

SPE Mini Project Report

Scientific Calculator using DevOps Tools

Seethana Sreevardhan Reddy

IMT2018068

Table of Contents

Introduction	2
Tools Used	2
Part-1: Setting up Tools	2
Java (OpenJDK8)	2
Installation and Configuration on Ubuntu:	2
Maven	3
Installation on Ubuntu:	3
IDE (IntelliJ IDEA)	3
Installation and Configuration on Ubuntu:	3
Git	4
Installation and Configuration on Ubuntu:	4
GitHub	4
Configuration:	4
Jenkins	5
Installation and Configuration on Ubuntu:	5
Docker (Containerization)	6
Installation and Configuration on Ubuntu:	6
Rundeck	7
Installation and Configuration on Ubuntu:	7
Docker Hub	8
Configuration:	8
ELK Stack	8
Installation and Configuration on Ubuntu:	8
Google Cloud	9
Configuration:	9
Part-2: Developing the project (CI/CD)	11
Process	11
Scientific Calculator	12
IntelliJ IDEA	12
Jenkins Pipeline	16
Rundeck	18
Google Cloud (Deployment Server)	20
FINAL STEP	21
Part-3: Continuous Monitoring	22
ELK Stack	22
Links	26
GitHub	26
DockerHub	26
References	26

Introduction

The main goal of this project is to develop a scientific calculator program using DevOps tools. The aim is to learn, understand and implement Continuous Integration, Continuous Deployment and Continuous Monitoring using DevOps tools.

Tools Used

1. Java (OpenJDK8)
2. Git
3. GitHub
4. Maven
5. Junit
6. Log4j
7. IntelliJ IDEA
8. Docker
9. Docker Hub
10. Jenkins
11. Rundeck
12. Google cloud
13. ELK Stack

Part-1: Setting up Tools

Java (OpenJDK8)

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

Installation and Configuration on Ubuntu:

1. `$ sudo apt update`
2. `$ sudo apt install openjdk-8-jdk`
3. To configure `$ JAVA_HOME` path
 - a. `$ sudo vim /etc/environment`
 - b. Add this path at the end of the file:
 - i. `AVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/bin"`
 - c. Restart the system to apply changes
 - d. `$ source /etc/environment`
 - e. Verify `JAVA_HOME` path
 - i. `$ echo $JAVA_HOME`

Maven

Maven is a project development management and development tool. Maven can be extended by plugins to utilize a number of other development tools for reporting or the build process.

Installation on Ubuntu:

1. \$ sudo apt install maven
2. \$ maven --version

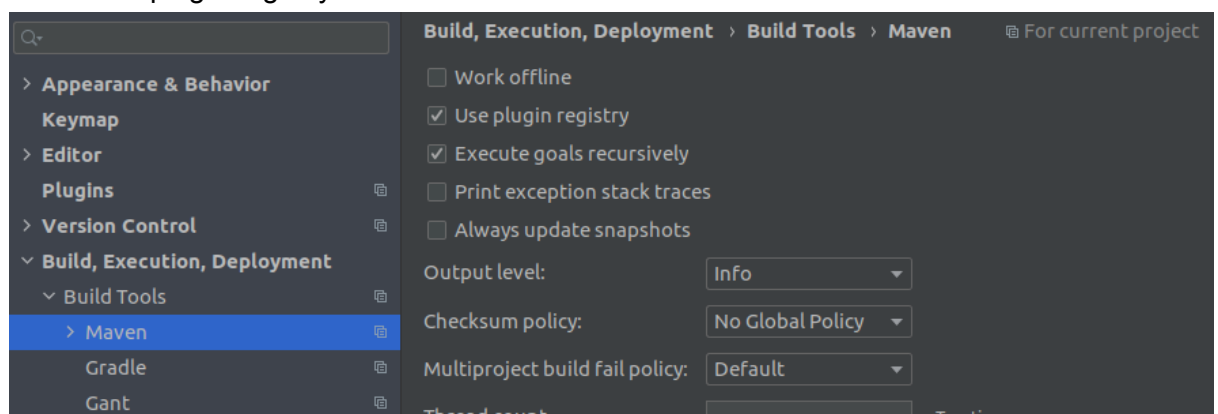
```
snowman@Snowpc:~$ mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 1.8.0_282, vendor: Private Build, runtime: /usr/lib/jvm/java-8-openjdk-amd64/jre
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "5.8.0-45-generic", arch: "amd64", family: "unix"
```

IDE (IntelliJ IDEA)

IntelliJ IDEA is an integrated development environment written in java for developing software.

Installation and Configuration on Ubuntu:

1. Download IntelliJ IDEA from <https://www.jetbrains.com/idea/download>
2. Extract it and go to /bin directory
3. To start the IDE
 - a. \$./idea.sh
4. Go to settings/build, executions, deployment/build tools/maven
5. Select use plugin registry then click on save.



Git

Git is a distributed version control system, it is a tool to manage project source code history. Git is one of the most widely-used popular version control systems in use today.

Installation and Configuration on Ubuntu:

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

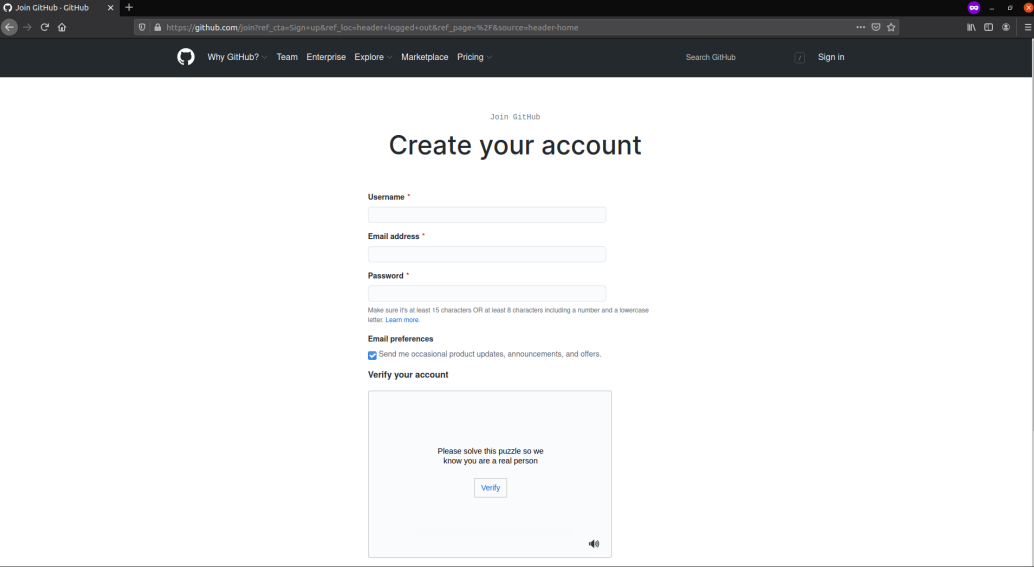
```
snowman@Snowpc:~$ git --version
git version 2.27.0
snowman@Snowpc:~$ |
```

GitHub

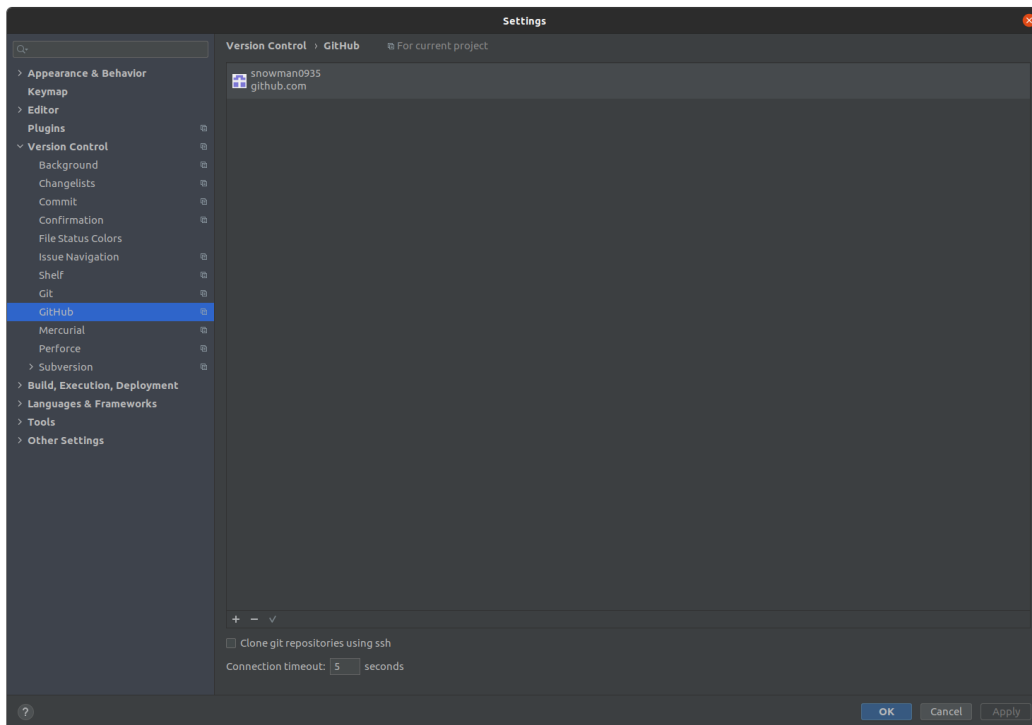
Github is a provider of internet hosting for software development and version control using Git. It offers distributed version control and source code management functionality of Git, plus its own feature.

