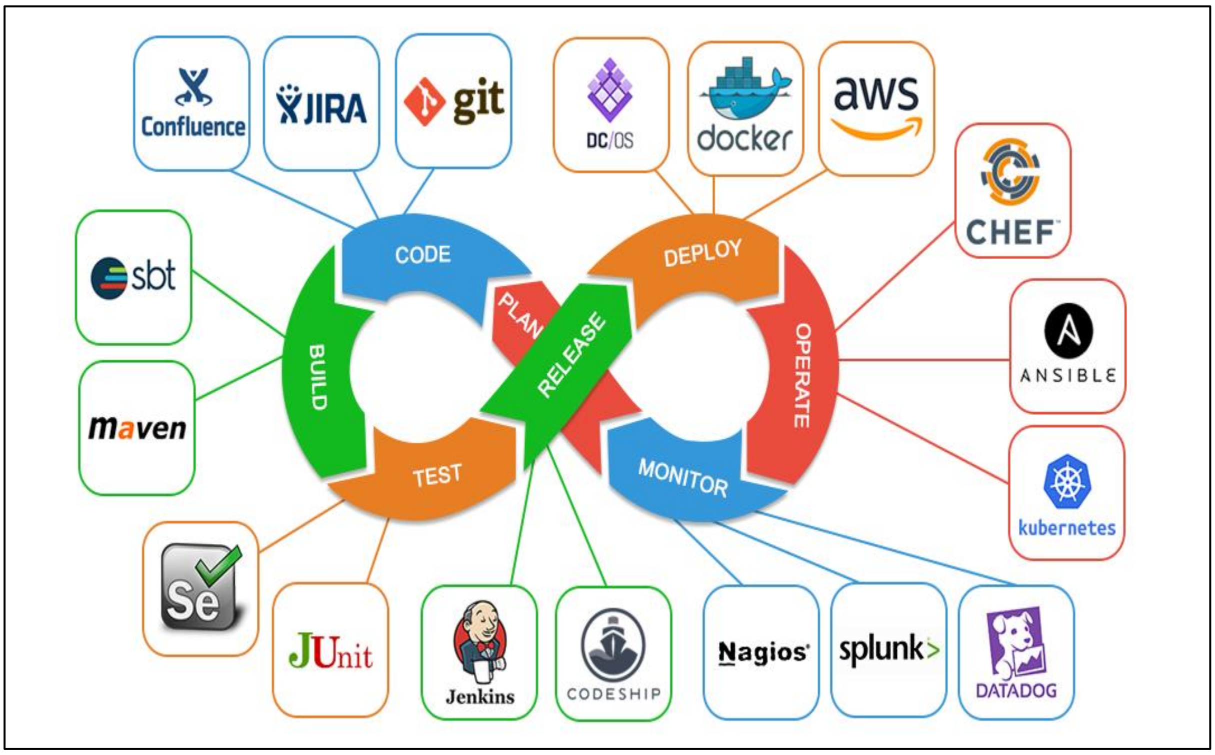
Scientific Calculator – Project Report

(Software Production Engineering)



Harshit Nigam

MT2021052

Project Objective:

The objective of this project is to build a scientific calculator with the help of automation tools, creating an integration and deployment chain. This includes source control management (SCM), Continuous Integration (CI), Continuous Deployment (CD), Configuration Management and Continuous Monitoring (CM). This is achieved by using DevOps tools like Jenkins, Docker, Maven, IntelliJ, and elk. Functionalities like Natural Logarithm, Square root, Power function and factorial are implemented and are taken through the Jenkins pipeline incrementally. The main objective of this project is to learn DevOps concepts CI/CD/CM that is achieved by creating a Jenkins pipeline.

What is DevOps?

DevOps (A shorter word for `Development` and `Operations`) is the combination of practices and tools designed to increase an organization’s ability to deliver applications and services faster than traditional software development processes. It is a collaboration of developers and operations engineers participating together in the entire service lifecycle from design through the development process to production.

Why DevOps?

1) DevOps has both development and operations team working side by side, which has led to shortening of development cycles.
2) Renews focus on customers and their experience of the product. DevOps promotes shorter releases and immediate feedback to enhance user experience.
3) Introduces automation to the development process and supports end-to-end responsibility.

Tools used:

1) Github: version control system
2) Jenkins: CI/CD pipeline
3) Maven: Build tool
4) Junit: Testing
5) Dockerhub: Containerisation/ holds Docker image of our application
6) AWS EC2: Provides a remote server to run our Docker image.
7) ELK: Continuous Monitoring

Project Structure:

1. Java project with maven on Intellij
2. Add some changes to pom.xml
3. Jenkins Pipeline for continuous deployment and integration
4. Create a pipeline and write script :
    i. git clone
    ii. mvn clean install
    iii. Create docker image from docker file
    iv. Push the created docker image to the docker hub
5. The last stage of pipeline script pulls and runs the docker image on remote server created through AWS EC2 instance.

Project Workflow:

1) Setup Java (8/11) environment for the project.
2) Use IntelliJ to create Maven project with Java CLI program on Scientific Calculator.
3) Linking the project with GitHub repository and using Git for SCM.
4) Install Jenkins and run on 8080 localhost port (Default).
5) Sign up in DockerHub in order to push images to Docker containers.
6) Create AWS account and an EC2 instance that works as a node machine on cloud.
7) Configure Jenkins with credentials and plugins.
8) Create Jenkins pipeline project and write pipeline script including stages like git clone, Maven Build, create Docker image, pushing image to Docker Hub, Cleaning previously build image and running image on remote server.
9) Build the pipeline and wait for successful running of all stages in the pipeline.
10) At last, the Docker image is pulled and run on the localhost.

Environment Setup for Cloud/Node machine:

My System Specifications:

OS: Windows 10 Home

Processor: Intel® Core™ i7-6500 CPU @ 2.50GHz 2.59GHz

RAM: 12GB

1. Installing Java:
   i) Install Java SE 8 using following link, https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html
   ii) Edit Environment variable path and add 'C:\Program Files\Java\jdk1.8.0_311\bin' .

2. Install IntelliJ (Community Edition): https://www.jetbrains.com/idea/download/#section=windows
3. Download Jenkins .war file (LTS version)

   https://www.jenkins.io/download/

i) Change directory to the directory where Jenkins.war file is present and Run command: java –jar jenkins.war –httpPort=9090 (you can put any free port number).

