SPE mini Project Report

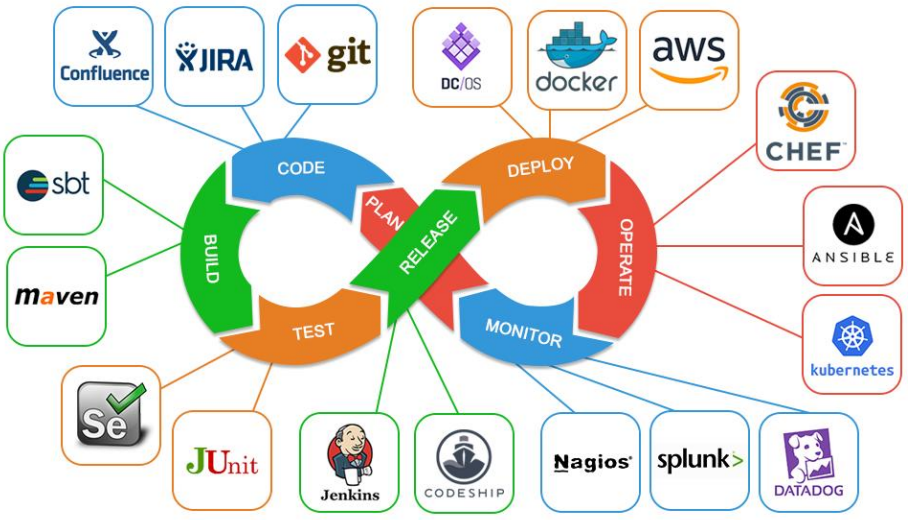
Aim: To build the scientific calculator using Devops pipeline.

Created by: Vaibhavi Tikone [MT2020006]

Github: https://github.com/VaibhaviTikone/Calculator_SPE

DockerHub: https://hub.docker.com/repository/docker/vaibhavi1998/calculator-spe

Devops:



DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). Our main goal is to automate the continuous integration and continuous deployment also known as CI/CD pipeline. Devops tools help to automate this process. The various steps or phases of this pipeline is listed below:

1. Continuous development
2. Continuous integration
3. Continuous testing
4. Continuous deployment
5. Continuous monitoring

In this project, our focus is to build calculator with 4 functionality and get familiar with whole devops tools chain. This tools chain is listed below:

1. Code – IntelliJ by JetBrains
2. Test – Junit
3. SCM – Git, GitHub (source control management tool)
4. Build – Maven
5. Integration – Jenkins
6. Containerization – Docker
7. Deployment – Ansible
8. Monitor – Logstash, Kibana, Elasticsearch

Find the references and links to the GitHub repository and docker repository at the end of the report.

Scientific Calculator:

We want to create a scientific calculator program with user menu driven operations as listed below:

- Square root function - √x
- Factorial function - x!
- Natural logarithm (base e) - ln(x)
- Power function - x^b

I have used Java to code scientific calculator. IntelliJ tool by JetBrains gives various facilities to the user to build their project. Square root function can find the square root of any positive integer. For negative integer I have not defined the function, as for negative integer square root value will be complex. For natural logarithm I have defined a function which can find the logarithm of any positive integer.

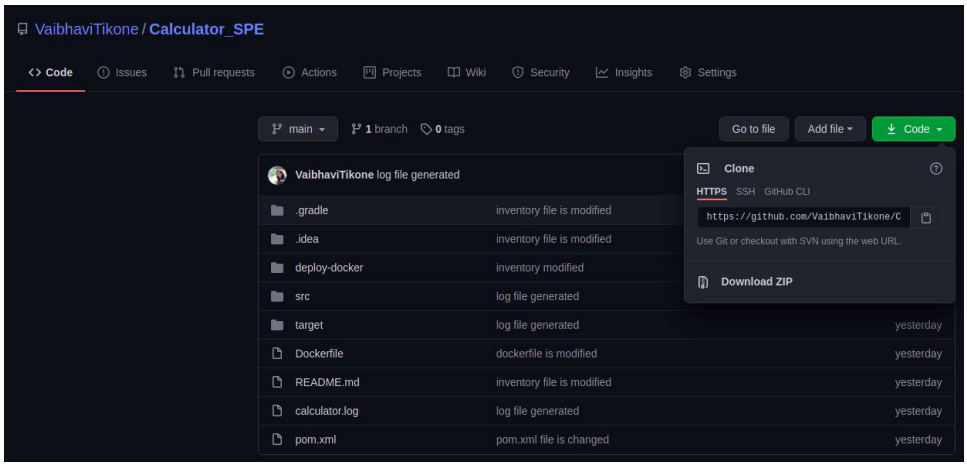
```java
    public double Sqrt(double sqrt)
    {
        logging.info("[INFO]: value: " + sqrt);
        if(sqrt<0)
        {
            logging.error("[ERROR]: square root is not defined for negative numbers");
            return Double.NaN;
        }
        return Math.sqrt(sqrt);
    }
    public double Factorial(double n)
    {
        logging.info("[INFO]: I/P for factorial: " + n);
        if(n<0) return Double.NaN;
        double factorial=1;
        for(int i = 2; i <= n; ++i)
             factorial*=i;
        logging.info("[INFO]: Factorial result: " + factorial);
        return factorial;
    }
    public double Log(double logVal)
    {
        logging.info("[INFO]: Value: "+logVal);
        return Math.log(logVal);
    }
    public double Power(double base, double exp)
    {
        logging.info("[INFO]: Base: "+base +" Exponent: "+exp);
        if(base==0 && exp ==0)
            return Double.NaN;
        return Math.pow(base,exp);
    }
```

```
-----------------------------------------------------------------------------------------------
                                Welcome to scietific calculator

1. Square root
2. Factorial
3. Natural log
4. Power
5. Exit

Enter your choice: 2
Enter the value: 36
Factorial of 36.0 = 3.719933267899012E41
=======================================================

1. Square root
2. Factorial
3. Natural log
4. Power
5. Exit

Enter your choice: 5


                                    Thanks for visiting
-----------------------------------------------------------------------------------------------
```

Source Control Management:

Git is a widely used version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily **used** for source code management in software development, but it can be **used** to keep track of changes in any set of files. As a single developer working on our own project, we can keep track of changes made in our source code using git terminal tool.

We can either start working with git by initializing directory as git, or we can make one repository on GitHub and clone it, and then we can push our source code to remote repository. I have used second approach. I have created one public repository on GitHub, and I cloned it.