Configuration:

1. Signup for github at <https://github.com/join>
2. Create a new repository
 - a. Enter repo name and click create
3. In IntelliJ go to setting/Version Control/GitHub and click +
4. Log via Github or token



The screenshot shows the GitHub 'Create your account' page in a web browser. The page has a dark header with navigation links like 'Why GitHub?', 'Team', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. Below the header, the main content area is white and titled 'Create your account'. It contains a form with fields for 'Username', 'Email address', and 'Password'. Below the password field, there is a note about password requirements and a link to 'Learn more'. There is also a checkbox for 'Email preferences' and a 'Verify your account' section with a puzzle image and a 'Verify' button.

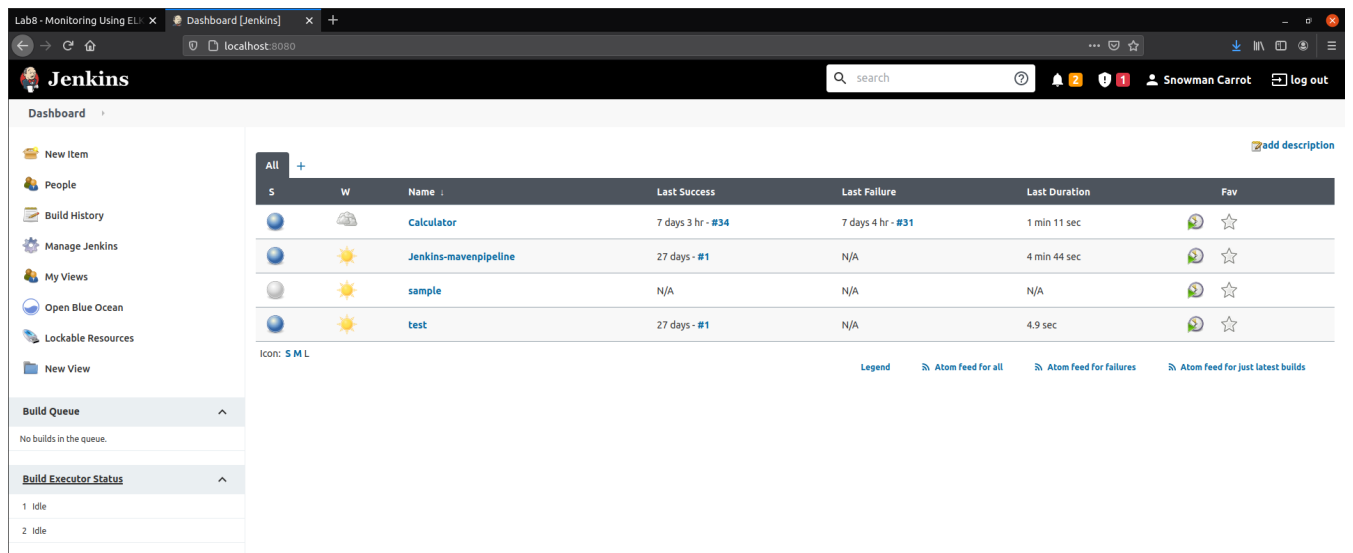


Jenkins

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration. Jenkins is used to build and test software projects. It allows developers to continuously deliver software by integrating with a large number of testing and deployment technologies.

Installation and Configuration on Ubuntu:

1. Add Key
 - a. `$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | apt-key add -`
2. Add the repository, update local package index and install
 - a. `$ sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`
 - b. `$ sudo apt update`
 - c. `$ sudo apt install jenkins`
3. Start jenkins
 - a. `$ sudo service jenkins start`
 - b. `$ sudo service jenkins status`
4. Jenkins runs on the localhost at port 8080 by default.
5. Open localhost:8080 in a browser.
6. Print default password to login into jenkins
 - a. `$ cat /var/lib/jenkins/secrets/initialAdminPassword`
7. Choose install suggested plugins, configure user.



Docker (Containerization)

Docker is an open source platform for developing, shipping and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host.

Installation and Configuration on Ubuntu:

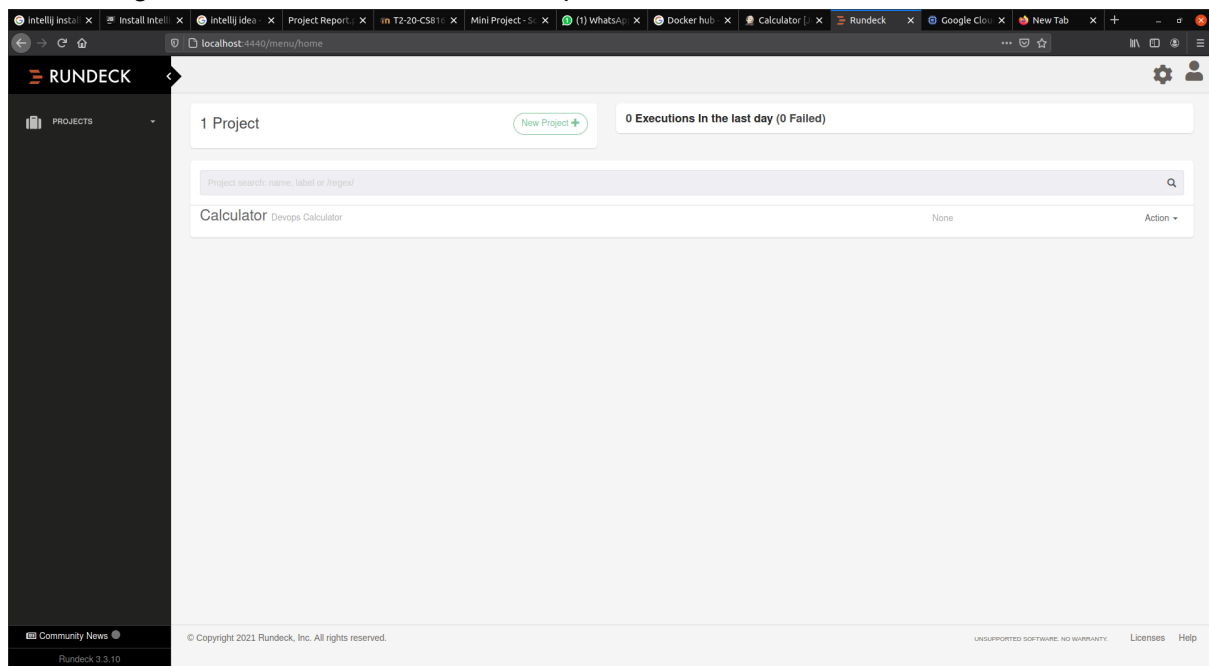
1. \$ sudo apt update
2. \$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
3. \$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
4. \$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
5. \$ sudo apt install docker-ce
6. \$ sudo service docker status
7. Execute docker without sudo (Very important)
 - a. \$ sudo groupadd docker
 - b. \$ sudo usermod -aG docker \${USER}
 - c. \$ newgrp docker
 - d. Restart the OS
 - e. Check if docker runs without sudo
 - i. \$ docker run hello-world

Rundeck

Rundeck is an open source automation service with a web console, command line tools and a WebAPI. Rundeck allows you to run tasks on any number of nodes from a web-based or CLI. Rundeck also includes other features that make it easy to scale up your automation efforts including: access control, workflow building, scheduling, logging and integration with external sources for node and optional data.

Installation and Configuration on Ubuntu:

1. `$ echo "deb https://rundeck.bintray.com/rundeck-deb /" | sudo tee -a /etc/apt/sources.list.d/rundeck.list`
2. `$ curl 'https://bintray.com/user/downloadSubjectPublicKey?username=bintray' | sudo apt-key add -`
3. `$ apt update`
4. `$ apt install rundeck`
5. Start Rundeck
 - a. `$ service rundeckd start`
6. Verify if Rundeck is running correctly
 - a. `$ tail -f /var/log/rundeck/service.log`
7. Open <http://localhost:4440> in a browser.
8. Login with admin as username and password



Docker Hub

Docker Hub is like Github but for Docker Images. Here, we can find official images created by companies as well as customized images from different users. We can create our own repositories.

Configuration:

1. Signup at <https://hub.docker.com> and create a repository

The screenshot shows the Docker Hub 'Create Repository' page. The page has a blue header with the Docker Hub logo and navigation links. The main content area is white and contains the 'Create Repository' form. The form has a dropdown menu for the repository name (currently showing 'snowman0935'), a text input for the description, and a section for visibility. The visibility section has two radio buttons: 'Public' (selected) and 'Private'. Below the visibility section is a 'Build Settings' section with a 'Connected' status indicator. At the bottom of the form are three buttons: 'Cancel', 'Create', and 'Create & Build'.