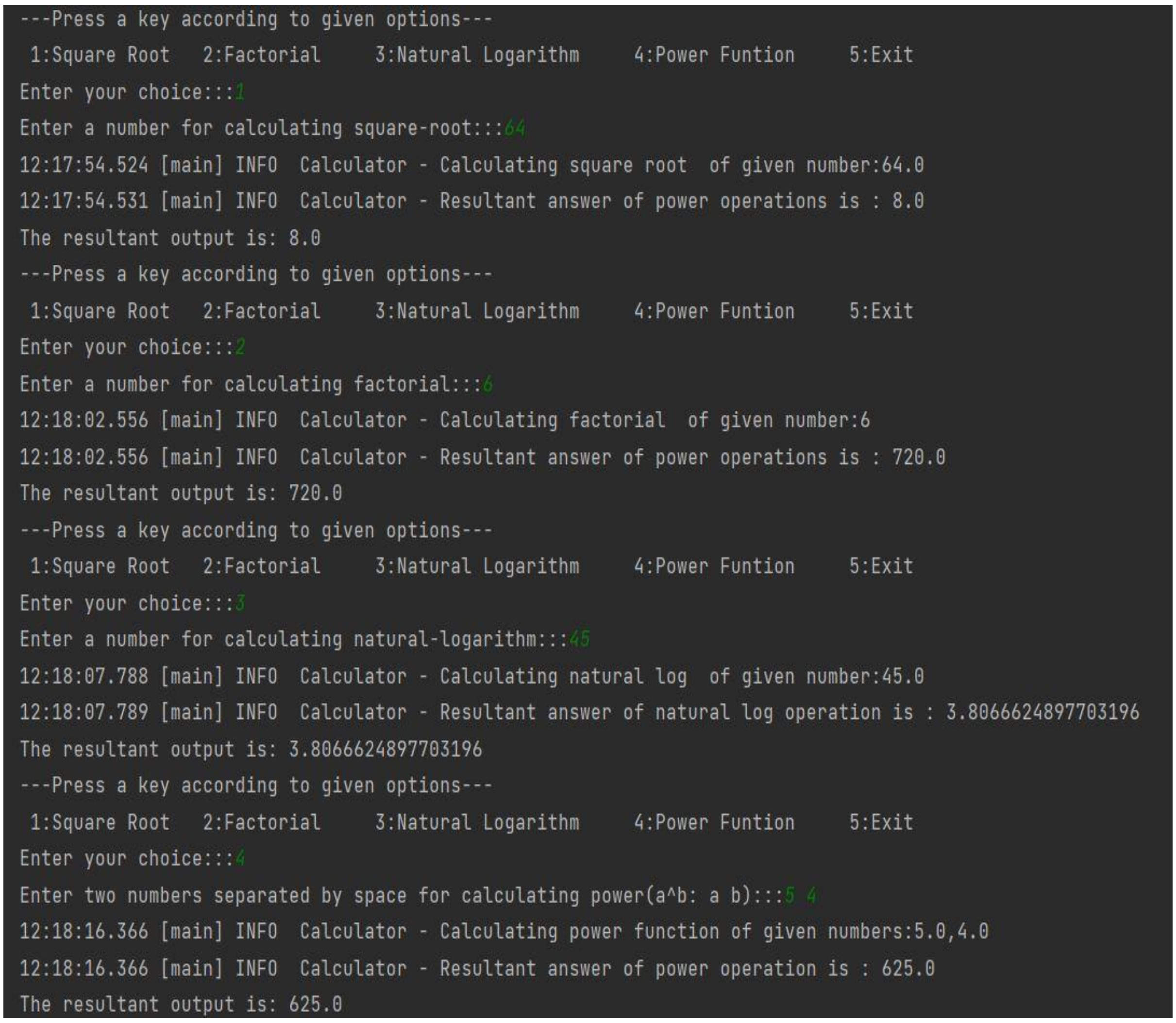
4. Install Docker Desktop for Windows https://docs.docker.com/desktop/windows/install/

1. Write Source Code in Java

Functionalities include,

i) Power Function
ii) Factorial
iii) Square root
iv) Natural Logarithm

Code: https://github.com/NightmareNight-em/devops_calculator/blob/main/src/main/java/Calculator.java



```
---Press a key according to given options---
 1:Square Root   2:Factorial     3:Natural Logarithm     4:Power Funtion     5:Exit
Enter your choice:::1
Enter a number for calculating square-root:::64
12:17:54.524 [main] INFO  Calculator - Calculating square root  of given number:64.0
12:17:54.531 [main] INFO  Calculator - Resultant answer of power operations is : 8.0
The resultant output is: 8.0
---Press a key according to given options---
 1:Square Root   2:Factorial     3:Natural Logarithm     4:Power Funtion     5:Exit
Enter your choice:::2
Enter a number for calculating factorial:::6
12:18:02.556 [main] INFO  Calculator - Calculating factorial  of given number:6
12:18:02.556 [main] INFO  Calculator - Resultant answer of power operations is : 720.0
The resultant output is: 720.0
---Press a key according to given options---
 1:Square Root   2:Factorial     3:Natural Logarithm     4:Power Funtion     5:Exit
Enter your choice:::3
Enter a number for calculating natural-logarithm:::45
12:18:07.788 [main] INFO  Calculator - Calculating natural log  of given number:45.0
12:18:07.789 [main] INFO  Calculator - Resultant answer of natural log operation is : 3.8066624897703196
The resultant output is: 3.8066624897703196
---Press a key according to given options---
 1:Square Root   2:Factorial     3:Natural Logarithm     4:Power Funtion     5:Exit
Enter your choice:::4
Enter two numbers separated by space for calculating power(a^b: a b):::5 4
12:18:16.366 [main] INFO  Calculator - Calculating power function of given numbers:5.0,4.0
12:18:16.366 [main] INFO  Calculator - Resultant answer of power operation is : 625.0
The resultant output is: 625.0
```

Unit Testing – Junit:

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behavior or state.

A JUnit test is a method contained in a class, which is only used for testing. This is called a Test class. To define that a certain method is a test method, annotate it with the @Test annotation. This method executes the code under test. You use an assert method, provided by JUnit or another assert framework, to check an expected result versus the actual result. These calls are typically called asserts or assert statements. Assert statements typically allow to define messages which are shown if the test fails. You should provide here meaningful messages to make it easier for the user to identify and fix the problem.

Code for tests (using JUnit): https://github.com/NightmareNight-em/devops_calculator/blob/main/src/test/java/CalculatorTest.java

