

CS 816 – Software Production Engineering

Mini Project – Scientific Calculator with DevOps

MT2022161-Aditya.M

Problem Statement:

Create a scientific calculator program with user menu driven operations

- Square root function - \sqrt{x}
- Factorial function - $x!$
- Natural logarithm (base e) - $\ln(x)$
- Power function – x^b

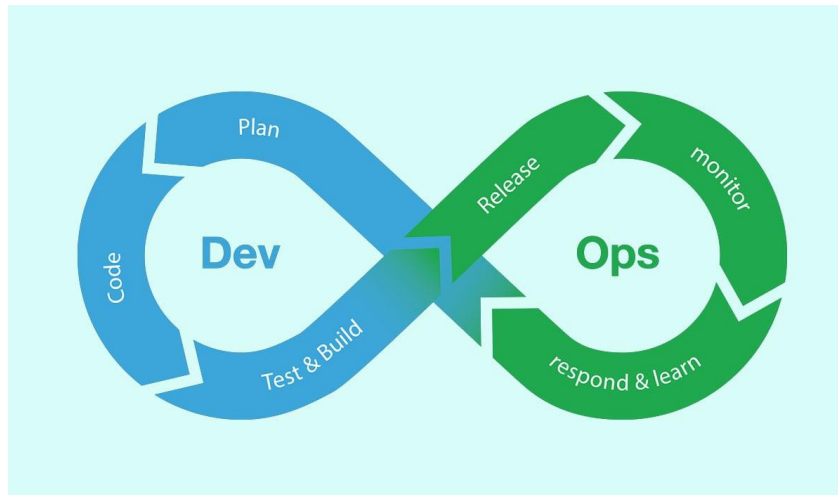
Links:

- Source Code: [adityagowda2000/SPE-Mini-Project \(github.com\)](https://github.com/adityagowda2000/SPE-Mini-Project)
- Docker Image: [adityagowda2000/spe-mini-project general | Docker Hub](https://hub.docker.com/r/adityagowda2000/spe-mini-project-general)



DevOps:

➤ What is DevOps?

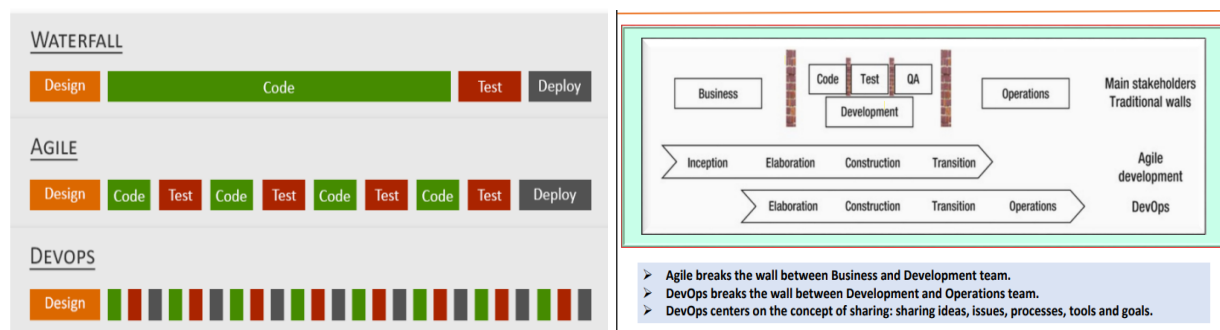


DevOps is a methodology in the software development and IT industry. Used as a set of practices and tools, DevOps integrates and automates the work of software development (*Dev*) and IT operations (*Ops*) as a means for improving and shortening the systems development life cycle. DevOps is complementary to agile software development several DevOps aspects came from the *agile* way of working.

DevOps enables formerly siloed roles—development, IT operations, quality engineering, and security—to coordinate and collaborate to produce better, more reliable products. By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster.

Teams have many DevOps tools to help them facilitate a DevOps culture in their organization. Most teams rely on several tools, building custom toolchains that fit their needs for each phase in the application lifecycle. While adopting a specific tool or technology is not the same as adopting DevOps, when the DevOps culture is present and the processes are defined, people can implement and streamline DevOps practices if they choose the proper tools.

- **Dev-** People involved in developing product
- **Ops** – System engineers, administrators, operations staff, release engineers, DBAs, network engineers and Security professionals.
- **Agile Software Development** – collaboration of customers, product management, developers and QA to fill in the gaps and rapidly iterate towards a better product.
- **DevOps** – extending Agile principles beyond the boundaries of “the code” to the entire delivered service.



➤ Why DevOps?

- DevOps enables faster development of new products and easier maintenance of existing deployments. DevOps also

promotes frequent code versions, reduces the implementation failure, and provides faster recovery time. DevOps has been proven to increase the speed, efficiency and quality of software delivery as well as improving staff morale and motivation.

- Goals of DevOps-
 - Improved deployment frequency
 - Faster time to market
 - Lower failure rate of new release
 - Shortened lead time between fixes
 - Faster mean time to recovery in the event of a new release crashing.

