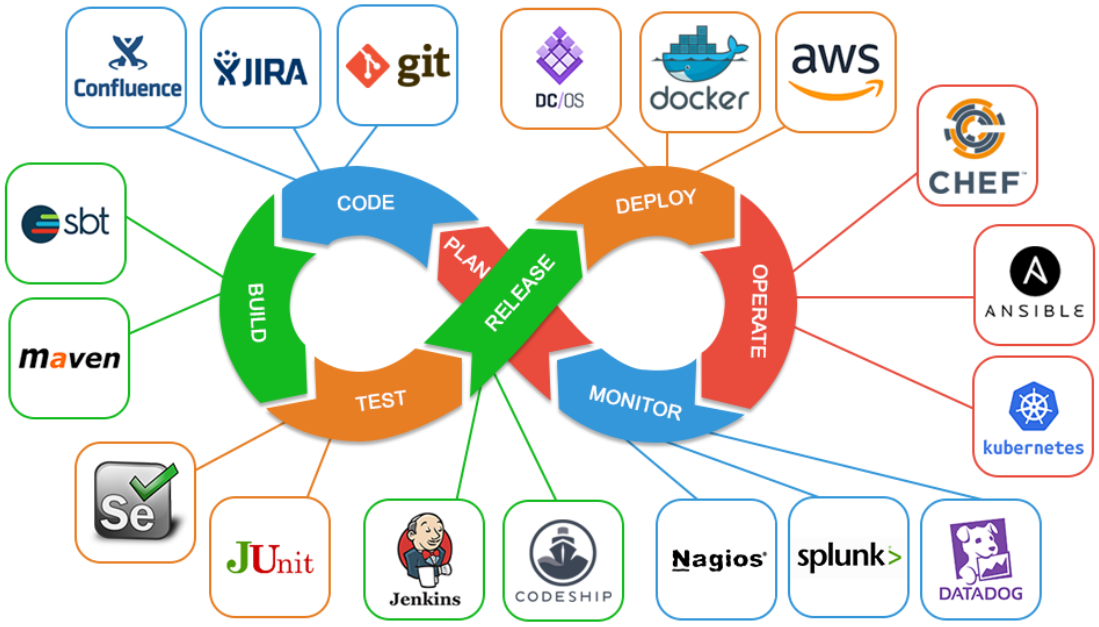
Scientific Calculator using Devops Methodologies

HIMANK JAIN

MT2021054

Software Production Engineering (CS 816)



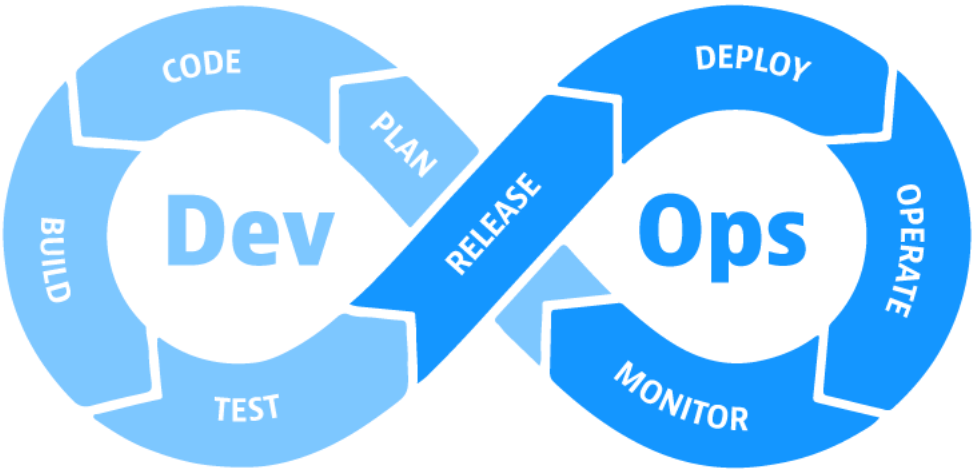
Github Profile: https://github.com/himankjn/MiniProject_SPE
DockerHub Profile: https://hub.docker.com/repository/docker/himankjn/check

Table Of Contents:

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

DevOps:



DevOps is a set of practices that combines software development and IT operations. It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from the Agile methodology.

The cultural philosophies, practices, and tools in DevOps increase an organisation's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organisations using traditional software development and infrastructure management processes. It enables faster development of new products and easier maintenance of existing deployments. This speed enables organisations to better serve their customers and compete more effectively in the market.

DevOps tool chain:

The project pipeline includes:

1. **Github**: Source control management tool
2. **Junit**: Testing
3. **Maven**: Build Automation & Dependency Management
4. **Jenkins, Github CLI**: Continuous Integration
5. **IntelliJ:** Integrated Development Environment
6. **Docker**: Containerization
7. **Docker Hub**: Docker Images
8. **Ansible**: Configuration Management and Remote Deployment
9. **VirtualBox**: Deployment on VirtualMachine
10. **ELK Stack**: Logs & Monitoring
11. **Java 11**: Language
12. **ngrok**: Exposing local machine port to the internet

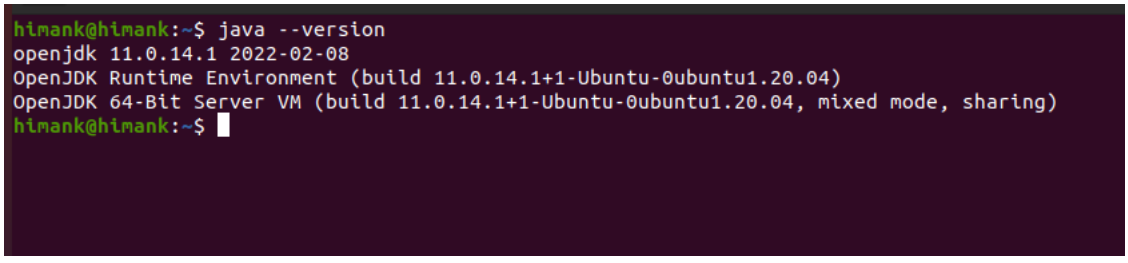
    We use two machines:

    1. **Host** [Ubuntu20]
    2. **Virtual Machine Remote Server** [Ubuntu18]

Setup:

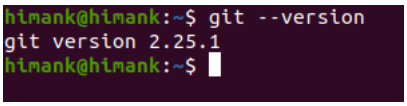
- Download and Install the following on your host machine.
  - **Java 11:** `sudo apt install openjdk-11-jdk`

    ```
    java --version
    ```

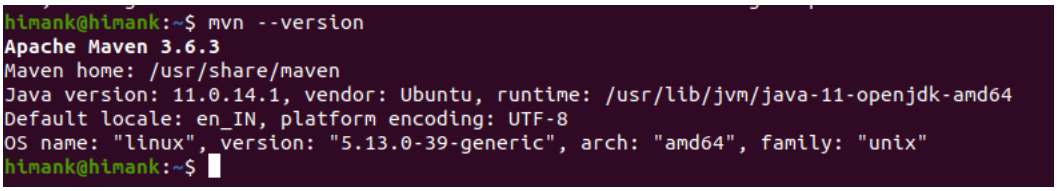
  - **Git:** `sudo apt install git`

```
git --version
```



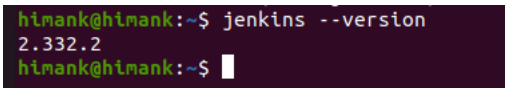
- **Maven:** `sudo apt install maven`

```
mvn --version
```



- **Jenkins:** `sudo apt install jenkins`

```
jenkins --version
```

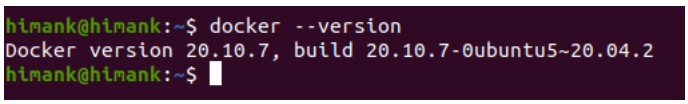


- **Ansible:** `sudo apt install ansible`

```
ansible --version
```

- **IntelliJ:** `sudo snap installintellij-idea-ultimate --classic`
- **Docker:** `sudo apt install docker.io`, `sudo snap install docker`

```
docker --version
```



- **ngrok:** `sudo snap install ngrok`

```
ngrok --version
```

```
himank@himank:~$ ngrok --version
ngrok version 2.3.40
himank@himank:~$
```

- **openssh**: `sudo apt-get install openssh-server`

```
ssh -V
```

```
himank@himank:~$ ssh -V
OpenSSH_8.2p1 Ubuntu-4ubuntu0.4, OpenSSL 1.1.1f  31 Mar 2020
himank@himank:~$
```

- Download and install **Docker** and **openssh** on your remote server(virtual machine):
  - **pip:** `sudo apt install python-pip`
  - **docker**: `pip install docker`
  - **openssh**: `sudo apt-get install openssh-server`

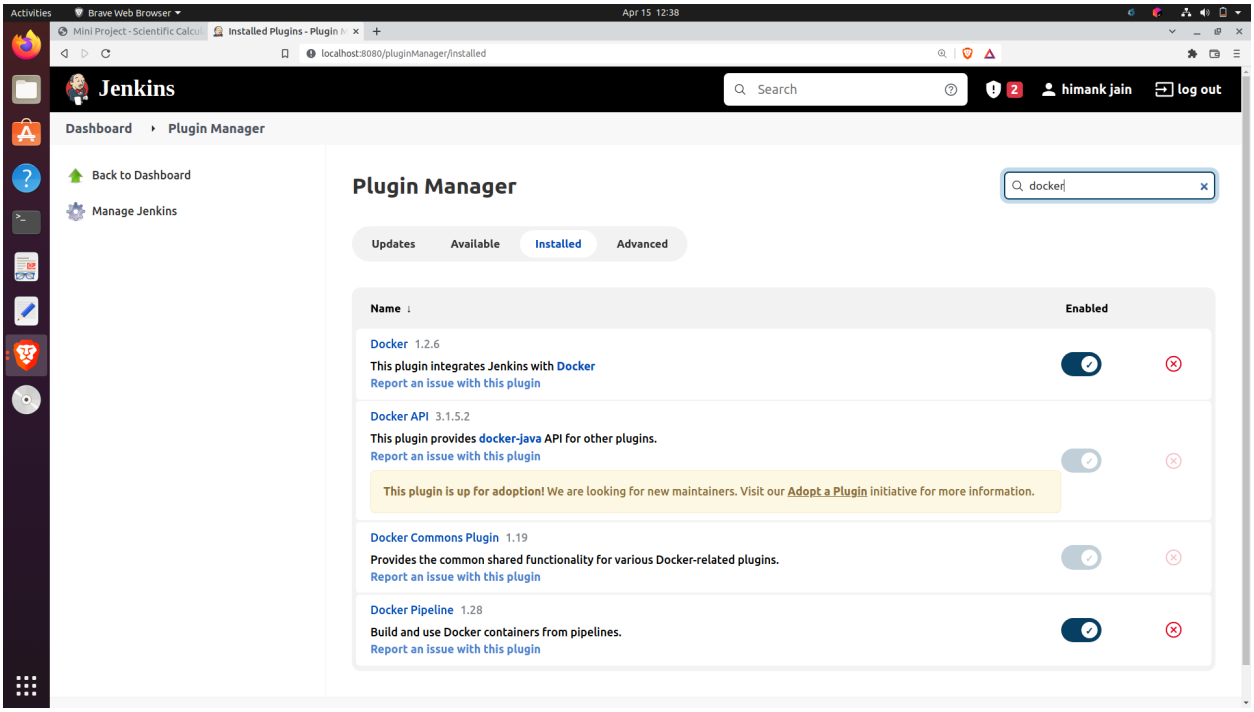
Jenkins Prerequisites:

Plugins:

Navigate to **Dashboard→Manage Jenkins→Manage Plugins**

Download & install the following plugins:

- **Docker**
- **Docker pipeline**
- **Ansible plugin**
- **Github Integration Plugin**
- **Maven Integration**
- **Pipeline**



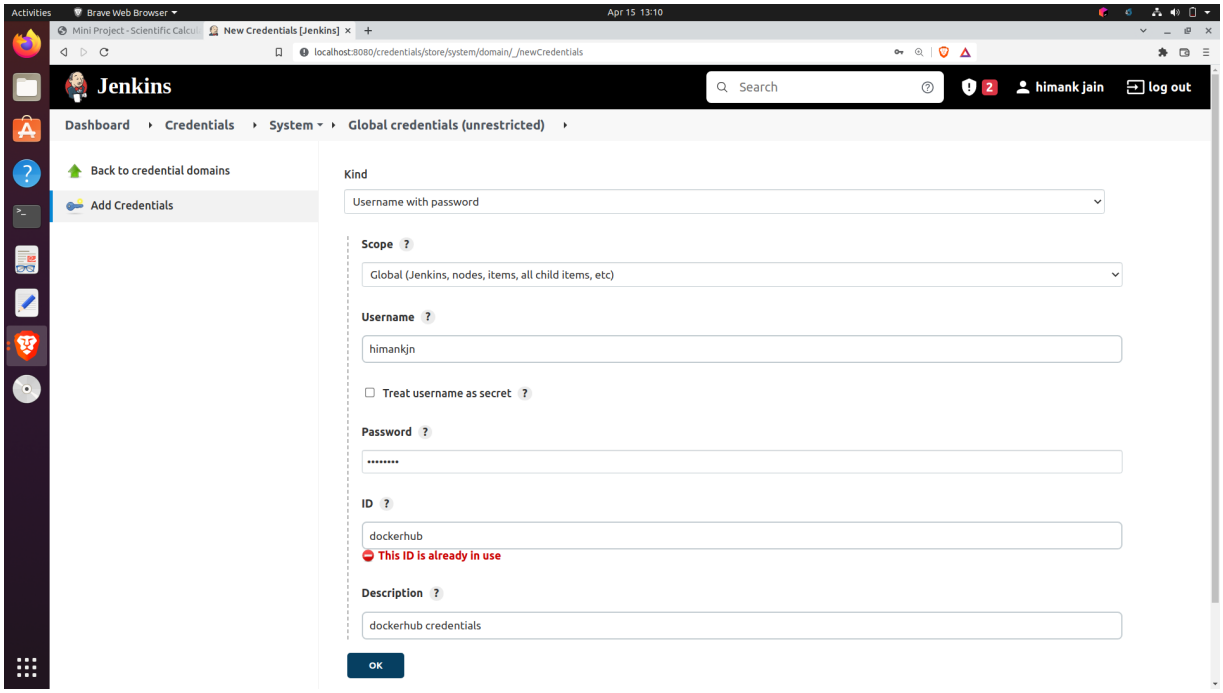
Credentials:

Add Dockerhub credentials:

Navigate to Manage Jenkins→ Manage Credentials→Add Credentials

Docker hub:

1. **Kind**: Username and Password
2. **Username**: DockerHub username
3. **Password**: DockerHub Password
4. **ID**: Identification [”dockerhub”]
5. **Description**

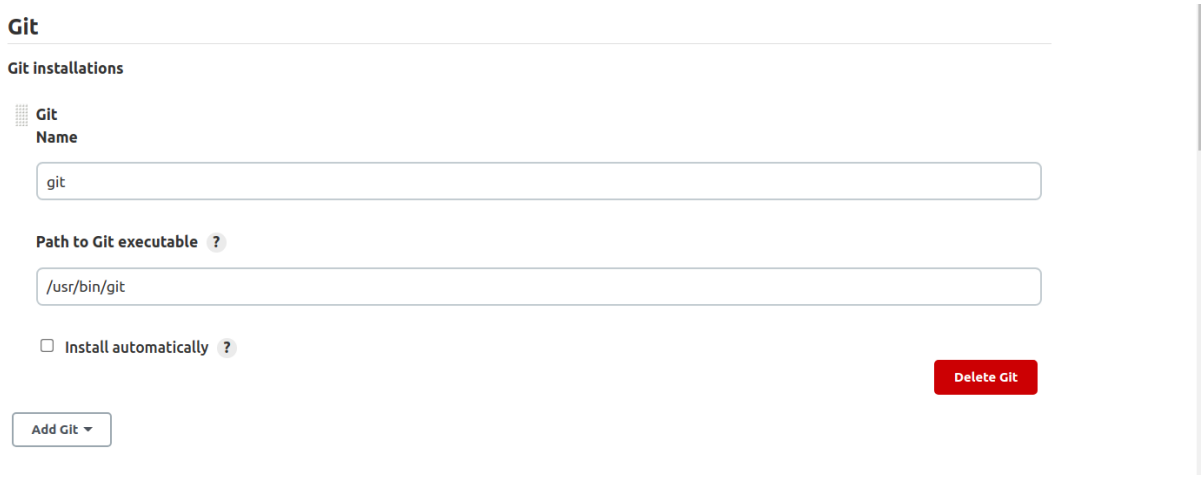


Tools Configuration:

- Navigate to **Manage Jenkins → Global Tool Configuration → Git:**

  Name: git

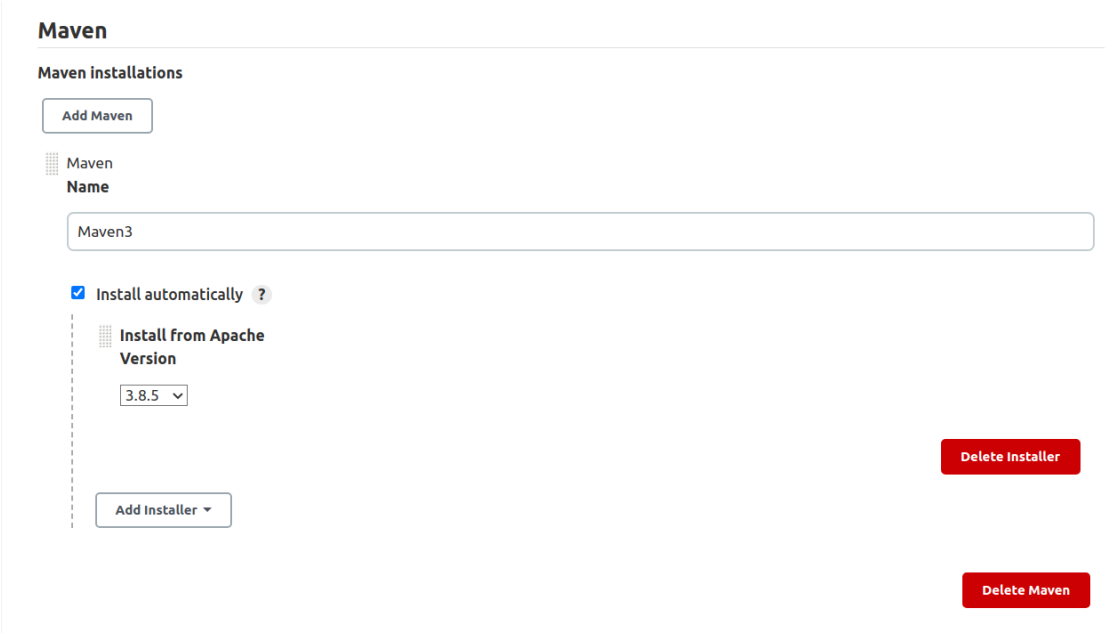
  Path to Git Executable: /usr/bin/git



- Navigate to **Manage Jenkins → Global Tool Configuration → Maven**

  Name: Maven3

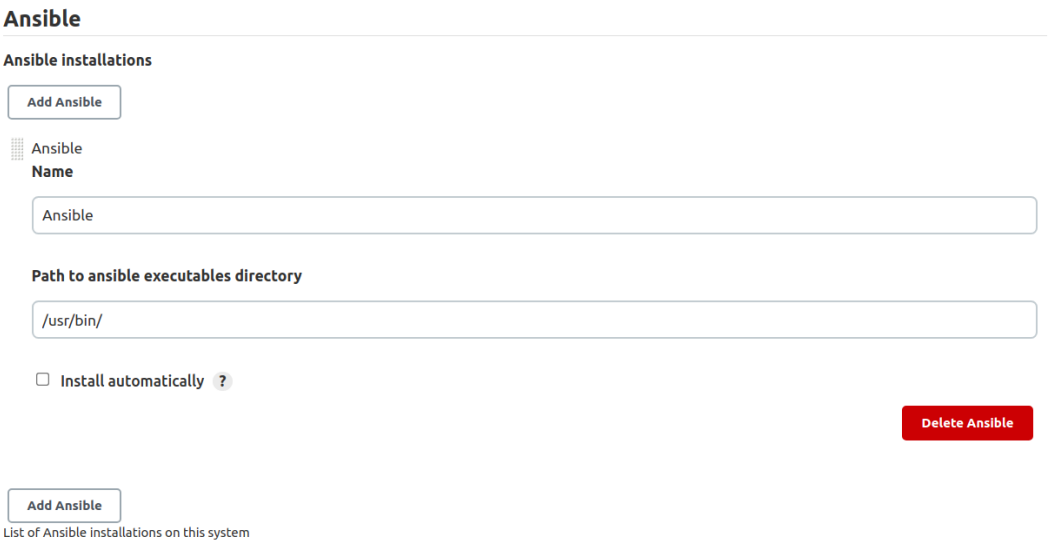
  Add Installer: Install from Apache



- Navigate to **Manage Jenkins → Global Tool Configuration → Ansible**

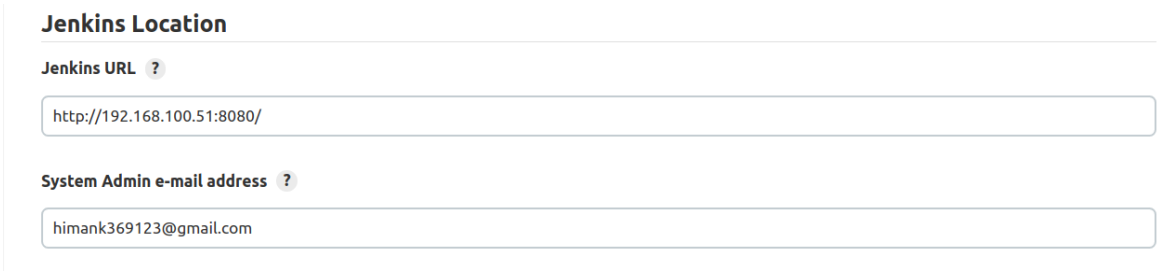
  Name: Ansible

  Path to ansible executable directory: /usr/bin/

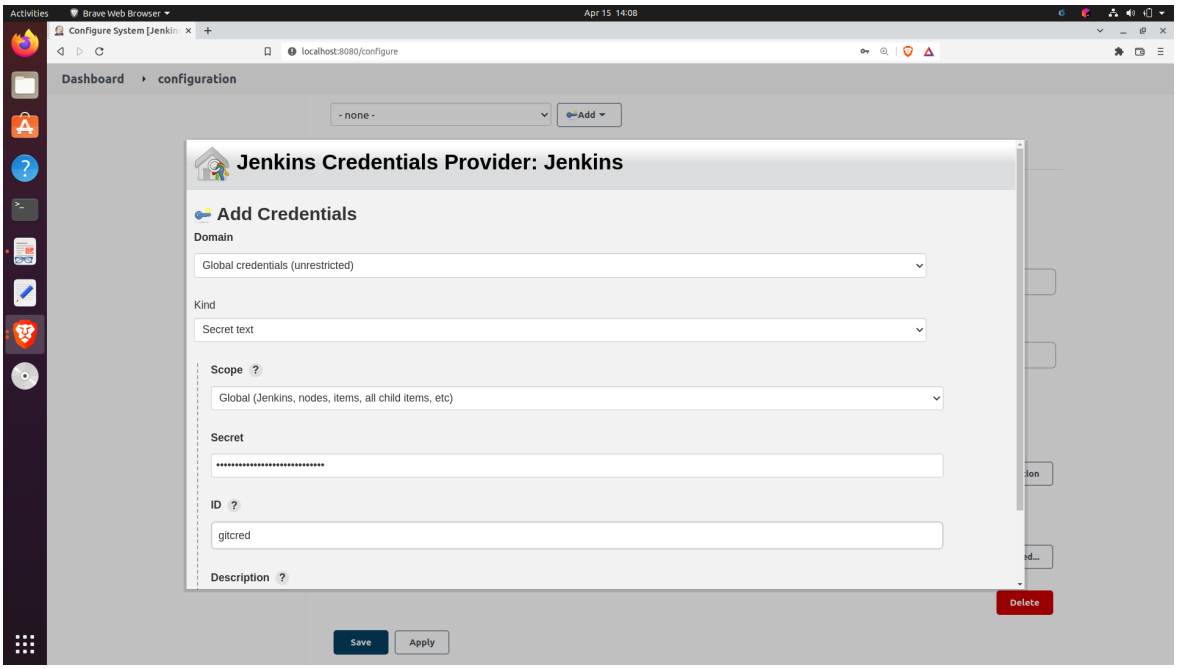


Jenkins Configuration:

- Navigate to Manage Jenkins → Configure System → Jenkins URL
  - Add your jenkins address with port
  - http://192.168.100.51:8080/
- Navigate to Manage Jenkins → Configure System → System Admin e-mail address:
  - Add your admin email address
  - himank369123@gmail.com

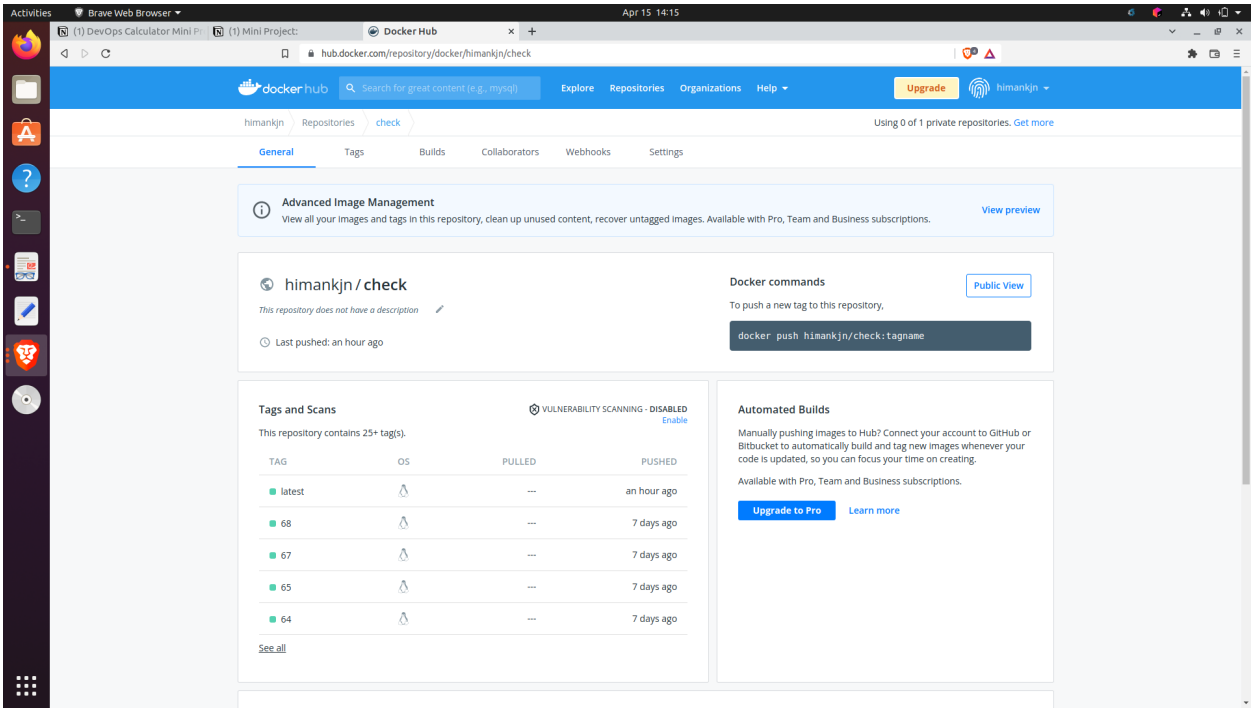


- Navigate to Manage Jenkins → Configure System → Github → Github Server:
  - **Name:** servername [gitserver]
  - **API URL**: https://api.github.com
  - **Credentials**: Add Credential
    - **Kind**: SecretText
    - **Secret**: Personal Access Token
    - **ID**: Identification [”gitcred”]
    - **Description**



Docker Hub:

- Create a new public repository with format : username/dockerimage
- We will use this repository for pushing out docker image and pulling it in remote server.



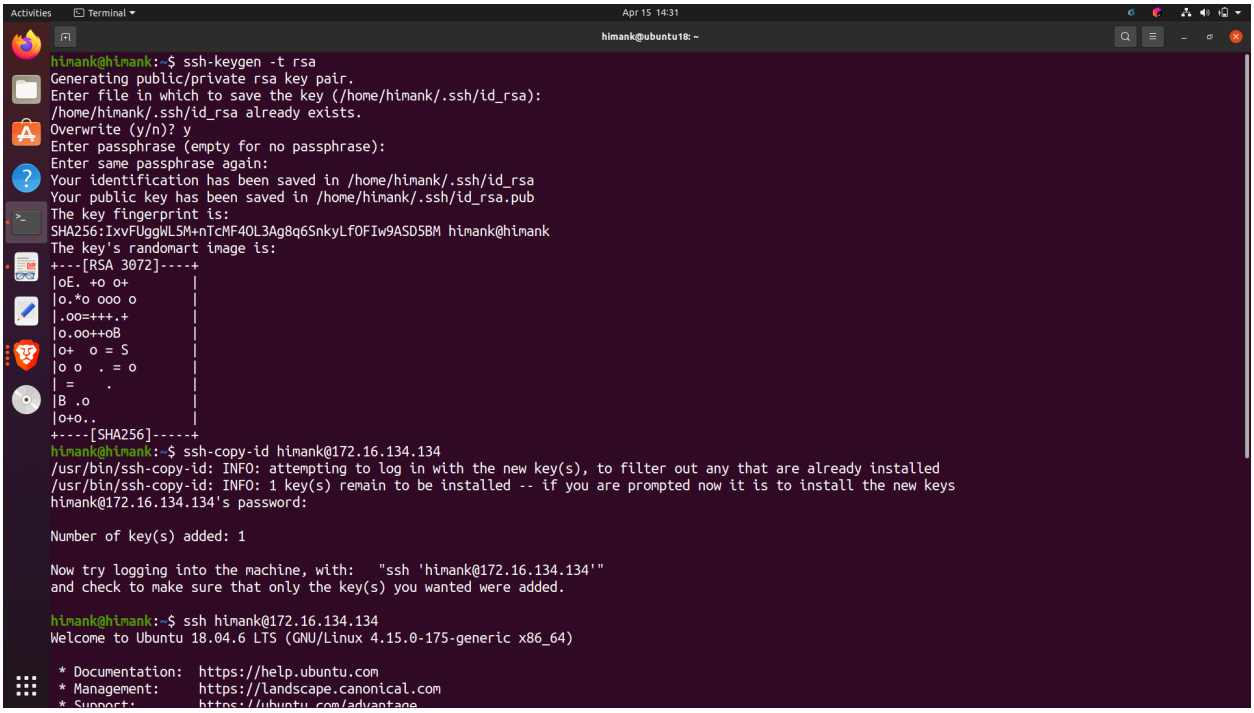
Steps:

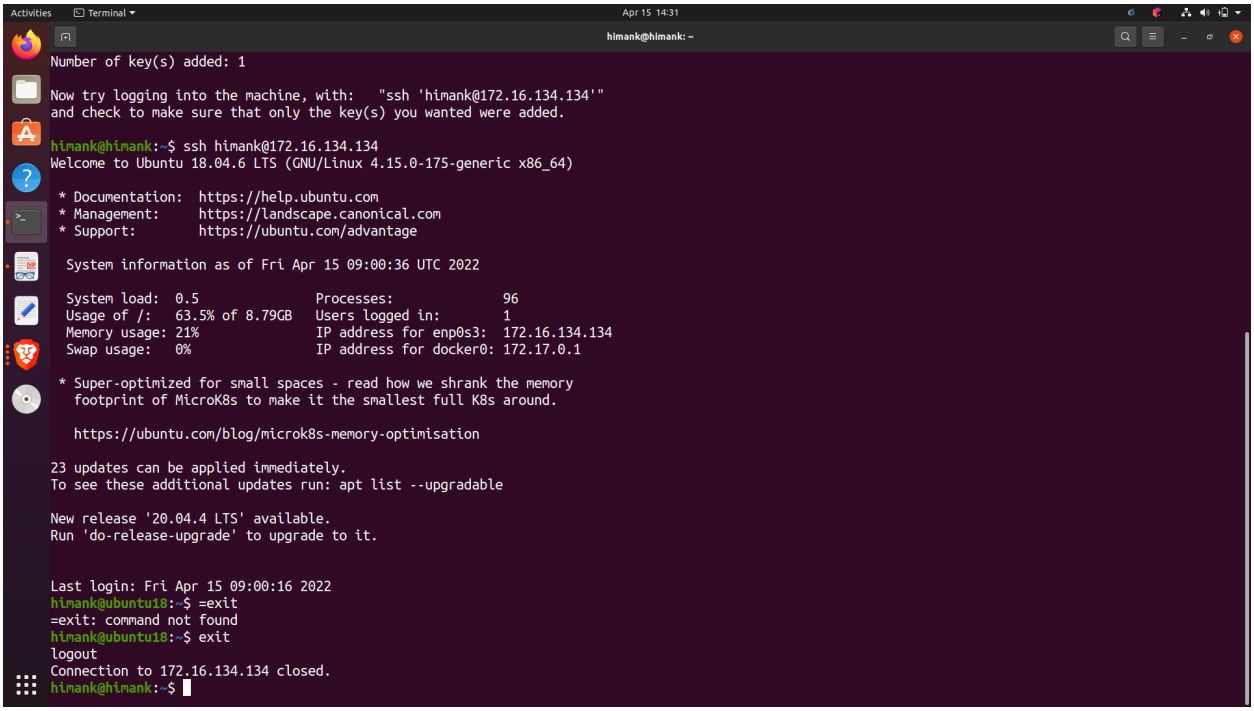
SSH Key:

- Generate ssh public key on local host and copy it to remote server to ensure login without password. Then test login using ssh.

