
```

Now we need to connect the host and remote machine under the Jenkins server. So we type the command

\$Sudo su jenkins.

We enter into Jenkins mode. We then run the following commands on the host machine. To get the IP address of the remote machine run the ifconfig command on the remote machine.

1. ssh-keygen -t rsa
Press enter to continue
2. ssh-copy-id remoteuser@remoteipaddress

```
bharath@bharath-VirtualBox:~$ sudo su jenkins
[sudo] password for bharath:
jenkins@bharath-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
/var/lib/jenkins/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ao6dQidHu8V4znjyPzGhWko7iBNRukBz0gj5LhZos jenkins@bharath-VirtualBox
The key's randomart image is:
+---[RSA 3072]-----+
|
|   o
|  . o .
| . + o .
|+=.+. .
|*o*. o.S
|oBo. .+=
|**= .*
|*.o= O.=
|Eoo.=oXoo.
+---[SHA256]-----+
jenkins@bharath-VirtualBox:~$ ssh-copy-id bharath@172.16.134.184
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/lib/jenkins/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
bharath@172.16.134.184's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'bharath@172.16.134.184'"
and check to make sure that only the key(s) you wanted were added.

jenkins@bharath-VirtualBox:~$
```

Now we have established a connection between the host and remote machines. Install the docker on the remote machine and try to run docker without sudo on the remote machine. The steps have been mentioned in the section of Docker.

Configuring Ansible on Jenkins:

Install Ansible plugins on the Jenkins server. Now got to Manage Jenkins->Global Tool Configuration. Click on add Ansible.

Ansible

Ansible Installations

Add Ansible

Ansible
Name

Ansible

Path to ansible executables directory

/usr/bin/

☐ Install automatically ?

Delete Ansible

Add Ansible

List of Ansible installations on this system

To get the path of Ansible run the following command in the terminal.
\$which ansible

Add the following pipeline script to the previous script.

```
stage('Step 5 : Ansible Deploy') {  
  steps {  
    //Ansible Deploy to remote server (managed host)  
    ansiblePlaybook colorized: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inventory', playbook: 'p2.yml'  
  }  
}
```

We can generate the above pipeline script using the pipeline syntax command.

Sample Step

ansiblePlaybook: invoke an ansible playbook ▼

ansiblePlaybook ?

Ansible tool

Ansible ▼

Playbook File path in workspace

p2.yml

Inventory File path in workspace

inventory

SSH connection credentials

- none - ▼ [Add](#)

Vault credentials

- none - ▼ [Add](#)

☐ Use become

Become username

root

☐ Use sudo (deprecated)

Sudo username (deprecated)

root

Host subset

Tags

Tags to skip

Task to start at

Number of parallel processes to use

☐ Disable the host SSH key check

☐ Colorized output

This will generate the pipeline script required for ansible deployment.

If we click build now and after successful completion of the process we can see the docker image deployed on our remote server.

	Step 1: Git clone	Step2 : Maven Build	Step3 : Build DockerImage	Push Image to Docker Hub	Ansible Deploy
is:	1s	14s	1s	34s	16s
in					
s)	734ms	10s	975ms	33s	3s

```
bharath@ellanti:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ellanti14/miniproject latest             fbd9f3073d62       24 hours ago       668MB
hello-world          latest             feb5d9fea6a5       6 months ago       13.3kB
bharath@ellanti:~$
```

We can run this docker image using the docker run command on our remote machine.

EIK Stack - Monitoring

Log4j:

Log4j is a fast, and flexible java-based logging system. Logging is a method of indicating a system's current state while it is functioning. Logs are used to collect and retain important data that may be evaluated at any time. We add the dependency of log4j in pom.xml for which screen shot is attached in the section of Maven.

Add log4j2.xml file in src/main/resources folder. Add logger functions in square root and other functions.

log4j2.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
  <Appenders>
    <File name="FileAppender" filename="MiniProject.log">
      <JSONLayout compact="true" eventEol="true">
        <KeyValuePair key="@timestamp" value="${date:yyyy-MM-dd'T'HH:mm:ss.SSSZ}" />
      </JSONLayout>
    </File>
  </Appenders>
  <Loggers>
    <Root level="debug">
      <AppenderRef ref="FileAppender"/>
    </Root>
  </Loggers>
</Configuration>

```

Logger functions in code

```

package calculator;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import java.lang.Math;
public class Calculator {
    private static final Logger logger = LogManager.getLogger(Calculator.class);
    public double SquareRoot(double x)
    {
        logger.info("Square Root");
        return Math.sqrt(x);
    }
}

```

Similarly add logger functions in other functions. Whenever a function is called a logger function in it will be called and it will be logged into log file. A log file will be generated after running the program in server.

