CS 816 - Software Production Engineering (SPE)

Mini Project - Scientific Calculator with DevOps

Name : Akshay Girish Thite
Roll No. : MT2021010
Batch : MTech CSE

Problem Statement :

Create a scientific calculator program with user menu driven operations

- Square root function - √x
- Factorial function - x!
- Natural logarithm (base e) - ln(x)
- Power function - x

Introduction :

In this project we implement a scientific calculator with the operations such as square root, factorial, natural logarithm and power function using DevOps tools. It is a command line menu driven application.
DevOps - DevOps can be defined as a set of practices that combine software development and IT operations. The main aim of DevOps is shortening the systems development life cycle and providing continuous delivery with high software quality.

Tools and Technology used :

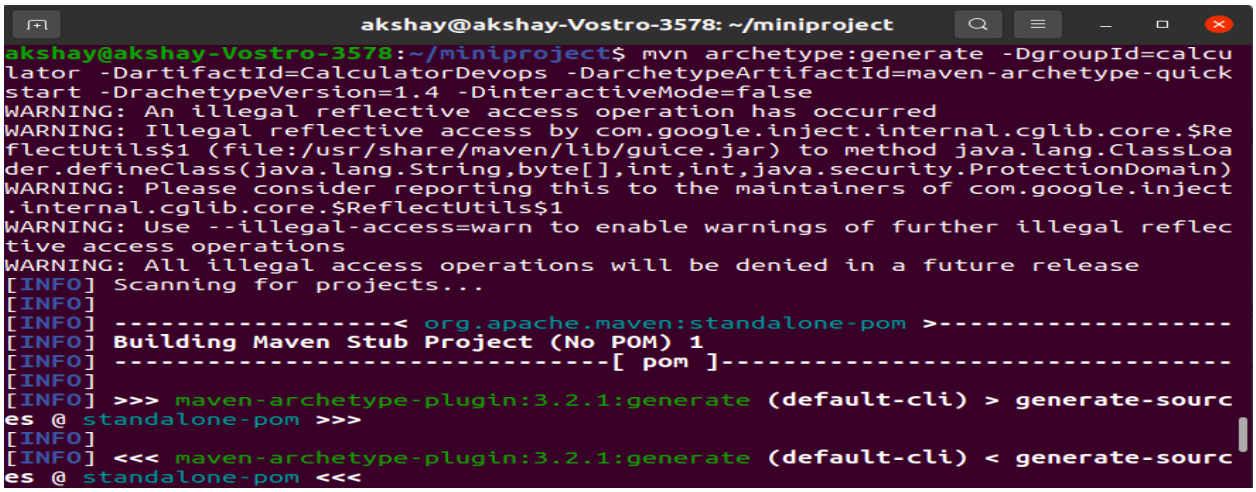
1. Programming language : Java
2. Testing : JUnit
3. Building tool : Apache Maven
4. Source Code Management : GitHub for Git
   https://github.com/AkshayThite10/CalculatorDevops.git
5. Containerization : Docker
6. Continuous Integration : Jenkins
7. Continuous Deployment : Ansible
8. Generating Logs : Log4j
9. Monitoring : ELK Stack

Building Tool (Apache Maven) :

Maven is a powerful project management tool. It is based on the project object model (POM). Project build, dependency and documentation can be done using maven. It is basically a tool that is used for building and managing any Java based application.
In this project, maven is used to build the java source code by configuring all the dependencies and creating a binary executable .jar file of it in the target directory.
“pom.xml” is a configuration file in a maven project. It contains and manages the dependencies, plugins and metadata for the project.

Steps in building :

1. In the project directory, create maven project hierarchy



Command : mvn archetype:generate -DgroupId=calculator -DartifactId=CalculatorDevops -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -DinteractiveMode=false

2. Writing the calculator java program in App.java and JUnit testing program in AppTest.java.

Functions of App.java :

```
    public static double sqroot(double num) {
        double res;
        res = Math.sqrt(num);
        System.out.print("\n");
        System.out.println("Square root = "+res);
        System.out.print("\n");
        logger.info("Square root computed");
        return res;
    }
    public static int fact(int num){
        int res=num,i;
        if(num==0)
        {
            res=1;
        }
        for(i=num-1;i>=1;i--)
        {
            res=res*i;
        }
        System.out.print("\n");
        System.out.println("Factorial = "+res);
        System.out.print("\n");
        logger.info("Factorial computed");
        return res;
    }
    public static double natlog(double num){
        double res = Math.log(num);
        System.out.print("\n");
        System.out.println("Natural log = "+ res);
        System.out.print("\n");
        logger.info("Natural log computed");
        return res;
    }
    public static double powerfxn(double num, double exponent){
        double res = Math.pow(num,exponent);
        System.out.print("\n");
        System.out.println("Power = "+ res);
        System.out.print("\n");
        logger.info("Power computed");
        return res;
    }
}
```

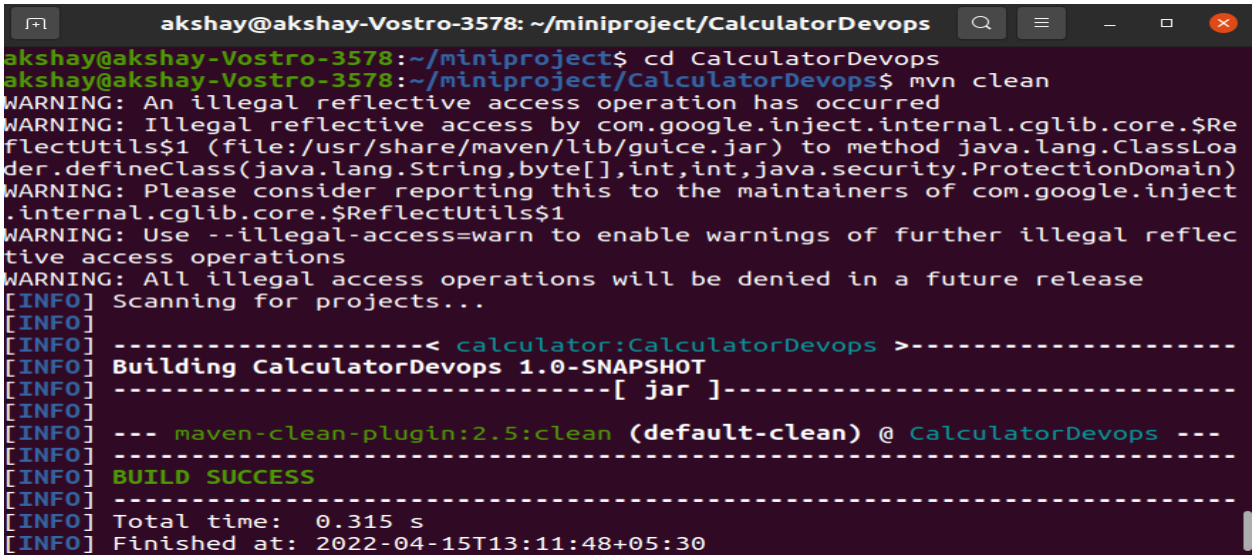
AppTest.java :

```
// package calculator;
import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;

public class AppTest {
    @Test
        public void test_squareRoot() {
            double a = 81.0;
            double expectedResult = 9.0;
            double result = App.sqroot(a);
            Assert.assertEquals(expectedResult, result, 0.0f);
        }
```

