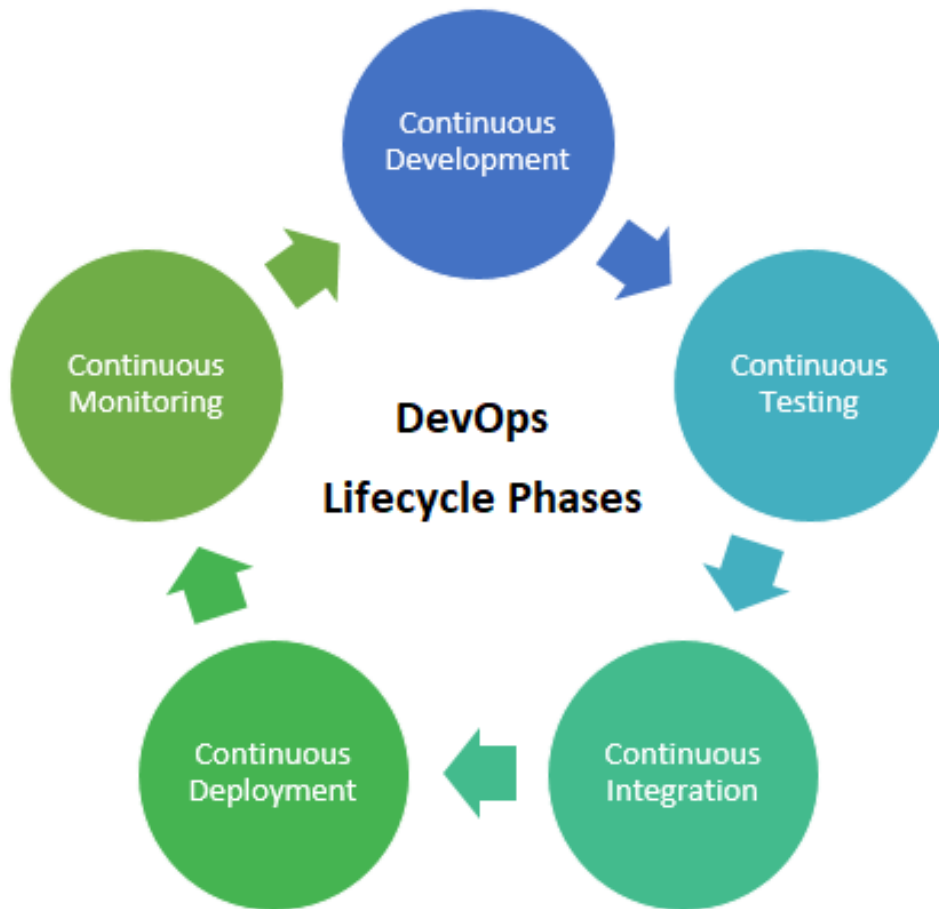


MINI PROJECT

SCIENTIFIC CALCULATOR WITH DEVOPS

-VAIBHAV TANDON "MT2020150"



GitHub Link:

<https://github.com/vtandonv/calculatorDevOps> (Project Initially Created)

<https://github.com/vtandonv/CalculatorUsingDevOps> (Project recreated for taking the screenshots)

DockerHub Link:

<https://hub.docker.com/repository/docker/vtandonv/calculatorusingdevopslatest>

Introduction

DevOps is a set of practices that combines software development and IT operations. It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.

The project is all about building a Java based Scientific Calculator application and integrating it with the DevOps tools (Jenkins, Ansible etc). The calculator will be able to perform exponentiation, natural logarithm, square root, and factorial of numbers given as input.

Talking about the overall build-integrate-deploy process of the application with DevOps it involves the following steps.

1. Git pull
2. Maven Build
3. Docker Build to image
4. Pushing Docker image
5. Ansible pull Docker image

The above 5 steps are performed in stages in a Continuous Integration- Continuous Development pipeline. The tool used for the same is Jenkins.

Setting up the environment

In order to successfully develop and deploy our application with DevOps the following software packages, tools and plugins are required:

1. **IntelliJ IDEA Ultimate Edition v2020.3:** This provides the environment for coding and testing of the application. Integrating it with Git allows us to directly commit and push the application to the Github repository. Download it from here using college student ID

<https://www.jetbrains.com/community/education/#students>

2. **Git install and setting up Github:** Git is a version control system which will contain the entire code base in the form of a repository. Any changes to our project can be directly pushed to our Github repository using either Git or Git plugin of IntelliJ. Command for the same is

```
$ sudo apt install git-all
```


3. **Setting up Maven on IntelliJ:** Maven is a build tool which resolves various dependencies present in our project like JUnit, log4j etc. It builds and packages the project in the form of a .jar file. Open a new Java application on IntelliJ as a Maven project.
4. **Installing Jenkins and setting up the Jenkins dashboard:** Jenkins is a powerful DevOps tool which streamlines our continuous integration and development in a single pipeline. To install Jenkins use command:


<https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-18-04>


Install various plugins like Git, Github, Docker and Ansible over the Jenkins


Enter an item name


» Required field


**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.

**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

IF

create a new item from other existing, you can use this option:

Fig: Logging into Jenkins in the localhost 8080 port and creating a Pipeline

-
5. **Installing Docker and making an account on Docker Hub:** Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. Install it using the command below:

<https://docs.docker.com/engine/install/ubuntu/>

6. **Setting up Ansible:** Ansible is a configuration management tool which is used to deploy the docker image and run the .jar file inside other host specified in Ansible inventory. Refer the below link for installation and setup.

https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Understanding the Jenkins pipeline stages

Jenkins is primarily used for streamlining the entire workflow. It has various plugins for each of the stages (like Git, Maven, Docker, Ansible etc) so that each and every stage can be continuously integrated and developed under one hood. By the word continuous we mean that as soon as the developer makes a single Git push to his repository the Jenkins will automatically perform all the steps which come under CI-CD scheme.

Stage 1: Git Pull

This stage involves creating a public GitHub repository and adding a ReadMe for the project.

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency

Basic Git Commands

Initialize the local directory as a Git repository.

\$ git init

Add the files in your new local repository. This stages them for the first commit.

\$ git add .

Adds the files in the local repository and stages them for commit. To unstage a file, use 'git reset HEAD YOUR-FILE'.

Commit the files that you've staged in your local repository.

```
$ git commit -m "First commit"
```

Commits the tracked changes and prepares them to be pushed to a remote repository. To remove this commit and modify the file, use 'git reset --soft HEAD~1' and commit and add the file again.

In Terminal, add the URL for the remote repository where your local repository will be pushed.

```
$ git remote add origin <REMOTE_URL>
```

Sets the new remote

```
$ git remote -v
```

Verifies the new remote URL

Push the changes in your local repository to GitHub.


```
$ git push origin main
```

Pushes the changes in your local repository up to the remote repository you specified as the origin

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 vtandonv ▾

Repository name *

CalculatorUsingDevOps ✓


Great repository names are short and memorable. Need inspiration? How about [silver-meme?](#)

Description (optional)

Building a Java based Scientific Calculator application using Maven build tool, integrating and developing using

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**


This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

Create repository

Fig 1.1: Creating a public Repository

main ▾


1 branch

0 tags

Go to file

Add file ▾

Code ▾

 vtandonv Initial commit


95973d4 now 1 commit

README.md

Initial commit

now

README.md



CalculatorUsingDevOps

Building a Java based Scientific Calculator application using Maven build tool, integrating and developing using Jenkins pipeline and finally deploying it over a host machine using Ansible

About

Building a Java based Scientific Calculator application using Maven build tool, integrating and developing using Jenkins pipeline and finally deploying it over a host machine using Ansible

Readme

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Fig 1.2: Adding ReadMe

Pipeline code for Git Pull:

```
stage('Git pull') {  
    steps {  
        git branch: 'main', url: 'https://github.com/vtandonv/CalculatorUsingDevOps'  
    }  
}
```

