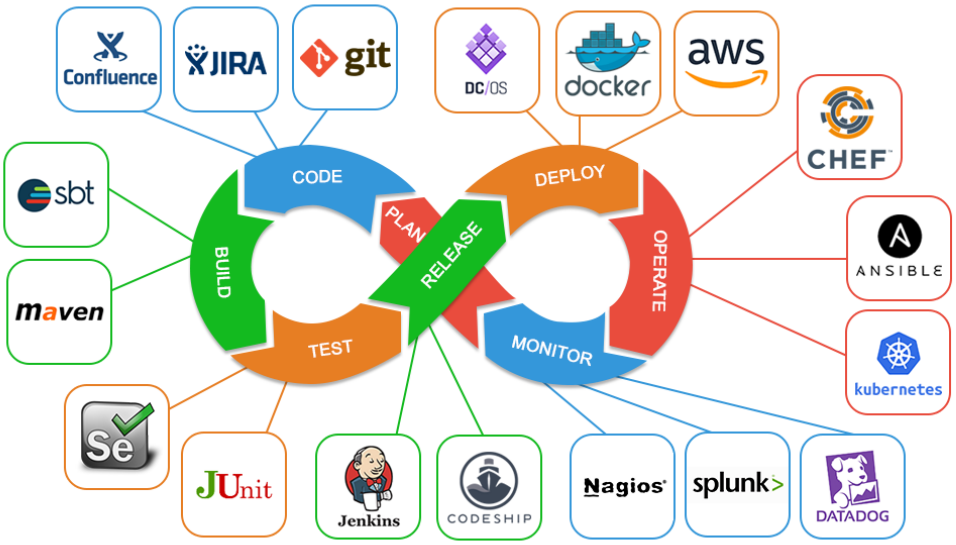
CS - 816 Software Production Engineering

DevOps Mini Project - Scientific Calculator



Prakhar Lad
MT2021096

GitHub Link: https://github.com/prakharlad123/scientificCalc.git
Docker Link: https://hub.docker.com/repository/docker/prakharavii/scienticficcalc

Table of Contents

1. Introduction 1

2. Source Code Management 3

3. Continuous Integration 14

4. Continuous Delivery 22

5. Continuous Deployment 24

6. Continuous Monitoring 26

7. Conclusion 31

8. Bibliography 32

1. Introduction

This report is designed for defining the details of a complete DevOps enabled application: Scientific Calculator. We have to automate the development, testing, and deployment pipeline with the help of dev-ops tools. The Scientific Calculator application which a Web-based application where users can perform the following operations:

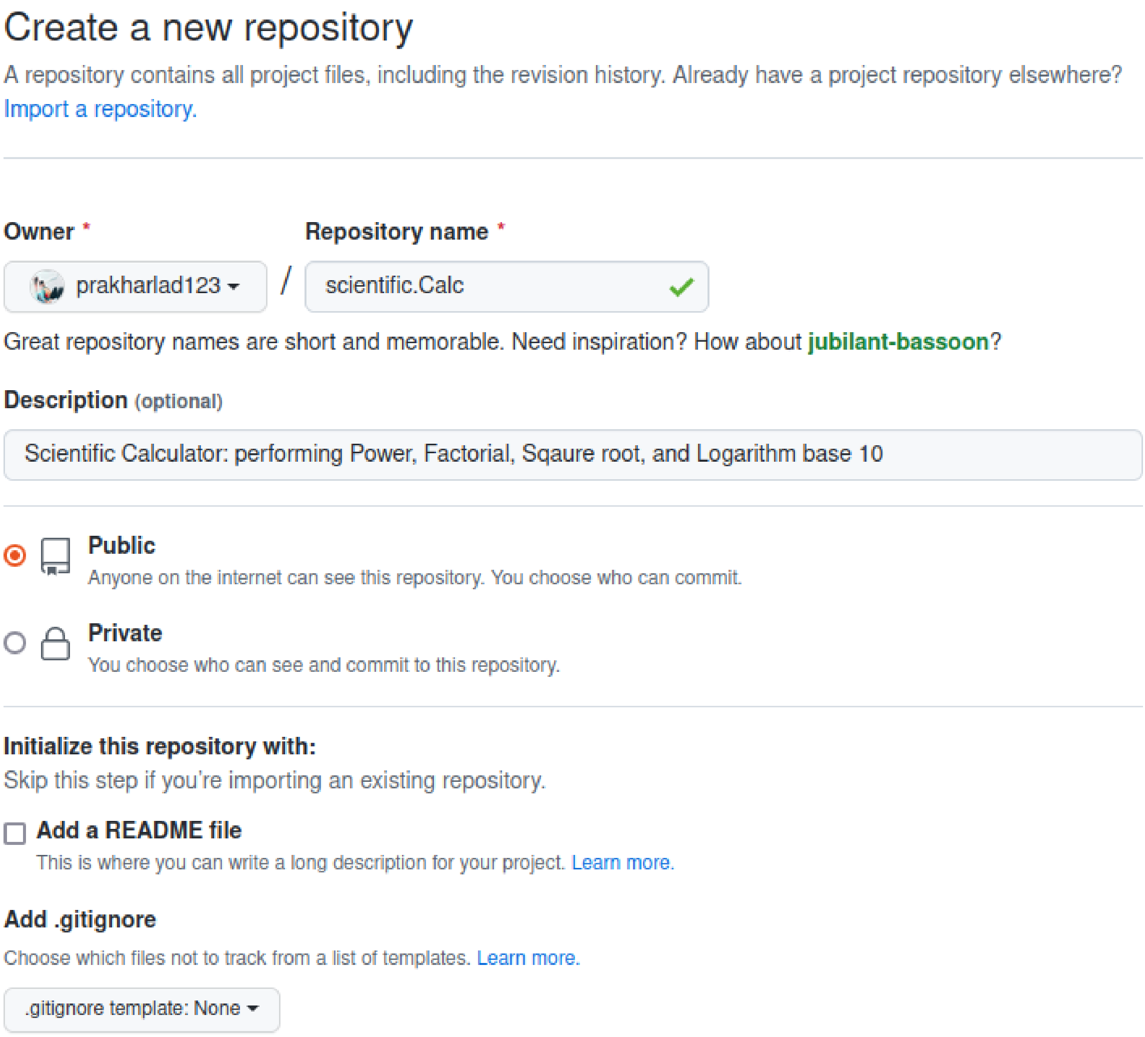
- ➔ Square root function - $\sqrt{x}$
- ➔ Factorial function - x!
- ➔ Logarithm (base-10) - $\log_{10}(x)$
- ➔ Power function - x^a

Tools included:

1. Development IDE: Intellij and Visual Studio Code
2. Programming Language: JAVA, HTML, CSS and JavaScript
3. Source Code Management: GitHub (https://github.com/prakharlad123/scientificCalc.git)
4. Docker image: https://hub.docker.com/repository/docker/prakharavii/scienticficcalc
5. Continuous Integration: Jenkins
6. Continuous Deployment: Ansible
7. Continuous Monitoring: ELK Stack

2. Source Code Management

2.1. Version Control System: GitHub



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * : prakharlad123

Repository name * : scientific.Calc ✓

Great repository names are short and memorable. Need inspiration? How about jubilant-bassoon?

Description (optional): Scientific Calculator: performing Power, Factorial, Sqaure root, and Logarithm base 10

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

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

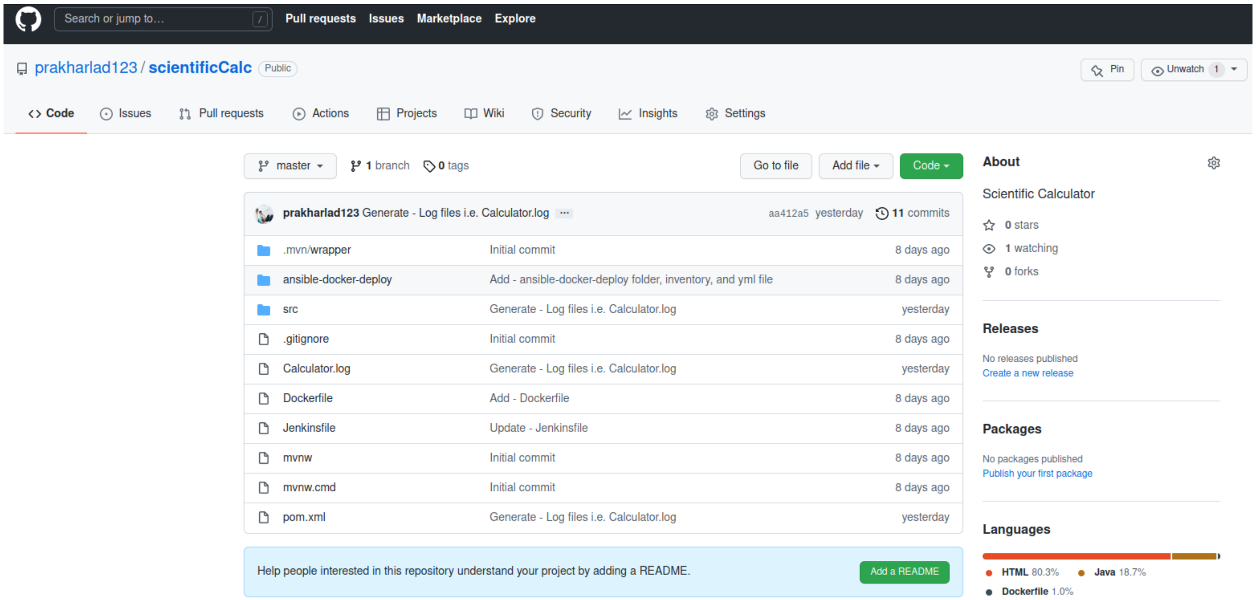
Initialize this repository with:
Skip this step if you're importing an existing repository.

□ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

Fig 1 : Creating Repository



prakharlad123 Generate - Log files i.e. Calculator.log … aa412a5 yesterday 11 commits

.mvn/wrapper	Initial commit	8 days ago
ansible-docker-deploy	Add - ansible-docker-deploy folder, inventory, and yml file	8 days ago
src	Generate - Log files i.e. Calculator.log	yesterday
.gitignore	Initial commit	8 days ago
Calculator.log	Generate - Log files i.e. Calculator.log	yesterday
Dockerfile	Add - Dockerfile	8 days ago
Jenkinsfile	Update - Jenkinsfile	8 days ago
mvnw	Initial commit	8 days ago
mvnw.cmd	Initial commit	8 days ago
pom.xml	Generate - Log files i.e. Calculator.log	yesterday

Help people interested in this repository understand your project by adding a README.

About

Scientific Calculator

0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● HTML 80.3% ● Java 18.7%
● Dockerfile 1.0%

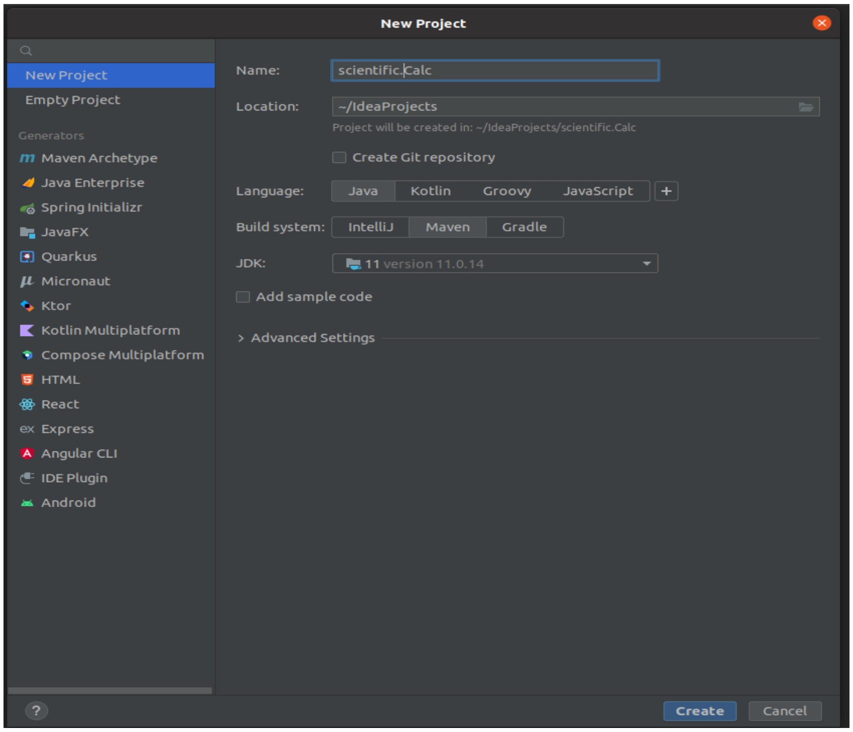
Fig 2 : Current Repository

Steps to create a GitHub Repository:

1. Login to the GitHub: https://www.github.com
2. Go to Your Repository
3. Create a new repository: scientificCalc (https://github.com/prakharlad123/scientificCalc.git)

2.2. Build Project: Intellij

In the **Scientific Calculator** program, **Apache Maven** is responsible for managing dependencies and building the project. It is Maven who finally outputs the SNAPSHOT jar of the project that has the compiled classes along with other classes the project depends on.



Name: scientific.Calc

Location: ~/IdeaProjects

Project will be created in: ~/IdeaProjects/scientific.Calc

□ Create Git repository

Language: Java | Kotlin | Groovy | JavaScript | +

Build system: IntelliJ | Maven | Gradle

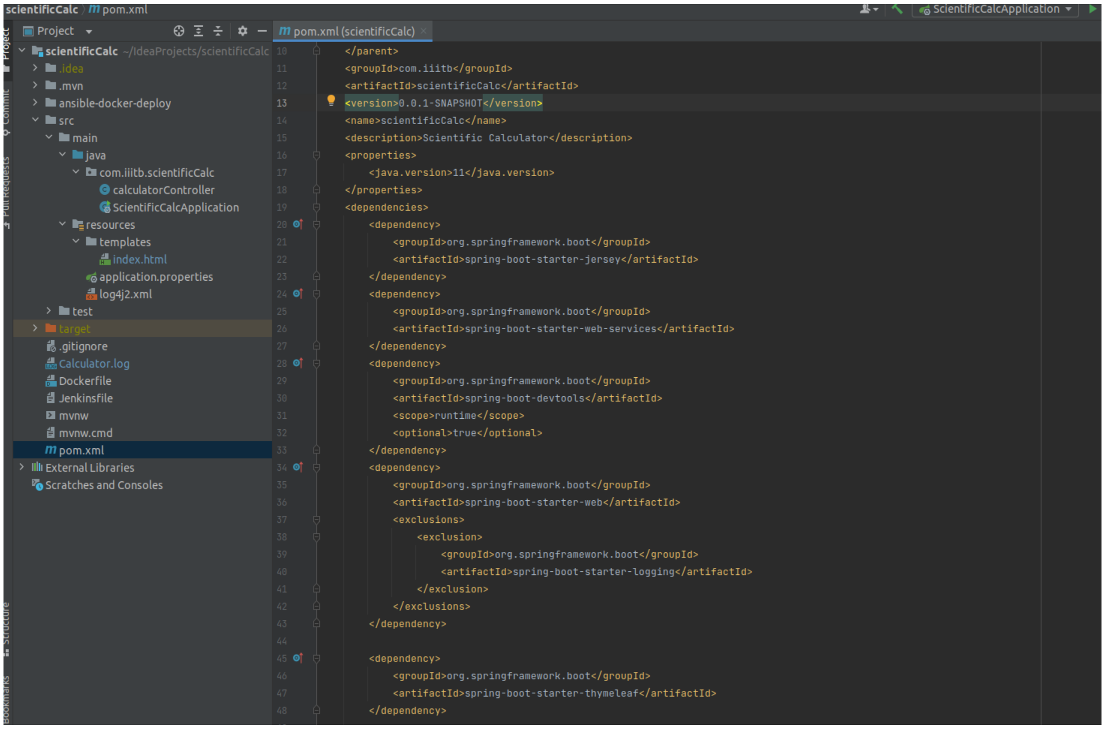
JDK: 11 version 11.0.14

□ Add sample code

› Advanced Settings

Fig 3 : Building Project

The **pom.xml** file in the project contains all the dependencies.

