MINI PROJECT - SCIENTIFIC CALCULATOR WITH DEVOPS

Naga Sri Vaishnavi Dhulipala : IMT2017514

March 17, 2021

1 INTRODUCTION

In this mini project, we develop a scientific calculator with 4 operations - Square root, Factorial, Natural Log and Power suing DevOps practices.
DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.

1. **Programming Language:** Java
2. **Testing:** JUnit
3. **Build tool:** Apache Maven
4. **Source Code Management:** GitHub for Git.
   `https://github.com/VaishnaviDhulipalla2902/CalculatorDevOps`
5. **Containerization:** Docker
6. **Continuous Integration:** Jenkins
7. **Continuous Deployment:** Ansible
8. **Generate Logs:** Log4j
9. **Monitoring:** ELK Stack

2 BUILD TOOL – MAVEN

Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process like ANT. In short terms we can tell maven is a tool that can be used for building and managing any Java-based project.
Here, maven is used to build the java source code by configuring all the dependencies thereby creating a .jar executable binary of it in the target folder.
In a Maven project, there is a file called "pom.xml". This configuration file for the Maven project. This file manages the metadata, dependencies and plugins for the project.

1. **Create a Maven project hierarchy using the following command in the project directory.**
   Set the path to the working directory and run the following command.

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

2. **Copy Java Code into the App.java file of the maven hierarchy.**
   Write the Calculator program in App.java file and JUnit Test Program in AppTest.java file.

3. **Clean, Compile, Install and Site**
   Clean the project hierarchy using

```
~$ mvn clean
```

   The main goal here is to clear the cache in the Maven hierarchy i.e. if there are any previous build in the 'target' folder, the folder would be deleted for a fresh build.
   Now compile all the java source code using

```
~$ mvn compile
```

   The main goal here is to compile all the source code files in the 'src/main/java/<pacakgename>' folder.
   Now, we create the build of the project using

```
~$ mvn install
```

   The main goal here is to create a binary executable .jar file of the project which is stored in the 'target' folder.

   An additional step we do here is

```
~$ mvn site
```

This command automatically creates a documentation type report of the project using HTML and CSS in the 'target/site' folder. Attaching a summary report screenshot of the same(more such reports in the 'target/site' folder of the GitHub link given) :

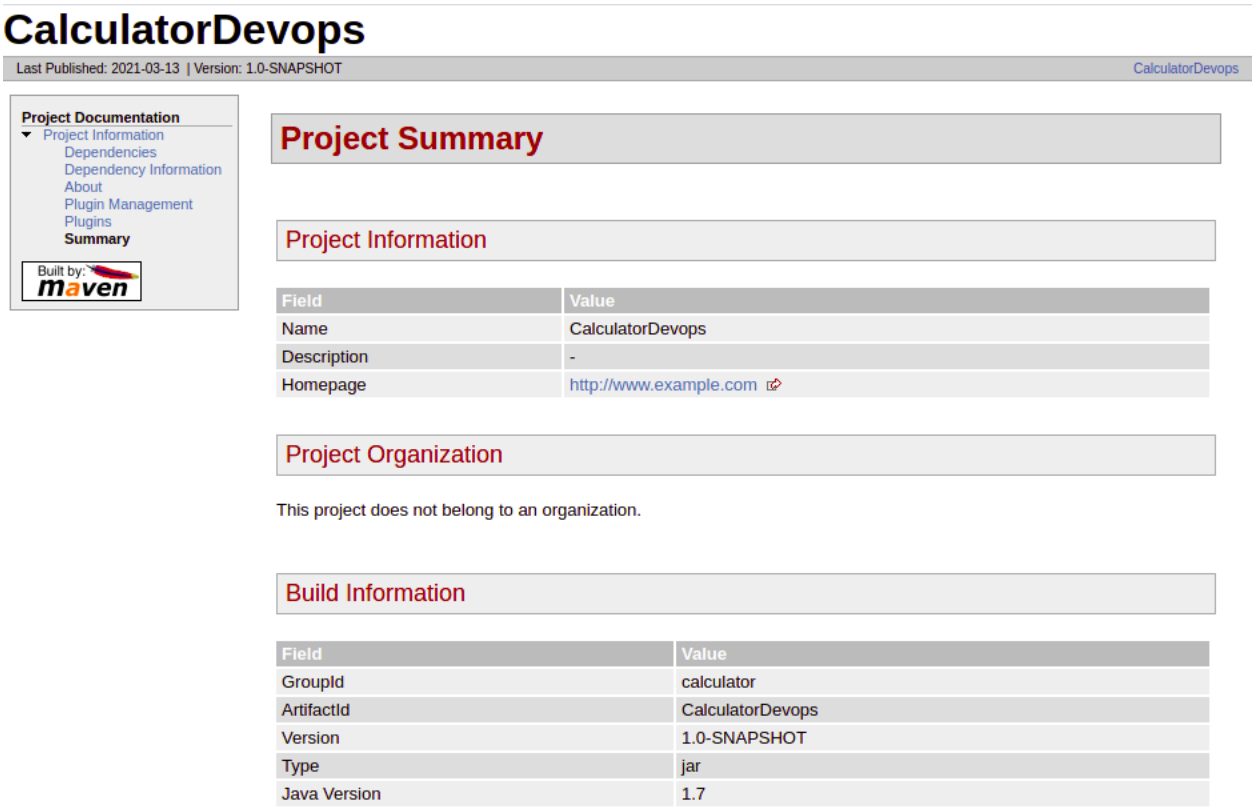

Figure 1: Maven Project Summary by 'mvn site'

