A REPORT ON

Calculator along with DevOps Toolchain

International Institute Of Information Technology, Bangalore

Computer Science and Engineering

SUBMITTED TO:

Prof. B.Thangaraju

SUBMITTED BY:

Shashank Agarwal (MT2019100)

1.0 Introduction :

We will use DevOps methodologies in order to facilitate continuous building, testing, deployment and monitoring. This is a JAVA based calculator project. This application provides user to add, subtract, divide or multiply two numbers.

2.0 Project Architecture

Language Used: -**JAVA**

IDE Used for JAVA: -**IntelliJ**

DevOps Tools Used:

GitHub: -SCM (Source Code Management)

Maven: -Build

Jenkins: - Continous Integration

Rundeck: - Continous Deployment

ELK: -Continuous Monitoring

3.0 Workflow:

3.1 Source Code Management (SCM):

GitHub is used for SCM (Source Code Management). It is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

GIT has various benefits as VCS(Version Control System) listed as follows :
1)Revert the code files back to their previous state.
2)Recall and revert the entire project back to its previous state.
3)Compare code changes over specific durations of time.
4)Find who last modified a piece of code that might be causing an issue or a problem. Who introduced a particular issue and when.

Advantages of using git:
• When multiple people collaborate on a project, it’s hard to keep track revisions who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository.

• Git branching model lets you have multiple local branches which are independent of each other. Having this also enables you to have friction-less context switching.

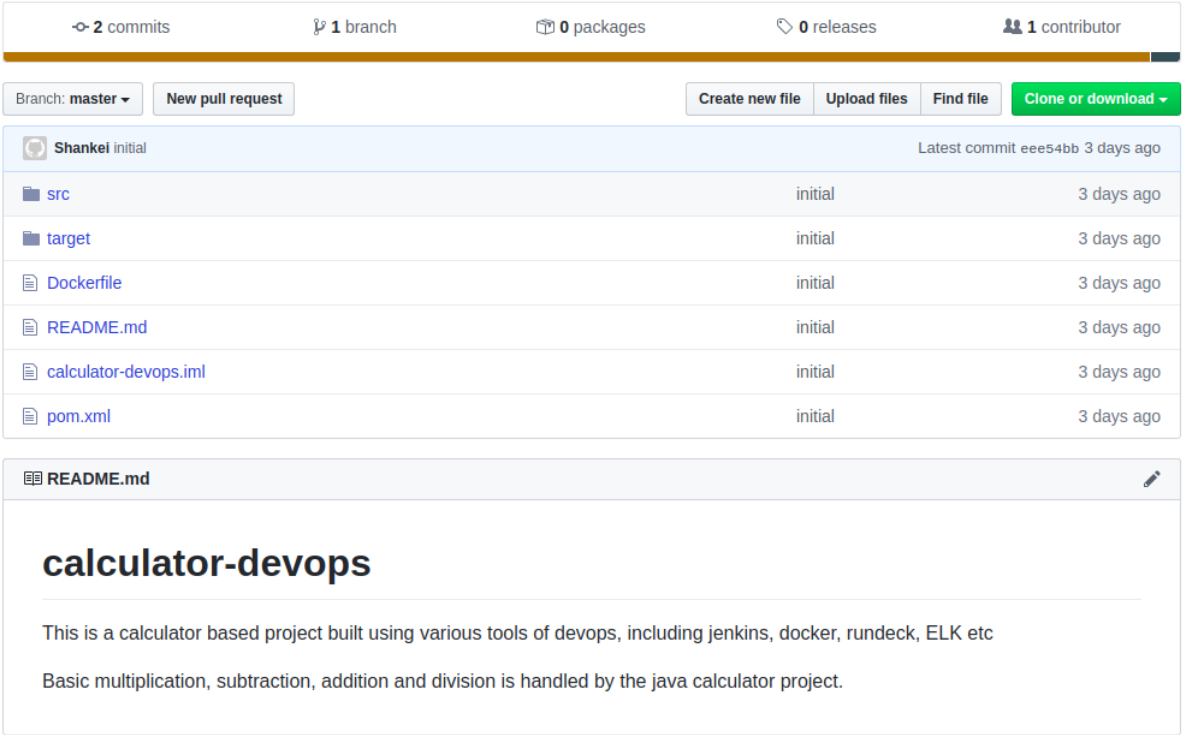
I pushed my whole project including Dockerfile to github.

URL of the GIT repository:

GitHub:https://github.com/shankei/calculator-devops.git

Steps used to add code to my git repository:

i) git add .
ii) git remote add origin *https://github.com/shankei/calculator-devops.git*
iii)git commit -m “initial”
iv)git push orrigin master



3.2 Build :

Maven is used with Jenkins for building the application as well as for dependency management. Maven made the dependency management task quite easy.
For the build step we integrated Maven with Jenkins so that whenever the build step is triggered Jenkins first pulls the updated code from the Github repo and then invokes maven to build the application.

Maven has the following advantages :

1. It makes a project easy to build.
2. It provides uniform build process (maven project can be shared by all the maven projects).
3. It provides project information (log document, cross referenced sources, mailing list, dependency list, unit test reports etc).
4. It is easy to migrate for new features of Maven.

Following is the pom.xml that we defined for our project.

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

```xml
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <dependency>
            <groupId>net.logstash.logback</groupId>
            <artifactId>logstash-logback-encoder</artifactId>
            <version>4.7</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
    </dependencies>
</project>
```

3.3 Continous Integration:

In our project, we have used ‘Jenkins’ for continuous integration. Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery.

Advantages of using Jenkins:

• Easily Configurable. Jenkins can be easily modified and extended. It deploys code instantly, generates test reports. Jenkins can be configured according to the requirements for continuous integrations and continuous delivery.

