Calculator with DevOps

Submitted by Harsh Shah - MT2021050

Links

Github Repo:-

https://github.com/harshshah8/Spe_mini_project

Docker Repo:-

Docker Hub

https://hub.docker.com/repository/docker/shahharsh8/speminiproject

Problem Statement

creating a calculator program with operations such as

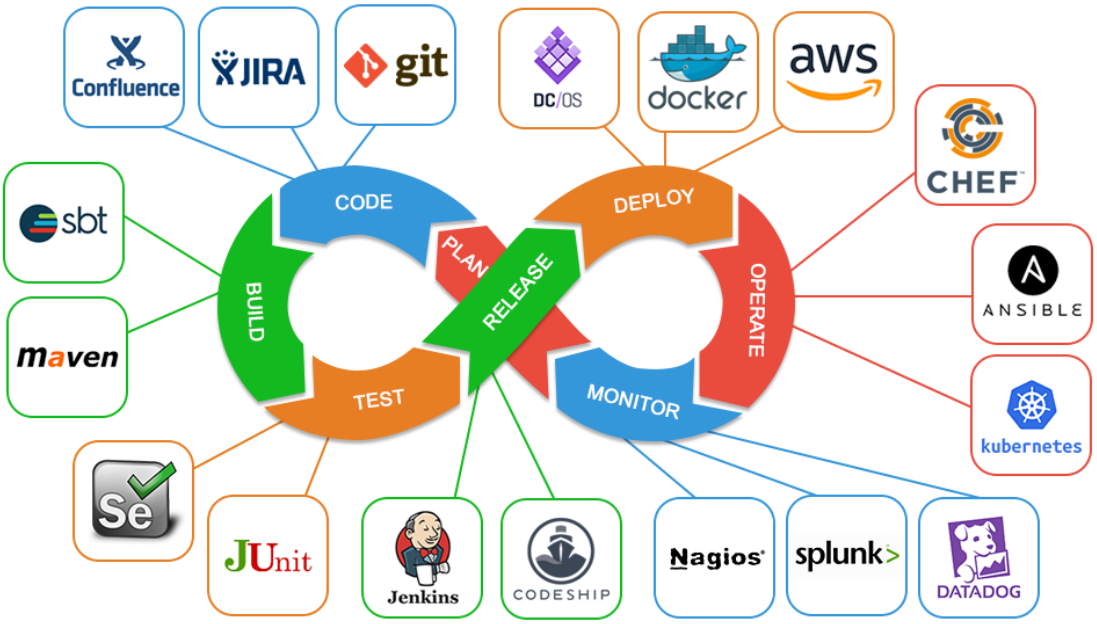
- Square root function - **√x**

- ▼ Factorial function - x!
- ▼ Natural logarithm (base e) - ln(x)
- ▼ Power function - x^b

The main objective of the project is to learn the DevOps concept of CI/CD which is achieved by creating a Jenkins pipeline.

DevOps

- What is DevOps?
    - DevOps is a set of cultural concepts, practices, and technologies that improves an organization's capacity to produce high-velocity applications and services, allowing it to evolve and improve products at a faster rate than traditional software development and infrastructure management methods. Organizations can better service their clients and compete in the market because of this quickness.
    - Development and operations teams are no longer "silos" in a DevOps architecture. These two teams are sometimes combined into a single team where the engineers work across the whole application lifecycle, from development and testing to deployment and operations, and develop a diverse set of abilities that aren't limited to a particular role.
    - Quality assurance and security teams may become more closely linked with development and operations, as well as throughout the application lifecycle, in some DevOps models. When everyone in a DevOps team is focused on security, this is referred to as DevSecOps.
    - These groups employ best practices to automate procedures that were previously manual and slow. They employ a technological stack and infrastructure that allows them to swiftly and reliably operate and evolve apps. These tools also assist engineers in independently completing tasks (such as deploying code or supplying infrastructure) that would ordinarily require assistance from other teams, hence increasing a team's velocity.



- Why DevOps?
    - From shopping to entertainment to banking, software and the Internet have changed the world and its sectors. Software is no longer just a means of supporting a business; it is now an intrinsic part of every aspect of it. Companies communicate with their customers using software that is supplied as online services or applications and may be used on a variety of devices. They also leverage software to revolutionize every component of the value chain, including logistics, communications, and operations, to improve operational efficiencies. Companies in today's environment must adapt how they produce and distribute software in the same manner that physical goods companies transformed how they design, build, and deliver things utilizing industrial automation throughout the twentieth century.

Tools Used

1. **Maven:** It's a Java-based application development tool that lets us add dependencies and build a jar file (a snapshot of our project) that can be run on any machine.
2. **GitHub:** Helps in automation through Jenkin Integration

3. Jenkins: It is used for DevOps(for Continuous Integration and Continuous Deployment portion)
4. **Docker**: It is used to make images through containerization.
5. **Ansible:** It automates and simplifies repetitive, complex, and tedious operations. It saves a lot of time when we install packages or configure large numbers of servers.
6. **WebHooks:** To automate the build process whenever the developer commits the code to GitHub.
7. **Ngrok:** To convert the private IP address of the local machine to a public IP address to perform webhook.

Steps

1. Install Java 11 and IntelliJ
2. Write your code in Maven
3. Push your code into Github
4. Create a repository in DockerHub for your project
5. Write Pipeline Script in Jenkins
    a. Git Pull
    b. Maven build
    c. Docker Image creation
    d. Pushing Image to Docker Hub
    e. Ansible Deploy

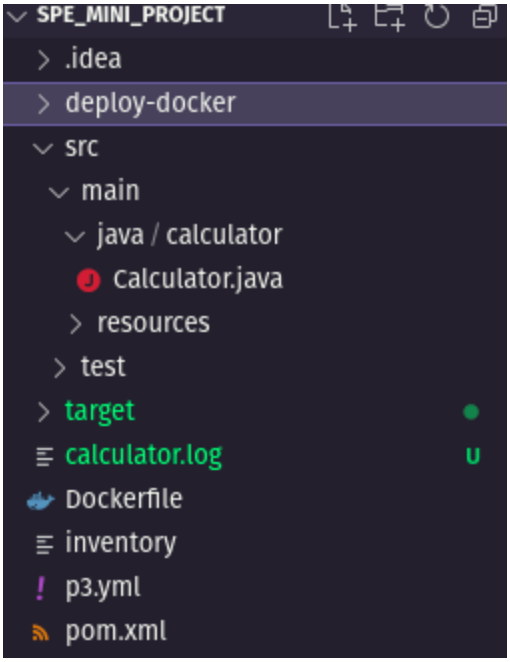
6. Build the project.

7. Pull the image into the remote server.

8. Run the image

Development, Software Build, and Test

- The code is developed in Java 11 and the IntelliJ IDE is utilized as the development environment. Log4j is used to keep track of logs for monitoring, and JUnit is used for unit testing.