  NOTE: Ensure remote server is up and running.

```
ssh-keygen -t rsa
ssh-copy-id himank@10.0.2.15
#10.0.2.15 is remote server[ubuntu18's] ip address
ssh himank@10.0.2.15
```





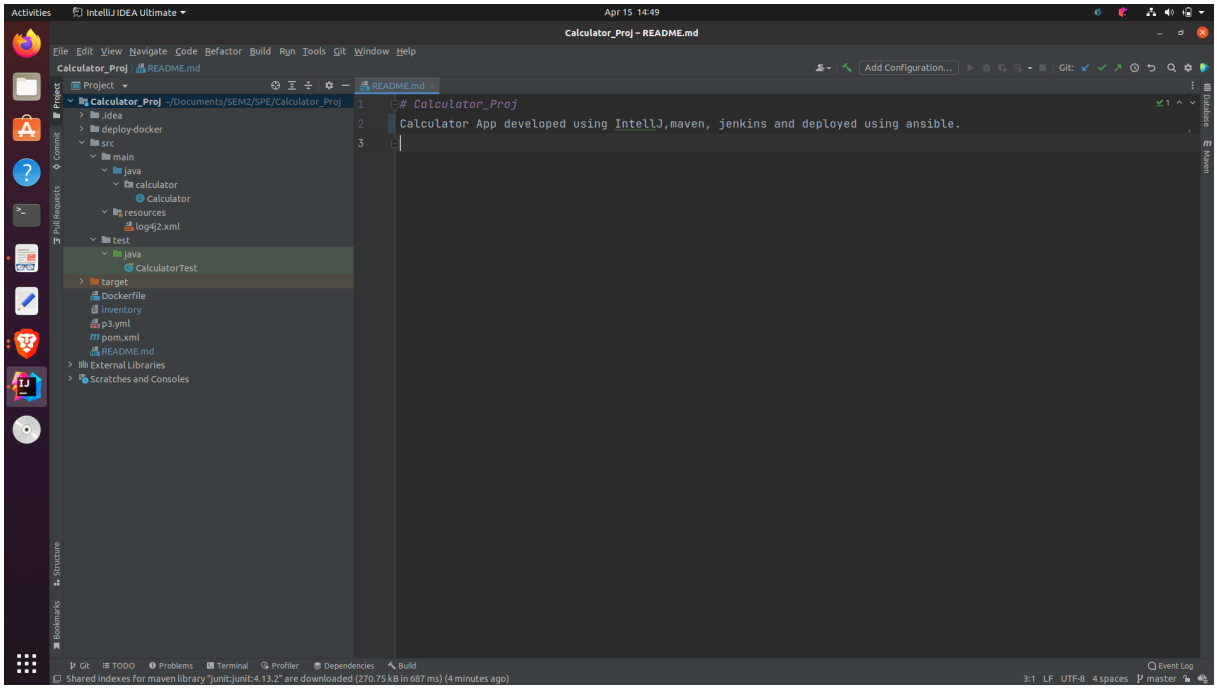
- Create new IntelliJ Maven project.

  IntelliJ→ New Project→Maven→[Give it a name]→Finish

  - Ensure a project folder structure like this

    - **p3.yml:** Playbook for ansible remote deployment
    - **inventory:** file for ansible connection to remote server

- **pom.xml:** Project Object Model i.e. maven file for dependencies.
- **Dockerfile:** docker file containing instructions to build the image with final jar file and run it.
- **src/main/java/calculator/Calculator.java:** Java file containing code for scientific calculator
- **src/test/java/CalculatorTest.java:** Java file containing testing code for scientific calculator



DockerFile: Docker file containing steps to build image.

```
FROM openjdk:8
COPY ./target/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java", "-jar", "calculat orDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

- uses openjdk:8 as base image for our new image
- copies the jar file from local file system to image
- changes the working directory
- executes java -jar command to run the jar file.

p3.yml: Ansible Playbook specifying the commands to execute on remote server.

```
---
- 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 himankjn/check
```

```
    - name: running container
      shell: docker run -it -d himankjn/check
```

- Starts docker in remote server.
- pulls the docker image himankjn/check from dockerhub
- runs the image and creates a container in remote server.

inventory: ansilble file with information about remote server

```
[ubuntu18]
172.16.134.134 ansible_user=himank
```

- specifies remote server ip address and user account.

src/main/java/calculator/Calculator.java: Java file containing code for scientific calculator

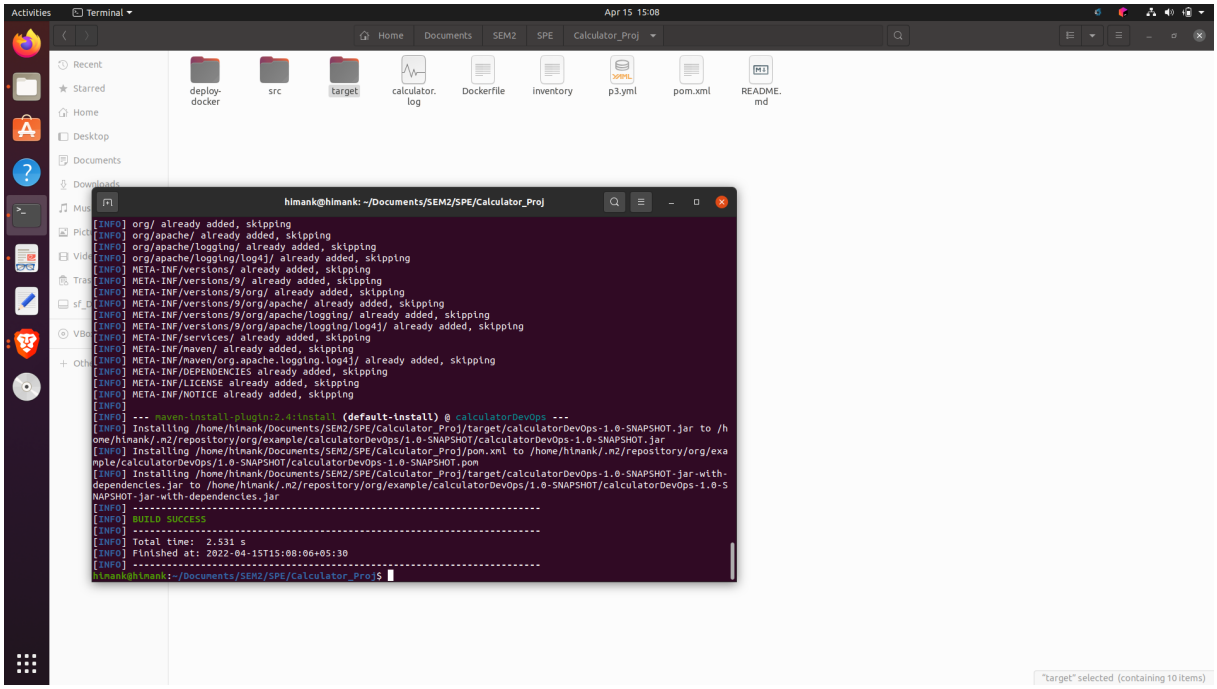
src/test/java/CalculatorTest.java: Java file containing testing code for scientific calculator

Build project using maven install command

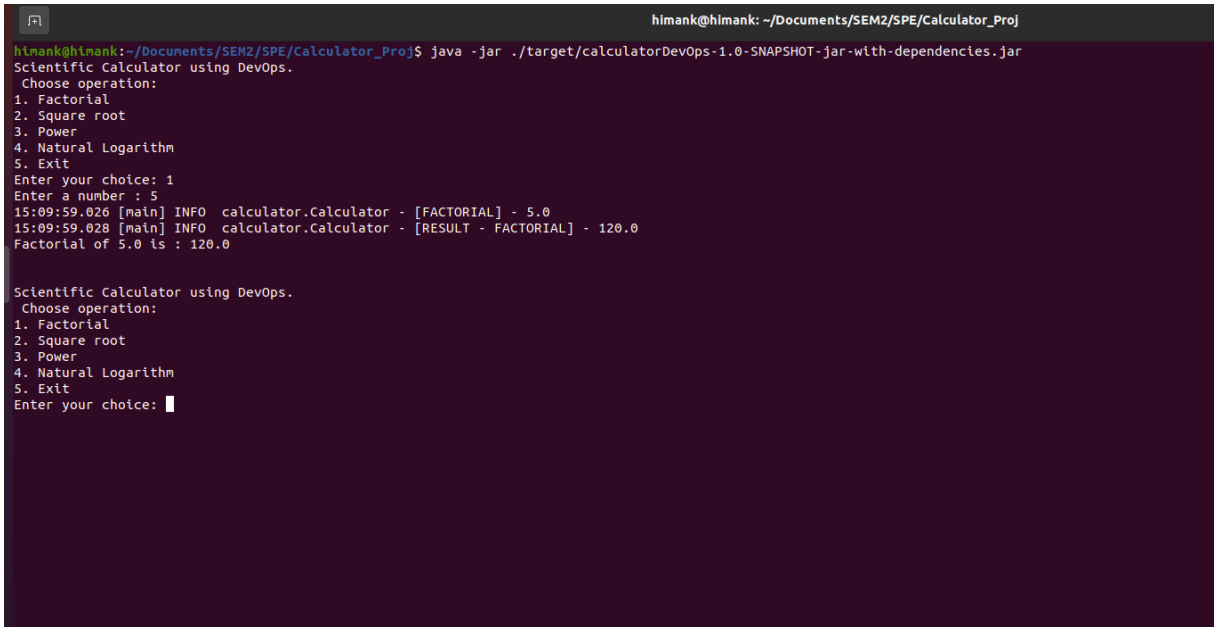
pom.xml: Maven file containing all dependencies.

Testing out the code on local machine:

- `mvn clean install` Takes the project code, compiles it, tests it and converts it into a jar executable. Run this command inside the project folder.



- `java -jar ./target/calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar` Tests the snapshot jar file created under target folder on local machine.

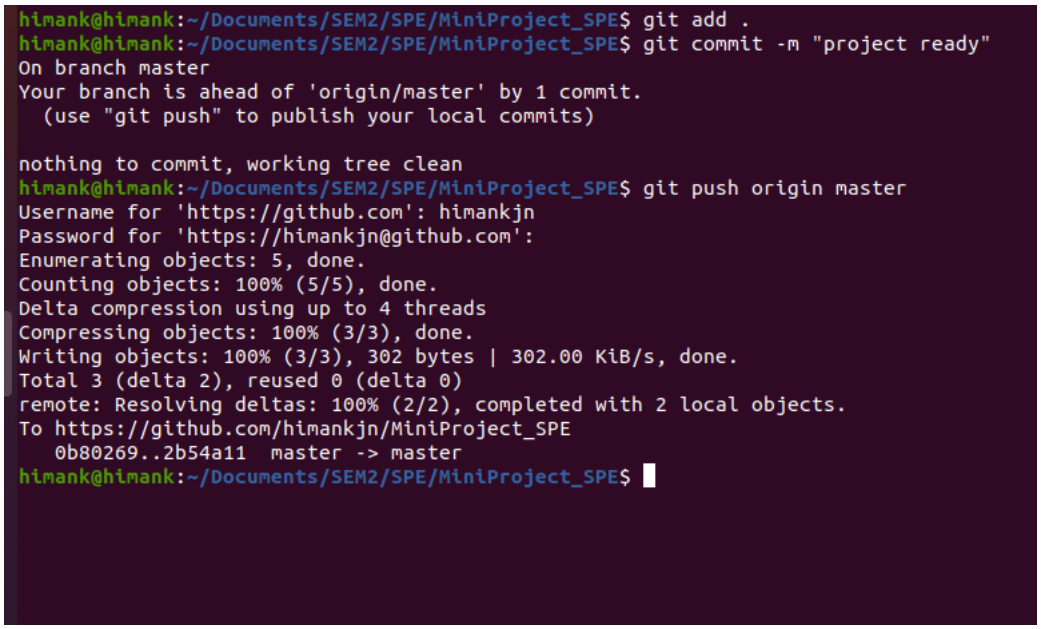


- Once your code is working flawlessly in local machine, push it to GitHub.

Push to Github:

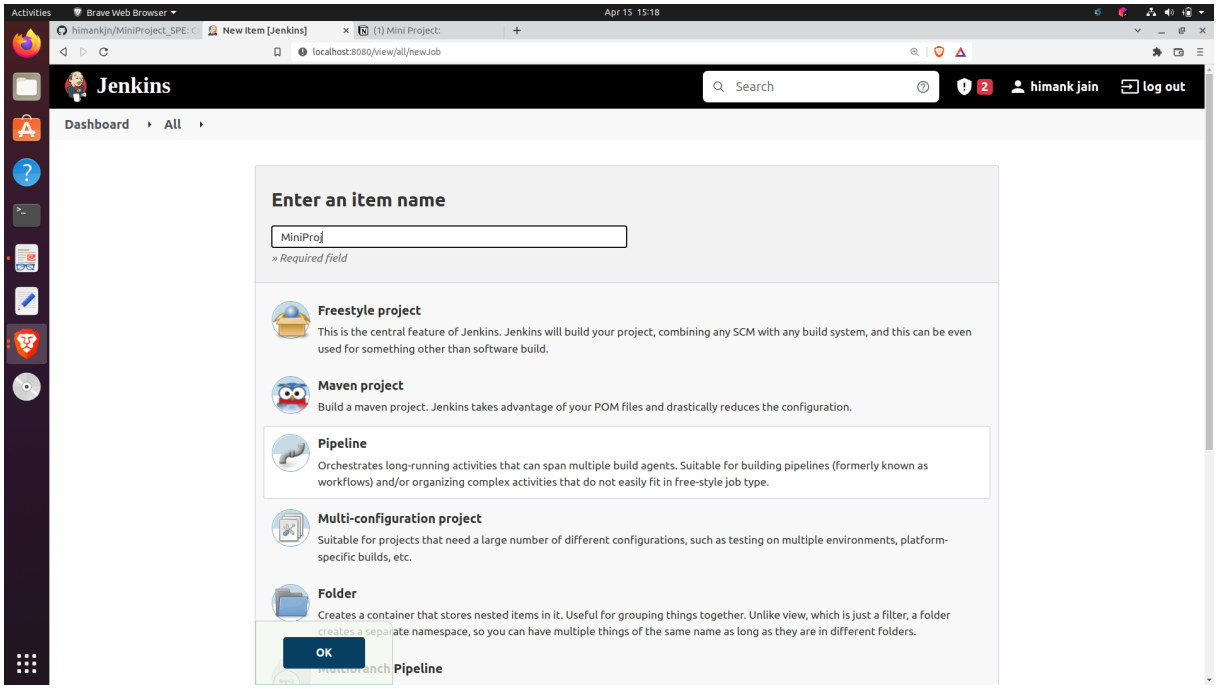
- push the code repository to GitHub for version control and automation.