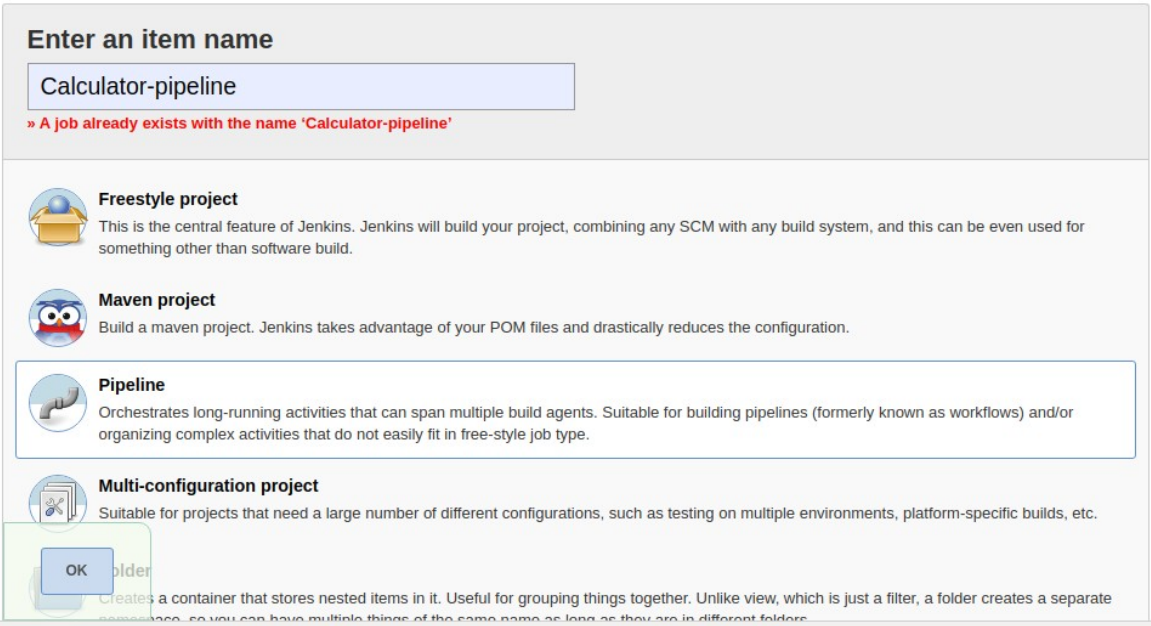
• Most of the integration work is automated. Hence fewer integration issues. This saves both time and money over the lifespan of a project.

Jenkins is a highly extensible product whose functionality can be extended through the installation of plugins like Maven, Github, Docker, Rundeck are installed in this project.

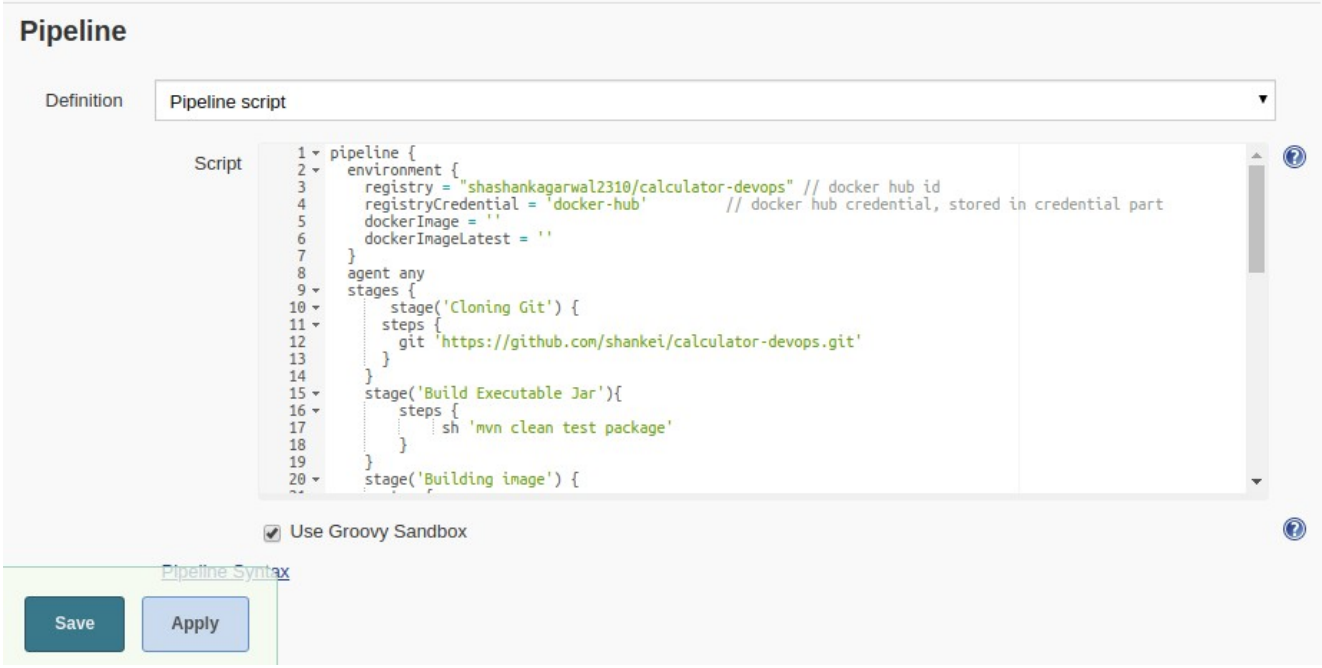
Jenkins Pipeline:

Jenkins pipeline consist of three stages.

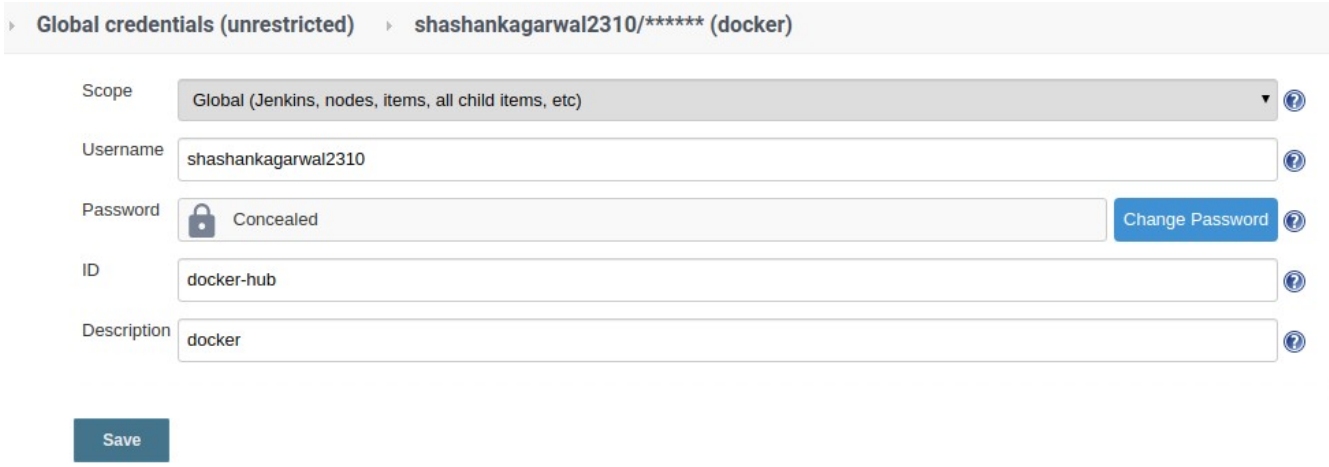
In first stage, we are polling the project from github and building the project using maven.