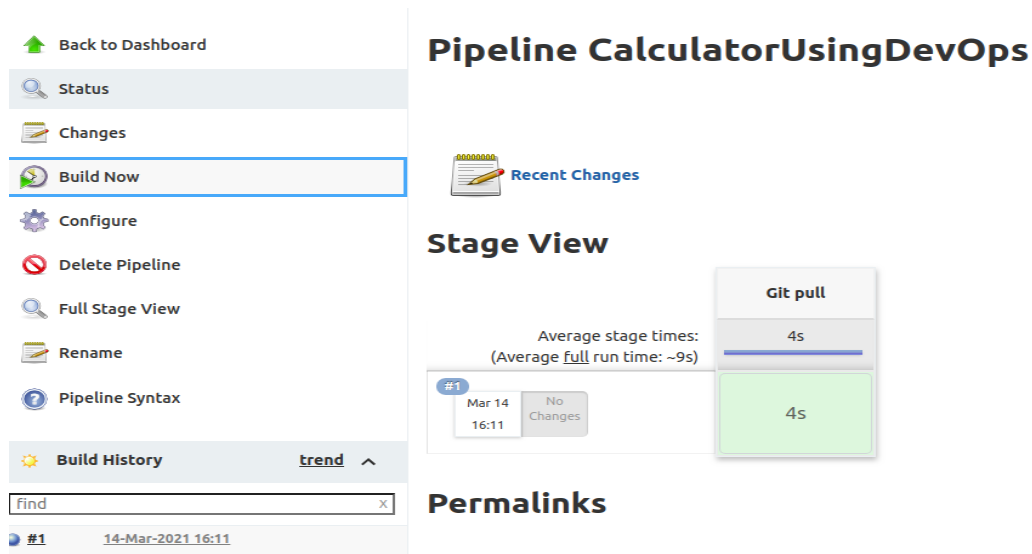


Fig 1.3: Successfully built Stage 1 of Pipeline

Stage 2: Maven Build

In this the entire project is being compiled, validated and tested by Maven and Junit. All this is handled by the Jenkins pipeline which outputs a target folder containing the .jar file.

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation.

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development. JUnit is linked as a JAR at compile-time

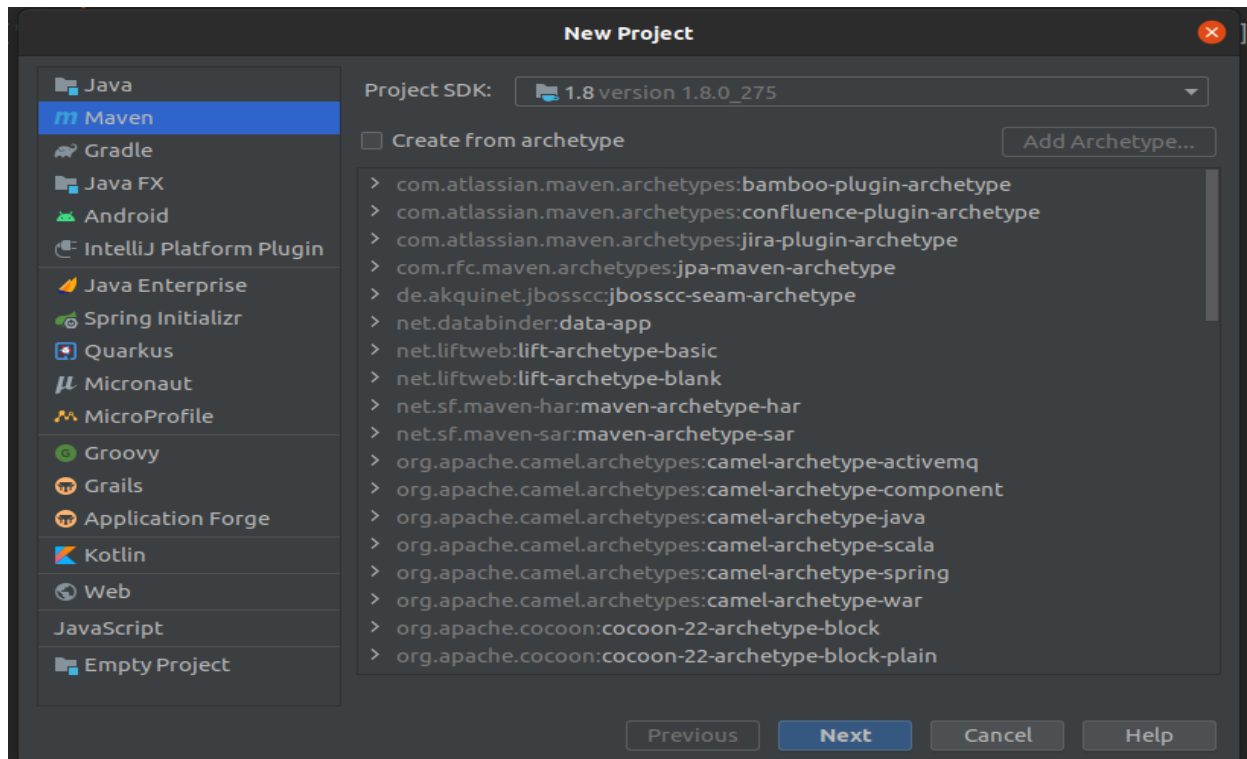


Fig 2.1: Creating a New Maven Project on IntelliJ

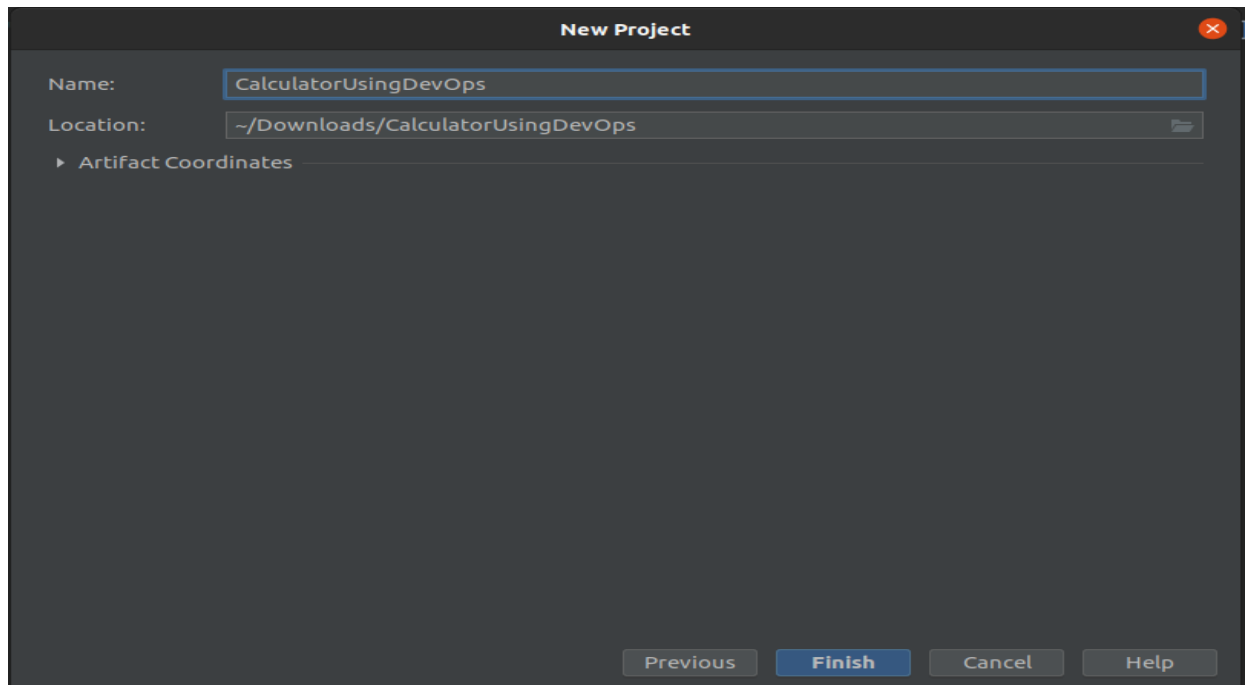
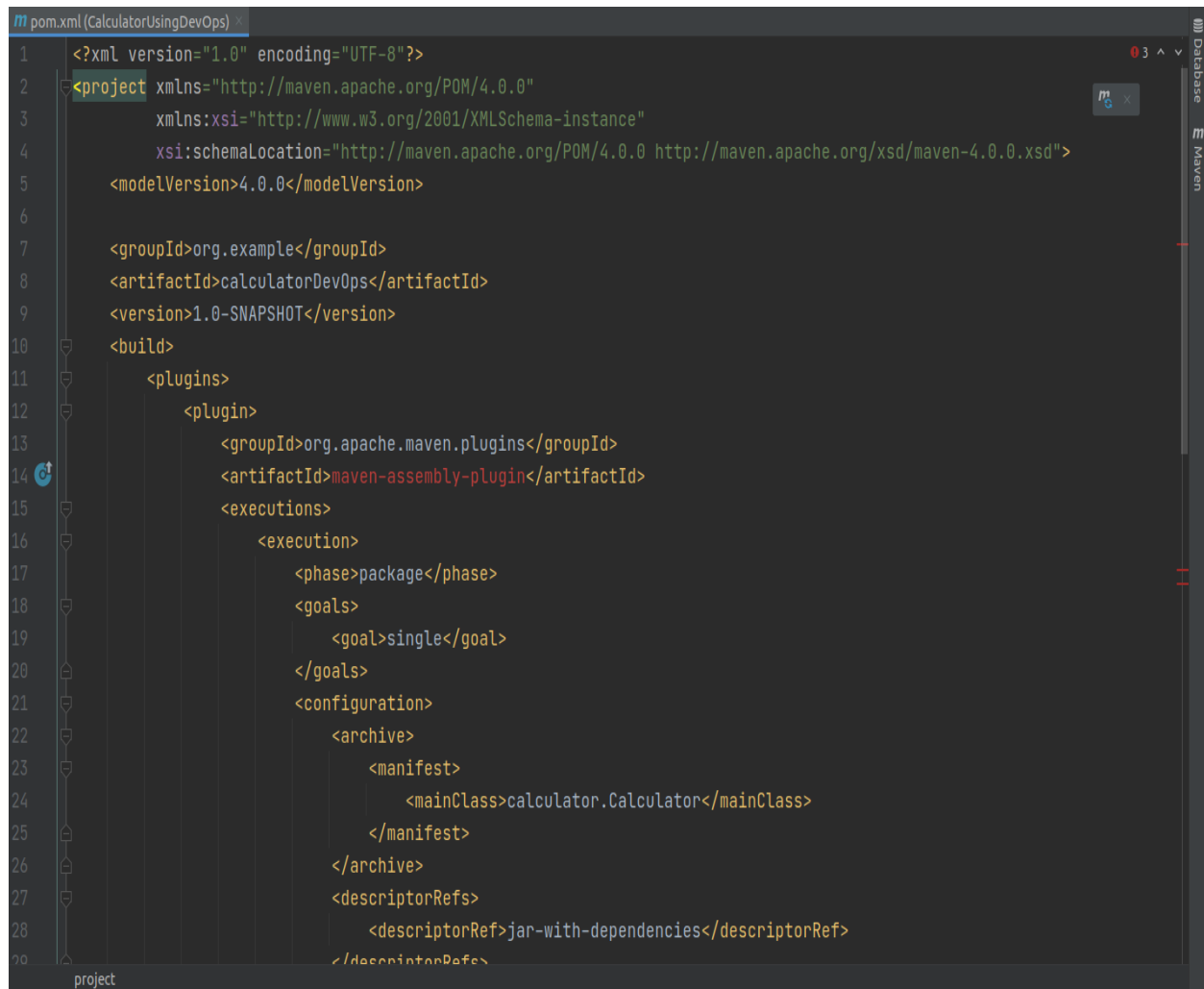


