

# Software Production Engineering

Scientific Calculator with DevOps

**Pratik Pawar, MT2020121**

**March 14, 2021**



## Contents

1	<a href="#">Introduction</a>	3
2	<a href="#">Source Code and Output</a>	4
3	<a href="#">Software Development Life Cycle (SDLC)</a>	5
3.1	<a href="#">Source Code Management (SCM)</a>	6
3.2	<a href="#">Build and Test</a>	7
3.3	<a href="#">Continuous Integration (CI)</a>	8
3.3.1	<a href="#">Jenkins Installation</a>	8
3.3.2	<a href="#">Jenkins Pipeline</a>	9
3.4	<a href="#">Continuous Delivery</a>	11
3.4.1	<a href="#">Docker Installation</a>	11
3.4.2	<a href="#">Building docker image</a>	12
3.4.3	<a href="#">Publishing docker image</a>	13
3.5	<a href="#">Continuous Deployment</a>	14
3.5.1	<a href="#">Rundeck Installation</a>	14
3.5.2	<a href="#">Creating Rundeck Project and Job</a>	15
3.5.3	<a href="#">Jenkins Pipeline</a>	18
3.6	<a href="#">Monitoring</a>	20
3.6.1	<a href="#">Create account on elastic cloud</a>	20
3.6.2	<a href="#">Parse log file using logstash</a>	21
3.6.3	<a href="#">Visualizing logs in Kibana</a>	22
4	<a href="#">References</a>	25

# 1. Introduction

What is DevOps?

DevOps is a set of practices that combines Dev and Ops. It is more over a philosophy, mindset, concept which extends agile principles beyond the boundaries to entire delivered services.

Why should one use DevOps?

- i. Improved development frequency.
- ii. Faster time to market.
- iii. Lower failure rate of new release.
- iv. Shortened lead time between fixes.
- v. Faster mean time to recovery in the event of a new release crashing.

In this project, command line Scientific Calculator is created and DevOps tools are used to automate the process. Currently, the calculator has following menu driven operations.

1. Square root function
2. Factorial function
3. Natural logarithm (base e)
4. Power function

The CI/CD pipeline is built using following tools:

1. Development
  - a. OS: Ubuntu 18.04
  - b. IntelliJ IDEA
  - c. Language: Java
  - d. Java environment: openjdk version "1.8.0-252"
  - e. Build: Apache Maven 3.6.0
  - f. SCM: Git, Github
2. Testing: JUnit
3. Integration: Apache Maven, Jenkins
4. Delivery: Docker, Dockerhub, Jenkins
5. Deployment: Docker, Rundeck, Jenkins
6. Monitoring: ELK stack

The entire source code can be found here:

<https://github.com/WidoutSyntax/Scientific-Calculator-with-Devops>

## 2. Source Code and Output

The source code is written in java and menu driven program will be run for users. The code has Calculator.java class in which simple java program is written. There is one more test class is written for unit testing. All dependencies are in pom.xml.

Calculator class code

```
public class Calculator {
    private static final Logger logger = LogManager.getLogger(Calculator.class);

    public Calculator()
    {

    }

    public static void main(String[] args)
    {
        Calculator calculator = new Calculator();
        Scanner scanner = new Scanner(System.in);
        do {
            System.out.println("Following operations are available");
            System.out.print("1. Square root\n" +
                            "2. Factorial\n" +
                            "3. Natural Log\n" +
                            "4. Power\n" +
                            "0. Exit\n" +
                            "\nEnter your choice: ");

            public double sqrt(double num)
            {
                logger.info("[SQUAREROOT] - " + num);
                double res = Math.sqrt(num);
                logger.info("[RESULT - SQUAREROOT] - " + res);
                return res;
            }

            public int fact(int num) {
                logger.info("[FACTORIAL] - " + num);
                int res = 1;
                for(int i=1;i<=num;i++)
                    res = res * i;
                logger.info("[RESULT - FACTORIAL] - " + res);
                return res;
            }

            public double nlog(double num)
            {
                logger.info("[LOGBASEE] - " + num);
                double res = Math.log(num);
                logger.info("[RESULT - LOGBASEE] - " + res);
                return res;
            }
        }
    }
}
```

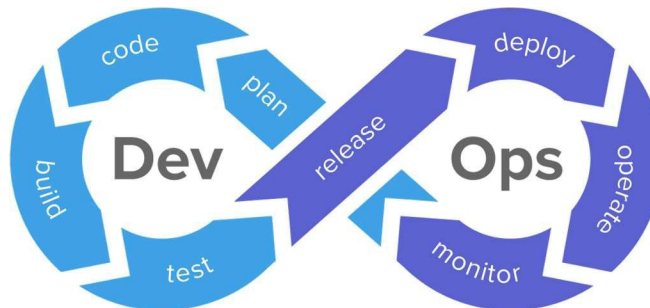
## Calculator test class code

```
public class CalculatorTest {
    Calculator calculator = new Calculator();
    private static final double delta = 1e-2;

    @Test
    public void squareRootCorrect(){
        assertEquals( message: "Square root of 16", expected: 4, calculator.sqrt( num: 16), delta);
        assertEquals( message: "Square root of 25", expected: 5, calculator.sqrt( num: 25), delta);
    }

    @Test
    public void squareRootIncorrect(){
        assertEquals( message: "Square root of 16", unexpected: 6, calculator.sqrt( num: 16), delta);
        assertEquals( message: "Square root of 25", unexpected: 7, calculator.sqrt( num: 25), delta);
    }
}
```

## 3. Software Development Life Cycle (SDLC)



There are many stages in software development life cycle. These stages are automated using various DevOps tools. There are no tools for planning as such. For code step Git and Github are used. For building and testing Maven and Junit are used respectively. At delivery step, docker image is built and pushed to Dockerhub. Deployment is done using Rundeck and in this project we are deploying project's docker image on docker container. And at last, for monitoring ELK stack is used.