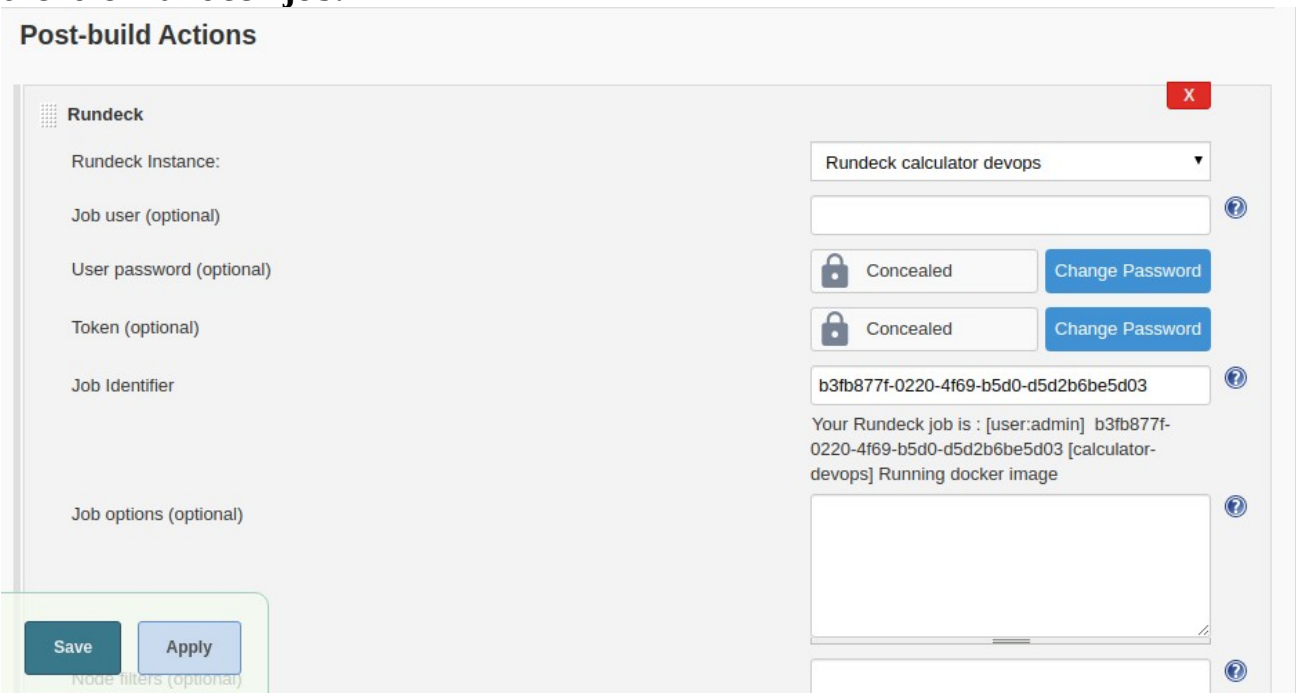
Write the script to run the pipeline.



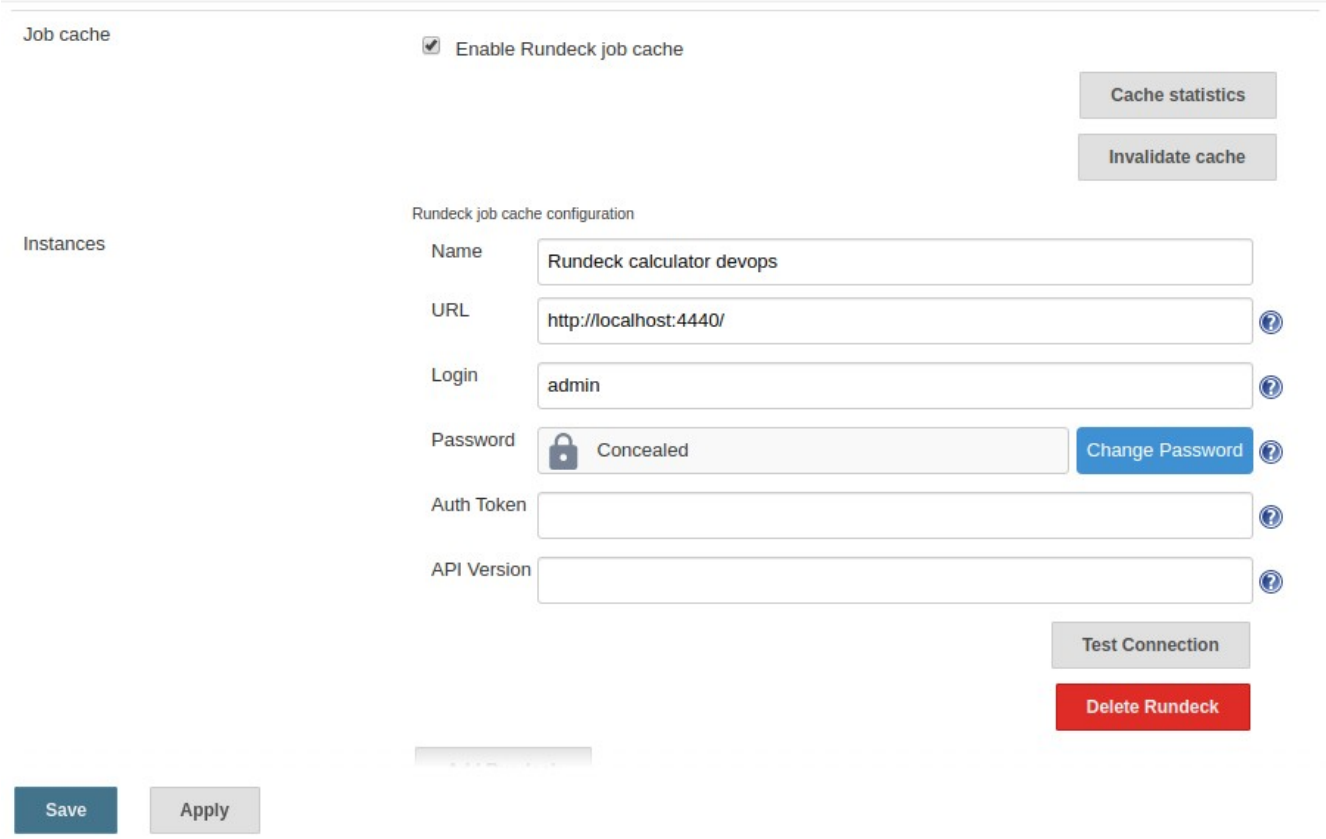
In second stage of the pipeline we are pushing the generated image to docker hub. Here, add docker hub credentials, i.e. username and password. In the "ID" field, the identity variable is the same used in the registryCredential above. i.e. docker-hub