Fig 2.2: Naming the project



The screenshot shows an IDE window titled "pom.xml (CalculatorUsingDevOps)". The XML content is as follows:

```
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>calculatorDevOps</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</project>
```

The IDE interface includes a sidebar on the left with a file explorer showing a "project" folder, and a sidebar on the right with a "Database" and "Maven" panel. The main editor area has a dark theme with syntax highlighting.

Fig 2.3: Adding Dependencies to pom.xml

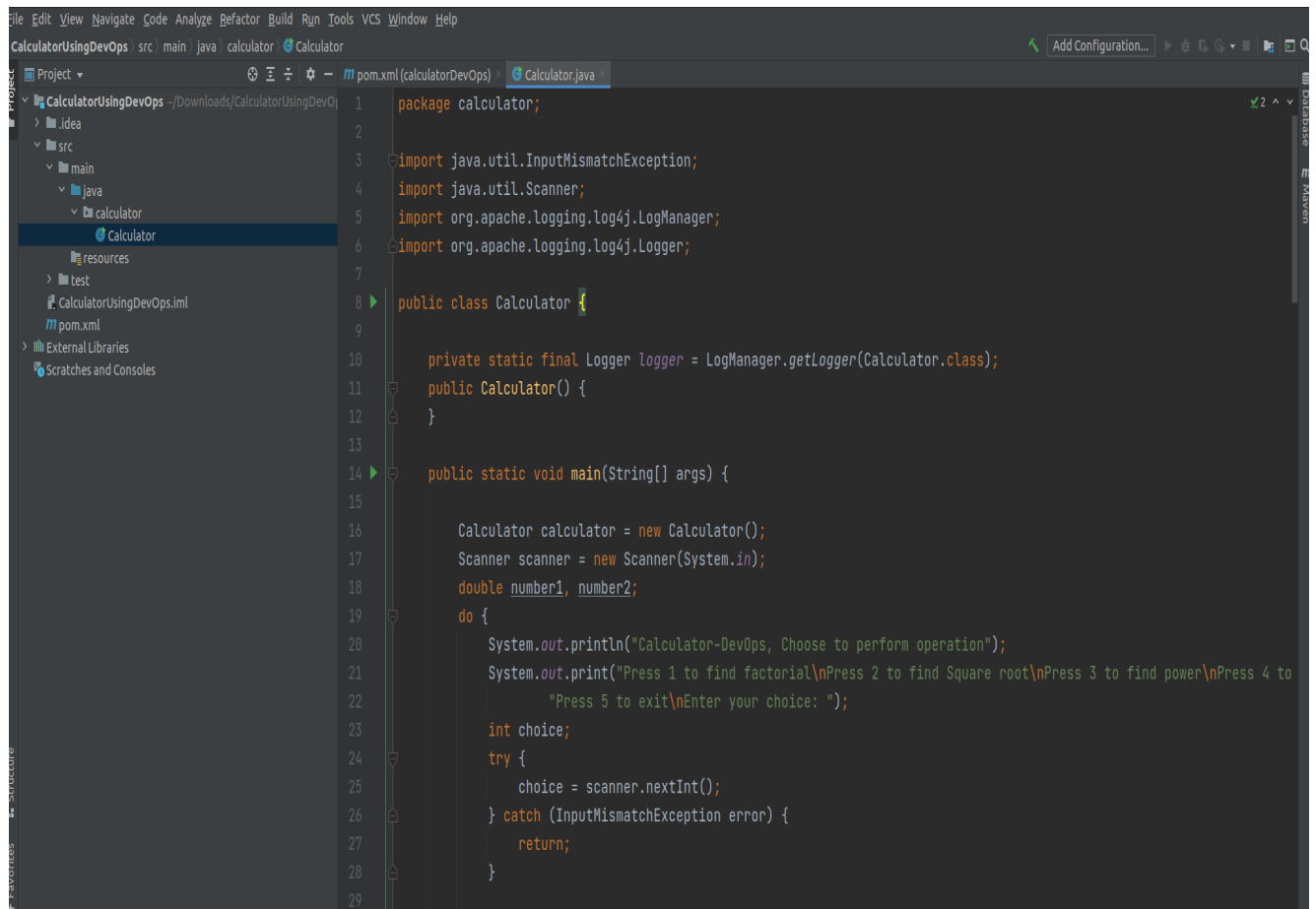


Fig 2.4: Writing the code for Scientific Calculator in calculator/Calculator.java