I wrote source code, and then I pushed it to the remote repository. Set of commands which I followed are:

cd <working_directoy>

git clone <link_to_remote_repositoy>

git add .

git commit –m “message”

git push (will ask for your GitHub credential)

git status (check if changes made to the code is pushed or not)

```
vaibhavi@G3:~/Documents/SPE/SPE_MINI/Calculator_SPE$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/main/java/Calculator.java

no changes added to commit (use "git add" and/or "git commit -a")
vaibhavi@G3:~/Documents/SPE/SPE_MINI/Calculator_SPE$ git add .
vaibhavi@G3:~/Documents/SPE/SPE_MINI/Calculator_SPE$ git commit -m "source code modified"
[main 8e1f3bb] source code modified
 1 file changed, 1 insertion(+), 8 deletions(-)
vaibhavi@G3:~/Documents/SPE/SPE_MINI/Calculator_SPE$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
vaibhavi@G3:~/Documents/SPE/SPE_MINI/Calculator_SPE$ git push
Username for 'https://github.com': vaibhavitikone
Password for 'https://vaibhavitikone@github.com':
Counting objects: 6, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 528 bytes | 528.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/VaibhaviTikone/Calculator_SPE.git
   95ba66c..8e1f3bb  main -> main
vaibhavi@G3:~/Documents/SPE/SPE_MINI/Calculator_SPE$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
vaibhavi@G3:~/Documents/SPE/SPE_MINI/Calculator_SPE$
```

Test

The **testing** is **important** since it discovers defects/bugs before the delivery to the client, which guarantees the quality of the **software**. It makes the **software** more reliable and easier to use. Software unit tests help the developer to verify that the logic of a piece of the program is correct. For Java there Junit as testing tool.

JUnit is a simple, open source framework to write and run repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks. JUnit features include:

- Assertions for testin expectd results
- Test fixtures for sharing common test data
- Test runners for running test

```java
@Test
public  void  factorialPositive()
{
    assertEquals( message: "[Positive Case]: Factorial of a number, ", expected: 120,calc.Factorial( n: 5),DELTA);
    assertEquals( message: "[Positive Case]: Factorial of a number, ", expected: 5040,calc.Factorial( n: 7),DELTA);
    assertEquals( message: "[Positive Case]: Factorial of a number, ",Double.NaN,calc.Factorial( n: -2),DELTA);
}
@Test
public void factorialNegative()
{
    assertNotEquals( message: "[Negative Case]: Factorial of a number,", unexpected: 24,calc.Factorial( n: 5),DELTA);
    assertNotEquals( message: "[Negative Case]: Factorial of a number,", unexpected: 120,calc.Factorial( n: 7),DELTA);
    assertNotEquals( message: "[Negative Case]: Factorial of a number,", unexpected: 5,calc.Factorial( n: -3),DELTA);
}
@Test
public  void  exponentiationPositive()
{
    assertEquals( message: "[Positive Case]: Power of a number, ", expected: 32,calc.Power( base: 2, exp: 5),DELTA);
    assertEquals( message: "[Positive Case]: Power of a number, ", expected: 0.1111111111,calc.Power( base: 3, exp: -2),DELTA);
    assertEquals( message: "[Positive Case]: Power of a number, ",Double.NaN,calc.Power( base: 0, exp: 0),DELTA);
}
@Test
public  void  exponentiationNegative()
{
    assertNotEquals( message: "[Negative Case]: Power of a Number, ", unexpected: 31,calc.Power( base: 2, exp: 5),DELTA) ;
    assertNotEquals( message: "[Negative Case]: Power of a Number, ", unexpected: 0.1111111122,calc.Power( base: 3, exp: -2),DELTA);
    assertNotEquals( message: "[Negative Case]: Power of a Number, ", unexpected: 1,calc.Power( base: 0, exp: 0),DELTA);
}
```

```java
@Test
public  void  logPositive()
{
    assertEquals( message: "[Positive Case]: Log of a number, ", expected: 2.302585092994,calc.Log( logVal: 10),DELTA);
    assertEquals( message: "[Positive Case]: Log of a number, ",Double.NEGATIVE_INFINITY,calc.Log( logVal: 0),DELTA);
    assertEquals( message: "[Positive Case]: Log of a number, ",Double.NaN,calc.Log( logVal: -5),DELTA);
}
@Test
public  void  logNegative()
{
    assertNotEquals( message: "[Negative Case]: Log of a number, ",  unexpected: 6, calc.Log( logVal: 3), DELTA);
    assertNotEquals( message: "[Negative Case]: Log of a number, ",  unexpected: -7.3, calc.Log( logVal: 1.6), DELTA);
    assertNotEquals( message: "[Negative Case]: Log of a number, ",  unexpected: 0, calc.Log( logVal: 0), DELTA);
    assertNotEquals( message: "[Negative Case]: Log of a number, ",  unexpected: 9.0, calc.Log( logVal: -5), DELTA);
}
@Test
public void sqrtPositive(){
    assertEquals( message: "[Positive Case]: SQRT of a number, ",  expected: 2, calc.Sqrt(4), DELTA);
    assertEquals( message: "[Positive Case]: SQRT of a number, ",  expected: 2.236067977499, calc.Sqrt(5), DELTA);
    assertEquals( message: "[Positive Case]: SQRT of a number, ",  expected: 0, calc.Sqrt(0), DELTA);
    assertEquals( message: "[Positive Case]: SQRT of a number, ", Double.NaN, calc.Sqrt(-2), DELTA);
}
@Test
public void sqrtNegative(){
    assertNotEquals( message: "[Negative Case]: SQRT of a number, ",  unexpected: 6, calc.Sqrt(4), DELTA);
    assertNotEquals( message: "[Negative Case]: SQRT of a number, ",  unexpected: 4.2, calc.Sqrt(2.1), DELTA);
    assertNotEquals( message: "[Negative Case]: SQRT of a number, ",  unexpected: 7.3, calc.Sqrt(-5), DELTA);
}
```

Build

Build means to compile the source code. When we work on projects, we need some extra packages which serves facilities of reusing the already written code. We can use these packages in our projects and can save our time. In other words, our source code has some dependencies

to run. To handle such dependencies with version, we need to use Devops tool. Maven tool is widely used for this purpose.

Maven is project build management code which manages the JAR files for additional dependencies. Maven downloads the JAR files and make them available during build of project. Maven has one configuration file for serving building facility to us. A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

POM file for my project is as shown below:

```xml
<?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>trail2</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.1</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.14.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.14.0</version>
        </dependency>
    </dependencies>
```

POM structure:

Project metadata	Project name, version etc. Output file type: JAR, WAR
Dependencies	List of projects we dependent on Spring, Hibernate, etc
Plugins	Additional custom tasks to run: generate JUnit test reports, etc

Project Coordinates

Name	Description
GroupID	Name of company, group or organization Conversation is to use reverse domain name: org.example
ArtifactID	Name of the project: trail2
Version	A specific release version like 1.0, 1.6, 2.5 If project is under active development then: 1.0-SNAPSHOT

When building my project, JAR file wasn't building the project on its own. So, I attached following script to POM.xml file.