3 SOURCE CODE MANAGEMENT – GITHUB

Source code management (SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors. SCM is also synonymous with Version control.
In this step, we create a repository on GitHub and then configure the maven project on out local machine and link it to the Git repository.

```
~$ git init
```

Open the command line on the root of the Maven project and type the above command to initialize the Maven project as a Git repository.

```
~$ git remote add origin
```

This command helps to link the Git repository with local repository.

```
~$ git add .
```

This command is to Stage the Maven project in Git environment. (".” command specifies to stage all the files in the current folder that are modified or created.)

```
~$ git commit -m "Commit Message"
```

This command is to commit the staged files to GitHub.

```
~$ git push -u origin main
```

This command is to push the changes to the GitHub repository.

4 CONTAINERIZATION – DOCKER

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.

1. **Install docker on the local machine**

```
~$ curl -fsSL https://get.docker.com -o get-docker.sh
~$ sh get-docker.sh
```

2. **Create a repository in DockerHub**
   `https://hub.docker.com/repository/docker/vaishnavi2902/calculator_devops`

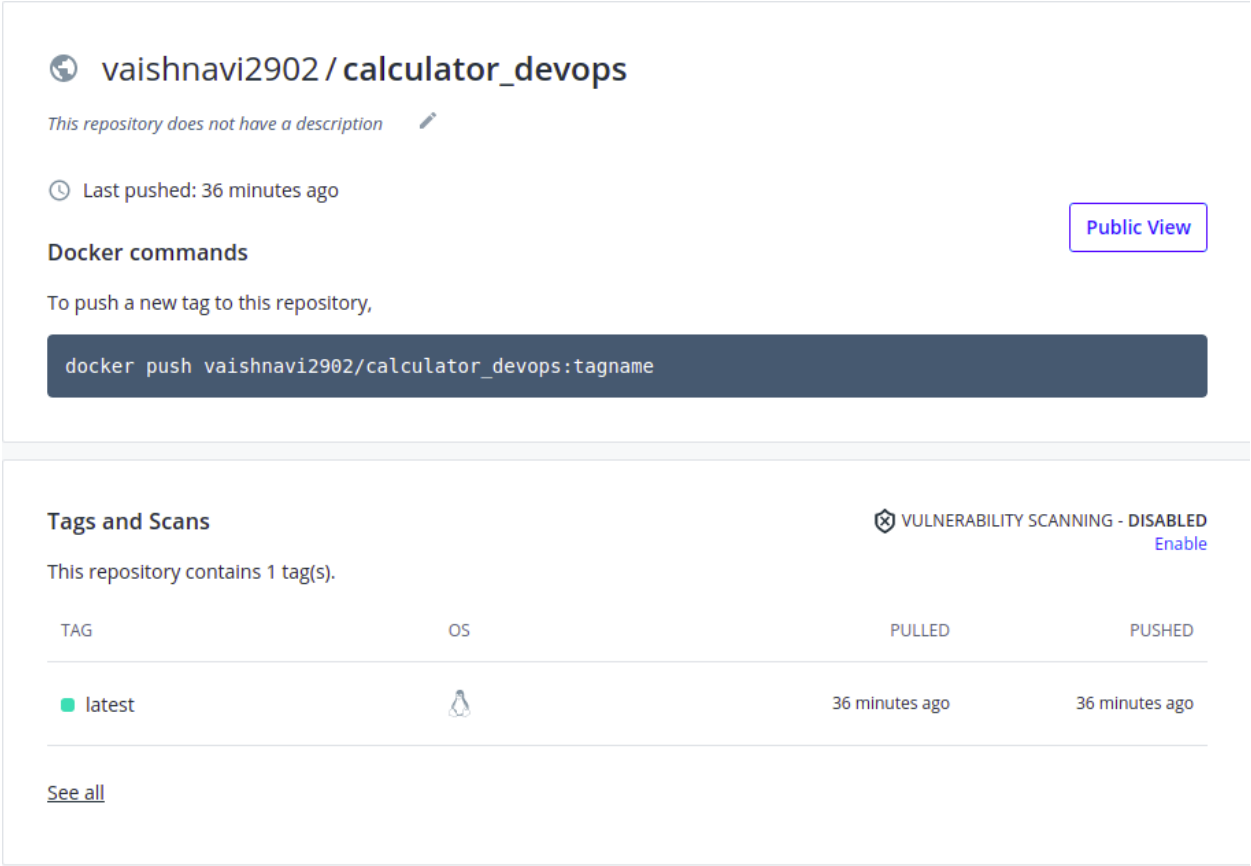

Figure 2: Docker Repository

3. **Create a Dockerfile**
   After creating a Dockerfile, push the changes to GitHub.

```
Dockerfile > ...
FROM openjdk:8
COPY ./target/calculatorDevops-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java","-cp","calculatorDevops-1.0-SNAPSHOT-jar-with-dependencies.jar","App"]
```