- **Calculator.java:** It contains the main code of the project. which contains the following functions.
  1. Natural Log
  2. Square Root
  3. Factorial
  4. Power
- **CalculatorTest.java:** Contains true and false positive test cases used to test the code when we build the project. It is performed using JUnit.
- **Output:**

```
Workspaces   Applications                    Apr 17  19:45                                   10 %
[INFO] META-INF/DEPENDENCIES already added, skipping
[INFO] META-INF/LICENSE already added, skipping
[INFO] META-INF/NOTICE already added, skipping
[INFO] Building jar: /home/harsh/Downloads/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/ 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/harsh/Downloads/Spe_mini_project/target/calculatorDevOps-1.0-SNAPSHOT.jar to /home/harsh/.m2/repository/org/exam
ple/calculatorDevOps/1.0-SNAPSHOT/calculatorDevOps-1.0-SNAPSHOT.jar
[INFO] Installing /home/harsh/Downloads/Spe_mini_project/pom.xml to /home/harsh/.m2/repository/org/example/calculatorDevOps/1.0-SNAPSHOT
/calculatorDevOps-1.0-SNAPSHOT.pom
[INFO] Installing /home/harsh/Downloads/Spe_mini_project/target/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar to /home/harsh/.
m2/repository/org/example/calculatorDevOps/1.0-SNAPSHOT/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.179 s
[INFO] Finished at: 2022-04-17T19:45:28+05:30
[INFO] ------------------------------------------------------------------------

Spe_mini_project on 鹏master [?] is 📦 1.0-SNAPSHOT via ☕ v16.0.2 took 8s
❯
```

- **Test-Cases:** For every functionality, two types of test cases are used, one is a True Positive and the other is a False Positive.

```java
public class CalculatorTest {
    private static final double DELTA = 1e-15;
    Calculator calculator = new Calculator();

    @Test
    public void factorialTruePositive(){
        assertEquals("Finding factorial of a number for True Positive", 120, calculator.fact(5), DELTA);
        assertEquals("Finding factorial of a number for True Positive", 24, calculator.fact(4), DELTA);
        assertEquals("Finding factorial of a number for True Positive", 6, calculator.fact(3), DELTA);
        assertEquals("Finding factorial of a number for True Positive", 1, calculator.fact(0), DELTA);
    }

    @Test
    public void factorialFalsePositive(){
        assertNotEquals("Finding factorial of a number for False Positive", 130, calculator.fact(5), DELTA);
        assertNotEquals("Finding factorial of a number for False Positive", 28, calculator.fact(4), DELTA);
        assertNotEquals("Finding factorial of a number for False Positive", 9, calculator.fact(3), DELTA);
        assertNotEquals("Finding factorial of a number for False Positive", 0, calculator.fact(0), DELTA);
    }

    @Test
```

- **Project Dependencies:** To use JUnit and log4j, we need to add certain jar files in the pom.xml file. So Maven will add those dependencies.