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

Maven commands from installing maven to building the project is as follows:

sudo apt update

sudo apt install maven

mvn --version //to check maven version

mvn clean

mvn install //to build the project

mvn test //to test the source code

```
vaibhavi@G3:~/Documents/SPE/SPE_MINI/Calculator_SPE$ mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< org.example:trail2 >-------------------------
[INFO] Building trail2 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ trail2 ---
[INFO] Deleting /home/vaibhavi/Documents/SPE/SPE_MINI/Calculator_SPE/target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.195 s
[INFO] Finished at: 2021-03-15T18:48:53+05:30
[INFO] ------------------------------------------------------------------------
```

```
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ trail2 ---
[INFO] Installing /home/vaibhavi/Documents/SPE/SPE_MINI/Calculator_SPE/target/trail2-1.0-SNAPSHOT.jar to /home/vaibhavi/.m2/
repository/org/example/trail2/1.0-SNAPSHOT/trail2-1.0-SNAPSHOT.jar
[INFO] Installing /home/vaibhavi/Documents/SPE/SPE_MINI/Calculator_SPE/pom.xml to /home/vaibhavi/.m2/repository/org/example/
trail2/1.0-SNAPSHOT/trail2-1.0-SNAPSHOT.pom
[INFO] Installing /home/vaibhavi/Documents/SPE/SPE_MINI/Calculator_SPE/target/trail2-1.0-SNAPSHOT-jar-with-dependencies.jar
to /home/vaibhavi/.m2/repository/org/example/trail2/1.0-SNAPSHOT/trail2-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.657 s
[INFO] Finished at: 2021-03-15T18:54:33+05:30
[INFO] ------------------------------------------------------------------------
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running CalculatorTest
15/Mar/2021:19:24:17 920 [Calculator.java] [INFO] Calculator [INFO]: Value: 10.0
15/Mar/2021:19:24:17 925 [Calculator.java] [INFO] Calculator [INFO]: Value: 0.0
15/Mar/2021:19:24:17 925 [Calculator.java] [INFO] Calculator [INFO]: Value: -5.0
15/Mar/2021:19:24:17 927 [Calculator.java] [INFO] Calculator [INFO]: Value: 3.0
15/Mar/2021:19:24:17 928 [Calculator.java] [INFO] Calculator [INFO]: Value: 1.6
15/Mar/2021:19:24:17 928 [Calculator.java] [INFO] Calculator [INFO]: Value: 0.0
15/Mar/2021:19:24:17 929 [Calculator.java] [INFO] Calculator [INFO]: Value: -5.0
15/Mar/2021:19:24:17 930 [Calculator.java] [INFO] Calculator [INFO]: Base: 2.0 Exponent: 5.0
15/Mar/2021:19:24:17 931 [Calculator.java] [INFO] Calculator [INFO]: Base: 3.0 Exponent: -2.0
15/Mar/2021:19:24:17 932 [Calculator.java] [INFO] Calculator [INFO]: Base: 0.0 Exponent: 0.0
15/Mar/2021:19:24:17 933 [Calculator.java] [INFO] Calculator [INFO]: Base: 2.0 Exponent: 5.0
15/Mar/2021:19:24:17 934 [Calculator.java] [INFO] Calculator [INFO]: Base: 3.0 Exponent: -2.0
15/Mar/2021:19:24:17 935 [Calculator.java] [INFO] Calculator [INFO]: Base: 0.0 Exponent: 0.0
15/Mar/2021:19:24:17 936 [Calculator.java] [INFO] Calculator [INFO]: I/P for factorial: 5.0
15/Mar/2021:19:24:17 937 [Calculator.java] [INFO] Calculator [INFO]: Factorial result: 120.0
15/Mar/2021:19:24:17 937 [Calculator.java] [INFO] Calculator [INFO]: I/P for factorial: 7.0
15/Mar/2021:19:24:17 938 [Calculator.java] [INFO] Calculator [INFO]: Factorial result: 5040.0
15/Mar/2021:19:24:17 938 [Calculator.java] [INFO] Calculator [INFO]: I/P for factorial: -2.0
15/Mar/2021:19:24:17 940 [Calculator.java] [INFO] Calculator [INFO]: I/P for factorial: 5.0
15/Mar/2021:19:24:17 940 [Calculator.java] [INFO] Calculator [INFO]: Factorial result: 120.0
15/Mar/2021:19:24:17 941 [Calculator.java] [INFO] Calculator [INFO]: I/P for factorial: 7.0
15/Mar/2021:19:24:17 941 [Calculator.java] [INFO] Calculator [INFO]: Factorial result: 5040.0
15/Mar/2021:19:24:17 942 [Calculator.java] [INFO] Calculator [INFO]: I/P for factorial: -3.0
15/Mar/2021:19:24:17 943 [Calculator.java] [INFO] Calculator [INFO]: value: 4.0
15/Mar/2021:19:24:17 943 [Calculator.java] [INFO] Calculator [INFO]: value: 5.0
```

Docker

When we are working on project, some dependencies are OS dependent. When more than one developer is working on project, they may be developing code on different OS with different package versions. So, to manage this configuration and allow the developer to deploy project on any OS they want, and to check the compatibility of project deployments on various OS and packages, we can do kernel level virtualization. Docker is a widely used container which provides lightweight virtualization. We can create one OS image, and we can install only those packages which are needed for specific project.

So, the plan is to create a docker image of jar file created after build process and push the latest image on to docker hub. The pushed procedure would be done after every build which would include removing preciously pushed docker image and replace with latest built one.

We can add docker in user list so that we can run it as user. We can get simple docker commands from its official website. We must install docker in our system and then to add it as user we can follow below commands:

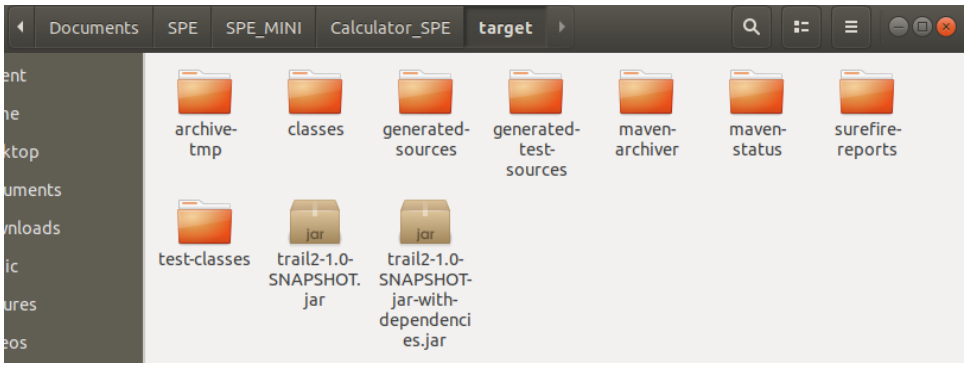
sudo usermod –aG docker <username>

sudo systemctl start docker

Docker run -it <image_name> //to run the docker as interactive terminal mode

We have various approaches to deploy our project on separate OS image. We can follow multipass approach using Virtual machine, we can have separate host in virtual box, or we can create one local user and use that host as another host. I made one local user.

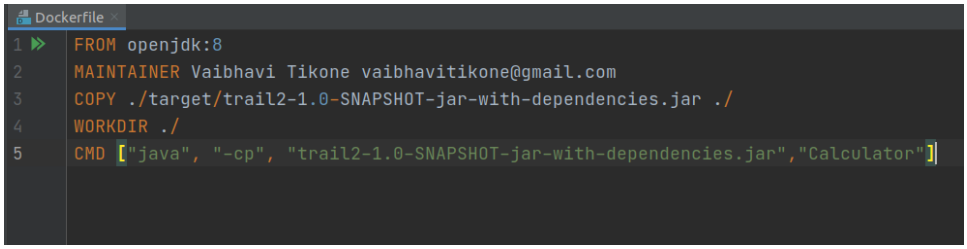
We can tell Jenkins to create image and push it to the DockerHub. When we build our project using maven, one target folder is generated which consists of some other directories.