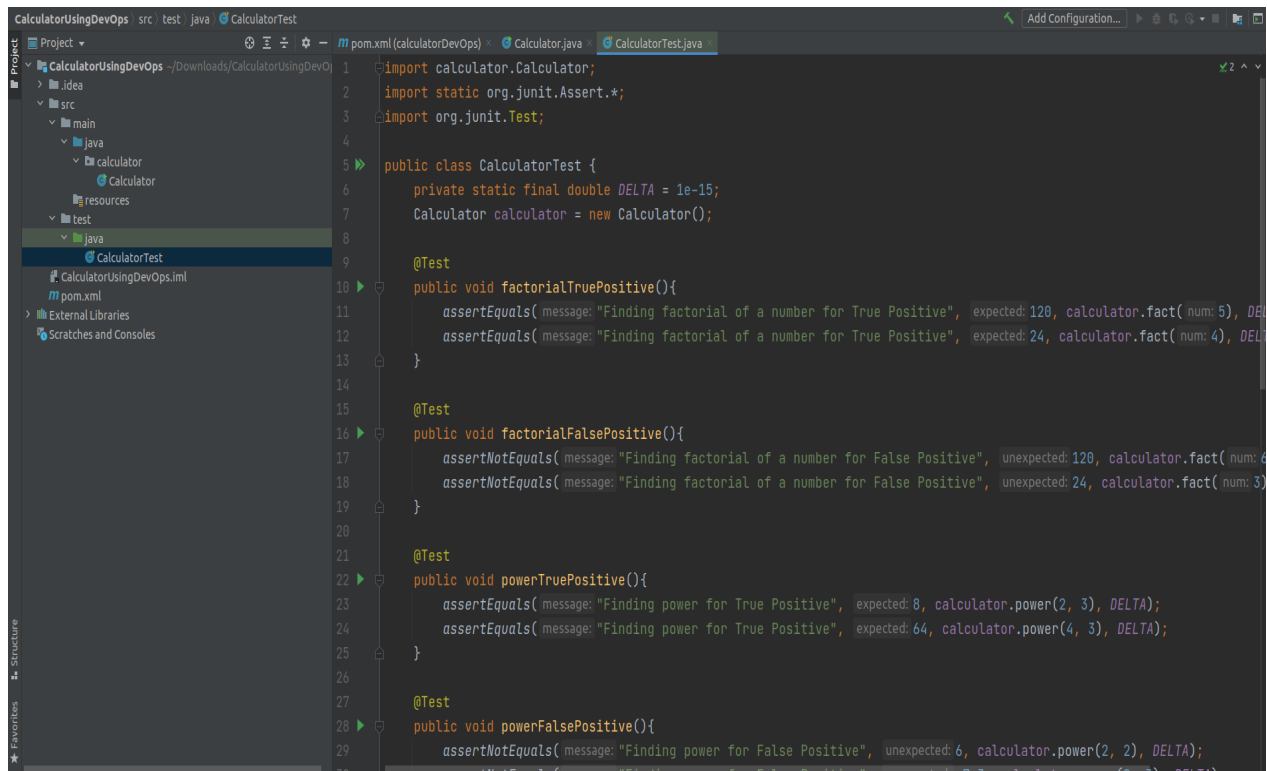


Fig 2.5: Writing Test Cases in CalculatorTest.java

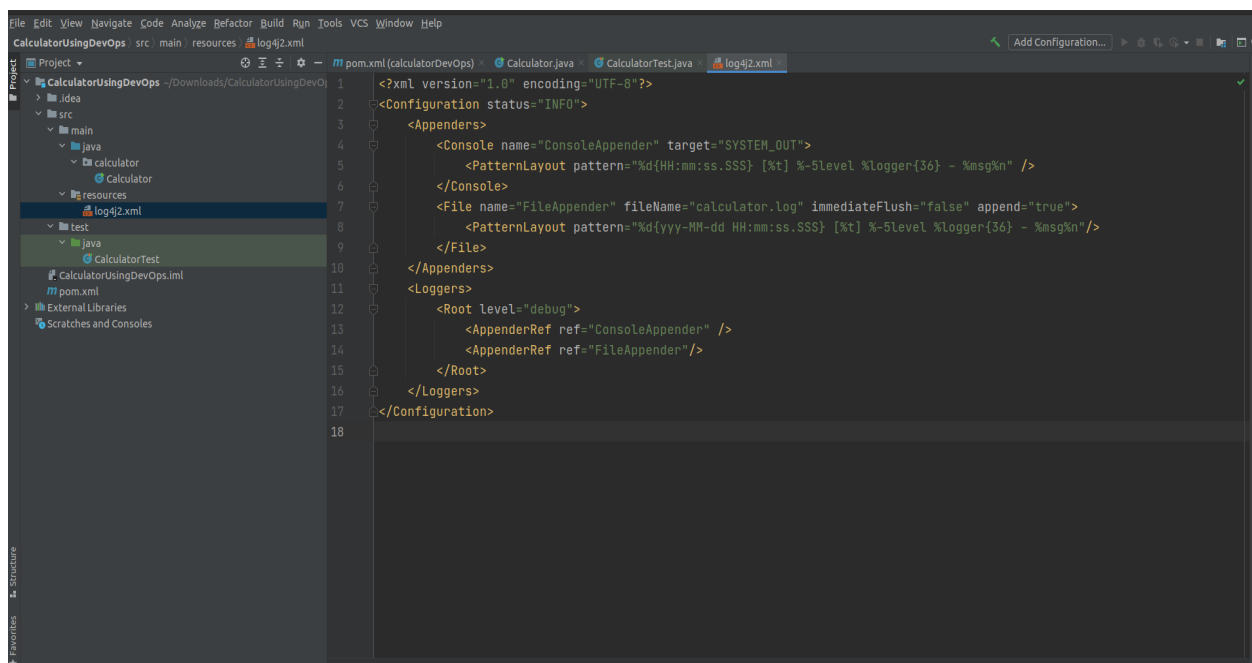


Fig 2.6: Making logger configuration file as log4j2.xml

```
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.595 s
[INFO] Finished at: 2021-03-14T15:40:03+05:30
[INFO] -----
```

Fig 2.7: Building using command *mvn clean package*

```
vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps$ cd target
vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps/target$ ls
archive-tmp          calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar  generated-sources    maven-archiver  surefire-reports
calculatorDevOps-1.0-SNAPSHOT.jar  classes          generated-test-sources  maven-status    test-classes
vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps/target$ java -jar calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar
Calculator-DevOps, Choose to perform operation
Press 1 to find factorial
Press 2 to find Square root
Press 3 to find power
Press 4 to find natural logarithm
Press 5 to exit
Enter your choice: 1
Enter a number : 3
15:42:26.115 [main] INFO  calculator.Calculator - [FACTORIAL] - 3.0
15:42:26.125 [main] INFO  calculator.Calculator - [RESULT - FACTORIAL] - 6.0
Factorial of 3.0 is : 6.0
```

Fig 2.8: Running the .jar file (with dependencies)

```

Terminal: Local < +
vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps$ git init
Initialised empty Git repository in /home/vaibhav/Downloads/CalculatorUsingDevOps/.git/
vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps$ git add .
vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps$ git commit -m "First commit"
[master (root-commit) f0007bf] First commit
 24 files changed, 468 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/compiler.xml
 create mode 100644 .idea/jarRepositories.xml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 CalculatorUsingDevOps.iml
 create mode 100644 calculator.log
 create mode 100644 pom.xml
 create mode 100644 src/main/java/calculator/Calculator.java
 create mode 100644 src/main/resources/log4j2.xml
 create mode 100644 src/test/java/CalculatorTest.java
 create mode 100644 target/calculator.log
 create mode 100644 target/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar
 create mode 100644 target/calculatorDevOps-1.0-SNAPSHOT.jar
 create mode 100644 target/classes/calculator/Calculator.class
 create mode 100644 target/classes/log4j2.xml
 create mode 100644 target/maven-archiver/pom.properties
 create mode 100644 target/maven-status/maven-compiler-plugin/compile/default-compile/createdFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/compile/default-compile/inputFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/createdFiles.lst
 create mode 100644 target/maven-status/maven-compiler-plugin/testCompile/default-testCompile/inputFiles.lst
 create mode 100644 target/surefire-reports/CalculatorTest.txt

vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps$ git remote add origin https://github.com/vtandonv/CalculatorUsingDevOps.git
vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps$ git remote -v
origin https://github.com/vtandonv/CalculatorUsingDevOps.git (fetch)
origin https://github.com/vtandonv/CalculatorUsingDevOps.git (push)

vaibhav@vaibhav:~/Downloads/CalculatorUsingDevOps$ git push origin --force HEAD:main
Username for 'https://github.com': vtandonv
Password for 'https://vtandonv@github.com':
Enumerating objects: 45, done.
Counting objects: 100% (45/45), done.
Delta compression using up to 4 threads
Compressing objects: 100% (32/32), done.
Writing objects: 100% (45/45), 1.76 MiB | 1.14 MiB/s, done.
Total 45 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/vtandonv/CalculatorUsingDevOps.git
+ 95073d4...f0007bf HEAD -> main (forced update)