In third stage of the pipeline we are connecting jenkins to rundeck and providing the id of rundeck job.



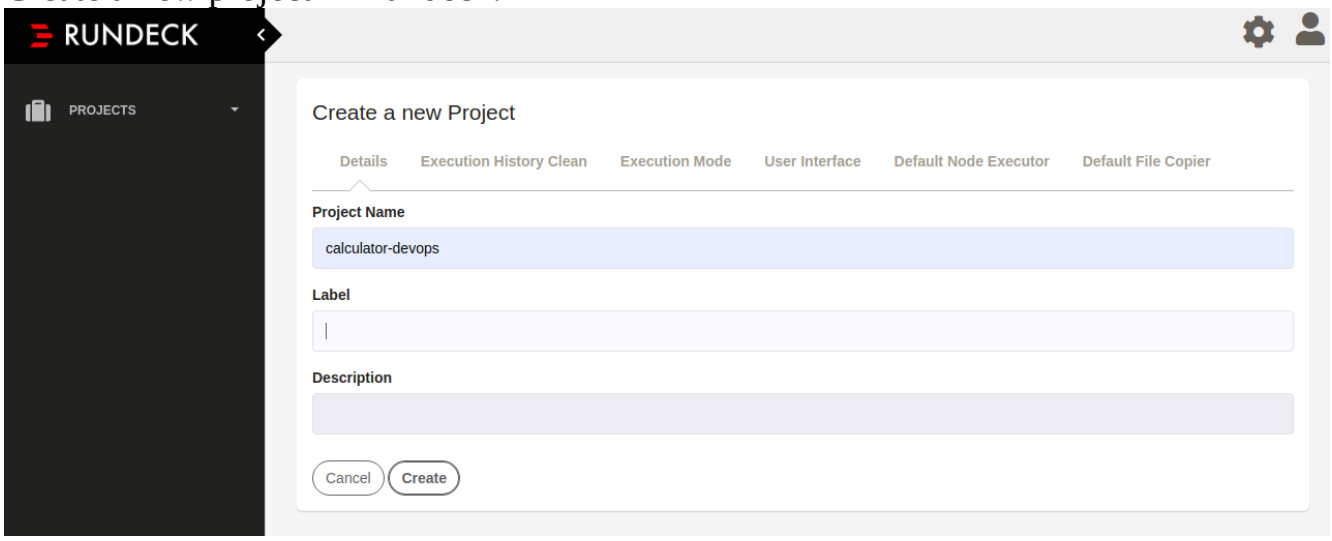
Create another free style project. In the Post-build Actions, the credentials will be picked directly from the configuration stored in Rundeck. Here, paste the job ID in the job identifier.



3.4 Continous Deployment

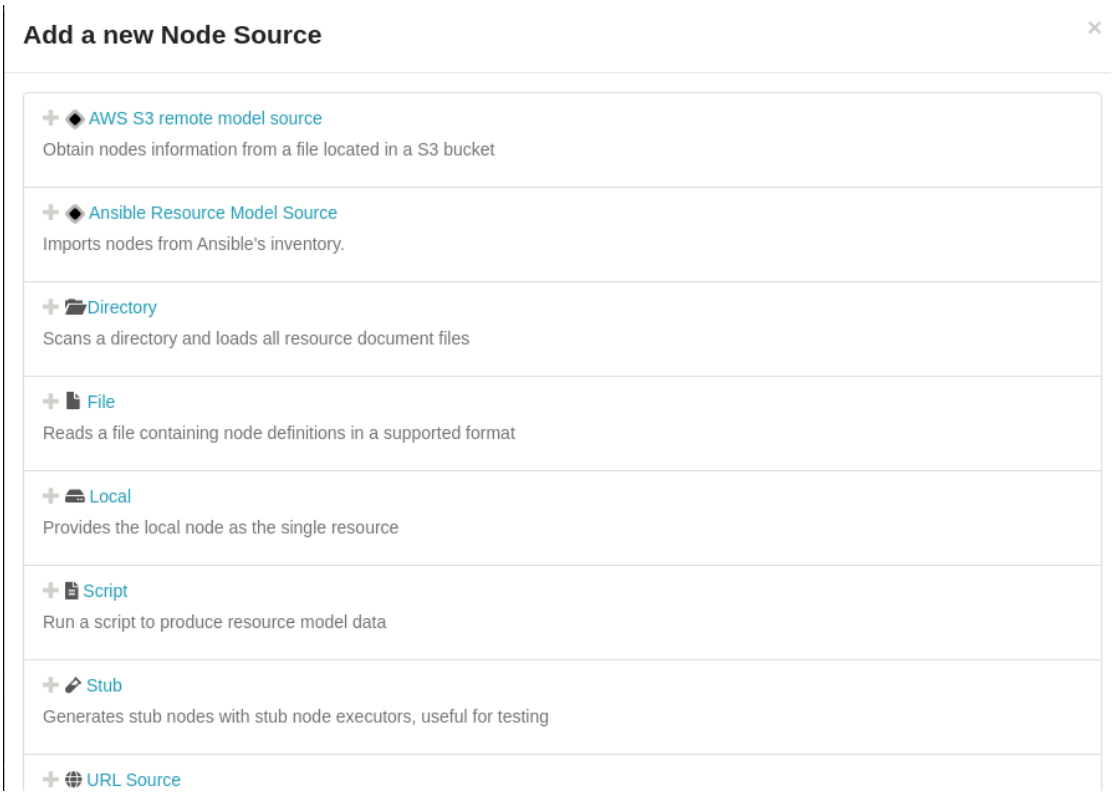
Rundeck is an orchestration tool for continuous deployment. It helps in standardising and automating complex release pipelines and deployment processes. It is an open source Web application written in Java and JavaScript. Rundeck source code is available on GitHub at https://github.com/rundeck. Many prominent multinational companies such as Walt Disney use Rundeck to carry out their operations. It is an easy-to-use tool that can be accessed via a Web interface, APIs or the command line.