ELK Stack

ELK consists of three open source softwares:

1. Elasticsearch: It is a search and analytics engine
2. Logstash: It 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
3. Kibana: It lets users visualize data with charts and graphs made from elasticsearch

Installation and Configuration on Ubuntu:

1. Download and extract Elasticsearch, Logstash and kibana from the following links
 - a. <https://www.elastic.co/downloads/elasticsearch>
 - b. <https://www.elastic.co/downloads/kibana>
 - c. <https://www.elastic.co/downloads/logstash>
 - d. <https://www.elastic.co/beats/> (Optional)

Google Cloud

Google Cloud Platform is a suite of cloud computing services. We create a VM on the cloud using this service to deploy our software.

Configuration:

1. Signup on <https://cloud.google.com/>
2. Go to Dashboard, then click on Compute Engine in the menu.
3. Click on VM Instances and then CREATE INSTANCE.
4. Name your Instance.
5. Select Ubuntu 20.04 LTS minimal Boot Disk.
6. Select allow http traffic.
7. Click on create.
8. Login to the instance by clicking on SSH in the VM instances section.
9. Now install Docker using the same installation procedure as before.

The screenshot displays the 'Create an instance' interface in the Google Cloud Platform console. On the left, a sidebar offers four options: 'New VM instance' (from scratch), 'New VM instance from template', 'New VM instance from machine image', and 'Marketplace'. The main area is titled 'Create an instance' and contains several configuration sections:

- Name:** A text field with 'instance-1' entered.
- Labels:** A section with an 'Add label' button.
- Region and Zone:** 'Region' is set to 'us-central1 (Iowa)' and 'Zone' is set to 'us-central1-a'.
- Machine configuration:** Includes tabs for 'General-purpose', 'Compute-optimised', 'Memory-optimised', and 'GPU'. The 'Series' is 'E2' and the 'Machine type' is 'e2-medium (2 vCPU, 4 GB memory)'. A summary shows '1 shared core', '4 GB' memory, and '0' GPUs.
- Confidential VM service:** A checkbox to 'Enable the Confidential Computing service on this VM instance' is unchecked.
- Container:** A checkbox to 'Deploy a container image to this VM instance' is unchecked.
- Boot disk:** A 'New 10 GB balanced persistent disk' is selected with the 'Image' set to 'Ubuntu 20.04 LTS Minimal'.
- Identity and API access:** The 'Service account' is 'Compute Engine default service account'. Under 'Access scopes', 'Allow default access' is selected.
- Firewall:** Under 'Add tags and firewall rules to allow specific network traffic from the Internet', 'Allow HTTP traffic' is selected.

At the bottom, a note states 'Your free trial credit will be used for this VM instance.' followed by 'GCP Free Tier' link. 'Create' and 'Cancel' buttons are at the bottom left, and a link for 'Equivalent REST or command line' is at the bottom right.

```
sreevardhan73@ubuntuserver: ~ — Mozilla Firefox
https://ssh.cloud.google.com/projects/sturdy-mode-308200/zones/us-central1-a/instances/ubuntusei ...
Connected, host fingerprint: ssh-rsa 0 D0:9D:BA:CE:98:7A:B5:FE:E2:42:53:F0:AE:06
:10:DB:47:B6:35:AC:76:3D:4A:C9:FB:03:EC:CF:BA:4A:DE:94
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1037-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Sat Mar 20 03:43:06 2021 from 35.235.241.51
sreevardhan73@ubuntuserver:~$
```

Part-2: Developing the project (CI/CD)

Now that all the tools are set up, we can start developing our project.

Process

1. Before we start working on the scientific calculator we have to integrate all our tools. In part-1 we have installed and set up all our tools. Now we have to integrate them.
2. First we have to set up a Jenkins pipeline, it has to pull our project from our github, run it, test it, build it, create a docker file, push it to docker hub and trigger a rundeck job.
3. Next, we create a rundeck job to deploy the docker image from docker hub onto the deployment server.
4. Now we are going to use IntelliJ to develop a scientific calculator in java. IntelliJ is integrated with various other tools and utilities such as git, github, maven, log4j, junit.
5. IntelliJ creates a file system for our project initially and we can start adding files to our project at the appropriate places.
 - a. The Source code i.e the java code, junit and log4j files in src
 - b. In the project folder we keep pom.xml, Dockerfile, yml files and other files needed for other tools.
6. Every time we update any part of the project we make a commit in the local git.
7. Once a stable version is developed we push the local repository code to our github.
8. Once the code is pushed a Jenkins pipeline is triggered which starts executing a pre-defined pipeline.
9. In our pipeline,
 - a. First Jenkins pulls the new code from github.
 - b. It will then compile, run and test the whole project using maven.
 - c. If the tests are successful it will then build the project and create a docker image based on the Dockerfile.
 - d. After building a docker image successfully it will then push the image to docker hub and then triggers a Rundeck job.
10. The Rundeck job we defined pulls the new docker image onto the deployment server/s and runs them.
11. Now, the deployment servers are ready for use and they start creating logs as they are being used.
12. We can pull all the logfiles everyday to the Continuous monitoring machine and use ELK stack to visualize all the log files. To monitor logs in real time we can use beats to transfer all the log files in real time to the logging/monitoring server to monitor in real time.

Scientific Calculator

The Scientific calculator has the following functions:

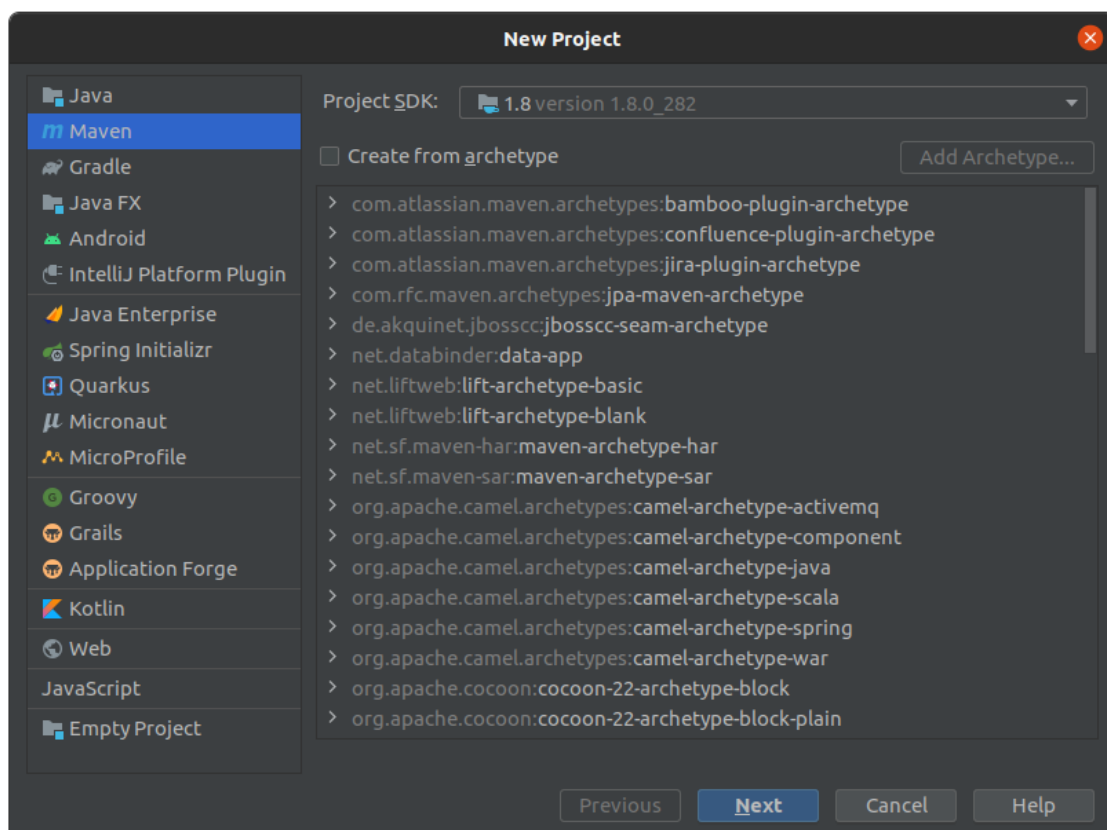
1. Square Root of x
2. Factorial of x
3. Natural log of x
4. x power b

The scientific calculator can be accessed through CLI only.

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Calcuator
Available functions
1 - Square Root of x
2 - Factorial of x
3 - Natural Log of x
4 - x Power b
Any other number to exit
```

IntelliJ IDEA

1. Setting up a new project
 - a. Open IntelliJ and select a new project.
 - b. Select maven from the menu.
 - c. Select project sdk 1.8 and click next.
 - d. Give a name and click Finish.



2. IntelliJ creates a file structure for our project.
3. Go to src/main. Right click on java and select new/package.
4. Write the scientific calculator code here.
5. Next, in src/main/resources create a log4j2.xml file and write the following code. This file creates a template for all the log files.

