Software Production Engineering

Mini Project - Scientific Calculator with DevOps

NIRAJ GUJARATHI : MT2021087

Introduction

I have developed a command-line scientific calculator application for the SPE mini project. It is a Java language based scientific calculator application with 4 operations -

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

The project is built using the maven build tool and, It is following DevOps practices.

What and Why DevOps ?

DevOps is a set of practices that combines Software Development (Dev) and IT Operations (Ops). It tries to implement continuous build - continuous integration and continuous deployment which indeed shorten the systems development life cycle and provide continuous delivery with quality software, used mainly for quick fixes.

Tools Used -

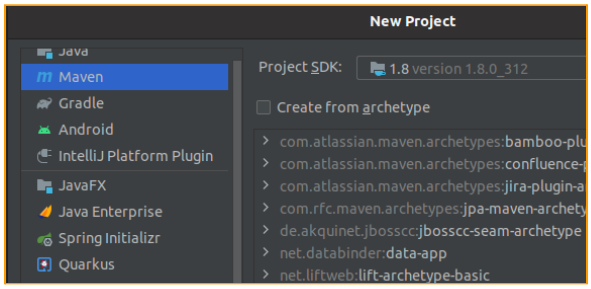
1. **Programming Language:** Java - 1.8 (JDK 8)
2. **Project Build / Project Management Tool:** maven (Apache)
3. **Unit testing Tool:** JUnit - version. 4.13.2
4. **SCM (Source Code Management):** Git with GitHub https://github.com/NirajGujarathi/calculator-devopsProject
5. **Logger / Log file generator:** log4j - version. 2.17.1
6. **Containerization:** Docker - DockerHub (public cloud repository for storing image) https://hub.docker.com/r/nirajg/calculator_devops/tags
7. **Continuous Integration:** Jenkins
8. **Continuous Deployment:** Ansible
9. **Monitoring Tool:** ELK Stack (visualization using - Kibana)

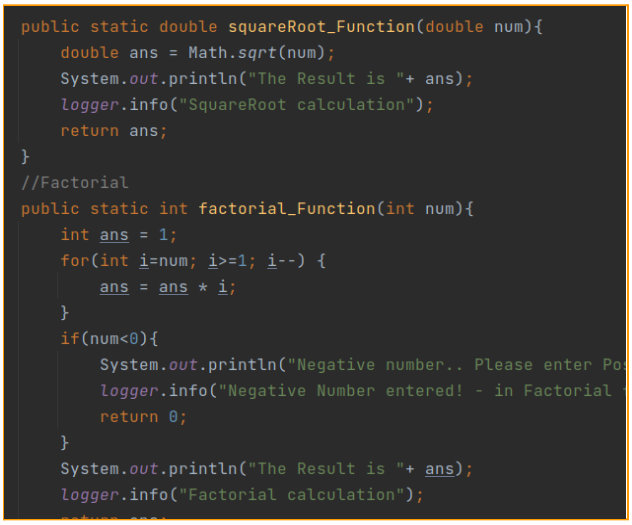
All links and Script files are attached at the end of the report .. (Please see below)

1. Programming Language - JAVA

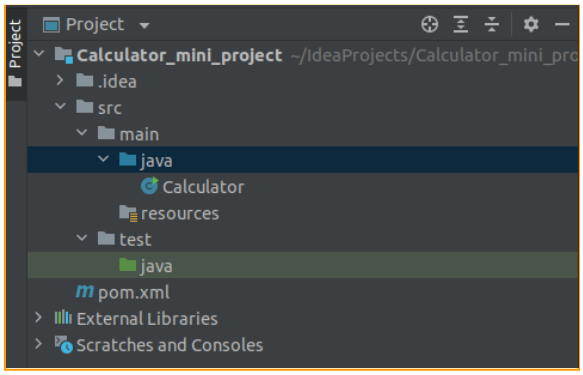
Implemented 4 functions for scientific calculator operations - added menu to choose which operation is to be performed, implemented switch case control structure to select / choose any one of the 4 operations in application.

Created maven project on IntelliJ IDE - it has created maven folder structure which contains src folder which will contain main function of application, test folder for JUnit test cases and pom.xml file which will handle the project dependencies.





```
public static double squareRoot_Function(double num){
    double ans = Math.sqrt(num);
    System.out.println("The Result is "+ ans);
    logger.info("SquareRoot calculation");
    return ans;
}
//Factorial
public static int factorial_Function(int num){
    int ans = 1;
    for(int i=num; i>=1; i--) {
        ans = ans * i;
    }
    if(num<0){
        System.out.println("Negative number.. Please enter Po
        logger.info("Negative Number entered! - in Factorial
        return 0;
    }
    System.out.println("The Result is "+ ans);
    logger.info("Factorial calculation");
```




```
}
public static double naturalLog_Function(double num){
    double ans = Math.log(num);
    System.out.println("The Result is "+ ans);
    logger.info("NaturalLog calculation");
    return ans;
}
public static double power_Function(double num, double exp){
    double ans = Math.pow(num,exp);
    System.out.println("The Result is "+ ans);
    logger.info("Power calculation");
    return ans;
```

Executing Calculator application on locally on IntelliJ

```
/usr/lib/jvm/java-8-openjdk-amd64/bin/java
-*-*-*-*-*-*-*-Calculator-*-*-*-*-*-*-*-

Calculator Devops Project
with following list of operations :
1. Square Root
2. Factorial
3. Natural Logarithm
4. Power
5. Exit from Application
Enter your choice(number): 1
Square Root - option selected

Enter number: 9
The Result is 3.0
```

2. Project Build / Project Management 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. In short terms we can say that maven is a tool that can be used for building and managing any Java-based project also useful in configuring all the dependencies it downloads these dependencies specified in pm.xml once it is downloaded from server it is stored in local project cache, and used for next builds.

Here, I have used maven which will be creating an executable .jar file which will package all dependencies specified in POM file; here dependencies such as JUnit and log4j will be bundled up with this executable binary jar in the target folder. pom.xml file manages the metadata, dependencies and plugins for the project.

Maven commands

mvn clean - Clean the project hierarchy and clear the cache in the Maven hierarchy, if target folder already present it will delete and create newly for fresh build

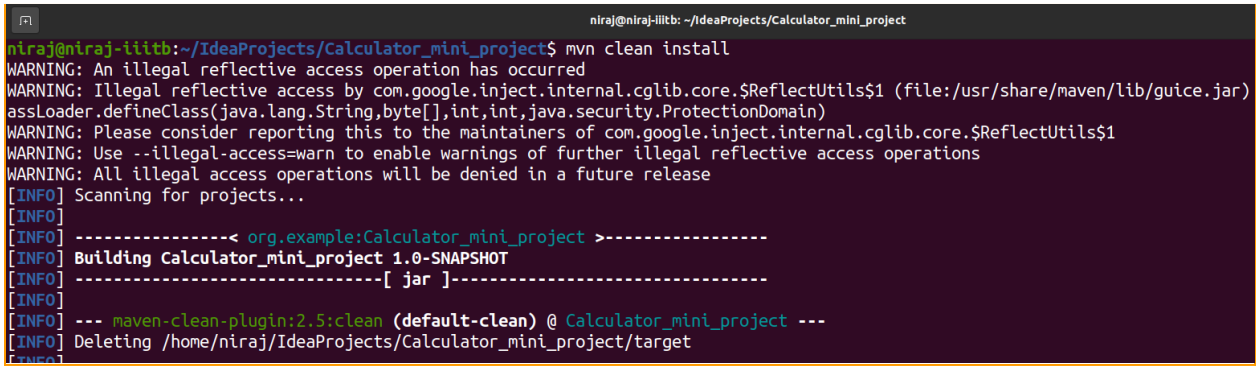
mvn compile - The main goal here is to compile all the source code files in the 'src/main/java/Application.Calculator' folder.

mvn install - Installs the package into the local repository, for use as a dependency in other projects locally

mvn package - take the compiled code and package it in its distributable format, such as a JAR.