DevOps Tools Used:

- **Git & GitHub**- Git is a distributed version control system which is used for management of changes /versions of your project source code. GitHub is a web based , git file hosting service which enables us to share/showcase our project and files to others.
- **Maven** -Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
- **Jenkins**- Jenkins is a Java-based open-source automation platform with Continuous Integration (CI) plugins. Jenkins is used to produce and test software projects on a regular basis, making it easier for

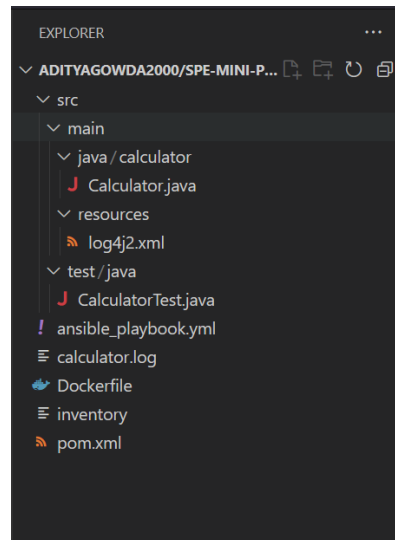
developers to incorporate changes and for users to get a new build. The Jenkins pipeline was utilized in this project to handle until delivery, i.e. continuous delivery

- **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.
- **Ansible**-Ansible is an open-source configuration management tool that provides a minimalist server automation framework based on YAML definitions. Its simplified infrastructure requirements and accessible syntax helped make Ansible one of the most popular configuration management tools to date.
- **ELK Stack**- ELK stack is an acronym for three open-source projects: Elasticsearch, Logstash, and Kibana. It is a log analysis platform that allows you to aggregate, process, and visualize data from multiple sources.
- **Ngrok**- To map private IP address of the local Jenkins server to a public IP address so that we can use GitHub webhooks for continuous deployment

Project Steps:

1. Source Code , build tool [Maven] and testing[Junit]

Code is written in Java 11. Log4j is used to generate the log files and Junit is used for unit testing.



Calculator.java is the implementation of the calculator functionalities and uses log4j for generating log file.

```
72
73     public double factorial(double number1) {
74         logger.info("[FACTORIAL] - " + number1);
75         double result = fact(number1);
76         logger.info("[RESULT - FACTORIAL] - " + result);
77         return result;
78     }
79
80
81
82     public double squareRoot(double number1) {
83         logger.info("[SQ ROOT] - " + number1);
84         double result = Math.sqrt(number1);
85         logger.info("[RESULT - SQ ROOT] - " + result);
86         return result;
87     }
88
89
90     public double power(double number1, double number2) {
91         logger.info("[POWER - " + number1 + " RAISED TO] " + number2);
92         double result = Math.pow(number1, number2);
93         logger.info("[RESULT - POWER] - " + result);
94         return result;
95     }
96
```

CalculatorTest.java contains Junit testcases for the functionalities implemented in Calculator.java.

```
5 public class CalculatorTest {
6     private static final double DELTA = 1e-15;
7     Calculator calculator = new Calculator();
8
9     @Test
10    public void factorialTruePositive(){
11        assertEquals("Finding factorial of a number for True Positive", 720, calculator.fact(6), DELTA);
12        assertEquals("Finding factorial of a number for True Positive", 1, calculator.fact(1), DELTA);
13        assertEquals("Finding factorial of a number for True Positive", 6, calculator.fact(3), DELTA);
14        assertEquals("Finding factorial of a number for True Positive", 24, calculator.fact(4), DELTA);
15        assertEquals("Finding factorial of a number for True Positive", 1, calculator.fact(0), DELTA);
16    }
17
18    @Test
19    public void factorialFalsePositive(){
20        assertEquals("Finding factorial of a number for False Positive", 113, calculator.fact(5), DELTA);
21        assertEquals("Finding factorial of a number for False Positive", 18, calculator.fact(6), DELTA);
22        assertEquals("Finding factorial of a number for False Positive", 42, calculator.fact(4), DELTA);
23        assertEquals("Finding factorial of a number for False Positive", 9, calculator.fact(2), DELTA);
24        assertEquals("Finding factorial of a number for False Positive", 0, calculator.fact(0), DELTA);
25    }
26 }
```

```
17:53:12.178 [main] INFO    calculator.Calculator - [POWER - 5.0 RAISED TO] 3.0
17:53:12.179 [main] INFO    calculator.Calculator - [RESULT - POWER] - 125.0
17:53:12.181 [main] INFO    calculator.Calculator - [POWER - 2.0 RAISED TO] 3.0
17:53:12.182 [main] INFO    calculator.Calculator - [RESULT - POWER] - 8.0
17:53:12.183 [main] INFO    calculator.Calculator - [POWER - 1.0 RAISED TO] 3.0
17:53:12.184 [main] INFO    calculator.Calculator - [RESULT - POWER] - 1.0
17:53:12.185 [main] INFO    calculator.Calculator - [POWER - 3.0 RAISED TO] 4.0
17:53:12.186 [main] INFO    calculator.Calculator - [RESULT - POWER] - 81.0
17:53:12.188 [main] INFO    calculator.Calculator - [POWER - 4.0 RAISED TO] 3.0
17:53:12.188 [main] INFO    calculator.Calculator - [RESULT - POWER] - 64.0
17:53:12.189 [main] INFO    calculator.Calculator - [POWER - 5.0 RAISED TO] 2.0
17:53:12.189 [main] INFO    calculator.Calculator - [RESULT - POWER] - 25.0
17:53:12.192 [main] INFO    calculator.Calculator - [NATURAL LOG] - 2.4
17:53:12.192 [main] INFO    calculator.Calculator - [RESULT - NATURAL LOG] - 0.8754687373538999
17:53:12.193 [main] INFO    calculator.Calculator - [NATURAL LOG] - 2.1
17:53:12.194 [main] INFO    calculator.Calculator - [RESULT - NATURAL LOG] - 0.7419373447293773
Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.512 sec

Results :

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

Test cases ran during the build step

Calculator.log is the log file generated by the log4j when calculator functionalities are executed.