```

Fig 2.9: Committing and pushing changes to GitHub using Git commands

main
1 branch
0 tags
Go to file
Add file
Code

vtandonv Create README.md
2af5da3 3 minutes ago 2 commits

.idea	First commit	13 minutes ago
src	First commit	13 minutes ago
target	First commit	13 minutes ago
CalculatorUsingDevOps.iml	First commit	13 minutes ago
README.md	Create README.md	3 minutes ago
calculator.log	First commit	13 minutes ago
pom.xml	First commit	13 minutes ago

README.md

CalculatorUsingDevOps

Building a Java based Scientific Calculator application using Maven build tool, integrating and developoing using Jenkins pipeline and finally deploying it over a host machine using Ansible

About

Building a Java based Scientific Calculator application using Maven build tool, integrating and developoing using Jenkins pipeline and finally deploying it over a host machine using Ansible

Readme

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Contributors 2

vaibhtan
 vtandonv Vaibhav Tandon

Fig 2.10: Changes visible on GitHub

Pipeline script for Maven Build:

```
stage('Maven Build') {
    steps {
        script {
            sh 'mvn clean install'
        }
    }
}
```

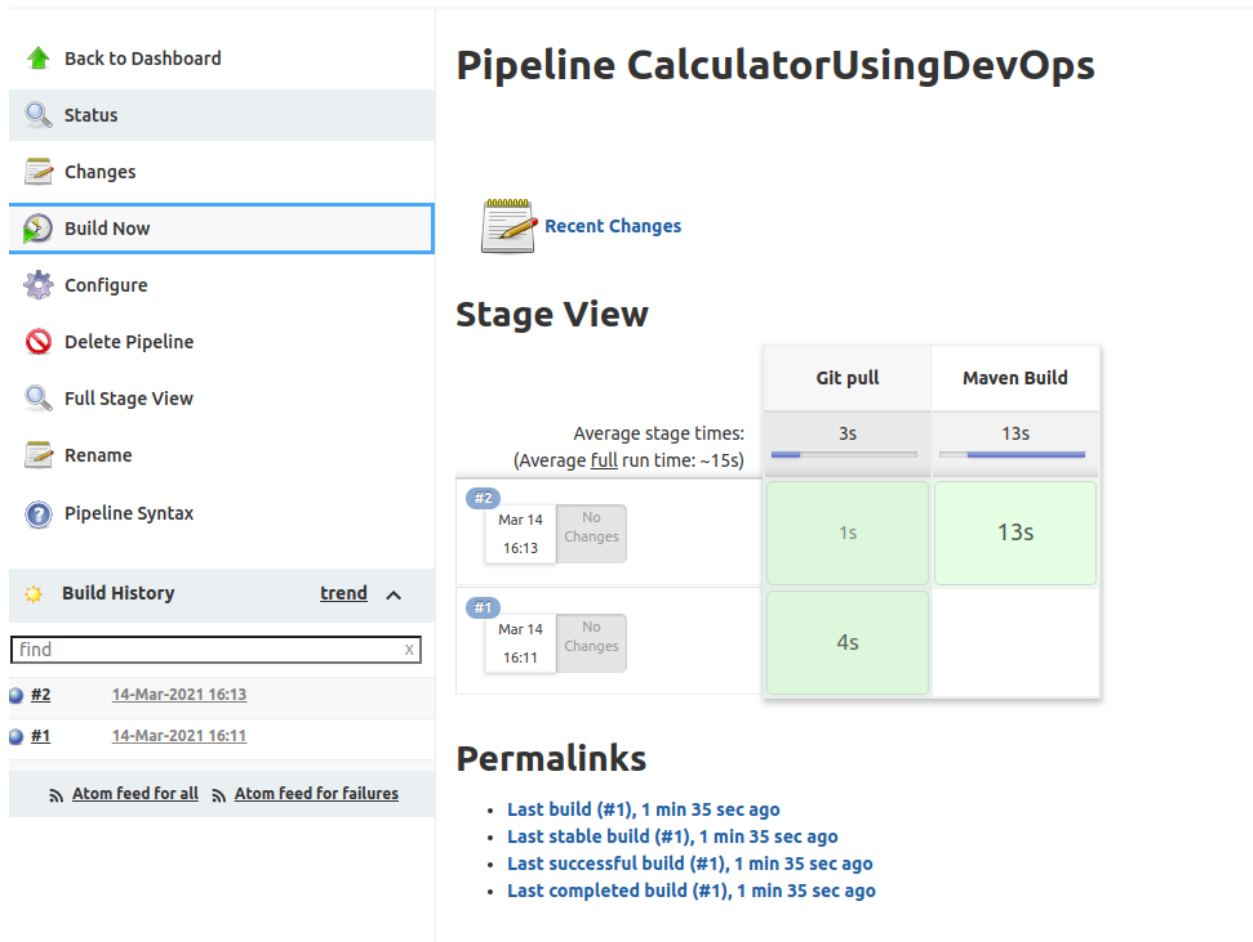


Fig 2.11: Successfully built Stage 2 of Pipeline

Stage 3: Docker Build to image

This step involves containerizing .jar file in a Docker image. For this we first create a Dockerfile in our project and add the name of our .jar file accordingly i.e.

FROM openjdk:8

COPY ./target/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar ./

WORKDIR ./

CMD ["java", "-jar", "calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar"]

We then push this file to our Github account

Pipeline script for this stage is:

```
stage('Docker Build to Image') {  
    steps {  
        script {  
            imageName = docker.build 'vtandonv/calculatorusingdevopslatest'  
        }  
    }  
}
```


[Back to Dashboard](#)
[Status](#)
[Changes](#)
[Build Now](#)
[Configure](#)
[Delete Pipeline](#)
[Full Stage View](#)
[Rename](#)
[Pipeline Syntax](#)

Build History [trend](#) ^

#3	14-Mar-2021 16:28
#2	14-Mar-2021 16:13
#1	14-Mar-2021 16:11

[Atom feed for all](#) [Atom feed for failures](#)

Pipeline CalculatorUsingDevOps

 **Recent Changes**

Stage View

Average stage times:
(Average full run time: ~28s)

	Git pull	Maven Build	Docker Build to Image
#3 Mar 14 16:28 1 commit	2s	9s	30s
#2 Mar 14 16:13 No Changes	1s	13s	
#1 Mar 14 16:11 No Changes	4s		

Permalinks

- [Last build \(#2\), 15 min ago](#)
- [Last stable build \(#2\), 15 min ago](#)
- [Last successful build \(#2\), 15 min ago](#)
- [Last completed build \(#2\), 15 min ago](#)

Fig 3.1: Successfully containerized .jar file to Docker Image

Stage 4: Pushing Docker image

In this step we push the image to the Docker hub.

Pipeline Script for this stage is:

```
stage('Push Docker Image') {  
    steps {  
        script {  
            docker.withRegistry("", 'docker-jenkins'){  
                imageName.push()  
            }  
        }  
    }  
}
```