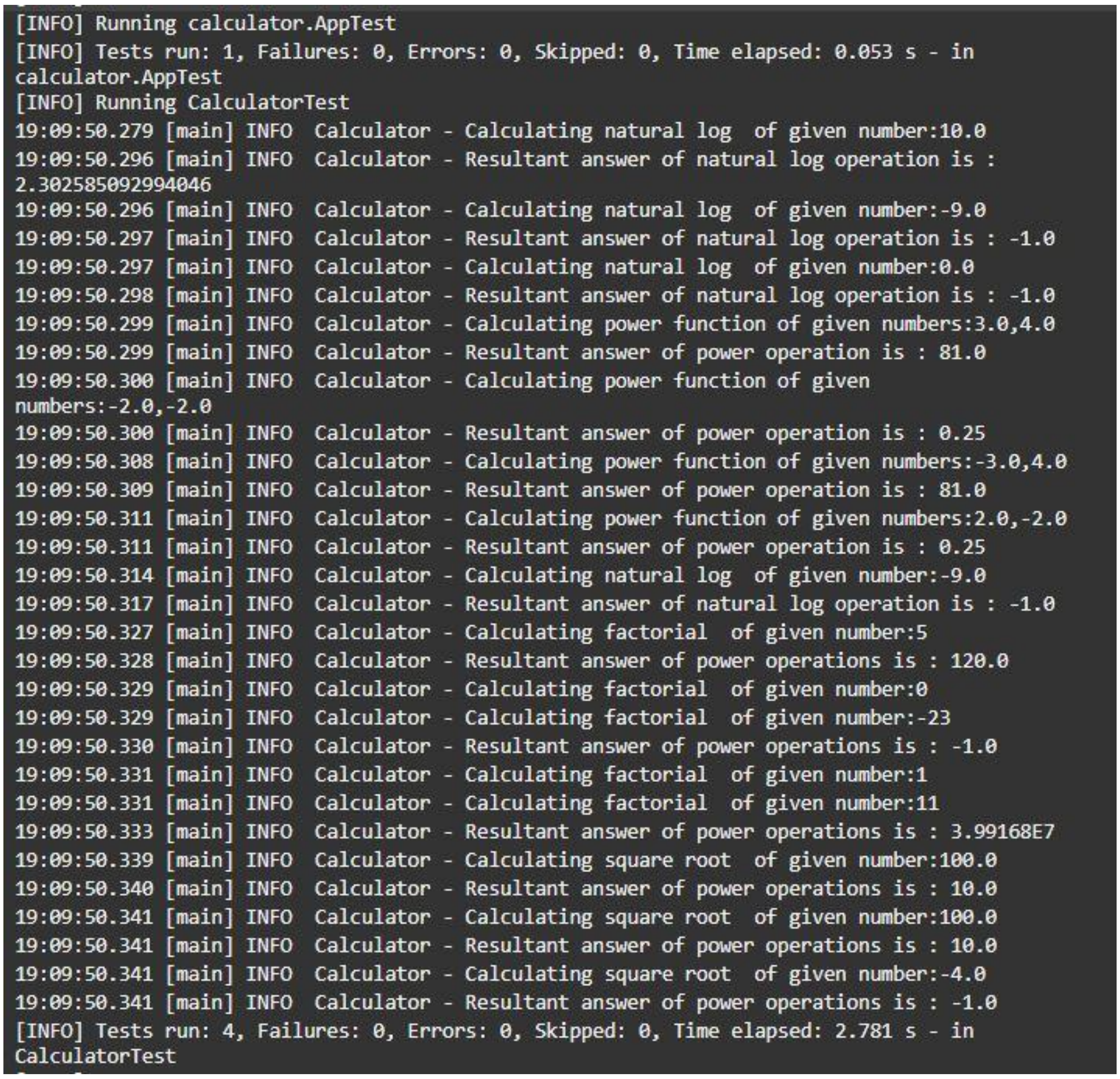
Fig.1 shows the number of test cases that passed and the test cases that failed to pass. This enables the developer to work on the issues that arise in the code and makes them easy to be identified to get them fixed.

2. Create Maven Project

i) Open IntelliJ and create Maven Project.

ii) Add appropriate dependencies in pom.xml to use log4j framework to track what happens in our application.

iii) Create a .java file in /src/main/java/ with source code and .java file in /src/main/test with test cases to test our application.



```
[INFO] Running calculator.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.053 s - in
calculator.AppTest
[INFO] Running CalculatorTest
19:09:50.279 [main] INFO  Calculator - Calculating natural log  of given number:10.0
19:09:50.296 [main] INFO  Calculator - Resultant answer of natural log operation is :
2.302585092994046
19:09:50.296 [main] INFO  Calculator - Calculating natural log  of given number:-9.0
19:09:50.297 [main] INFO  Calculator - Resultant answer of natural log operation is : -1.0
19:09:50.297 [main] INFO  Calculator - Calculating natural log  of given number:0.0
19:09:50.298 [main] INFO  Calculator - Resultant answer of natural log operation is : -1.0
19:09:50.299 [main] INFO  Calculator - Calculating power function of given numbers:3.0,4.0
19:09:50.299 [main] INFO  Calculator - Resultant answer of power operation is : 81.0
19:09:50.300 [main] INFO  Calculator - Calculating power function of given
numbers:-2.0,-2.0
19:09:50.300 [main] INFO  Calculator - Resultant answer of power operation is : 0.25
19:09:50.308 [main] INFO  Calculator - Calculating power function of given numbers:-3.0,4.0
19:09:50.309 [main] INFO  Calculator - Resultant answer of power operation is : 81.0
19:09:50.311 [main] INFO  Calculator - Calculating power function of given numbers:2.0,-2.0
19:09:50.311 [main] INFO  Calculator - Resultant answer of power operation is : 0.25
19:09:50.314 [main] INFO  Calculator - Calculating natural log  of given number:-9.0
19:09:50.317 [main] INFO  Calculator - Resultant answer of natural log operation is : -1.0
19:09:50.327 [main] INFO  Calculator - Calculating factorial  of given number:5
19:09:50.328 [main] INFO  Calculator - Resultant answer of power operations is : 120.0
19:09:50.329 [main] INFO  Calculator - Calculating factorial  of given number:0
19:09:50.329 [main] INFO  Calculator - Calculating factorial  of given number:-23
19:09:50.330 [main] INFO  Calculator - Resultant answer of power operations is : -1.0
19:09:50.331 [main] INFO  Calculator - Calculating factorial  of given number:1
19:09:50.331 [main] INFO  Calculator - Calculating factorial  of given number:11
19:09:50.333 [main] INFO  Calculator - Resultant answer of power operations is : 3.99168E7
19:09:50.339 [main] INFO  Calculator - Calculating square root  of given number:100.0
19:09:50.340 [main] INFO  Calculator - Resultant answer of power operations is : 10.0
19:09:50.341 [main] INFO  Calculator - Calculating square root  of given number:100.0
19:09:50.341 [main] INFO  Calculator - Resultant answer of power operations is : 10.0
19:09:50.341 [main] INFO  Calculator - Calculating square root  of given number:-4.0
19:09:50.341 [main] INFO  Calculator - Resultant answer of power operations is : -1.0
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.781 s - in
CalculatorTest
```

Fig.1

3. Host the project on GitHub

i) Create a new repository on GitHub

ii) Configure Maven project on local machine and link it to GitHub repository using following commands,

```
$ git init  // Initialize Maven project as Git local repository
$ git remote add origin "URL of the GitHub repository"
$ git add .   // stage Maven project in Git environment
$ git commit –m "Commit Message"
$ git push –u origin main
```

NOTE: To run these commands in Windows environment, we need application like `Git Bash`. Alternatively, `IntelliJ` can be used to connect with remote repository, add, commit and push.

4. Create an automated pipeline using Jenkins

Automation is the key principle of DevOps. Automation aims to remove human intervention and execute the entire development cycle from version control to deployment automatically using triggers, scripts etc.

Here, we use Jenkins pipeline, a powerful tool to make a CI/ CD pipeline and create flexible and customized stages to perform tasks involved in software development lifecycle.