```
calculator.log
1 2023-03-15 15:55:35.597 [main] INFO    calculator.Calculator - [SQ ROOT] - 16.0
2 2023-03-15 15:55:35.601 [main] INFO    calculator.Calculator - [RESULT - SQ ROOT] - 4.0
3 2023-03-15 15:55:35.602 [main] INFO    calculator.Calculator - [SQ ROOT] - 1.0
4 2023-03-15 15:55:35.603 [main] INFO    calculator.Calculator - [RESULT - SQ ROOT] - 1.0
5 2023-03-15 15:55:35.603 [main] INFO    calculator.Calculator - [SQ ROOT] - 81.0
6 2023-03-15 15:55:35.604 [main] INFO    calculator.Calculator - [RESULT - SQ ROOT] - 9.0
7 2023-03-15 15:55:35.604 [main] INFO    calculator.Calculator - [SQ ROOT] - 36.0
8 2023-03-15 15:55:35.605 [main] INFO    calculator.Calculator - [RESULT - SQ ROOT] - 6.0
9 2023-03-15 15:55:35.606 [main] INFO    calculator.Calculator - [SQ ROOT] - 3.0
10 2023-03-15 15:55:35.606 [main] INFO    calculator.Calculator - [RESULT - SQ ROOT] - 1.7320508075688772
11 2023-03-15 15:55:35.607 [main] INFO    calculator.Calculator - [SQ ROOT] - 4.0
12 2023-03-15 15:55:35.607 [main] INFO    calculator.Calculator - [RESULT - SQ ROOT] - 2.0
13 2023-03-15 15:55:35.607 [main] INFO    calculator.Calculator - [SQ ROOT] - 81.0
14 2023-03-15 15:55:35.608 [main] INFO    calculator.Calculator - [RESULT - SQ ROOT] - 9.0
15 2023-03-15 15:55:35.608 [main] INFO    calculator.Calculator - [SQ ROOT] - 36.0
16 2023-03-15 15:55:35.609 [main] INFO    calculator.Calculator - [RESULT - SQ ROOT] - 6.0
17 2023-03-15 15:55:35.610 [main] INFO    calculator.Calculator - [NATURAL LOG] - 1.0
18 2023-03-15 15:55:35.610 [main] INFO    calculator.Calculator - [RESULT - NATURAL LOG] - 0.0
19 2023-03-15 15:55:35.612 [main] INFO    calculator.Calculator - [POWER - 2.0 RAISED TO] 3.0
20 2023-03-15 15:55:35.612 [main] INFO    calculator.Calculator - [RESULT - POWER] - 8.0
21 2023-03-15 15:55:35.613 [main] INFO    calculator.Calculator - [POWER - 3.0 RAISED TO] 3.0
22 2023-03-15 15:55:35.613 [main] INFO    calculator.Calculator - [RESULT - POWER] - 27.0
23 2023-03-15 15:55:35.614 [main] INFO    calculator.Calculator - [POWER - 4.0 RAISED TO] 3.0
```

Log file generated after successful build

Pom.xml should be configured to build the code and should add all the dependencies which are required, which are log4j and junit in our case. It is used by Maven to build the .JAR file .

```
pom.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6
7     <groupId>org.example</groupId>
8     <artifactId>calculatorDevOps</artifactId>
9     <version>1.0-SNAPSHOT</version>
10    <build>
11        <plugins>
12            <plugin>
13                <groupId>org.apache.maven.plugins</groupId>
14                <artifactId>maven-assembly-plugin</artifactId>
15                <executions>
16                    <execution>
17                        <phase>package</phase>
18                        <goals>
19                            <goal>single</goal>
20                        </goals>
21                        <configuration>
22                            <archive>
23                                <manifest>
24                                    <mainClass>calculator.Calculator</mainClass>
25                                </manifest>
26                            </archive>
27                            <descriptorRefs>
28                                <descriptorRef>jar-with-dependencies</descriptorRef>
29                            </descriptorRefs>
30                        </configuration>
31                    </execution>
32                </executions>
33            </plugin>
34        </plugins>
35    </build>
36    <dependencies>
37        <dependency>
38            <groupId>junit</groupId>
39            <artifactId>junit</artifactId>
40            <version>RELEASE</version>
41            <scope>test</scope>
42        </dependency>
43        <dependency>
44            <groupId>org.apache.logging.log4j</groupId>
45            <artifactId>log4j-api</artifactId>
46            <version>2.14.0</version>
47        </dependency>
48        <dependency>
49            <groupId>org.apache.logging.log4j</groupId>
50            <artifactId>log4j-core</artifactId>
51            <version>2.14.0</version>
52        </dependency>
53    </dependencies>
54
55    <properties>
56        <maven.compiler.source>8</maven.compiler.source>
57        <maven.compiler.target>8</maven.compiler.target>
58    </properties>
59
60 </project>
```


Log4j2.xml is added for logger configuration.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Configuration status="INFO">
3     <Appenders>
4         <Console name="ConsoleAppender" target="SYSTEM_OUT">
5             <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
6         </Console>
7         <File name="FileAppender" fileName="calculator.log" immediateFlush="false" append="true">
8             <PatternLayout pattern="%d{yyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
9         </File>
10    </Appenders>
11    <Loggers>
12        <Root level="debug">
13            <AppenderRef ref="ConsoleAppender" />
14            <AppenderRef ref="FileAppender" />
15        </Root>
16    </Loggers>
17 </Configuration>
18
```