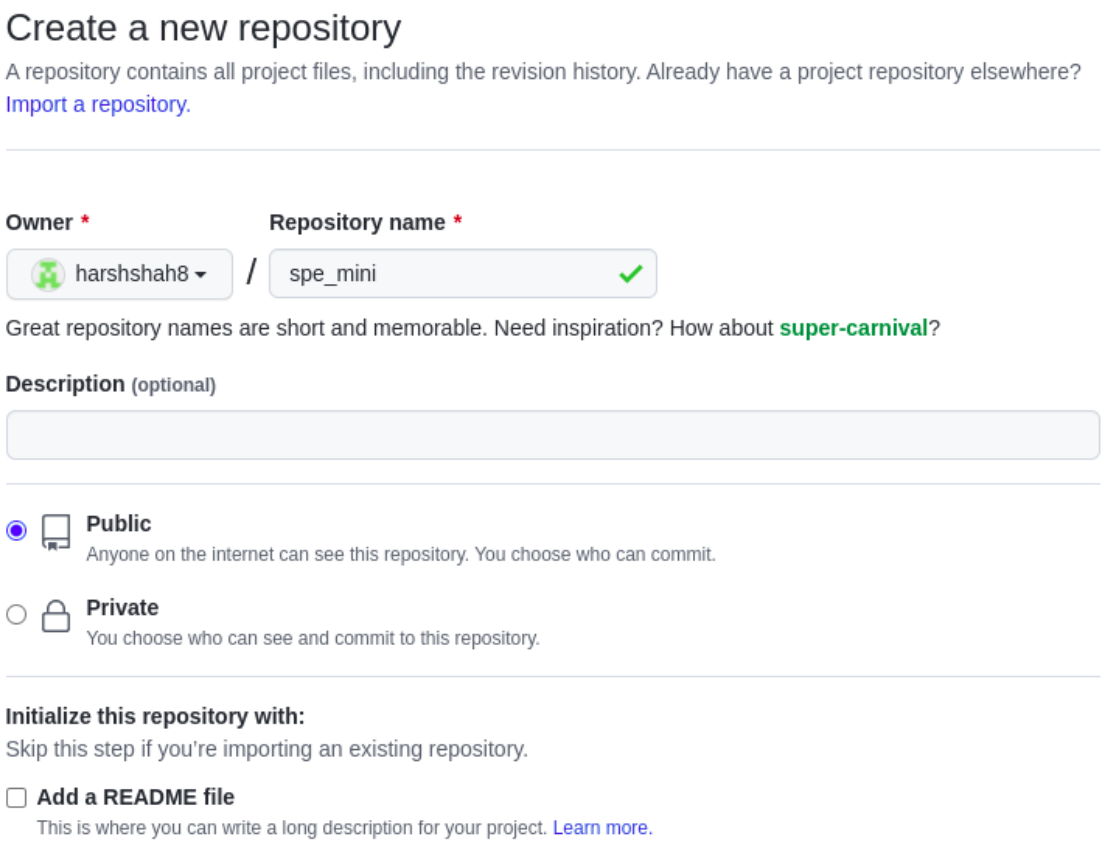
```xml
    <groupId>org.example</groupId>
    <artifactId>calculatorDevOps</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-assembly-plugin</artifactId>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                        <configuration>
                            <archive>
                                <manifest>
                                    <mainClass>calculator.Calculator</mainClass>
                                </manifest>
                            </archive>
                            <descriptorRefs>
                                <descriptorRef>jar-with-dependencies</descriptorRef>
                            </descriptorRefs>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

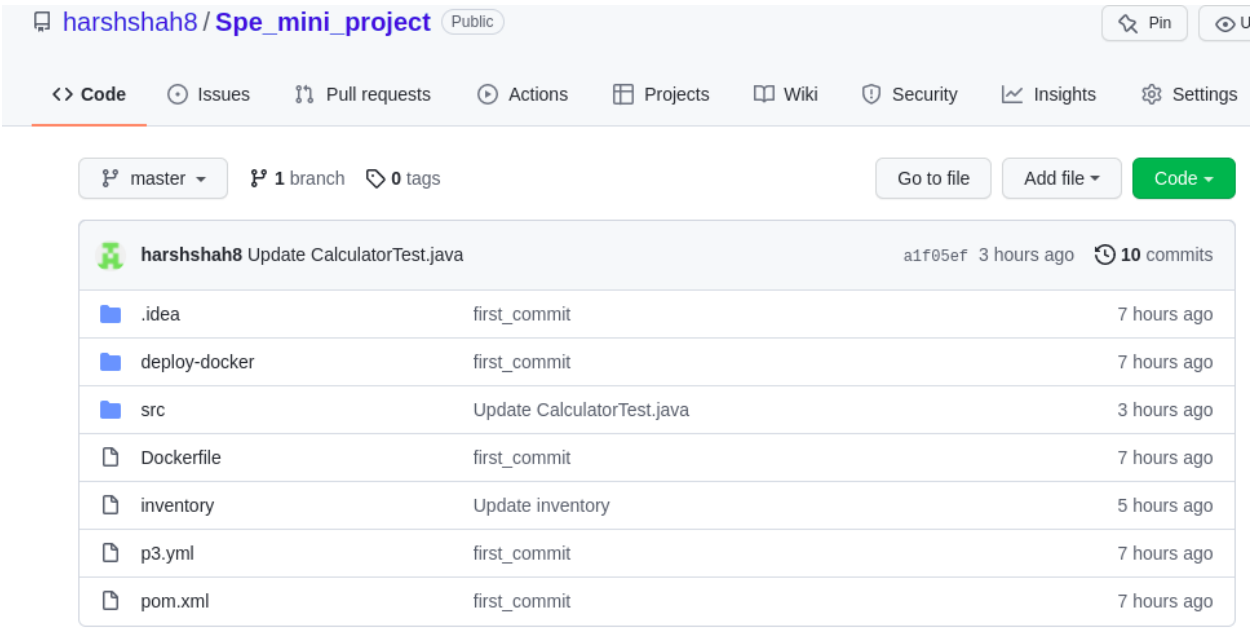
- So now by doing **$ mvn clean install** the complete code will be built and all test cases will be checked and in the current folder, a new folder will get created named “target” in which **.jar**
  will be generated.

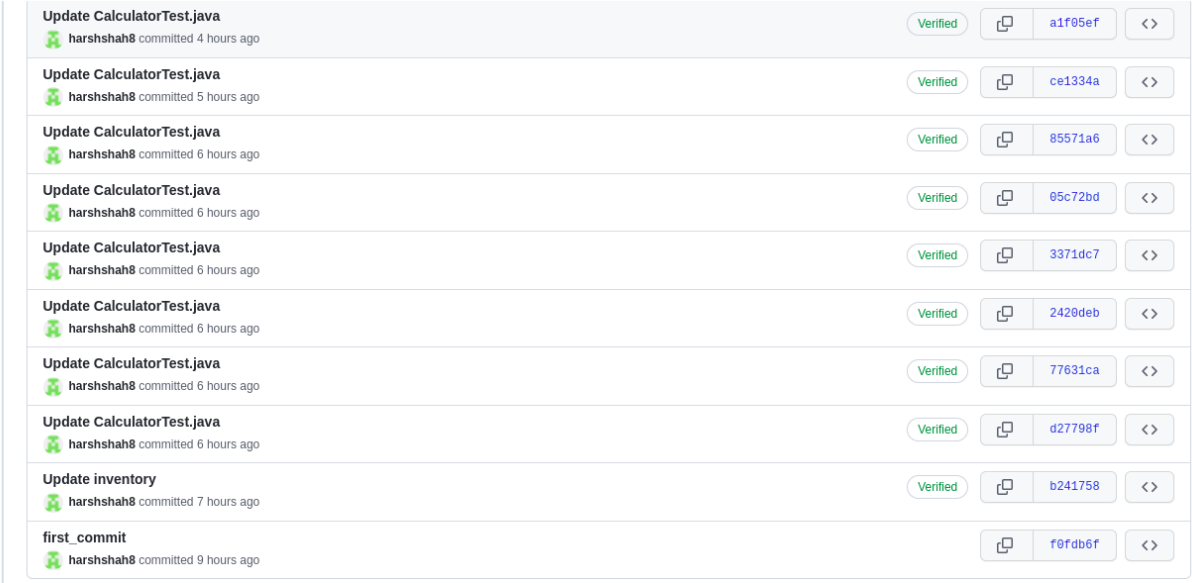
Source Code Management - GitHub

- The basic goal of SCM is to keep software in its current state (known as the "baseline") while allowing developers to work on new versions for new features or repairs. This is accomplished with the help of GitHub.