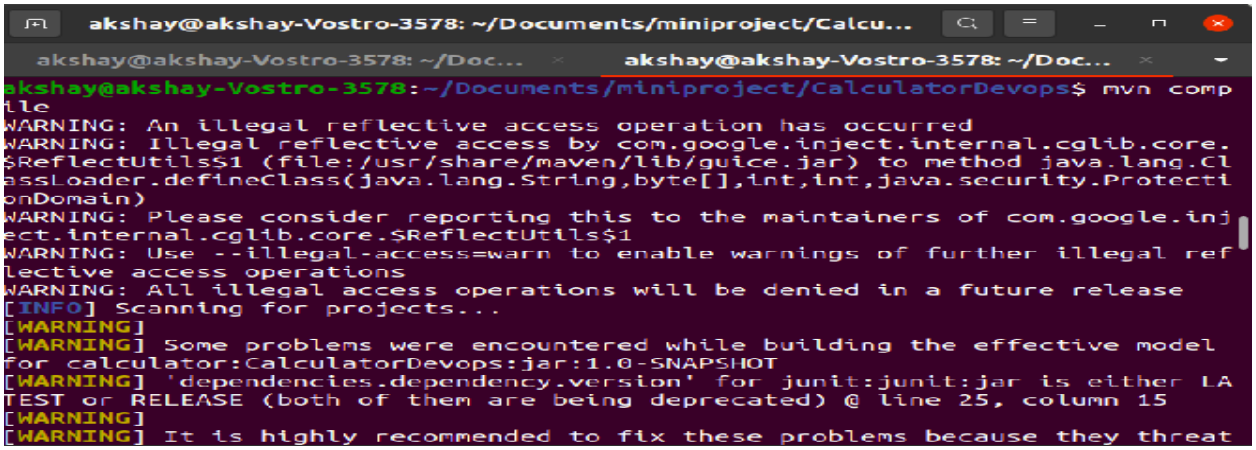
```
15      @Test
16          public void test_factorial() {
17              int a = 4;
18              int expectedResult = 24;
19              int result = App.fact(a);
20              Assert.assertEquals(expectedResult, result);
21          }
22
23      @Test
24          public void test_naturalLog() {
25              double a = 2.718;
26              double expectedResult = 1.0;
27              double result = App.natlog(a);
28              Assert.assertEquals(expectedResult, result, 0.2f);
29          }
30
31      @Test
32          public void test_power() {
33              double a = 2.0;
34              double b = 10.0;
35              double expectedResult = 1024.0;
36              double result = App.powerfxn(a,b);
37              Assert.assertEquals(expectedResult, result, 0.0f);
38          }
39
40  }
```

3. Maven clean, compile, install and site



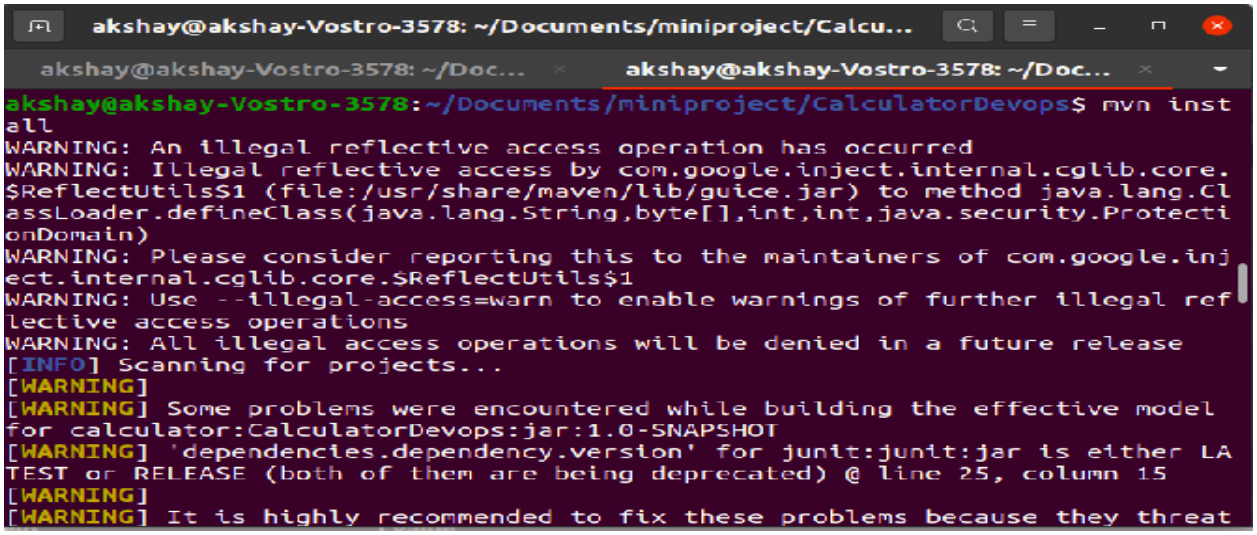
Command : mvn clean

It is used for cleaning the project hierarchy. It cleans cache in the maven hierarchy. If there are any previous builds in the ‘target’ directory then they would get deleted. Now, the new build can take place.



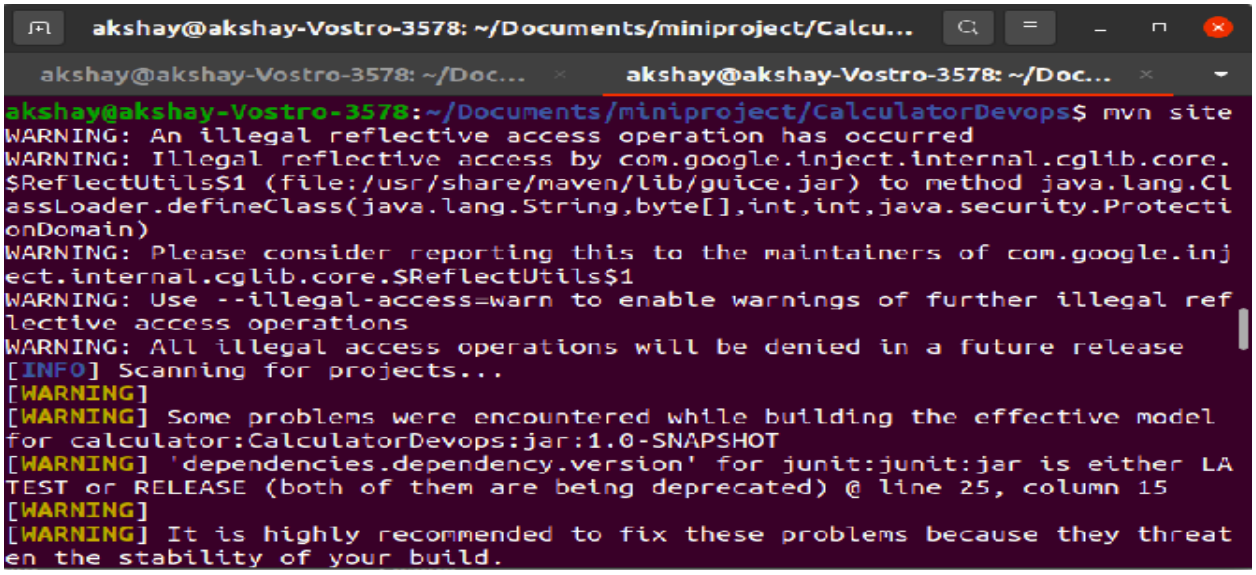
Command : mvn compile

It compiles all the source code in the ‘src/main/java/<packagename>’ directory.



Command : mvn install

It is used for building the project. It creates a binary executable .jar file of the project in the ‘target’ folder.