Create a new project in Rundeck.

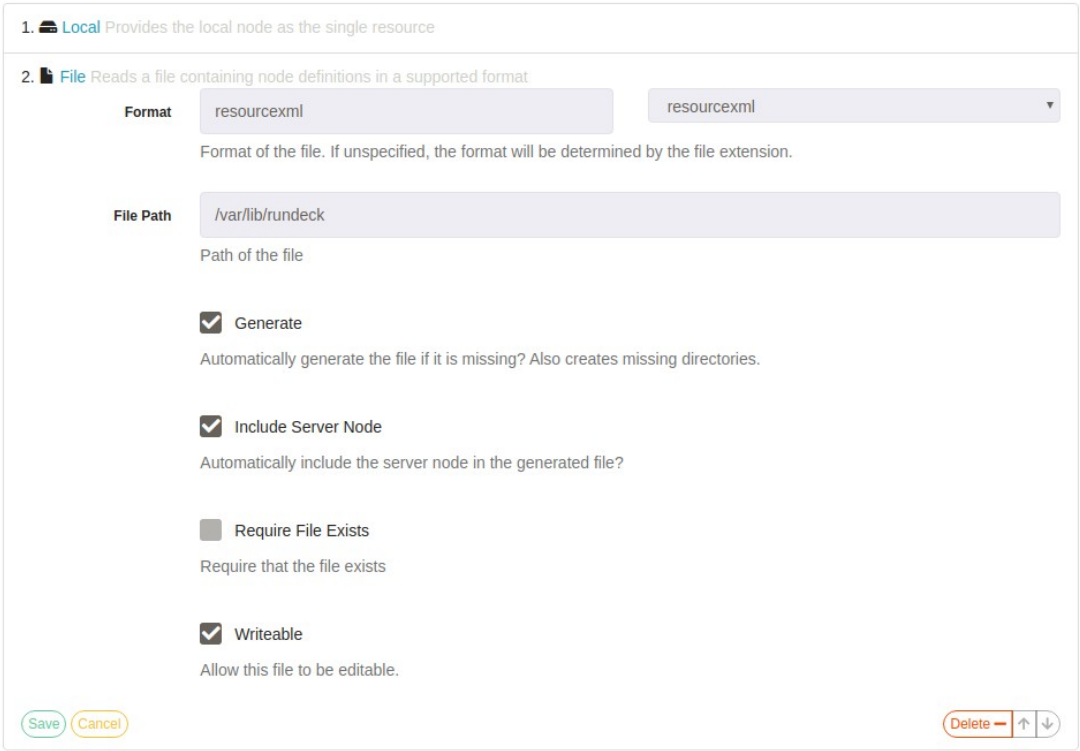


Click on add a new node.

Click on File to be added to create rundeck as a node.



Specify path of rundeck to create xml file as devops.xml



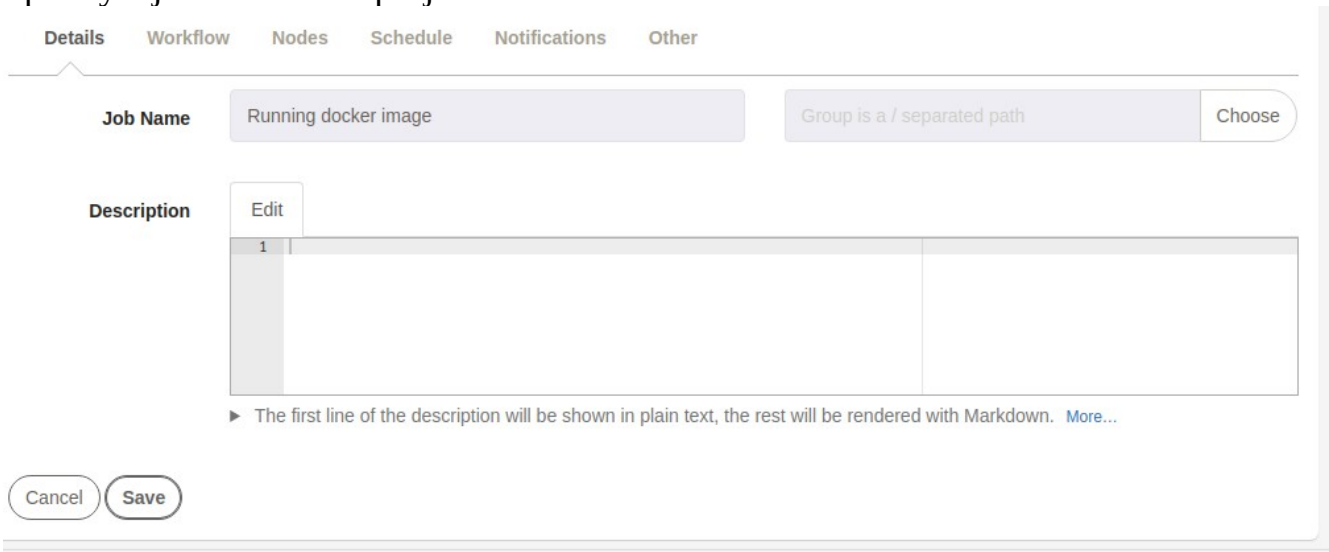
we can see devops.xml has been created.

```
root@shashank:/var/lib/rundeck# cat devops.xml
<?xml version="1.0" encoding="UTF-8"?>

<project>
  <node name="localhost" description="Rundeck server node" tags="" hostname="loc
alhost" osArch="amd64" osFamily="unix" osName="Linux" osVersion="5.3.0-42-generi
c" username="rundeck"/>
</project>
root@shashank:/var/lib/rundeck#
```