```
#run the following in project folder
git init
git add .
git commit -m "project ready"
git remote add origin https://github.com/himankjn/MiniProject_SPE.git
git push origin master
```

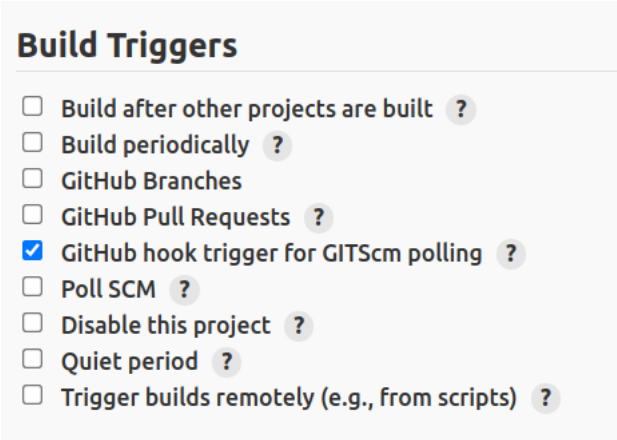


Jenkins Project:

- Create a new pipeline project in jenkins with pipeline script and make specific changes according to your credentials.
- **New Item → Pipeline**



- Give Project a name and click ok
- Give it a description.
- Enable **GitHub hook trigger for GITScm polling** under **Build Triggers**



- Under **Pipeline** add following script with required changes

```
pipeline {
    // The “agent” section configures on which nodes the pipeline can be run.
    // Specifying “agent any” means that Jenkins will run the job on any of the
    // available nodes.
    agent any

      environment {
```

```
            registry = "himankjn/check"
            registryCredential = 'dockerhub'
            dockerImage = ''
        }

  stages {
        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/himankjn/MiniProject_SPE.git', branch: 'master'
            }
        }
        stage('Maven Build') {
            steps {
                // Maven build
                sh 'mvn clean install'
            }
        }
        stage('Building our image') {
            steps{
                script {
                    dockerImage = docker.build registry + ":latest"
                }
            }
        }
        stage('Deploy our image') {
            steps{
                script {
                    docker.withRegistry( '', registryCredential ) {
                        dockerImage.push()
                    }
                }
            }
        }

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

            }
        }
    }
}
```

Stages in Pipeline:

1. **Git Pull**

   It is used to pull remote repository from your github using Jenkins

   ```
   stage('Git pull'){
     steps{
       git url: 'https://github.com/himankjn/MiniProject_SPE.git', branch: 'master'
     }
   }
   ```

2. **Maven Build**

   It is used to create the jar file which will contain our source code along with dependencies.

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

3. **Docker Image Creation**

It is used to create images in our local system which is pushed onto our docker hub so that we can pull the image and run the application on different servers.

```
environment{
  registry = 'himankjn/calculator'
  dockerImage = ''
    }
stage('Docker Image Creation'){
  steps{
    script{
      dockerImage = docker.build registry + ':latest'
    }
  }
}
```

`:latest` is the tag name of the image

`docker build -t imagename .` will build an image according to the DockerFile contents in current directory.

4. **Deploying the Docker Image to Docker Hub**

   use the above created docker credentials and image to push the image to docker hub.

```
environment{
  registryCredential: dockerhubcred
}
```

```
stage("Pushing Image to Docker Hub"){
  steps{
    script{
      docker.withRegistry( '', registryCredential ){
                      dockerImage.push()
                  }
    }
  }
}
```

5. **Ansible Deploy**

   It builds Infrastructure for remote servers using the playbook.

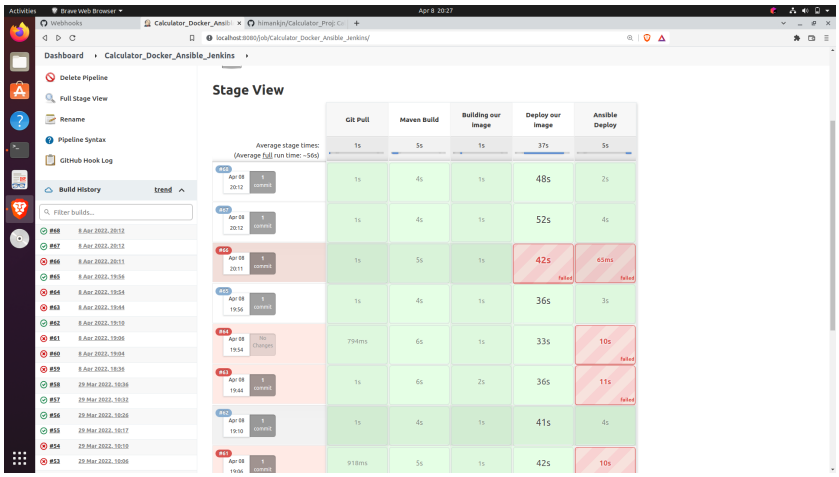
   ansible-playbook command uses 3 files:

   - playbook **[p3.yml]**: Steps to perform in remote server
   - **inventory**: IP address and username of remote server

```
stage('Ansible Deploy') {
            steps {
                ansiblePlaybook colorized: true, disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inventory', playboo

            }
       }
```

6. **Jenkins Build:**
   Click on Build Now and let the pipeline run.



7. **Run the image in a remote server**

   Once the pipeline executes ansible playbook would have pulled the image from docker hub to remote server.

   check the image availability in remote server using: `docker images`

```
File  Machine  View  Input  Devices  Help
himank@ubuntu18:~$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED        SIZE
himankjn/check    latest    fcc56fe19e9d   3 hours ago    528MB
himankjn/check    <none>    ed458c021449   4 hours ago    528MB
himank@ubuntu18:~$ _
```

   Check if the container is created and running in background: `docker ps`

```
himank@ubuntu18:~$ docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS         PORTS     NA
MES
e5a9b46b3854   himankjn/check   "java -jar calculato…"   3 seconds ago   Up 2 seconds             jo
vial_wing
himank@ubuntu18:~$
```

   Run the docker container by attaching it to a terminal using : `docker exec -it e5 /bin/bash`

   Tip: we can specify just the first 2-3 letters of the container id and it still works.