Figure 3: Docker File

5 CONTINUOUS INTEGRATION – JENKINS

1. **Add Jenkins to the Docker group**

```
sudo apt install openssh-server
sudo su - jenkins
```

By executing the above commands, we get logged into jenkins. Here we configure Jenkins to use Docker image via ssh.

```
mkdir .ssh
cd .ssh
ssh-keygen -t rsa
ssh-copy-id vaishnavi@localhost
ssh vaishnavi@localhost
```

After executing the last command, we get automatically directed outside the Jenkins user.
We can now start Jenkins with the command

```
sudo systemctl start jenkins
```

Jenkins starts at port number 8080 so we login on to `http://localhost:8080` on to the browser.

2. **Manage Plugins**
   We need to install all the required plugins like Build pipeline, Docker, GitHub, Maven Integration, Ansible etc. After they are all done installing, we need to restart Jenkins and add Docker credentials. In the Jenkins dashboard we add credentials to the Docker Hub repository and we set an unique id which is equal to docker with Registry credentials id in pipeline script.

3. **Create a New Pipeline on Jenkins**
   Create new item, and select pipeline.

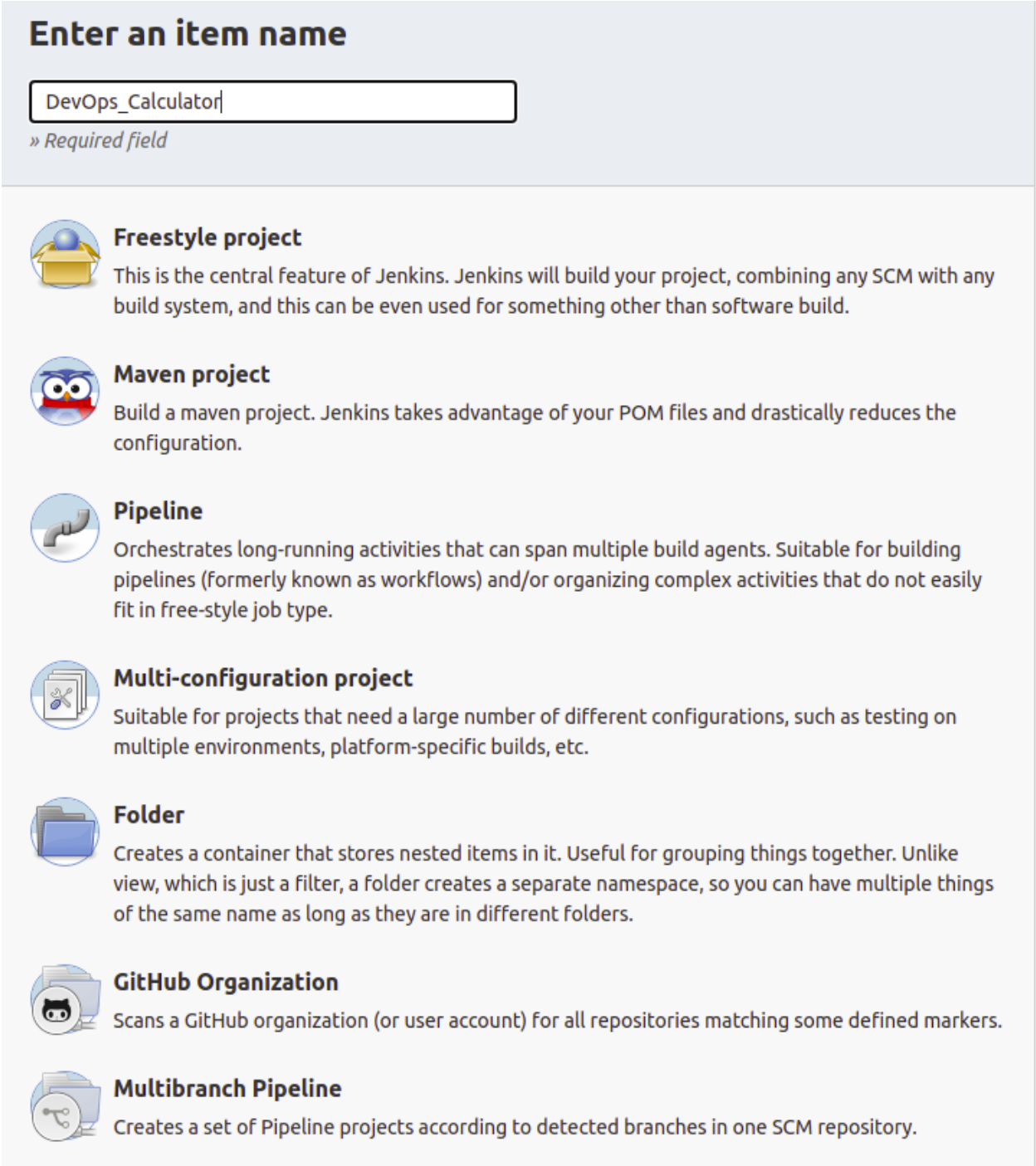

Figure 4: Creating Jenkins Pipeline

4. **Setup a Jenkins Pipeline** Write a Pipeline Script which includes steps like Git Clone, Maven Build, Building a Docker image, Pushing the docker image to docker hub and Ansible Deployment. After we are done setting up every aspect of our DevOps tool chain, we may now build the jenkins job. This job for now can be manually triggered but for other SCM like GitLab it can be triggered based on events

such as new push or new pull but for GitHub we might have set up a webhook. This can be done via port forwarding; we have to make jenkins port 8080 open on wide internet so it can be triggered based on event. But for now we perform this job manually.

Jenkins Pipeline Script

```
pipeline{
    environment{
        docker_image = ""
    }
    agent any
    stages{
        stage('Step 1: Git Clone'){
            steps{
                git branch: 'main', url: 'https://github.com/VaishnaviDhulipalla2902/CalculatorDevOps.git'
            }
        }
        stage('Step 2: Maven Build'){
            steps{