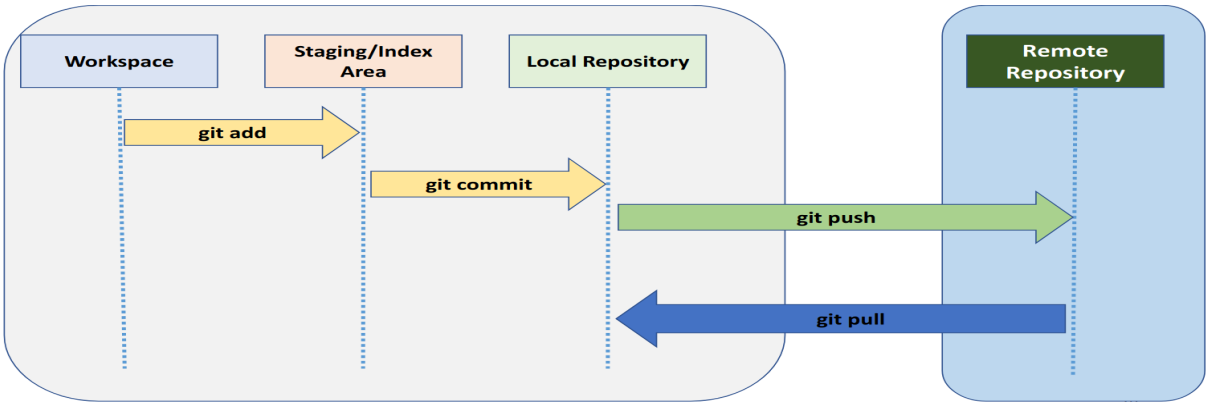


Command : mvn site

It creates a documentation type report using HTML and CSS in the ‘target/site’ directory.

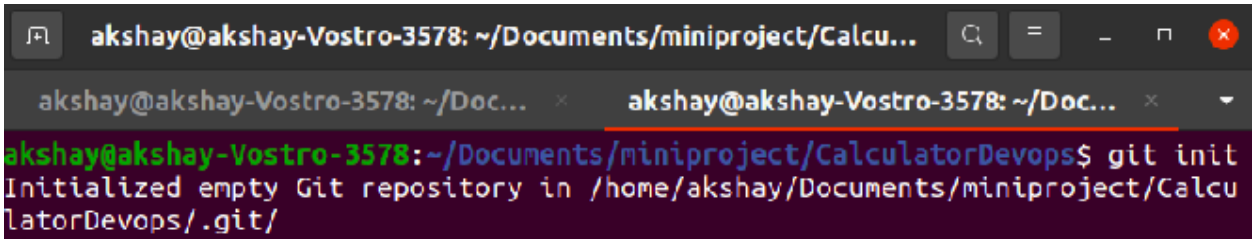
Source Code Management :

It is used to track modifications to the source code repository. It tracks the history of changes to the code base. It also helps to resolve conflicts when merging updates from multiple contributors. Source Code Management is also known as the Version Control System as it stores the various versions of the project which may be useful later.



Steps to push the Maven project from the workspace to the remote GitHub repository :

1. Initializing Git in the current workspace.

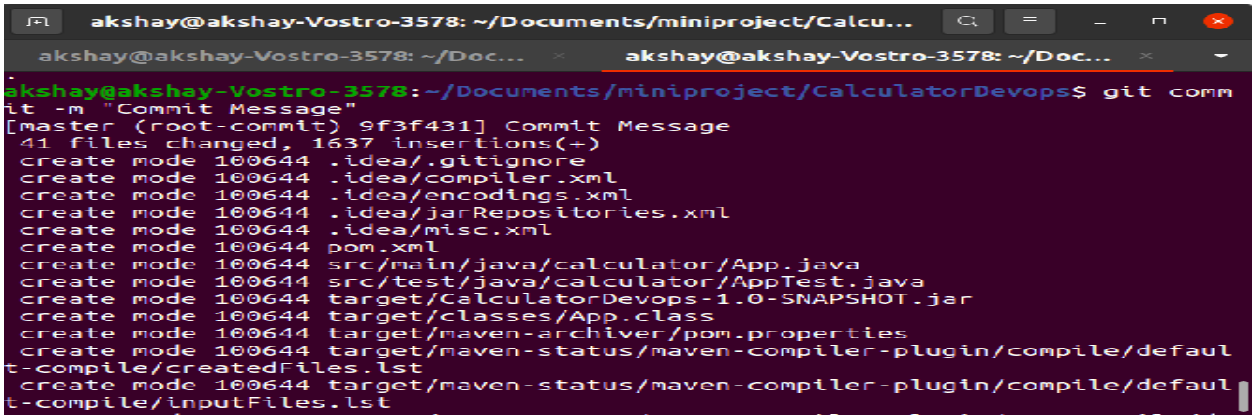


Command : git init

2. Adding the project to the staging area.

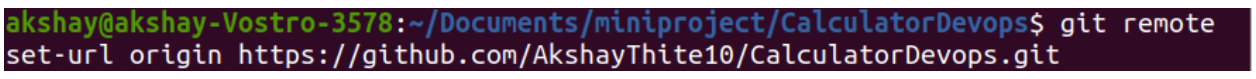
Command : git add .

3. Committing the changes to the local git repository.



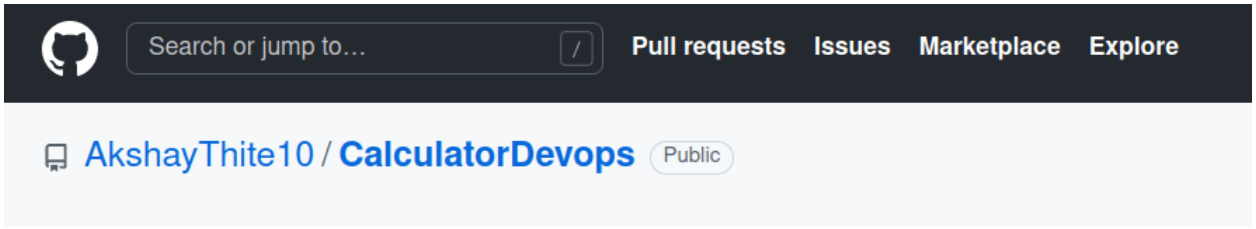
Command : git commit -m “Commit Message”

4. Setting an origin for a remote repository on GitHub.



Command : git remote add origin “url of the remote repository” (if origin not set earlier)
git remote set-url origin “url of the remote repository” (if origin was set)

5. Pushing the project to the remote repository to the ‘master’ branch.




```
akshay@akshay-Vostro-3578:~/Documents/miniproject/CalculatorDevops$ git push -u
 origin master
Username for 'https://github.com': AkshayThite10
Password for 'https://AkshayThite10@github.com':
Branch 'master' set up to track remote branch 'master' from 'origin'.
Everything up-to-date
```