```
himank@ubuntu18:~$ docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED             STATUS             PO
RTS     NAMES
e5a9b46b3854   himankjn/check   "java -jar calculato…"   About a minute ago   Up About a minute
        jovial_wing
himank@ubuntu18:~$ docker exec -it e5 /bin/bash
root@e5a9b46b3854:/# ls
bin                                                dev   lib64  proc  srv  var
boot                                               etc   media  root  sys
calculator.log                                     home  mnt    run   tmp
calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar  lib   opt    sbin  usr
root@e5a9b46b3854:/#
```

   once inside the container, List the files using `ls`

run the executable jar using `java -jar calculatorDeOps-1.0-SNAPSHOT-jar-with-dependencies.jar`

```
       jovial_wing
himank@ubuntu18:~$ docker exec -it e5 /bin/bash
root@e5a9b46b3854:/# ls
bin                                                dev   lib64  proc  srv  var
boot                                               etc   media  root  sys
calculator.log                                     home  mnt    run   tmp
calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar  lib   opt    sbin  usr
root@e5a9b46b3854:/# ls
bin                                                dev   lib64  proc  srv  var
boot                                               etc   media  root  sys
calculator.log                                     home  mnt    run   tmp
calculatorDevOps-1.0-SNAPSHOT-jar-with-dependencies.jar  lib   opt    sbin  usr
root@e5a9b46b3854:/# 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