                sh 'mvn clean package'
            }
        }
        stage('Step 3: Build Docker Image'){
            steps {
                script {
                    docker_image = docker.build "vaishnavi2902/calculator_devops:latest"
                }
            }
        }
        stage('Step 4: Push docker image to hub') {
            steps {
                script {
                    docker.withRegistry('', 'docker') {
                        docker_image.push()
                    }
                }
            }
        }
        stage('Step 5: Ansible Deployment'){
            steps{
                ansiblePlaybook becomeUser: null,
                colorized: true,
                credentialsId: 'docker',
                installation: 'Ansible'
                disableHostKeyChecking: true,
                inventory: 'deployment/inventory',
                playbook: 'deployment/deploy.yml',
                sudoUser: null
            }
        }
    }

}
```

Figure 5: Jenkins Pipeline script

6 CONTINUOUS DEPLOYMENT – ANSIBLE

Ansible is an open-source automation tool, or platform, used for IT tasks such as configuration management, application deployment, intraservice orchestration, and provisioning. Ansible is mainly used to perform a lot of tasks that otherwise are time-consuming, complex, repetitive, and can make a lot of errors or issues.

1. **Create an Ansible playbook** We make a new directory in our working directory to create the Ansible playbook. We need to be careful about giving the correct path of docker image and the correct python version for which we can do

```
~$ which python
```

deploy.yml

deployment > deploy.yml

```
---
- name: Pull Docker image of Calculator
  hosts: all
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Pull image
      docker_image:
        name: vaishnavi2902/calculator_devops:latest
        source: pull
```

Figure 6: Ansible Playbook

inventory

deployment > inventory

```
localhost ansible_user=vaishnavi
```

Figure 7: Inventory File

2. **Configure Ansible in Jenkins** We need to go to Jenkins -> Dashboard -> Manage Jenkins -> Global Tool Configuration and add Ansible there. Here, we also need to give the correct path to Ansible.