- Create a new repository at https://github.com/ to get started. We can build a new repository by providing it with a unique name connected with the user. The SCM, which will be connected to Jenkins as an input, will manage our code.
  - **Steps:**
    1. Create a public repository.
    2. $ git init.
    3. $ git add. .

4. $ git remote add origin <github repo URL>.
5. $ git commit -m “Message here”.
6. $ git push origin master.




harshshah8 / Spe_mini_project Public
Pin
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
1 branch
0 tags
Go to file
Add file
Code
harshshah8 Update CalculatorTest.java
a1f05ef 3 hours ago
10 commits
.idea first_commit 7 hours ago
deploy-docker first_commit 7 hours ago
src Update CalculatorTest.java 3 hours ago
Dockerfile first_commit 7 hours ago
inventory Update inventory 5 hours ago
p3.yml first_commit 7 hours ago
pom.xml first_commit 7 hours ago

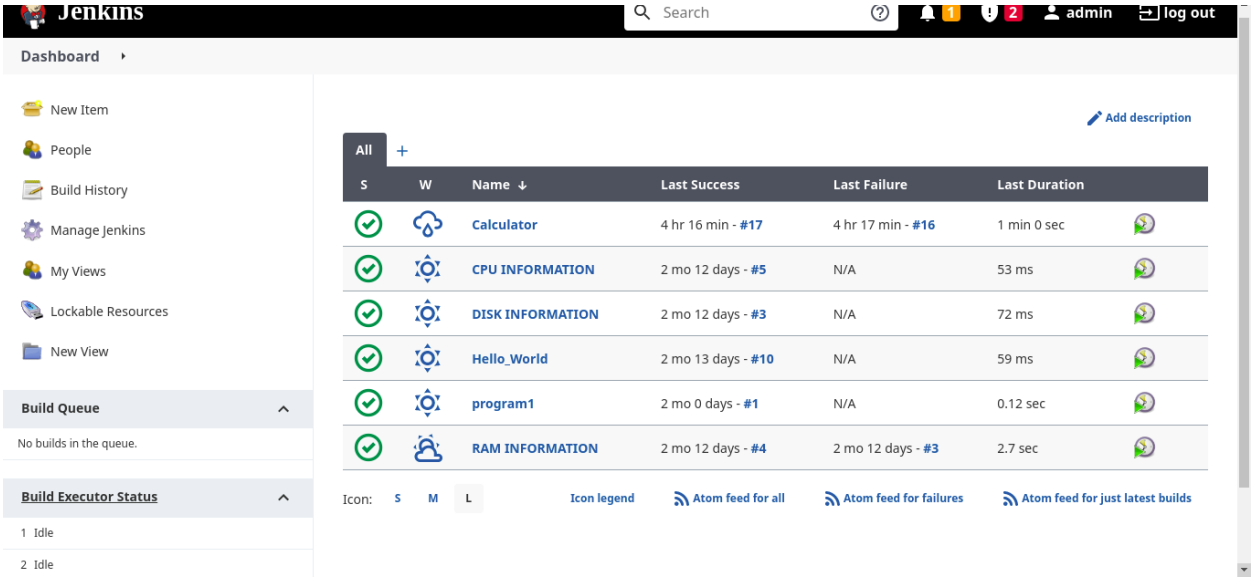

Update CalculatorTest.java
harshshah8 committed 4 hours ago
Verified
a1f05ef
Update CalculatorTest.java
harshshah8 committed 5 hours ago
Verified
ce1334a
Update CalculatorTest.java
harshshah8 committed 6 hours ago
Verified
85571a6
Update CalculatorTest.java
harshshah8 committed 6 hours ago
Verified
05c72bd
Update CalculatorTest.java
harshshah8 committed 6 hours ago
Verified
3371dc7
Update CalculatorTest.java
harshshah8 committed 6 hours ago
Verified
2420deb
Update CalculatorTest.java
harshshah8 committed 6 hours ago