10:44:11.737 [main] INFO  calculator.Calculator - [FACTORIAL] - 5.0
10:44:11.741 [main] INFO  calculator.Calculator - [RESULT - FACTORIAL] - 120.0
Factorial of 5.0 is : 120.0
```

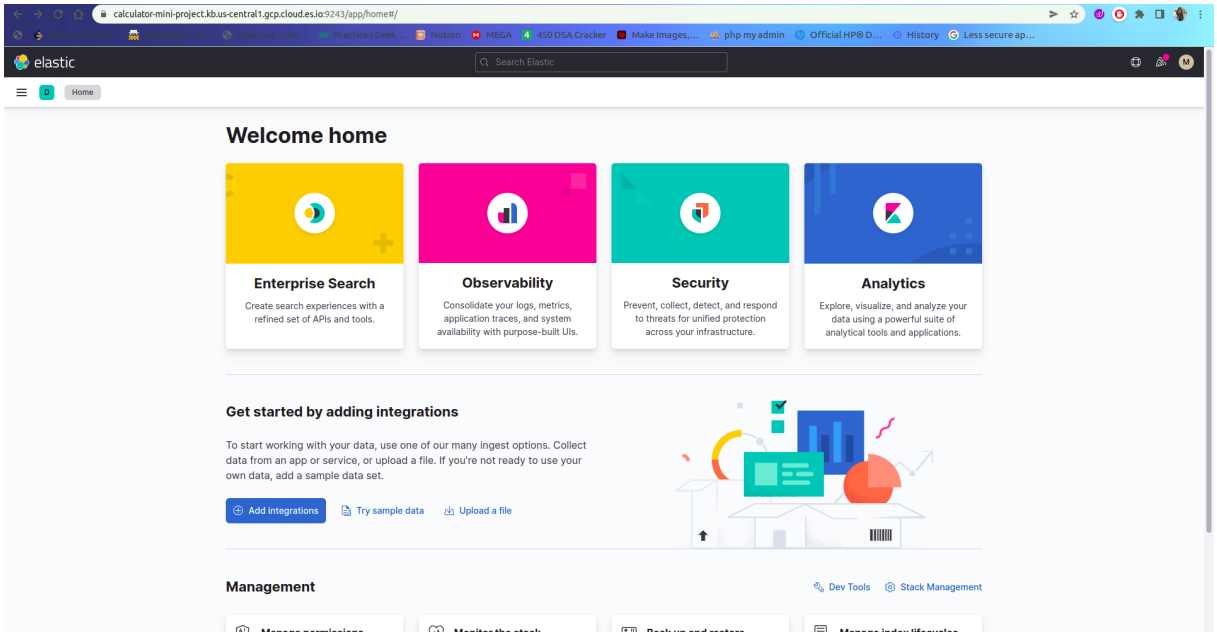
check the log file `cat calculator.log`

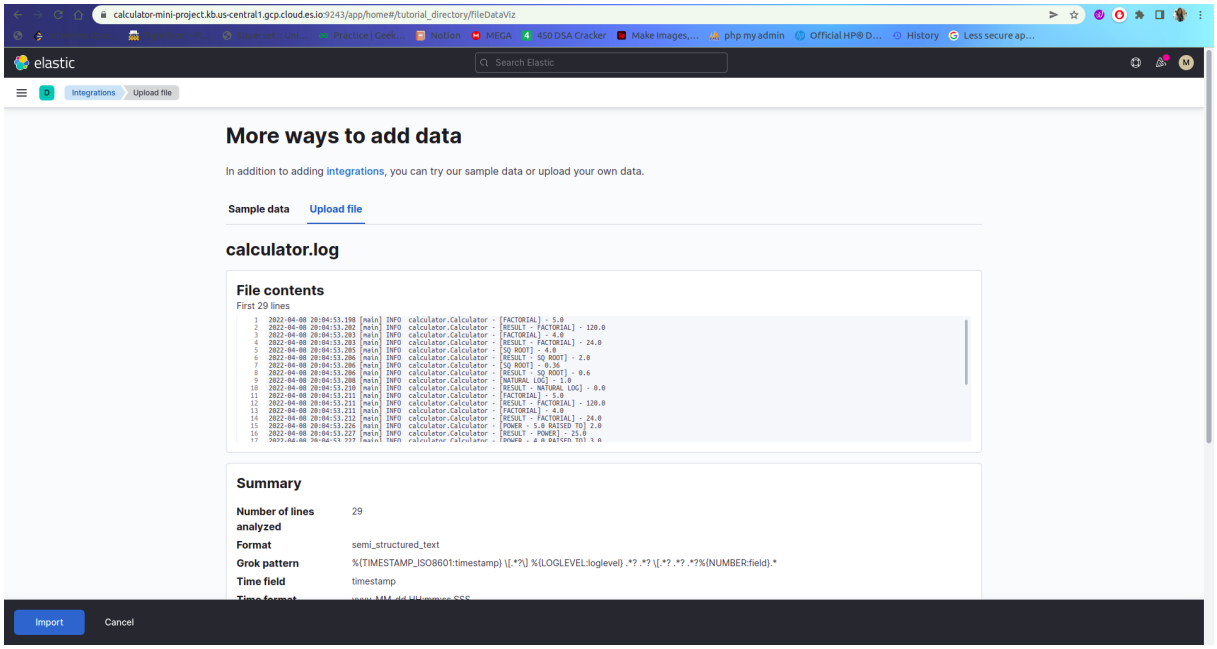
```
Scientific Calculator using DevOps.
 Choose operation:
1. Factorial
2. Square root
3. Power
4. Natural Logarithm
5. Exit
Enter your choice: 5
Exiting....
root@e5a9b46b3854:/# cat calculator.log
2022-04-15 10:44:11.737 [main] INFO  calculator.Calculator - [FACTORIAL] - 5.0
2022-04-15 10:44:11.741 [main] INFO  calculator.Calculator - [RESULT - FACTORIAL] - 120.0
2022-04-15 10:44:59.322 [main] INFO  calculator.Calculator - [FACTORIAL] - 5.0
2022-04-15 10:44:59.327 [main] INFO  calculator.Calculator - [RESULT - FACTORIAL] - 120.0
2022-04-15 10:45:01.405 [main] INFO  calculator.Calculator - [SQ ROOT] - 4.0
2022-04-15 10:45:01.406 [main] INFO  calculator.Calculator - [RESULT - SQ ROOT] - 2.0
2022-04-15 10:45:04.121 [main] INFO  calculator.Calculator - [POWER - 1.0 RAISED TO] 2.0
2022-04-15 10:45:04.122 [main] INFO  calculator.Calculator - [RESULT - POWER] - 1.0
root@e5a9b46b3854:/#
```

8. **ELK STACK:**

ELK Stack is a collection of three open-source products — Elasticsearch, Logstash, and Kibana. ELK stack provides centralized logging in order to identify problems with servers or applications. It allows you to search all the logs in a single place. It also helps to find issues in multiple servers by connecting logs during a specific time frame.
E stands for ElasticSearch: used for storing logs