```

1  <Configuration status="INFO">
2      <Appenders>
3          <Console name="ConsoleAppender" target="SYSTEM_OUT">
4              <PatternLayout pattern="%d{dd/MM/yyyy:HH:mm:ss SSS} [%F] [%level] %logger{36} %msg%n"/>
5          </Console>
6          <File name="FileAppender" filename="calculator.log" immediateFlush="false" append="true">
7              <PatternLayout pattern="%d{dd/MM/yyyy:HH:mm:ss SSS} [%F] [%level] %logger{36} %msg%n"/>
8          </File>
9      </Appenders>
10     <Loggers>
11         <Root level="debug">
12             <AppenderRef ref="ConsoleAppender"/>
13             <AppenderRef ref="FileAppender"/>
14         </Root>
15     </Loggers>
16 </Configuration>

```

6. Next, in src/java create a new java file to write all the tests.

```

1  import calculator.calculator;
2  import static org.junit.Assert.*;
3  import org.junit.Test;
4
5  public class CalculatorTest {
6      private static final double DELTA = 1e-15;
7      calculator calc = new calculator();
8
9      @Test
10     public void squarerootTruePositive(){
11         assertEquals( message: "Square root of an int - True Positive", expected: 2, calc.sqrt( 4), DELTA);
12         assertEquals( message: "Square root of a double- True Positive", expected: 1.8708286933869707, calc.sqrt( 3.5), DELTA);
13     }
14
15     @Test
16     public void squarerootFalsePositive(){
17         assertEquals( message: "Square root of an int - False Positive", unexpected: 2, calc.sqrt( 5), DELTA);
18         assertEquals( message: "Square root of a Double- False Positive", unexpected: 5.7, calc.sqrt( 30), DELTA);
19     }
20
21     @Test
22     public void factorialTruePositive(){
23         assertEquals( message: "Factorial of an int - True Positive", expected: 2, calc.factorial( 2), DELTA);
24         assertEquals( message: "Factorial of a double - True Positive", expected: 24, calc.factorial( 4.0), DELTA);
25     }
26
27     @Test
28     public void factorialFalsePositive(){
29         assertEquals( message: "Factorial of an int - False Positive", unexpected: 120, calc.factorial( 6), DELTA);
30         assertEquals( message: "Factorial of a double - False Positive", unexpected: 720, calc.factorial( 4.0), DELTA);
31     }
32
33     @Test
34     public void logTruePositive(){
35         assertEquals( message: "Log- True Positive", expected: 1.6094379124341003, calc.log( 5), DELTA );
36         assertEquals( message: "Log - TruePositive", expected: 4.04305126783455, calc.log( 57), DELTA );
37         assertEquals( message: "Log - True Positive", expected: 0, calc.log( 1), DELTA );
38     }
39 }

```

7. Next, in the project folder create the following files:
 - a. Pom.xml
 - i. Dependencies can be easily added by the generate tool (Right click).
 - ii. All the necessary plugins and dependencies of the java project have to be added here. Maven reads this file and installs all the listed dependencies and plugins.

- iii. We add a maven assembly plugin to create a jar file. Which can be used to easily execute our project on any computer.

```
calculator.java x Dockerfile x remote-server x pom.xml (Calculator) x Calculator.iml x logstash.conf x .gitignore x
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-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>org.example</groupId>
8   <artifactId>Calculator</artifactId>
9   <version>1.0-SNAPSHOT</version>
10  <build>
11    <plugins>
12      <plugin>
13        <groupId>org.apache.maven.plugins</groupId>
14        <artifactId>maven-assembly-plugin</artifactId>
15        <executions>
16          <execution>
17            <phase>package</phase>
18            <goals>
19              <goal>single</goal>
20            </goals>
21            <configuration>
22              <archive>
23                <manifest>
24                  <mainClass>calculator.calculator</mainClass>
25                </manifest>
26              </archive>
27              <descriptorRefs>
28                <descriptorRef>jar-with-dependencies</descriptorRef>
29              </descriptorRefs>
30            </configuration>
31          </execution>
32        </executions>
33      </plugin>
34    </plugins>
35  </build>
36  <dependencies>
37    <dependency>
38      <groupId>org.apache.logging.log4j</groupId>
39      <artifactId>log4j-api</artifactId>
40      <version>2.14.0</version>
41    </dependency>
42    <dependency>
43      <groupId>org.apache.logging.log4j</groupId>
44      <artifactId>log4j-core</artifactId>
45      <version>2.14.0</version>
46    </dependency>
47    <dependency>
48      <groupId>junit</groupId>
49      <artifactId>junit</artifactId>
50      <version>4.12</version>
51    </dependency>
52  </dependencies>
53
54  <properties>
55    <maven.compiler.source>8</maven.compiler.source>
56    <maven.compiler.target>8</maven.compiler.target>
57  </properties>
58
59
60 </project>
```

b. Dockerfile

- i. We create a dockerfile to let docker know how to build an image.
- ii. Here we use an openjdk base image and once the image runs it will copy the calculator jar file and execute it.

```
calculator.java x Dockerfile x remote-server x pom.xml (Calculator) x Calculator
1 FROM openjdk:8
2 MAINTAINER Sreevardhan seethana.sreevardhan@iiitb.org
3 COPY ./target/Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar ./
4 WORKDIR ./
5 CMD ["java", "-jar", "Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

c. Logstash.conf

- i. This is a configuration file for logstash.

```
1 input{
2   plugin {
3     settings
4   }
5 }
6
7 filter {
8   plugin {
9     settings
10  }
11 }
12
13 output {
14   plugin {
15     settings
16   }
17 }
18
19
```

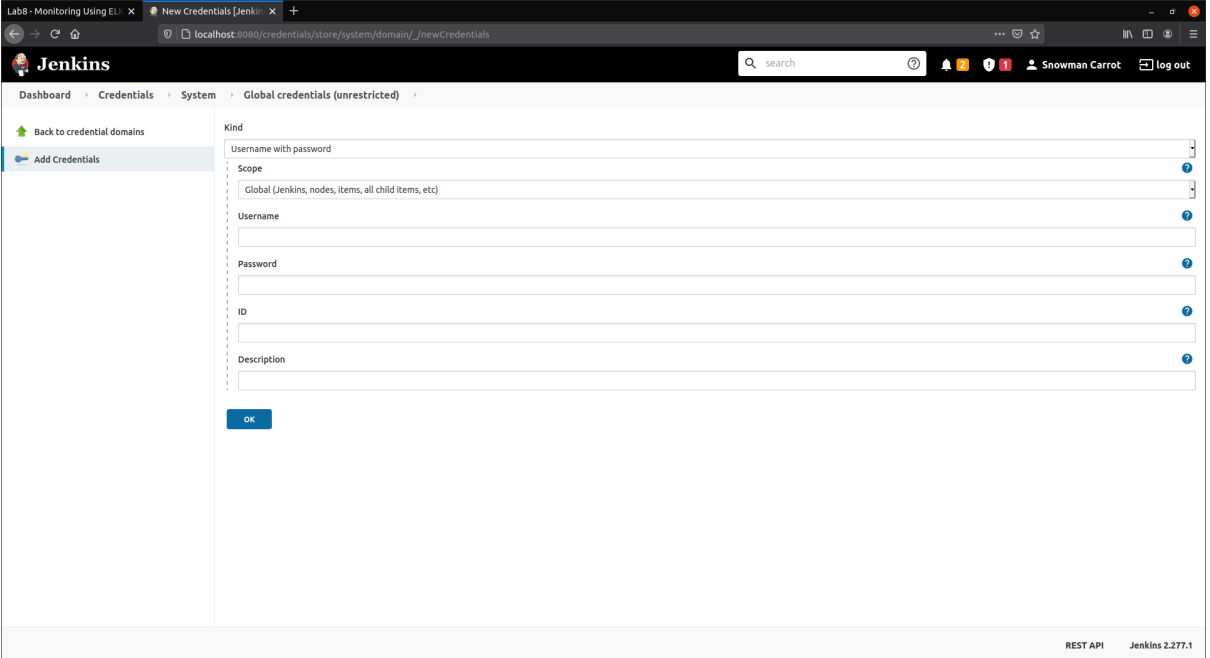
Calculator / src / main

- Project
- Calculator sources root, ~/Downloads/Calculator
 - .idea
 - out
 - src
 - java
 - CalculatorTest
 - main
 - java
 - calculator
 - calculator
 - resources
 - META-INF
 - log4j2.xml
 - target
 - .gitignore
 - Calculator.iml
 - calculator.log
 - Dockerfile
 - logstash.conf
 - pom.xml
 - remote-server
 - External Libraries
 - Scratches and Consoles

8. All java packages, test files can be run independently to check for errors at any point of time.
9. Once the project is ready, click on build and run the code for testing before committing changes to it.
10. After this click on commit and push to push the code to github.

Jenkins Pipeline

1. Login to jenkins on <http://localhost:8080>.
2. Go to Dashboard/manage Jenkins/ manage plugins/available.
3. Install the following plugins
 - a. Docker
 - b. Docker plugin
 - c. Git
 - d. Github Integration Plugin
 - e. Junit plugin
 - f. Maven integration
 - g. Pipeline
 - h. Rundeck plugin
4. Click on install without restart after selecting plugins.
5. Go to manage jenkins/manage credentials/ jenkins/ Global Credentials/
6. Click on Add Credentials.
7. Enter Docker Hub username, password, and give an ID to these credentials.