I have used **mvn clean package** in pipeline script so that it can generate distributable jar file which can gets execute on docker container - it will compile the code and package compiled code for distributable format as well as builds project and runs test cases for unit testing (if any configured)

Run the command, mvn clean install



3. Unit testing Tool- JUnit (ver. 4.13.2)

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development. Following are snippets from the Calculator Project that implements two types of unit test cases for each of the calculator operations - method.
True positive assertEquals, where we expect the output of the

True negative, assertNotEquals, where we expect the method to not be equal to what we have stated / expected.

```
public class CalculatorTest {
    @Test
    public void test_squareRoot_AssertEquals() {  //true positive
        double num = 25.0;
        double expectedResult = 5.0;
        double result = Calculator.squareRoot_Function(num);
        Assert.assertEquals( message: "square root of a number for Tr
    }

    @Test
    public void test_squareRoot_AssertNotEquals(){
        double num = 36.0;
        double expectedResult = 4.0;
        double result = Calculator.squareRoot_Function(num);
        Assert.assertNotEquals( message: "square root of a number for
    }
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running Application.CalculatorTest
The Result is 2
The Result is 16.0
The Result is 5.0
The Result is 0.999896315728952
The Result is 1.791759469228055
The Result is 27.0
The Result is 6.0
The Result is 720
Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.729 sec

Results :

Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
```

Got test case failure during mvn compile phase.
It is a logical error caused in the test_power_False_positive function I have used assertEquals which has to be assertNotEquals since we are expecting result has to be false.

```
Results :

Failed tests:   test_power_False_Positive(Application.CalculatorTest): expected:<32.0> but was:<16.0>

Tests run: 8, Failures: 1, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  7.983 s
[INFO] Finished at: 2022-04-14T16:44:45+05:30
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test (default-test) on p
```

Rectified error and got maven build success

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.623 s
[INFO] Finished at: 2022-04-14T16:48:08+05:30
[INFO] ------------------------------------------------------------------------
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$
```

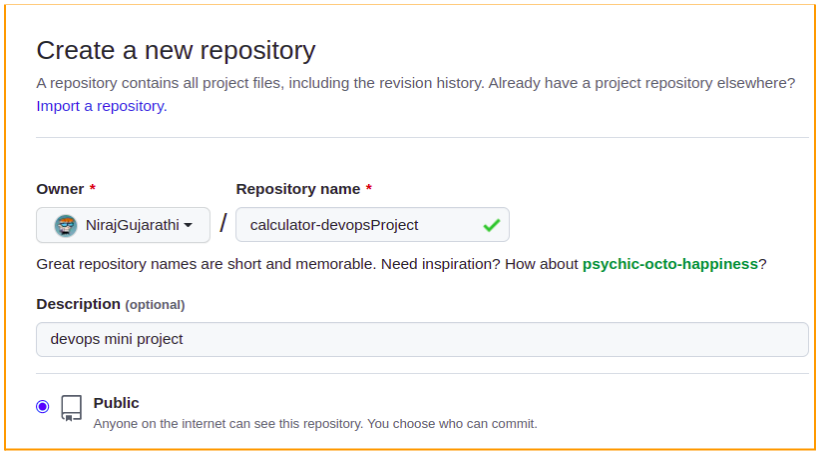
At end it creates executable jar with dependencies

```
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ Calculator_mini_project ---
[INFO] Building jar: /home/niraj/IdeaProjects/Calculator_mini_project/target/Calculator_mini_project-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-assembly-plugin:2.2-beta-5:single (default) @ Calculator_mini_project ---
```

4. Source Code management Tool - Git with GitHub

Source code management (SCM) is used to track modifications to a source code repository. SCM is also synonymous with Version Control System (VCS). SCM tracks the running history of changes as well as backup previous checkpoints in project codebase and helps resolve conflicts when merging updates, or any merge conflicts that happen when multiple developers working on the same module of project.

I have created a repository on GitHubhttps://github.com/NirajGujarathi/calculator-devopsProject and then configured the Maven project of my local repository to update the local folder structure to remote GitHub repo using Git commands.



Git Commands →

cd <path_to_souce code_folder>
git init - it will initialize my java maven project as a Git repository.
git remote add origin <GitHub_URL>.git - Link the local Git repository to the remote GitHub repository whose URL is specified in the command and the link is given the name origin

```
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git remote add origin https://github.com/NirajGujarathi/calculator-devopsProject.git
```

git add . - Adds all the unstaged files and untracked changes in the current directory to the staging area for committing.

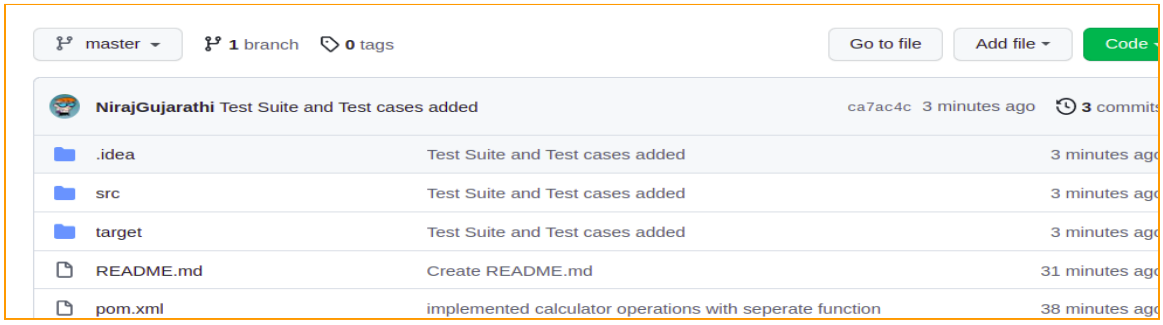
```
niraj@niraj-iiitb: ~/IdeaProjects/Calculator_mini_project
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git init
Initialized empty Git repository in /home/niraj/IdeaProjects/Calculator_mini_project/.git/
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git add .
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .idea/.gitignore
        new file:   .idea/compiler.xml
        new file:   .idea/jarRepositories.xml
        new file:   .idea/misc.xml
        new file:   .idea/uiDesigner.xml
        new file:   pom.xml
        new file:   src/main/java/Application/Calculator.java
        new file:   src/test/java/Application/CalculatorTest.java
```

git commit -m "<commit_message>" - Commit the added changes to the local Git repository also creates checkpoint for version history based on commit message one can get idea about changes happened on that steam / branch

```
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git commit -m "Test Suite and Test cases added"
[master ca7ac4c] Test Suite and Test cases added
 14 files changed, 168 insertions(+), 7 deletions(-)
 create mode 100644 .idea/vcs.xml
 create mode 100644 target/Calculator_mini_project-1.0-SNAPSHOT-jar-with-dependencies.jar
 create mode 100644 target/Calculator_mini_project-1.0-SNAPSHOT.jar
 create mode 100644 target/classes/Application/Calculator.class
 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/Application.CalculatorTest.txt
 create mode 100644 target/surefire-reports/TEST-Application.CalculatorTest.xml
 create mode 100644 target/test-classes/Application/CalculatorTest.class
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git push -u origin master
Username for 'https://github.com': NirajGujarathi
Password for 'https://NirajGujarathi@github.com':
Enumerating objects: 47, done.
Counting objects: 100% (47/47), done.
Delta compression using up to 4 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (37/37), 1.78 MiB | 2.89 MiB/s, done.
Total 37 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/NirajGujarathi/calculator-devopsProject.git
   4aef442..ca7ac4c  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

git push -u origin master - Push the latest commits from the local Git repository master branch to the remote GitHub master branch

```
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git push -u origin master
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (18/18), 4.10 KiB | 524.00 KiB/s, done.
Total 18 (delta 0), reused 0 (delta 0)
To https://github.com/NirajGujarathi/calculator-devopsProject.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 742 bytes | 247.00 KiB/s, done.
From https://github.com/NirajGujarathi/calculator-devopsProject
 * branch            master     -> FETCH_HEAD
   ed9fc0d..4aef442  master     -> origin/master
Updating ed9fc0d..4aef442
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```



5. Logger / Log File Generator - log4j (2.17.1 - latest patched version no vulnerability :))

Logging keeps track of all the operations that were performed in the application, warnings, debugging information, errors etc. Apache log4j is a Java-based logging utility originally written by Ceki Gülcü. log4j is java logging framework - these dependency is added to maven pom.xml file so that final executable jar file gets bundled with these dependency of log4j

Its configuration is provided via the log4j2.xml file which is stored under the resources directory of the project and for this project, it is configured as follows,