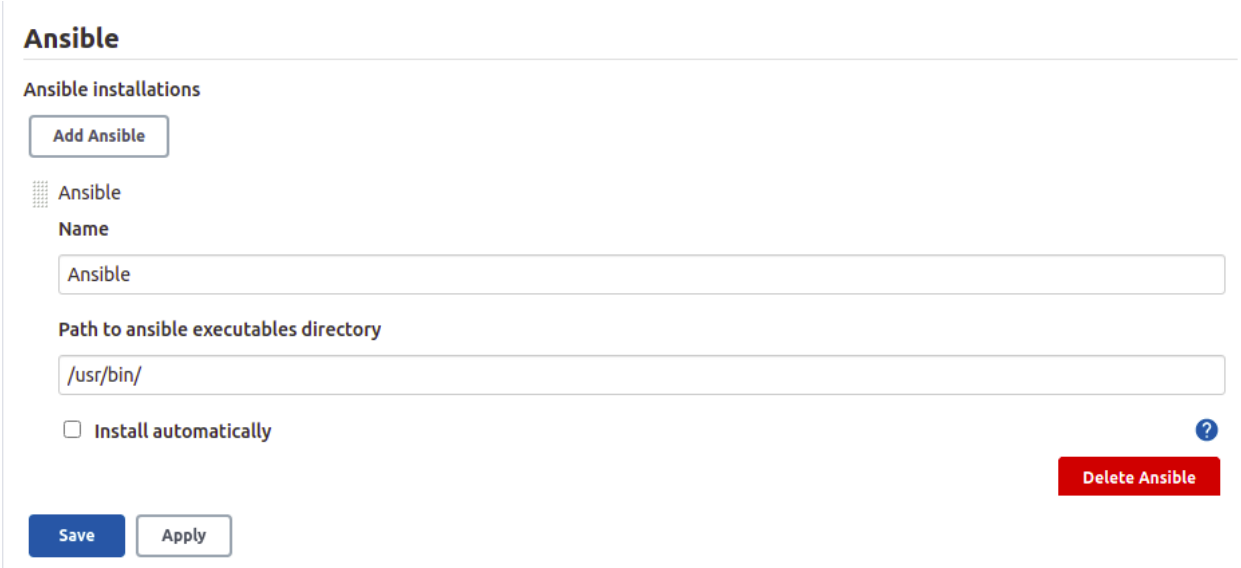

Figure 8: Ansible Configuration in Jenkins

3. **Add Ansible Stage in the Jenkins pipeline script**

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

7 LOG MANAGEMENT – LOG4J

Logging keeps track of all the operations that were performed in the application, warnings, debugging information, errors etc. Log4j has a plugin in Maven which is convenient for operations which is seen in the pom.xml file.

1. **Create Log4j file** We write a script for Logging named log4j2.xml.

log4j2.xml

src > main > resources > log4j2.xml

```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>
```

Figure 9: Log4j2.xml file

2. **Add Logger functions**
   We need to add logger functions in the main application code. Add the below line inside the main class.

```
    private static final Logger logger = LogManager.getLogger(App.class);

```

   And add some lines inside the functions to generate logs.

```
    logger.info("Your Message");

```

8 BUILD JENKINS PIPELINE

Now, we need to build the project on Jenkins. While we build it, we might encounter a lot of errors which we fix along the way. The full stage pipeline looks like,