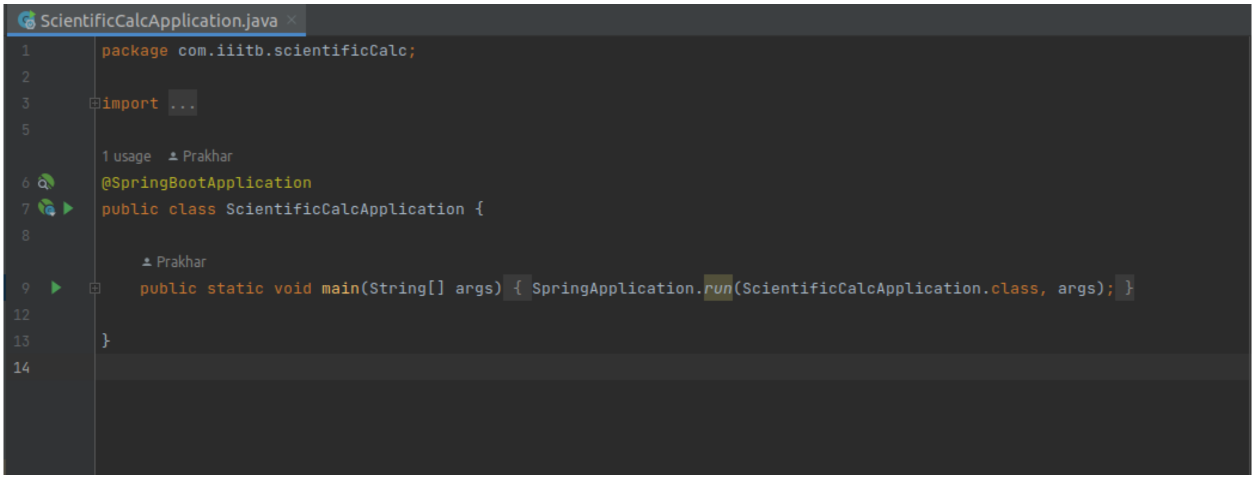


```xml
</parent>
<groupId>com.iiitb</groupId>
<artifactId>scientificCalc</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>scientificCalc</name>
<description>Scientific Calculator</description>
<properties>
    <java.version>11</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jersey</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web-services</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <exclusions>
            <exclusion>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-logging</artifactId>
            </exclusion>
        </exclusions>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
```

Fig 4 : pom.xml

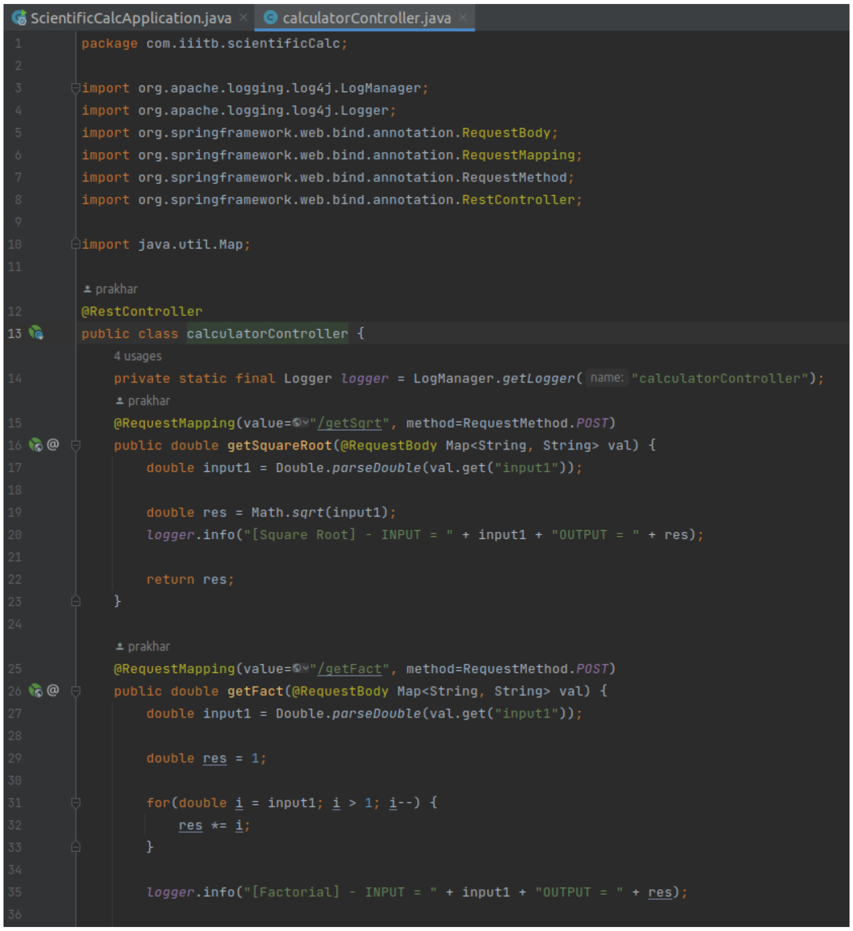
2.3. Source Code and Output: Intellij and Web-app

2.3.1. Java Code



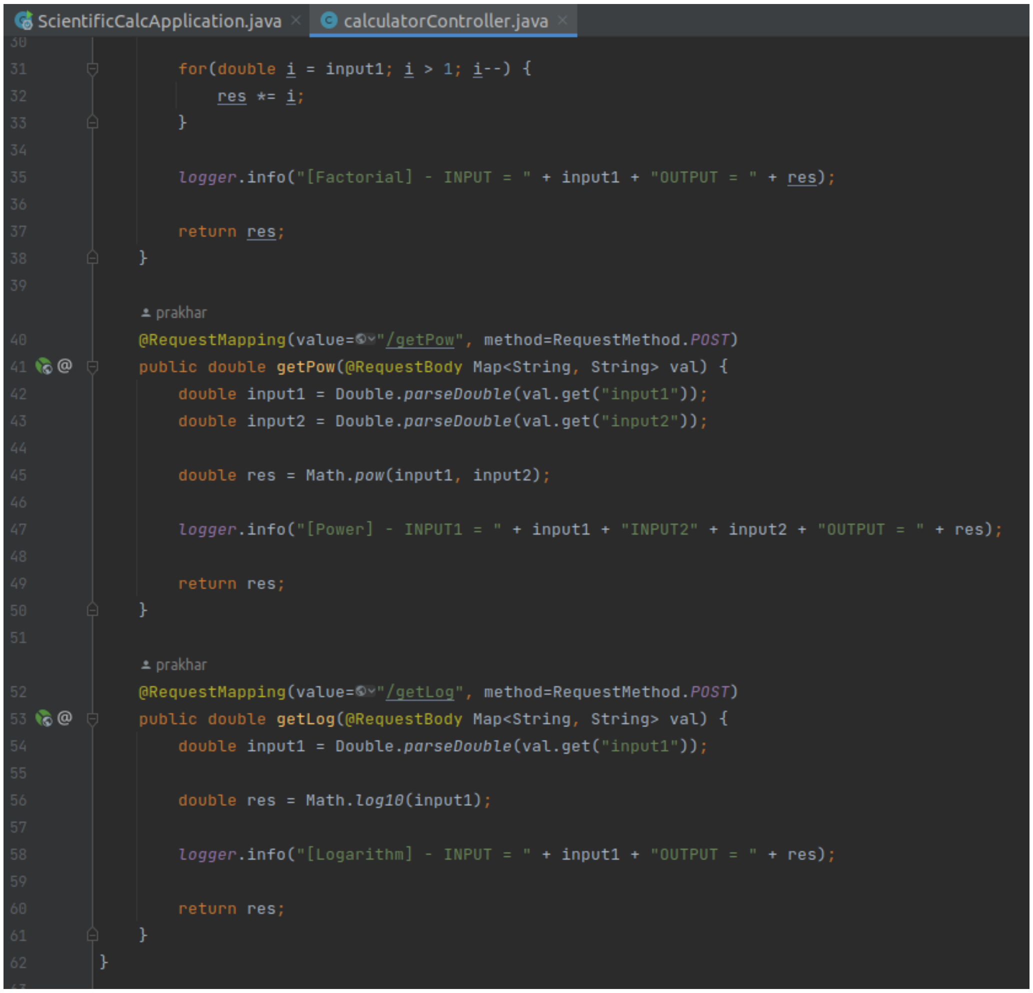
```
package com.iiitb.scientificCalc;

import ...

1 usage  Prakhar
@SpringBootApplication
public class ScientificCalcApplication {

    Prakhar
    public static void main(String[] args) { SpringApplication.run(ScientificCalcApplication.class, args); }

}
```

Fig 5 : scientificCalcApplication



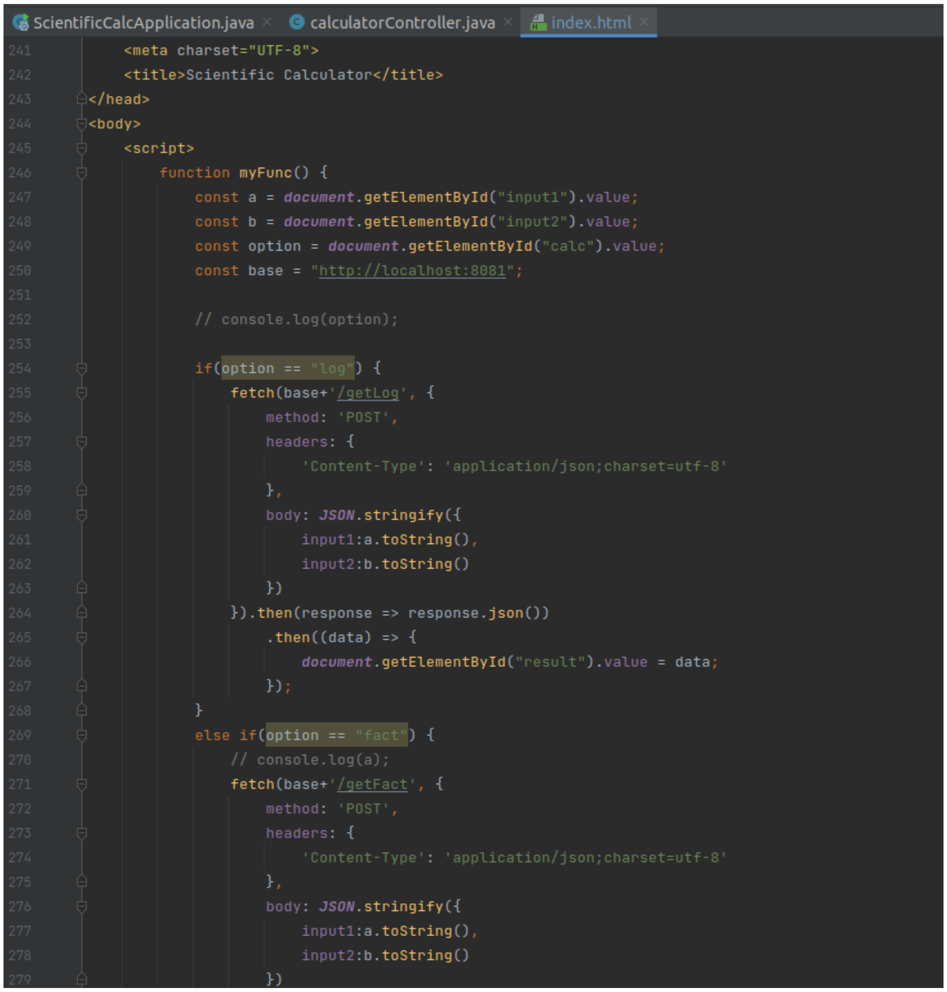
```
package com.iiitb.scientificCalc;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

prakhar
@RestController
public class calculatorController {
    4 usages
    private static final Logger logger = LogManager.getLogger( name: "calculatorController");
    prakhar
    @RequestMapping(value="/getSqrt", method=RequestMethod.POST)
    public double getSquareRoot(@RequestBody Map<String, String> val) {
        double input1 = Double.parseDouble(val.get("input1"));

        double res = Math.sqrt(input1);
        logger.info("[Square Root] - INPUT = " + input1 + "OUTPUT = " + res);

        return res;
    }

    prakhar
    @RequestMapping(value="/getFact", method=RequestMethod.POST)
    public double getFact(@RequestBody Map<String, String> val) {
        double input1 = Double.parseDouble(val.get("input1"));

        double res = 1;

        for(double i = input1; i > 1; i--) {
            res *= i;
        }

        logger.info("[Factorial] - INPUT = " + input1 + "OUTPUT = " + res);