## 3.1 Source Code Management (SCM)

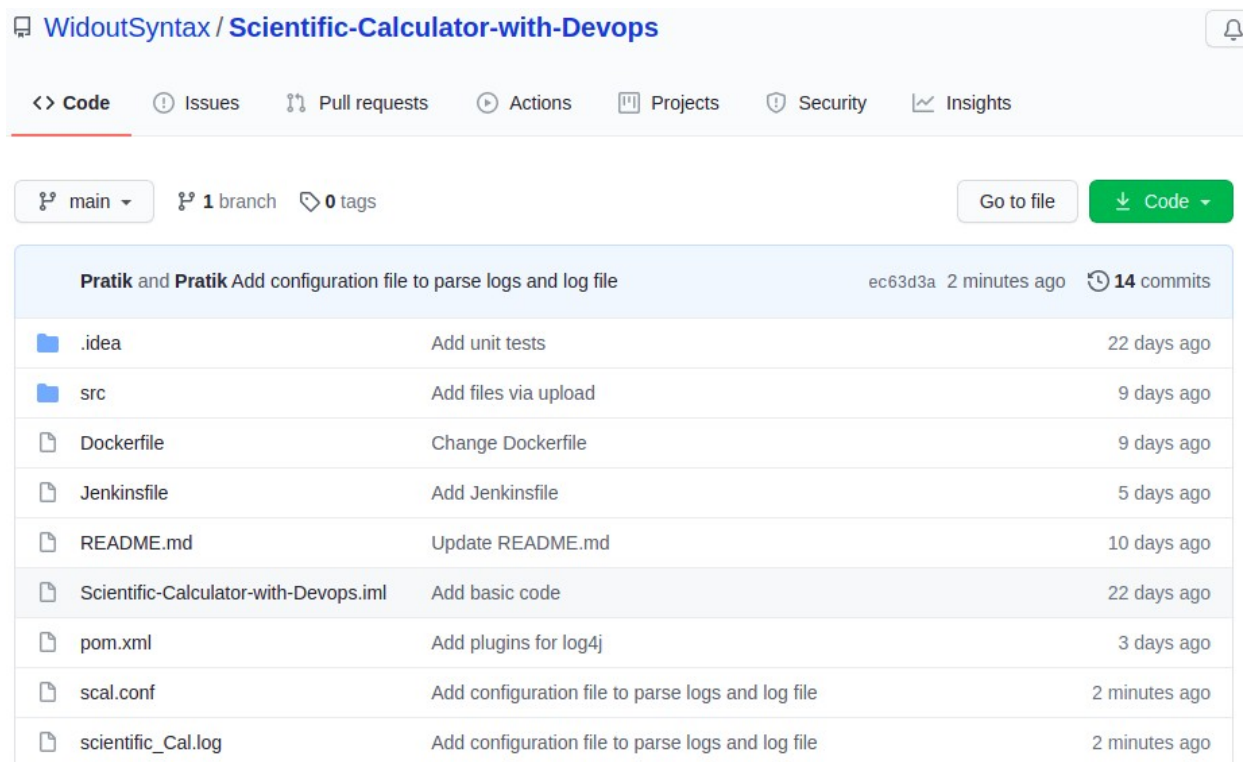
Source Code Management is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors. SCM is also synonymous with Version Control. Here, we Git is used as SCM. Git is a distributed version control system. Github is an online repository hosting system.

The development of this project is done incrementally. Some of the important increments are as follows:

1. Add basic code.
2. Add unit tests.
3. Add Dockerfile.
4. Add main class detection.
5. Add files via upload.

All other commits can be seen here:

<https://github.com/WidoutSyntax/Scientific-Calculator-with-Devops/commits>



WidoutSyntax / Scientific-Calculator-with-Devops

<> Code Issues Pull requests Actions Projects Security Insights

main 1 branch 0 tags Go to file Code

Pratik and Pratik Add configuration file to parse logs and log file		ec63d3a 2 minutes ago 14 commits
.idea	Add unit tests	22 days ago
src	Add files via upload	9 days ago
Dockerfile	Change Dockerfile	9 days ago
Jenkinsfile	Add Jenkinsfile	5 days ago
README.md	Update README.md	10 days ago
Scientific-Calculator-with-Devops.iml	Add basic code	22 days ago
pom.xml	Add plugins for log4j	3 days ago
scal.conf	Add configuration file to parse logs and log file	2 minutes ago
scientific_Cal.log	Add configuration file to parse logs and log file	2 minutes ago

## 3.2 Build and Test

In the project, Apache Maven is used for managing dependencies and building the project. Maven is used for adding jars and other dependencies of the project. It is Maven who finally outputs the SNAPSHOT jar of the project that has compiled classes along with other classes the project depends on. All dependencies are in pom.xml file. Project's pom.xml file is shown below.

The screenshot shows an IDE with several tabs: 'pom.xml (Scientific-Calculator-with-Devops)', 'log4j2.xml', 'Dockerfile', 'Calculator.java', and 'CalculatorT'. The 'pom.xml' tab is active, displaying the following XML content:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4
5         <modelVersion>4.0.0</modelVersion>
6
7         <groupId>org.example</groupId>
8         <artifactId>Scientific-Calculator-with-Devops</artifactId>
9         <version>1.0-SNAPSHOT</version>
10
11     <dependencies>
12         <dependency>
13             <groupId>junit</groupId>
14             <artifactId>junit</artifactId>
15             <version>4.13.2</version>
16         </dependency>
17         <dependency>
18             <groupId>org.apache.logging.log4j</groupId>
19             <artifactId>log4j-api</artifactId>
20             <version>2.14.0</version>
21         </dependency>
22         <dependency>
23             <groupId>org.apache.logging.log4j</groupId>
24             <artifactId>log4j-core</artifactId>
25             <version>2.14.0</version>
26         </dependency>
27     </dependencies>
```

Here, main file name is Calculator. So, when one tries to run program using `java -jar` command he will get main file not found error. To resolve this issue, following code snippet is added in pom.xml.

```
<build>
  <plugins>
    <plugin>
      <!-- Build an executable JAR -->
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.0.2</version>
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
            <classpathPrefix>lib/</classpathPrefix>
            <mainClass>Calculator</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

To build the project without running test cases:

```
> mvn -B -DskipTests clean package
```

clean will remove the target folder. Code will be compiled and packaged in distributable format such as jar. In this command, compiled source is not tested using unit testing.

Junit is a Java unit testing framework. It is used to write and run repeatable automated tests.

To test the project:

```
> mvn test
```

This command will test compiled source code using a suitable unit testing framework. These tests should not require the code to be packaged or deployed.

### **3.3 Continuous Integration (CI)**

Continuous integration is a development practice where developers integrate code into shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated testing. Jenkins is used for CI. Basically, Jenkins keeps an eye on SCM system for changes in the code and builds the code as and when it detects the changes.

#### **3.3.1 Jenkins Installation**

To install Jenkins follow the step given below:

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-ci.org/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`
3. `sudo apt update`
4. `sudo apt install Jenkins`
5. `sudo systemctl enable --now jenkins`

Jenkins will run on <http://localhost:8080/>

Create new user.