Log file

```

{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":328554000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Factorial","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":329516000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Square Root","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":329773000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Square Root","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":330336000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Square Root","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":330624000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Square Root","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":331142000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Logarithm","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":331333000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Logarithm","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":331781000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Factorial","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":331979000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Factorial","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":332460000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Power","endOfBatch":false,"logg"},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":332651000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Power","endOfBatch":false,"logg"},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":333101000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Power","endOfBatch":false,"logg"},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":333283000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Power","endOfBatch":false,"logg"},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":333705000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Logarithm","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.929Z","offset":279298,"_source":{"epochSecond":1650279298,"nanoOfSecond":333903000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Logarithm","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.930Z","offset":279307,"_source":{"epochSecond":1650279307,"nanoOfSecond":886000000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Square Root","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.931Z","offset":279311,"_source":{"epochSecond":1650279311,"nanoOfSecond":885000000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Square Root","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.931Z","offset":279316,"_source":{"epochSecond":1650279316,"nanoOfSecond":862000000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Square Root","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.933Z","offset":279333,"_source":{"epochSecond":1650279333,"nanoOfSecond":186000000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Power","endOfBatch":false,"logg"},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.934Z","offset":279342,"_source":{"epochSecond":1650279342,"nanoOfSecond":927000000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Logarithm","endOfBatch":false,""},{"_type":"log","@version":1,"@timestamp":"2015-07-29T16:50:27.934Z","offset":279347,"_source":{"epochSecond":1650279347,"nanoOfSecond":961000000,"thread":"main","level":"INFO","loggerName":"calculator.Calculator","message":"Factorial","endOfBatch":false,""}

```

Elasticsearch, Logstash, and Kibana make up the ELK stack, which is an acronym for a stack that consists of three popular projects: Elasticsearch, Logstash, and Kibana. The ELK stack, also known as Elasticsearch, allows you to aggregate logs from all of your systems and applications, analyse them, and visualise them for application and infrastructure monitoring, speedier troubleshooting, security analytics, and more.

1. Elasticsearch is a distributed search and analytics engine built on Apache Lucene.
2. Logstash is an open-source data ingestion tool that allows you to collect data from a variety of sources, transform it, and send it to your desired destination.
3. Kibana is a data visualization and exploration tool for reviewing logs and events.

1. We first create an account in ELK stack web page.
2. We then create a deployment and then upload our log file.

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](#)

Visualize data from a log file

Upload your file, analyze its data, and optionally import the data into an Elasticsearch index.

The following file formats are supported:






-  Delimited text files, such as CSV and TSV
-  Newline-delimited JSON
-  Log files with a common format for the timestamp

You can upload files up to 100 MB.




[Select or drag and drop a file](#)


- Then press import and then give some index name and then click view index in discover.