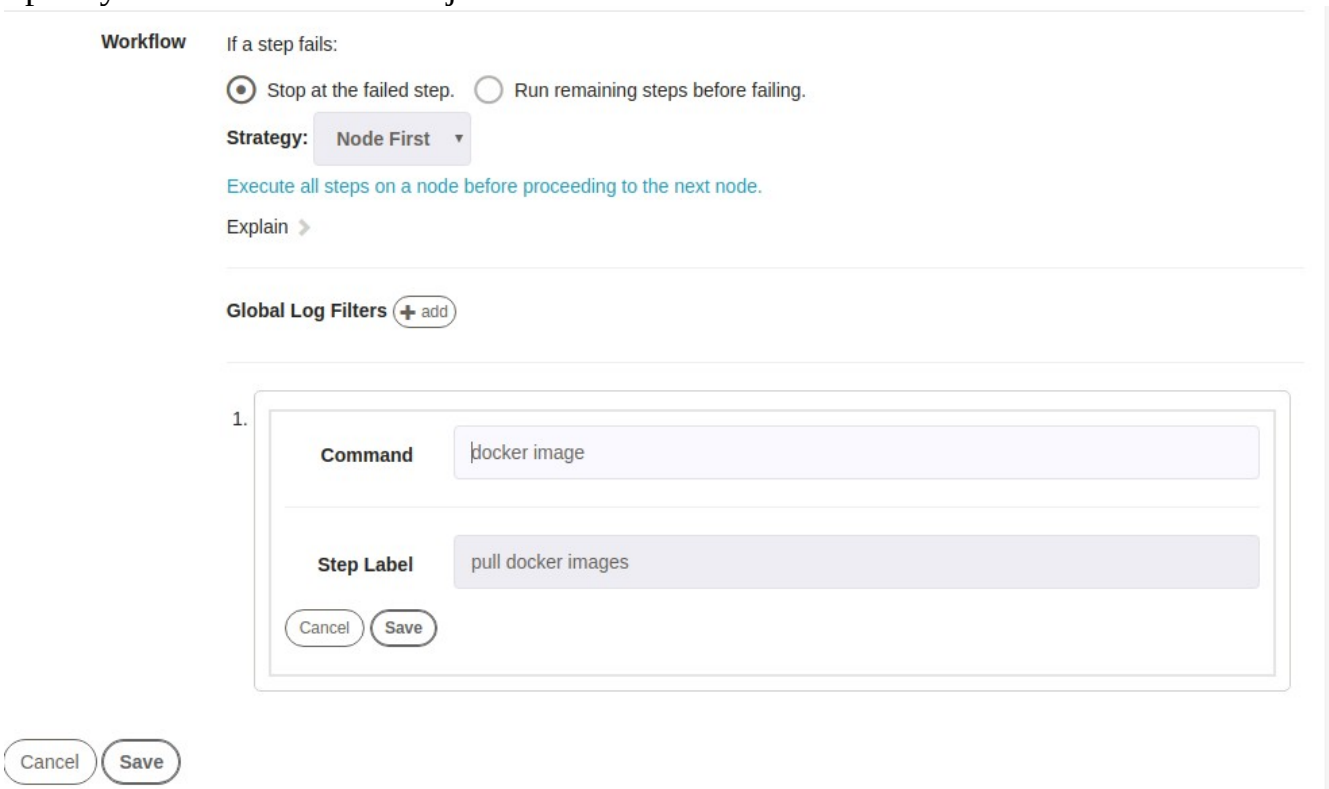
But here, if we need to run docker on Rundeck, we need to add Rundeck to the root group.

```
root@shashank:/var/lib/rundeck#
root@shashank:/var/lib/rundeck#  usermod -aG docker rundeck
root@shashank:/var/lib/rundeck# sudo grep rundeck /etc/gshadow
docker:!::rundeck,jenkins
rundeck:!::
root@shashank:/var/lib/rundeck#
```

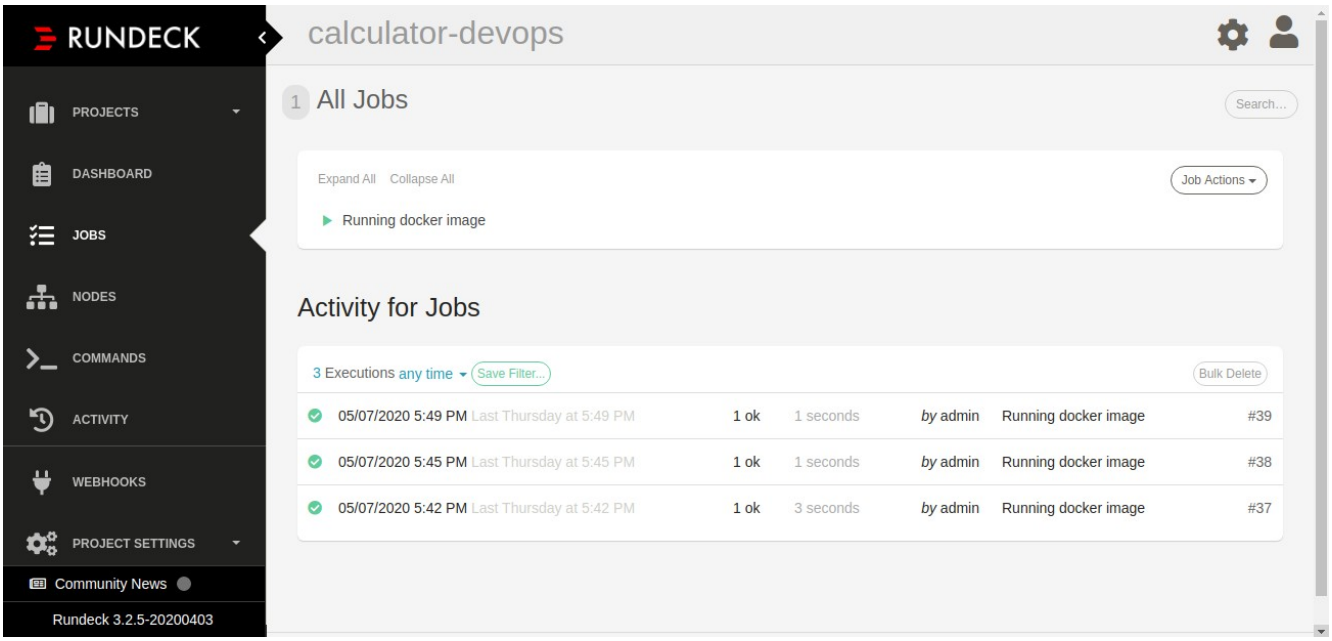
Specify a job in the new project in Rundeck.