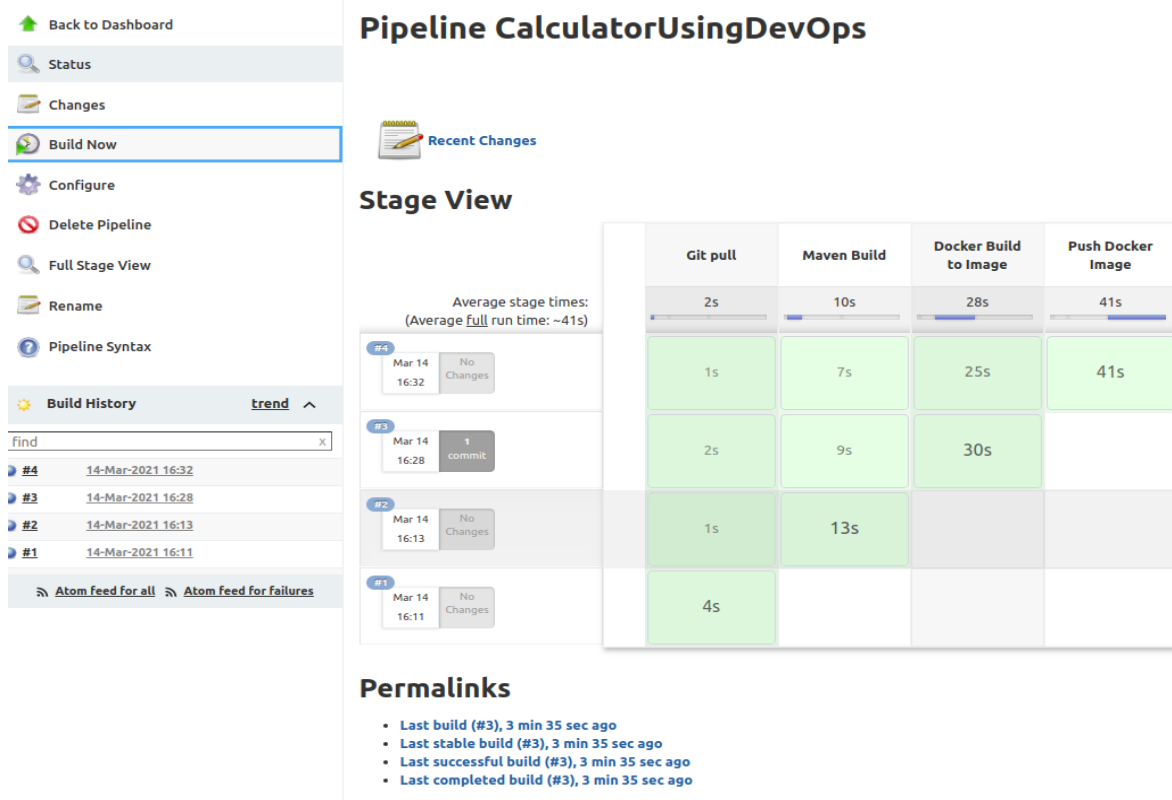


Fig 4.1: Pushed Docker image to Docker Hub

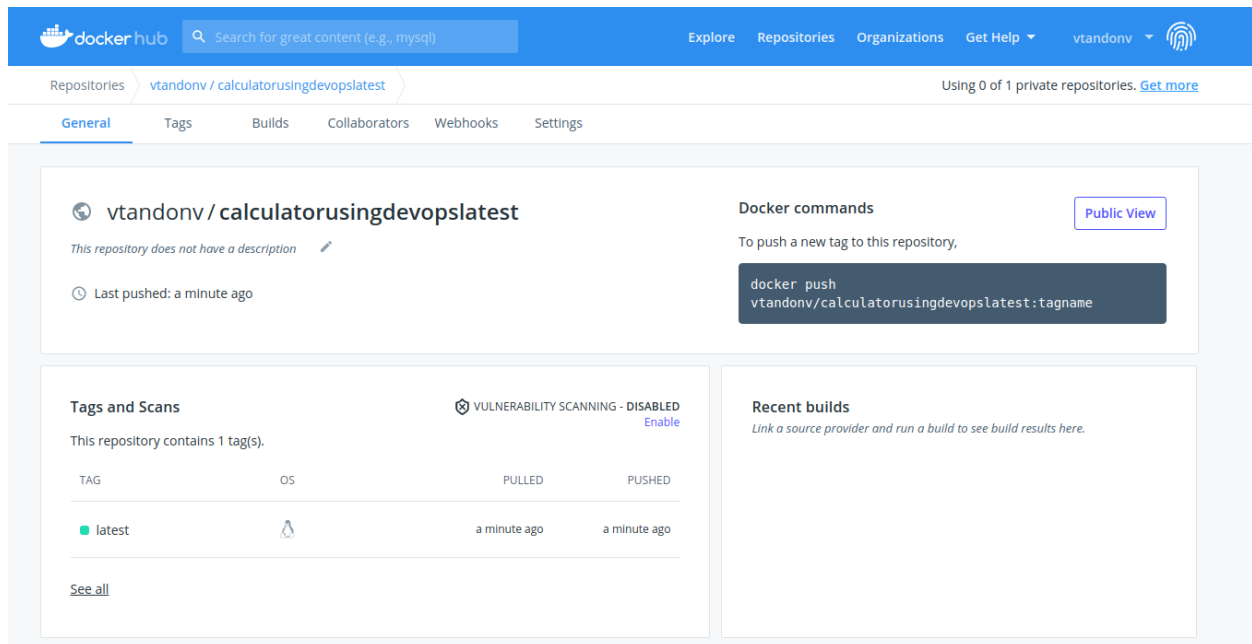


Fig 4.2: Changes reflected to Docker Hub Repository

Stage 5: Ansible pull Docker image

In this step we install Ansible on the controller node and deploy our docker image in all our hosts. In this step we create a directory named `deploy-docker` in which we first create a file named `calc-deploy.yml` which contains the instructions required to deploy the docker image to our hosts.

calc-deploy.yml

- name: Pull docker image of Calculator

hosts: all

tasks:

- name: Pull image

docker_image:

name: vtandonv/calculatorusingdevopslatest

source: pull

After that in the same directory we add our inventory file which contains the details of our hosts i.e. their name and IP address.

inventory

[ubuntu]

172.16.129.220 ansible_user=vtandonv

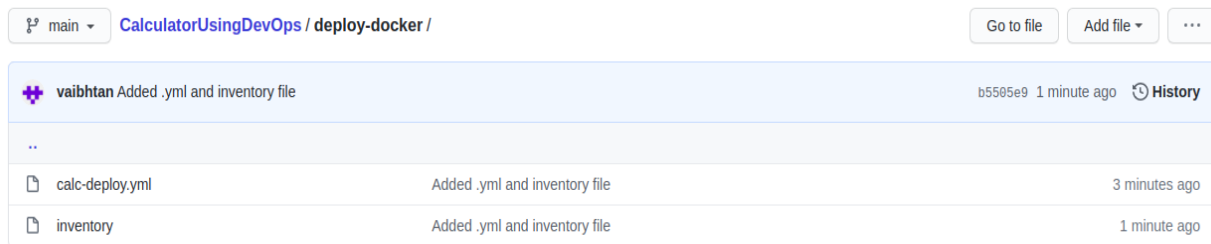


Fig 5.1: Added .yaml and inventory file to the Github account

Now we add our Jenkins pipeline syntax for Ansible i.e.

```
stage('Ansible pull docker image') {
```

```
    steps { ansiblePlaybook becomeUser: null, colored: true, disableHostKeyChecking: true,
installation: 'Ansible', inventory: 'deploy-docker/inventory', playbook:
'deploy-docker/calc-deploy.yml', sudoUser: null
```

```
    }
```

```
}
```