To run this file, we must run the command:

java –cp pathto<jar_file> <main_class>

I have added one dockerfile to my project so that Jenkins can get the JAR file and other information from that file.



Pipeline

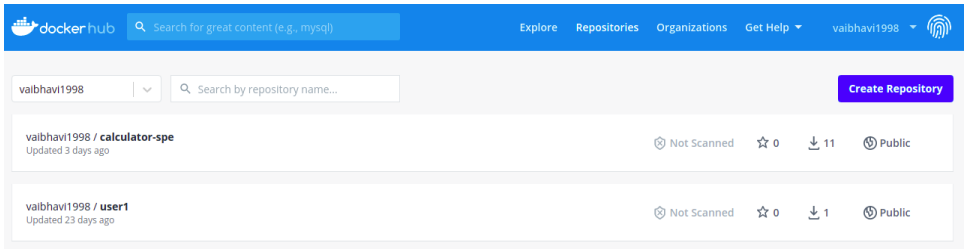
Definition

Pipeline script

Script

```
    agent any

        stages {
            stage('Step 1 Git clone') {
                steps {
                    git branch: 'main', url: 'https://github.com/VaibhaviTikone/Calculator_SPE.git'
                }
            }
            stage('Step 2 Maven') {
                steps {
                sh 'mvn clean package'
                }
            }
            stage('Step 3 Docker build to Image') {
                steps {
                    script{
                            imageName = docker.build "vaibhavi1998/calculator-spe:latest"
                    }
                }
```

After building the Jenkins pipeline, docker image will get pushed to DockerHub.



Ansible:

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. There are two types of hosts in ansible: controller host, and manager host. Controller host is which manages all other hosts where we want to deploy our project. And manager hosts are those on which we are going to deploy our project.

Ansible connects to the hosts it manages using openssh or winrm and run tasks. In the ansible inventory file we list out manager hosts or group of manager hosts. We can list out all the tasks in the ansible playbook file. One can get the additional information from the official website.

We need to install openssh-server with ansible. So whole process of installation will be done by following below commands:

sudo apt install openssh-server

sudo apt update

sudo apt install ansible

ansible --version //to check the ansible version

We must add manager host to inventory file of ansible as given below. I have created a local user named “tom” on local machine. And then I have added that host to /etc/ansible/hosts file.

```
vaibhavi@G3:~$ cat /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.
localhost ansible_user=tom
```

Dockerfile | deploy-image.yml

```
---
- name: Pull docker image of calculator mini project
  hosts: all
  tasks:
    - name: Pull calculator devops image
      docker_image:
        name: vaibhavi1998/calculator-spe
        source: pull
```

deploy-image.yml | inventory

```
localhost ansible_user=tom ansible_python_interpreter=/usr/bin/python3
```

To connect the local user and controller node I followed below commands:

On manager host terminal:

ssh-keygen –t rsa

On controller node terminal:

ssh-keygen –t rsa

ssh-copy-id tom@localhost

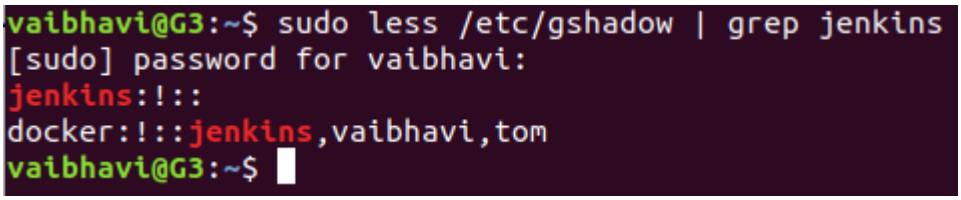
Integration

Continuous Integration (**CI**) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. Here I have used Jenkins as an integration tool.

Jenkins is a free and open-source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. We need to add Jenkins to docker group, so that Jenkins can use docker for build docker image. We can follow below commands:

sudo usermod –aG docker Jenkins

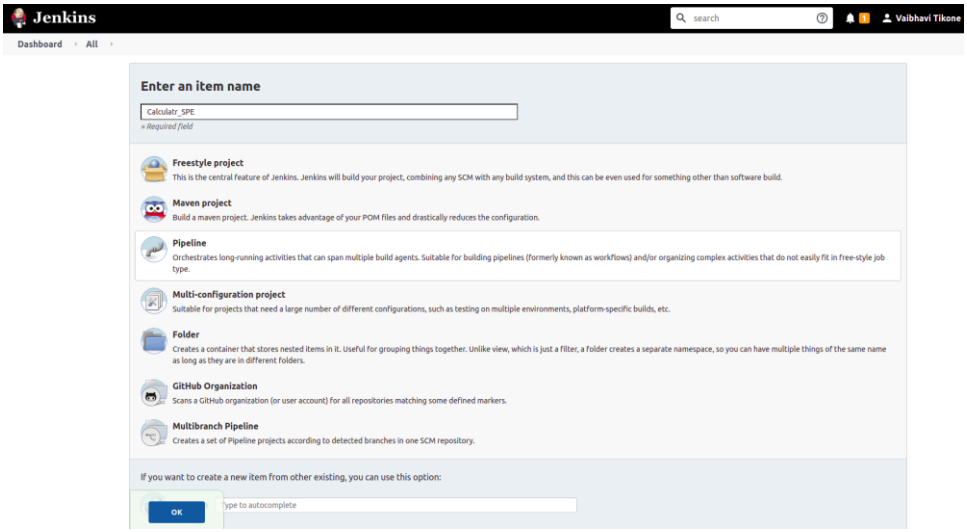
sudo less /etc/gshadow | grep jenkins // to verify



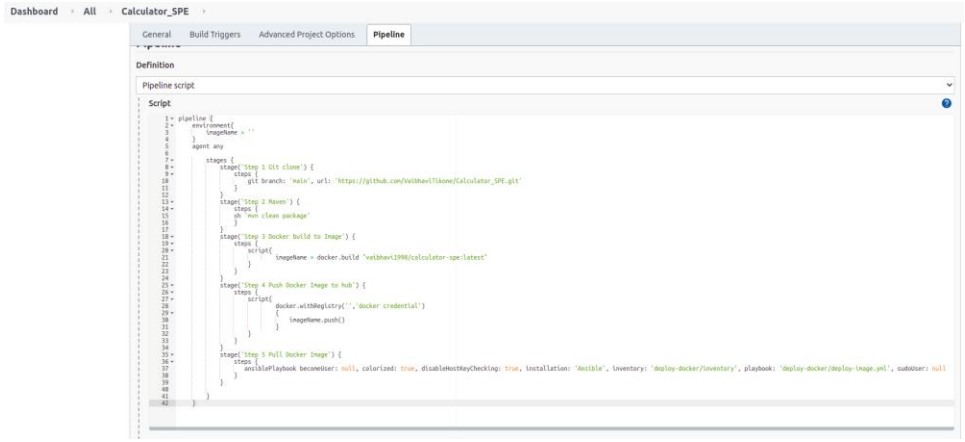
sudo systemctl start jenkins //jenkins starts at port number 8080

login on to http://localhost:8080 onto your browser.