\$sudo apt install maven

\$mvn clean install – will build the complete code after adding the required dependencies and all the test cases will be checked . New folder called Target will be created which contains the .JAR file.

```
INFO] Building jar: /home/aditya/IdeaProjects/SPE-Mini-Project/target/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar
INFO] META-INF/MANIFEST.MF already added, skipping
INFO] META-INF/ already added, skipping
INFO] org/ already added, skipping
INFO] org/apache/ already added, skipping
INFO] org/apache/logging/ already added, skipping
INFO] org/apache/logging/log4j/ already added, skipping
INFO] META-INF/versions/9/ already added, skipping
INFO] META-INF/versions/9/ already added, skipping
INFO] META-INF/versions/9/org/ already added, skipping
INFO] META-INF/versions/9/org/apache/ already added, skipping
INFO] META-INF/versions/9/org/apache/logging/ already added, skipping
INFO] META-INF/versions/9/org/apache/logging/log4j/ already added, skipping
INFO] META-INF/services/ already added, skipping
INFO] META-INF/maven/ already added, skipping
INFO] META-INF/maven/org.apache.logging.log4j/ already added, skipping
INFO] META-INF/DEPENDENCIES already added, skipping
INFO] META-INF/LICENSE already added, skipping
INFO] META-INF/NOTICE already added, skipping
INFO]
INFO] --- maven-install-plugin:2.4:install (default-install) @ calculatorDevOps ---
INFO] Installing /home/aditya/IdeaProjects/SPE-Mini-Project/target/calculatorDevOps-1.0-SNAPSHOT.jar to /home/aditya/.m2/repository/org/example/calculatorDevOps/1.0-SNAPSHOT/calculatorDevOps-1.0-SNAPSHOT.jar
INFO] Installing /home/aditya/IdeaProjects/SPE-Mini-Project/pom.xml to /home/aditya/.m2/repository/org/example/calculatorDevOps/1.0-SNAPSHOT/calculatorDevOps-1.0-SNAPSHOT.pom
INFO] Installing /home/aditya/IdeaProjects/SPE-Mini-Project/target/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar to /home/aditya/.m2/repository/org/example/calculatorDevOps/1.0-SNAPSHOT/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar
INFO] BUILD SUCCESS
INFO]
INFO] Total time: 9.494 s
INFO] Finished at: 2023-03-21T17:53:15+05:30
```

- **cd target** – to go into the target folder containing the JAR file.
- **Java -jar calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar**
To run the JAR file

```

aditya@aditya-Aspire-A515-51G:~/IdeaProjects/SPE-Mini-Project$ cd target
aditya@aditya-Aspire-A515-51G:~/IdeaProjects/SPE-Mini-Project/target$ java -jar calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar
Scientific Calculator using DevOps.
Choose operation:
1. Factorial
2. Square root
3. Power
4. Natural Logarithm
5. Exit
Enter your choice: 1
Enter a number : 5
17:57:42.688 [main] INFO  calculator.Calculator - [FACTORIAL] - 5.0
17:57:42.698 [main] INFO  calculator.Calculator - [RESULT - FACTORIAL] - 120.0
Factorial of 5.0 is : 120.0

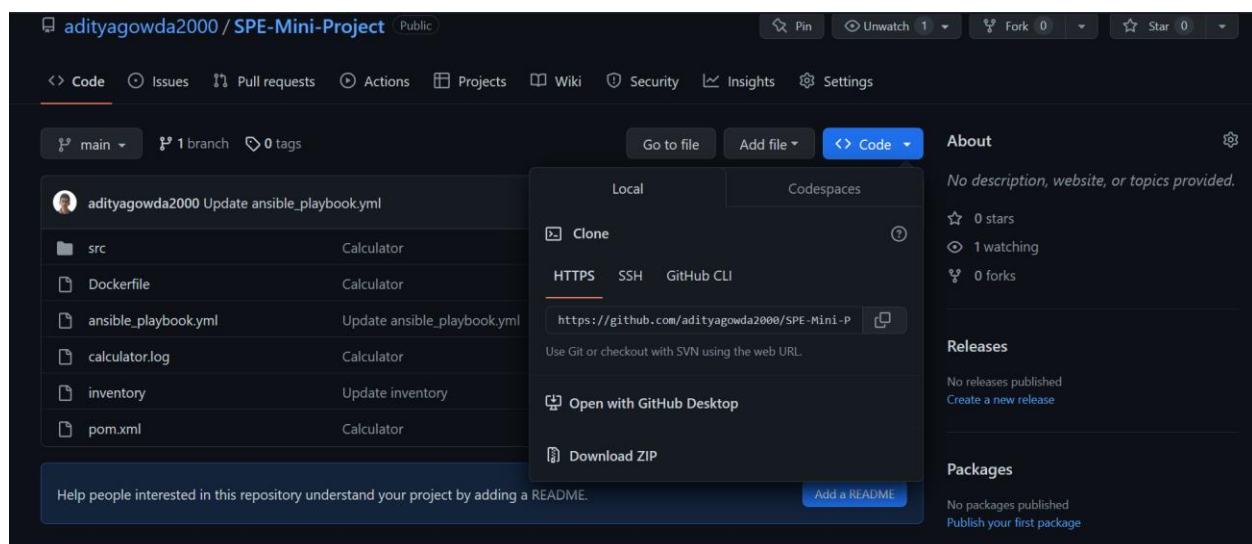
Scientific Calculator using DevOps.
Choose operation:
1. Factorial
2. Square root
3. Power
4. Natural Logarithm
5. Exit
Enter your choice: 3

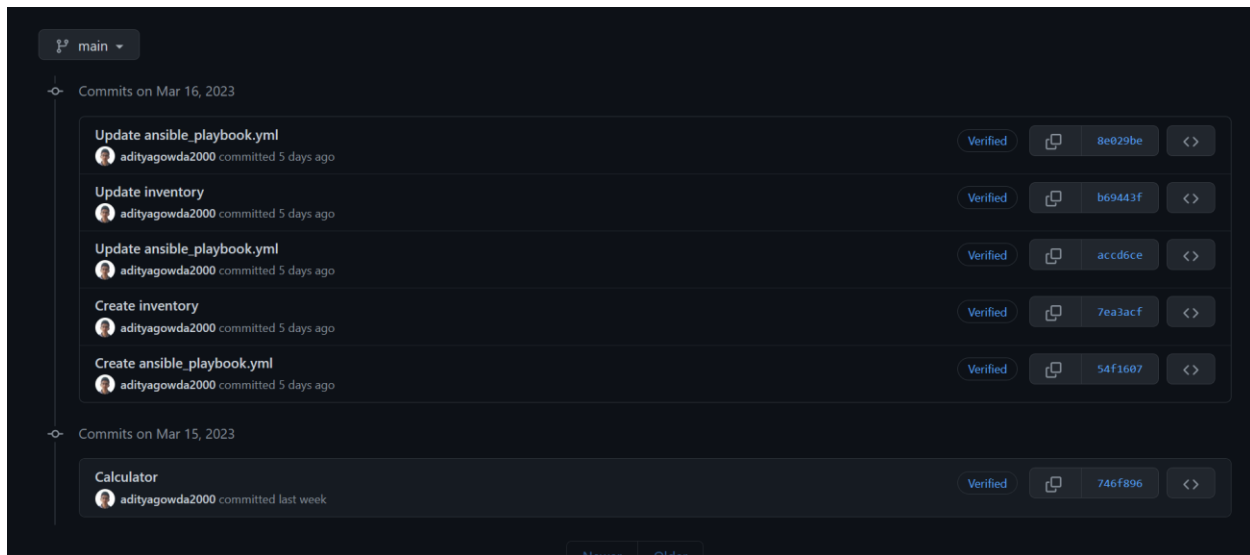
```