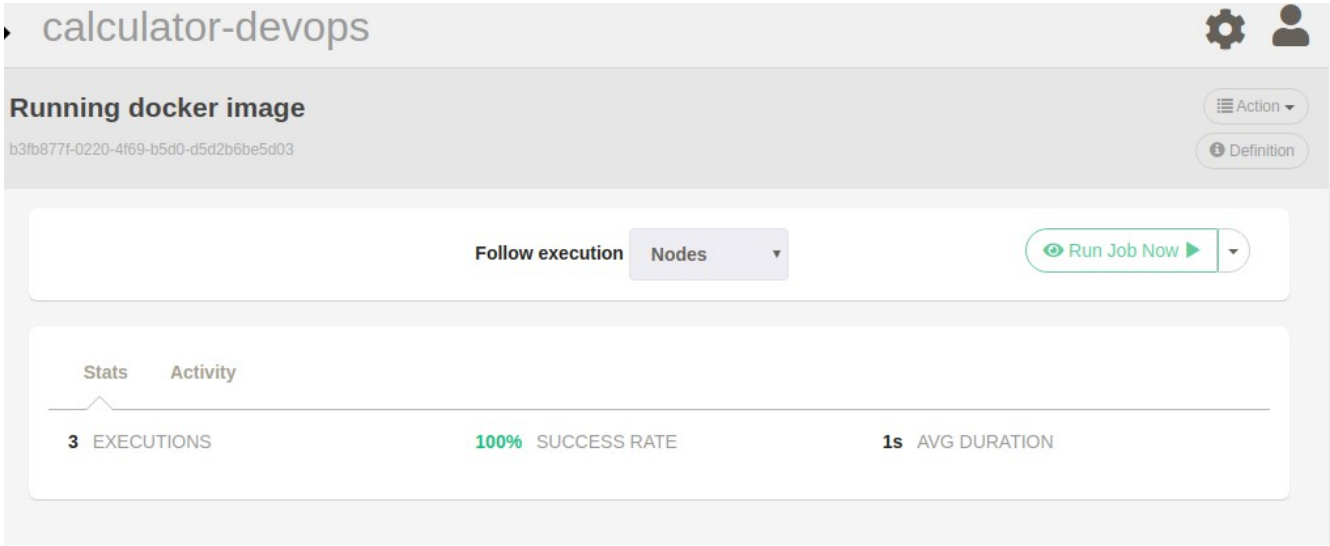


Specify the command for the job.

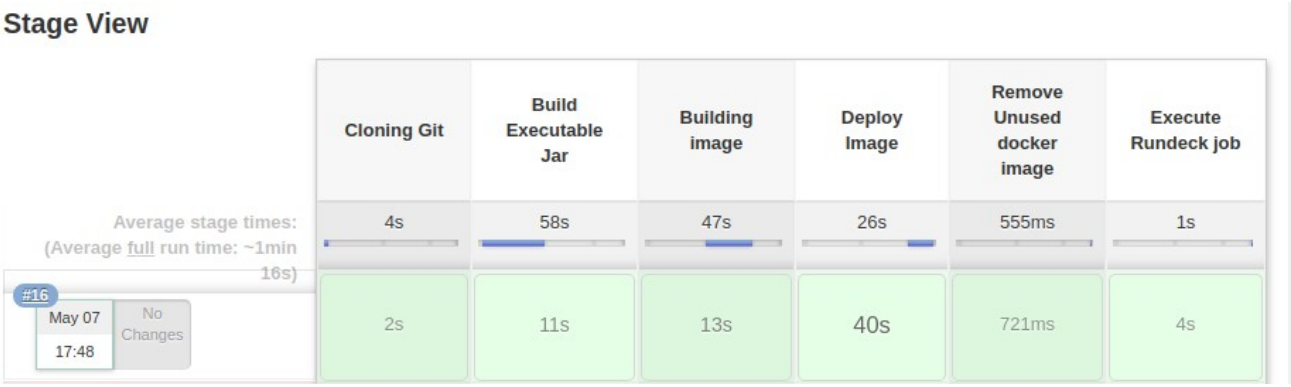


Job can be seen executed.





Pipeline in function



After following the steps, we can see the pipeline completely functional, following all the steps of a DevOps lifecycle.

```
root@shashank:/var/lib/rundeck# docker run -i -t shashankagarwal2310/calculator-devops
Calculator-DevOps, Choose to perform operation
Press 1 to Add
Press 2 to Subtract
Press 3 to Multiply
Press 4 to Divide
Press any other key to exit
Enter your choice:
1
Enter the first number : 5
Enter the second number : 4
[INFO ] 2020-05-10 13:48:29,001 method:calculator.Calculator.add(Calculator.java:63)
Adding two numbers 5.0 and 4.0
[INFO ] 2020-05-10 13:48:29,050 method:calculator.Calculator.add(Calculator.java:65)
Result of addition is 9.0
Addition result is : 9.0
Calculator-DevOps, Choose to perform operation
Press 1 to Add
Press 2 to Subtract
Press 3 to Multiply
Press 4 to Divide
Press any other key to exit
Enter your choice:
4
Enter the first number : 1
Enter the second number : 0
[INFO ] 2020-05-10 13:48:38,505 method:calculator.Calculator.divide(Calculator.java:87)
Dividing two numbers 1.0 and 0.0
[ERROR] 2020-05-10 13:48:38,507 method:calculator.Calculator.divide(Calculator.java:101)
Number cannot be divided by zero Case of Positive Infinity 1.0/0.0
[INFO ] 2020-05-10 13:48:38,508 method:calculator.Calculator.divide(Calculator.java:103)
Result of dividing is Infinity
Multiplication result is : Infinity
Calculator-DevOps, Choose to perform operation
Press 1 to Add
Press 2 to Subtract
Press 3 to Multiply
Press 4 to Divide
```

This image shows how we call the container in which we deployed the image and access the container