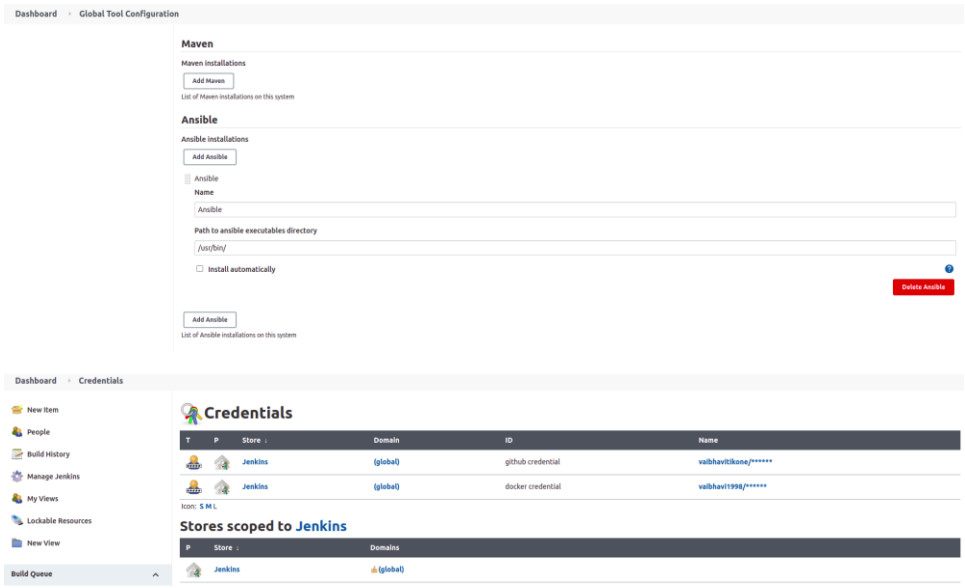
We can create one pipeline in Jenkins for our project.



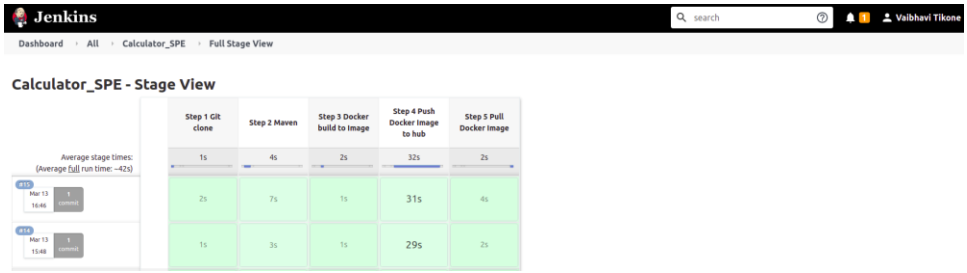
Jenkins manages one script file, where we can include steps for pipeline.



We can set the credentials of docker, ansible and git globally in Jenkins.



We must build our pipeline using build now, and if all stages are successfully completed then the stage view will look like picture mentioned below.



Monitoring

Fundamentally, Continuous Monitoring (CM), sometimes called Continuous Control Monitoring (CCM), is an automated process by which Devops personnel can observe and detect compliance issues and security threats during each phase of the DevOps pipeline. Outside DevOps, the process may be expanded to do the same for any segment of the IT infrastructure in question. It helps teams or organizations monitor, detect, study key relevant metrics, and find ways to resolve said issues in real-time.

Continuous monitoring needs logs of different functions and how they performed. I have used log4j tool for logging purpose. log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License. I haved used org.apache.logger package. I have added logg statements in my code, as shown below:

```
public double Log(double logVal)
{
    logging.info("[INFO]: Value: "+logVal);
    return Math.log(logVal);
}
public double Power(double base, double exp)
{
    logging.info("[INFO]: Base: "+base +" Exponent: "+exp);
    if(base==0 && exp ==0)
        return Double.NaN;
    return Math.pow(base,exp);
}
```

```
case 1: //square root
    double sqrt;
    try
    {
        System.out.print("Enter value: ");
        sqrt = scanner.nextDouble();
    }
    catch (InputMismatchException im)
    {
        logging.error("[ERROR]: Expecting double value encountered different value type. ");
        return;
    }
    System.out.println("Square root of " + sqrt + ": " + calc.Sqrt(sqrt));
    System.out.println("=====================================================");
    break;
```

I have created one xml file to generate the log for continuous monitoring.

deploy-image.yml | inventory | log4j2.xml | Calculator.java | CalculatorTest.java

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

    <Loggers>
        <Root level="debug">
            <AppenderRef ref="ConsoleAppender"/>
            <AppenderRef ref="FileAppender"/>
        </Root>
    </Loggers>

</Configuration>

```

We can submit this logfile to monitoring tool. I have used Elasticsearch, Kibana and logstash tools. Logstash tool needs one configuration file where we specify the path of our log file.

```
input {
  file {
    path => "/home/vaibhavi/Documents/SPE/SPE_MINI/Calculator_SPE/calculator.log"
    start_position => "beginning"
  }
}

filter {
  grok {
    match => [
      "message", "%{HTTPDATE:timestamp_string} \[%{GREEDYDATA:thread}\] \[%{LOGLEVEL:level}\] %{GREEDYDATA:logger}\: %{GREEDYDATA:line}"
    ]
  }

  date {
    match => ["timestamp_string", "dd/MMM/YYYY:HH:mm:ss SSS"]
  }

  mutate {
    remove_field => [timestamp_string]
  }
}

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "sample_calculator_elastic"
  }