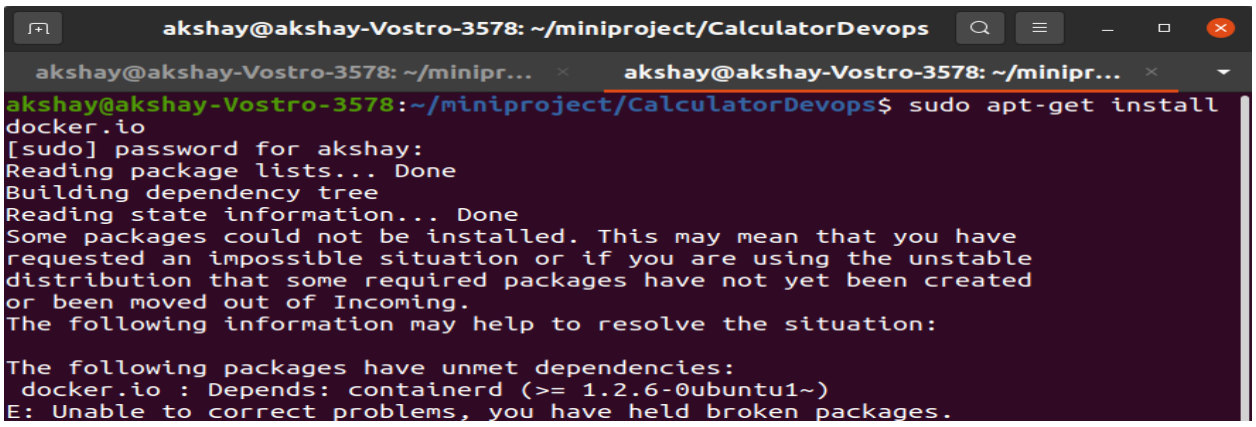
Command : git push -u origin master

Containerization (Docker) :

Docker is an open platform that is used for developing, shipping and running applications. It separates application from infrastructure. So, delivery of software is faster. It is possible to manage infrastructure in the same way as that of application.

Steps :

1. Install Docker.



Command : sudo apt-get install docker.io

2. Creating a repository in dockerhub.

Link : https://hub.docker.com/r/akshaythite10/calculator_devops

3. Creating a Dockerfile

Docker automatically builds the images according to the instructions present in the Dockerfile. Dockerfile is a file that contains all the commands to build an image. Here, we need to create a container that executes the built maven project.

Dockerfile

```
1  FROM openjdk:8
2  COPY ./target/CalculatorDevops-1.0-SNAPSHOT-jar-with-dependencies.jar ./
3  WORKDIR ./
4  CMD ["java", "-cp","CalculatorDevops-1.0-SNAPSHOT-jar-with-dependencies.jar","App"]
5
```

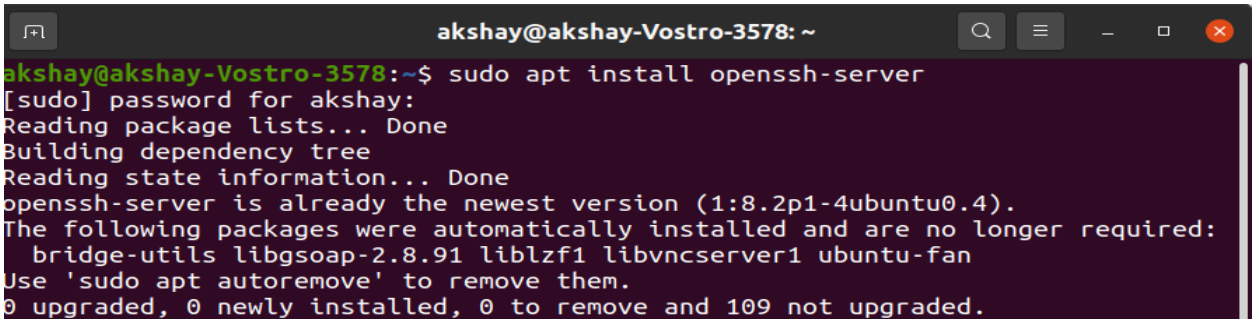
After creating this Dockerfile, push it to the GitHub repository.
After running docker build using this Dockerfile as the source will create the required docker image that is ready to run on our application. Docker image needs to be built and then to be pushed to the DockerHub.

Continuous Integration (Jenkins) :

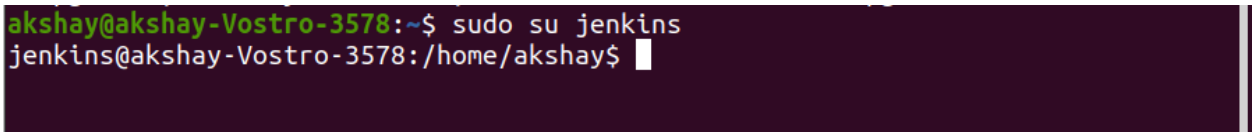
Steps :

1. Add Jenkins to the Docker group.

Command : sudo apt install openssh-server
This will install the openssh server.



Command : sudo su jenkins
It is used for logging in to jenkins.



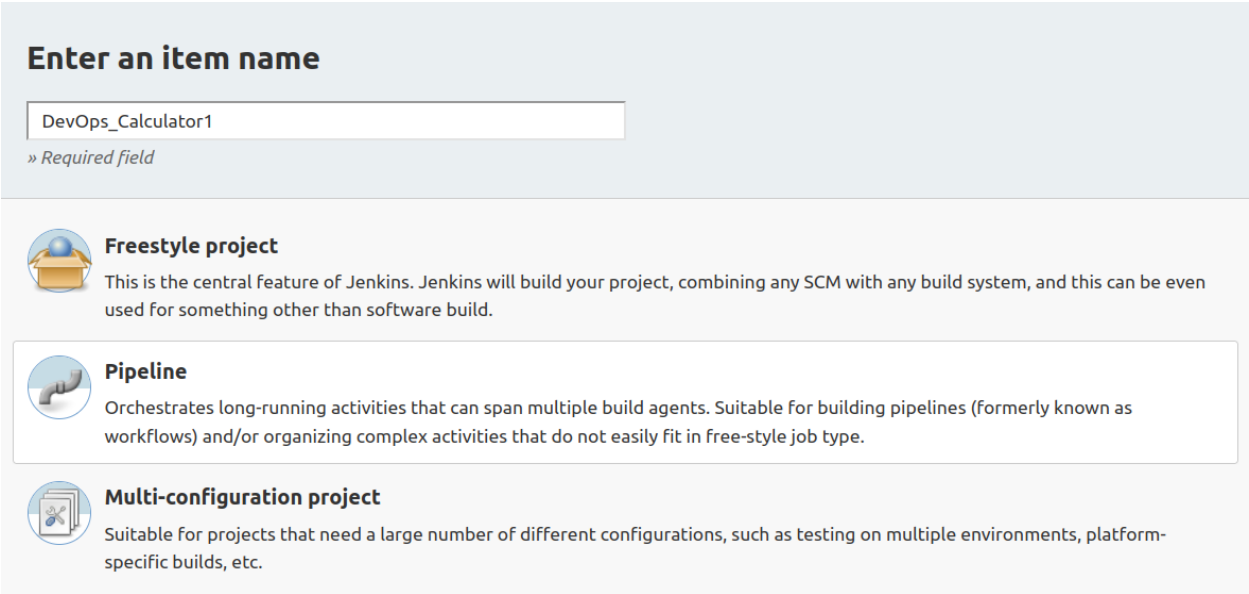
Commands : mkdir .ssh
cd .ssh
ssh-keygen -t rsa
ssh-copy-id akshay@localhost
ssh akshay@localhost

These commands are used for configuring jenkins to use docker via ssh. After executing these commands we get directed out of Jenkins user.