```
<?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="FileAppender"/>
        </Root>
    </Loggers>
</Configuration>
```

Now we need to add logger functions in the main application code which uses the logger which is initialized as,

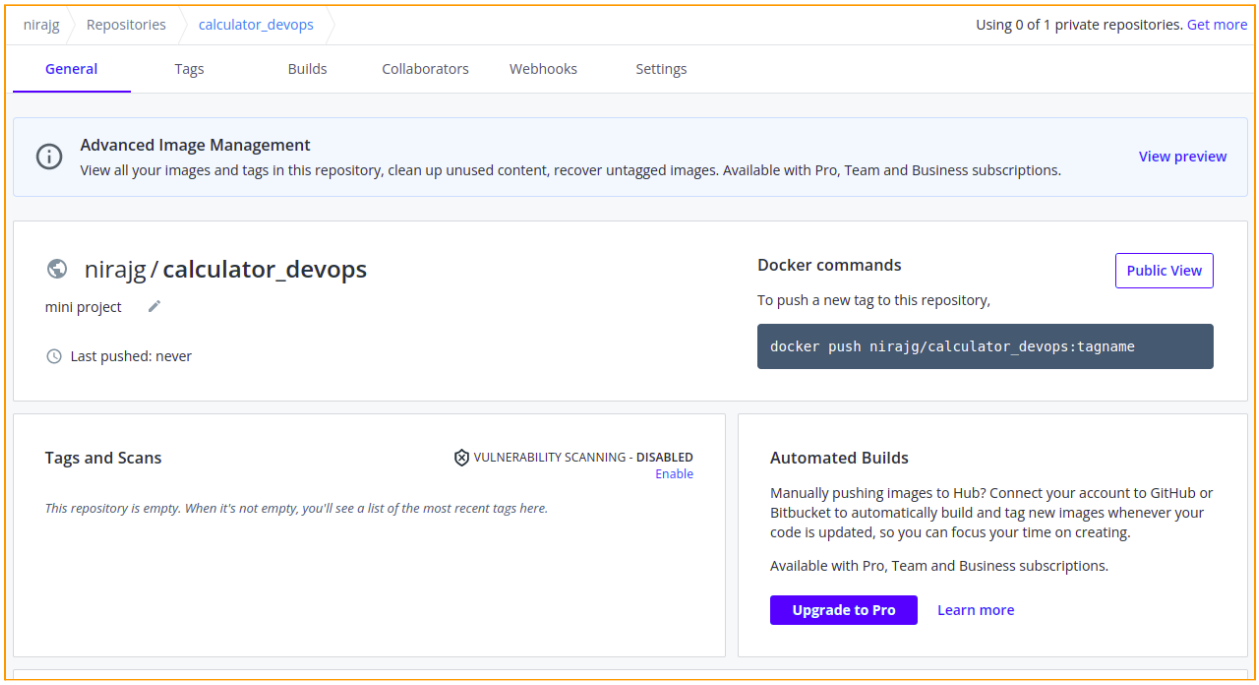
private static final Logger logger = LogManager.getLogger(Calculator.class);

And then the logger can be called to generate log messages as follows,

logger.info("SquareRoot calculation");
logger.info("Factorial calculation");
logger.info("NaturalLog calculation");
logger.info("Power calculation");
logger.info("Invalid Input! Closing Application");

6. Containerization - Docker

Docker is an open platform for developing, shipping, and running applications. Docker helps you to seperate your applications from your infrastructure so you can deliver software quickly.
Docker containerization - it is OS level virtualization it is a modern day technique used instead of VM machines for deployment of software.
Using Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.
Created docker hub repo - https://hub.docker.com/repository/docker/nirajg/calculator_devops

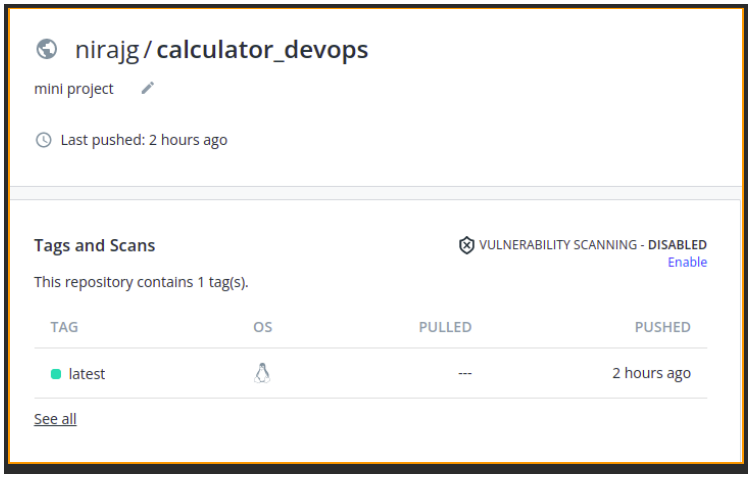


Docker builds images automatically by reading the instructions from a Dockerfile - a text file that contains all commands, in order, needed to build a given image.
In this project, we need to create a container that executes the built Maven project. JAR file that is being created using the following command.
java -cp <path_to_jar_file> <class_package_name>

Dockerfile

```
FROM openjdk:8
COPY ./target/Calculator_mini_project-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java","-cp","Calculator_mini_project-1.0-SNAPSHOT-jar-with-dependencies.jar","Application.Calculator"]
```

Once image is created by following steps in Dockerfile, next step is to push this image into DockerHub which is going to be automated by Jenkins Pipeline Script *(covered in next section)*



7. Continuous Integration - jenkins

Jenkins - Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

Install ssh server on machine and login as jenkins user

- *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.

- *cd . ssh*
- *ssh-keygen –t rsa*
- *ssh–copy–id niraj@localhost*
- *ssh niraj@localhost*

After executing the last command, we get automatically directed outside the Jenkins user.

```
jenkins@niraj-iiitb:~$ cd .ssh
jenkins@niraj-iiitb:~/.ssh$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins
/var/lib/jenkins/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkin
Your public key has been saved in /var/lib/jenkins/.s
The key fingerprint is:
SHA256:Afu54A50UqZiUIO5txv+uN4Jvob7foGMseL24eGW+FU je
The key's randomart image is:
+---[RSA 3072]----+
| oo   .          |
|o. .   o         |
|..    + .        |
|o..  + . o       |
| *oo+ oES        |
|+.=o.+.. .       |
|ooo=oo. .        |
|.=**=+           |
|o*%X+ .          |
+----[SHA256]-----+
```

```
Now try logging into the machine, with:   "ssh 'niraj@localhost'"
and check to make sure that only the key(s) you wanted were added.

jenkins@niraj-iiitb:~/.ssh$ ssh niraj@loclahost
ssh: Could not resolve hostname loclahost: Temporary failure in name resolution
```

```
jenkins@niraj-iiitb:~/.ssh$ ssh niraj@localhost
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-39-generic x86_64)

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

4 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
```

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.