Sample output

2. Version Control with GitHub

- Create new repository on GitHub and copy the repository URL
- Go to the directory containing the source code in the local machine
- \$git init
- \$git remote add origin <github repo URL>
- \$git add .
- \$git commit -m "Commit Message"
- \$git push origin master





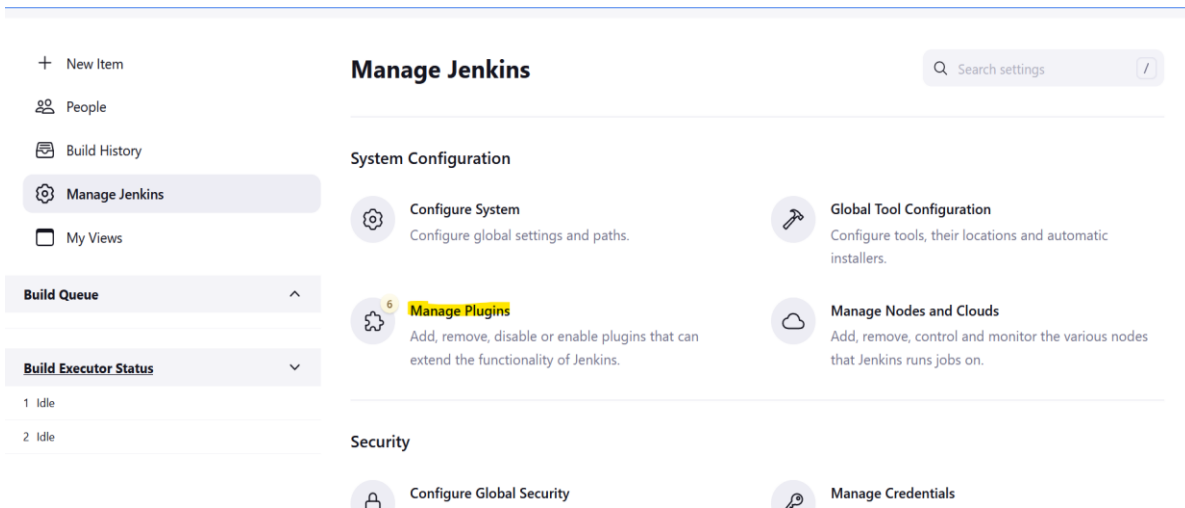
3. Jenkins-

Jenkins installation and setup- Install Jenkins on your computer to use it as a Jenkins server.

- `$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -`
- `$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable/etc/apt/sources.list.d/jenkins.list`
- `$ sudo apt update`
- `$ sudo apt install Jenkins`
- `$ sudo cat/var/lib/Jenkins/secrets/initialAdminPassword`
=>to copy initial password for Jenkins
- Go to localhost/8080 for the Jenkins page
- Enter the initial Admin Password and then create new user profile.

Plugins- following plugins must be installed

- Git
- Maven
- Ansible
- Docker



Ngrok setup

Used to convert the private IP address of the Jenkins server into public address so we can use webhooks to automatically rebuild the project when new changes are pushed into the GitHub repository.

- Install ngrok on you system after creating a free account

Ngrok Installation

1. Sign up in <https://ngrok.com/>
2. Download ngrok from: <https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz>
3. Then extract ngrok from the terminal: `$sudo tar xvfz ~/Downloads/ngrok-stable-linux-amd64.tgz -C /usr/local/bin`
4. Copy Authtoken from: <https://dashboard.ngrok.com/get-started/your-authtoken>
5. Add Authtoken: `$ngrok authtoken <token>`
6. Execute `$ngrok http 8080`; copy the public ip address for your local host.