The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `localhost:8080/credentials/store/system/domain/_/newCredentials`. The page title is "Jenkins". The breadcrumb navigation shows "Dashboard > Credentials > System > Global credentials (unrestricted)". On the left sidebar, there are links for "Back to credential domains" and "Add Credentials". The main form is titled "New Credentials" and contains the following fields:

- Kind:** A dropdown menu with "Username with password" selected.
- Scope:** A dropdown menu with "Global (Jenkins, nodes, items, all child items, etc)" selected.
- Username:** A text input field.
- Password:** A text input field.
- ID:** A text input field.
- Description:** A text input field.

At the bottom of the form is a blue "OK" button. The footer of the page shows "REST API" and "Jenkins 2.277.1".

8. Go to Dashboard/manage Jenkins/Configure system/rundeck
9. Click on add rundeck, give a name and enter URL, username and password of the local rundeck instance.

10. Then click on Test Connection.

Rundeck

Job cache

☐ Enable Rundeck job cache
Rundeck job cache configuration

Instances

Name
rundeck server

URL
http://localhost:4440

Login
admin

Password
Concealed Change Password

Auth Token ?

API Version ?

Test Connection

Delete Rundeck

Add Rundeck

List of Rundeck instances

Save Apply

11. Click on new item>pipeline

12. Next in the pipeline add the following code.

General Build Triggers Advanced Project Options **Pipeline**

Script

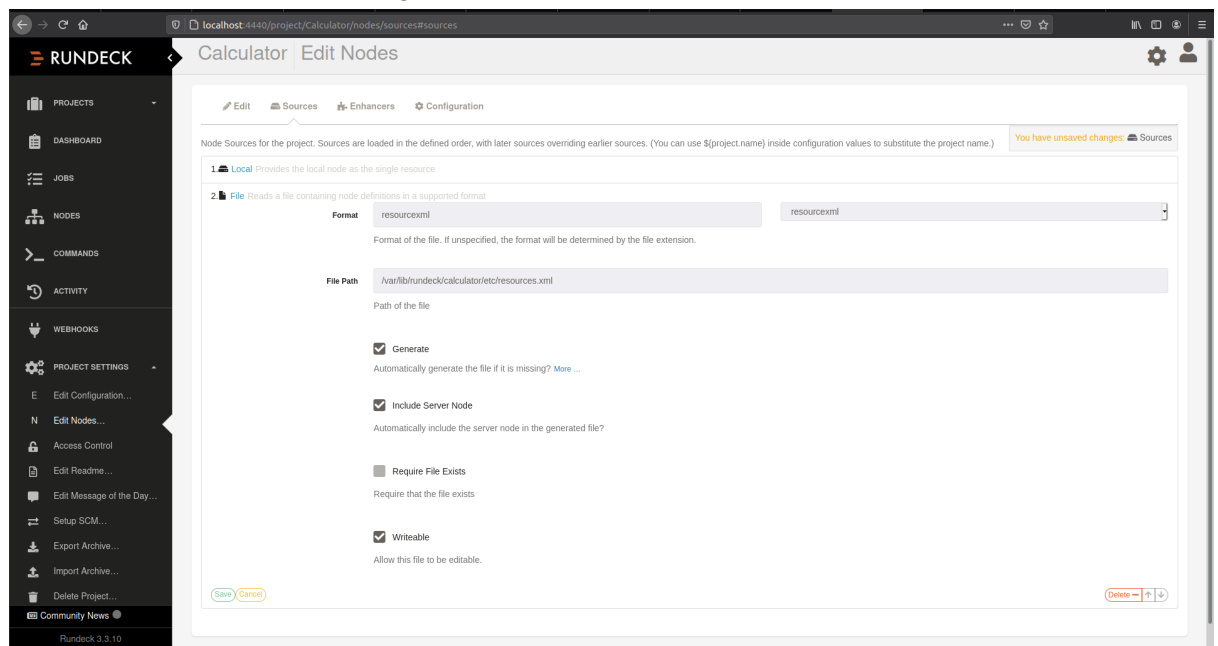
```
1 pipeline {
2   environment {
3     registry = "snowman0935/calculator"
4     registryCredential = 'dockerhub'
5     dockerimage = ''
6   }
7   agent any
8
9   triggers {
10    githubPush()
11  }
12
13  stages {
14    stage('Get code') {
15      steps {
16        git 'https://github.com/snowman0935/Calculator_Devops'
17      }
18    }
19    stage('Test Code') {
20      steps {
21        sh "mvn clean test"
22      }
23    }
24    stage('Build') {
25      steps {
26        sh "mvn -B -DskipTests clean package"
27      }
28    }
29    stage('Build Docker Image'){
30      steps{
31        script{
32          dockerimage = docker.build registry
33        }
34      }
35    }
36    stage('Push image to Docker HUB'){
37      steps{
38        script{
39          docker.withRegistry( '', registryCredential ){
40            dockerimage.push()
41          }
42        }
43      }
44    }
45    stage('Deploy on Node') {
46      steps {
47        script {
48          step([
49            $class: "RundeckNotifier",
50            includeRundeckLogs: true,
51            rundeckInstance: "rundeck server",
52            jobId: "07befa1f-0849-4121-9cc7-fcb05a4122f8",
53            shouldWaitForRundeckJob: true,
54            shouldFailTheBuild: true,
55            tailLog: true
56          ])
57        }
58      }
59    }
60  }
61 }
62
```

Save Apply

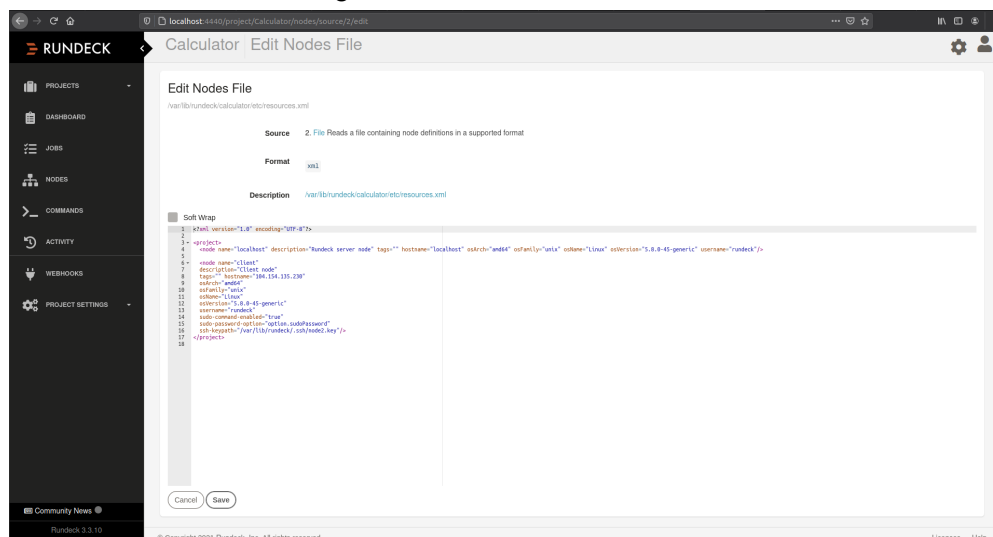
13. Update the values in the pipeline and save.

Rundeck

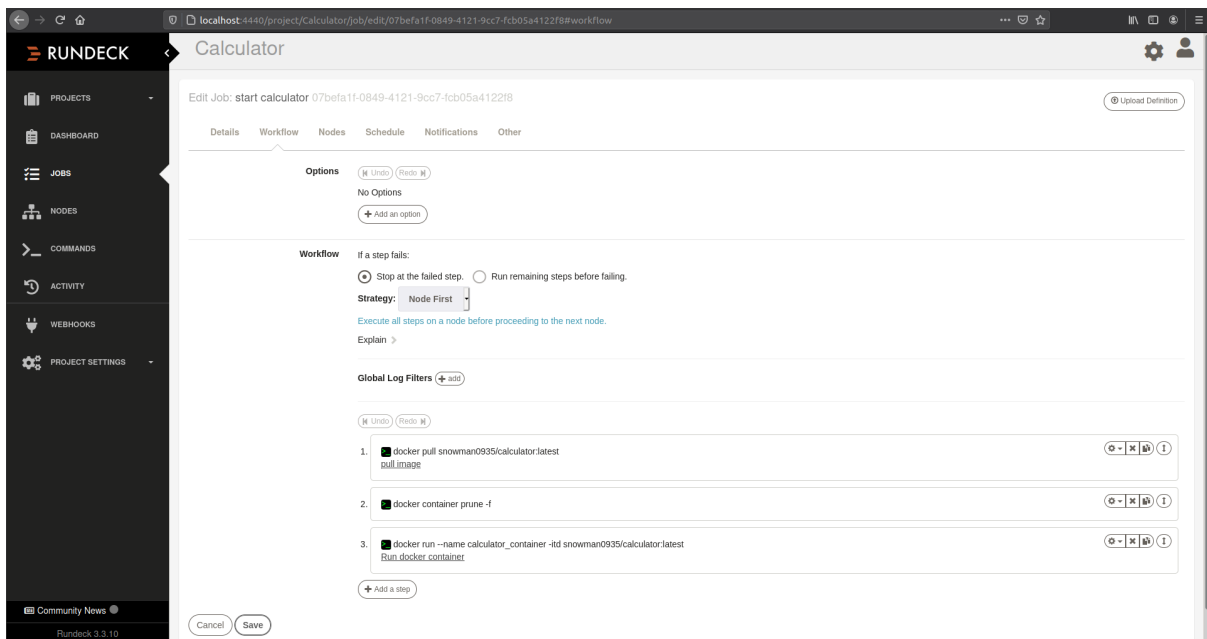
1. Login to Rundeck at <http://localhost:4440>
2. Create a new project. Give a name and click create.
3. Now, open the project go to settings/edit Nodes/Sources.
4. Click on add a new source > file.
5. Select resourcexml format.
6. Enter this file path `/var/lib/rundeck/calculator/etc/resources.xml`
7. Select:
 - a. Generate
 - b. Include Server Node
 - c. Writable.
8. Click on save and then save again.