File processedIndex createdIngest pipeline createdData uploadedData view created


 Import complete

Index	11
Data view	11
Ingest pipeline	11-pipeline
Documents ingested	22

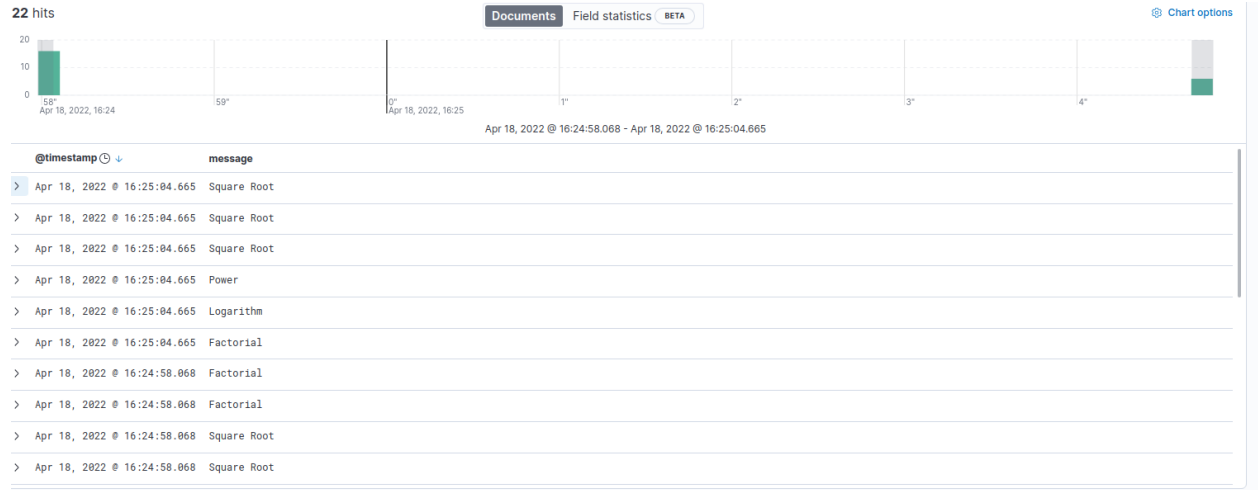
View index in Discover

Index Management

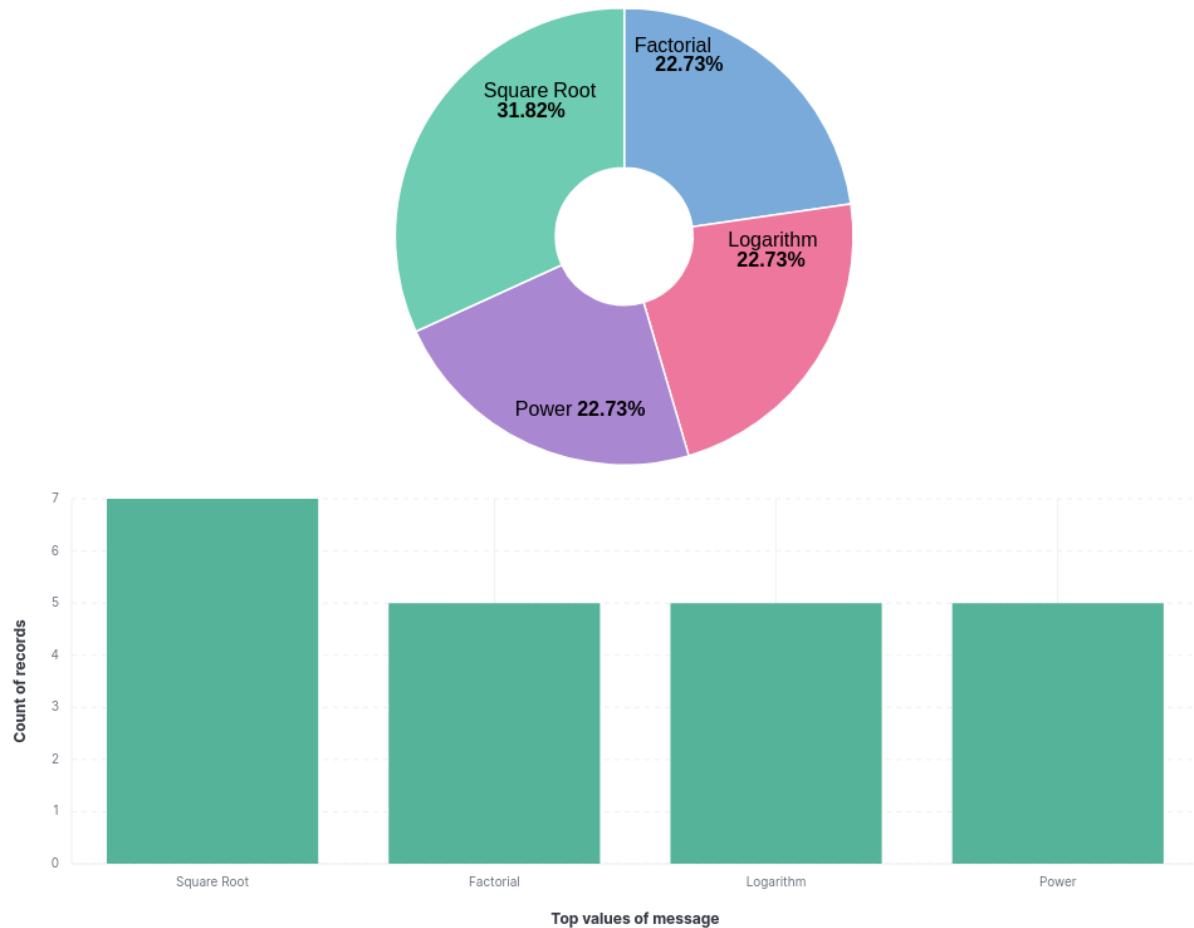
Data View Management

Create Filebeat configuration

- We can visualize different fields.



5. The below are the graphs for message fields.



Github Repo : <https://github.com/Bharath14/MiniProject>

Docker Hub repo :

<https://hub.docker.com/repository/docker/ellanti14/miniproject>