  stdout {
    codec => rubydebug
  }
}
```

We need to download elasticsearch, Kibana and Logstash, and then we need to run it from our terminal. I followed below commands:

./path-to-elasticsearch/bin/elasticsearch

./path-to-kibana/bin/kibana

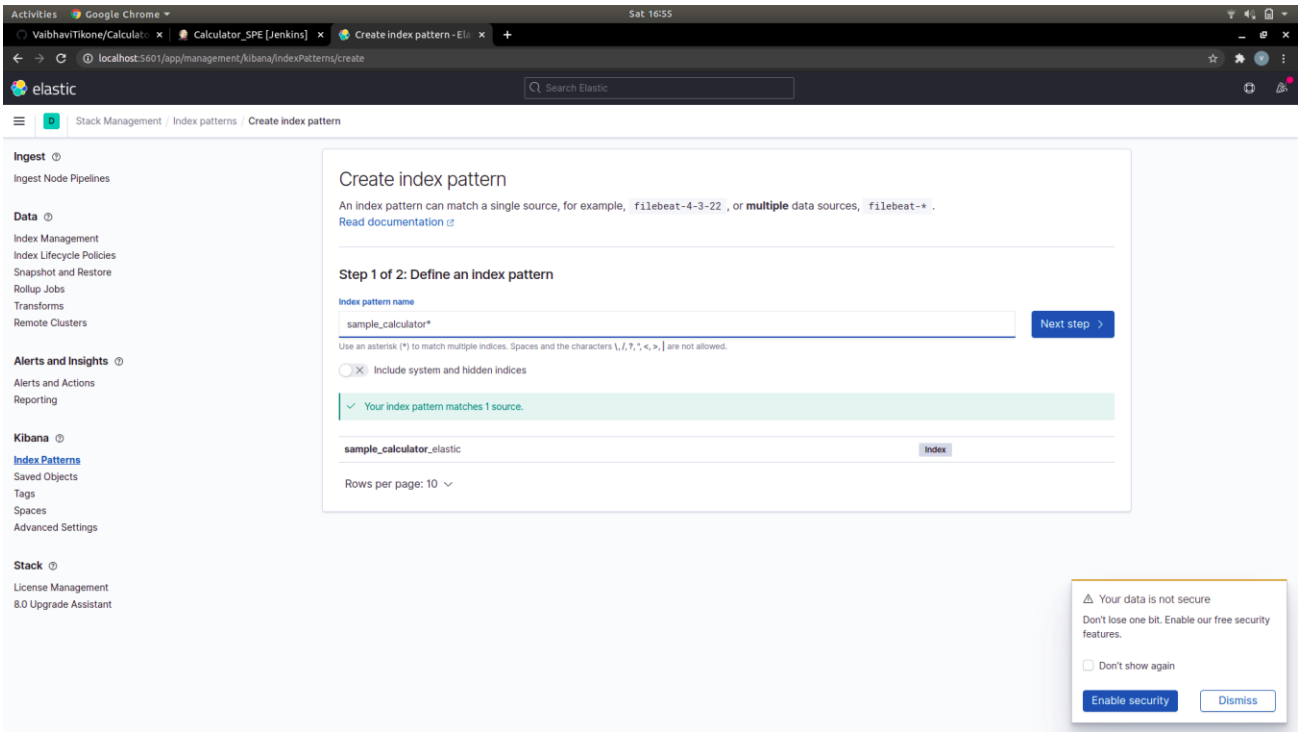
./path-to-logstash/bin/logstash -f /path-to-logfile

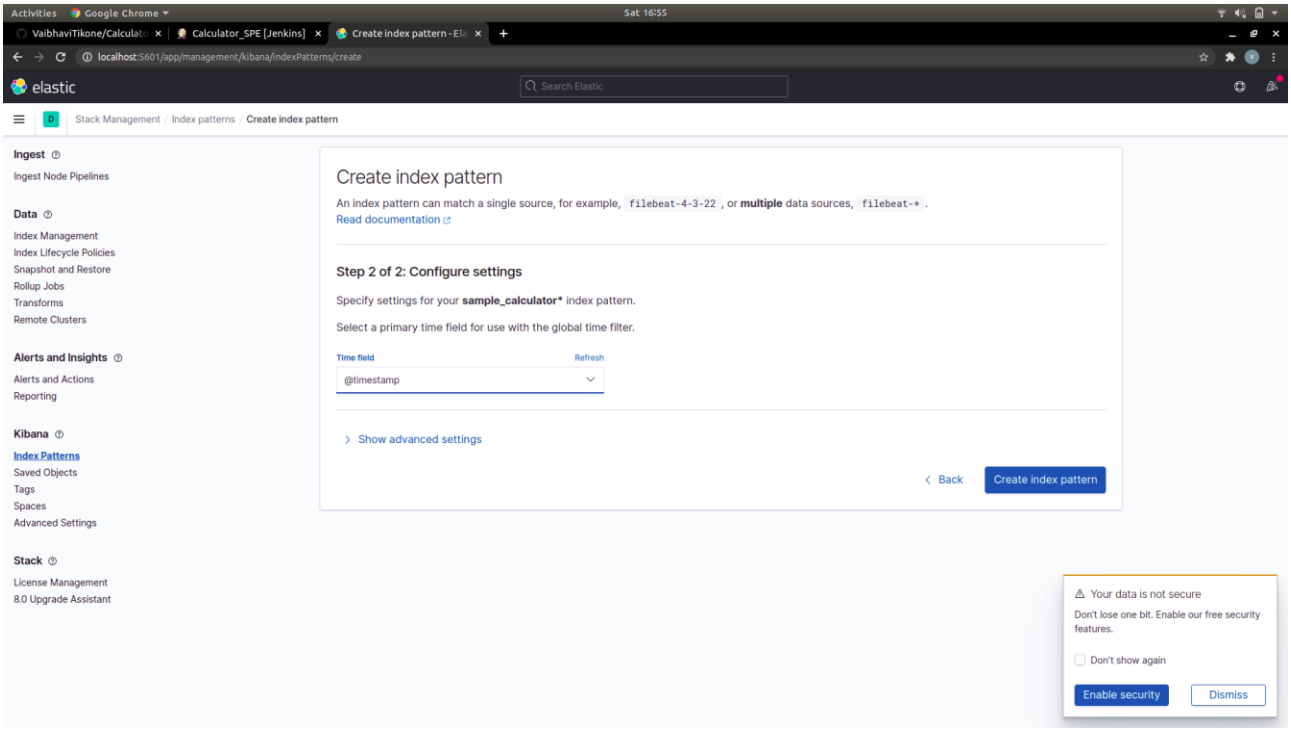
```
vaibhavi@G3:~$ cd Downloads/elasticsearch-7.11.1/
vaibhavi@G3:~/Downloads/elasticsearch-7.11.1$ ./bin/elasticsearch
[2021-03-17T18:02:38,565][INFO ][o.e.n.Node               ] [G3] version[7.11.1]
, pid[28097], build[default/tar/ff17057114c2199c9c1bbecc727003a907c0db7a/2021-02
-15T13:44:09.394032Z], OS[Linux/5.4.0-67-generic/amd64], JVM[AdoptOpenJDK/OpenJD
K 64-Bit Server VM/15.0.1/15.0.1+9]
[2021-03-17T18:02:38,575][INFO ][o.e.n.Node               ] [G3] JVM home [/home
/vaibhavi/Downloads/elasticsearch-7.11.1/jdk], using bundled JDK [true]
[2021-03-17T18:02:38,576][INFO ][o.e.n.Node               ] [G3] JVM arguments [
-Xshare:auto, -Des.networkaddress.cache.ttl=60, -Des.networkaddress.cache.negati
ve.ttl=10, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.encodin
g=UTF-8, -Djna.nosys=true, -XX:-OmitStackTraceInFastThrow, -XX:+ShowCodeDetailsI
nExceptionMessages, -Dio.netty.noUnsafe=true, -Dio.netty.noKeySetOptimization=tr
ue, -Dio.netty.recycler.maxCapacityPerThread=0, -Dio.netty.allocator.numDirectAr
enas=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -Djava.loc
ale.providers=SPI,COMPAT, -XX:+UseG1GC, -Djava.io.tmpdir=/tmp/elasticsearch-3140
182187981932685, -XX:+HeapDumpOnOutOfMemoryError, -XX:HeapDumpPath=data, -XX:Err
orFile=logs/hs_err_pid%p.log, -Xlog:gc*,gc+age=trace,safepoint:file=logs/gc.log:
utctime,pid,tags:filecount=32,filesize=64m, -Xms3909m, -Xmx3909m, -XX:MaxDirectM
emorySize=2049966080, -XX:G1HeapRegionSize=4m, -XX:InitiatingHeapOccupancyPercen
t=30, -XX:G1ReservePercent=15, -Des.path.home=/home/vaibhavi/Downloads/elasticse
arch-7.11.1, -Des.path.conf=/home/vaibhavi/Downloads/elasticsearch-7.11.1/config
, -Des.distribution.flavor=default, -Des.distribution.type=tar, -Des.bundled_jdk
=true]
```