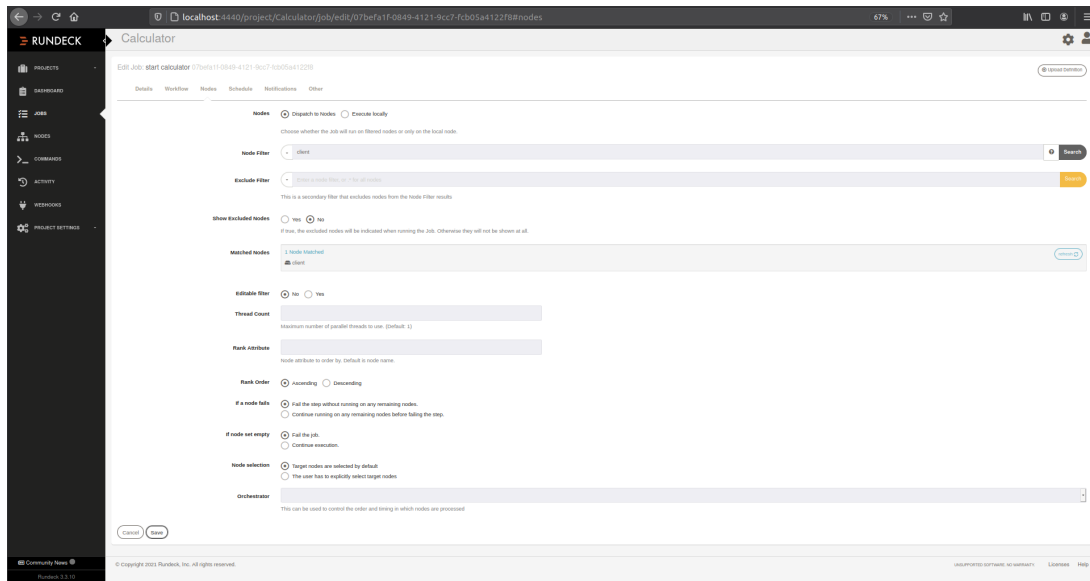
9. Go to edit and modify resources.xml file.
10. Enter the following code.



11. Here we are entering the details of the deployment server. So, we need the username, ip address, private key location for ssh and some more configuration details.
- Ip address: In google cloud go to Compute Engine/VM instances. Here we can see the instance we created before and the public ip address is shown. Update this ip address in the resources.xml file.
 - Private key:
 - go to `/var/lib/rundeck/.ssh`.
 - `$ ssh-keygen` (give file name: node2.key)
 - `$ chown rundeck:rundeck /var/lib/rundeck/.ssh/*`
 - Copy the public key: `$ cat .id_rsa.pub`
 - Paste this in the VM deployment server in `/home/user/.ssh/`: (user = current username or a new user can be added for this)
 - `$ cat >> authorized_keys`
 - `$ sudo service ssh restart`
 - User: update the username of the VM.
12. Click on save.
13. Now go to commands, select our VM node and run the following to test connection:
- `$ uname -a`
14. Once the connection is tested go to jobs, create a new job.
15. Give a name, then go to workflow
16. Click on add step then command. Add the following commands.
- `docker pull <docker repo address>`
 - `docker container prune -f`
 - `docker run --name calculator_container -itd <image name>`



17. Then go to nodes and select Dispatch to Nodes.
18. Select the VM deployment server node and save.



Google Cloud (Deployment Server)

1. Go to Compute Engine/VM instances.
2. Start the VM Instance we created earlier if it is switched off.
3. Click on SSH to connect to the instance.
4. Paste the public key in the home directory of the user in `authorized_keys` file.

```
sreevardhan73@ubuntuserver: /home/rundeck/.ssh — Mozilla Firefox
https://ssh.cloud.google.com/projects/sturdy-mode-308200/zones/us-central1-a/instances/ubuntuserver...
Connected, host fingerprint: ssh-rsa 0 D0:9D:BA:CE:98:7A:B5:FE:E2:42:53:F0:AE:06
:10:DB:47:B6:35:AC:76:3D:4A:C9:FB:03:EC:CF:BA:4A:DE:94
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1037-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Sat Mar 20 03:43:06 2021 from 35.235.241.51
sreevardhan73@ubuntuserver:~$ ls
snap
sreevardhan73@ubuntuserver:~$ cd ../
sreevardhan73@ubuntuserver:/home$ ls
rundeck sreevardhan73 ubuntu
sreevardhan73@ubuntuserver:/home$ cd rundeck/
sreevardhan73@ubuntuserver:/home/rundeck$ ls
snap
sreevardhan73@ubuntuserver:/home/rundeck$ cd ../.ssh/
sreevardhan73@ubuntuserver:/home/rundeck/.ssh$ ls
authorized_keys
sreevardhan73@ubuntuserver:/home/rundeck/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDQDeV5G0deowtZ4RVWk0d01IwxnFmodLoLq4Jnd4LMQvFL66X2td8e0LwB2mv6z90XMIcP/LkVqLk
FVvpLgN0JQnqJw8Nu11cCUJ8L7/ge2c3UYSFuoA0oMQEKNGa/06G5+CV0e8qHB1RLrJzqD19IZ4NCr1xIFXCGS9fIqSzfZsdPVBup8NV1Nve/9KDnwA
QxgLVfJX527LVM1nqQh523oiw7KSagiuNzXrjAm9MCNICMQ42mzyhtHyskr/MtCYd4mUJXTP8U4viX10YyT//0HPnyg1WLKft0i9fa0DwBHn1aLgk8z
Z+1wrMY0oEUA+j24P1Ld4//Xop5L9CcxIgaH8Wo3+1AgpvfGugwGnh6oQckKrdPR96aS4Mou8X6akV3XZtExYCbV6k5U1CRsG5ImoYtQmUFDmur8aWhe
LvnSgtajK0msGrr0Sg40D+QVzipVt0jc1kgcx8mxz0KvnPGPkACNBjLWivPHbv/MrAo/HDmyCTpyNqULrcdjCCiRns= root@Snowpc
sreevardhan73@ubuntuserver:/home/rundeck/.ssh$
```

FINAL STEP

1. Go to jenkins, click on the pipeline we have created.
2. Click on Build Now.
3. The whole pipeline should work and our CI/CD is ready.

The screenshot shows the Jenkins Pipeline Calculator interface. The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, Pipeline Syntax, and GitHub Hook Log. The main area displays the 'Pipeline Calculator' for the 'Calculator' pipeline. It shows a 'Stage View' table with columns for 'Get code', 'Test Code', 'Build', 'Build Docker Image', 'Push Image to Docker HUB', and 'Deploy on Node'. The table lists several build runs with their respective stage durations. The 'Deploy on Node' stage for runs #31, #30, and #29 is marked as 'Failed'.