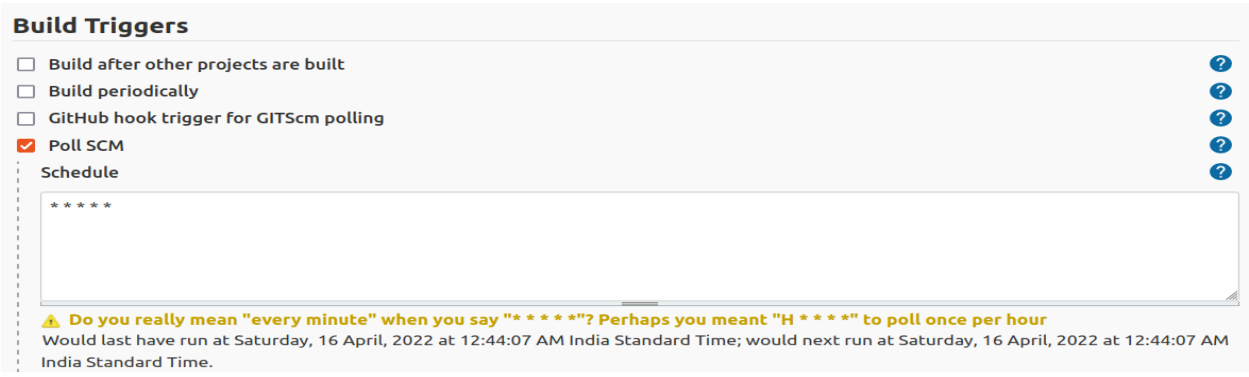
Command : sudo systemctl start jenkins
It is used for starting jenkins at the respective port number.

2. Manage Plugins : All the required plugins need to be installed in jenkins such as Pipeline, Docker, Maven Integration, Ansible, GitHub etc. After installing, Jenkins needs to be restarted and Docker credentials need to be added. In Jenkins dashboard, credentials to the DockerHub repository needs to be added and a unique id is set that is equal to docker with Registry credentials id in the pipeline script.

3. Create a new pipeline in Jenkins.



4. Setup a Jenkins Pipeline : Write a pipeline script for cloning GitHub repository, Maven Build, Docker image build, push docker image to docker hub and Deployment through Ansible. After writing the pipeline script, build the Jenkins job. The job can be built manually or by using poll SCM.



```
pipeline{
    environment{
        docker_image = ""
        registryCredential='docker'
    }
    agent any
    stages{
        stage('Step 1: Git Clone'){
            steps{
                git branch: 'master', url: 'https://github.com/AkshayThite10/CalculatorDevops.git'
            }
        }
        stage('Step 2: Maven Build'){
            steps{
                sh 'mvn clean compile assembly:single'
            }
        }
        stage('Step 3: Build Docker Image'){
            steps {
                script {
                    docker_image = docker.build "akshaythite10/calculator_devops:latest"
                }
            }
        }
        stage('Step 4: Push docker image to hub') {
            steps {
                script {
                    docker.withRegistry('https://registry.hub.docker.com/', registryCredential) {
                        docker_image.push()
                    }
                }
            }
        }
        stage('Step 5: Ansible Deployment'){
            steps{
                ansiblePlaybook becomeUser: 'null',
                colorized: true,
                credentialsId: 'akshaygit',
                installation: 'Ansible',
                inventory: 'deployment/inventory',
                playbook: 'deployment/deploy.yml',
                sudoUser: 'null'
            }
        }
    }
}
```

Continuous Deployment (Ansible) :

Ansible is an open source tool or platform. It is used for configuration management, application deployment, intraservice orchestration and provisioning

Steps :

1. Creating Ansible playbook : In the playbook we need to provide the correct python version and path of the docker image. This playbook is created in another directory inside our working directory.

deploy.yml

deployment > deploy.yml

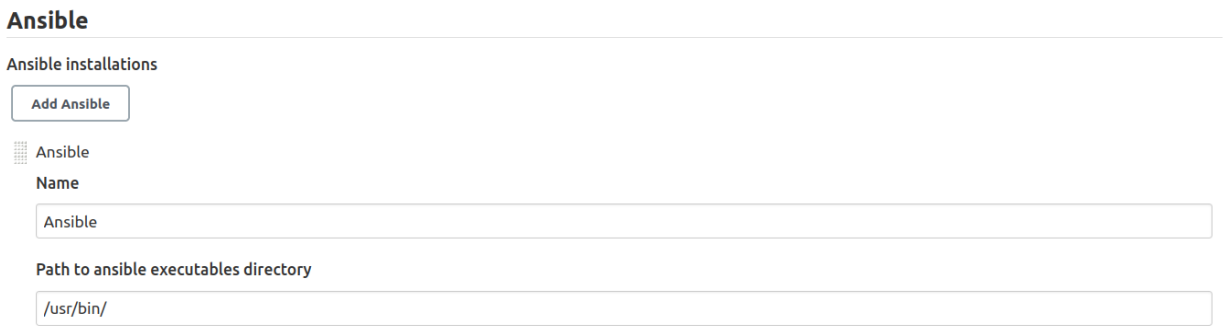
```yaml
---
- name: Pull Docker image of Calculator
  hosts: all
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Pull image
      docker_image:
        name: akshaythite10/calculator_devops:latest
        source: pull
```

inventory

deployment > inventory

```
localhost ansible_user=akshay
```

2. Configuring Ansible in Jenkins : Go to Jenkins -> Dashboard -> Manage Jenkins -> Global Tool Configuration. Add Ansible here by giving the correct path.



3. Add Ansible stage in the Jenkins pipeline script.

```
stage('Step 5: Ansible Deployment'){
    steps{
        ansiblePlaybook becomeUser: 'null',
        colorized: true,
        credentialsId: 'akshaygit',
        installation: 'Ansible',
        inventory: 'deployment/inventory',
        playbook: 'deployment/deploy.yml',
        sudoUser: 'null'
    }
```

Log Management (log4j) :

It is used for keeping track of operations performed in the application.

Steps :

1. Create a log4j file : Script is written in a file named log4j2.xml

log4j2.xml

src > main > resources > log4j2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
    <Appenders>
        <Console name="ConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{dd/MMM/yyy:HH:mm:ss SSS} [%F] [%level] %logger{36} %msg%n"/>
        </Console>
        <File name="FileAppender" fileName="calculator_devops.log" immediateFlush="false" append="true">
            <PatternLayout pattern="%d{dd/MMM/yyy:HH:mm:ss SSS} [%F] [%level] %logger{36} - %msg%n"/>
        </File>
    </Appenders>
    <Loggers>
        <Root level="debug">
<!--            <AppenderRef ref="ConsoleAppender"/>-->
        <AppenderRef ref="FileAppender"/>
        </Root>
    </Loggers>
</Configuration>

```

2. Adding logger functions : Add these to the calculator program.

private static final Logger logger = LogManager.getLogger(App.class);
logger.info(“Message”); // For generating the logs.

Building Jenkins Pipeline :

Stage View

	Step 1: Git Clone	Step 2: Maven Build	Step 3: Build Docker Image	Step 4: Push docker image to hub	Step 5: Ansible Deployment
Average stage times: (Average full run time: ~1min 8s)	1s	7s	9s	18s	5s
#34 Apr 16 00:36 No Changes	1s	4s	10s	31s	8s
#33 Apr 16 00:31 No Changes	2s	13s	7s failed	427ms failed	443ms failed
#32 Apr 14 17:47 1 commit	1s	10s	14s	31s	5s
#31 Apr 14 17:32 1 commit	1s	4s	11s	31s	4s
#30 Apr 14 17:30 1 commit	1s	5s failed	394ms failed	416ms failed	427ms failed

Now, the docker image can be pulled and runned inside the local machine to generate the logs which act as the input for the ELK Stack.