Verified
77631ca
Update CalculatorTest.java
harshshah8 committed 6 hours ago
Verified
d27798f
Update inventory
harshshah8 committed 7 hours ago
Verified
b241758
first_commit
harshshah8 committed 9 hours ago
f0fdb6f

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 utilised in this project to handle until delivery, i.e. continuous delivery. http://localhost:8080 is the URL for the Jenkins service. To go to it, open a web browser and type this URL into the address bar.

- **Setup Steps:**

  1. wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |sudo apt-key add –
  2. sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable /etc/apt/sources.list.d/jenkins.list'
  3. sudo apt update
  4. sudo apt install Jenkins



- **Pre-requisites for Jenkins:**

  ▼ We will need variety of plugins to utilising Jenkins Pipeline.

  1. **Install following things from Plugin Manager**

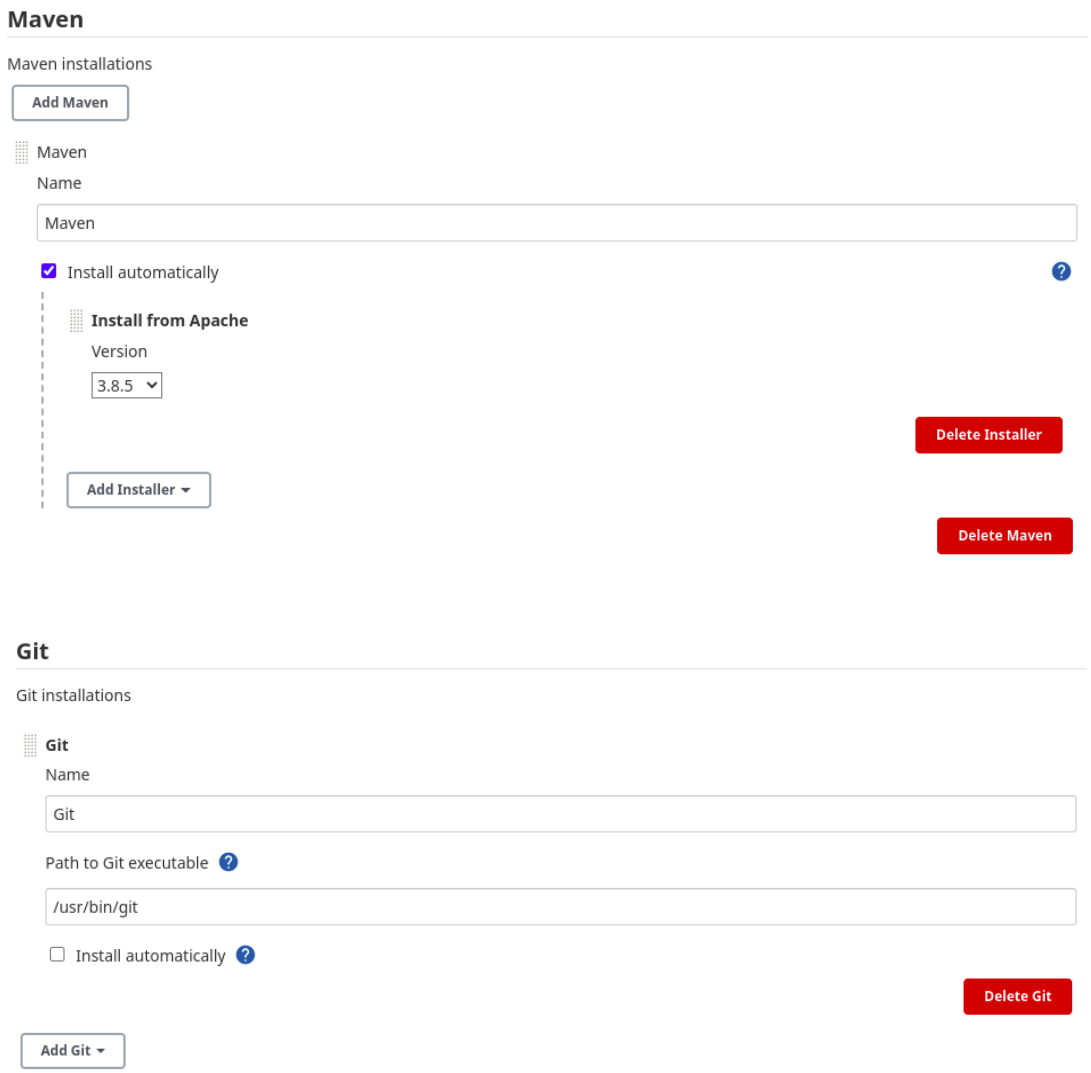
a. Maven

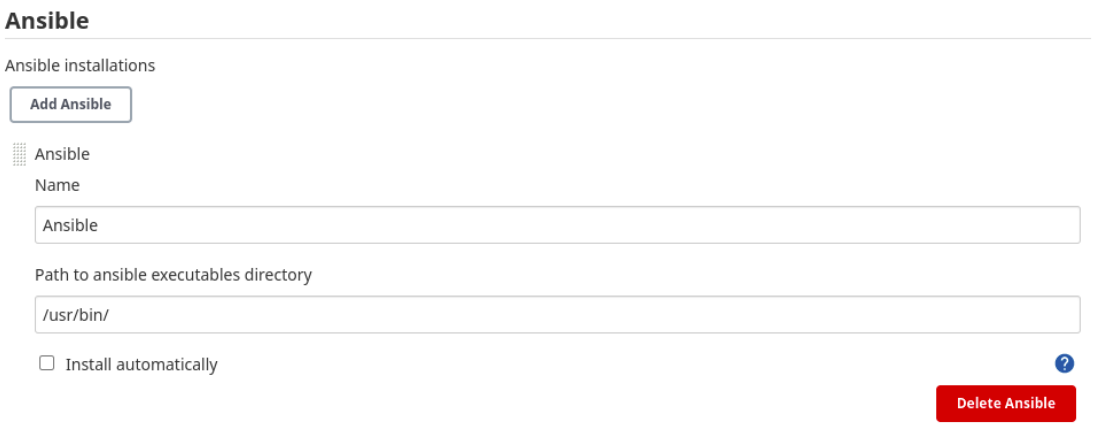
b. Git

c. Ansible

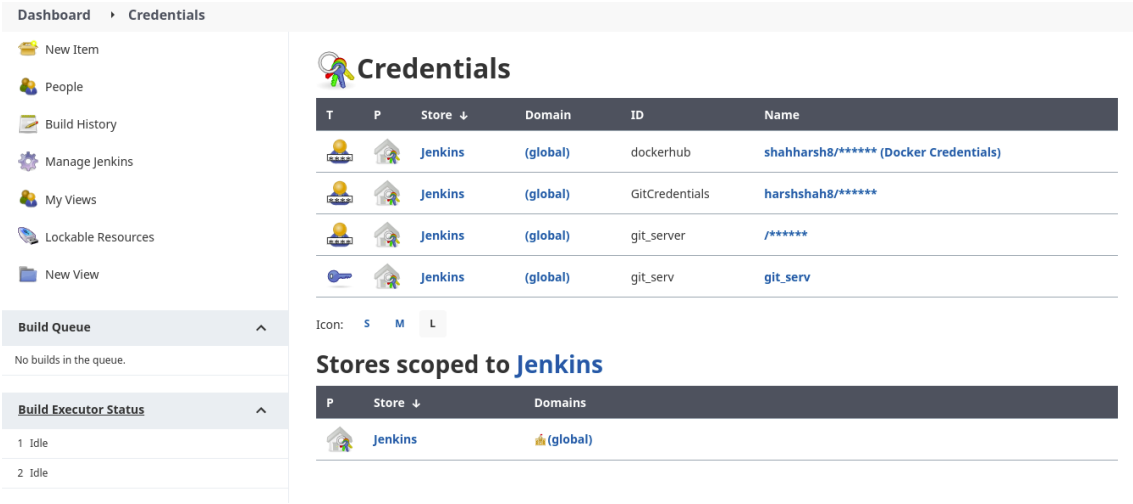
d. Docker

2. **Manage Jenkins -> Global Tool Configuration**

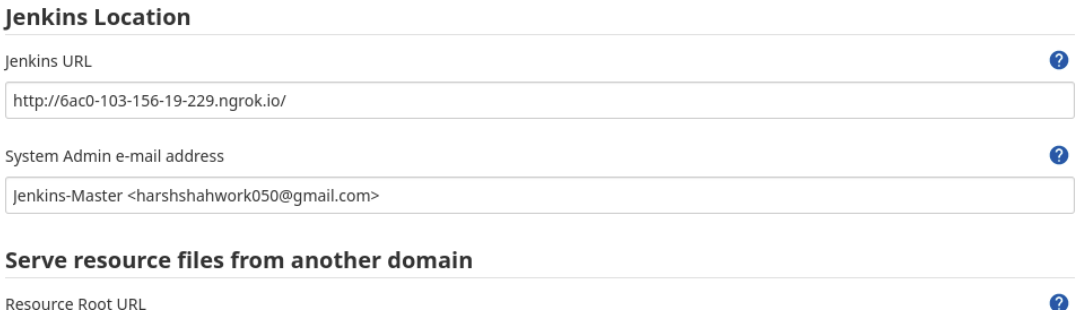


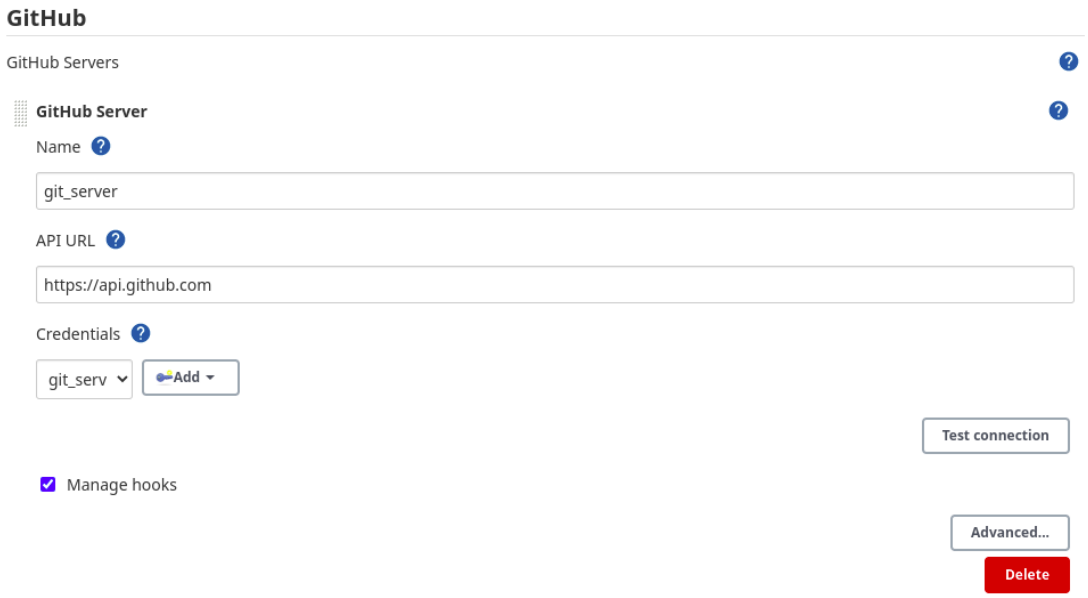


3. **Manage Jenkins → Manage Credentials**



4. **Manage Jenkins → Configure System**





Jenkins Pipeline

1. **Git Pull:** It pulls the remote repository from github using jenkins.