- \$ngrok 8080 => to expose port no 8080 to a public IP

```
ngrok
Add OAuth and webhook security to your ngrok (its free!): https://ngrok.com/free

Session Status      online
Account             aditya@manne@gmail.com (Plan: Free)
Version             3.2.1
Region              India (ln)
Latency             45ms
Web Interface        http://127.0.0.1:4840
Forwarding           https://ead4-183-156-19-229.in.ngrok.io -> http://localhost:8080

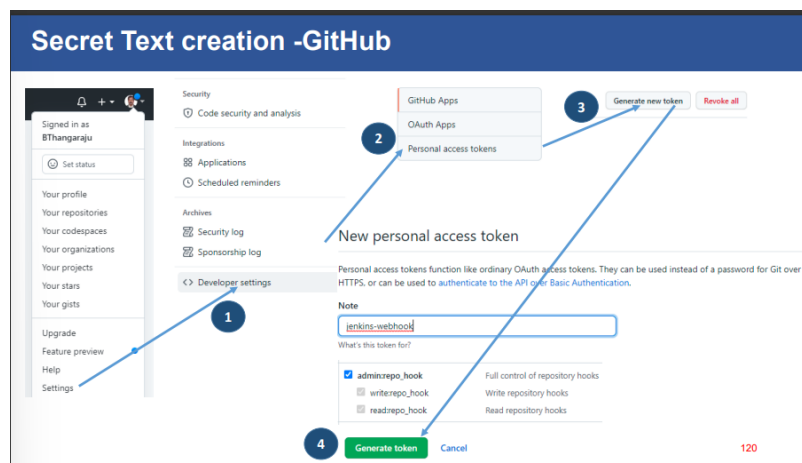
Connections         ttl    opn    rt1    rt5    p50    p90
                   86     0      0.00   0.03   0.31   30.85

HTTP Requests
-----
POST /manage/descriptorByName/hudson.plugins.git.GitTool/checkHome 200 OK
POST /manage/descriptorByName/hudson.plugins.git.GitTool/checkName 200 OK
GET /manage/configureTools/ 200 OK
GET /manage/configureTools/ 302 Found
GET /administrativeMonitor/jenkins.diagnostics.URICheckEncodingMonitor/checkURICheckEncoding 200 OK
GET /manage/ 200 OK
POST /manage/descriptorByName/org.jenkinsci.plugins.github.config.GitHubServerConfig/fillCredential 200 OK
POST /manage/descriptorByName/org.jenkinsci.plugins.github.config.GitHubServerConfig/fillCredential 200 OK
POST /manage/descriptorByName/hudson.tasks.shell/checkShell 200 OK
POST /manage/descriptorByName/hudson.plugins.email-ext.MailerAccount/fillCredentialsIdItems 200 OK
```

- Make note of the ngrok ip address

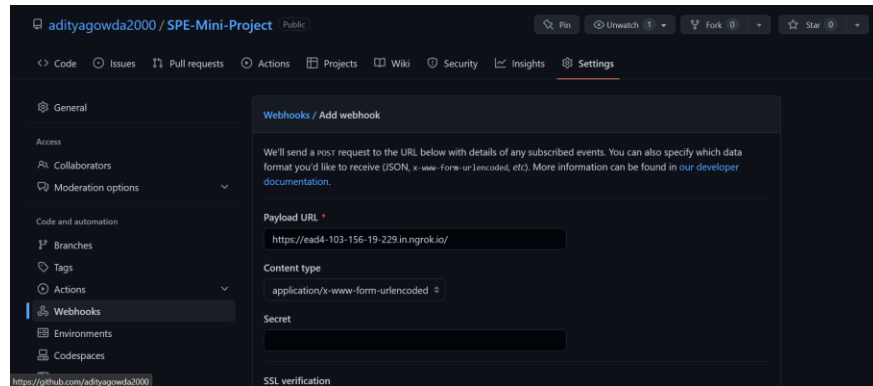
Secret Text Creation – GitHub

Create a secret text/personal access token and make note of it



GitHub Webhook setup


- In the GitHub repository's setting add a webhook with payload as the Jenkins server's public IP address[ngrok address] and also add the secret access token which was copied in the above step.




Jenkins server configuration

- In configure system add the ngrok address as the Jenkins URL

Jenkins Location

Jenkins URL 


System Admin e-mail address 

- Copy personal access token from the webhook configuration


Jenkins Credentials Provider: Jenkins


Add Credentials


Domain

Global credentials (unrestricted) 

Kind

Secret text 

Scope 

Global (Jenkins, nodes, items, all child items, etc) 

Secret

Create new Pipeline project with GitHub hook trigger for GITScm polling as a build trigger.

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☐ Build periodically

☒ GitHub hook trigger for GITScm polling

☐ Poll SCM

Pipeline Script:

```
1 pipeline {
2   environment{
3     dockerImage = " "
4   }
5
6   agent any
7
8   stages {
9     stage('Git pull') {
10      steps {
11        git branch: 'main', url: 'https://github.com/adityagowda2000/SPE-Mini-Project'
12      }
13    }
14    stage('Maven Build') {
15      steps {
16        sh 'mvn clean install'
17      }
18    }
19    stage('Build Docker Image'){
20      steps{
21        script{
22          echo "Build Docker Image"
23          dockerImage = docker.build("adityagowda2000/spe-mini-project")
24        }
25      }
26    }
27    stage('Push Docker Image'){
28      steps{
29        script{
30          echo "Push to Docker Hub"
31          docker.withRegistry('', "Docker-Hub"){
32            dockerImage.push('latest')
33          }
34        }
35      }
36    }
37    stage('Deploy with Ansible'){
38      steps{
39        script{
40          echo "Deploying image to Ansible"
41          ansiblePlaybook colored: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inventory', playbook: 'ansible_playbook.yml'
42        }
43      }
44    }
45  }
46 }
47
48 }
```

Stages:

- **Git pull** – to pull the code from the github repository

```
stage('Git pull') {  
    steps {  
        git branch: 'main', url:  
        'https://github.com/adityagowda2000/SPE-Mini-Project'  
    }  
}
```

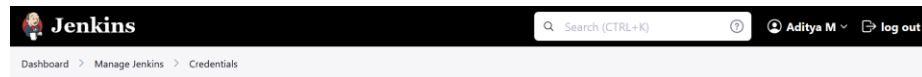
- **Maven Build**- To build the cloned project and generate jar file with all the dependencies.