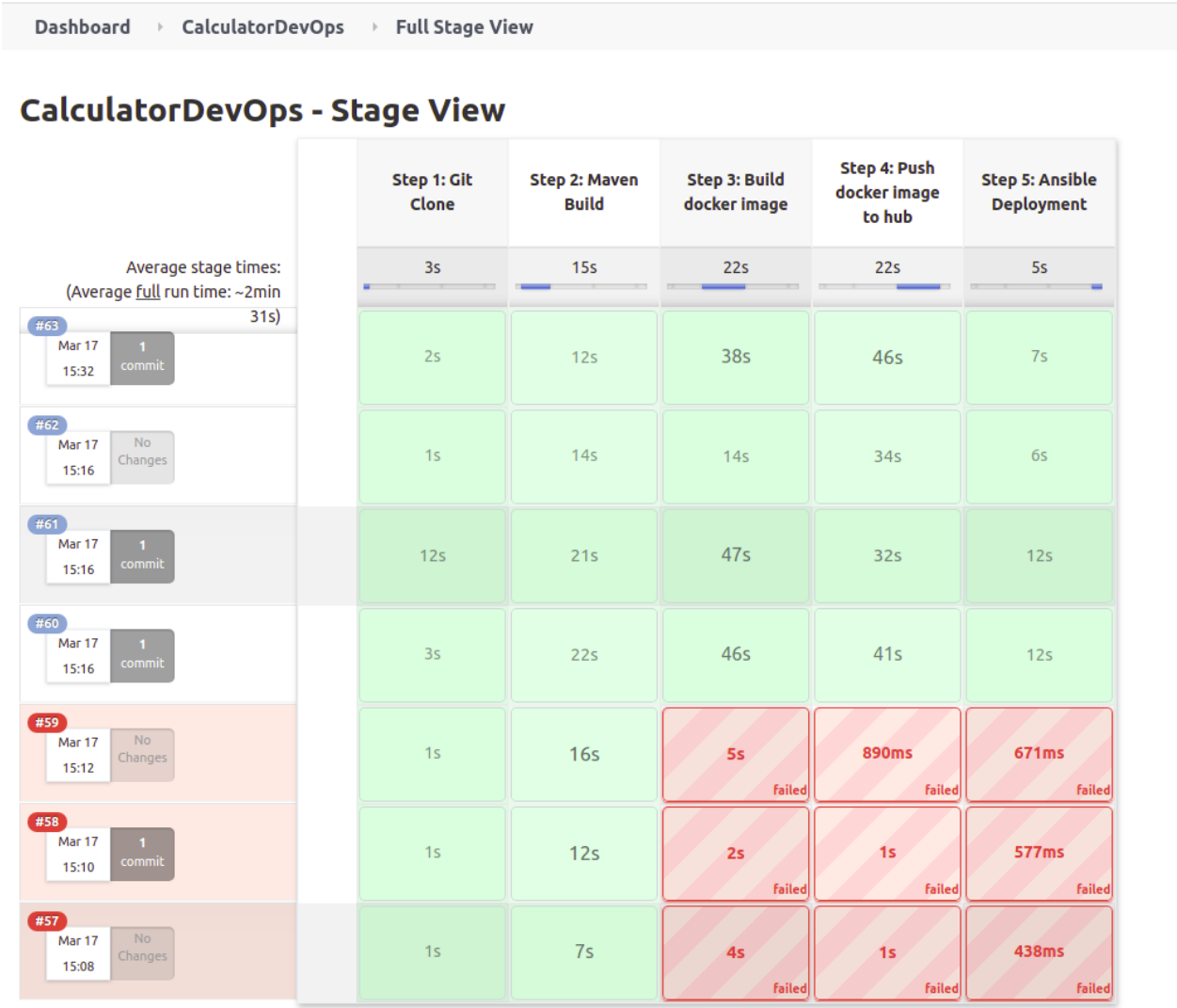

Figure 10: Full Stage Pipeline Jenkins

There are 5 stages in my Jenkins pipeline:

1. Git Clone
2. Maven Build
3. Build Docker Image
4. Push Docker Image to Hub
5. Ansible Deployment

After doing all this, the docker image is pulled on to the local machine. We can run this image on the local machine to generate logs which act as input to the ELK monitoring.

```
~$ docker images
~$ docker run -it <docker_image>
```

```
vaishnavi@vaishnavi-Inspiron-5567:~$ docker images
REPOSITORY                          TAG       IMAGE ID       CREATED          SIZE
vaishnavi2902/calculator_devops     latest    dbc81185e661   57 seconds ago   516MB
vaishnavi2902/calculator_devops     <none>    b01d2f0cd7dc   11 minutes ago   516MB
vaishnavi2902/calculator_devops     <none>    9e0682c4e274   17 minutes ago   516MB
vaishnavi2902/calculator_devops     <none>    316469bea218   22 minutes ago   516MB
vaishnavi2902/calculator_devops     <none>    33faaea61730   36 minutes ago   516MB
vaishnavi2902/calculator_devops     <none>    c6adc45b964d   38 minutes ago   516MB
<none>                              <none>    abd5788a5fc9   38 minutes ago   516MB
vaishnavi2902/calculator_devops     <none>    dc29a09f0117   2 hours ago      514MB
vaishnavi2902/calculator_devops     <none>    e041bab92b4c   4 hours ago      514MB
vaishnavi2902/calculator_devops     <none>    247b61b4b6d8   22 hours ago     514MB
vaishnavi2902/calculator_devops     <none>    cdc59a63abc1   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    992dce989151   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    db9e51d3e889   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    57ceb67beaf1   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    37a34165e43f   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    15fa1b5f16ca   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    98c8ba597418   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    8918137eeee0   3 days ago       514MB
<none>                              <none>    46823ec89a11   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    68e58606c864   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    88be39e2e2e4   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    e1e08e65447e   3 days ago       514MB
<none>                              <none>    7d0002685354   3 days ago       514MB
vaishnavi2902/calculator_devops     <none>    ce59bd4019ad   4 days ago       514MB
<none>                              <none>    ec1ba12d853a   4 days ago       514MB
vaishnavi2902/calculator_devops     <none>    d2192ac0d31c   4 days ago       514MB
vaishnavi2902/calculator_devops     <none>    1b28a03cc025   4 days ago       514MB
vaishnavi2902/calculator_devops     <none>    8af94bac6484   4 days ago       514MB
vaishnavi2902/calculator_devops     <none>    8c9cb4d36c0f   4 days ago       514MB
vaishnavi2902/calculator_devops     <none>    4e58a2614445   4 days ago       514MB
vaishnavi2902/calculator_devops     <none>    a3b989f2ac26   4 days ago       514MB
vaishnavi2902/calculator_devops     <none>    fa2e553fd591   4 days ago       514MB
vaishnavi2902/calculator_devops     <none>    8ff3d4ee6218   4 days ago       514MB
<none>                              <none>    4ac6012c7e64   4 days ago       514MB
<none>                              <none>    f234fbcb349e   4 days ago       514MB
openjdk                             8         f28d33bb11eb   7 days ago       514MB
hello-world                         latest    d1165f221234   11 days ago      13.3kB
vaishnavi@vaishnavi-Inspiron-5567:~$ docker run -it vaishnavi2902/calculator_devops
------------Calculator--------------
Choices of Operations:

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

Figure 11: Docker Images

Now we know that the application executes smooth and we run the application for a few times to generate the log files. For that we do,