```
                                                    niraj@niraj-iiitb: ~
niraj@niraj-iiitb:~$ systemctl start jenkins
```

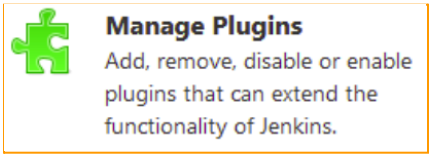
Getting permission denied error when jenkin user is accessing the docker.sock changed permission of that file to executable permission chmod 777

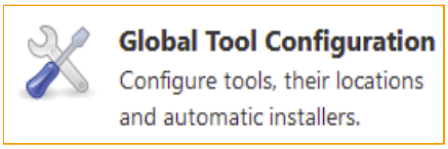
Solution - Adding docker and jenkins user into same group so that jenkins can access docker.sock and giving executable permission to that group

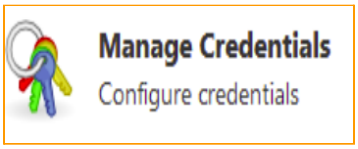
```
+ docker pull node:6-alpine
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post
"http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/create?fromImage=node&tag=6-alpine": dial unix /var/run/docker.sock: connect: permission denied
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
```

```
niraj@niraj-iiitb:~/IdeaProjects/CalculatorDevOps-main$ usermod -aG docker jenkins
usermod: Permission denied.
usermod: cannot lock /etc/passwd; try again later.
niraj@niraj-iiitb:~/IdeaProjects/CalculatorDevOps-main$ sudo usermod -aG docker jenkins
[sudo] password for niraj:
niraj@niraj-iiitb:~/IdeaProjects/CalculatorDevOps-main$ sudo usermod -aG docker jenkins
niraj@niraj-iiitb:~/IdeaProjects/CalculatorDevOps-main$ sudo usermod -aG docker jenkins
niraj@niraj-iiitb:~/IdeaProjects/CalculatorDevOps-main$ sudo chmod 777 /var/run/docker.sock
niraj@niraj-iiitb:~/IdeaProjects/CalculatorDevOps-main$
```

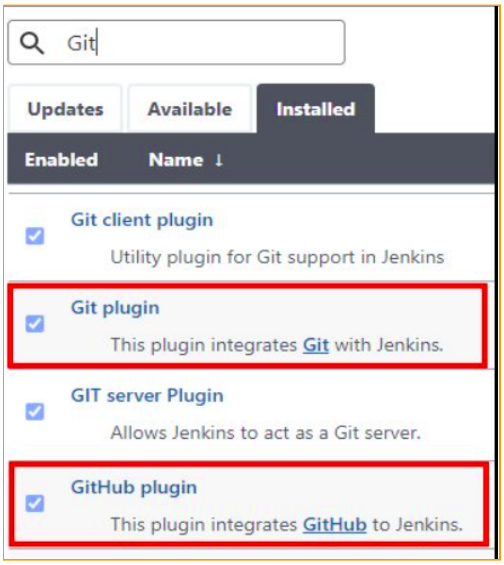
Manage Plugins in Jenkins, Add credentials and Configurations of installed Plugins
We need to install all the required plugins like Build pipeline, Docker, GitHub, Maven Integration, Ansible etc. After they are all done installing, we need to restart Jenkins and add Docker credentials.

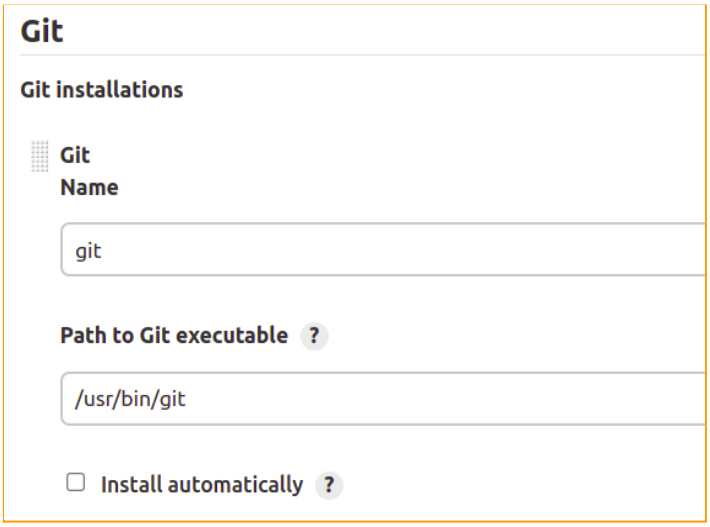




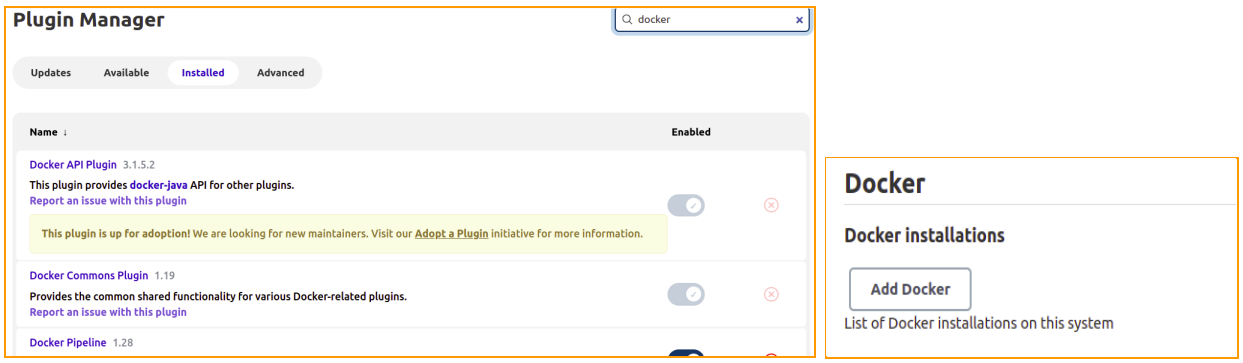


Configuring Git and GitHub with Jenkins - Jenkins allows you to schedule your build and facilitates easy transfer of data from the GitHub repository to Jenkins machine.
Installing GitHub plugins





Installing docker api plugins

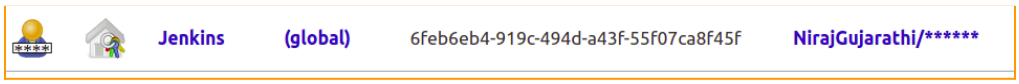


Installing Ansible Plugins

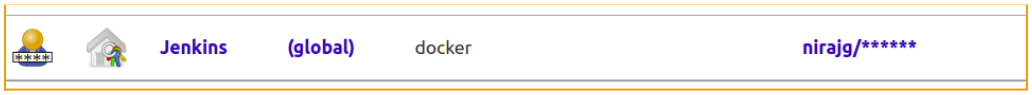
Add credentials Mange Jenkins → Manage Credentials

In the Jenkins dashboard we add credentials for GitHub which will bemused to pull code from remote repository also add credentials of the DockerHub repository and we set an unique id which is “docker” this ID will be used at stage 4 and stage 5 in Pipeline Script for pushing Docker Image image to docker hub.

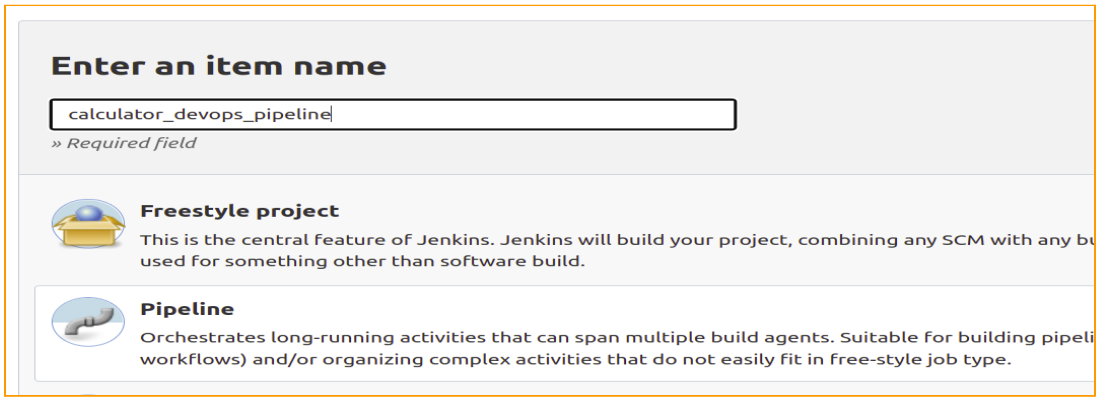
Adding GitHub credentials to jenkins key management -

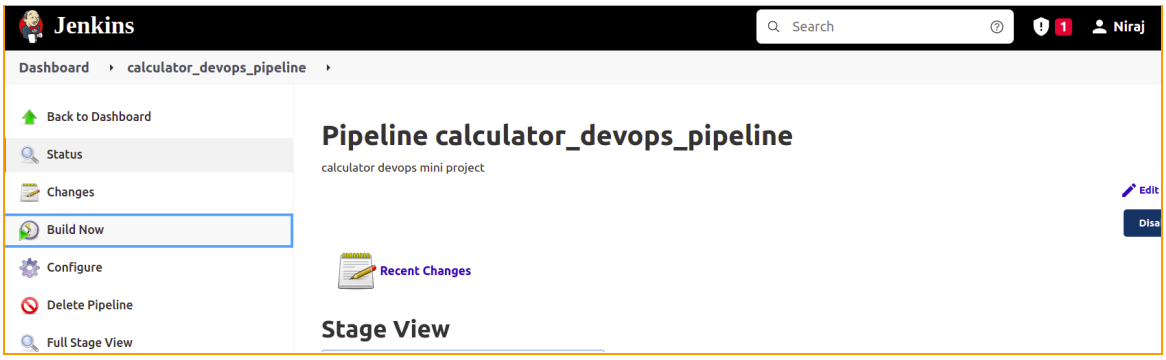


Adding DockerHub credentials to push image to dockerhub -

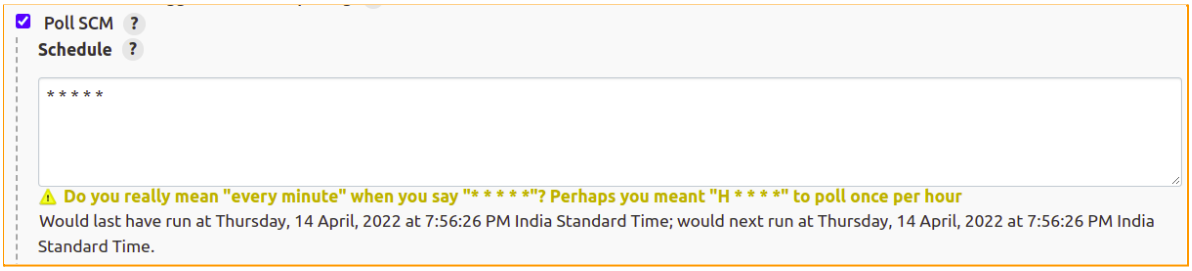


Create a New Pipeline in Jenkins - Create new item, and select pipeline.





Adding Poll SCM whenever ever new changes made into GitHub Repository it will automatically triggers the Jenkins Pipeline Build after every minute



Pipeline Script Code

Script