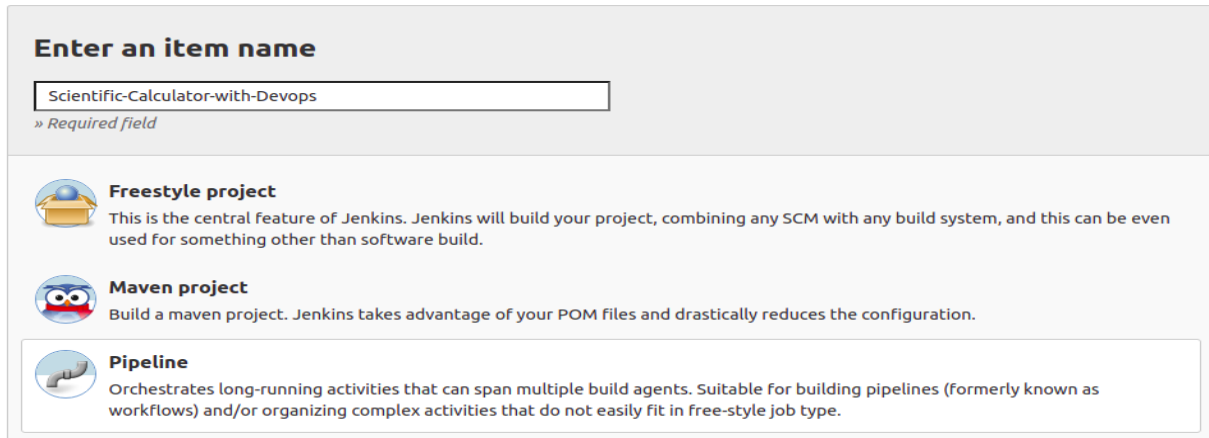
Install Git, Maven, Junit, Docker, Rundeck plugins.



### 3.3.2 Jenkins Pipeline

Follow the steps to create Jenkins pipeline:

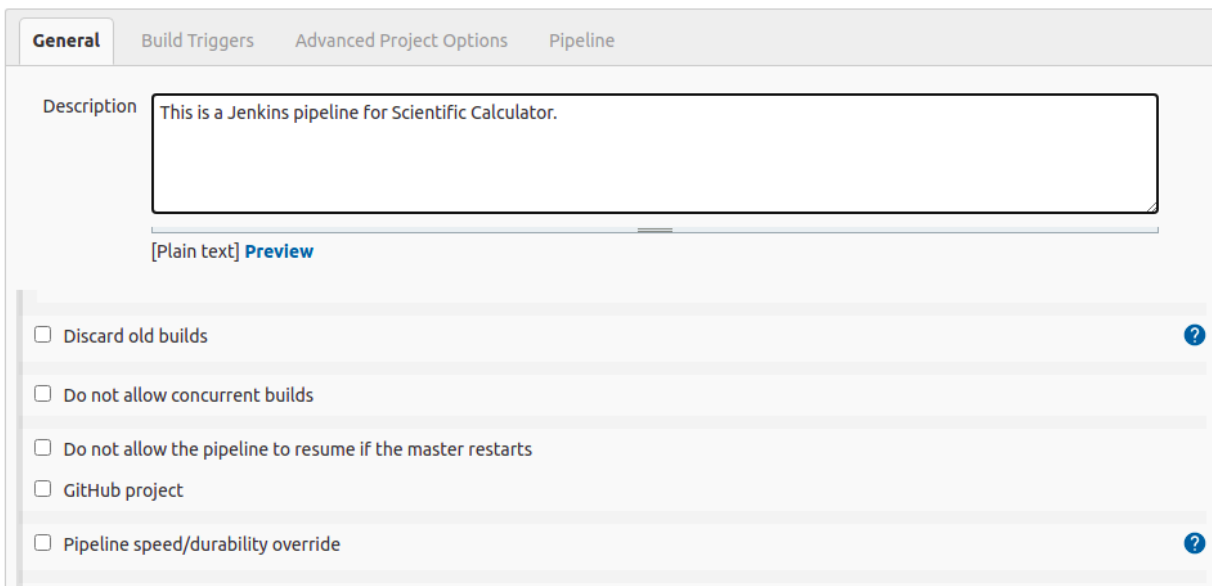
1. Create new Jenkins pipeline as shown below



The screenshot shows the 'Enter an item name' dialog box in Jenkins. At the top, there is a text input field containing 'Scientific-Calculator-with-Devops'. Below the field, it says '» Required field'. Below the input field, there are three options, each with an icon and a description:

- 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.

2. Add some description



The screenshot shows the 'General' tab of the Jenkins configuration page. The 'Description' field is filled with the text 'This is a Jenkins pipeline for Scientific Calculator.' Below the description field, there is a '[Plain text] Preview' link. Below the description field, there are several checkboxes for project options:

- ☐ Discard old builds
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the master restarts
- ☐ GitHub project
- ☐ Pipeline speed/durability override

### 3. Choose build trigger

### 4. Add the following code in pipeline script:

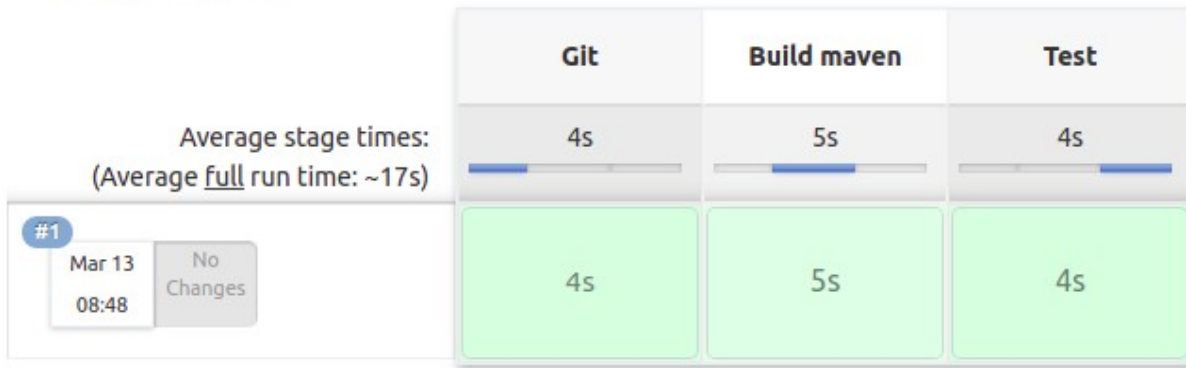
```
pipeline {
  agent any
  stages {
    stage('Git') {
      steps {
        // Get some code from a GitHub repository
        git branch: 'main', url: 'https://github.com/WidoutSyntax/Scientific-
Calculator-with-Devops.git'
      }
    }
    stage('Build maven') {
      steps {
        // Run Maven on a Unix agent.
        sh "mvn -B -DskipTests clean package"
      }
    }

    stage('Test') {
      steps {
        // Run Maven on a Unix agent.
        sh "mvn test"
      }
    }
  }
}
```

}

5. Save the pipeline
6. Click the build now button to check everything is working fine.

## Stage View



Until now, SCM pulling, building, testing is automated and jar file is created in target folder. This is continuous integration.