```

Fig 6 : calculatorController



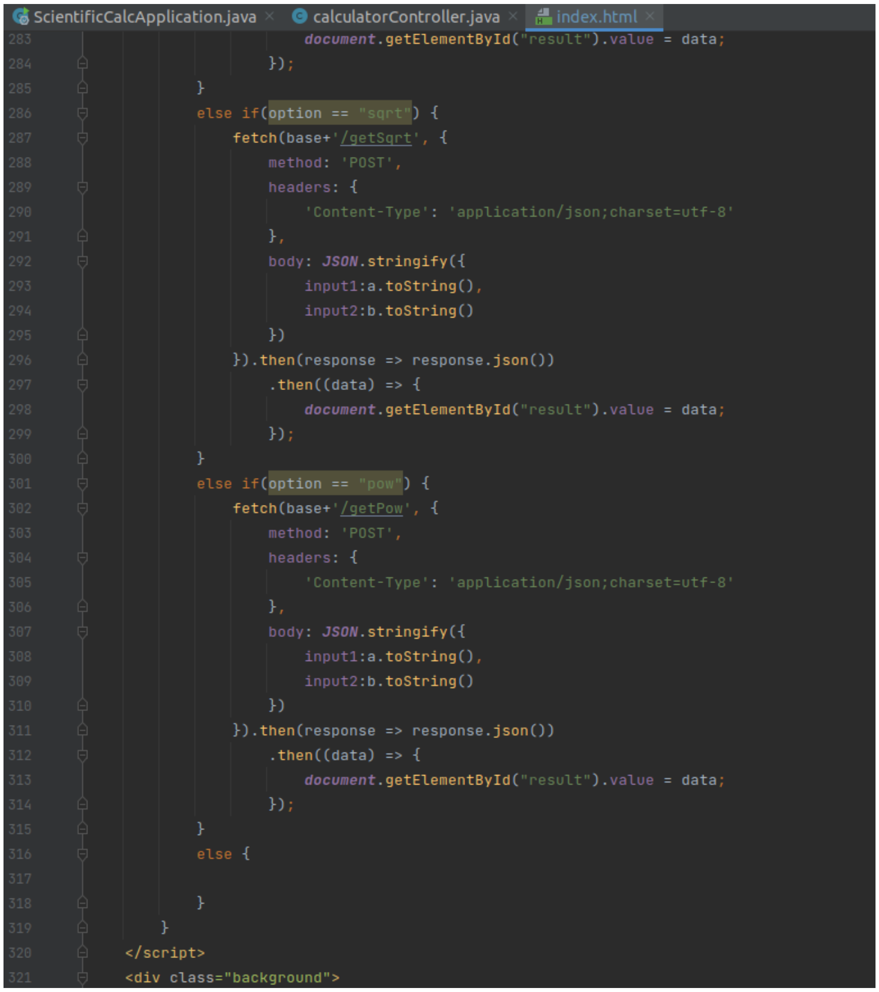
```java
        for(double i = input1; i > 1; i--) {
            res *= i;
        }

        logger.info("[Factorial] - INPUT = " + input1 + "OUTPUT = " + res);

        return res;
    }

    @RequestMapping(value="/getPow", method=RequestMethod.POST)
    public double getPow(@RequestBody Map<String, String> val) {
        double input1 = Double.parseDouble(val.get("input1"));
        double input2 = Double.parseDouble(val.get("input2"));

        double res = Math.pow(input1, input2);

        logger.info("[Power] - INPUT1 = " + input1 + "INPUT2" + input2 + "OUTPUT = " + res);

        return res;
    }

    @RequestMapping(value="/getLog", method=RequestMethod.POST)
    public double getLog(@RequestBody Map<String, String> val) {
        double input1 = Double.parseDouble(val.get("input1"));

        double res = Math.log10(input1);

        logger.info("[Logarithm] - INPUT = " + input1 + "OUTPUT = " + res);

        return res;
    }
}
```

Fig 7 : calculatorController (cont...)



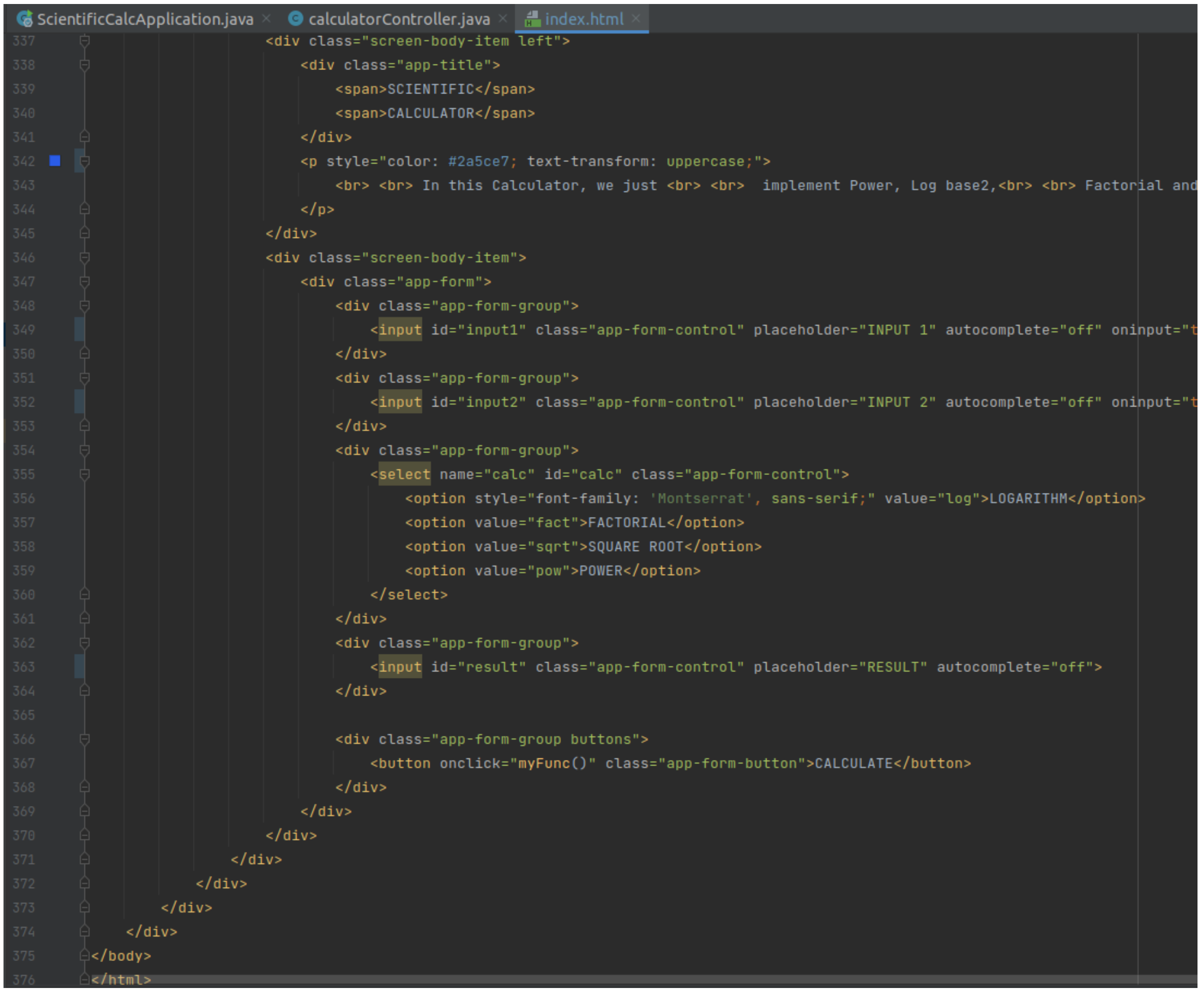
```
    <meta charset="UTF-8">
    <title>Scientific Calculator</title>
</head>
<body>
    <script>
        function myFunc() {
            const a = document.getElementById("input1").value;
            const b = document.getElementById("input2").value;
            const option = document.getElementById("calc").value;
            const base = "http://localhost:8081";

            // console.log(option);

            if(option == "log") {
                fetch(base+'/getLog', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json;charset=utf-8'
                    },
                    body: JSON.stringify({
                        input1:a.toString(),
                        input2:b.toString()
                    })
                }).then(response => response.json())
                    .then((data) => {
                        document.getElementById("result").value = data;
                    });
            }
            else if(option == "fact") {
                // console.log(a);
                fetch(base+'/getFact', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json;charset=utf-8'
                    },
                    body: JSON.stringify({
                        input1:a.toString(),
                        input2:b.toString()
                    })
```

Fig 8 : index.html (Front End Code)



```
                    document.getElementById("result").value = data;
                });
        }
        else if(option == "sqrt") {
            fetch(base+'/getSqrt', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json;charset=utf-8'
                },
                body: JSON.stringify({
                    input1:a.toString(),
                    input2:b.toString()
                })
            }).then(response => response.json())
                .then((data) => {
                    document.getElementById("result").value = data;
                });
        }
        else if(option == "pow") {
            fetch(base+'/getPow', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json;charset=utf-8'
                },
                body: JSON.stringify({
                    input1:a.toString(),
                    input2:b.toString()
                })
            }).then(response => response.json())
                .then((data) => {
                    document.getElementById("result").value = data;
                });
        }
        else {

        }
    }
</script>
<div class="background">
```

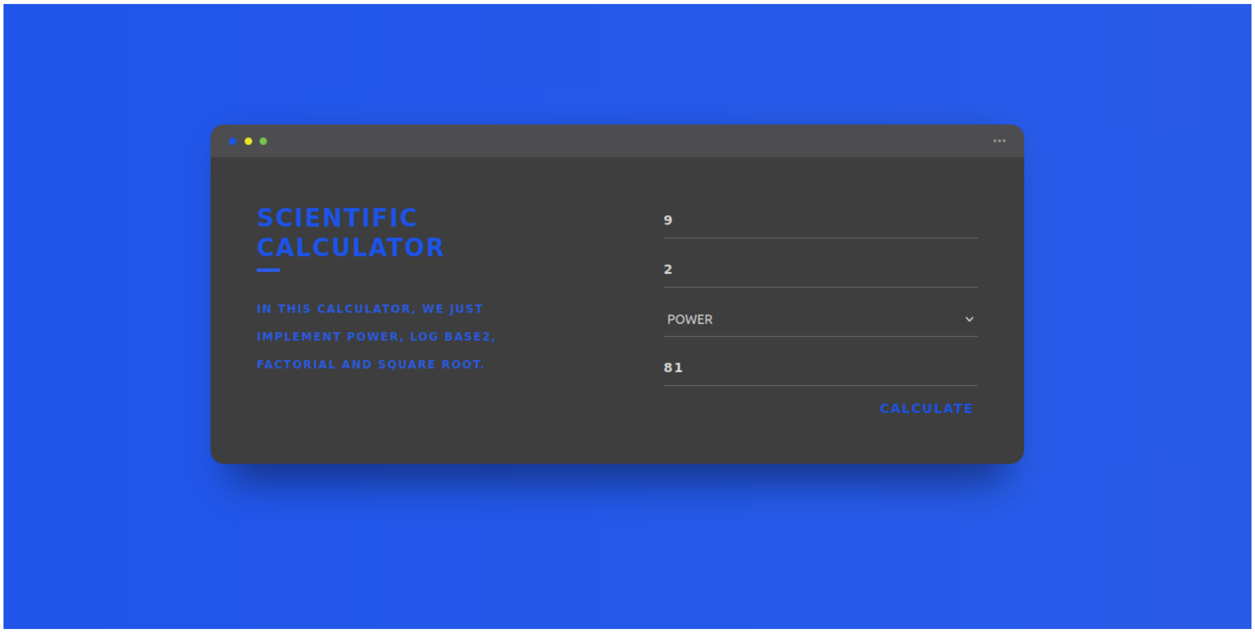
Fig 9 : index.html (JavaScript)



```html
                    <div class="screen-body-item left">
                        <div class="app-title">
                            <span>SCIENTIFIC</span>
                            <span>CALCULATOR</span>
                        </div>
                        <p style="color: #2a5ce7; text-transform: uppercase;">
                            <br> <br> In this Calculator, we just <br> <br>  implement Power, Log base2,<br> <br> Factorial and
                        </p>
                    </div>
                    <div class="screen-body-item">
                        <div class="app-form">
                            <div class="app-form-group">
                                <input id="input1" class="app-form-control" placeholder="INPUT 1" autocomplete="off" oninput="t
                            </div>
                            <div class="app-form-group">
                                <input id="input2" class="app-form-control" placeholder="INPUT 2" autocomplete="off" oninput="t
                            </div>
                            <div class="app-form-group">
                                <select name="calc" id="calc" class="app-form-control">
                                    <option style="font-family: 'Montserrat', sans-serif;" value="log">LOGARITHM</option>
                                    <option value="fact">FACTORIAL</option>
                                    <option value="sqrt">SQUARE ROOT</option>
                                    <option value="pow">POWER</option>
                                </select>
                            </div>
                            <div class="app-form-group">
                                <input id="result" class="app-form-control" placeholder="RESULT" autocomplete="off">
                            </div>

                            <div class="app-form-group buttons">
                                <button onclick="myFunc()" class="app-form-button">CALCULATE</button>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

Fig 10 : index.html (HTML)

2.3.2. Front End



SCIENTIFIC CALCULATOR

IN THIS CALCULATOR, WE JUST IMPLEMENT POWER, LOG BASE2, FACTORIAL AND SQUARE ROOT.

9

2

POWER

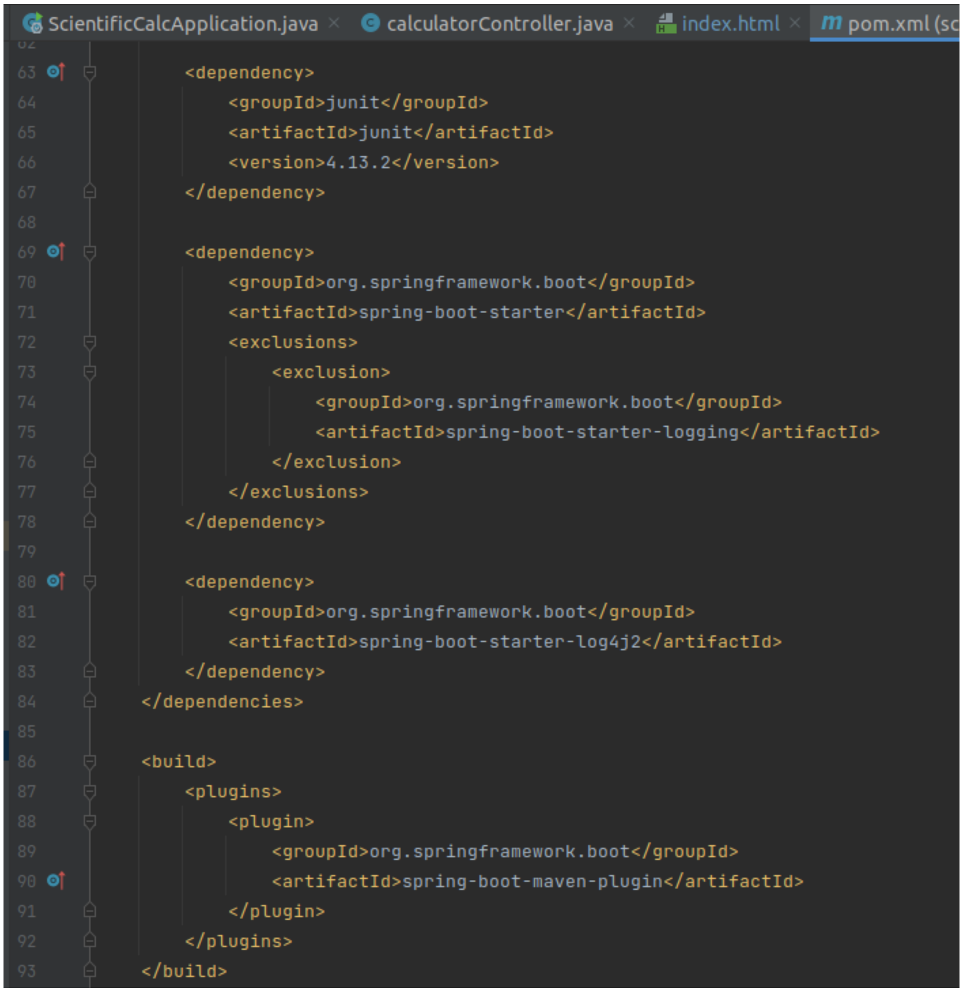
81

Fig 11 : Web Page

2.4. Log Management:

Logging is an important feature that helps developers to trace out the errors. It provides the ability to capture the log file. In our application we are using log4j2 to generate the loggers.

1. To set it up we need to add “**org.apache.logging.log4j**” dependency in pom.xml.

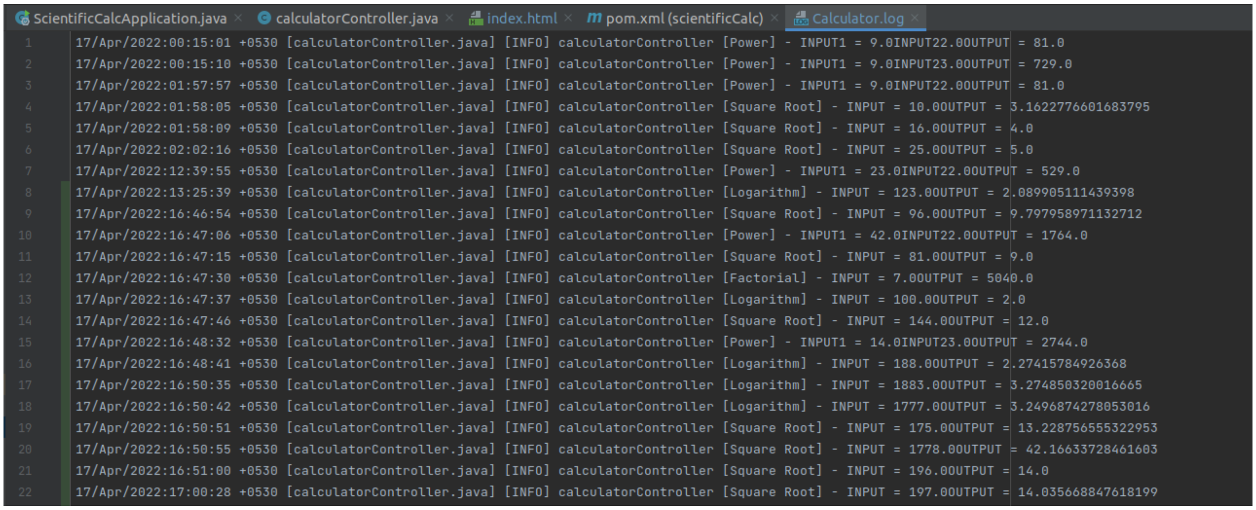