1. Download Generic java package Jenkins.war

2. Open CMD with path of the .war file and execute it using following command,

```
$ java -jar jenkins.war
```

3. For the interface of Jenkins, go to **localhost:8080** in your browser.

When Jenkins is launched for the first time, it asks for a 32 bit encryption key for authentication which can be found at following path (For Windows OS):

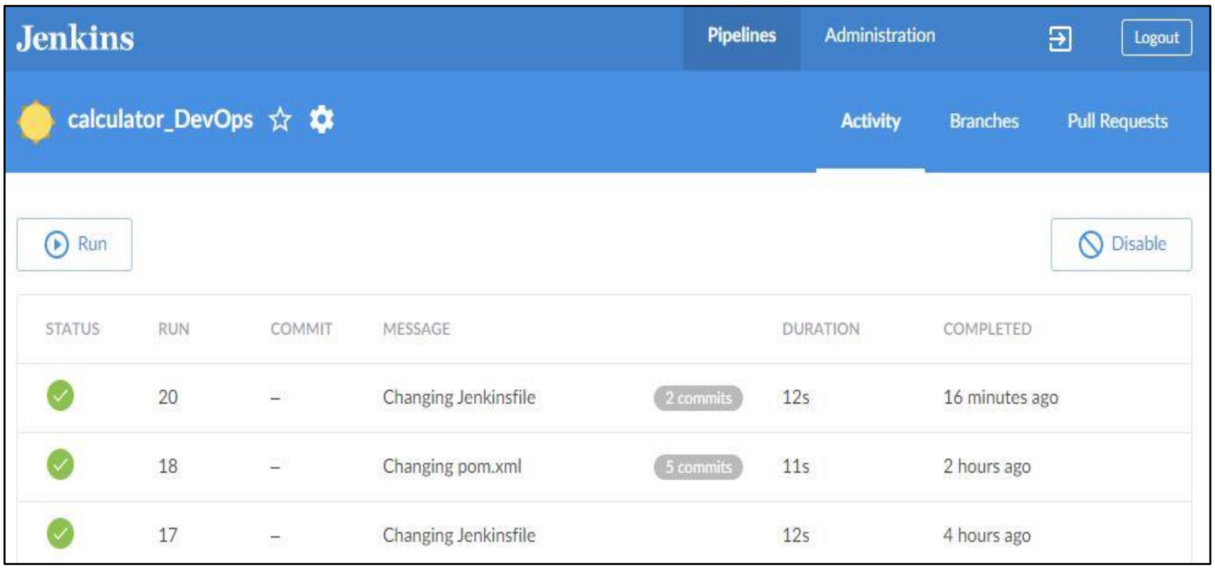
C://Users/{Username}/.jenkins/secrets/master.key

4. Jenkins configuration page opens up. Install suggested plugins, then signup with new user details, which will henceforth, open up the Jenkins dashboard.

5. Go to Manage Jenkins -> Manage Plugins -> Under Available section and install the following plugins to write the pipeline and restart Jenkins using above java command.

i) Blue Ocean
ii) Docker Pipeline
iii) Maven Integration Plugin
iv) Pipeline
v) Pipeline Maven Integration Plugin
vi) Pipeline utility steps
vii) SSH pipeline steps, SSH plugin, Publish over SSH (if using a remote server and connection using SSH).

6. Make a 'Pipeline' project, selecting `Git` in `Source Code Management` tab, `Invoke top level Maven Targets` in `Build Environment` tab. Put repository link and credentials for Github access by Jenkins which will trigger build on SCM change.



STATUS	RUN	COMMIT	MESSAGE		DURATION	COMPLETED
✓	20	–	Changing Jenkinsfile	2 commits	12s	16 minutes ago
✓	18	–	Changing pom.xml	5 commits	11s	2 hours ago
✓	17	–	Changing Jenkinsfile		12s	4 hours ago

7. Create Jenkins Pipeline script in a file named **Jenkinsfile** (same name and no extension) and save this file in root directory of Github repository.

Jenkinsfile. It is a simple text file used to write Jenkins pipeline and to automate CI process. It is often termed as `Pipeline as a code` which are of two types, Declarative and Scripted.

Pipeline stages:

i) Git clone – Checkout SCM
ii) mvn clean compile test package – Getting executable .jar file
iii) Building Docker Image
iv) Pushing docker image into DockerHub repository
v) Cleaning previously built images
vi) Running Dokcer container on development server (localhost)

Pipeline Script:

https://github.com/NightmareNight-em/devops_calculator/blob/main/Jenkinsfile

Note: pom.xml file for Maven project must be in root directory of GitHub repository.

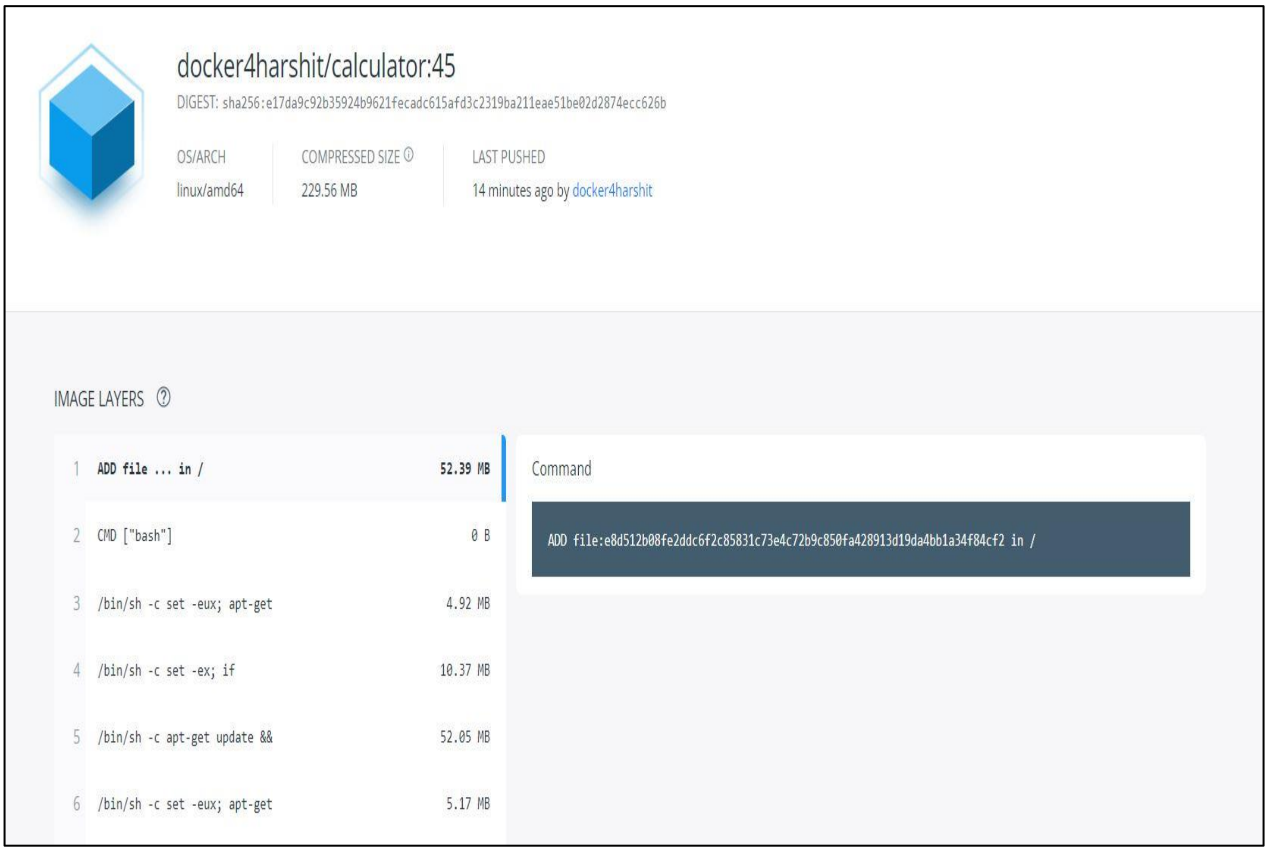
5. Containerization – DockerHub

DockerHub is an open-source project that is promoting cloud-native development through containerization and microservices.