## 3.4 Continuous Delivery

Continuous delivery is an extension of continuous integration since it automatically deploys all code changes to a testing or pre-production environment after the build stage. This is like automated release process.

In the delivery stage of our pipeline, docker image of application is built using Dockerfile and then that image is delivered to Dockerhub. Docker and Jenkins are used for continuous delivery.

### 3.4.1 Docker Installation

Docker is a tool designed to make it easier to create, deploy and run applications by using containers. Containers allow to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. In a way, Docker is like a virtual machine but rather than creating whole virtual operating system,

docker allows applications to use the same linux kernel as the system they are running on and only requires applications be shipped with things required to run application.

Follow the steps to set your system for docker:

1. To install docker  
>apt-get install docker.io
2. Give permission to Jenkins to run docker commands  
>sudo usermod -aG docker jenkins

### 3.4.2 Building docker image

Maven build outputs a jar SNAPSHOT with all required dependencies in target folder. A Dockerfile is a text document a user could call on the command line to assemble an image. To build docker image, following code is added in pipeline script. To add code click on configure button.

```
environment {  
    registry = "pratikiiitb/scientific-calculator"  
    registryCredential = 'dockerhub'  
    dockerImage = "  
}  
  
stage('Building docker image') {  
    steps{  
        script {  
            dockerImage = docker.build registry + ":latest"  
        }  
    }  
}
```

Dockerfile:

```
FROM openjdk:8  
COPY ./target/Scientific-Calculator-with-Devops-1.0-SNAPSHOT.jar ./  
WORKDIR ./  
CMD ["java", "-jar", "Scientific-Calculator-with-Devops-1.0-SNAPSHOT.jar"]
```

### 3.4.3 Publishing docker image

Now, pratikiiitb/scientific-calculator docker image is present on machine on which Jenkins is running.

To publish this docker image on dockerhub first sign up on dockerhub. Dockerhub is a hosted repository service provided by Docker for finding and sharing container images. Add the following script in pipeline so that Jenkins will run it after all previous scripts are successfully executed.

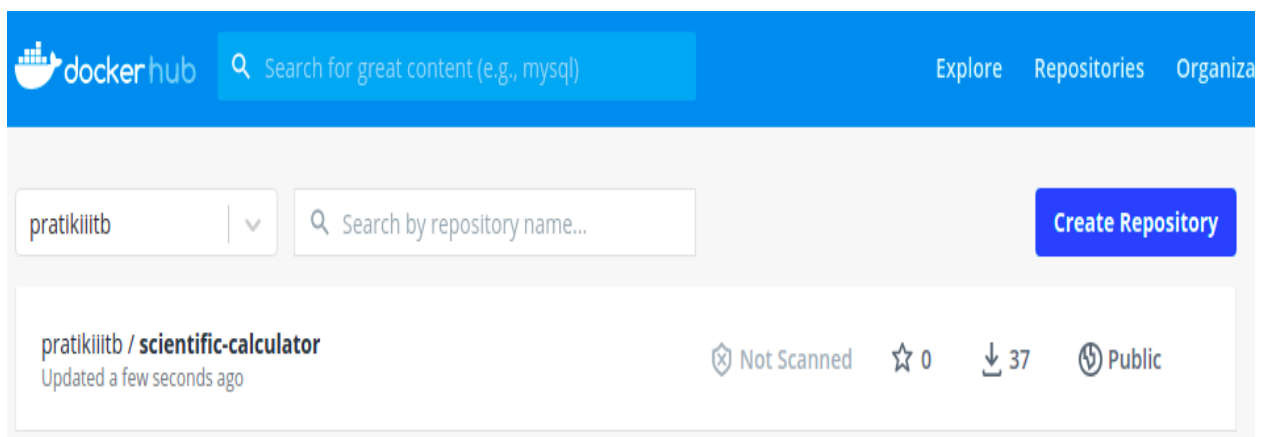
```
stage('Push docker image to dockerhub') {  
    steps{  
        script {  
            docker.withRegistry( '', registryCredential ) {  
                dockerImage.push()  
            }  
        }  
    }  
}
```

Click build now to check everything is working fine till continuous delivery

#### Stage View



Dockerhub:



So far, we are done with continuous integration and continuous delivery. Now the next step is to deploy that latest docker image on managed and controlled nodes.

## **3.5 Continuous Deployment**

Continuous deployment goes one step further than continuous delivery. With continuous deployment every change that passes all stages of pre production pipeline is released to customers. There is no human intervention and only a failed test will prevent a new change to be deployed to production. Continuous deployment is an excellent way to accelerate the feedback loop with your customers and take pressure off the team as there is not a Release day anymore. Developers can focus on building softwares, and their work can go live in minutes.

After the deliverable (docker image in my case) is created and published to Dockerhub, I used Rundeck to fetch the image and deploy it in a container. And the Rundeck job is invoked by Jenkins.

### **3.5.1 Rundeck Installation**

Rundeck is a centralized console for managing the automation in environment. It allows to control who can do what and leverage existing scripts. One can interact with Rundeck using web interface, command line or API. Rundeck is an automation tool that executes Rundeck jobs on Rundeck nodes. Rundeck jobs can be thought of a sequence of instructions and Rundeck nodes could be anything like a web server, container, etc.