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

- **Build Docker Image**- To create a docker image using the Dockerfile present in the source code. [More explanation in Containerization step].

```
stage('Build Docker Image'){  
    steps{  
        script{  
            echo "Build Docker Image"  
            dockerImage = docker.build("adityagowda2000/spe-  
mini-project")  
        }  
    }  
}
```

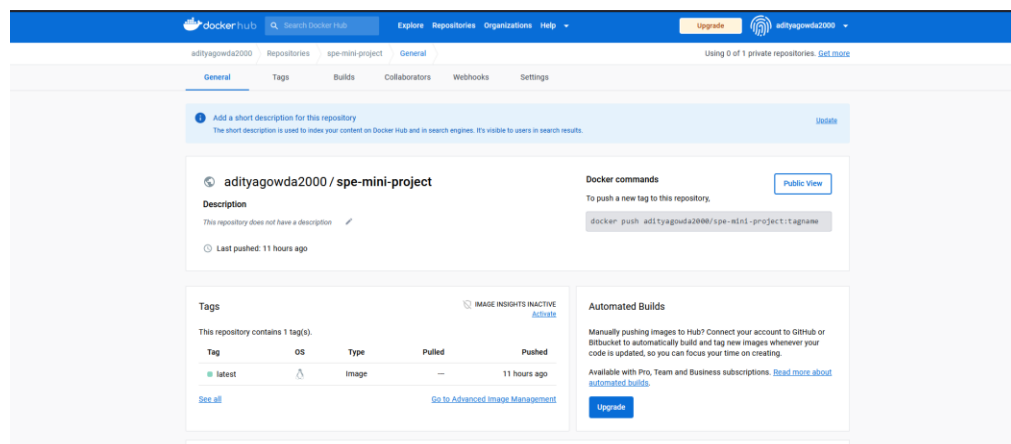

- **Push Docker Image-** Push the created image to Docker hub. Note credentials for your docker hub account should be added in Jenkins server.



Credentials

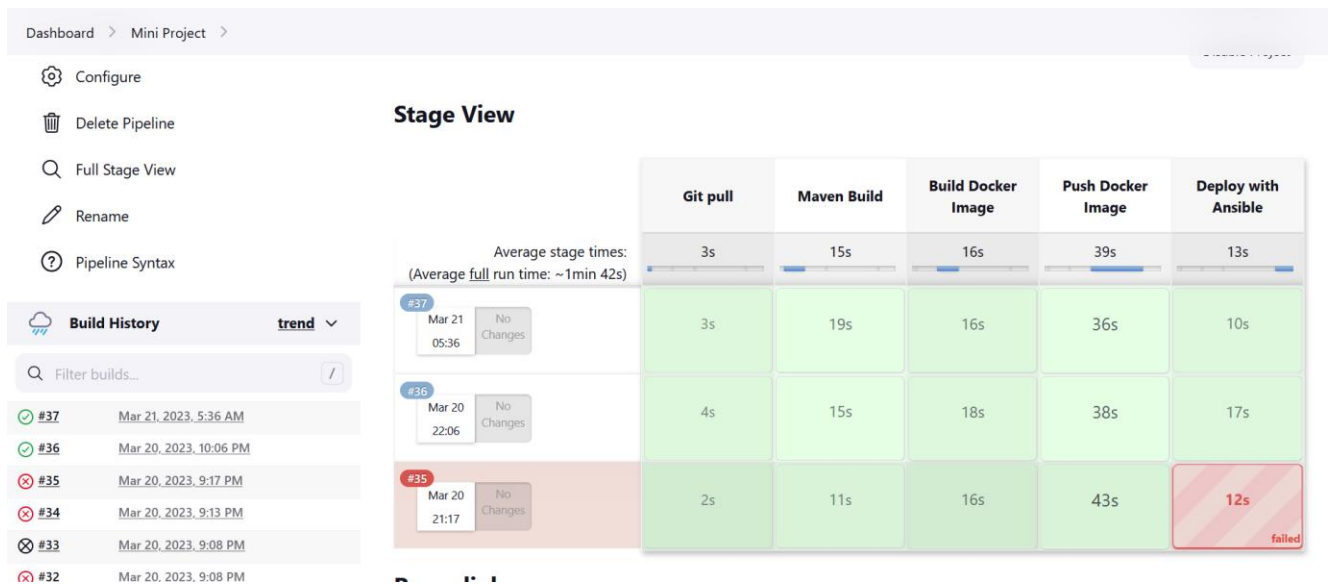
T	P	Store i	Domain	ID	Name
		System	(global)	Docker-Hub	Docker-Hub

```
stage('Push Docker Image'){
    steps{
        script{
            echo "Push to Docker Hub"
            docker.withRegistry('', "Docker-Hub"){
                dockerImage.push('latest')
            }
        }
    }
}
```



- **Deploy with Ansible-** Pulling the docker image and then deploying it onto the deployment server using `ansible_playbook.yml` present in the source code.[More explanation in deployment steps].

```
stage('Deploy with Ansible'){
    steps{
        script{
            echo "Deploying image to Ansible"
            ansiblePlaybook colored: true,
disableHostKeyChecking: true, installation:
'Ansible', inventory: 'inventory', playbook:
'ansible_playbook.yml'
        }
    }
}
```