```
akshay@akshay-Vostro-3578:~$ docker images
REPOSITORY                                                      TAG         IMAGE ID       CREATED          SIZE
akshaythite10/calculator_devops                                 latest      31428e23e78c   52 minutes ago   528MB
registry.hub.docker.com/akshaythite10/calculator_devops         latest      31428e23e78c   52 minutes ago   528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      b9d6e1f23b2a   32 hours ago     528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      1ded0602f868   32 hours ago     528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      e0f10756c4a1   32 hours ago     528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      ac9aaac7b88b   33 hours ago     528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      c2a138db85ec   37 hours ago     528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      fceedeac6c96   5 days ago       528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      4cbf06df29b6   5 days ago       528MB
<none>                                                          <none>      ab294932b825   5 days ago       528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      8632a0d8c5b7   5 days ago       528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      18c63ff486e2   5 days ago       528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      f29c9fc4efc2   5 days ago       528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      178eded678ba   5 days ago       528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      52a6a0ee417e   5 days ago       528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      c6e8233522b5   5 days ago       528MB
registry.hub.docker.com/akshaythite10/calculator_devops         <none>      20d11e81ea2c   5 days ago       528MB
<none>                                                          <none>      a5539cacf28a   5 days ago       528MB
<none>                                                          <none>      b7c1beddbd08   5 days ago       528MB
<none>                                                          <none>      11479b9c8742   5 days ago       528MB
<none>                                                          <none>      633d07acf8ff   5 days ago       528MB
<none>                                                          <none>      42d0a65293c9   5 days ago       526MB
nodejs_app                                                      latest      a87f31171f38   10 days ago      89.7MB
akshay/docker                                                   latest      e3411ee836c7   10 days ago      124MB
<none>                                                          <none>      57fdaf8c395f   10 days ago      124MB
openjdk                                                         8           18fbe41f975e   2 weeks ago      526MB
ubuntu                                                          latest      2b4cba85892a   6 weeks ago      72.8MB
ubuntu                                                          <none>      54c9d81cbb44   2 months ago     72.8MB
node                                                            10-alpine   aa67ba258e18   12 months ago    82.7MB
```

Command : docker images
It displays all the docker images.

Running the docker image inside a docker container.

```
akshay@akshay-Vostro-3578:~$ docker run -it akshaythite10/calculator_devops
...........Calculator Program...........
Operation choices :

1. Square root
2. Factorial
3. Natural log
4. Power
5. Exit

Enter your choice :
1
Square root
Enter a number :
64

Square root = 8.0

Operation choices :

1. Square root
2. Factorial
3. Natural log
4. Power
5. Exit

Enter your choice :
2
Factorial
Enter a number
6

Factorial = 720
```

```
Operation choices :

1. Square root
2. Factorial
3. Natural log
4. Power
5. Exit

Enter your choice :
3
Natural log
Enter a number :
1024

Natural log = 6.931471805599453

Operation choices :

1. Square root
2. Factorial
3. Natural log
4. Power
5. Exit

Enter your choice :
4
Power
Enter a number :
2
Enter exponent :
10

Power = 1024.0

Operation choices :

1. Square root
2. Factorial
3. Natural log
4. Power
5. Exit

Enter your choice :
5
akshay@akshay-Vostro-3578:~$
```

```
akshay@akshay-Vostro-3578:~$ sudo docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED         STATUS                     PORTS     NAMES
27bfb478d41a   akshaythite10/calculator_devops    "java -cp Calculator…"   6 minutes ago   Exited (0) 5 minutes ago             pensive_goodall
8af08e961571   1ded0602f868                       "java -cp Calculator…"   32 hours ago    Exited (255) 9 hours ago             flamboyant_maxwell
31befcb35e8b   e0f10756c4a1                       "java -cp Calculator…"   32 hours ago    Exited (255) 9 hours ago             dreamy_haibt
fca9b11aec5b   c2a138db85ec                       "java -cp Calculator…"   37 hours ago    Exited (255) 9 hours ago             romantic_khorana
577a4d401370   fceedeac6c96                       "java -cp Calculator…"   5 days ago      Exited (0) 5 days ago                elated_ardinghelli
fcc2ceca074f   fceedeac6c96                       "java -cp Calculator…"   5 days ago      Exited (0) 5 days ago                competent_yalow
9cf1ac7e87bd   4cbf06df29b6                       "java -cp Calculator…"   5 days ago      Exited (1) 5 days ago                practical_dewdney
2f82dbdfb1e5   8632a0d8c5b7                       "java -cp Calculator…"   5 days ago      Exited (1) 5 days ago                flamboyant_keldysh
9ed8362fc993   8632a0d8c5b7                       "java -cp Calculator…"   5 days ago      Exited (1) 5 days ago                optimistic_lovelace
6df7e3dbfc23   nodejs_app:latest                  "docker-entrypoint.s…"   10 days ago     Exited (0) 5 days ago                nodejs_app
a6830ace77f9   akshay/docker                      "./run.sh arg1"          10 days ago     Exited (0) 10 days ago               container1
a5e466d6c35b   ubuntu                             "/bin/bash"              6 weeks ago     Exited (0) 6 weeks ago               Jenkins_Agent
d65e87d4d838   54c9d81cbb44                       "bash"                   6 weeks ago     Exited (0) 6 weeks ago               user1
```

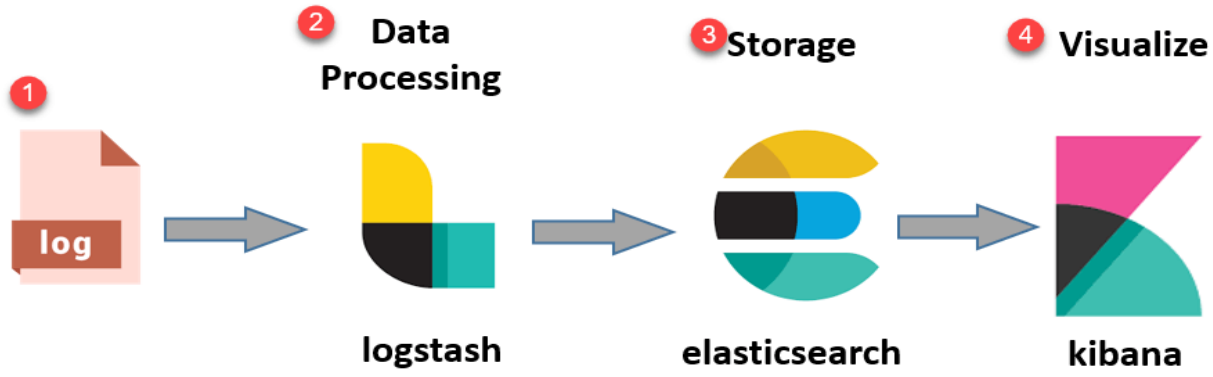
Command : sudo docker ps -a
It displays all the details of the docker containers.