```
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-log4j2</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

Fig 12 : log4j2 dependencies

2. Then create a file with the name: “**log4j2.xml**” under the resources folder.
3. And then write the logger statements inside the code.

After all these steps a log file will get generated (with name and path as specified in log4j2.xml).



```
17/Apr/2022:00:15:01 +0530 [calculatorController.java] [INFO] calculatorController [Power] - INPUT1 = 9.0INPUT22.0OUTPUT = 81.0
17/Apr/2022:00:15:10 +0530 [calculatorController.java] [INFO] calculatorController [Power] - INPUT1 = 9.0INPUT23.0OUTPUT = 729.0
17/Apr/2022:01:57:57 +0530 [calculatorController.java] [INFO] calculatorController [Power] - INPUT1 = 9.0INPUT22.0OUTPUT = 81.0
17/Apr/2022:01:58:05 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 10.0OUTPUT = 3.1622776601683795
17/Apr/2022:01:58:09 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 16.0OUTPUT = 4.0
17/Apr/2022:02:02:16 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 25.0OUTPUT = 5.0
17/Apr/2022:12:39:55 +0530 [calculatorController.java] [INFO] calculatorController [Power] - INPUT1 = 23.0INPUT22.0OUTPUT = 529.0
17/Apr/2022:13:25:39 +0530 [calculatorController.java] [INFO] calculatorController [Logarithm] - INPUT = 123.0OUTPUT = 2.089905111439398
17/Apr/2022:16:46:54 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 96.0OUTPUT = 9.797958971132712
17/Apr/2022:16:47:06 +0530 [calculatorController.java] [INFO] calculatorController [Power] - INPUT1 = 42.0INPUT22.0OUTPUT = 1764.0
17/Apr/2022:16:47:15 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 81.0OUTPUT = 9.0
17/Apr/2022:16:47:30 +0530 [calculatorController.java] [INFO] calculatorController [Factorial] - INPUT = 7.0OUTPUT = 5040.0
17/Apr/2022:16:47:37 +0530 [calculatorController.java] [INFO] calculatorController [Logarithm] - INPUT = 100.0OUTPUT = 2.0
17/Apr/2022:16:47:46 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 144.0OUTPUT = 12.0
17/Apr/2022:16:48:32 +0530 [calculatorController.java] [INFO] calculatorController [Power] - INPUT1 = 14.0INPUT23.0OUTPUT = 2744.0
17/Apr/2022:16:48:41 +0530 [calculatorController.java] [INFO] calculatorController [Logarithm] - INPUT = 188.0OUTPUT = 2.27415784926368
17/Apr/2022:16:50:35 +0530 [calculatorController.java] [INFO] calculatorController [Logarithm] - INPUT = 1883.0OUTPUT = 3.274850320016665
17/Apr/2022:16:50:42 +0530 [calculatorController.java] [INFO] calculatorController [Logarithm] - INPUT = 1777.0OUTPUT = 3.2496874278053016
17/Apr/2022:16:50:51 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 175.0OUTPUT = 13.228756555322953
17/Apr/2022:16:50:55 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 1778.0OUTPUT = 42.16633728461603
17/Apr/2022:16:51:00 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 196.0OUTPUT = 14.0
17/Apr/2022:17:00:28 +0530 [calculatorController.java] [INFO] calculatorController [Square Root] - INPUT = 197.0OUTPUT = 14.035668847618199
```

Fig 13 : Calculator.log (Log Files)

2.5. Build .jar file:

To create the .jar file, we have to open terminal and then run “***mvn install***”

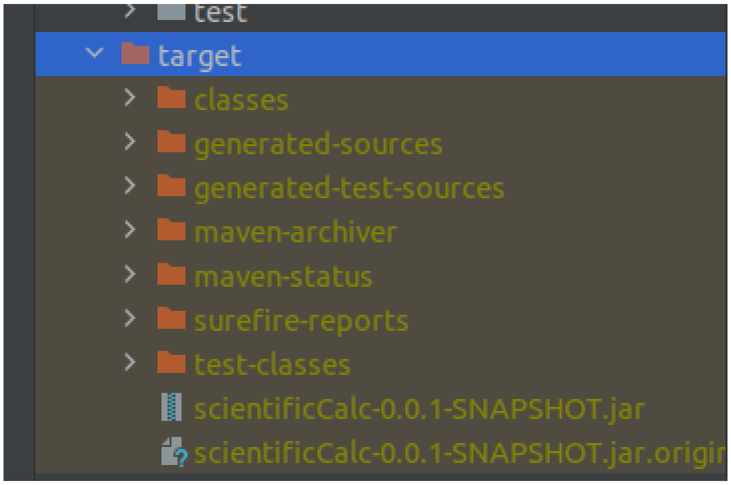


Fig 14 : .jar file

3. Continuous Integration

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It's a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository where builds and tests then run. Automated tools are used to assert the new code's correctness before integration.

A source code version control system is the crux of the CI process. The version control system is also supplemented with other checks like automated code quality tests, syntax style review tools, and more.

3.1. Jenkins Installation:

1. First, add the repository key to the system:

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
```

2. When the key is added, the system will return OK. Next, append the Debian package repository address to the server's sources.list:

```
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
```

3. When both of these are in place, run update so that apt will use the new repository:

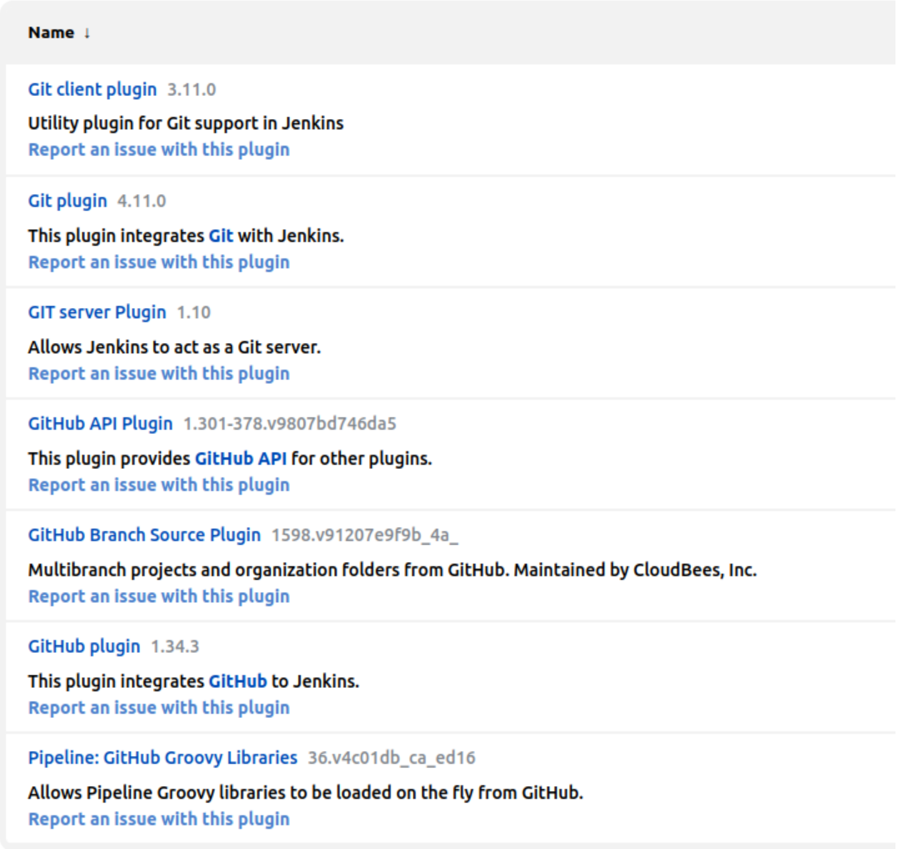
```
sudo apt update
```

4. Finally, install Jenkins and its dependencies:

```
sudo apt install jenkins
```

3.2. Install Plugins and Add Credentials in Jenkins:

Install **Git**, **Maven**, **JUnit**, **Ansible**, **Docker** plugin by going to **Manage Jenkins** → **Manage Plugins**



Name ↓

Git client plugin 3.11.0
Utility plugin for Git support in Jenkins
Report an issue with this plugin

Git plugin 4.11.0
This plugin integrates Git with Jenkins.
Report an issue with this plugin

GIT server Plugin 1.10
Allows Jenkins to act as a Git server.
Report an issue with this plugin

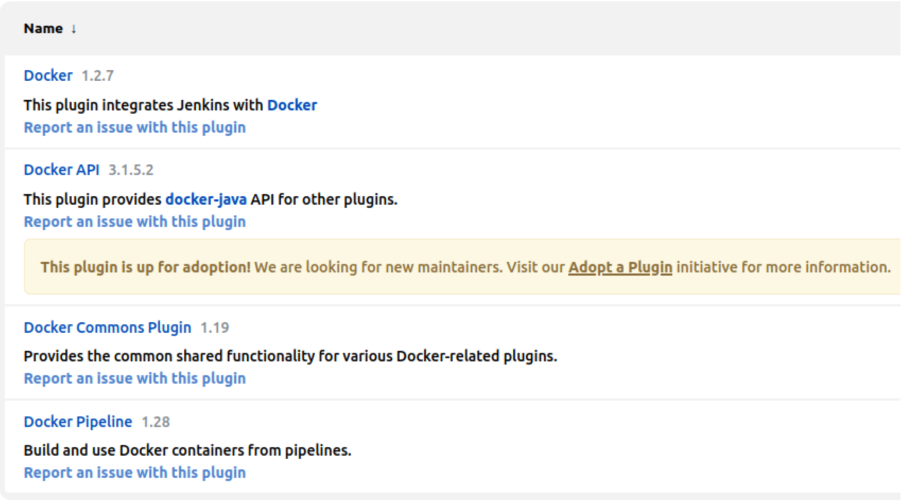
GitHub API Plugin 1.301-378.v9807bd746da5
This plugin provides GitHub API for other plugins.
Report an issue with this plugin

GitHub Branch Source Plugin 1598.v91207e9f9b_4a_
Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc.
Report an issue with this plugin

GitHub plugin 1.34.3
This plugin integrates GitHub to Jenkins.
Report an issue with this plugin

Pipeline: GitHub Groovy Libraries 36.v4c01db_ca_ed16
Allows Pipeline Groovy libraries to be loaded on the fly from GitHub.
Report an issue with this plugin

Fig 15 : Git plugins



Name ↓

Docker 1.2.7
This plugin integrates Jenkins with Docker
Report an issue with this plugin

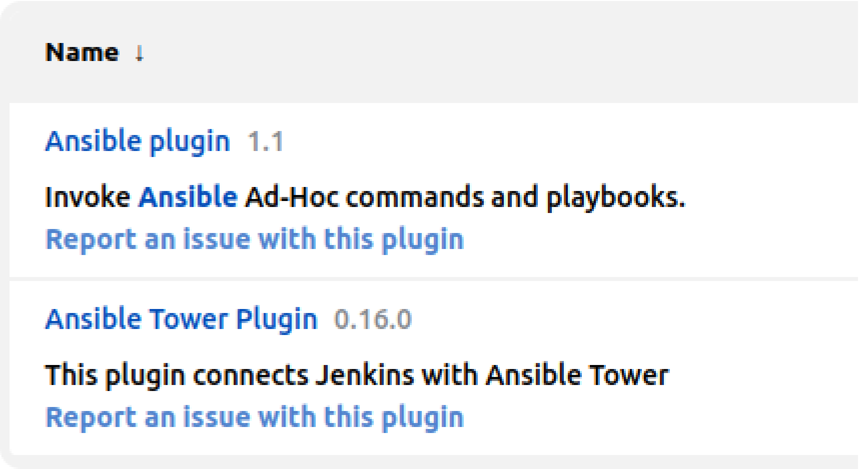
Docker API 3.1.5.2
This plugin provides docker-java API for other plugins.
Report an issue with this plugin

This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

Docker Commons Plugin 1.19
Provides the common shared functionality for various Docker-related plugins.
Report an issue with this plugin

Docker Pipeline 1.28
Build and use Docker containers from pipelines.
Report an issue with this plugin

Fig 16 : Docker plugins



Name ↓

Ansible plugin 1.1

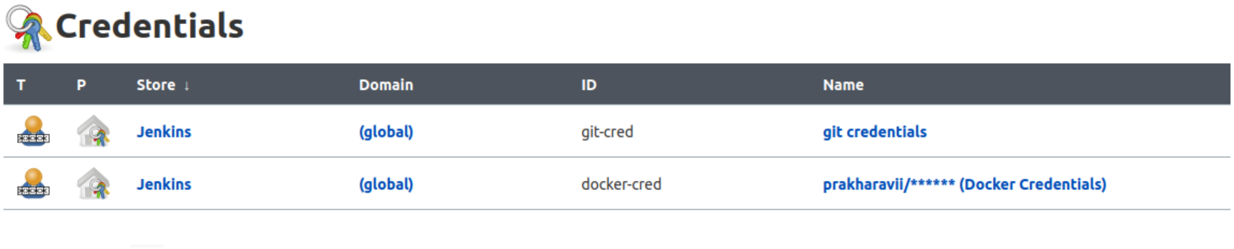
Invoke Ansible Ad-Hoc commands and playbooks.
Report an issue with this plugin

Ansible Tower Plugin 0.16.0

This plugin connects Jenkins with Ansible Tower
Report an issue with this plugin

Fig 17 : Ansible plugins

Add git and docker hub credentials to Jenkins by going to Credentials.



Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	git-cred	git credentials
		Jenkins	(global)	docker-cred	prakharavii/****** (Docker Credentials)

Fig 18 : Credentials

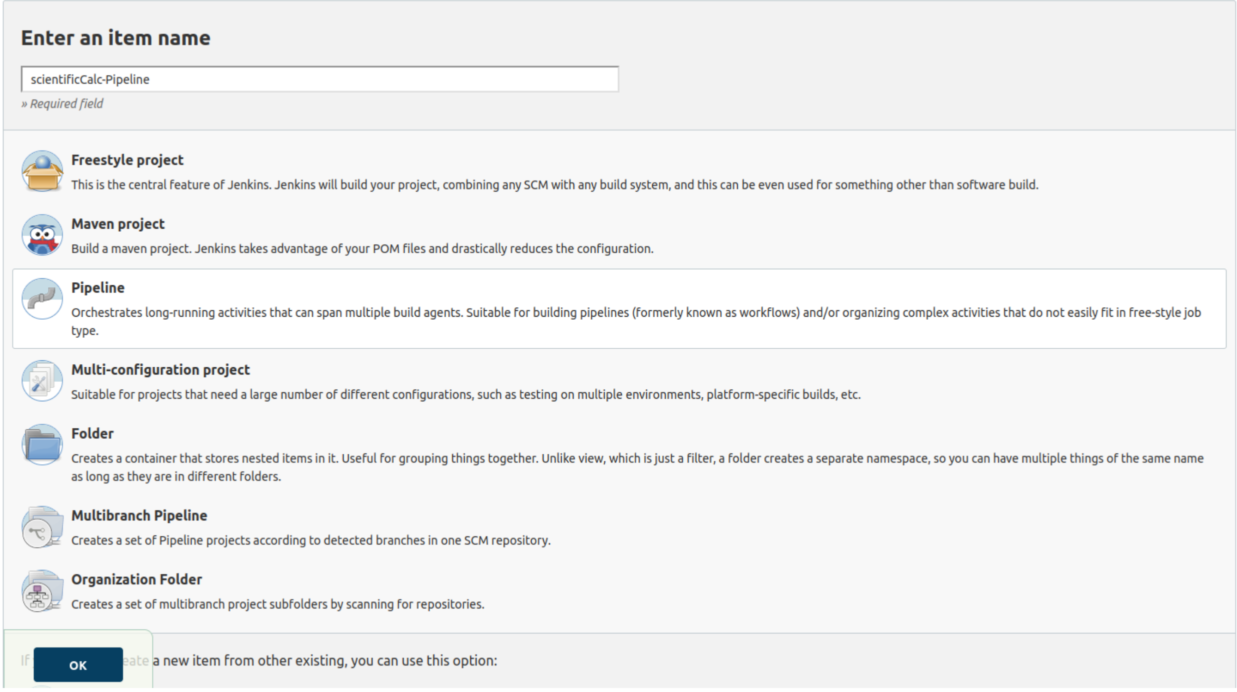
3.3. Jenkins Pipeline:

Jenkins Pipeline (or simply "Pipeline") is **a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins**. A continuous delivery pipeline is an automated expression of your process for getting software from version control right through to your users and customers.

To create Jenkins pipeline, follow the given steps:

1. Go to Jenkins Dashboard.
2. Click on new Item.

3. Enter Project pipeline name.
4. Click on pipeline.
5. Click OK.



Enter an item name

scientificCalc-Pipeline

» 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.

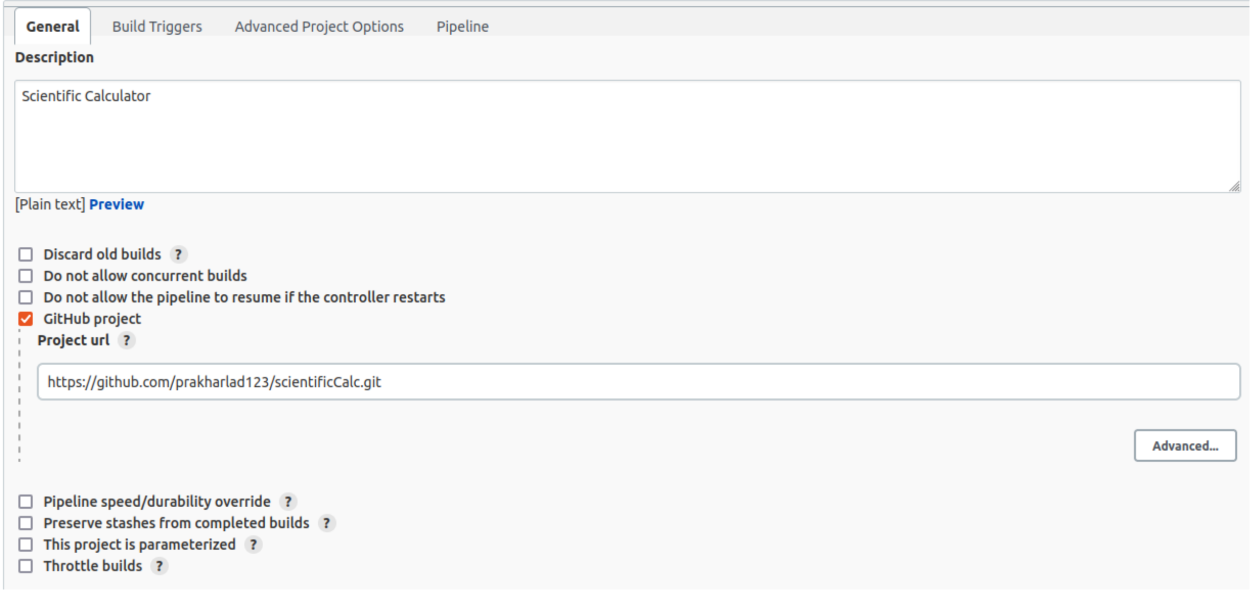
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

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

If OK eate a new item from other existing, you can use this option:

Fig 19 : New Pipeline

6. After this a new window will get opened. Provide description.



Description

Scientific Calculator

[Plain text] **Preview**

☐ Discard old builds ?
☐ Do not allow concurrent builds
☐ Do not allow the pipeline to resume if the controller restarts
☑ GitHub project

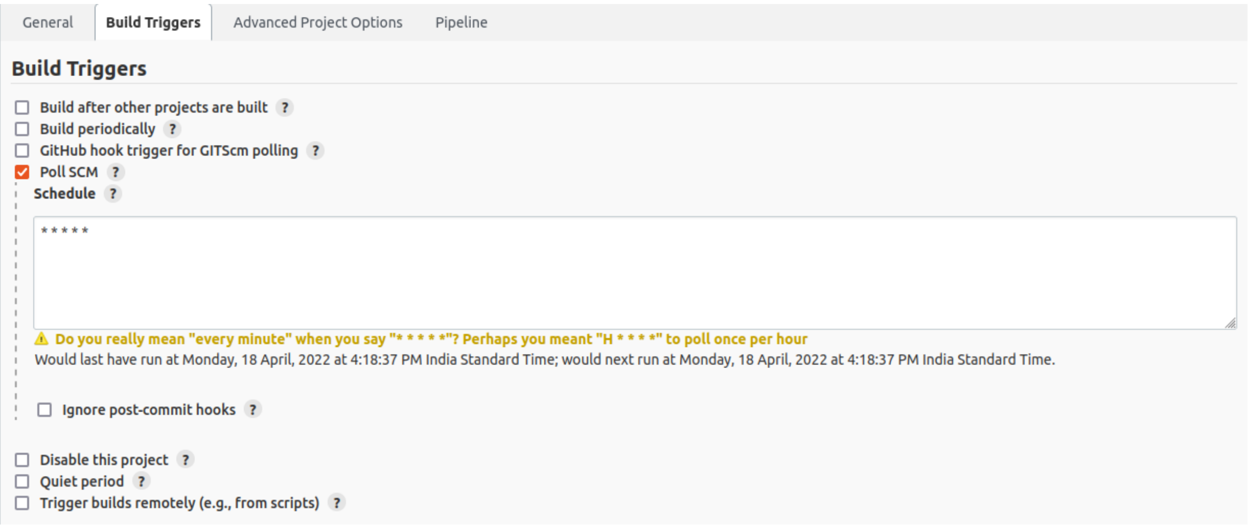
Project url ?

https://github.com/prakharlad123/scientificCalc.git

☐ Pipeline speed/durability override ?
☐ Preserve stashes from completed builds ?
☐ This project is parameterized ?
☐ Throttle builds ?

Fig 20 : General Setting

7. Select poll SCM and give input like: * * * * * to poll for the SCM changes every minute.



Build Triggers

□ Build after other projects are built ?
□ Build periodically ?
□ GitHub hook trigger for GITScm polling ?
☑ Poll SCM ?

Schedule ?