```
vaibhavi@G3:~$ cd Downloads/kibana-7.11.1-linux-x86_64/
vaibhavi@G3:~/Downloads/kibana-7.11.1-linux-x86_64$ ./bin/kibana
  log   [18:03:34.759] [info][plugins-service] Plugin "visTypeXy" is disabled.
  log   [18:03:34.822] [warning][config][deprecation] Config key [monitoring.clu
ster_alerts.email_notifications.email_address] will be required for email notifi
cations to work in 8.0."
  log   [18:03:34.979] [info][plugins-system] Setting up [101] plugins: [taskMan
ager,licensing,globalSearch,globalSearchProviders,code,usageCollection,xpackLega
cy,kibanaUsageCollection,telemetryCollectionManager,telemetry,telemetryCollectio
nXpack,securityOss,newsfeed,mapsLegacy,kibanaLegacy,translations,share,legacyExp
ort,embeddable,uiActionsEnhanced,esUiShared,expressions,charts,bfetch,data,home,
observability,console,consoleExtensions,apmOss,searchprofiler,painlessLab,grokde
bugger,management,indexPatternManagement,advancedSettings,fileUpload,savedObject
s,visualizations,visTypeVislib,visTypeTimeseries,visTypeTimeseriesEnhanced,visTy
peTimelion,features,licenseManagement,dataEnhanced,watcher,canvas,visTypeVega,vi
sTypeMarkdown,visTypeTable,visTypeTagcloud,visTypeMetric,tileMap,regionMap,input
ControlVis,mapsOss,lensOss,graph,timelion,dashboard,dashboardEnhanced,visualize,
discover,discoverEnhanced,savedObjectsManagement,spaces,security,savedObjectsTag
ging,maps,lens,reporting,lists,encryptedSavedObjects,dashboardMode,cloud,upgrade
Assistant,snapshotRestore,fleet,indexManagement,remoteClusters,crossClusterRepli
cation,rollup,indexLifecycleManagement,enterpriseSearch,ml,beatsManagement,trans
```

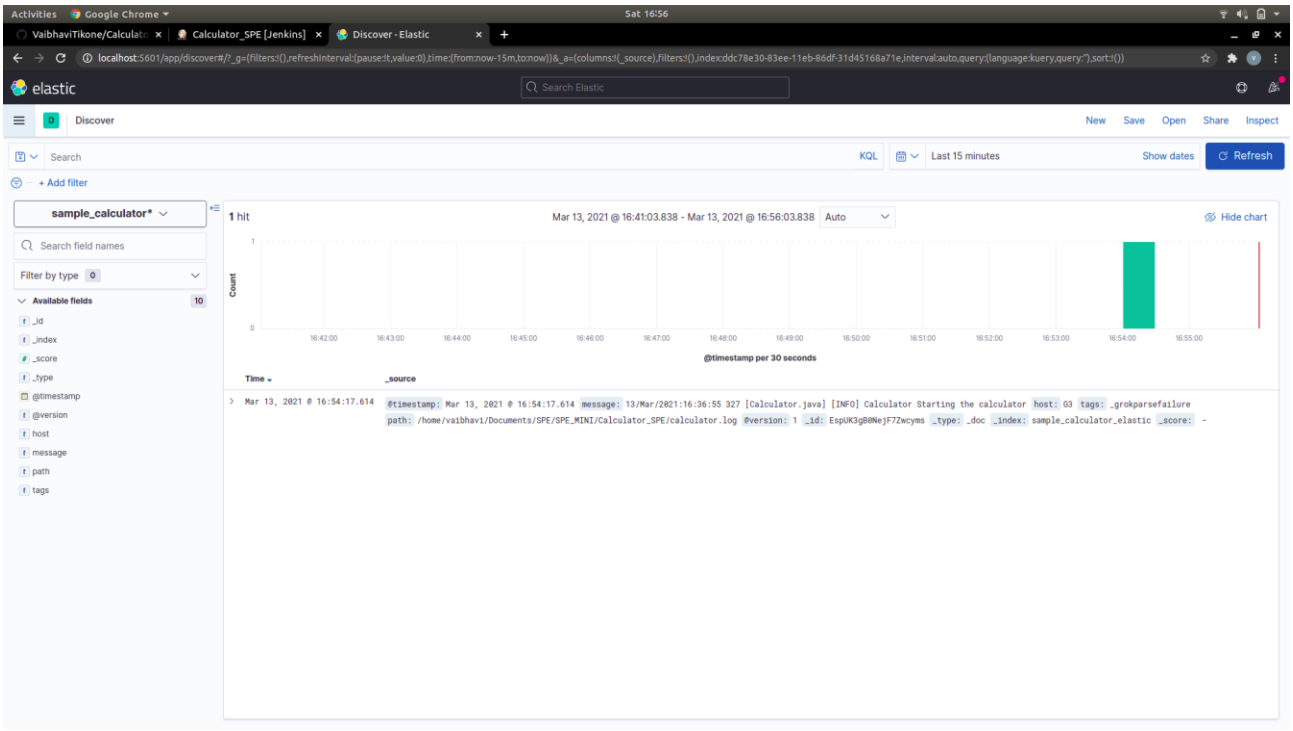
```
Logstash API endpoint {:port=>9600}
{
    "@timestamp" => 2021-03-15T13:24:32.155Z,
          "host" => "G3",
          "line" => "10.0",
         "level" => "INFO",
       "message" => "15/Mar/2021:18:54:32 155 [Calculator.java] [INFO] Calculato
r [INFO]: Value: 10.0",
      "@version" => "1",
        "thread" => "Calculator.java",
          "path" => "/home/vaibhavi/Documents/SPE/SPE_MINI/Calculator_SPE/calcul
ator.log",
        "logger" => "Calculator [INFO]: Value"
}
{