```
akshay@akshay-Vostro-3578:~$ docker start 27
27
akshay@akshay-Vostro-3578:~$ docker exec -it 27 "/bin/bash"
root@27bfb478d41a:/# ls
CalculatorDevops-1.0-SNAPSHOT-jar-with-dependencies.jar  boot                  dev  home  lib64  mnt  proc  run   srv  tmp  var
bin                                                      calculator_devops.log  etc  lib   media  opt  root  sbin  sys  usr
root@27bfb478d41a:/# cat calculator_devops.log
15/Apr/2022:20:04:38 239 [App.java] [INFO] App - Square root computed
15/Apr/2022:20:04:45 449 [App.java] [INFO] App - Factorial computed
15/Apr/2022:20:04:55 785 [App.java] [INFO] App - Natural log computed
15/Apr/2022:20:05:05 709 [App.java] [INFO] App - Power computed
```

Commands : docker start <container_id>
docker exec -it <container_id> “/bin/bash”
These commands are used to start the docker container and then check for the log file.

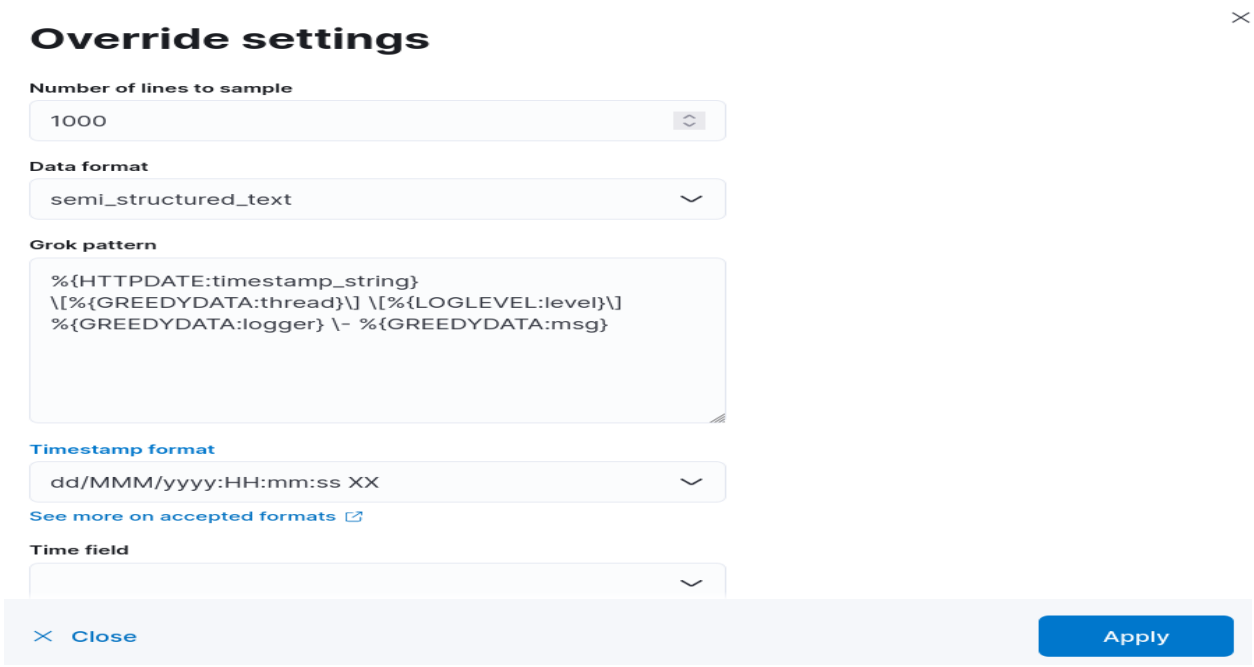
Continuous Monitoring (ELK) :

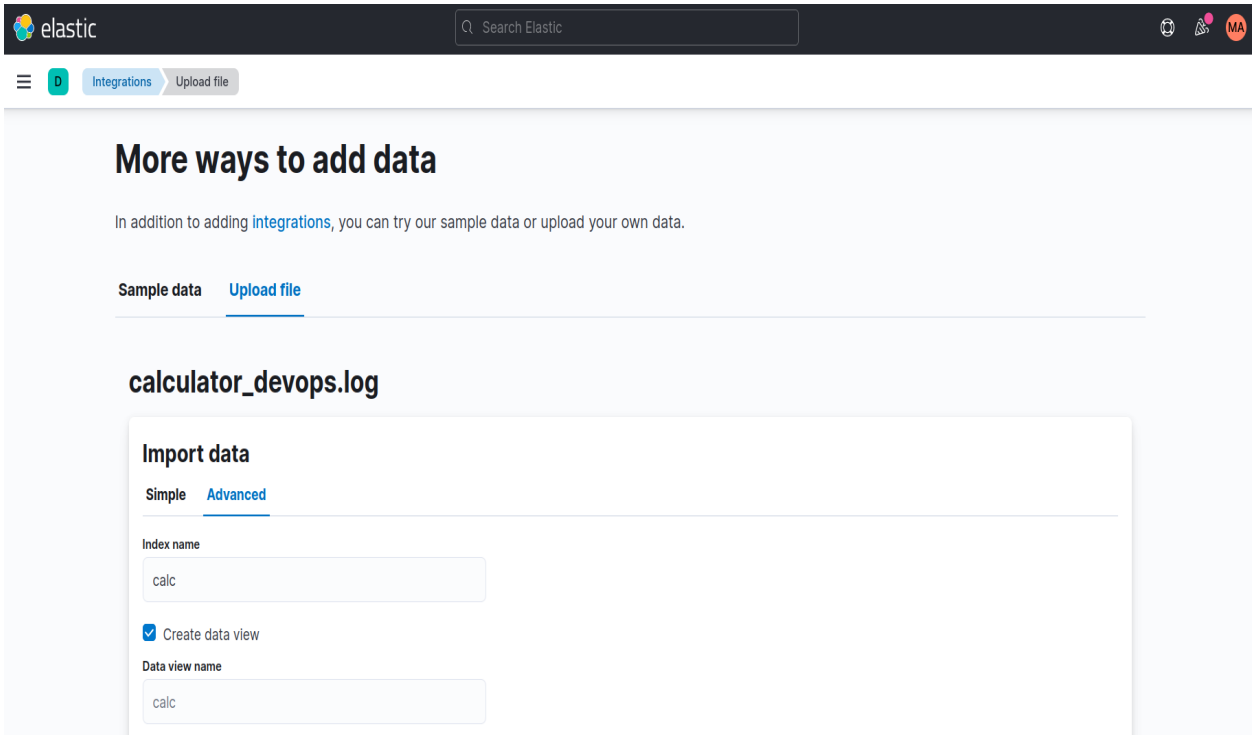
ELK stands for Elasticsearch, Logstash and Kibana. It aggregates logs from systems and applications, analyzes these logs and creates visualization for application and infrastructure monitoring, faster troubleshooting and security analytics.

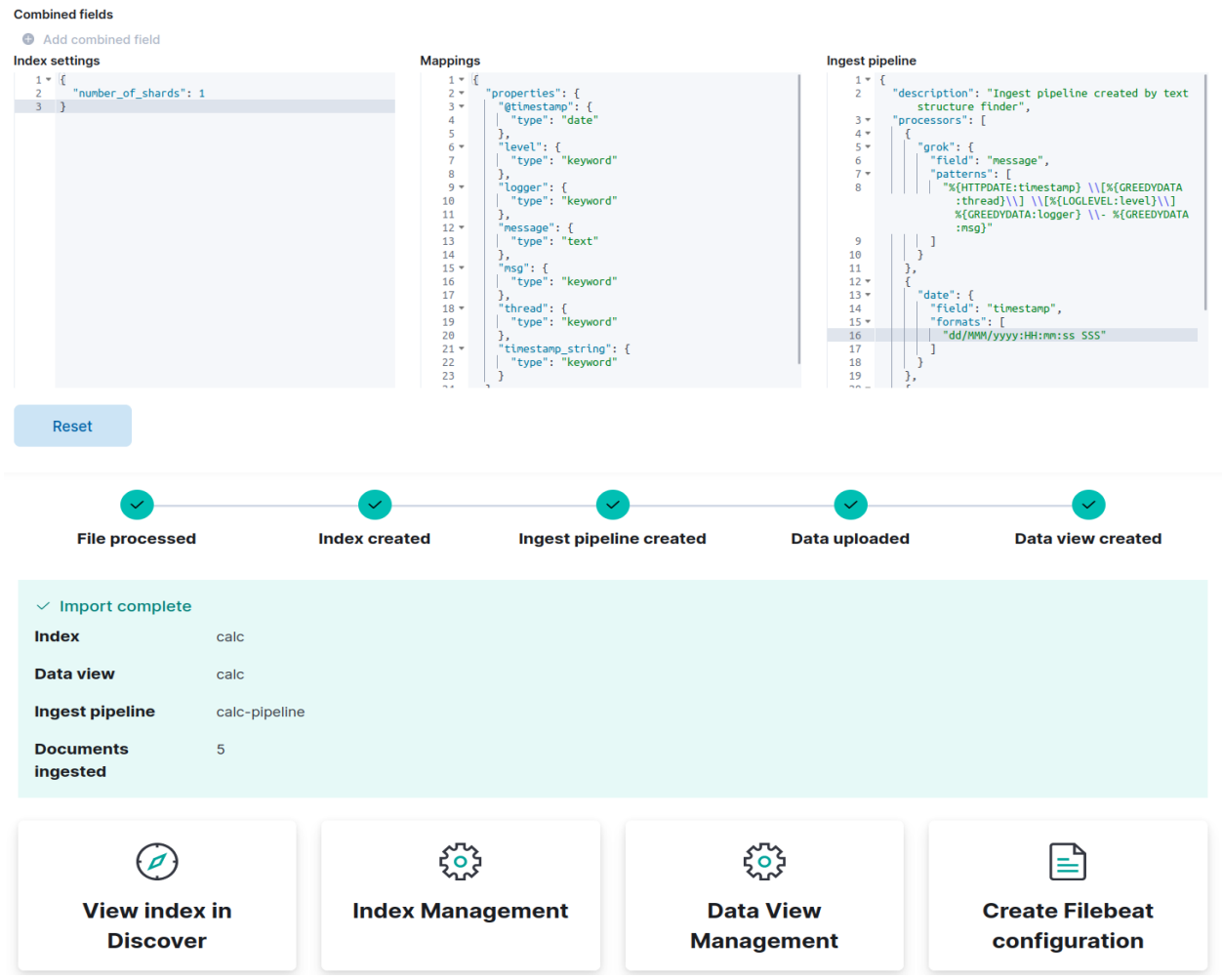


Steps :

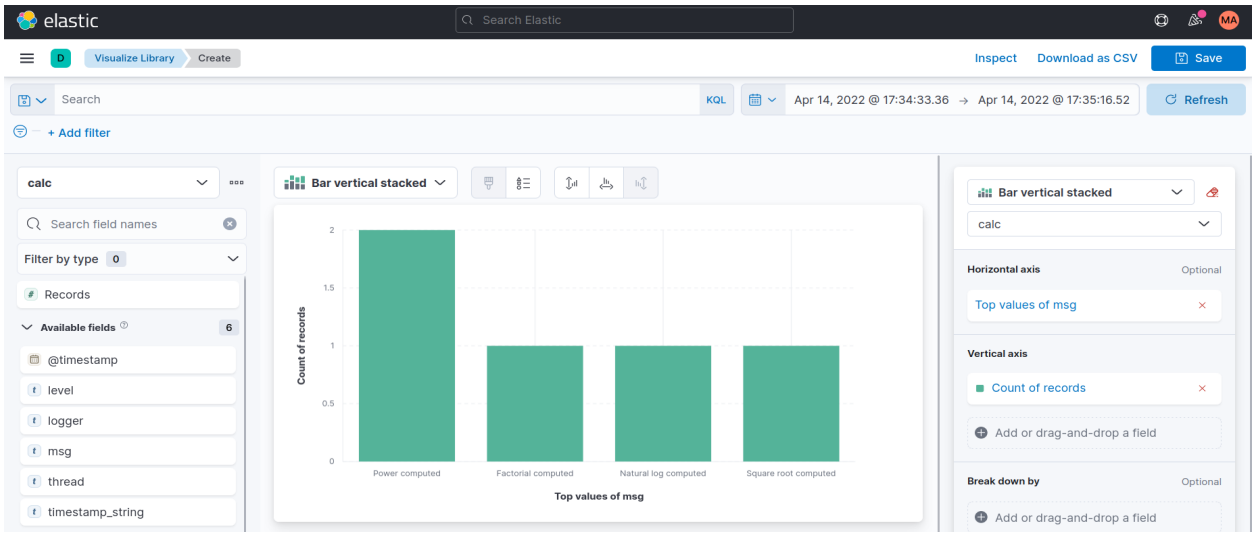
1. Upload the log file using the following override settings







2. Data Visualization



Errors Faced while Execution :

1. In the third stage of pipeline - Building Docker Image