Here, it is worth mentioning the concept of `Virtualization` wherein many guest operating systems can be run on same host OS without the need of extra hardware resources.

Containerization is a special type of virtualization where applications run in independent spaces called containers, all of which share same OS kernel to perform tasks independently.

Herein, we use DockerHub to create a repository containing our `container` which holds the docker image of the jar file which came as a result of build process performed by Maven. We first build the docker image and then push it into dockerhub repository in a container.



docker4harshit/calculator:45

DIGEST: sha256:e17da9c92b35924b9621fecadc615afd3c2319ba211eae51be02d2874ecc626b

OS/ARCH	COMPRESSED SIZE	LAST PUSHED
linux/amd64	229.56 MB	14 minutes ago by docker4harshit

IMAGE LAYERS

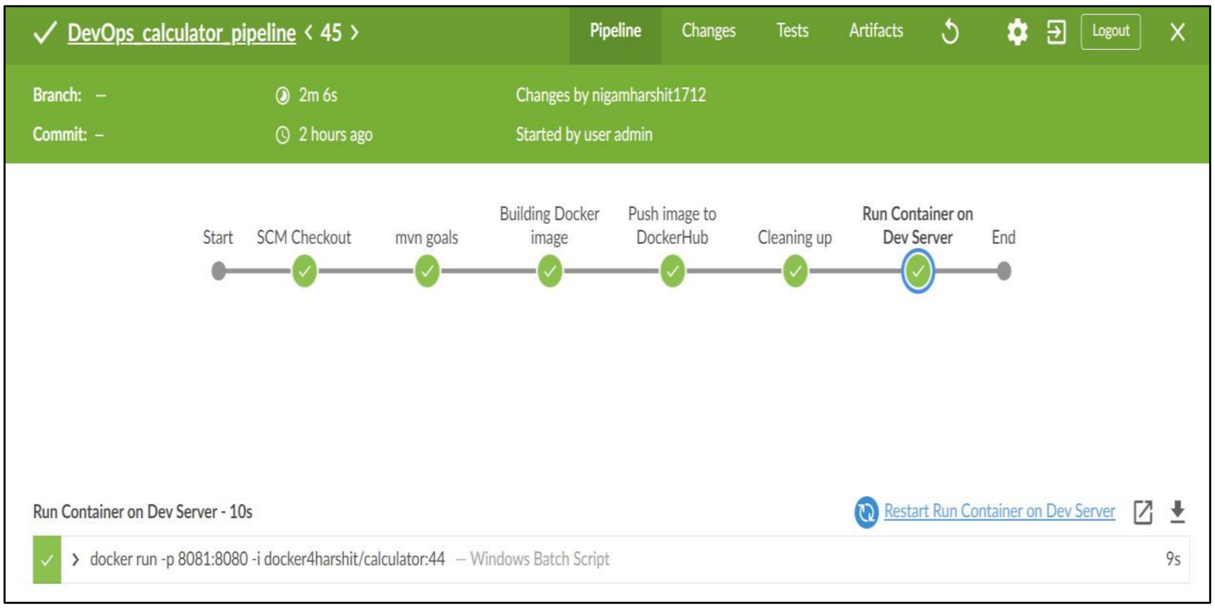
1	ADD file ... in /	52.39 MB
2	CMD ["bash"]	0 B
3	/bin/sh -c set -eux; apt-get	4.92 MB
4	/bin/sh -c set -ex; if	10.37 MB
5	/bin/sh -c apt-get update &&	52.05 MB
6	/bin/sh -c set -eux; apt-get	5.17 MB

Command

```
ADD file:e8d512b08fe2ddc6f2c85831c73e4c72b9c850fa428913d19da4bb1a34f84cf2 in /
```

Fig 2. Docker image with tag: 45 in repository `docker4harshit`

6. Working of Pipeline

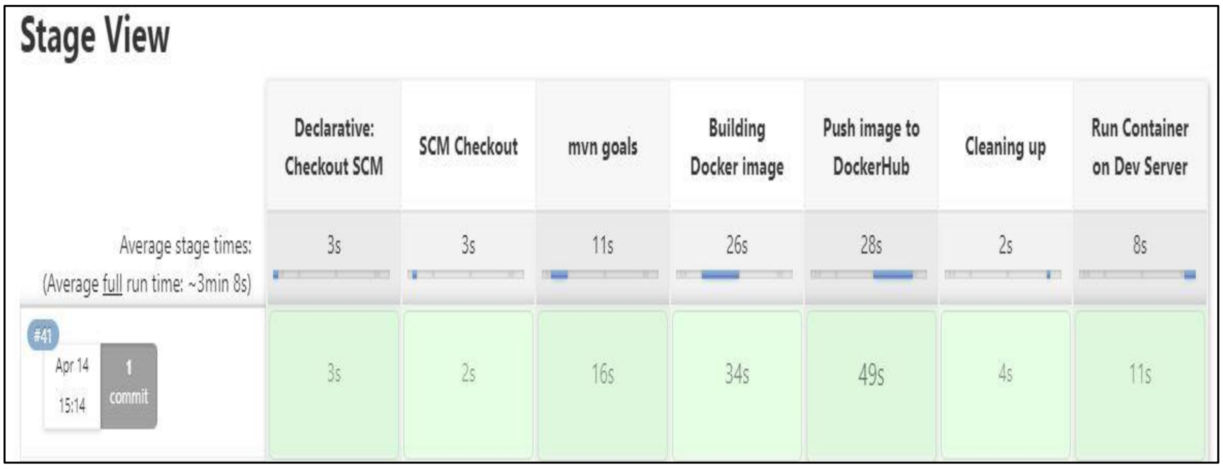


Branch: –
Commit: –
2m 6s
2 hours ago
Changes by nigamharshit1712
Started by user admin

Run Container on Dev Server - 10s

```
> docker run -p 8081:8080 -i docker4harshit/calculator:44 – Windows Batch Script    9s
```

Stages displayed using BlueOcean (Plugin) dashboard



Stage View

	Declarative: Checkout SCM	SCM Checkout	mvn goals	Building Docker image	Push image to DockerHub	Cleaning up	Run Container on Dev Server
Average stage times: (Average full run time: ~3min 8s)	3s	3s	11s	26s	28s	2s	8s
#41 Apr 14 15:14 1 commit	3s	2s	16s	34s	49s	4s	11s

Stage 1:

Cloning from our GitHub repository.

Stage 2:

Command `mvn clean compile test package` is run on our local terminal by Jenkins. This results in a .jar executable file in /target folder.

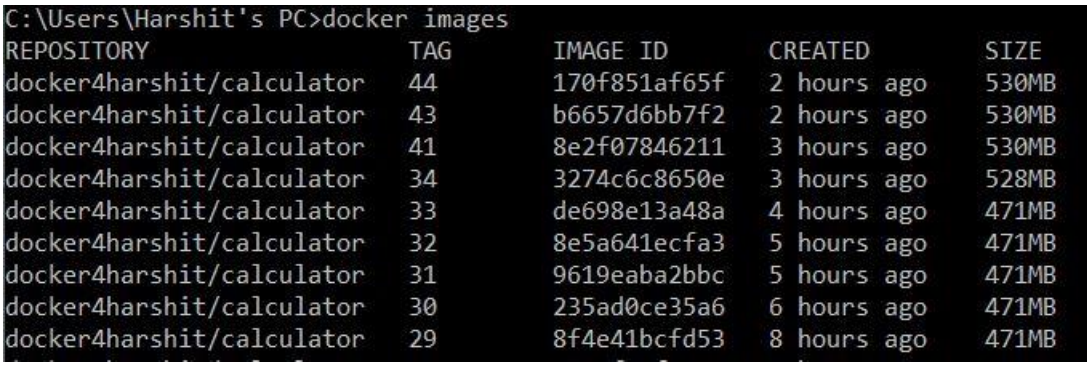
Stage 3 and 4:

Docker image of the jar file is built using Dockerfile (containing information about the location and execution command of the jar file among other details (see fig. 2).