```
pipeline{
    environment{
            docker_image = ""
            registryCredential='docker'
    }
    agent any
    stages{
        stage('Stage 1: Git Clone Project'){
            steps{
                git branch: 'master', url: 'https://github.com/NirajGujarathi/calculator-devopsProject.git'
            }
        }
        stage('Stage 2: Maven Build and JUnit Testing'){
            steps{
                sh 'mvn clean package'
            }
        }
        stage('Stage 3: Building Docker Image'){
            steps {
                script {
                    docker_image = docker.build "nirajg/calculator_devops:latest"
                }
            }
        }
        stage('Stage 4: login to dockerhub and push docker image') {
            steps {
                script {
                    docker.withRegistry('', 'docker') {
                        docker_image.push()
                    }
                }
            }
        }
        stage('Stage 5: Ansible Deployment to machine'){
            steps{
                ansiblePlaybook becomeUser: 'null', colorized: true, credentialsId: 'docker', installation: 'A
            }
        }
    }
}
```

8. 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.

Ansible Settings -

In Calculator Project working directory, I have kept all Ansible Deployment related file into new directory named- AnsibleDeployment

Need to be careful about giving the correct path of docker image and the correct python version

Which contains **inventory file**

```
localhost ansible_user=niraj
```

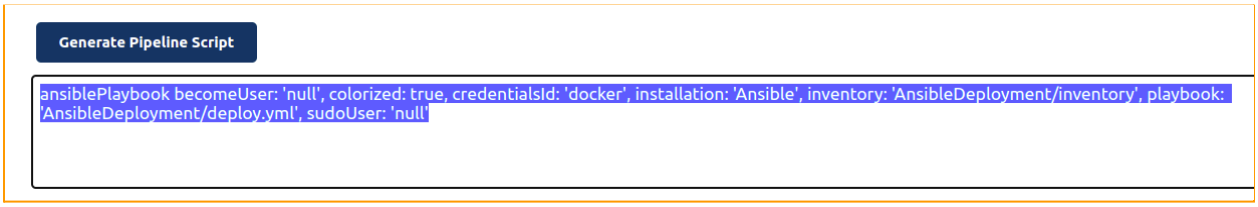
And **deploy.yml** file which is procedure or steps to be executed at ansible machine / server end (cookbook in case chef)

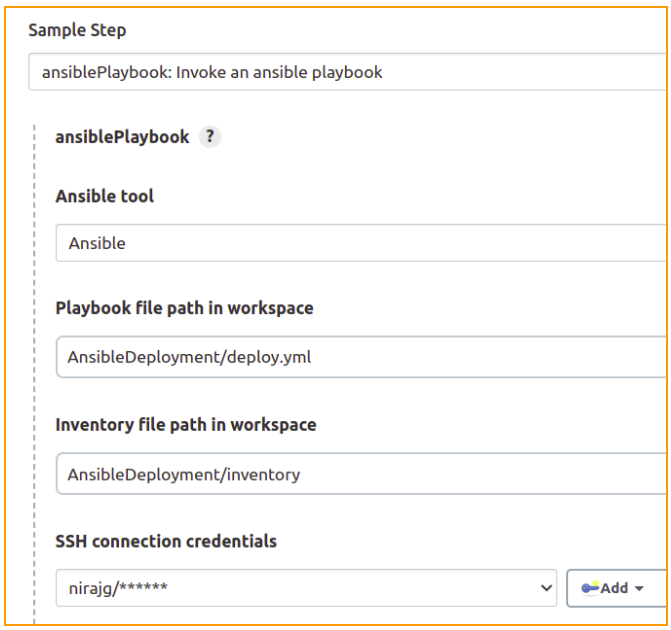
```
---
- name: Pull Docker image of calculator_devops from DockerHub
  hosts: all
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Pull image
      docker_image:
        name: nirajg/calculator_devops
        source: pull
```

Ansible Deployment Pipeline Script at Stage 5

Ansible Plugins are added into jenkins now generating Ansible Pipeline Syntax

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





Errors Encountered and Solutions in Pipeline Executions

Output - stage 2 maven build and test build is successful

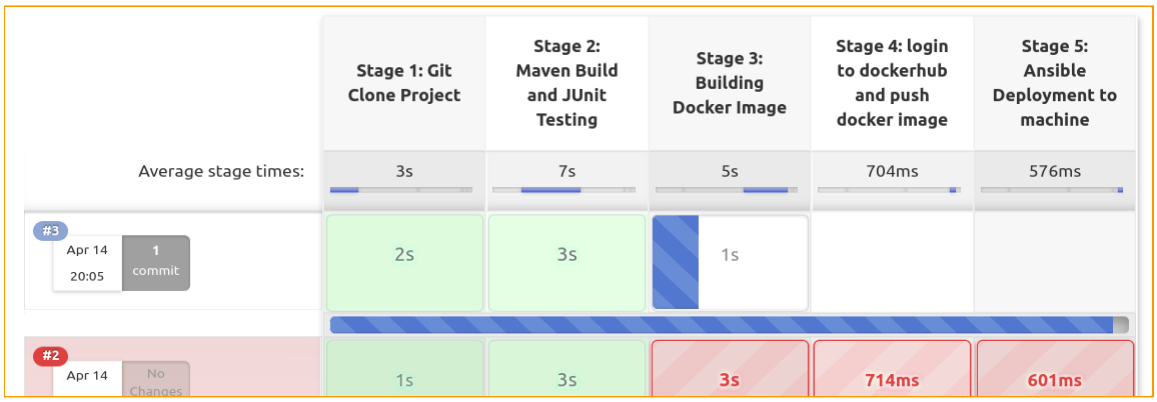
```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running Application.CalculatorTest
The Result is 2
The Result is 16.0
The Result is 5.0
The Result is 0.999896315728952
The Result is 1.791759469228055
The Result is 27.0
The Result is 6.0
The Result is 720
Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.305 sec