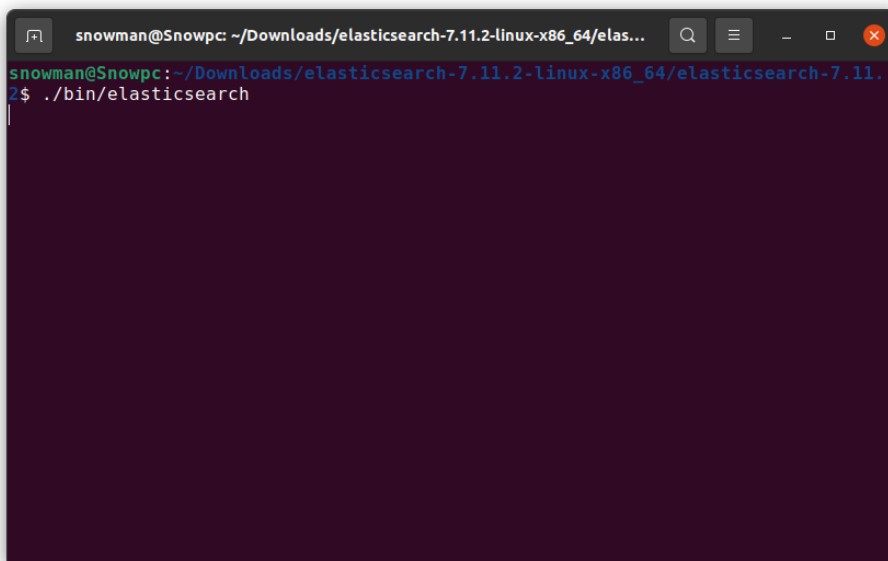
	Get code	Test Code	Build	Build Docker Image	Push Image to Docker HUB	Deploy on Node
Average stage times: (Average full run time: ~55s)	885ms	4s	4s	2s	21s	13s
#34 Mar 20, 2021 9:10 AM	1s	5s	4s	4s	31s	17s
#33 Mar 20, 2021 08:17 AM	747ms	5s	5s	3s	22s	15s
#32 Mar 20, 2021 08:09 AM	690ms	5s	8s	4s	21s	16s
#31 Mar 20, 2021 08:06 AM	1s	5s	5s	1s	21s	14s Failed
#30 Mar 20, 2021 8:03 AM	671ms	4s	4s	1s	27s	14s Failed
#29 Mar 20, 2021 7:57 AM	1s	4s	4s	3s	28s	10s Failed

Part-3: Continuous Monitoring

ELK Stack

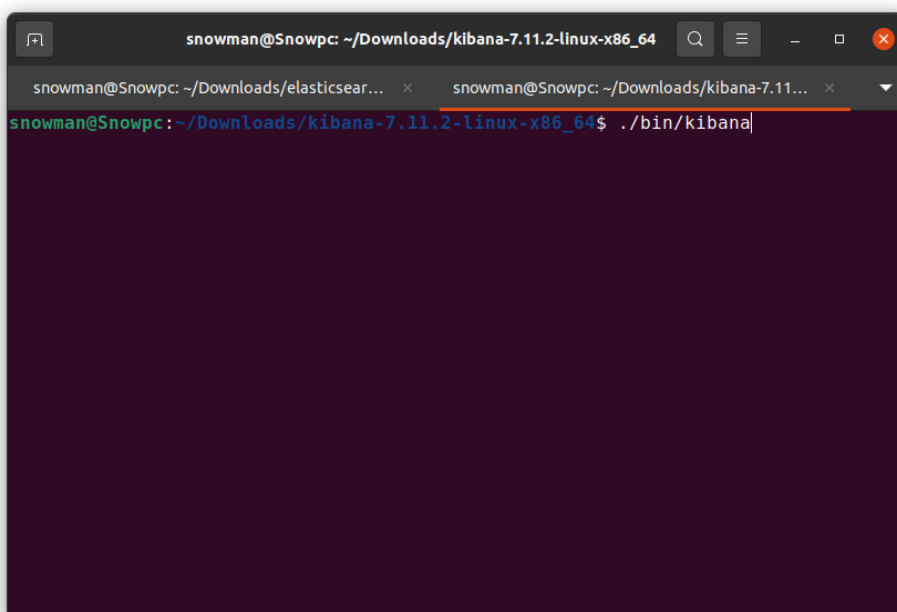
Once the project is deployed logs are generated when it is used.

1. Download the log file (calculator.log) from the VM Instance.
2. Create a logstash configuration file in the local computer and update the absolute path to the log file we just downloaded.
3. Now, Run ELK Stack
 - a. Go to elasticsearch directory, then \$./bin/elasticsearch



```
snowman@Snowpc: ~/Downloads/elasticsearch-7.11.2-linux-x86_64/elas...
snowman@Snowpc: ~/Downloads/elasticsearch-7.11.2-linux-x86_64/elasticsearch-7.11.2$ ./bin/elasticsearch
```

- b. Elasticsearch runs on port number 9200, you can verify it by visiting <http://localhost:9200>
- c. Next go to kibana directory and \$./bin/kibana



```
snowman@Snowpc: ~/Downloads/kibana-7.11.2-linux-x86_64
snowman@Snowpc: ~/Downloads/elasticsearch-7.11.2-linux-x86_64$ ./bin/kibana
```

- d. Kibana runs on port number 5601, you can verify it by visiting <http://localhost:5601>
- e. Next go to logstash directory and \$ `./bin/logstash -f /path/to/configuration/file`

```
snowman@Snowpc: ~/Downloads/Calculator_deploy
1 input {
2   file {
3     path => "/home/snowman/Downloads/Calculator/calculator.log"
4     start_position => "beginning"
5   }
6 }
7
8 filter {
9   grok {
10    match => [
11      "message", "%{HTTPDATE:timestamp_string} \[%{GREEDYDATA:thread}\] \[%{LOGLEVEL:level}\] %{GREEDYDATA:logger} \[%{GREEDYDATA:action}\] \- %{GREEDYDATA:line}"
12    ]
13  }
14
15  date {
16    match => ["timestamp_string", "dd/MMM/YYYY:HH:mm:ss SSS"]
17  }
18
19  mutate {
20    remove_field => [timestamp_string]
21  }
22 }
23
24 output {
25   elasticsearch {
26     hosts => ["http://localhost:9200"]
27     index => "calculator_elastic"
28   }
29
30   stdout {
31     codec => rubydebug
32   }
33 }
```

```
snowman@Snowpc: ~/Downloads/logstash-7.11.2-linux-x86_64/logstash-7.11.2
snowman@Snowpc: ~/Downloads/logstash-7.11.2-linux-x86_64/logstash-7.11.2$ ./bin/logstash -f /home/snowman/Downloads/Calculator_deploy/
authorized keys   calc logstash.conf  dockerfile
snowman@Snowpc: ~/Downloads/logstash-7.11.2-linux-x86_64/logstash-7.11.2$ ./bin/logstash -f /home/snowman/Downloads/Calculator_deploy/calc_logstash.conf
Using bundled JDK: /home/snowman/Downloads/logstash-7.11.2-linux-x86_64/logstash-7.11.2/jdk
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
```

4. Open Kibana in a web browser at <http://localhost:5601>
5. Go to Management -> Stack Management
6. Look for Kibana -> Index Patterns -> Create Index Pattern, set your index pattern based on the index pattern provided in logstash configuration file followed by *
7. And in the next step select @timestamp as your Time field
8. Hit Create Index Pattern, and you are ready to analyze the data.

elastic

Stack Management / Index patterns / Create index pattern

Ingest

Ingest Node Pipelines

Data

Index Management

Index Lifecycle Policies

Snapshot and Restore

Rollup Jobs

Transforms

Remote Clusters

Alerts and Insights

Alerts and Actions

Reporting

Kibana

[Index Patterns](#)

Saved Objects

Tags

Spaces

Advanced Settings

Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22`, or **multiple** data sources, `filebeat-*`.
[Read documentation](#)

Step 2 of 2: Configure settings

Specify settings for your `calc*` index pattern.

Select a primary time field for use with the global time filter.