Henceforth, previously build docker images are deleted from the repository.

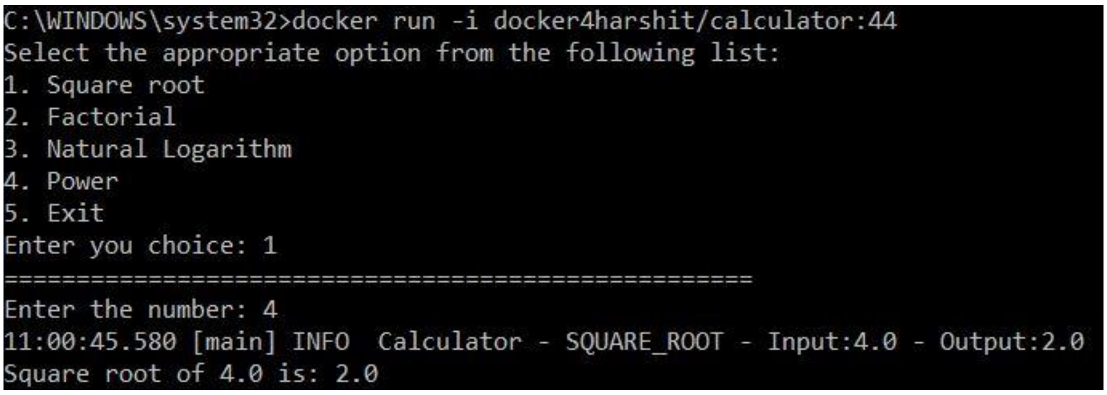
Stage 5:

The below docker images are pulled from dockerhub repository.

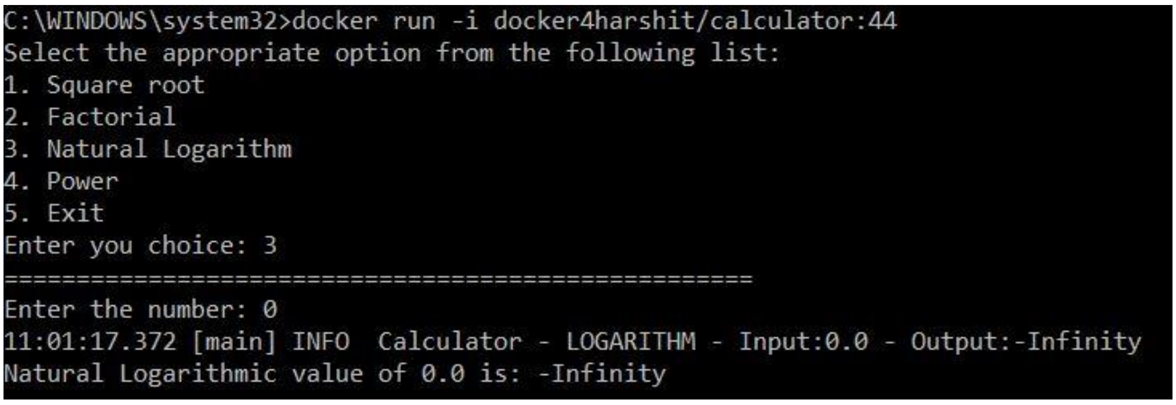


```
C:\Users\Harshit's PC>docker images
REPOSITORY                   TAG       IMAGE ID       CREATED        SIZE
docker4harshit/calculator    44        170f851af65f   2 hours ago    530MB
docker4harshit/calculator    43        b6657d6bb7f2   2 hours ago    530MB
docker4harshit/calculator    41        8e2f07846211   3 hours ago    530MB
docker4harshit/calculator    34        3274c6c8650e   3 hours ago    528MB
docker4harshit/calculator    33        de698e13a48a   4 hours ago    471MB
docker4harshit/calculator    32        8e5a641ecfa3   5 hours ago    471MB
docker4harshit/calculator    31        9619eaba2bbc   5 hours ago    471MB
docker4harshit/calculator    30        235ad0ce35a6   6 hours ago    471MB
docker4harshit/calculator    29        8f4e41bcfd53   8 hours ago    471MB
```

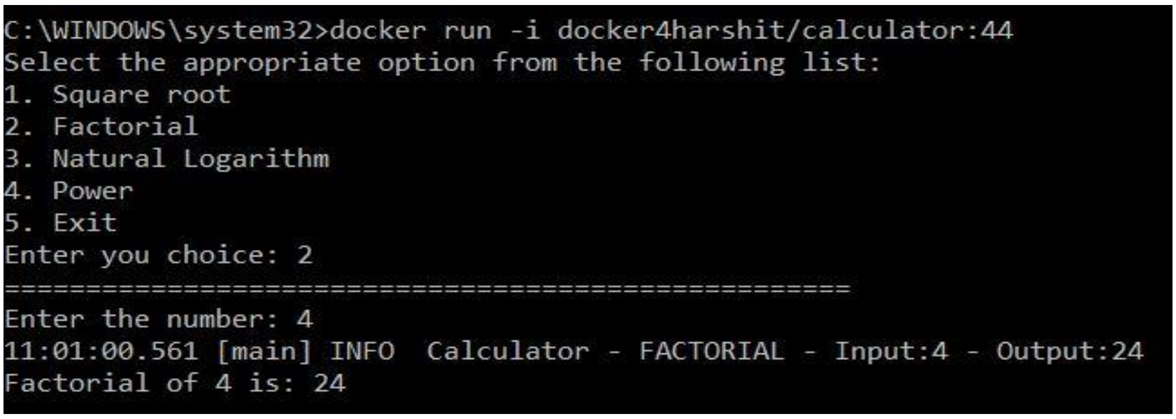
Below screenshots describe image (our Java application) running on terminal in local machine.