3.6 Continous Monitoring

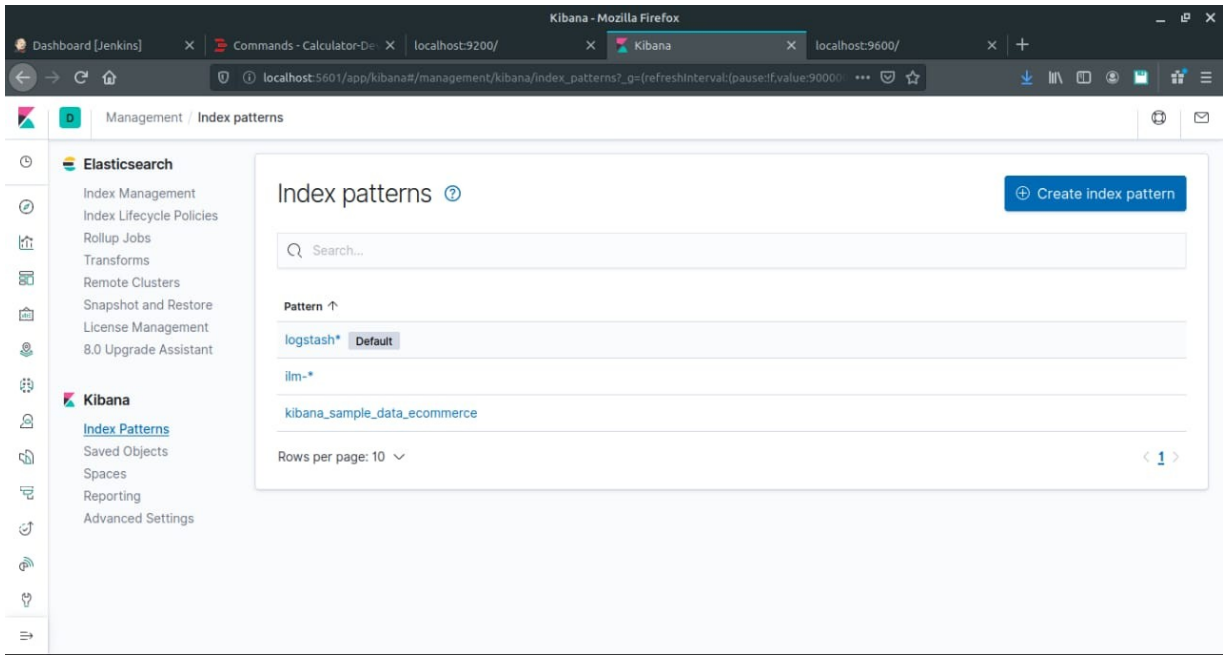
Continuous Monitoring is the formal process of defining an agency's IT systems, categorizing each of these systems by the level of risk, application of the controls, continuous monitoring of the applied controls, and the assessment of the effectiveness of these controls against security threats.

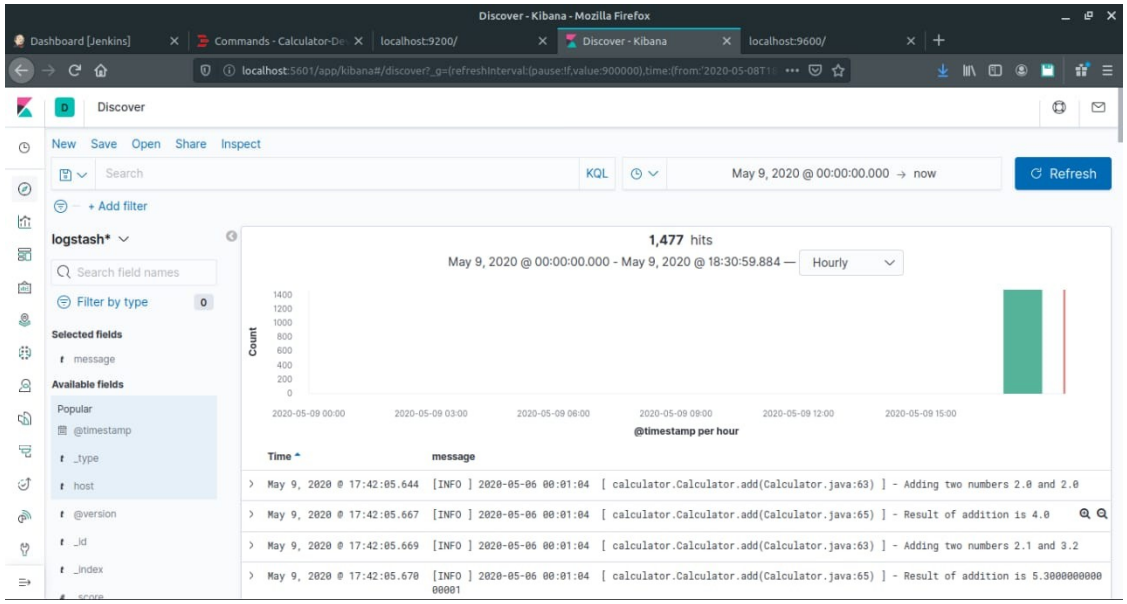
In our project, we have used 'ELK Stack' for continuous monitoring."ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch. Finally with the Kibana, we can generate visualisations using the log data.

```
input
{
    file {
        path => "/var/log/DevOps-Calculator/calculator.log"
        type => "logs"
        start_position => "beginning"
    }

}

output
{
    elasticsearch {
        hosts => ["localhost:9200"]
    }

    stdout {
        codec => dots
    }

}
```

Kibana Dashboard

In the management, toggle to Index Patterns under Kibana and create a new Create index pattern. Search for logstash-* and select the log file of the calculator program fetched by ElasticSearch. Follow to the next slide for the dashboard generated by Kibana.





Logs based on the @timestamp for the calculator program showing the INFO messages

4.0 References

[1]https://medium.com/

[2]https://www.youtube.com/

[3]https://www.docker.com/

[4]https://guides.github.com/

[5]https://stackoverflow.com/

[6]https://docs.docker.com/get-started/

[7]https://jenkins.io/doc/tutorials/

[8]https://maven.apache.org/guides/getting-started

[9]https://docs.rundeck.com/docs/tutorials/

[10]https://www.elastic.co/guide/en/elasticsearch/reference/current/gettingstarted.html