Time field

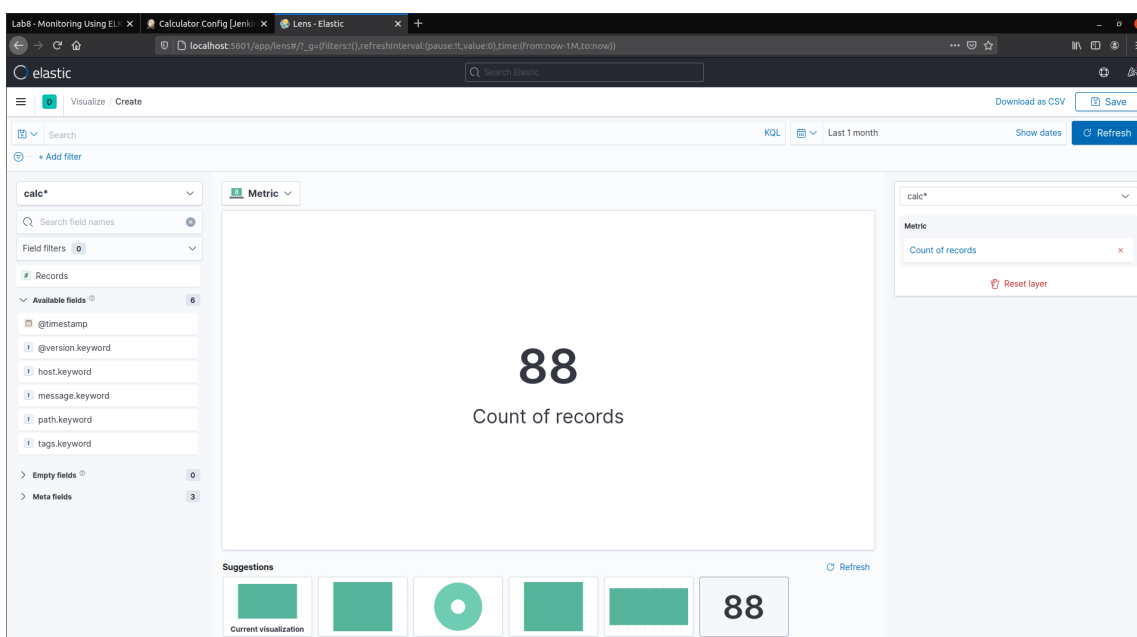
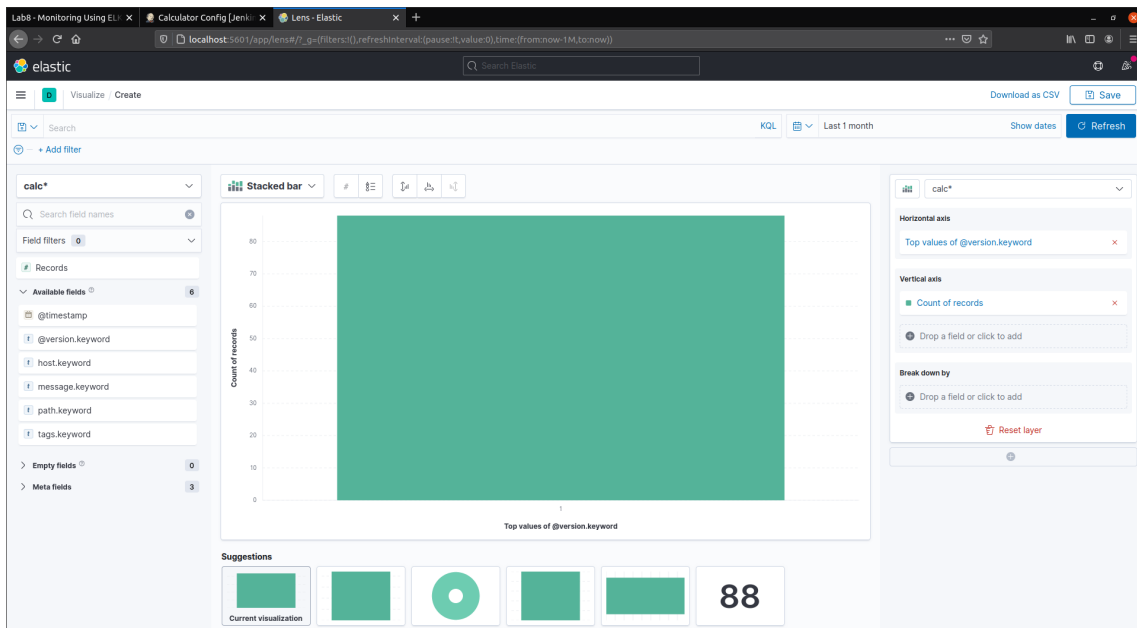
@timestamp

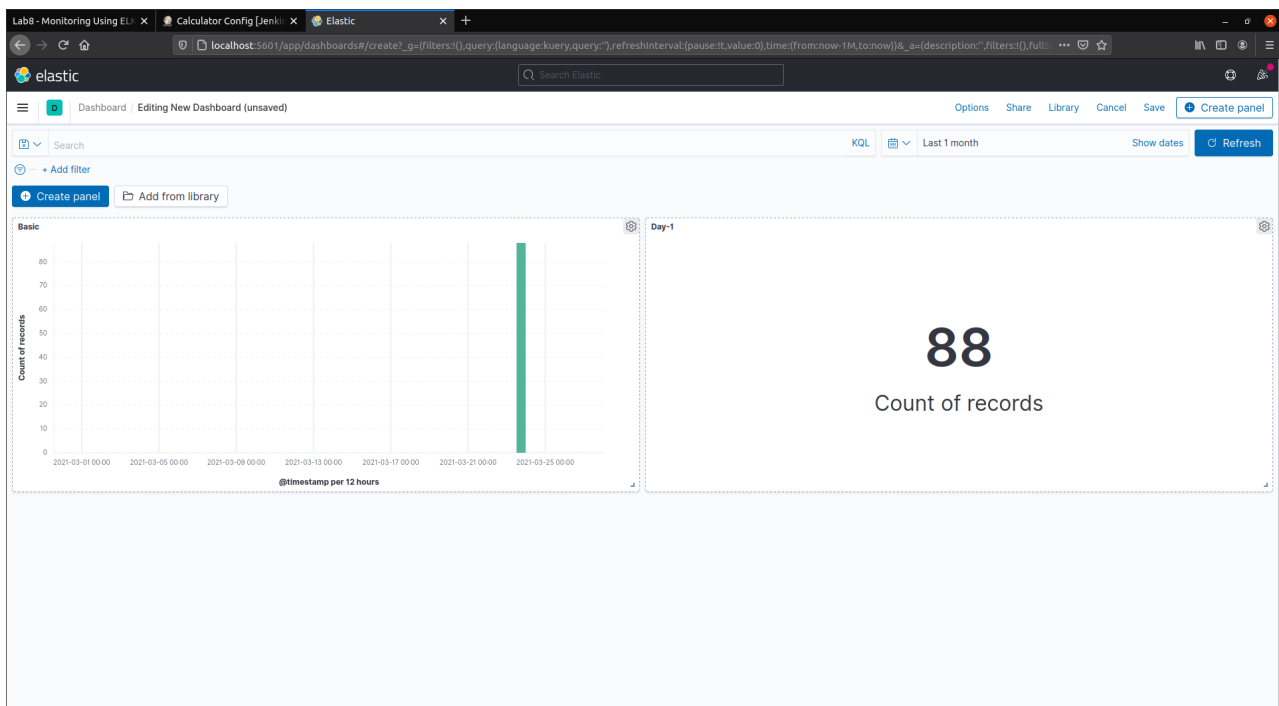
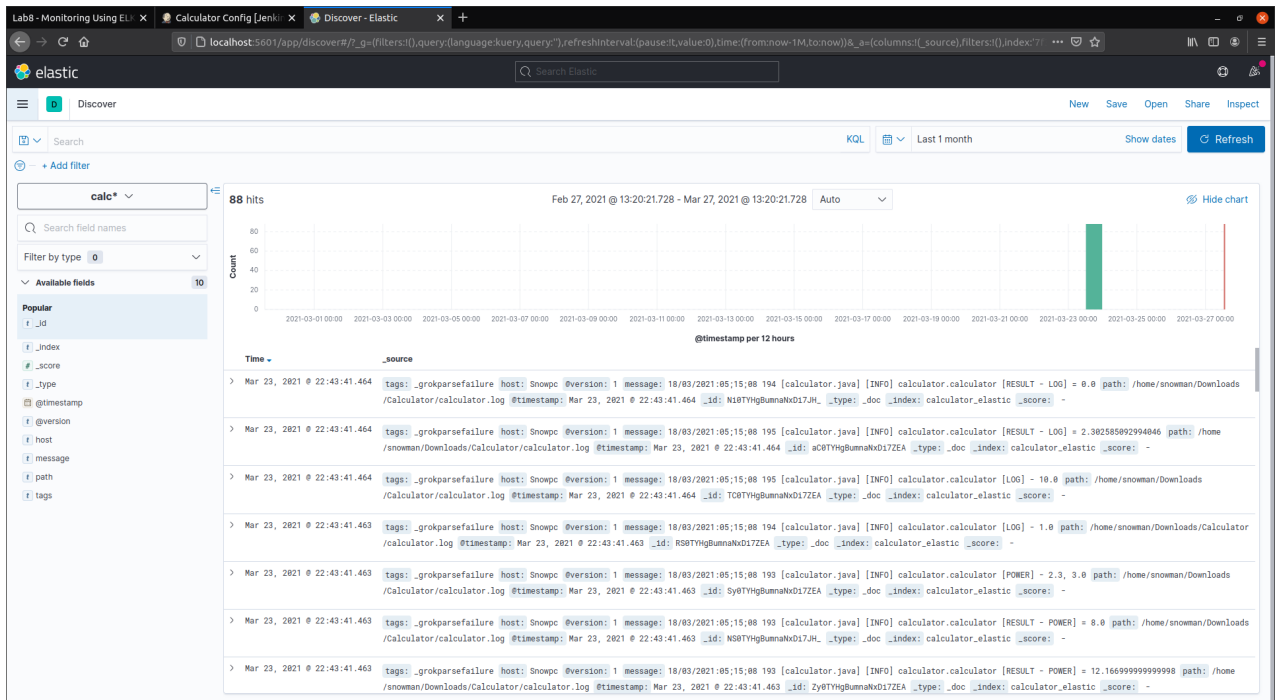
Refresh

> Show advanced settings

< Back

Create index pattern





Links

GitHub

https://github.com/snowman0935/Calculator_Devops

DockerHub

<https://hub.docker.com/repository/docker/snowman0935/calculator>

References

1. <https://www.journaldev.com/33645/maven-commands-options-cheat-sheet>
2. <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04>
3. <https://www.edureka.co/community/55640/jenkins-docker-docker-image-jenkins-pipeline-docker-registry>
4. <https://docs.docker.com/engine/install/linux-postinstall/>
5. <https://www.jenkins.io/doc/>
6. <https://docs.rundeck.com/docs/manual/03-getting-started.html>