```
stage('Git Pull') {
    steps {
        // Get code from a GitHub repository
        // Make sure to add your own git url and credentialsId
        git url: 'https://github.com/harshshah8/Spe_mini_project.git', branch: 'master',
        credentialsId: 'GitCredential'
    }
}
```

2. **Maven Build:** It generates a jar file that contains our source code as well as any dependencies. The existing target folder with old dependencies will be deleted, and a new target folder with the new jar file will be created.

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

3. **Docker Image Creation:** It's used to produce images on our local system that are then posted to our Docker hub, allowing us to pull the image and run the application on other servers. environment just creates variables which can be used later. **:latest** is the tag name of the image.

```
    agent any

    environment {
        registry = "shahharsh8/speminiproject"
        registryCredential = 'dockerhub'
        dockerImage = ''
    }
```

```
stage('Building our image') {
    steps{
        script {
            dockerImage = docker.build registry + ":latest"
        }
    }
}
```

4. **Deploying Docker Image:** Here we are deploying the image into DockerHub so that anyone can pull the image. We have to run this command **$ sudo chmod 666 /var/run/docker.sock** in localhost in order to give the permission.

```
stage('Deploy our image') {
    steps{
        script {
            docker.withRegistry( '', registryCredential ) {
                dockerImage.push()
            }
        }
    }
}
```

5. **Ansible Deploy:**

```
stage('Ansible Deploy') {
    steps {
        //Ansible Deploy to remote server (managed host)
        ansiblePlaybook colorized: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inventory', playbook: 'p3.yml'

    }
}
```

Average stage times: (Average full run time: ~1min 12s)	1s	13s	1s	11s	9s
#17 Apr 17 17:40 No Changes	971ms	12s	2s	27s	15s
#16 Apr 17 17:38 No Changes	2s	20s	940ms failed	72ms failed	82ms failed
#15 Apr 17 17:38 No Changes	907ms	21s	2s failed	201ms failed	246ms failed
#14 Apr 17 17:34 1 commit	2s	14s	5s failed	215ms failed	121ms failed
#13 Apr 17 16:16 2 commits	1s	15s	1s	28s	33s
#12 Apr 17 15:59 1 commit	1s	13s	1s	27s	11s failed
#11 Apr 17 15:56 1 commit	1s	9s failed	66ms failed	60ms failed	67ms failed
#10 Apr 17 15:55 2 commits	1s	9s failed	61ms failed	73ms failed	57ms failed
#9 Apr 17 15:36 1 commit	1s	6s failed	54ms failed	50ms failed	55ms failed
#8 Apr 17 15:02 No Changes	934ms	12s	1s	28s	29s

- After successful execution of this pipeline, we can find out the docker image on our host by the terminal.

```
1 bash  2 bash

~
❯ docker images
REPOSITORY                  TAG       IMAGE ID       CREATED        SIZE
shahharsh8/speminiproject   latest    4c28fa87ab79   5 hours ago    662MB
shahharsh8/speminiproject   <none>    6cee052cde3c   7 hours ago    662MB
shahharsh8/speminiproject   <none>    2dbb4248f1ab   7 hours ago    662MB
shahharsh8/speminiproject   <none>    ea5ab75425a0   8 hours ago    662MB
shahharsh8/speminiproject   <none>    fffd5c45403f   8 hours ago    662MB
shahharsh8/speminiproject   <none>    e0073ded02dd   8 hours ago    662MB
shahharsh8/speminiproject   <none>    de913913584c   8 hours ago    662MB
shahharsh8/speminiproject   <none>    99a8d5d78ee9   8 hours ago    662MB
shahharsh8/speminiproject   <none>    a1277c2f3132   10 hours ago   662MB
```

- We can run this image by **docker run -it --name speminiproject 4c2** on running this image, the jar file will get executed.

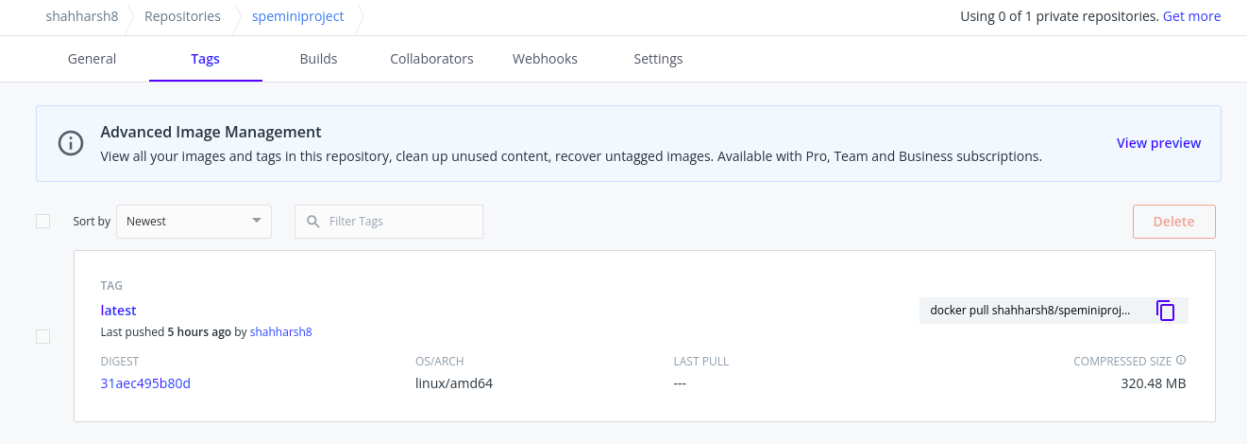
```
~
❯ docker run -it --name speminiproject 4c2
Scientific Calculator using DevOps.
 Choose operation:
1. Factorial
2. Square root
3. Power
4. Natural Logarithm
5. Exit
Enter your choice:
```

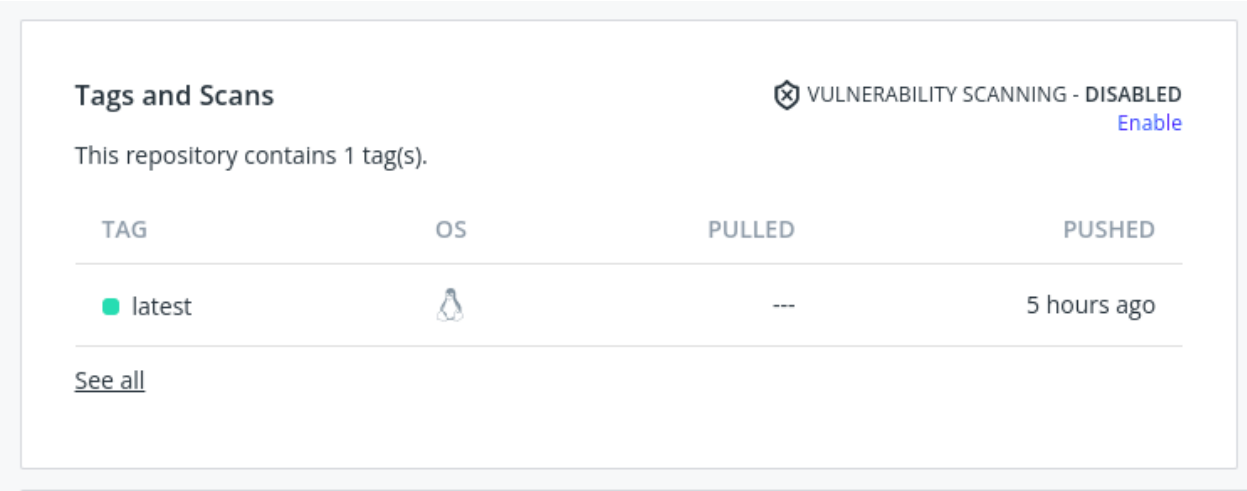
Containerize

- Docker is an operating system virtualization platform that allows applications to be delivered in containers. As a result, rather than just supplying software, the full environment is provided as a Docker image, including all software dependencies.
- So, using open-JDK 11 and the calculator jar file, we'll create a docker image. After that, the image will be posted to the Docker Hub (we need to create a public repository on the docker hub before pushing the image). Ansible will then fetch this image from Docker Hub and deploy it across many machines.

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

- To build a docker image, a docker file is used in which script is written. In the above mentioned docker file,
- **FROM:** It imports the base image openjdk11 inorder 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.





Deployment

- Ansible is a platform for open-source automation. It's an automation engine that executes Ansible playbooks, which are defined tasks that define environments and workflows.
- **Steps to Install**
  - Sudo apt install openssh-server
  - Ssh-keygen -t rsa
  - Ssh-copy-id <username>@<IP>
  - Sudo apt install ansible
- **Playbook**