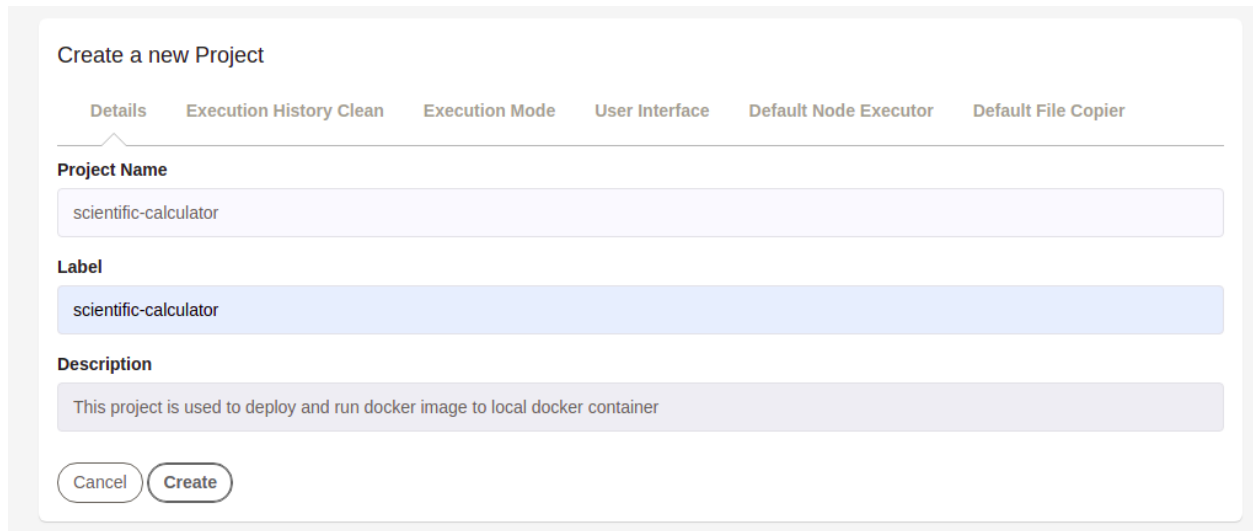
To install Rundeck, follow the steps:

1. Rundeck requires java 8 on your system check for java 8 if not present change to java 8 environment.
2. `>echo "deb https://rundeck.bintray.com/rundeck-deb /" | sudo tee -a /etc/apt/sources.list.d/rundeck.list`
3. `>curl 'https://bintray.com/user/downloadSubjectPublicKey?username=bintray' | sudo apt-key add -`
4. `>sudo apt-get update`
5. `>sudo apt-get install rundeck`
6. `>sudo service rundeckd start`

Rundeck runs on <http://localhost:4440/> and default username and password is admin and admin.

### 3.5.2 Creating Rundeck Project and Job

#### 1. Create new Rundeck project



Create a new Project

Details Execution History Clean Execution Mode User Interface Default Node Executor Default File Copier

**Project Name**

scientific-calculator

**Label**

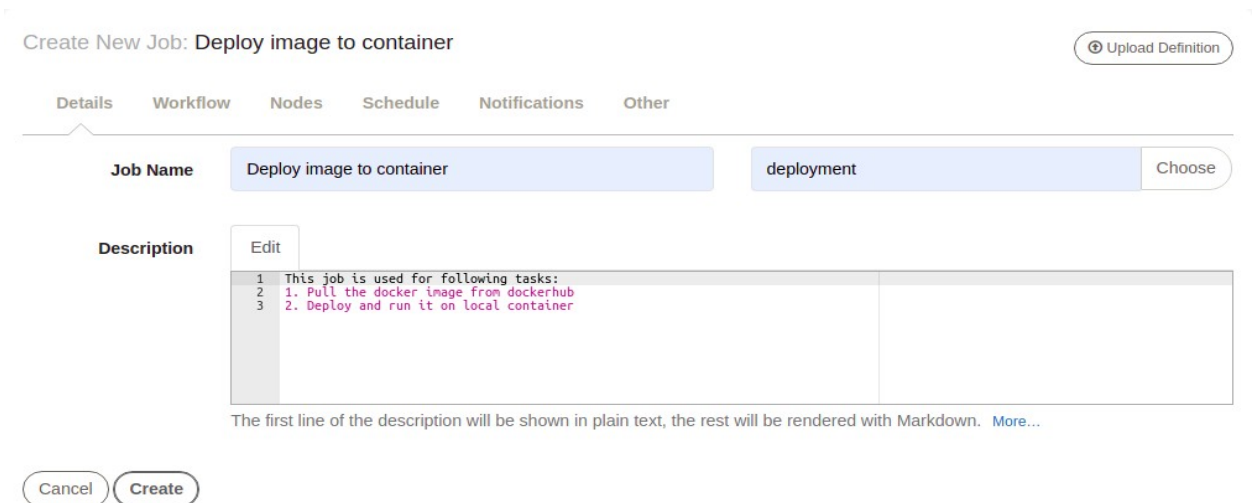
scientific-calculator

**Description**

This project is used to deploy and run docker image to local docker container

Cancel Create

#### 2. Create new Rundeck job by clicking on jobs tab on left side and do the following configuration settings



Create New Job: Deploy image to container Upload Definition

Details Workflow Nodes Schedule Notifications Other

**Job Name** Deploy image to container deployment Choose

**Description** Edit

```
1 This job is used for following tasks:
2 1. Pull the docker image from dockerhub
3 2. Deploy and run it on local container
```

The first line of the description will be shown in plain text, the rest will be rendered with Markdown. [More...](#)

Cancel Create

### 3. Workflow

Create New Job: Deploy image to container

Upload Definition

Details Workflow Nodes Schedule Notifications Other

#### Options

Undo Redo

No Options

+ Add an option

#### Workflow

If a step fails:

☒ Stop at the failed step. ☐ Run remaining steps before failing.

Strategy: Node First

Execute all steps on a node before proceeding to the next node.





Explain

Global Log Filters + add

### 4. Add following steps

Global Log Filters + add

Undo Redo Revert All Changes

1.  `docker rm -f calcontainer`  
[Remove previous container](#)
2.  `docker rmi -f pratikiitb/scientific-calculator:latest`  
[Remove previous image](#)
3.  `docker pull pratikiitb/scientific-calculator:latest`  
[Pull latest docker image from dockerhub](#)
4.  `docker run -t -d --name calcontainer pratikiitb/scientific-calculator:latest`  
[Run the calcontainer in detach mode](#)

+ Add a step



## 5. Nodes

Create New Job: Deploy image to container

Details

Workflow

Nodes

Schedule

Notifications

Other

Nodes

☐ Dispatch to Nodes

☒ Execute locally

Choose whether the Job will run on filtered nodes or only on the local node.