    "@timestamp" => 2021-03-15T13:24:32.161Z,
          "host" => "G3",
          "line" => "0.0",
         "level" => "INFO",
       "message" => "15/Mar/2021:18:54:32 161 [Calculator.java] [INFO] Calculato
r [INFO]: Base: 0.0 Exponent: 0.0",
      "@version" => "1",
        "thread" => "Calculator.java",
```

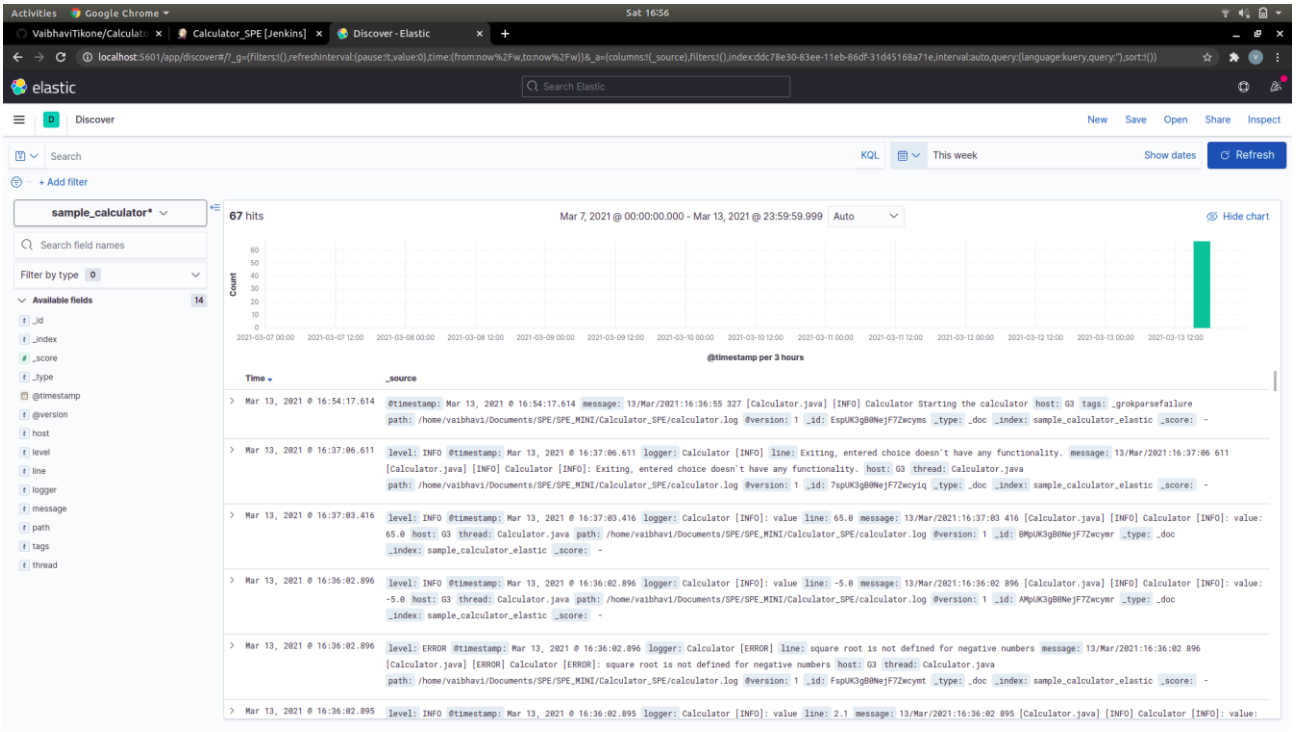
Elasticsearch runs on port number 9200, Kibana runs on port number 5601 and logstash runs on port number 9600. Setting up kibana includes creating a new index pattern under management. Add a new index pattern and press next to choose @timestamp from next window and this will create a new index pattern to view your logs.

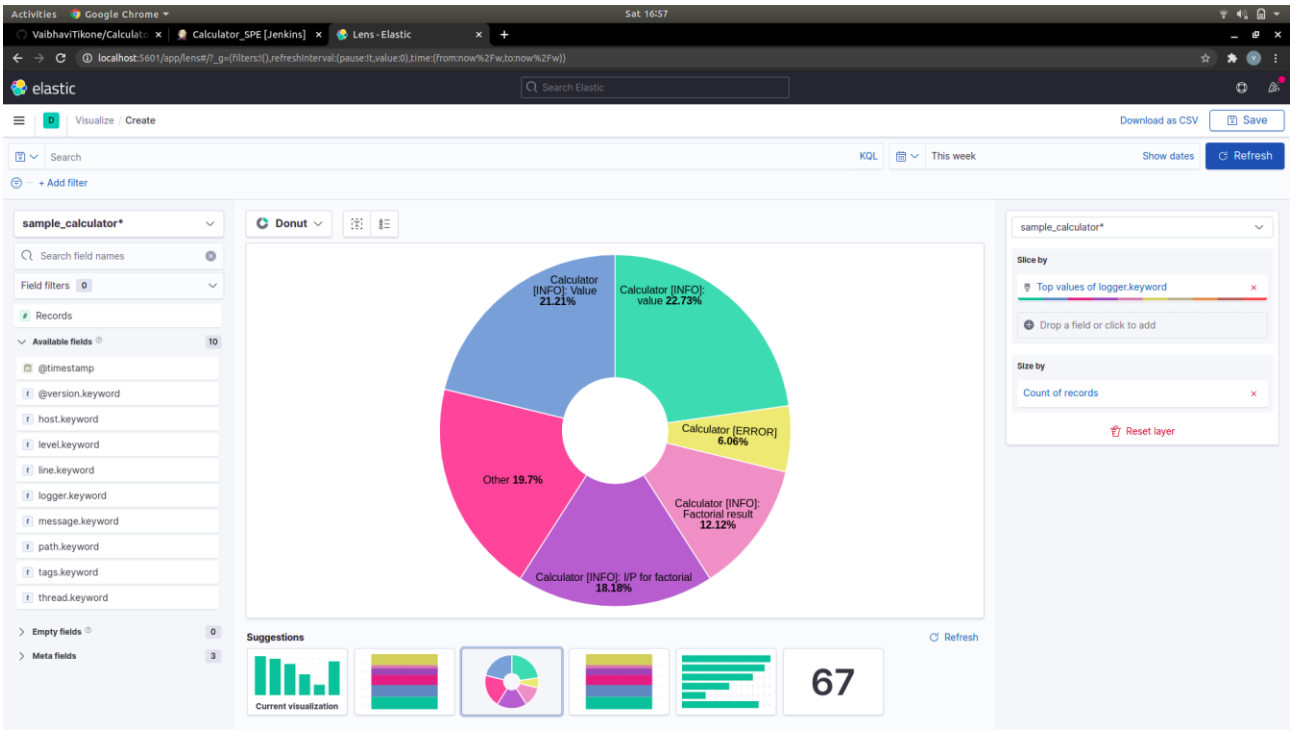




Now you can view your logs and virtualize them based on the chosen index pattern. Make sure to select the time range before it, it is usually configured to ~before 15 minutes, set it appropriately. And click on the discover or visualize to see through the logs and monitor them.




Activities
Google Chrome
Sat 16:56
Calculator_SPE [Jenkins]
Discover - Elastic
elastic
Search Elastic
Discover
New
Save
Open
Share
Inspect
Search
KQL
This week
Show dates
Refresh
+ Add filter
sample_calculator*
67 hits
Mar 7, 2021 @ 00:00:00.000 - Mar 13, 2021 @ 23:59:59.999
Auto
Hide chart
Search field names
Filter by type
Available fields
Count
@timestamp per 3 hours
Time
_source


Activities
Google Chrome
Sat 16:57
Calculator_SPE [Jenkins]
Lens - Elastic
elastic
Search Elastic
Visualize / Create
Download as CSV
Save
Search
KQL
This week
Show dates
Refresh
+ Add filter
sample_calculator*
Donut
Search field names
Field filters
Records
Available fields
@timestamp
@version.keyword
host.keyword
level.keyword
line.keyword
logger.keyword
message.keyword
path.keyword
tags.keyword
thread.keyword
Empty fields
Meta fields
Calculator [INFO]: Value 21.21%
Calculator [INFO]: value 22.73%
Calculator [ERROR] 6.06%
Calculator [INFO]: Factorial result 12.12%
Calculator [INFO]: I/P for factorial 18.18%
Other 19.7%
Slice by
Top values of logger.keyword
Drop a field or click to add
Size by
Count of records
Reset layer
Suggestions
Refresh
Current visualization
67