Results :

Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
```

Error in stage 3 - Inside docker file it is unable to locate jar file

Solution jar file is getting stored in *./target/<jar_file_name>.jar* class-path name is not mentioned correctly, modified Dockerfile with correct path of jar_file

Changed Dockerfile and pushed into github repository - due to poll SCM post minute pipeline triggered automatically

```
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git add .
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git commit -m "changes in jar_name
[master cbbd477] changes in jar_name in DockerFile
 1 file changed, 2 insertions(+), 2 deletions(-)
niraj@niraj-iiitb:~/IdeaProjects/Calculator_mini_project$ git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
```



Also failing since I have used only maven clean in stage 2 - changed pipeline script
It should be maven clean package to package code into executable jar and create jar file

Issue solved of stage 3

Stage Logs (Stage 3: Building Docker Image)

Checks if running on a Unix-like node (self time 266ms)

Shell Script -- docker build -t "$JD_IMAGE" . (self time 10s)

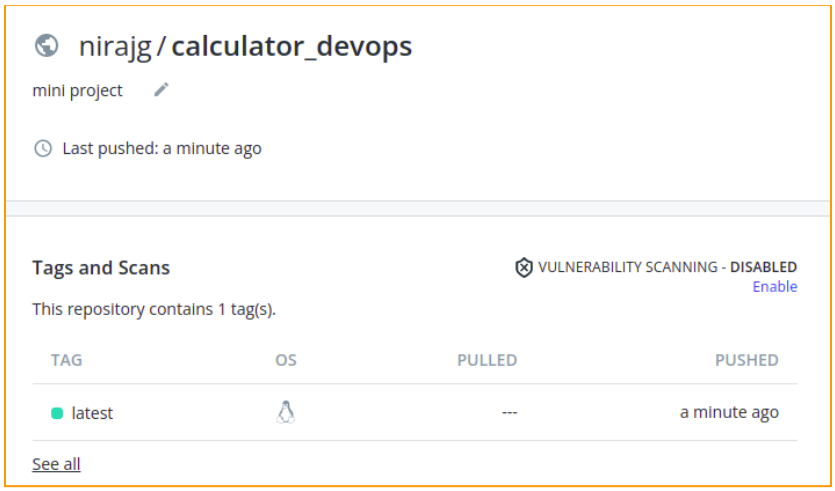
```
+ docker build -t nirajg/calculator_devops:latest .
Sending build context to Docker daemon  5.995MB

Step 1/4 : FROM openjdk:8
 ---> 18fbe41f975e
Step 2/4 : COPY ./target/Calculator_mini_project-1.0-SNAPSHOT-jar-with-depen
 ---> f088d5200c88
Step 3/4 : WORKDIR ./
 ---> Running in 5c1c45c7bfdf
Removing intermediate container 5c1c45c7bfdf
 ---> 7104d2eec855
Step 4/4 : CMD ["java","-cp","Calculator_mini_project-1.0-SNAPSHOT-jar-with-
 ---> Running in 09550ffc8af1
Removing intermediate container 09550ffc8af1
 ---> 7a41a15858f9
Successfully built 7a41a15858f9
Successfully tagged nirajg/calculator_devops:latest
```

Error - Stage 4 is not executing - failing due to invalid docker credentials - Unable to build image and push to docker hub

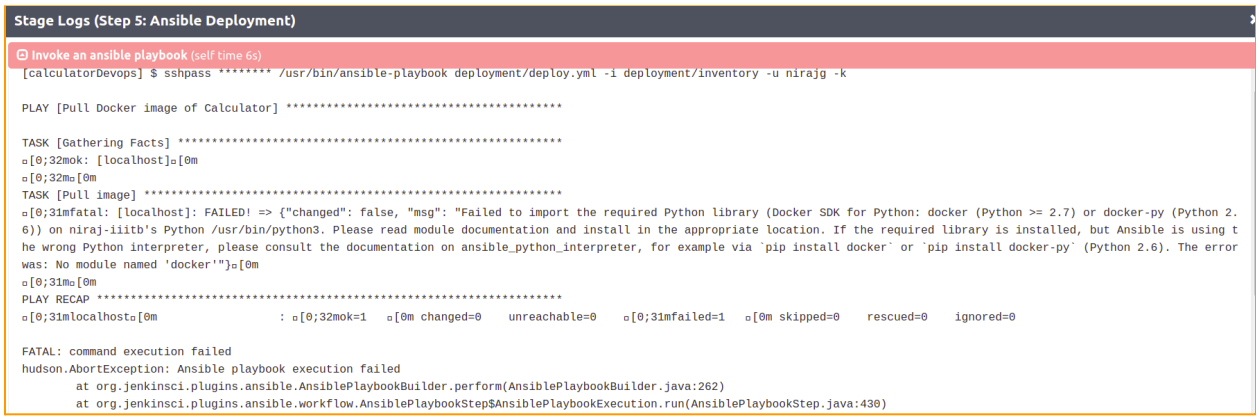
Solution - Resolved by making changes into jenkins docker hub manage credentials

```
Successfully built 7a41a15858f9
Successfully tagged nirajg/calculator_devops:latest
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Stage 4: login to dockerhub and push docker image)
[Pipeline] script
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withDockerRegistry
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // withEnv
```

At end of stage 4 - Image pushed successful to DockerHub



Error - At stage 5 Ansible Deployment error due to docker.py is not present in container (localhost)



Solution -

- *sudo apt-get install pip*
- *pip install docker.py*