```
* * * * *
```

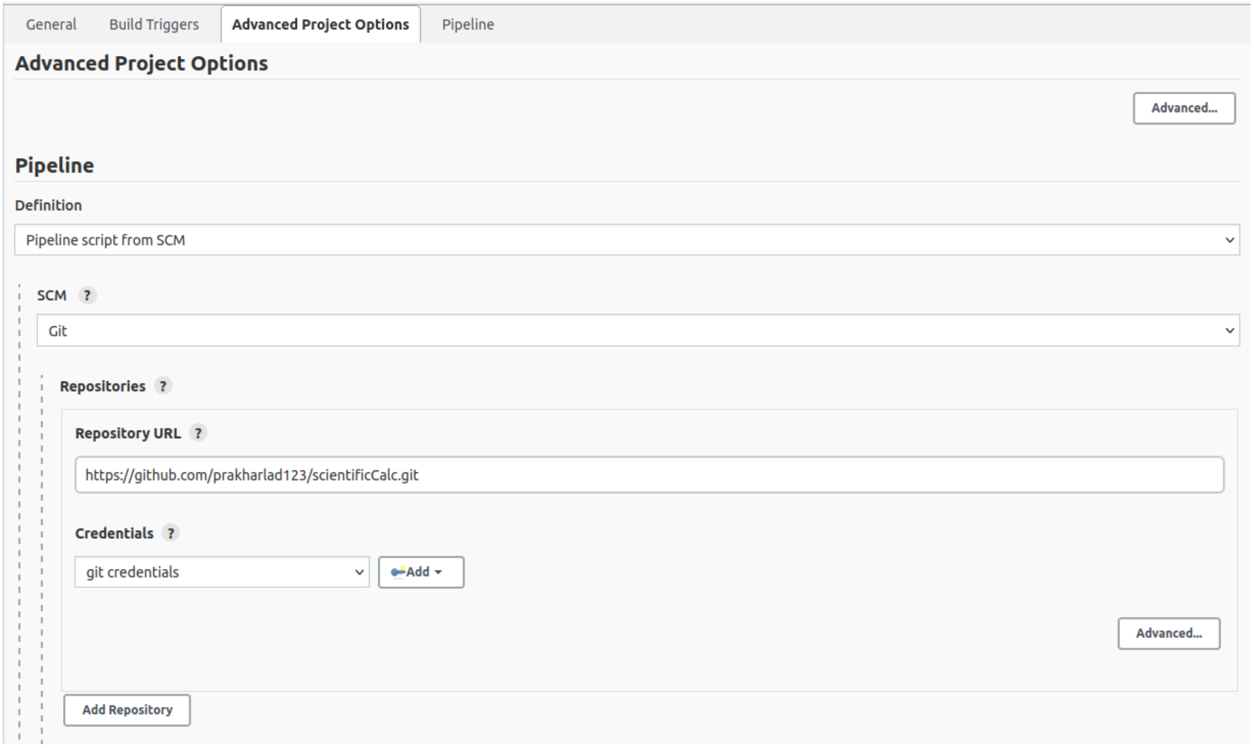
⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

Would last have run at Monday, 18 April, 2022 at 4:18:37 PM India Standard Time; would next run at Monday, 18 April, 2022 at 4:18:37 PM India Standard Time.

□ Ignore post-commit hooks ?

□ Disable this project ?
□ Quiet period ?
□ Trigger builds remotely (e.g., from scripts) ?

Fig 21 : Build trigger Setting



Advanced Project Options

Pipeline

Definition: Pipeline script from SCM

SCM ?: Git

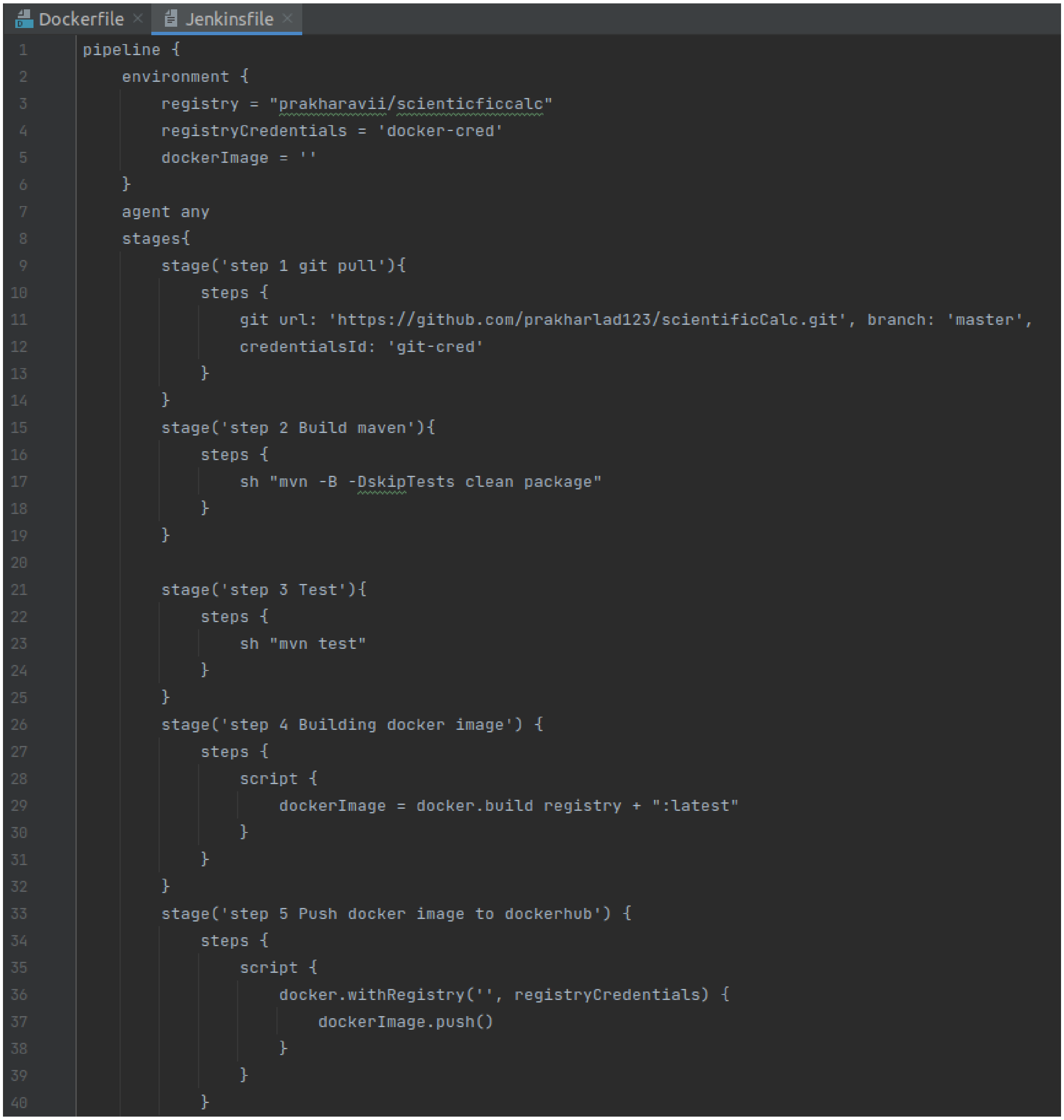
Repositories ?

Repository URL ?: https://github.com/prakharlad123/scientificCalc.git

Credentials ?: git credentials

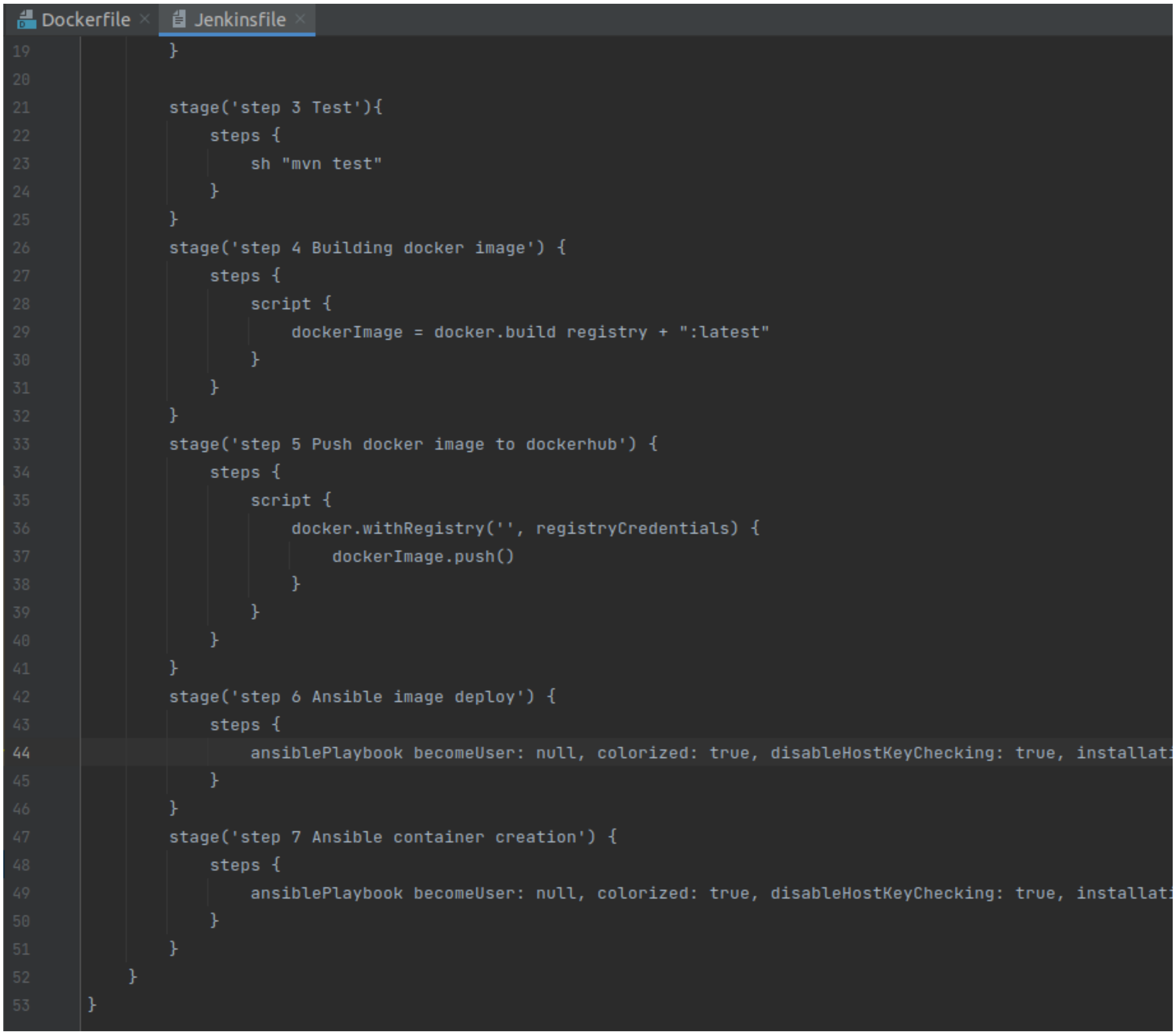
Fig 22 : Pipeline Setting

8. Jenkin File



```
pipeline {
    environment {
        registry = "prakharavii/scienticficcalc"
        registryCredentials = 'docker-cred'
        dockerImage = ''
    }
    agent any
    stages{
        stage('step 1 git pull'){
            steps {
                git url: 'https://github.com/prakharlad123/scientificCalc.git', branch: 'master',
                credentialsId: 'git-cred'
            }
        }
        stage('step 2 Build maven'){
            steps {
                sh "mvn -B -DskipTests clean package"
            }
        }

        stage('step 3 Test'){
            steps {
                sh "mvn test"
            }
        }
        stage('step 4 Building docker image') {
            steps {
                script {
                    dockerImage = docker.build registry + ":latest"
                }
            }
        }
        stage('step 5 Push docker image to dockerhub') {
            steps {
                script {
                    docker.withRegistry('', registryCredentials) {
                        dockerImage.push()
                    }
                }
            }
```

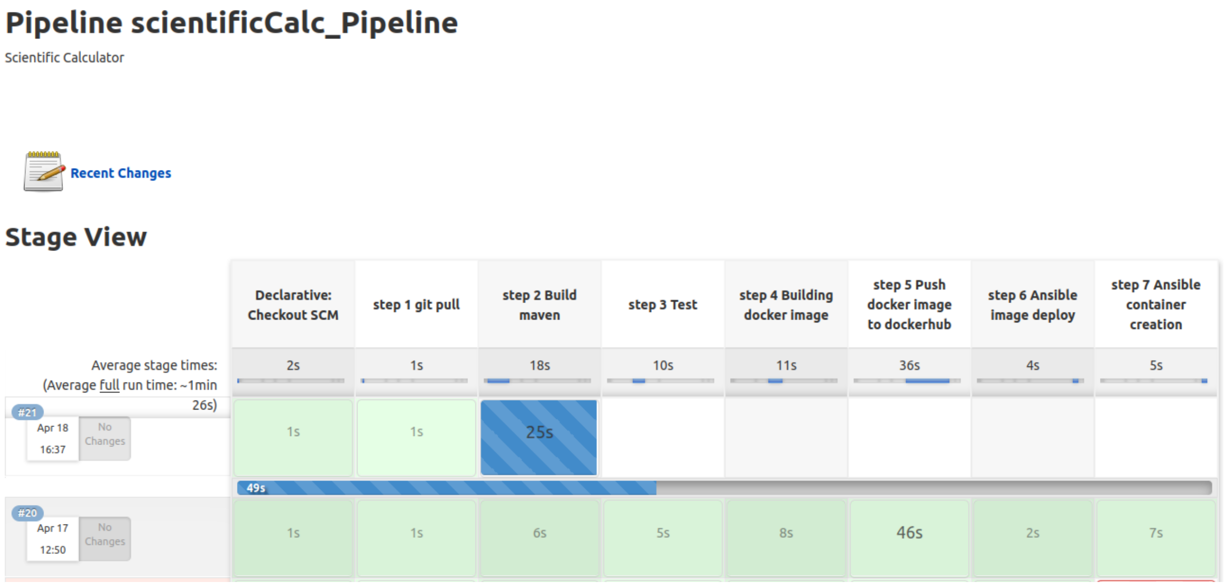
Fig 23 : Jenkinsfile



```
        }

        stage('step 3 Test'){
            steps {
                sh "mvn test"
            }
        }
        stage('step 4 Building docker image') {
            steps {
                script {
                    dockerImage = docker.build registry + ":latest"
                }
            }
        }
        stage('step 5 Push docker image to dockerhub') {
            steps {
                script {
                    docker.withRegistry('', registryCredentials) {
                        dockerImage.push()
                    }
                }
            }
        }
        stage('step 6 Ansible image deploy') {
            steps {
                ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true, installati
            }
        }
        stage('step 7 Ansible container creation') {
            steps {
                ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true, installati
            }
        }
    }
}
```

Fig 24 : Jenkinsfile (cont...)

9. To build the project create on **Build now**



Pipeline scientificCalc_Pipeline

Scientific Calculator

Recent Changes

Stage View

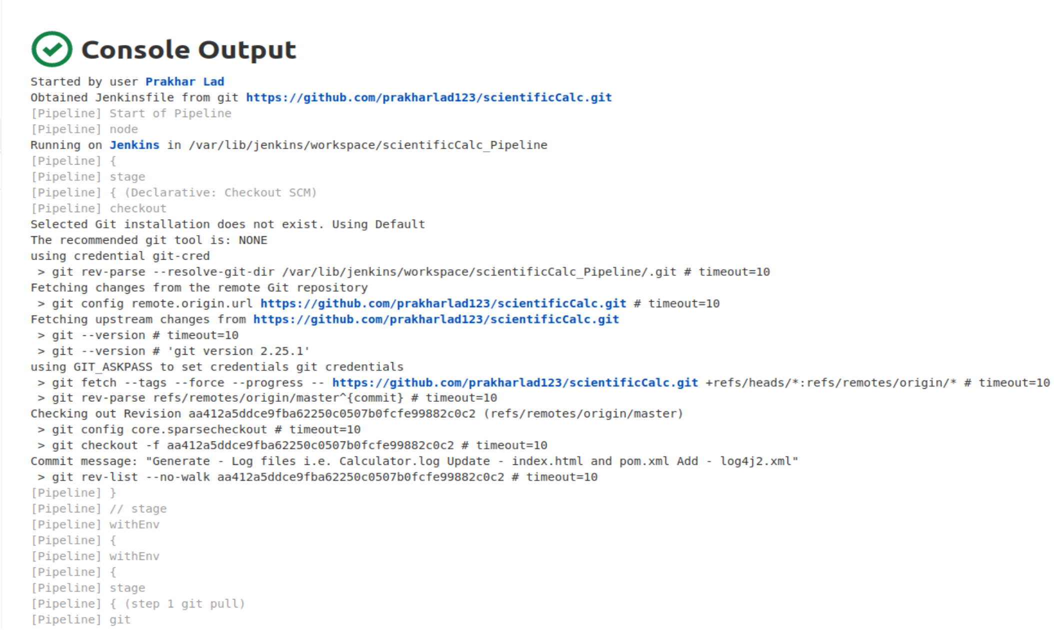
	Declarative: Checkout SCM	step 1 git pull	step 2 Build maven	step 3 Test	step 4 Building docker image	step 5 Push docker image to dockerhub	step 6 Ansible image deploy	step 7 Ansible container creation
Average stage times: (Average full run time: ~1min 26s)	2s	1s	18s	10s	11s	36s	4s	5s
#21 Apr 18 16:37 No Changes	1s	1s	25s					
49s								
#20 Apr 17 12:50 No Changes	1s	1s	6s	5s	8s	46s	2s	7s

Fig 25 : Build Project

Here we can see the overall status of our pipeline:

Blue dot: Pipeline succeeded, Red Dot: Pipeline failed, Black Dot: Pipeline aborted.

10. To view the detailed outline of pipeline: Click on build number then console output.



Console Output

```
Started by user Prakhar Lad
Obtained Jenkinsfile from git https://github.com/prakharlad123/scientificCalc.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/scientificCalc_Pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential git-cred
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/scientificCalc_Pipeline/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/prakharlad123/scientificCalc.git # timeout=10
Fetching upstream changes from https://github.com/prakharlad123/scientificCalc.git
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
using GIT_ASKPASS to set credentials git credentials
 > git fetch --tags --force --progress -- https://github.com/prakharlad123/scientificCalc.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision aa412a5ddce9fba62250c0507b0fcfe99882c0c2 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f aa412a5ddce9fba62250c0507b0fcfe99882c0c2 # timeout=10
Commit message: "Generate - Log files i.e. Calculator.log Update - index.html and pom.xml Add - log4j2.xml"
 > git rev-list --no-walk aa412a5ddce9fba62250c0507b0fcfe99882c0c2 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (step 1 git pull)
[Pipeline] git
```

Fig 26 : Console Output

4. Continuous Delivery

Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, *safely* and *quickly* in a *sustainable* way.

4.1. Docker:

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

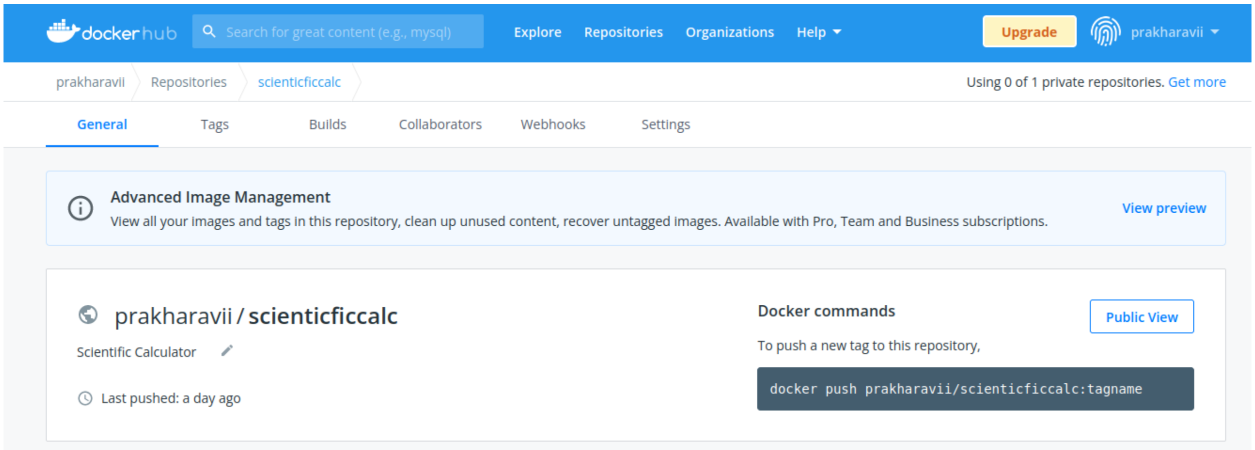
To install Docker, follow the given steps:

```
#To install docker
        user$ apt-get install docker.io
#Give permission to Jenkins to run docker commands
        user$ sudo usermod -aG docker jenkins
```

Here we are going to create a docker image of our application's generated jar file from Jenkins - Maven on top of **Open-jdk-11** as a base image and push the latest along with build number wise images to **DockerHub.** The steps to push the image are mentioned in the Jenkins file as a stage.

➔ The docker file tells the Image should be built/created using openjdk 11, after that we copy the created jar file and copy it to the working directory, and specify the command that should get triggered whenever a container is getting started.

- ➔ Pushed images can be found here : https://hub.docker.com/repository/docker/prakharavii/scienticficcalc



Using 0 of 1 private repositories. Get more

Advanced Image Management

View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions.

prakharavii / scienticficcalc

Scientific Calculator

Last pushed: a day ago

Docker commands

To push a new tag to this repository,

```
docker push prakharavii/scienticficcalc:tagname
```

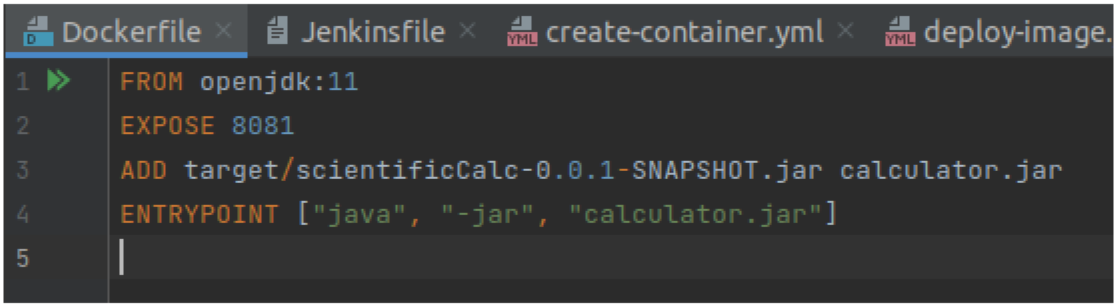
Fig 27 : Dockerhub

Create Dockerfile (same name) file to write docker image script in home directory of project.

Maven Build creates a jar SNAPSHOT with all required dependencies in the target folder that we need to create the image.

For jar file we need java as prerequisites. That's why we need to build an image from the openjdk:11 version because we use Java 11 to create this project.

Add the Entry point command for the image while it runs in the container.



```
FROM openjdk:11
EXPOSE 8081
ADD target/scientificCalc-0.0.1-SNAPSHOT.jar calculator.jar
ENTRYPOINT ["java", "-jar", "calculator.jar"]
```

Fig 28 : Dockerfile

5. Continuous Deployment

Continuous deployment is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users.

5.1. Ansible:

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows.

To install ansible run command:

1. Install python3

    ```
    sudo apt-get install python3-docker
    ```

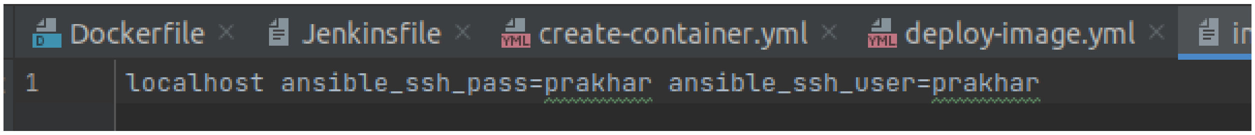
2. Install Open SSH

    ```
    sudo apt install openssh-server
    ssh-keygen –t rsa
    sudo apt update
    ```

3. Install Ansible

    ```
    sudo apt install ansible
    ```

5.2. Inventory File:



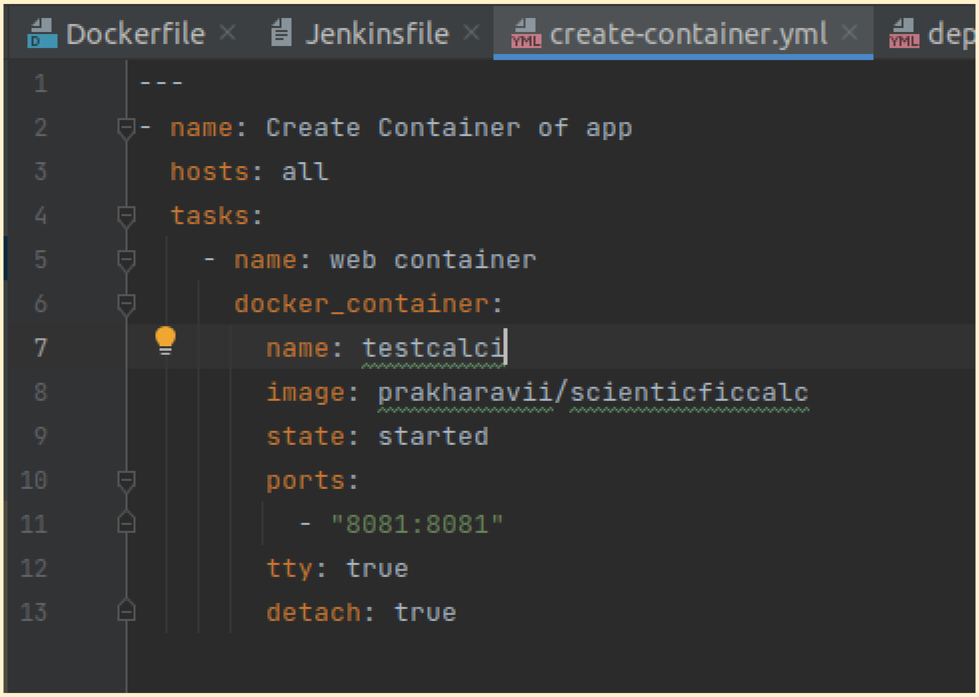
```
localhost ansible_ssh_pass=prakhar ansible_ssh_user=prakhar
```

Fig 29 : inventory

5.3. Playbook files:

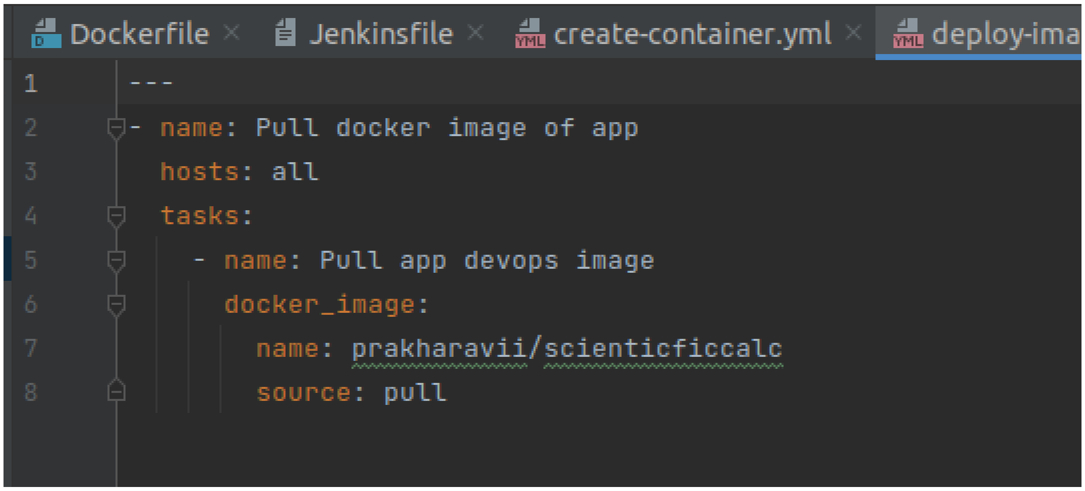
For our project we are going to specify following steps in the playbook:

1. Pulling the Scientific Calculator Image
2. Removing previous container.
3. Creating new Container using the pulled image
4. Remove unused and unnecessary Images



```
---
- name: Create Container of app
  hosts: all
  tasks:
    - name: web container
      docker_container:
        name: testcalci
        image: prakharavii/scienticficcalc
        state: started
        ports:
          - "8081:8081"
        tty: true
        detach: true
```

Fig 30 : create-container.yml



```
---
- name: Pull docker image of app
  hosts: all
  tasks:
    - name: Pull app devops image
      docker_image:
        name: prakharavii/scienticficcalc
        source: pull
```

Fig 31 : deploy-image.yml

6. Continuous Monitoring

Continuous monitoring refers to the process and technology required to incorporate monitoring across each phase of your DevOps and IT operations lifecycles. It helps to continuously ensure the health, performance, and reliability of your application and infrastructure as it moves from development to production.

6.1. ELK Stack:

The ELK Stack is a collection of three open-source products — **Elasticsearch, Logstash, and Kibana.** ELK stack provides centralized logging in order to identify problems with servers or applications. It allows you to search all the logs in a single place. It also helps to find issues in multiple servers by connecting logs during a specific time frame.

- E stands for ElasticSearch: used for storing logs
- L stands for LogStash : used for both shipping as well as processing and storing logs
- K stands for Kibana: is a visualization tool (a web interface) which is hosted through Nginx or Apache

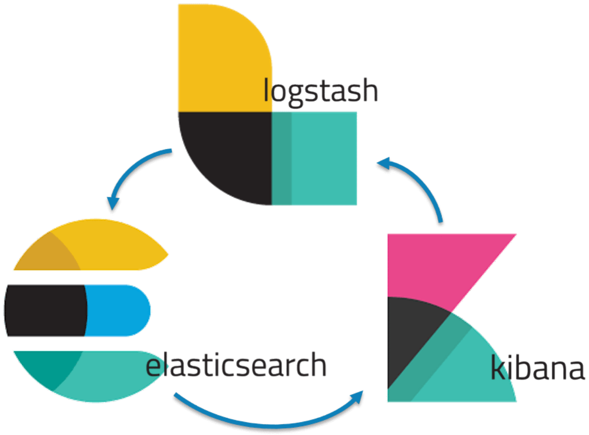
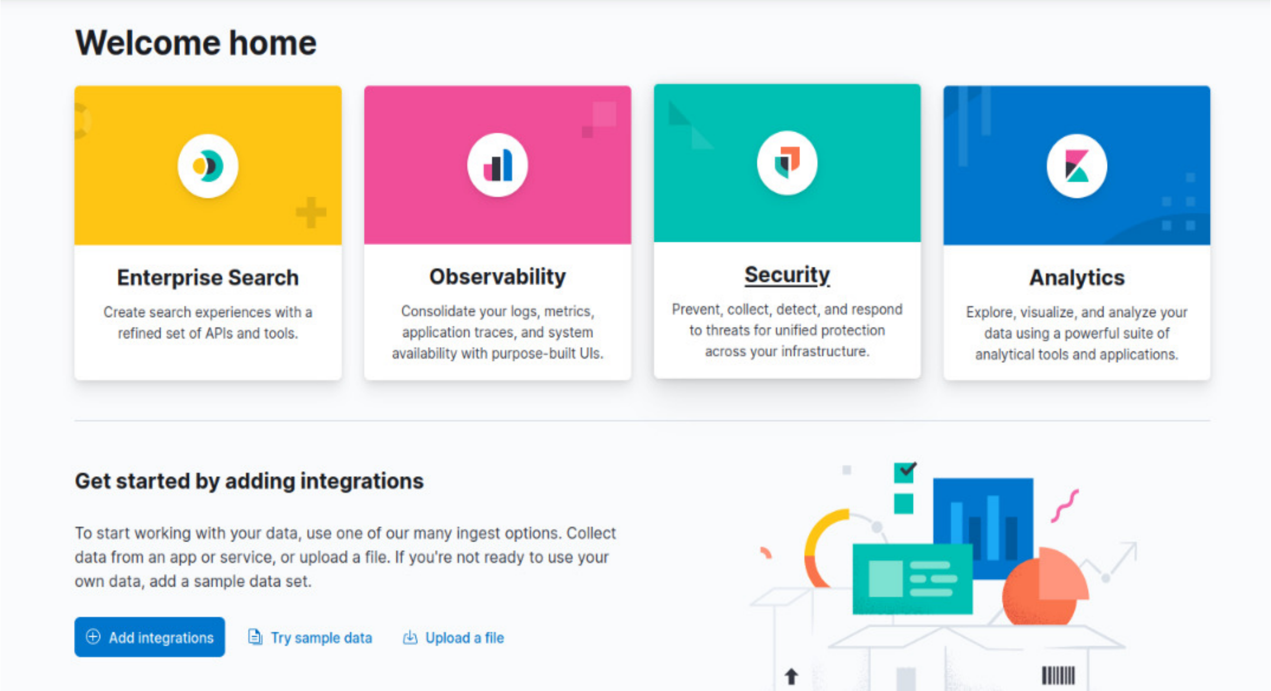


Fig 32 : ELK stack

6.2. Create an Elastic Account:

Follow the steps given below to create account

1. https://www.elastic.co/cloud/ Go to this link and enter your email id
2. After logging in click on create deployment
3. Name your deployment
4. Keep the default configuration or you could change it as needed
5. Click on create button at the bottom
6. The deployment will be created and shows username and password window.
7. Save the username password somewhere else as password will not be shown again



Welcome home

Enterprise Search

Create search experiences with a refined set of APIs and tools.

Observability

Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.

Security

Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.

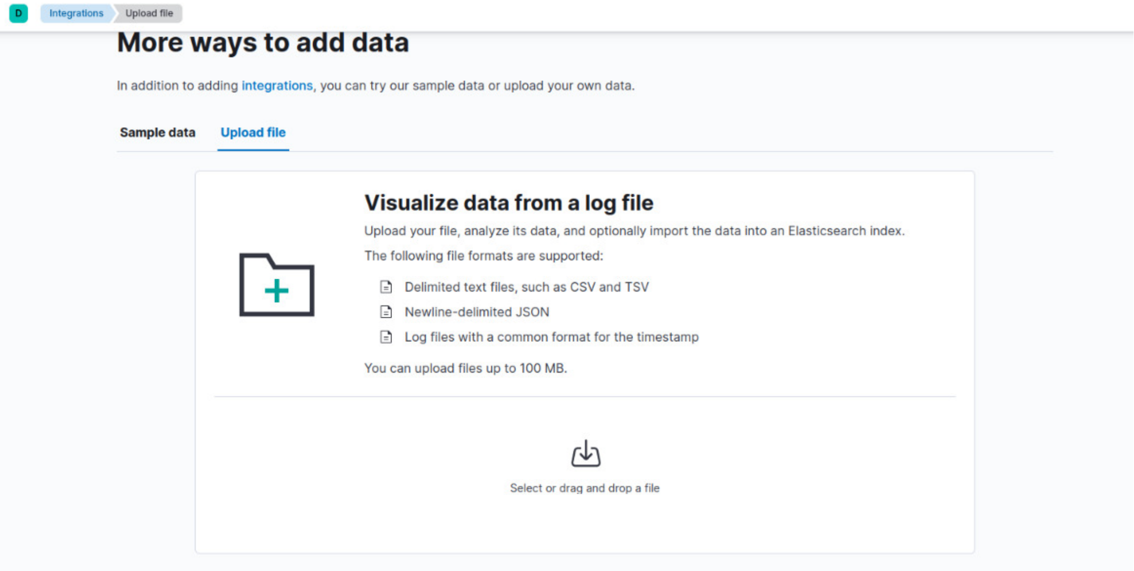
Analytics

Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.

Get started by adding integrations

To start working with your data, use one of our many ingest options. Collect data from an app or service, or upload a file. If you're not ready to use your own data, add a sample data set.

Fig 33 : Home page of Elastic



More ways to add data

In addition to adding integrations, you can try our sample data or upload your own data.

Visualize data from a log file

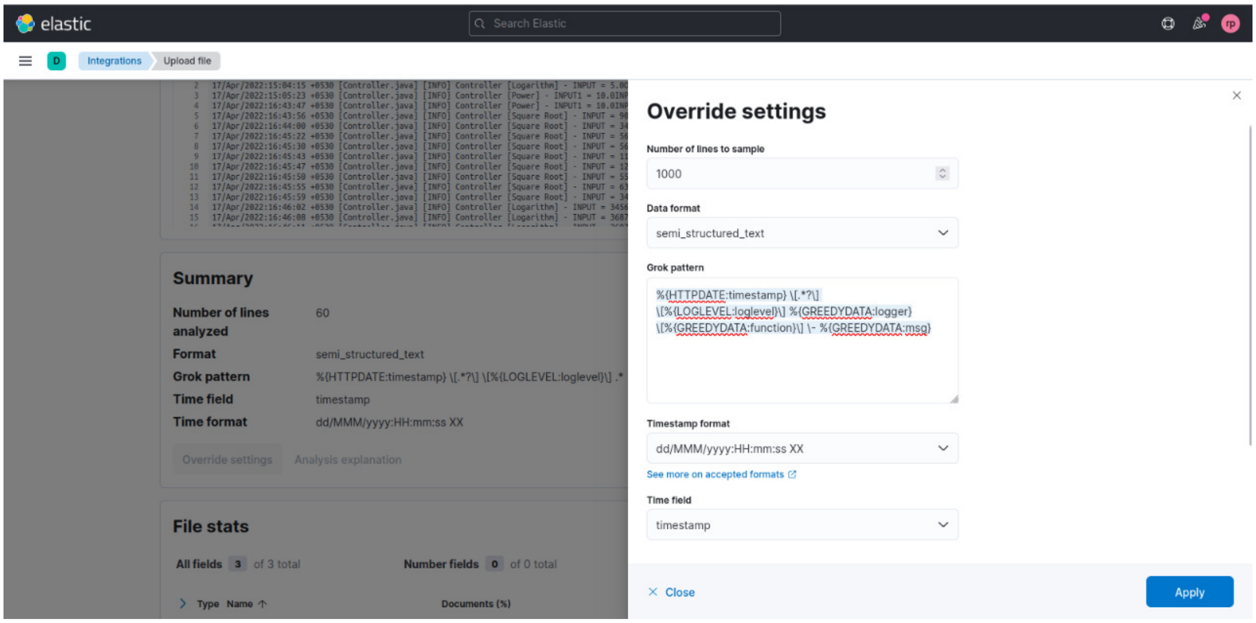
Upload your file, analyze its data, and optionally import the data into an Elasticsearch index.

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.

Fig 34 : Add log file