```yaml
---
- name: Pull docker image of Calculator
  hosts: all
  tasks:

    - name: Start docker service
      service:
        name: docker
        state: started

    - name: pull docker image
      shell: docker pull shahharsh8/speminiproject

    - name: running container
      shell: docker run -it -d shahharsh8/speminiproject
```

- **Inventory File:**

```
[ubuntu18]
10.0.2.15 ansible_user=harsh
```

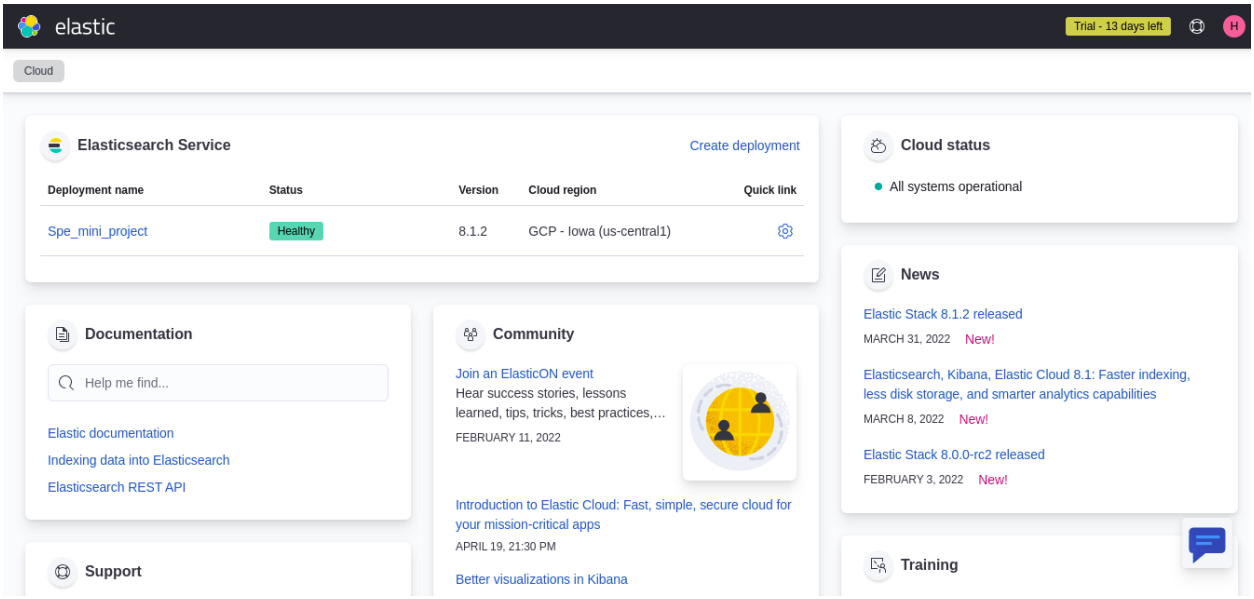
Continuous Monitoring

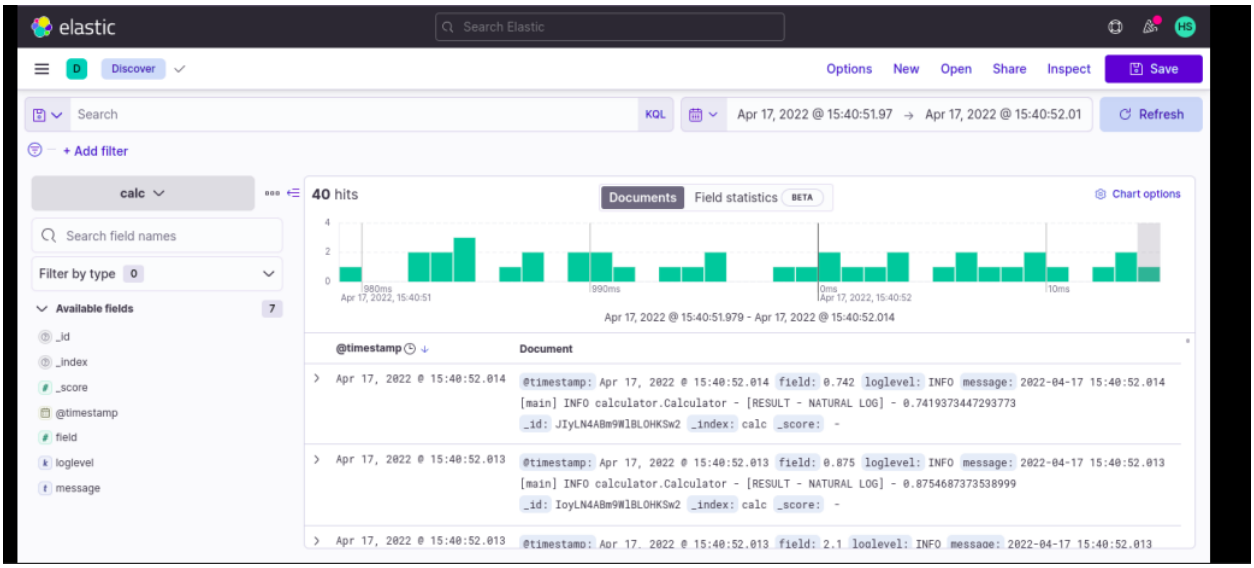
- After you've completed your deployment, the following step is to monitor your system. Monitoring entails determining whether or not the software is performing as intended.
- ELK stack makes the monitoring tool for any deployed software, it analyzes the logs and the same analysis can then be viewed on the kibana dashboard. ELK stack is

comprises 3 independent components: **Elasticsearch, Logstash, Kibana.**

```
❯ ls
calculator.log  deploy-docker  Dockerfile  inventory  p3.yml  pom.xml  src  target