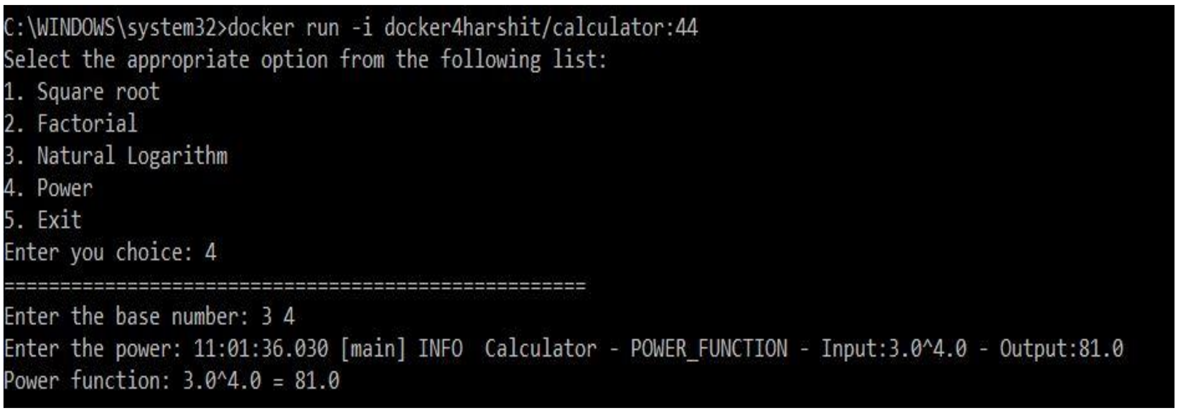
```
C:\WINDOWS\system32>docker run -i docker4harshit/calculator:44
Select the appropriate option from the following list:
1. Square root
2. Factorial
3. Natural Logarithm
4. Power
5. Exit
Enter you choice: 1
=====================================================
Enter the number: 4
11:00:45.580 [main] INFO  Calculator - SQUARE_ROOT - Input:4.0 - Output:2.0
Square root of 4.0 is: 2.0
```



```
C:\WINDOWS\system32>docker run -i docker4harshit/calculator:44
Select the appropriate option from the following list:
1. Square root
2. Factorial
3. Natural Logarithm
4. Power
5. Exit
Enter you choice: 3
=====================================================
Enter the number: 0
11:01:17.372 [main] INFO  Calculator - LOGARITHM - Input:0.0 - Output:-Infinity
Natural Logarithmic value of 0.0 is: -Infinity
```



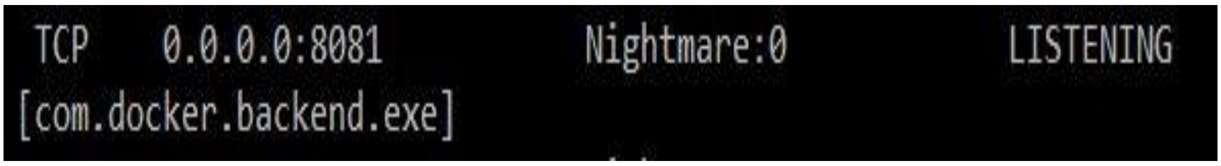
```
C:\WINDOWS\system32>docker run -i docker4harshit/calculator:44
Select the appropriate option from the following list:
1. Square root
2. Factorial
3. Natural Logarithm
4. Power
5. Exit
Enter you choice: 2
=====================================================
Enter the number: 4
11:01:00.561 [main] INFO  Calculator - FACTORIAL - Input:4 - Output:24
Factorial of 4 is: 24
```



```
C:\WINDOWS\system32>docker run -i docker4harshit/calculator:44
Select the appropriate option from the following list:
1. Square root
2. Factorial
3. Natural Logarithm
4. Power
5. Exit
Enter you choice: 4
=====================================================
Enter the base number: 3 4
Enter the power: 11:01:36.030 [main] INFO  Calculator - POWER_FUNCTION - Input:3.0^4.0 - Output:81.0
Power function: 3.0^4.0 = 81.0
```

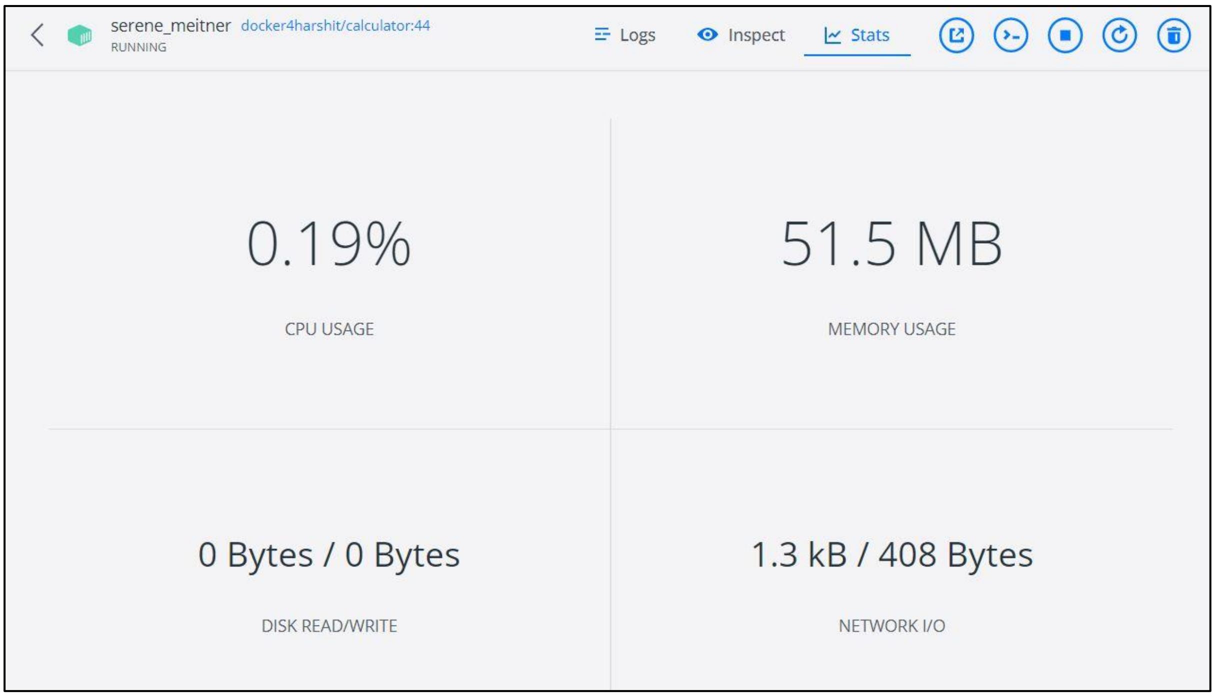
>docker run –p 8081:8080 –i docker4harshit/calculator:44

The above command is run in this stage which binds local port 8081 to the container port 8080 and runs the docker image `docker4harshit:44` on this port.



```
TCP    0.0.0.0:8081           Nightmare:0            LISTENING
[com.docker.backend.exe]
```