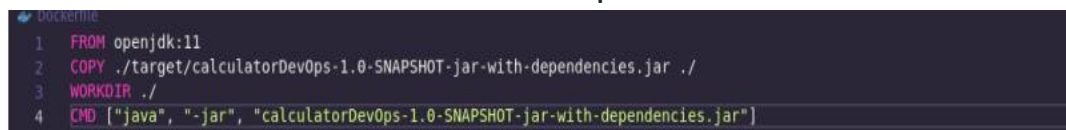


4. Containerization using Dockerfile:

- Install Docker on your computer which is acting as the Jenkins server : [How To Install and Use Docker on Ubuntu 20.04 | DigitalOcean](#)
- Give permission for docker to be executed without sudo command.
 - Solution 1:
`$ sudo usermode -a -G docker $USER`
 - Solution 2:
`$_sudo chmod 777 /var/run/docker.sock`

*[Caution: Running **sudo chmod 777 /var/run/docker.sock** will solve your problem but it will open the docker socket for everyone which is a security vulnerability]*

- We will create a container using JDK 11 as base image and on top of it the generated JAR file after build will be added. To do this we will use run the Dockerfile present in the source code.



```
1 FROM openjdk:11
2 COPY ./target/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar ./
3 WORKDIR ./
4 CMD ["java", "-jar", "calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

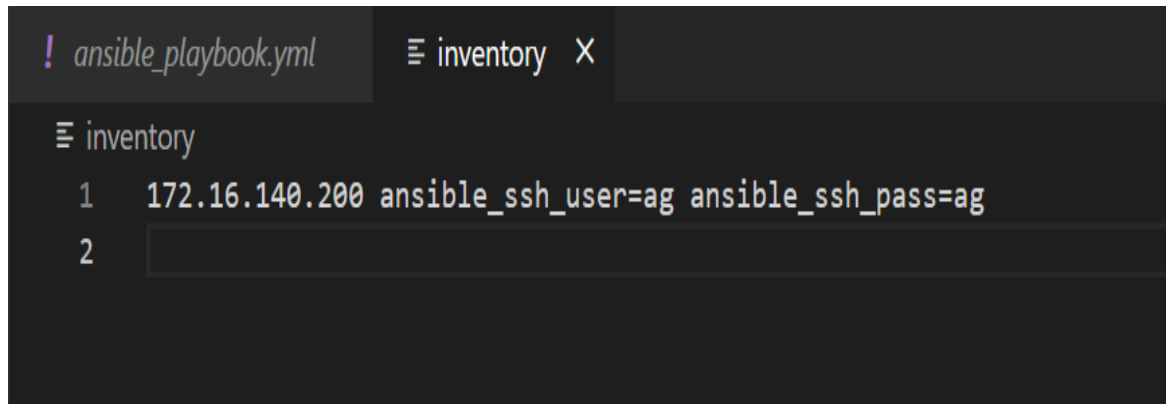
- FROM: It imports the base image openjdk11 in order to create a new image.
- COPY: It can copy a file(should be in the same directory as the Dockerfile) into the image in its root directory.
- WORKDIR: it changes the current working directory.
- CMD: runs the command inside the image when the image is launched.

5. Deployment with Ansible Playbook

- **Install ansible on the jenkins server:**
 - `$ sudo apt install ansible`

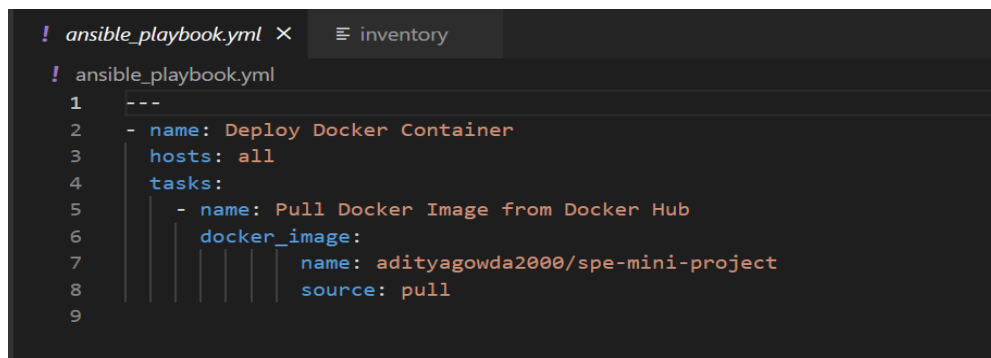
Note: We are going to create a new user account and consider it as the client machine.

- **Create a new user on your computer which acts as deployment server:**
 - `$ adduser user_name`
 - `$ Usermode -aG sudo user_name =>` to give sudo permissions to this user account
 - `$ su user_name =>` to switch back to `user_name` which act's as deployment server
- **Configure all the clients to which the configuration has to be pushed by the Jenkins server**
 - Add the client servers in the inventory file.
 - append the client hosts ip address along with user name and password in hosts inventory file.
 - `[machine]`
`172.16.140.200 ansible_ssh_user=ag ansible_ssh_pass=ag`



```
! ansible_playbook.yml  inventory X
inventory
1 172.16.140.200 ansible_ssh_user=ag ansible_ssh_pass=ag
2
```

- ansible_playbook.yml should be configured to pull the docker image and deploy it.



```
! ansible_playbook.yml X  inventory
! ansible_playbook.yml
1 ---
2 - name: Deploy Docker Container
3   hosts: all
4   tasks:
5     - name: Pull Docker Image from Docker Hub
6       docker_image:
7         name: adityagowda2000/spe-mini-project
8         source: pull
9
```

Now the docker image will be available in deployment_server, we can run the image as container using the command –

- \$ docker run -it adityagowda2000/spe-mini-project

6. Monitoring Log files using ELK stack

- Clone <https://github.com/deviantony/docker-elk>
- Unzip it and open terminal in the directory
- \$docker-compose up -d => to create the image
- Launch the elastic server by running the docker image:
 - \$docker run -it docker-elk-main_kibana
- Now ELK sack is live in localhost/5601
- Login with “elastic” as user name and “changeme” as password.
- Upload the log file generated manually.