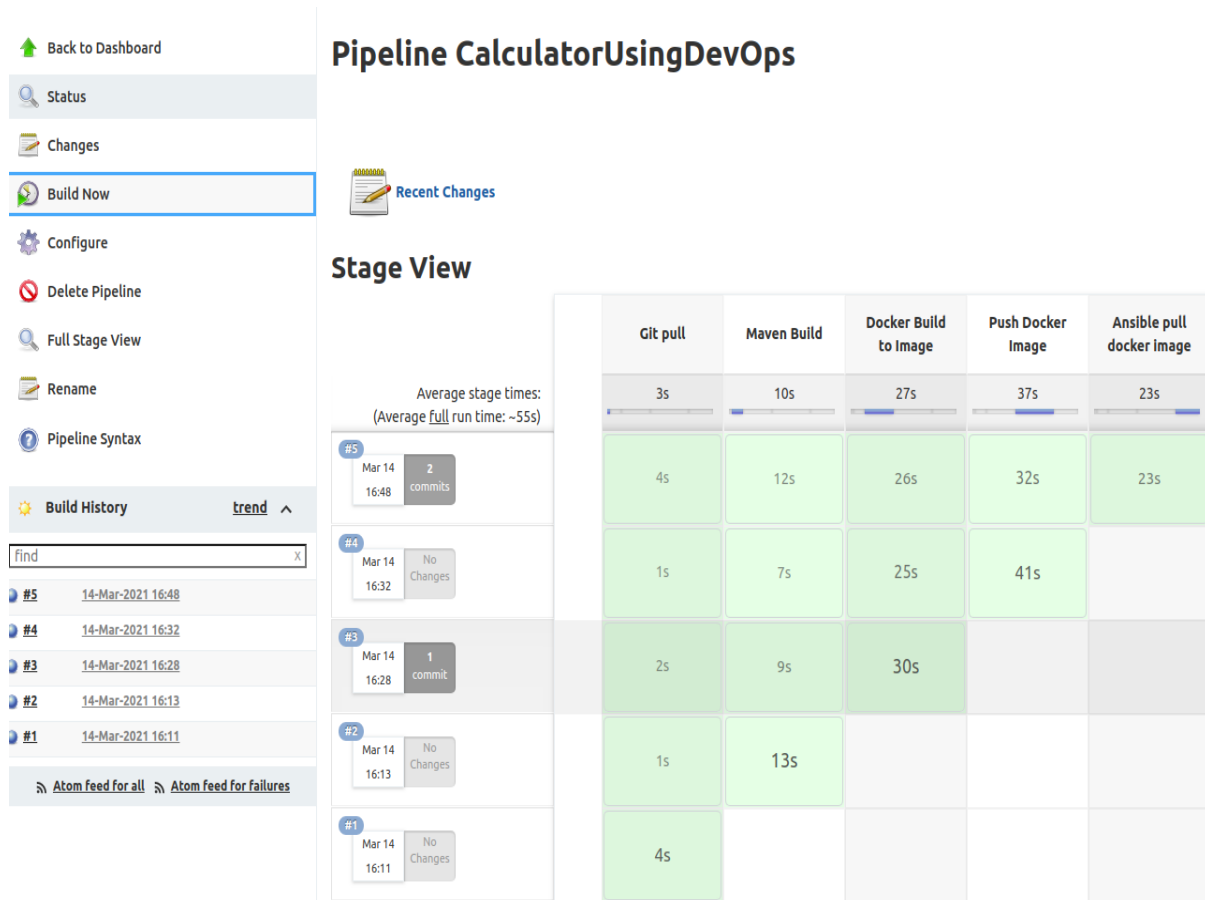
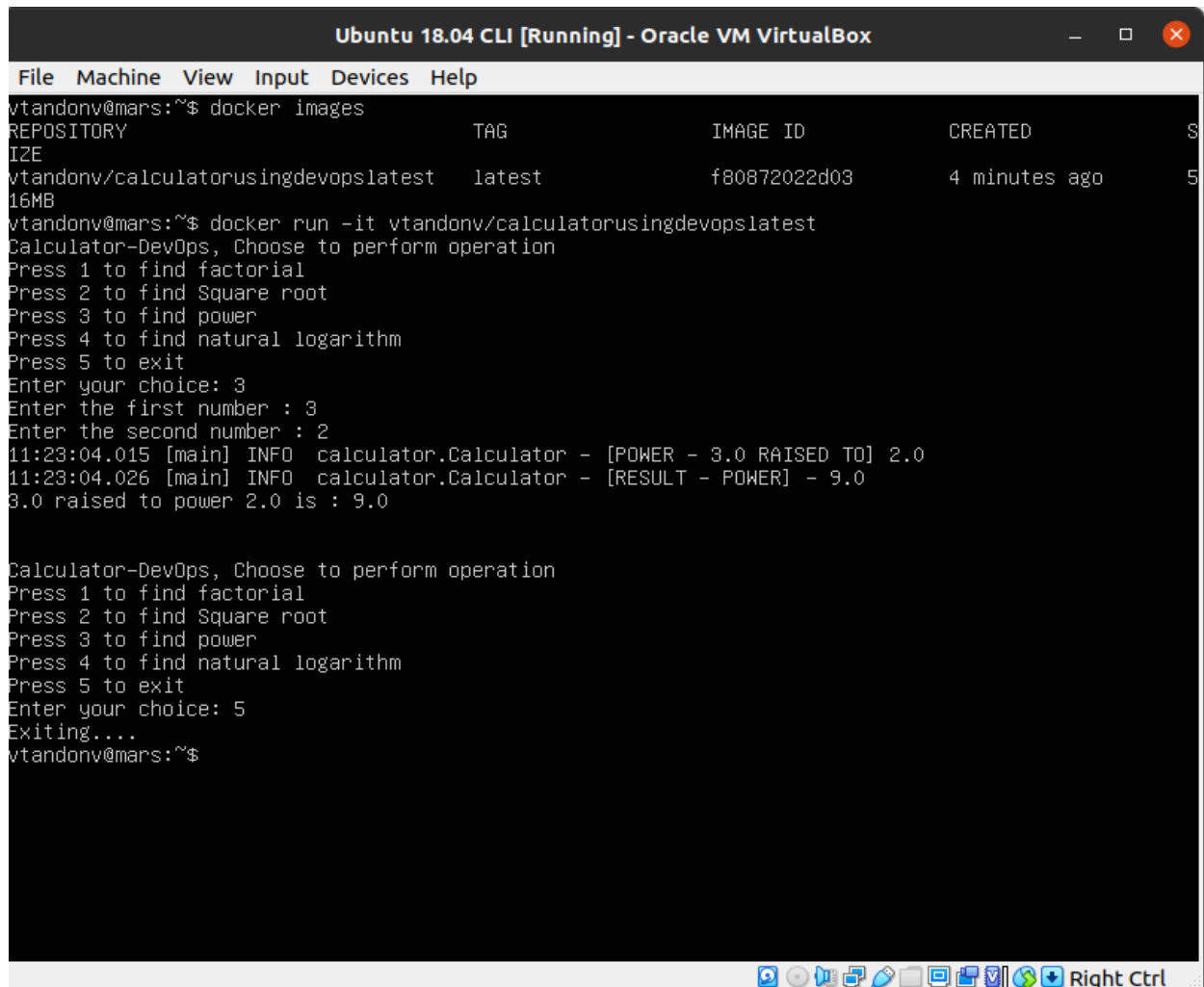


Fig 5.2: Docker image successfully deployed to our host



```
vtandonv@mars:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
vtandonv/calculatorusingdevopslatest  latest             f80872022d03       4 minutes ago      516MB
vtandonv@mars:~$ docker run -it vtandonv/calculatorusingdevopslatest
Calculator-DevOps, Choose to perform operation
Press 1 to find factorial
Press 2 to find Square root
Press 3 to find power
Press 4 to find natural logarithm
Press 5 to exit
Enter your choice: 3
Enter the first number : 3
Enter the second number : 2
11:23:04.015 [main] INFO  calculator.Calculator - [POWER - 3.0 RAISED TO] 2.0
11:23:04.026 [main] INFO  calculator.Calculator - [RESULT - POWER] - 9.0
3.0 raised to power 2.0 is : 9.0

Calculator-DevOps, Choose to perform operation
Press 1 to find factorial
Press 2 to find Square root
Press 3 to find power
Press 4 to find natural logarithm
Press 5 to exit
Enter your choice: 5
Exiting....
vtandonv@mars:~$
```

Fig 5.3: .jar file running on Docker image pulled from the Docker hub on our host

Continuous Monitoring using ELK (ElasticSearch - Logstash - Kibana)

"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.

Sign Up for 15 days trial on Elastic Cloud and then create your first deployment

Create deployment

What do you want to do with your data?


Put your data to work for you with a pre-configured solution. [Learn more](#)



Elastic Stack

Deploy Elasticsearch and Kibana. Search, analyze, and visualize data from any source, in any format.


✓ Selected



Elastic Enterprise Search

Easily implement powerful search experiences for your website, app, or workplace with refined APIs and tools.


Select



Elastic Observability

Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.

Select



Elastic Security

Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.

Select

Select hardware profile

☒ **I/O Optimized** Recommended

Use for all-purpose workloads, including time-series data like logs and metrics. [See details](#)

☐ **Compute Optimized**

Run CPU-intensive workloads or run smaller workloads cost-effectively when you need less memory and storage. [See details](#)

☐ **Memory Optimized**

Perform memory-intensive operations efficiently, including workloads with frequent aggregations. [See details](#)

Deployment settings


Choose the cloud provider, region, and Elastic Stack version.

 Google Cloud  us-central1 (Iowa) v7.11.2 Expand >

Name your deployment

You can always change this later.