```
Processing triggers for man-db (2.9.1-1) ...
niraj@niraj-iiitb:~/IdeaProjects/CalculatorDevOps-main$ pip install docker-py
Collecting docker-py
  Downloading docker_py-1.10.6-py2.py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 495 kB/s
Collecting docker-pycreds>=0.2.1
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl (9.0 kB)
Requirement already satisfied: six>=1.4.0 in /usr/lib/python3/dist-packages (from docker-py) (1.14.0)
Requirement already satisfied: requests!=2.11.0,>=2.5.2 in /usr/lib/python3/dist-packages (from docker-py) (2.22.0)
Collecting websocket-client>=0.32.0
  Downloading websocket_client-1.3.2-py3-none-any.whl (54 kB)
     |████████████████████████████████| 54 kB 331 kB/s
Installing collected packages: docker-pycreds, websocket-client, docker-py
  WARNING: The script wsdump is installed in '/home/niraj/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed docker-py-1.10.6 docker-pycreds-0.4.0 websocket-client-1.3.2
niraj@niraj-iiitb:~/IdeaProjects/CalculatorDevOps-main$
```

After all 5 stages pipeline executed successfully

Stage View

	Stage 1: Git Clone Project	Stage 2: Maven Build and JUnit Testing	Stage 3: Building Docker Image	Stage 4: login to dockerhub and push docker image	Stage 5: Ansible Deployment to machine
Average stage times: (Average full run time: ~1min 34s)	2s	7s	8s	16s	4s
#7 Apr 14 20:44 — No Changes	1s	10s	11s	44s	6s
#6 Apr 14 20:38 — 1 commit	2s	9s	11s	1min 3s	20s
#5 Apr 14 20:19 — 1 commit	1s	9s	13s	2s failed	612ms failed
#4 Apr 14 20:08 — No Changes	1s	3s	3s failed	602ms failed	534ms failed
#3 Apr 14 20:05 — 1 commit	2s	3s	3s failed	610ms failed	612ms failed
#2 Apr 14 19:58 — No Changes	1s	3s	3s failed	714ms failed	601ms failed
#1 Apr 14 19:57 — No Changes	6s	14s	11s failed	694ms failed	552ms failed

Running Calculator Application on Container Image

To check latest images of calculator_devops -

- ***docker images***
- ***docker ps -a***

```
niraj@niraj-iiitb:~$ docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
nirajg/calculator_devops            latest    032ee1ac1889   22 hours ago    528MB
nirajg/calculator_devops            <none>    1455c77abaee   47 hours ago    528MB
nirajg/calculator_devops            <none>    b932307a9543   2 days ago      528MB
nirajg/calculator_devops            <none>    7e2882b46612   2 days ago      528MB
```

To run application image on container

- ***docker run -it nirajg/calculator_devops***

```
niraj@niraj-iiitb:~$ docker run -it nirajg/calculator_devops
-*-*-*-*-*-*-*-Calculator-*-*-*-*-*-*-*-

Calculator Devops Project
with following list of operations :
1. Square Root
2. Factorial
3. Natural Logarithm
4. Power
5. Exit from Application
Enter your choice(number): 1
Square Root - option selected

Enter number: 4
The Result is 2.0
```

```
Calculator Devops Project
with following list of operations
1. Square Root
2. Factorial
3. Natural Logarithm
4. Power
5. Exit from Application
Enter your choice(number): 2
Factorial - option selected

Enter number: 6
The Result is 720

Calculator Devops Project
with following list of operations
1. Square Root
2. Factorial
3. Natural Logarithm
4. Power
5. Exit from Application
Enter your choice(number): 1
Square Root - option selected

Enter number: 8
The Result is 2.8284271247461903
```

To execute a container with the latest **container-ID** here I get the latest log file of recently executed operations.

docker exec -it <Container_ID> '/bin/bash'

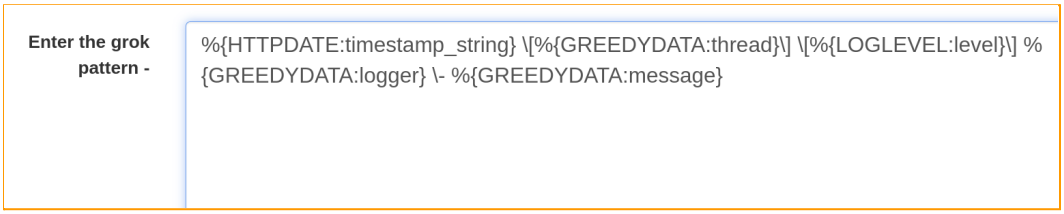
```
niraj@niraj-iiitb:~$ docker ps -a
CONTAINER ID   IMAGE                       COMMAND                  CREATED       STATUS
f3561b8edad5   nirajg/calculator_devops    "java -cp Calculator…"   3 hours ago   Up 7 minut
a99193af95c3   b932307a9543                "java -cp Calculator…"   2 days ago    Exited (0)
45de921f0d69   abf                         "java -cp calculator…"   3 days ago    Exited (0)
```

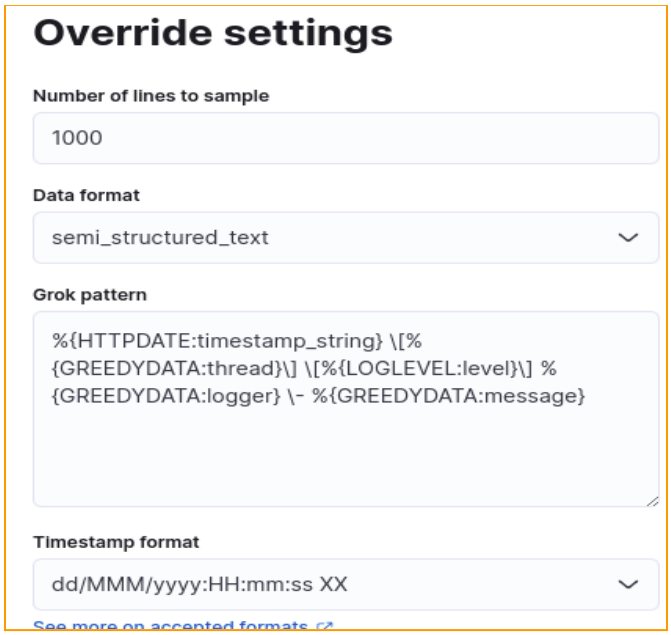
```
niraj@niraj-iiitb:~$ docker exec -it f3561b '/bin/bash'
root@f3561b8edad5:/# ls
Calculator_mini_project-1.0-SNAPSHOT-jar-with-dependencies.jar  boot                  dev
bin                                                             calculator_devops.log  etc
```

9. Monitoring Tool: ELK Stack

"**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.

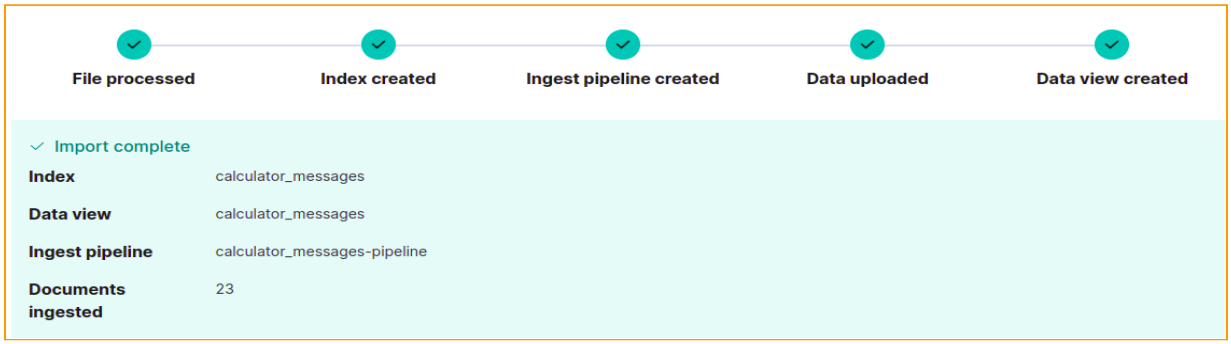
Kibana Visualization of log-file using - **GROK** pattern to decode logging messages



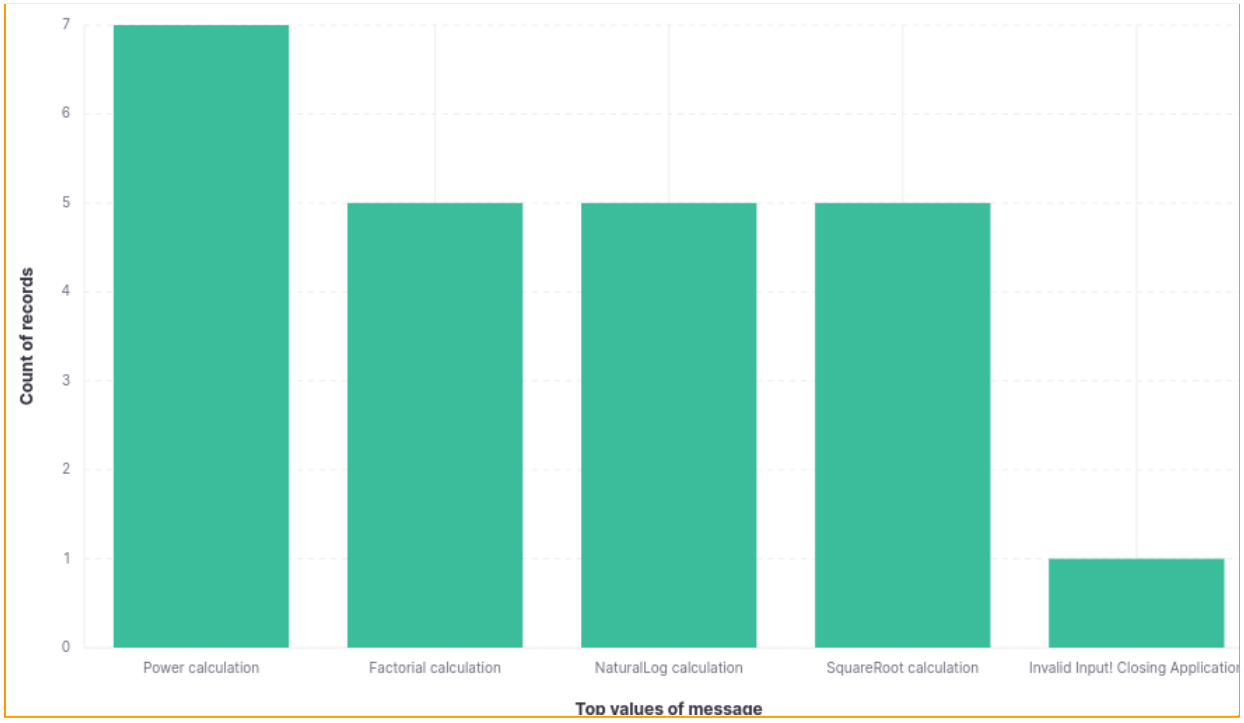


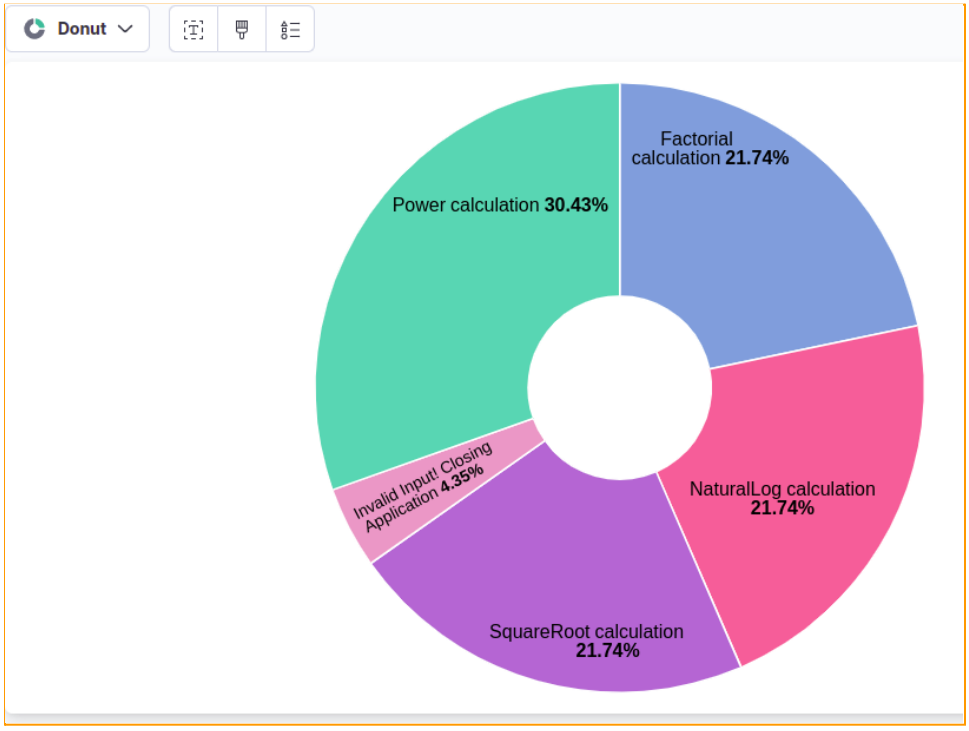
GROK pattern →

%{HTTPDATE:timestamp_string} \[%{GREEDYDATA:thread}\] \[%{LOGLEVEL:level}\] %{GREEDYDATA:logger} \- %{GREEDYDATA:message}



Kibana Visualization





Log file →