```
~$ docker docker ps -a
```

```
vaishnavi@vaishnavi-Inspiron-5567:~$ docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED             STATUS                      PORTS     NAMES
8a5d55cc4ac9   vaishnavi2902/calculator_devops  "java -cp calculator…"   About a minute ago  Exited (0) 11 seconds ago             goofy_murdock
cc6398968b23   b01d2f0cd7dc                   "java -cp calculator…"   10 minutes ago      Exited (130) 4 minutes ago            interesting_galois
bf324390e346   b01d2f0cd7dc                   "java -cp calculator…"   11 minutes ago      Exited (0) 11 minutes ago             loving_cerf
8dcd5400fb59   316469bea218                   "java -cp calculator…"   21 minutes ago      Exited (1) 21 minutes ago             boring_boyd
f42132998cdb   316469bea218                   "java -cp calculator…"   22 minutes ago      Exited (1) 22 minutes ago             serene_williamson
cd7b3cc79362   33faaea61730                   "java -cp calculator…"   36 minutes ago      Up 34 minutes                         beautiful_villani
4ecb006ceaba   dc29a09f0117                   "java -cp Calculator…"   2 hours ago         Up 2 hours                            optimistic_pike
edce1b4092d0   dc29a09f0117                   "java -cp Calculator…"   2 hours ago         Exited (0) 2 hours ago                kind_banzai
cce1c0f8695e   dc29a09f0117                   "java -cp Calculator…"   2 hours ago         Up 2 hours                            blissful_dijkstra
3f7d4e6d09b3   dc29a09f0117                   "java -cp Calculator…"   2 hours ago         Exited (0) 2 hours ago                jolly_heisenberg
96d221e89bad   f234fbcb349e                   "/bin/sh -c 'mvn pac…"   4 days ago          Created                               loving_jepsen
a5fc459c1946   f234fbcb349e                   "/bin/sh -c 'mvn pac…"   4 days ago          Exited (127) 4 days ago               beautiful_haibt
8301173a0de7   f234fbcb349e                   "/bin/sh -c 'mvn pac…"   4 days ago          Exited (127) 4 days ago               jovial_edison
42b339a4513c   hello-world                    "/hello"                 4 days ago          Exited (0) 4 days ago                 nifty_carson
```

Figure 12: .

Now, we start the container with the image and check for the log file inside the container.

```
~$ docker start <container_id>
~$ docker exec -it <container_id> "/bin/bash"
```

```
vaishnavi@vaishnavi-Inspiron-5567:~$ docker start 8a5d55cc4ac9
8a5d55cc4ac9
vaishnavi@vaishnavi-Inspiron-5567:~$ docker exec -it 8a5d55cc4ac9 "/bin/bash"
root@8a5d55cc4ac9:/# ls
bin                                                    dev   lib64  proc  srv  var
boot                                                   etc   media  root  sys
calculatorDevops-1.0-SNAPSHOT-jar-with-dependencies.jar home  mnt    run   tmp
calculator_devops.log                                  lib   opt    sbin  usr
root@8a5d55cc4ac9:/#
```

Figure 13: Docker Images

9 CONTINUOUS MONITORING – ELK

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. ELK stack gives us the ability to aggregate logs from all the systems and applications, analyze these logs, and create visualizations for application and infrastructure monitoring, faster troubleshooting, security analytics, and more.
First, we run ElasticSearch(from the directory it is installed in) using,

```
~$ /bin/elasticsearch
```

Elastic Search runs on `http://localhost:9200`.
Next run Kibana(from the directory it is installed in) using,

```
~$ /bin/kibana
```

Kibana runs on `http://localhost:5601`.
Next run Logstash(from the directory it is installed in) using,