Spe_mini_project on master [?] is 1.0-SNAPSHOT via v16.0.2
❯ cat calculator.log
2022-04-17 15:40:51.979 [main] INFO  calculator.Calculator - [SQ ROOT] - 4.0
2022-04-17 15:40:51.982 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 2.0
2022-04-17 15:40:51.982 [main] INFO  calculator.Calculator - [SQ ROOT] - 1.0
2022-04-17 15:40:51.983 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 1.0
2022-04-17 15:40:51.983 [main] INFO  calculator.Calculator - [SQ ROOT] - 9.0
2022-04-17 15:40:51.984 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 3.0
2022-04-17 15:40:51.984 [main] INFO  calculator.Calculator - [SQ ROOT] - 36.0
2022-04-17 15:40:51.984 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 6.0
2022-04-17 15:40:51.986 [main] INFO  calculator.Calculator - [SQ ROOT] - 3.0
2022-04-17 15:40:51.987 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 1.7320508075688772
2022-04-17 15:40:51.987 [main] INFO  calculator.Calculator - [SQ ROOT] - 4.0
2022-04-17 15:40:51.989 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 2.0
2022-04-17 15:40:51.989 [main] INFO  calculator.Calculator - [SQ ROOT] - 9.0
2022-04-17 15:40:51.990 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 3.0
2022-04-17 15:40:51.990 [main] INFO  calculator.Calculator - [SQ ROOT] - 36.0
2022-04-17 15:40:51.991 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 6.0
2022-04-17 15:40:51.993 [main] INFO  calculator.Calculator - [NATURAL LOG] - 1.0
2022-04-17 15:40:51.994 [main] INFO  calculator.Calculator - [RESULT - NATURAL LOG] - 0.0
2022-04-17 15:40:51.995 [main] INFO  calculator.Calculator - [NATURAL LOG] - 1.0
2022-04-17 15:40:51.995 [main] INFO  calculator.Calculator - [RESULT - NATURAL LOG] - 0.0
2022-04-17 15:40:51.998 [main] INFO  calculator.Calculator - [POWER - 2.0 RAISED TO] 3.0
2022-04-17 15:40:51.999 [main] INFO  calculator.Calculator - [RESULT - POWER] - 8.0
2022-04-17 15:40:52.000 [main] INFO  calculator.Calculator - [POWER - 3.0 RAISED TO] 3.0
2022-04-17 15:40:52.000 [main] INFO  calculator.Calculator - [RESULT - POWER] - 27.0
2022-04-17 15:40:52.001 [main] INFO  calculator.Calculator - [POWER - 4.0 RAISED TO] 3.0
2022-04-17 15:40:52.002 [main] INFO  calculator.Calculator - [RESULT - POWER] - 64.0
2022-04-17 15:40:52.003 [main] INFO  calculator.Calculator - [POWER - 5.0 RAISED TO] 3.0
2022-04-17 15:40:52.003 [main] INFO  calculator.Calculator - [RESULT - POWER] - 125.0
2022-04-17 15:40:52.005 [main] INFO  calculator.Calculator - [POWER - 2.0 RAISED TO] 3.0
2022-04-17 15:40:52.006 [main] INFO  calculator.Calculator - [RESULT - POWER] - 8.0
2022-04-17 15:40:52.006 [main] INFO  calculator.Calculator - [POWER - 3.0 RAISED TO] 3.0
2022-04-17 15:40:52.007 [main] INFO  calculator.Calculator - [RESULT - POWER] - 27.0
2022-04-17 15:40:52.008 [main] INFO  calculator.Calculator - [POWER - 4.0 RAISED TO] 3.0
```





Use Docker Image on Remote Server

- Install Docker and pip on the remote server:
  - **sudo apt install python-pip**
  - **pip install docker**
  - **sudo apt-get install docker.io**
- A connection can be established using SSH key.
- **SSH key generation:**
  - Run following commands on machine:
    - $ sudo apt get install openssh-server
    - $ sudo su Jenkins
    - $ ssh keygen -t rsa
    - $ ssh-copy-id harsh@172.19.114.214

WebHooks

- Webhooks are messages that are sent automatically whenever something changes. In our scenario, the webhook will automatically start the Jenkins pipeline if we make any updates to the GitHub repo.

- Ngrok uses secure tunnels to connect local servers behind NATs (Network Address Translation) and firewalls to the public internet. It has a real-time web interface that allows you to inspect any HTTP traffic passing through your tunnels. It allows you to connect to the internet via a web server running on your local system. Simply specify the port on which your web server is listening to ngrok.

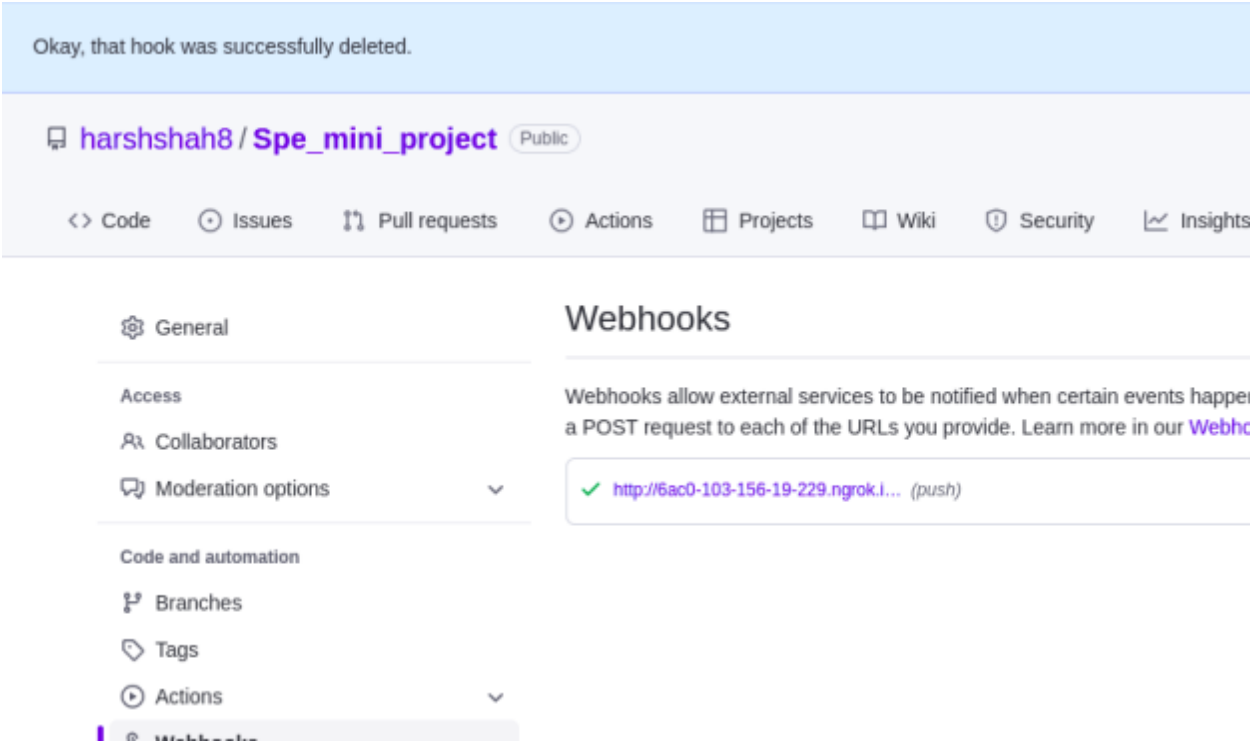
```
~
❯ ngrok http 172.16.130.99:8080;
```

```
1 bash  2 bash  3 bash  4 bash
ngrok by @inconshreveable

Session Status                online
Session Expires               1 hour, 59 minutes
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://f12b-119-161-98-68.ngrok.io -> http://172.16.130.99:8080
Forwarding                    https://f12b-119-161-98-68.ngrok.io -> http://172.16.130.99:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

- In github repository go to settings and in Webhooks add ngrok address in payload url and the personal access token in secret.



- In Jenkins → Configure System

Challenges

- Maven Build Error: An error was received while developing the project in Intellij: "Are you trying to execute JRE instead of JDK."My machine had two versions of JDK installed, and deleting one of them fixed the problem.
- I was not able to connect my local system to virtual box through SSH. After changing adapter setting it was resolved.
- Error while connecting to the Docker Daaemon socket : Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock. We need to run the command sudo chmod 666 /var/run/docker.sock on the local as

well as the server machine. We need to run this command everytime we restart the server machine.