Image running on port 8081



0.19%
CPU USAGE

51.5 MB
MEMORY USAGE

0 Bytes / 0 Bytes
DISK READ/WRITE

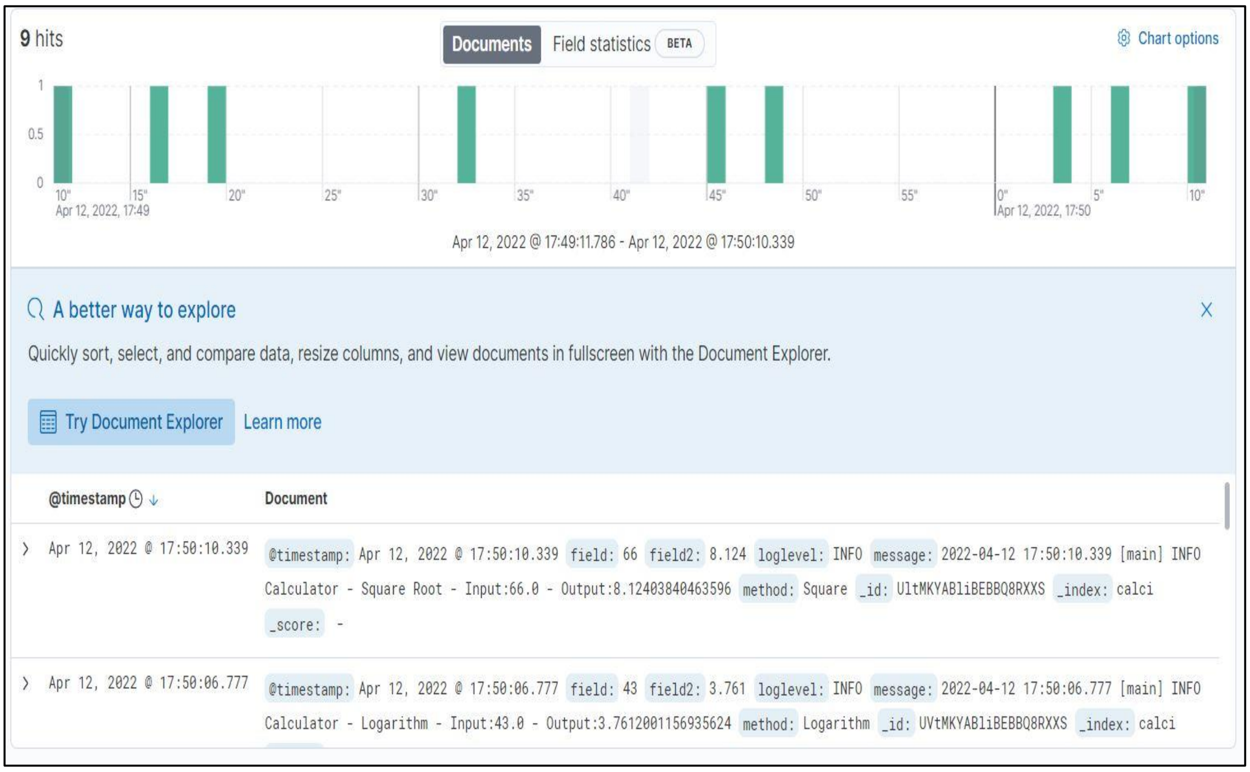
1.3 kB / 408 Bytes
NETWORK I/O

Container started – Image running

Jenkins pipeline helps to make an automatic flow for the DevOps and helps us in Continuous Integration and Deployment on cloud platforms. This Jenkins job for now is manually triggered but for other SCM it can be triggered based on events such as a new push or new pull but for GitHub, you might have set up a webhook. This can be done via port forwarding; you have to make your Jenkins port 8080 open on the wide web so it can be triggered based on the event. For now, we will skip it and we will build it manually.

7. Continuous Monitoring – ELK

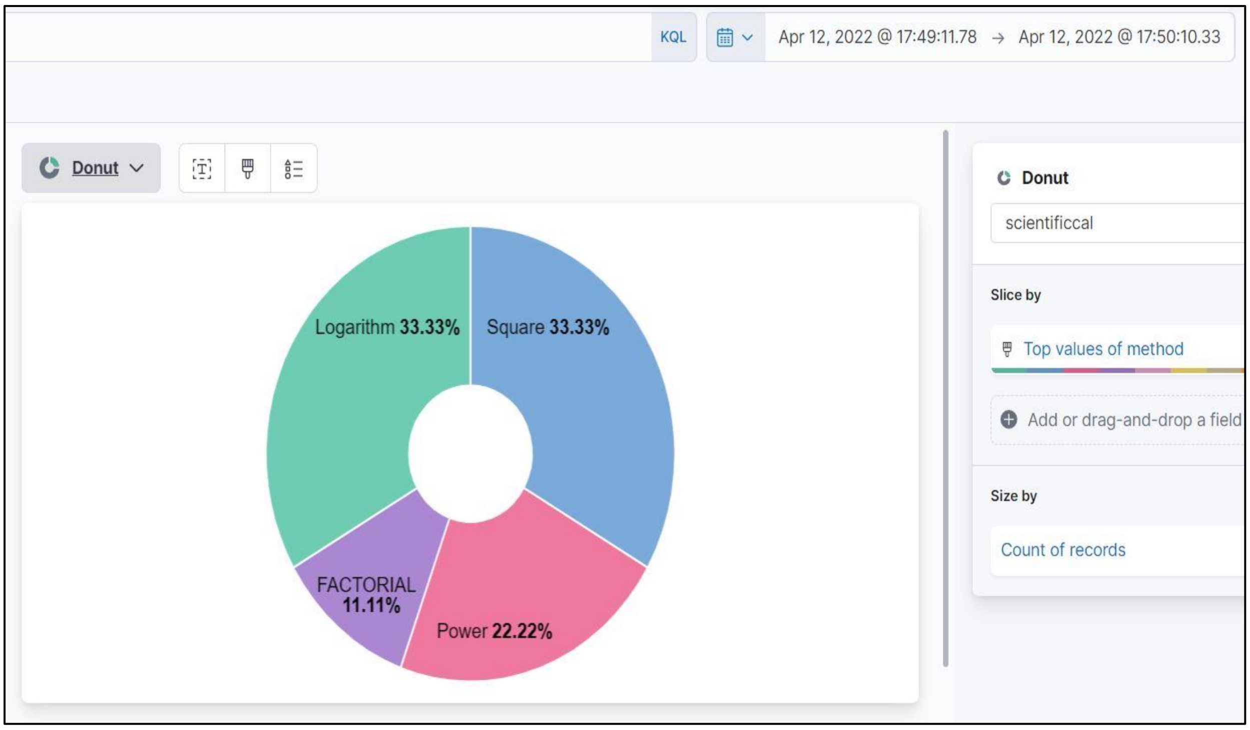
The ELK Stack is a log management platform – collection of three open-source products – Elasticsearch, Logstash and Kibana- from Elastic.



Apr 12, 2022 @ 17:49:11.786 - Apr 12, 2022 @ 17:50:10.339

Quickly sort, select, and compare data, resize columns, and view documents in fullscreen with the Document Explorer.

```
@timestamp                    Document
Apr 12, 2022 @ 17:50:10.339   @timestamp: Apr 12, 2022 @ 17:50:10.339 field: 66 field2: 8.124 loglevel: INFO message: 2022-04-12 17:50:10.339 [main] INFO
                              Calculator - Square Root - Input:66.0 - Output:8.12403840463596 method: Square _id: UltMKYABliBEBBQ8RXXS _index: calci
                              _score: -
Apr 12, 2022 @ 17:50:06.777   @timestamp: Apr 12, 2022 @ 17:50:06.777 field: 43 field2: 3.761 loglevel: INFO message: 2022-04-12 17:50:06.777 [main] INFO
                              Calculator - Logarithm - Input:43.0 - Output:3.7612001156935624 method: Logarithm _id: UVtMKYABliBEBBQ8RXXS _index: calci
```



Problems faced while working on the project

1) Docker within a docker official image doesn't support SSH. I got below error with all the ways I tried out.

   ```
   [ssh-agent] Could not find a suitable ssh-agent provider.
   ```

   To resolve this issue, I had option of running docker image locally or change docker image build of the code over Ubuntu docker image and run our created docker image on it.

2) Docker run command wasn't working until I discovered that it has to be used with `-i` options to work for applications with user inputs so that image is run in interactive mode.

3) pom.xml needed multiple updated dependencies and plugins to work correctly.

THANK YOU.