```
~$ /bin/logstash -f <path_conf_file>
```

Logstash runs on `https://localhost:9600`.

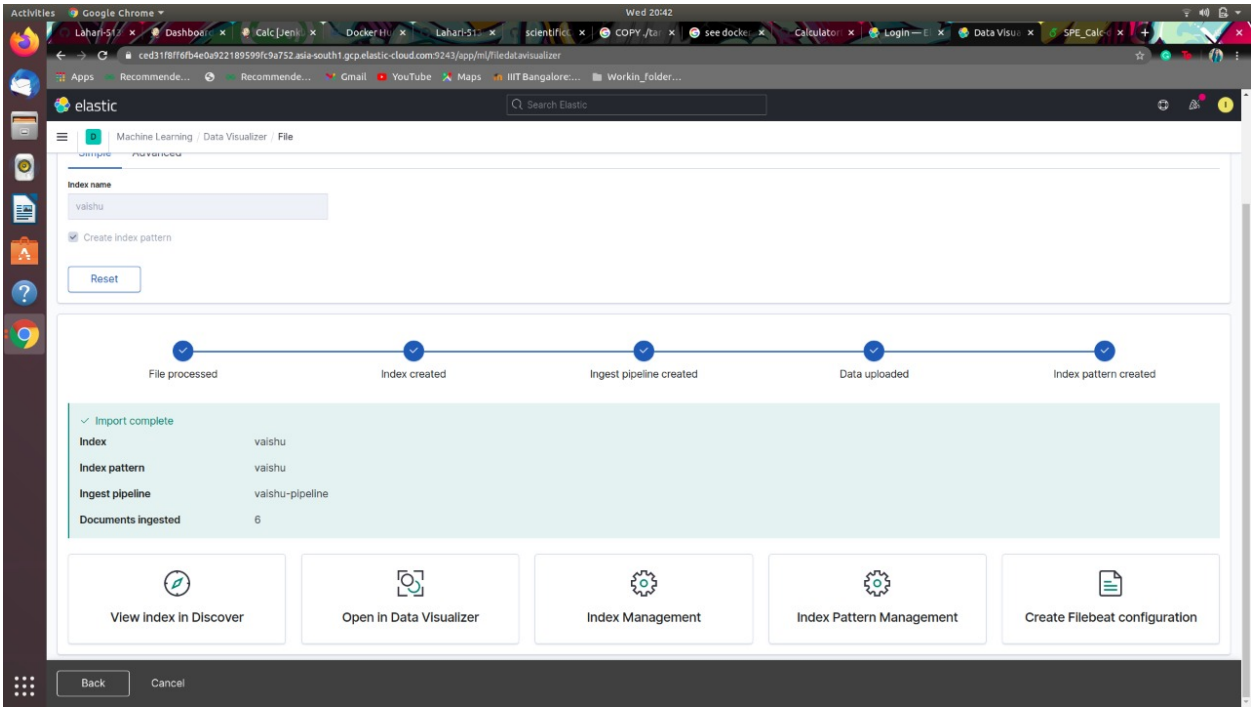

Figure 14: ELK

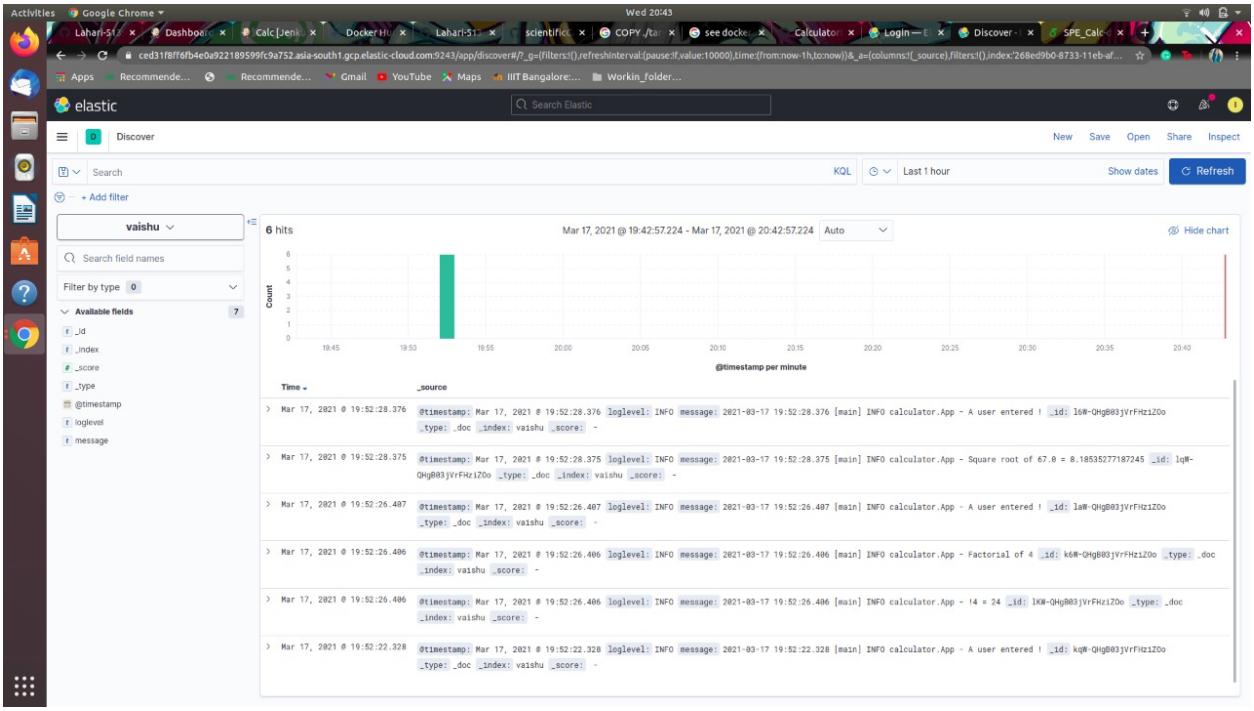

Figure 15: ELK

10 SCREENSHOTS OF THE RUNNING APPLICATION

```
vaishnavi@vaishnavi-Inspiron-5567:~$ docker run -it vaishnavi2902/calculator_devops
------------Calculator--------------
Choices of Operations:

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

Enter your choice(number):
1
You choose Square Root!!

Enter number: 4

The Result is 2.0

Choices of Operations:

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

```
Enter your choice(number):
2
You choose Factorial!!

Enter number: 4

The Result is 24

Choices of Operations:

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

```
Enter your choice(number):
3
You choose Natural Log!!

Enter number: 6

The Result is 1.791759469228055

Choices of Operations:

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

```
Enter your choice(number):
4
You choose Power!!

Enter number: 2
exponent: 7

The Result is 128.0

Choices of Operations:

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

Enter your choice(number):
5
vaishnavi@vaishnavi-Inspiron-5567:~$
```