```
17/Apr/2022:15:04:15 +0530 [Controller.java] [INFO] Controller [Logarithm] - INPUT = 5.00
17/Apr/2022:15:05:23 +0530 [Controller.java] [INFO] Controller [Power] - INPUT1 = 10.0INP
17/Apr/2022:16:43:47 +0530 [Controller.java] [INFO] Controller [Power] - INPUT1 = 10.0INP
17/Apr/2022:16:43:56 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 90
17/Apr/2022:16:44:00 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 34
17/Apr/2022:16:45:22 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 56
17/Apr/2022:16:45:30 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 56
17/Apr/2022:16:45:43 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 11
17/Apr/2022:16:45:47 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 12
17/Apr/2022:16:45:50 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 55
17/Apr/2022:16:45:55 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 63
17/Apr/2022:16:45:59 +0530 [Controller.java] [INFO] Controller [Square Root] - INPUT = 34
17/Apr/2022:16:46:02 +0530 [Controller.java] [INFO] Controller [Logarithm] - INPUT = 3456
17/Apr/2022:16:46:08 +0530 [Controller.java] [INFO] Controller [Logarithm] - INPUT = 3687
```

Summary

Number of lines analyzed: 60
Format: semi_structured_text
Grok pattern: %{HTTPDATE:timestamp} \[.*?\] \[%{LOGLEVEL:loglevel}\] .*
Time field: timestamp
Time format: dd/MMM/yyyy:HH:mm:ss XX

File stats

All fields 3 of 3 total   Number fields 0 of 0 total

Override settings

Number of lines to sample: 1000

Data format: semi_structured_text

Grok pattern:

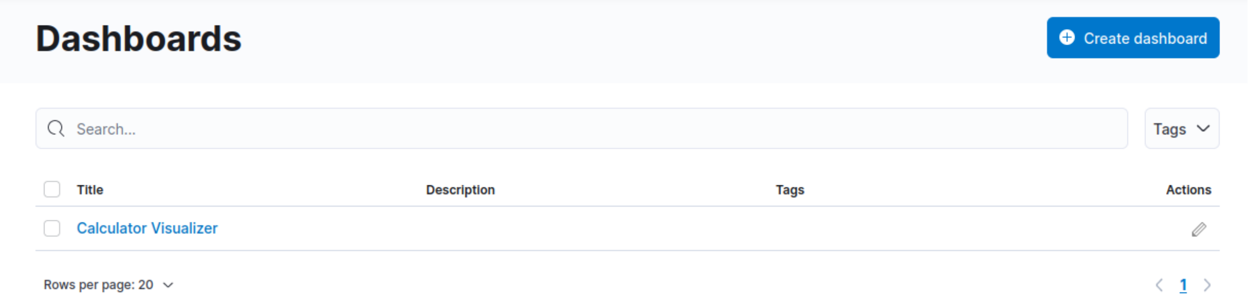
```
%{HTTPDATE:timestamp} \[.*?\]
\[%{LOGLEVEL:loglevel}\] %{GREEDYDATA:logger}
\[%{GREEDYDATA:function}\] \- %{GREEDYDATA:msg}
```

Timestamp format: dd/MMM/yyyy:HH:mm:ss XX

See more on accepted formats

Time field: timestamp

Fig 35 : Grok code



Dashboards

⊕ Create dashboard

Search...

Tags ⌄

□	Title	Description	Tags	Actions
□	Calculator Visualizer			✎

Rows per page: 20 ⌄

‹ 1 ›

Fig 35 : Dashboard

- Click on Create visualization → Select Index file
- Now to create visualisation , drag and drop available fields that are identified from Grok pattern to working area and create different visualise charts.
- Add multiple visualise charts to the Kibana dashboard.

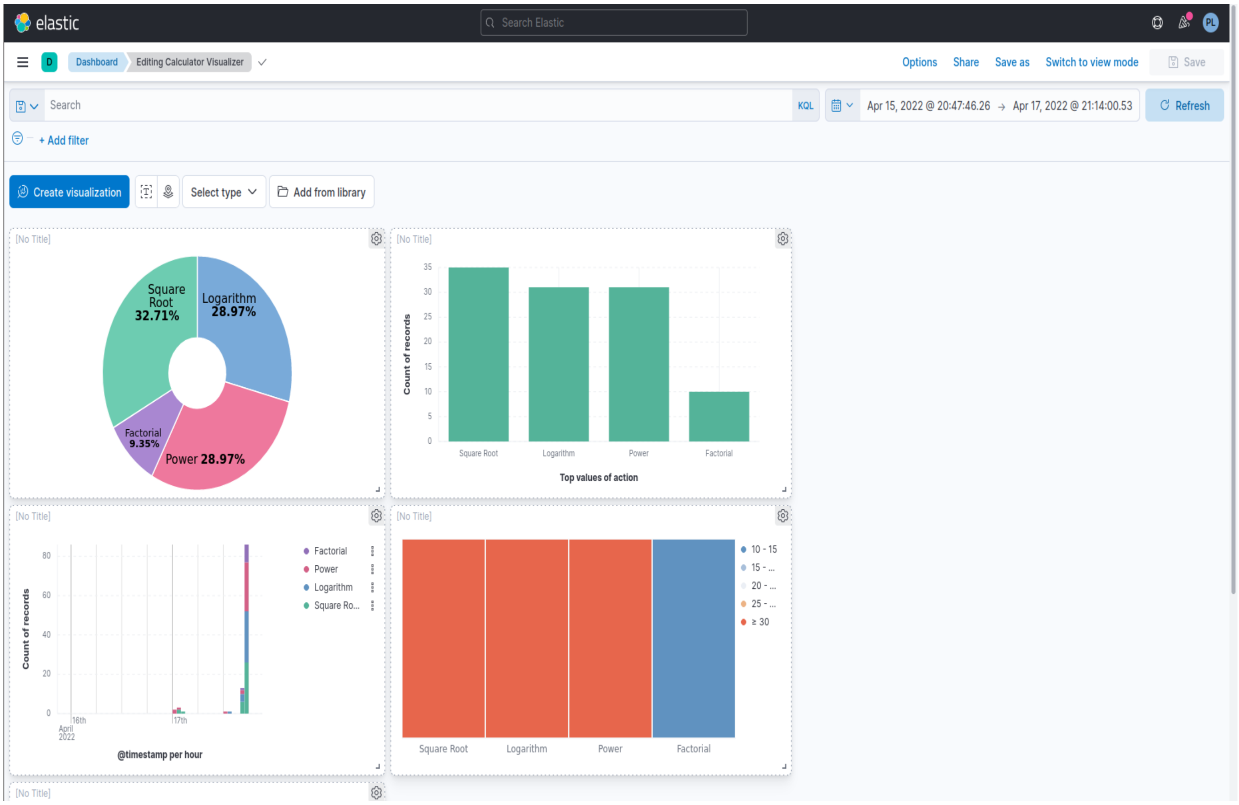


Fig 36 : Visualization

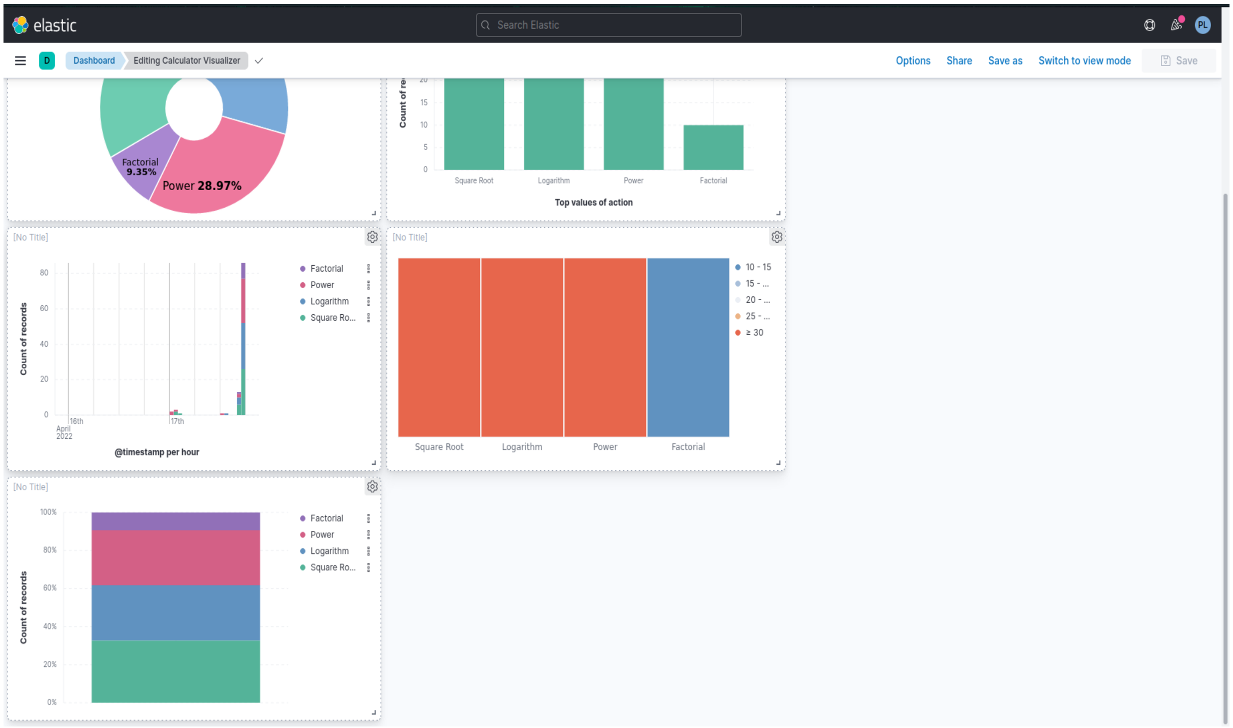


Fig 37 : Visualization

7. Conclusion

In this project, I automated the entire SDLC using DevOps tools and approaches. With the help of which the task of development team and operations team becomes easy as the DevOps pipeline gives the comfort of making code changes easily and also reduces the chances of post production release errors. The toolchain allows software companies to quickly integrate, build, test and deploy newer versions of their products to the production hosts.. These kinds of tools and approaches are very helpful in companies where the need for daily deployment is very high.

8. Bibliography

Spring Framework : https://spring.io/projects/spring-framework

Jenkins : https://www.jenkins.io/

Docker : https://www.docker.com/

Ansible : https://www.ansible.com/

ELK Stack : https://www.elastic.co/cloud/