Cancel

Create

## 6. Note the UUID of the job

scientific-calculator

deployment

Deploy image to container

This job is used for following tasks: [Less](#)

1. Pull the docker image from dockerhub

2. Deploy and run i on local container

79181c66-def5-4019-b387-dc62ca2e881c

## 7. Run this job in Rundeck to check everything is working fine.

deployment

Deploy image to container

Succeeded 0.00:16 at 8:47 am

#47

Run Again

Log Output »

100% 1/1 COMPLETE

0 FAILED

0 INCOMPLETE

0 NOT STARTED

Node	Start time	Duration
localhost		0.00:00
> Remove previous container	8:46:43 am	0.00:00
> Remove previous image	8:46:43 am	0.00:00
> Pull latest docker image from dockerhub	8:46:43 am	0.00:10
> Run the calcontainer in detach mode	8:46:53 am	0.00:05

```
pratik@pratik-Inspiron-5559:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
pratikiiitb/scientific-calculator	latest	1af6cea812fc	22 hours ago	514MB
hello-world	latest	d1165f221234	8 days ago	13.3kB
pratikiiitb/user1	latest	401a857f32fc	2 weeks ago	298MB
openjdk	8	9324460525ca	4 weeks ago	514MB
mysql	latest	2933adc350f3	4 weeks ago	546MB
ubuntu	latest	f63181f19b2f	7 weeks ago	72.9MB

```
pratik@pratik-Inspiron-5559:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6e211b5ee87b	pratikiiitb/scientific-calculator:latest	"java -jar Scientifi..."	47 seconds ago	Exited (1) 38 seconds ago		calcontainer
be11651b2686	hello-world	"/hello"	5 days ago	Exited (0) 5 days ago		vigorous_goldstine
747025f2c1f6	hello-world	"/hello"	5 days ago	Exited (0) 5 days ago		charming_ellis
8ca530434ddd	ubuntu	"/bin/bash"	6 days ago	Exited (255) 5 days ago		caldevops

Now this Rundeck job should be triggered from Jenkins.

### 3.5.3 Jenkins Pipeline

Follow the steps given below to trigger the Rundeck job from Jenkins.

1. Add Rundeck instance in Jenkins  
(Manage Jenkins → Configure System → Add Rundeck)

#### Rundeck

Job cache

☐ Enable Rundeck job cache

Rundeck job cache configuration

Instances

Name	scientific-calculator	
URL	http://localhost:4440	?
Login	admin	?
Password	.....	?
Auth Token		?
API Version		?

Your Rundeck instance is alive, and your credentials are valid !

Test Connection

Delete Rundeck

Add Rundeck

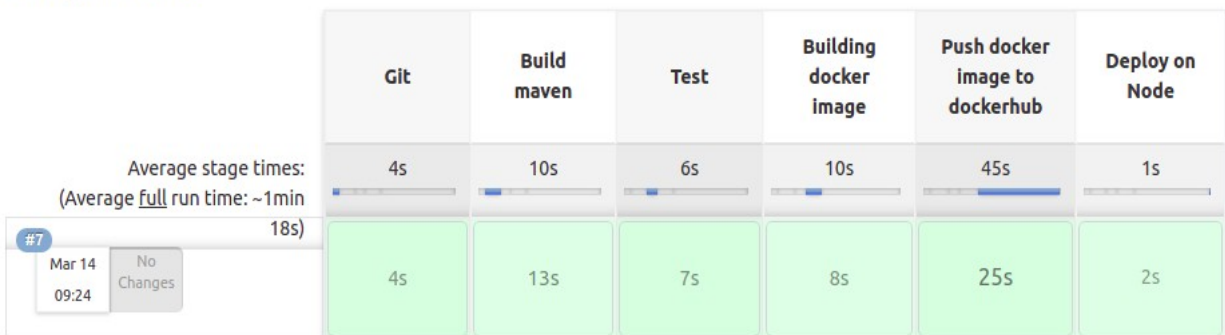
2. Add the following script in pipeline script

```
stage('Deploy on Node') {
  steps {
    script {
      step([
        $class: "RundeckNotifier",
        includeRundeckLogs: true,
        rundeckInstance: "scientific-calculator",
        jobId: "79181c66-def5-4019-b387-dc62ca2e881c",
        shouldWaitForRundeckJob: true,
        shouldFailTheBuild: true,
        tailLog: true
      ])
    }
  }
}
```

3. Click on Save.

Final pipeline view

### Stage View



1. Developers push the code on Github
2. Jenkins detects the change and build the code.
3. Jenkins test the compiled code.
4. Jenkins trigger the ob of building docker image.
5. Push that docker image on Dockerhub
6. Jenkins trigger Rundeck job to deploy docker image on container.

## 3.6 Monitoring

Monitoring provides feedback from production and delivers information about an application's performance and usage patterns. When performance or other issues arise, relevant data about the issues are sent back to development teams through automated monitoring.

Here, I used ELK stack for monitoring.

Elasticsearch is a modern search and analytics engine which is based on Apache Lucene. It is used as to store and index logs and can be then queried to extract meaningful insights. It can be used for numerous types of data including textual, numerical, geospatial, structured, and unstructured.

Logstash is a tool that is used for parsing logs. It is very useful in parsing unstructured logs and giving them structure so that logs can be efficiently searched and analyzed. Log aggregated and processed by Logstash go through 3 stages – collection, processing and dispatching.

Kibana adds a visualization layer to the Elastic Stack. It is a browser-based user interface that can be used to search, analyze and visualize the data stored in Elasticsearch indices.

elasticsearch, logstash, kibana can be downloaded and run locally or elasticsearch and kibana can be run on cloud and logs can be sent using logstash which runs on local environment. I used the later approach.

### 3.6.1 Create account on elastic cloud

Follow the steps given below to create trail 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

### 3.6.2 Parse log file using logstash

I used log4j to collect logs. When the program is being run on any node the logs get collected in scientific\_Cal.log file.

1. So, after deployment of docker image to container use the following command to run program in container

```
>docker exec -it calcontainer /bin/bash
```

```
>java -jar Scientific-Calculator-with-Devops-1.0-SNAPSHOT.jar
```

OR

```
>docker exec -it calcontainer java -jar Scientific-Calculator-with-Devops-1.0-SNAPSHOT.jar
```

2. Copy log file on server side i.e., localhost in my case

```
>docker cp calcontainer:scientific_Cal.log
```

```
/home/pratik/scientific_Cal.log
```