```
root@f3561b8edad5:/# cat calculator_devops.log
16/Apr/2022:14:53:41 166 [Calculator.java] [INFO] Application.Calculator - SquareRoot calculation
16/Apr/2022:14:53:47 385 [Calculator.java] [INFO] Application.Calculator - SquareRoot calculation
16/Apr/2022:14:53:53 635 [Calculator.java] [INFO] Application.Calculator - NaturalLog calculation
16/Apr/2022:14:53:59 567 [Calculator.java] [INFO] Application.Calculator - Factorial calculation
16/Apr/2022:14:54:04 699 [Calculator.java] [INFO] Application.Calculator - Factorial calculation
```

URLs and Script File

GitHub -

Repository - https://github.com/NirajGujarathi/calculator-devopsProject

DockerHub -

Docker Image - https://hub.docker.com/r/nirajg/calculator_devops/tags

Repository - https://hub.docker.com/r/nirajg/calculator_devops

Pipeline script -

```
pipeline{
  environment{
      docker_image = ""
      registryCredential='docker'
  }
  agent any
  stages{
    stage('Stage 1: Git Clone Project'){
      steps{
        git branch: 'master', url: 'https://github.com/NirajGujarathi/calculator-devopsProject.git'
      }
    }
    stage('Stage 2: Maven Build and JUnit Testing'){
      steps{
        sh 'mvn clean package'
      }
    }
    stage('Stage 3: Building Docker Image'){
      steps {
        script {
          docker_image = docker.build "nirajg/calculator_devops:latest"
        }
      }
    }
    stage('Stage 4: login to dockerhub and push docker image') {
      steps {
        script {
          docker.withRegistry('', 'docker') {
            docker_image.push()
          }
        }
      }
    }
    stage('Stage 5: Ansible Deployment to machine'){
      steps{
        ansiblePlaybook becomeUser: 'null', colorized: true, credentialsId: 'docker', installation: 'Ansible', inventory:
'AnsibleDeployment/inventory', playbook: 'AnsibleDeployment/deploy.yml', sudoUser: 'null'
      }
    }
  }
}
```

Dockerfile -

```
FROM openjdk:8
COPY ./target/Calculator_mini_project-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java","-cp","Calculator_mini_project-1.0-SNAPSHOT-jar-with-dependencies.jar","Application.Calculator"]
```

Inventory file -

```
localhost ansible_user=niraj
```

deploy.yml playbook file -

```
---
- name: Pull Docker image of calculator_devops from DockerHub
  hosts: all
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Pull image
      docker_image:
        name: nirajg/calculator_devops
        source: pull
```

GROK pattern -

```
%{HTTPDATE:timestamp_string} \[%{GREEDYDATA:thread}\] \[%{LOGLEVEL:level}\] %{GREEDYDATA:logger} \-
%{GREEDYDATA:message}
```

Thank You !