```
+ docker build -t akshaythite10/calculator_devops:latest .
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post
"http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&
cpusetmems=&cpushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=default&rm=1&shmsize=0&
t=akshaythite10%2Fcalculator_devops%3Alatest&target=&ulimits=null&version=1": dial unix /var/run/docker.sock: connect: permission
denied
```

Solution : Modified permissions of /var/run/docker.sock
Command : sudo chmod 777 /var/run/docker.sock

2. In last stage of pipeline - Ansible Deployment

```
[DevOps_Calculator] $ sshpass ******** /usr/bin/ansible-playbook deployment/deploy.yml -i deployment/inventory -u AkshayThite10 -k

PLAY [Pull Docker image of Calculator] *************************************

TASK [Gathering Facts] *****************************************************
[0;32mok: [localhost][0m
[0;32m[0m
TASK [Pull image] **********************************************************
[0;31mfatal: [localhost]: FAILED! => {"changed": false, "msg": "Failed to import the required Python library (Docker SDK for Python:
docker (Python >= 2.7) or docker-py (Python 2.6)) on akshay-Vostro-3578's Python /usr/bin/python3. Please read module documentation
and install in the appropriate location. If the required library is installed, but Ansible is using the wrong Python interpreter,
please consult the documentation on ansible_python_interpreter, for example via `pip install docker` or `pip install docker-py`
(Python 2.6). The error was: No module named 'docker'"}[0m
[0;31m[0m
PLAY RECAP *****************************************************************
[0;31mlocalhost[0m                  : [0;32mok=1    [0m changed=0    unreachable=0    [0;31mfailed=1   [0m skipped=0    rescued=0
ignored=0


FATAL: command execution failed
hudson.AbortException: Ansible playbook execution failed
        at org.jenkinsci.plugins.ansible.AnsiblePlaybookBuilder.perform(AnsiblePlaybookBuilder.java:262)
        at org.jenkinsci.plugins.ansible.workflow.AnsiblePlaybookStep$AnsiblePlaybookExecution.run(AnsiblePlaybookStep.java:430)
        at org.jenkinsci.plugins.ansible.workflow.AnsiblePlaybookStep$AnsiblePlaybookExecution.run(AnsiblePlaybookStep.java:351)
        at
org.jenkinsci.plugins.workflow.steps.AbstractSynchronousNonBlockingStepExecution$1$1.call(AbstractSynchronousNonBlockingStepExecution
.java:47)
        at hudson.security.ACL.impersonate2(ACL.java:449)
        at hudson.security.ACL.impersonate(ACL.java:461)
        at
org.jenkinsci.plugins.workflow.steps.AbstractSynchronousNonBlockingStepExecution$1.run(AbstractSynchronousNonBlockingStepExecution.ja
va:44)
        at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)
        at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
        at java.base/java.lang.Thread.run(Thread.java:829)
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: Ansible playbook execution failed
Finished: FAILURE
```

Solution : Installed a python library.
Command : pip install docker-py