3. Run the logstash using following command

```
>bin/logstash -f /home/pratik/Documents/SPE/MiniProject/Scientific-Calculator-with-Devops/scal.conf
```

```
{
  "@timestamp" => 2021-03-14T05:39:35.816Z,
  "level" => "INFO",
  "action" => "LOGBASEE",
  "line" => "2.719",
  "host" => "pratik-Inspiron-5559",
  "@version" => "1",
  "path" => "/home/pratik/scientific_Cal.log",
  "message" => "14/Mar/2021:11:09:35 816 [Calculator.java] [INFO] Calculator [LOGBASEE] - 2.719",
  "thread" => "Calculator.java",
  "logger" => "Calculator"
}

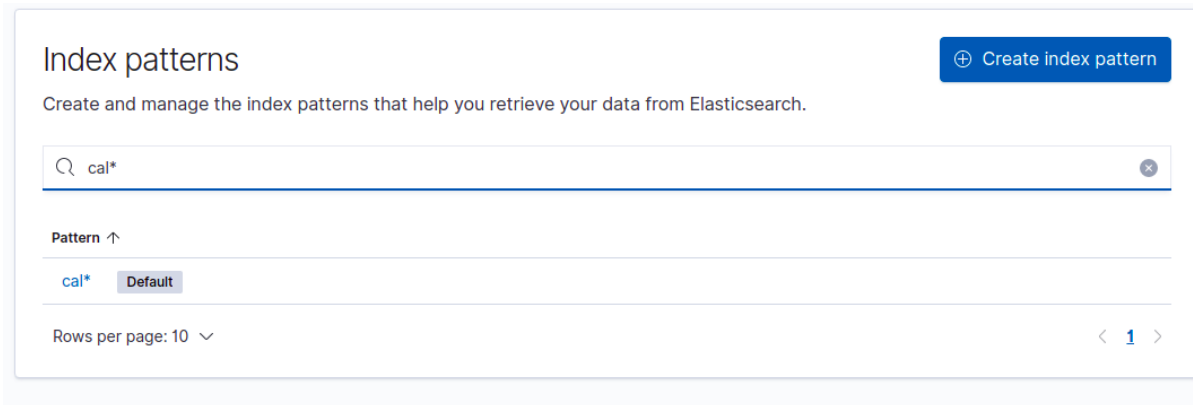
{
  "@timestamp" => 2021-03-14T05:39:44.856Z,
  "level" => "INFO",
  "action" => "RESULT - LOGBASEE",
  "line" => "1.0000002576038383",
  "host" => "pratik-Inspiron-5559",
  "@version" => "1",
  "path" => "/home/pratik/scientific_Cal.log",
  "message" => "14/Mar/2021:11:09:44 856 [Calculator.java] [INFO] Calculator [RESULT - LOGBASEE] - 1.0000002576038383",
  "thread" => "Calculator.java",
  "logger" => "Calculator"
}
```

scal.conf is a configuration file. Output is given to elasticsearch running on elastic cloud. We have to mention cloud link and username, password in conf file.

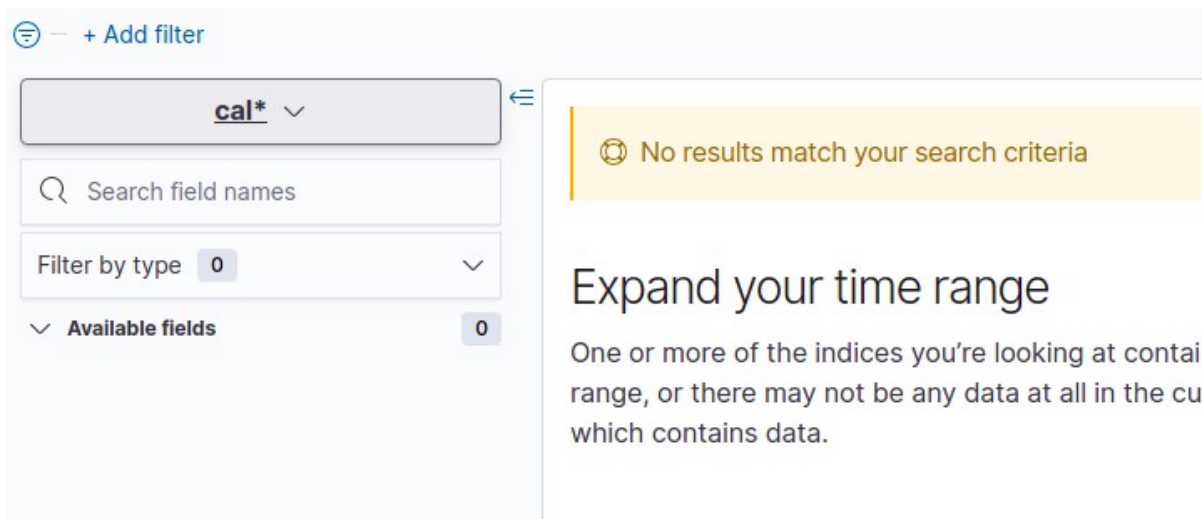
### 3.6.3 Visualizing Logs in Kibana

Follow the given steps in Kibana:

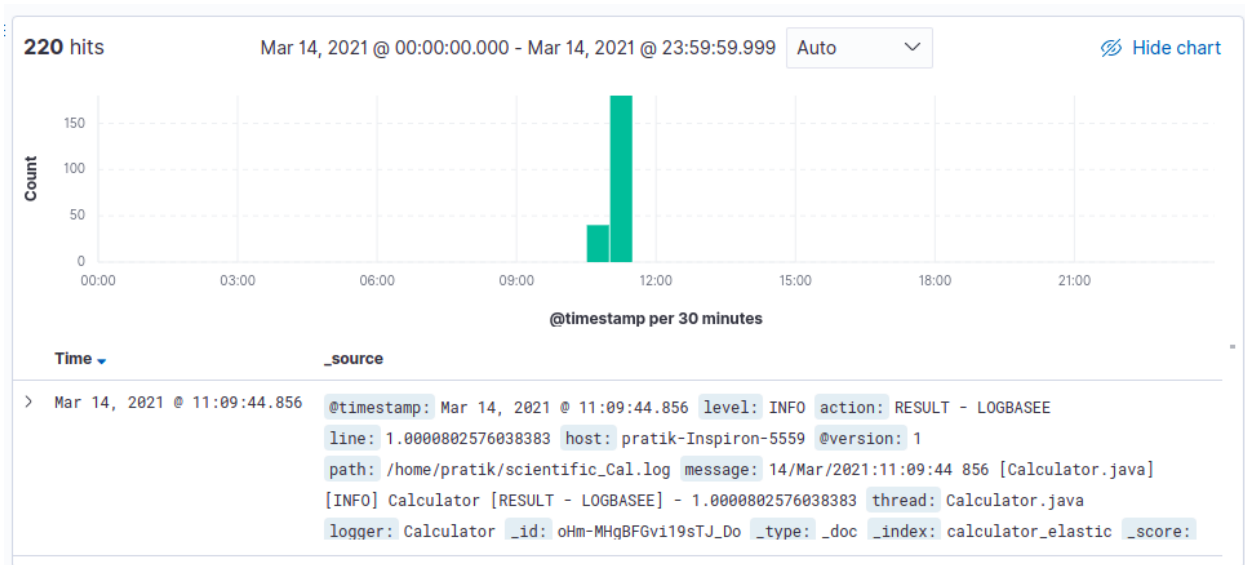
1. On the Kibana dashboard, go to Management → Stack Management → Kibana → Index Pattern. Create the index pattern cal-\*. We could see calculator\_elastic. Click on Next step



2. Go to discover on Kibana dashboard. We could see index pattern dropdown.



### 3. Select time period using calendar icon



### 4. Add filters as shown below and update graph

The 'EDIT FILTER' dialog box is shown with the following configuration:

- Field:** action.keyword
- Operator:** is
- Value:** SQUAREROOT
- ☐ Create custom label?