Calculator-ELK

 Customize

✓ Create deployment

Equivalent API request

Fig: Creating deployment

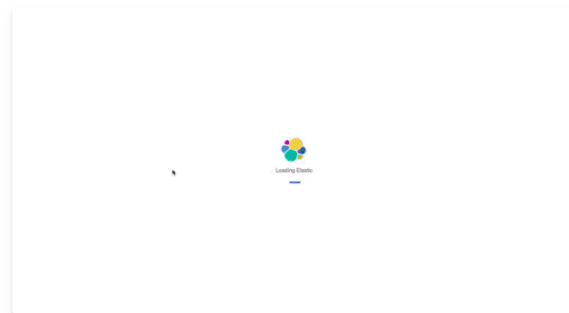
✓ Your deployment has been created.

Get started with your deployment

The next step is to ingest data and create visualizations in Kibana.

[Open Kibana](#)

Forgot to save your credentials? [Reset your deployment password](#)



Deployment name

Calculator-ELK

[Edit](#)

Deployment ID: 9cac22e

Deployment version

v7.11.2

Deployment status

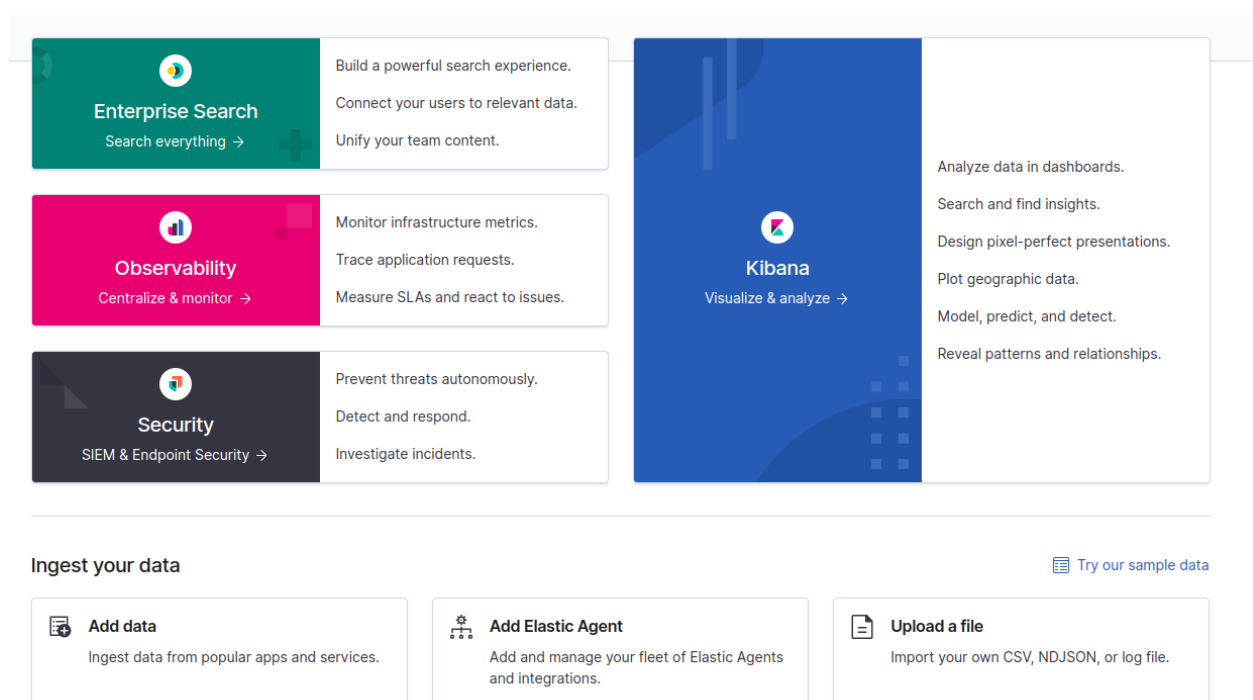
● Healthy

[Open Kibana](#)

[Manage](#)

Fig: Deployment Created

Click on Upload a file



The dashboard displays three main sections for data ingestion:

- Enterprise Search**: Build a powerful search experience. Connect your users to relevant data. Unify your team content. [Search everything →](#)
- Observability**: Monitor infrastructure metrics. Trace application requests. Measure SLAs and react to issues. [Centralize & monitor →](#)
- Security**: Prevent threats autonomously. Detect and respond. Investigate incidents. [SIEM & Endpoint Security →](#)

Kibana: Visualize & analyze →

Analyze data in dashboards.
Search and find insights.
Design pixel-perfect presentations.
Plot geographic data.
Model, predict, and detect.
Reveal patterns and relationships.

Ingest your data [Try our sample data](#)

- Add data**: Ingest data from popular apps and services.
- Add Elastic Agent**: Add and manage your fleet of Elastic Agents and integrations.
- Upload a file**: Import your own CSV, NDJSON, or log file.

Fig: Upload a the calculator.log file from target folder

elastic

Machine Learning / Data Visualizer / File

Overview Anomaly Detection Data Frame Analytics **Data Visualizer** Settings

calculator.log

Import data EXPERIMENTAL

Simple Advanced

Index name

calculator_logs

☒ Create index pattern

Import

Fig: Assign an index name

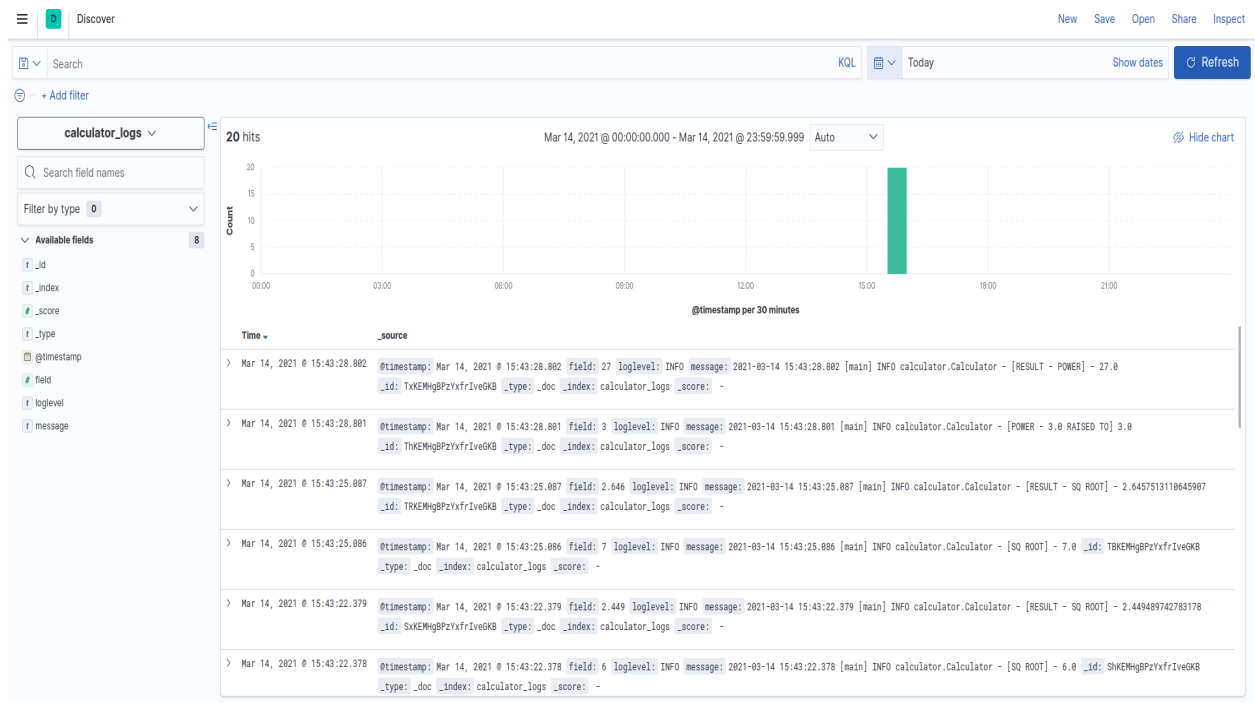


Fig: Visualization using Kibana

Challenges Faced:

1. Log file not getting generated

Solution: Added log4j.api and log4j.core dependencies to pom.xml and built the artifact using *mvn clean package* instead of *mvn clean install* and finally ran the jar file with dependencies.

2. Error in pushing docker image to Ansible remote user

Solution: Installed pip ,pip3-docker and docker on both local and remote server. Added public-private RSA key using ssh-copy-id command so that we can login into the remote server from the host user without any password.