Buttons: Cancel, Save

5. Create pie chart using visualize and bucket filters

cal\*

Data Options

**Buckets**

Split slices

Aggregation Terms help

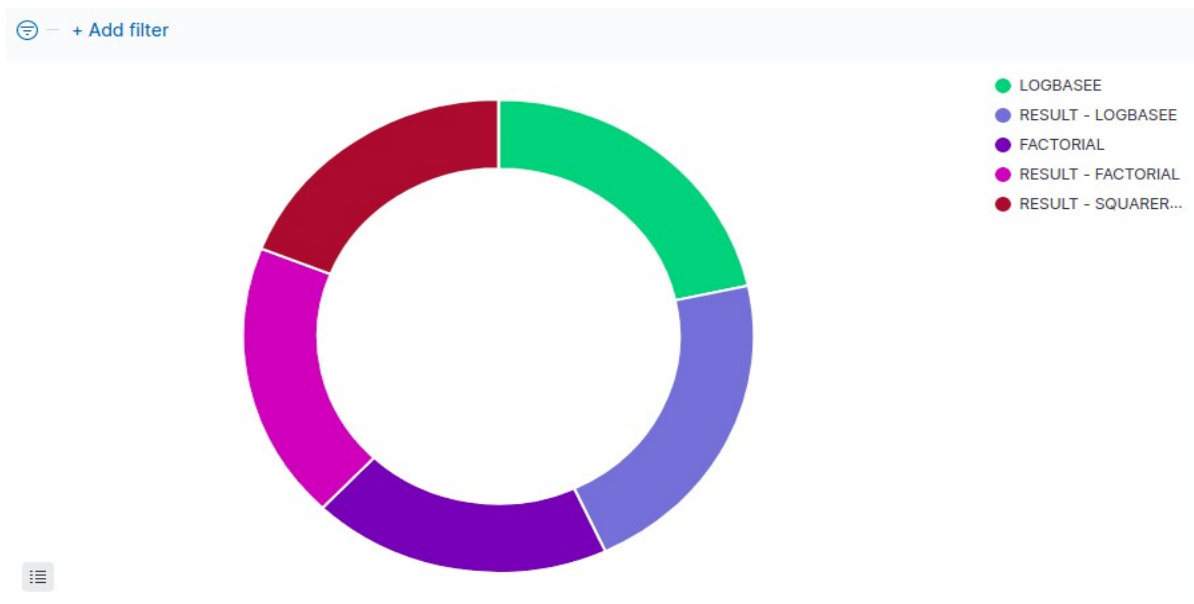
Terms

Field action.keyword

Order by Metric: Count

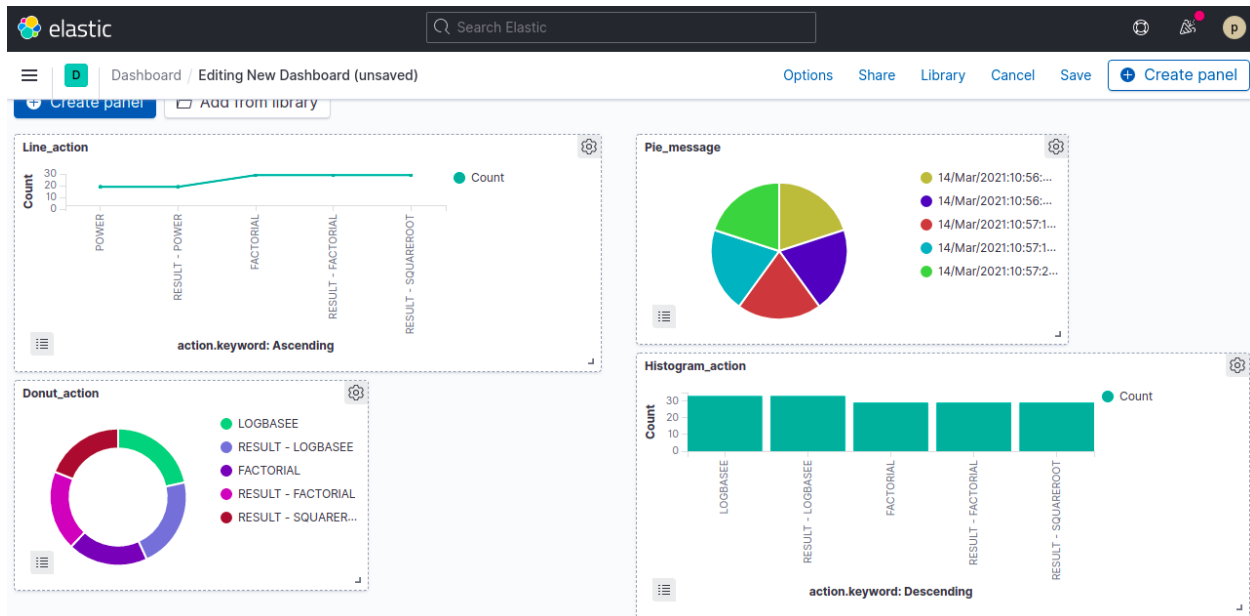
Order Size

Discard Update





Dashboard can also be created in kibana. First create views and multiple views can be seen on a single dashboard. Final view can look like



## 4. References

Following are the references that I have used to solve issues while developing this project:

- I. <https://stackoverflow.com/questions/12472645/how-do-i-schedule-jobs-in-jenkins>
- II. <https://stackoverflow.com/questions/34401165/log4j2-not-logging-to-file>
- III. <https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-shade-plugin/3.2.4>
- IV. <https://docs.docker.com/engine/reference/commandline/exec/>
- V. <https://git-scm.com/doc>
- VI. <https://www.elastic.co/cloud/>
- VII. All Lab documents