L stands for LogStash : used for both shipping as well as processing and storing logs
K stands for Kibana: is a visualization tool (a web interface) which is hosted through Nginx or Apache.

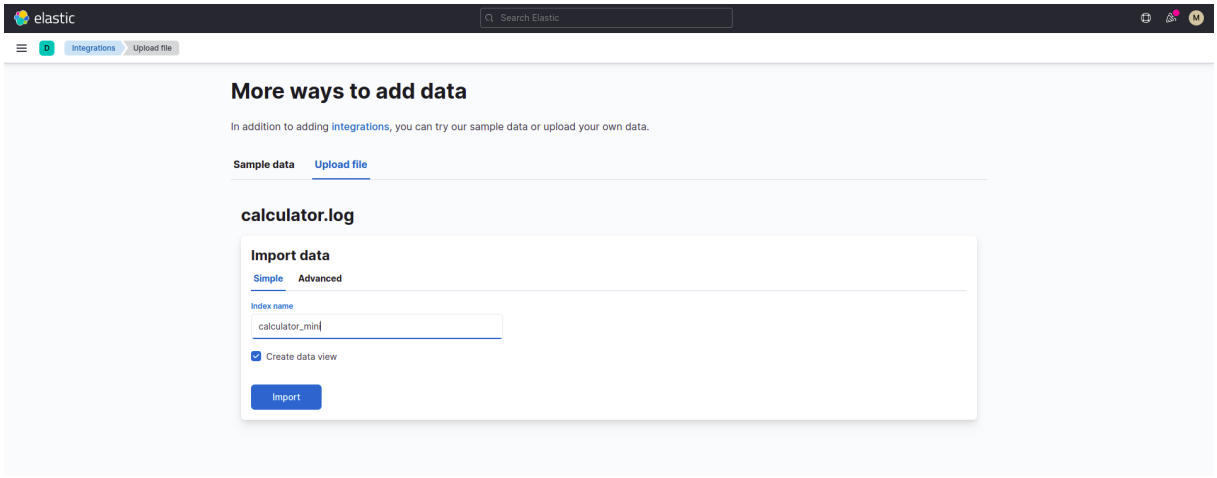
a. Upload your log file by clicking on Upload a file



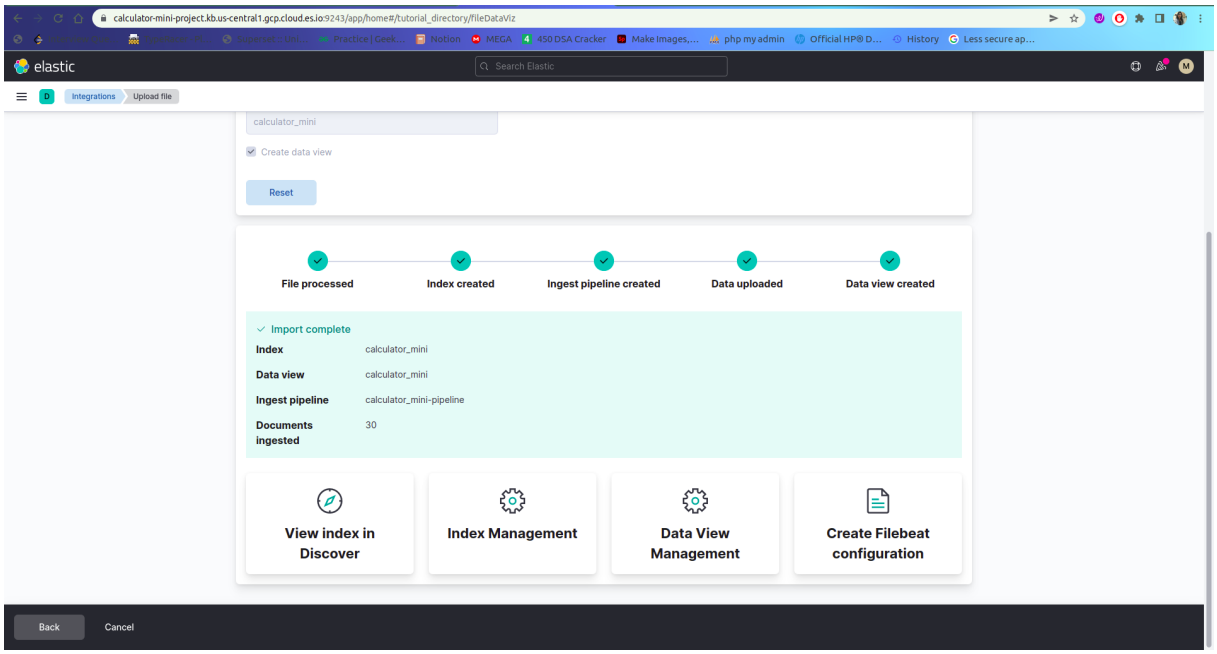


b. Click on Import.

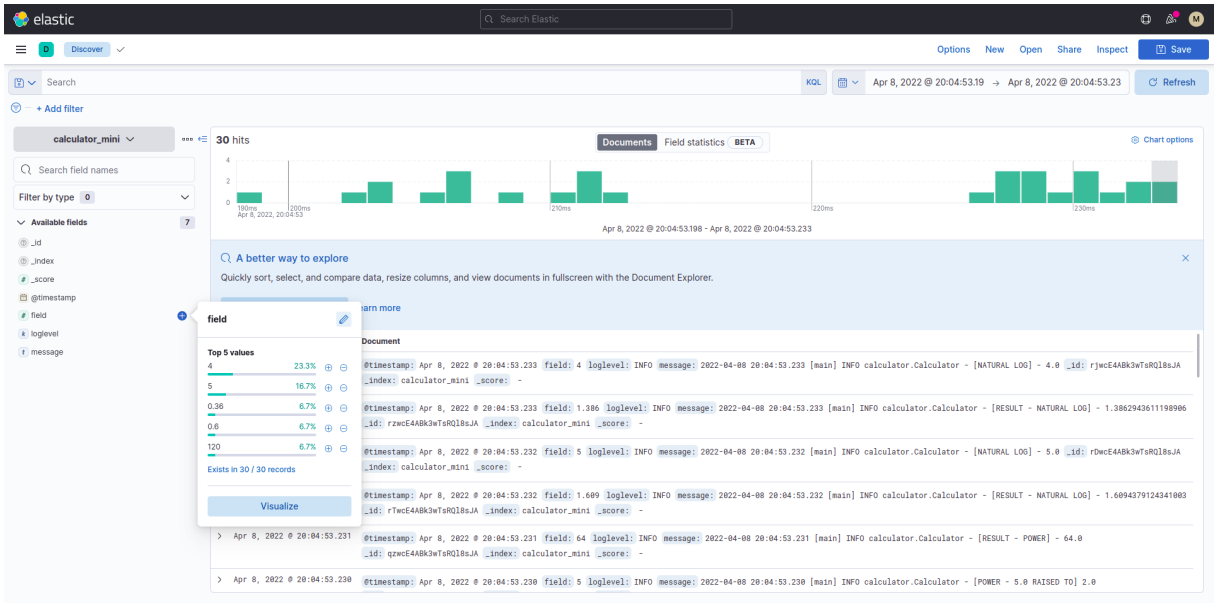
c. Add index name and click on import



d. Click on View index to discover



e. click on visualize

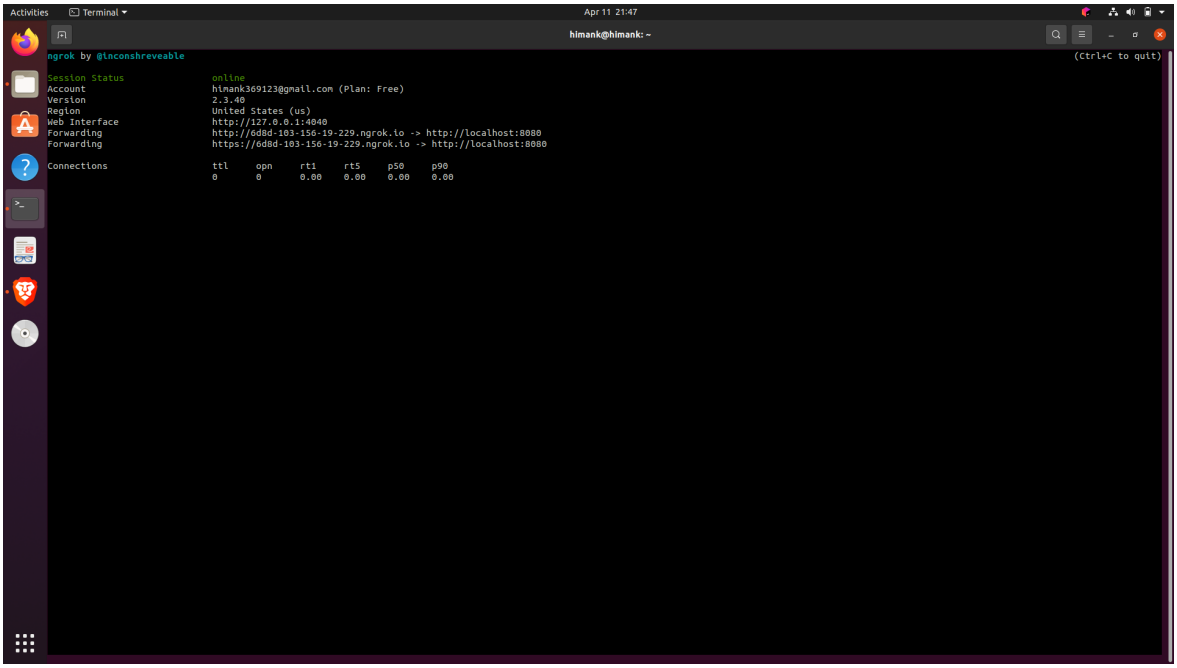


Using Github Webhook for automated builds:

Expose the jenkins address to the internet by making it public using ngrok.

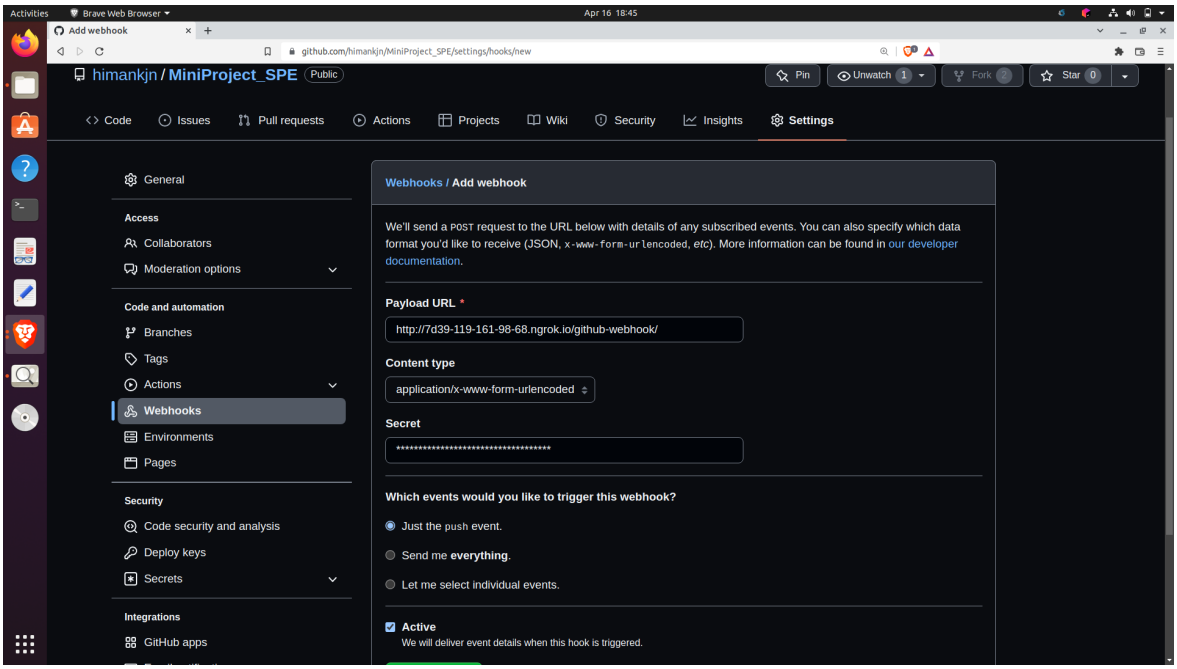
- Sign up in https://ngrok.com/
- Download ngrok from: https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz
- Extract ngrok from the terminal: $sudo tar xvzf ~/Downloads/ngrok-stable-linux-amd64.tgz -C /usr/local/bin
- Copy Authtoken from: https://dashboard.ngrok.com/get-started/your-authtoken
- Add Authtoken: $ngrok authtoken <token>
- Execute `ngrok http 172.16.134.225:8080;`

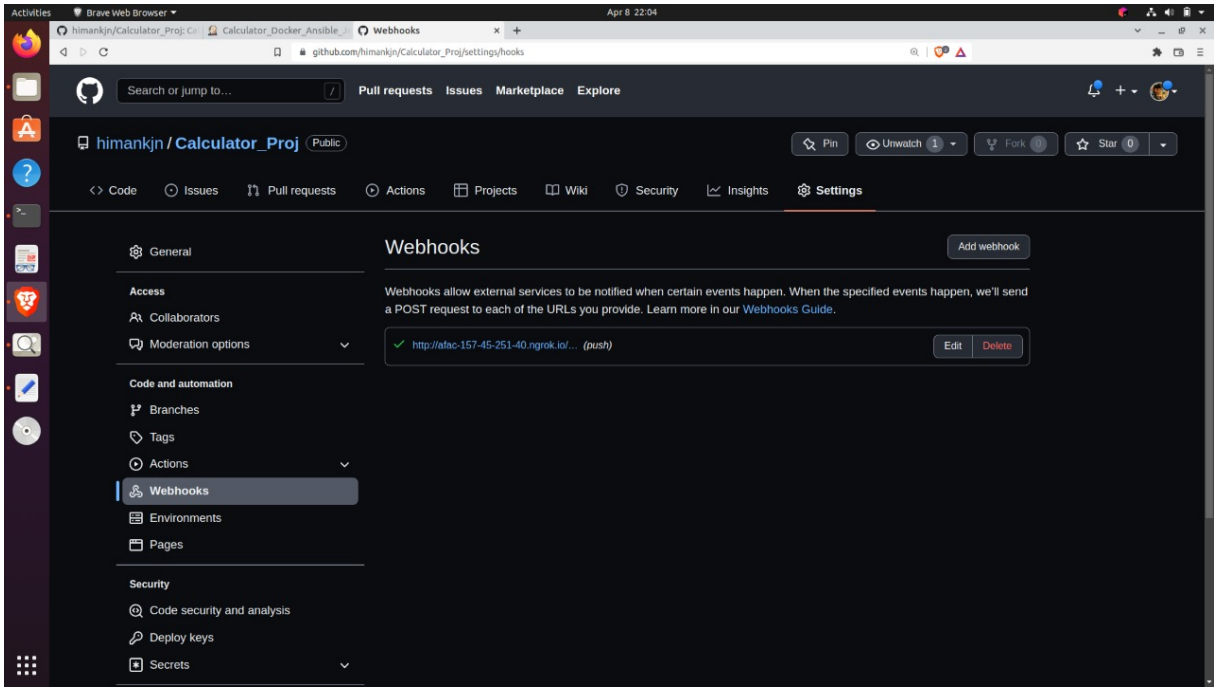
  This generates a public address that redirects to localhost:8080

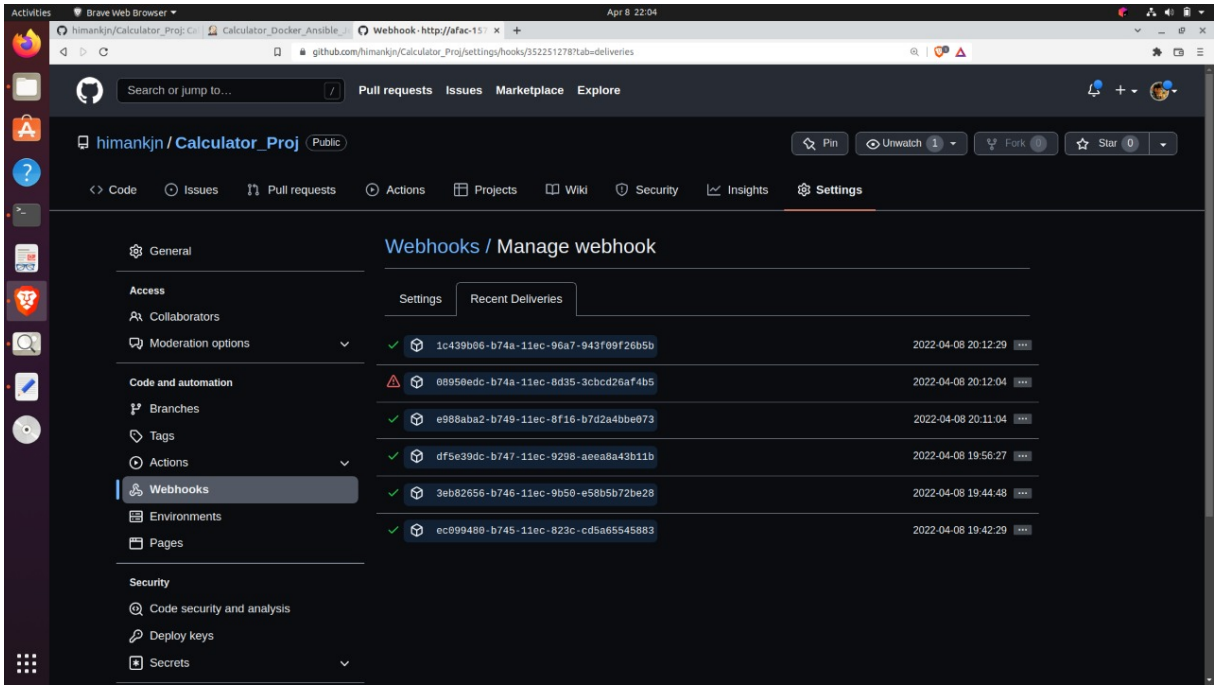


- Navigate to Github Project Repo → Settings → Webhooks:
  - Payload URL: ngrok_generated_IP/github-webbhook

    http://7d39-119-161-98-68.ngrok.io/github-webhook/
  - Secret: Personal Access Token

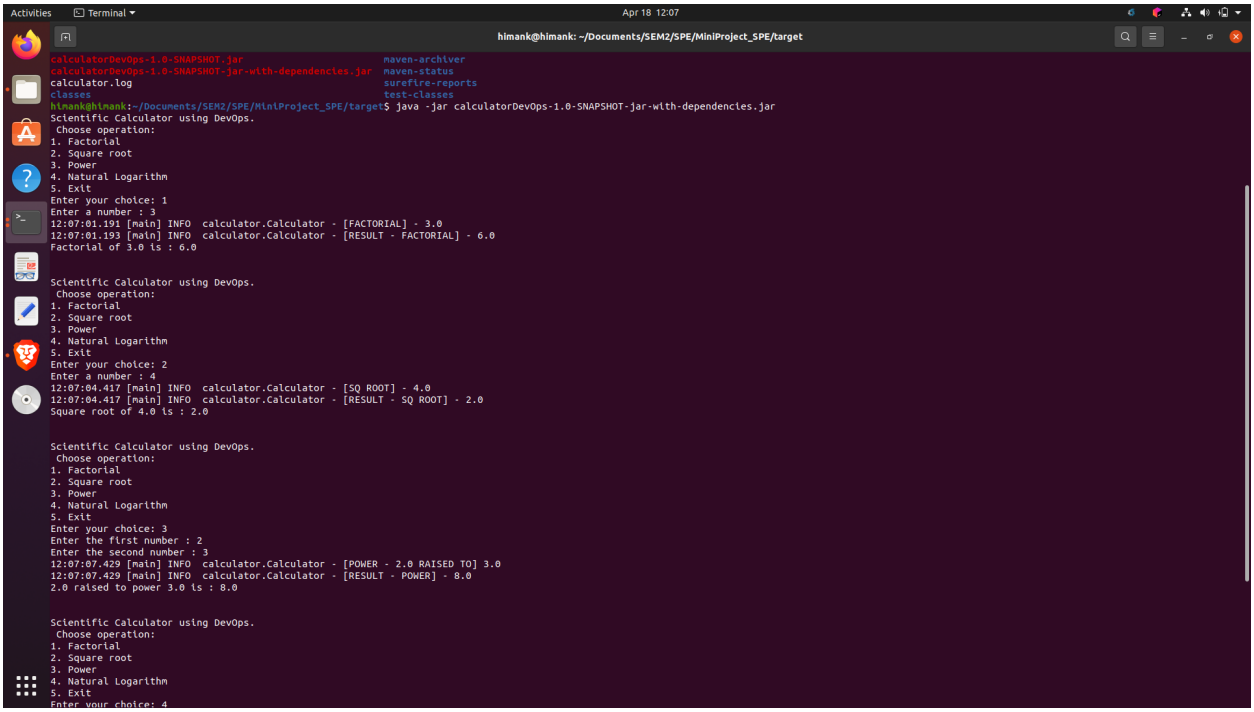


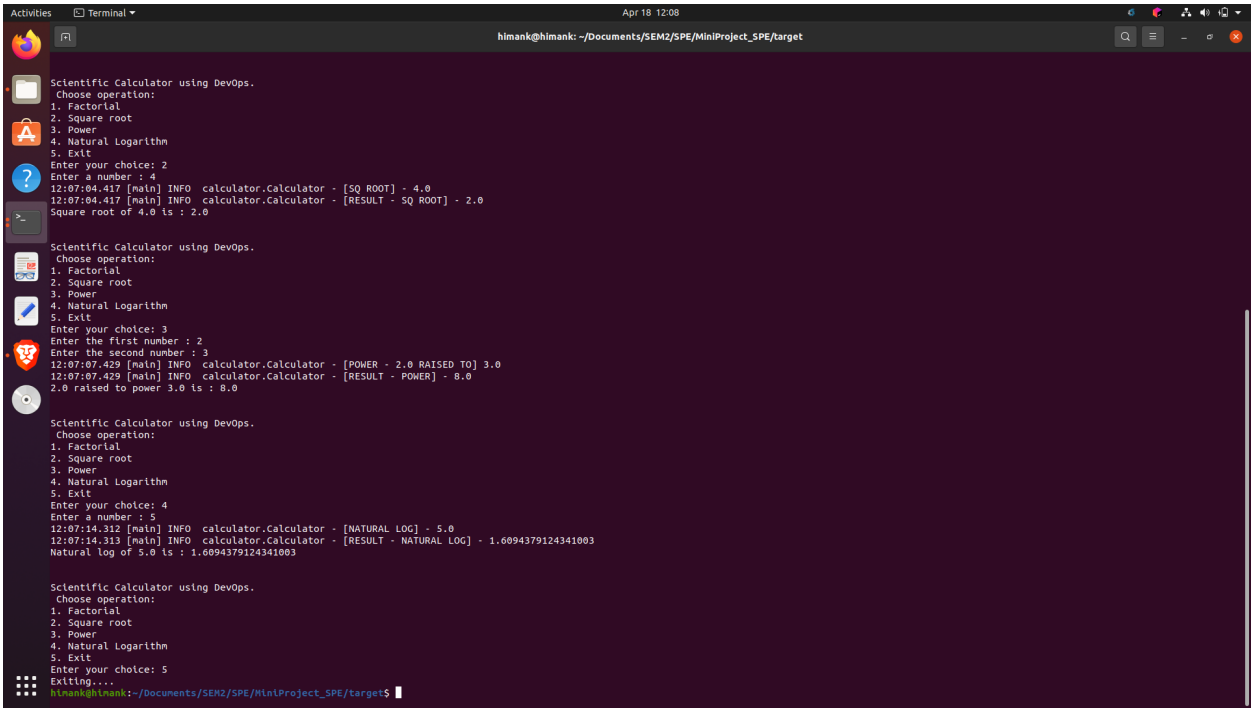




Application Screenshots:

The Calculator Application has a CLI interface and provides following options:

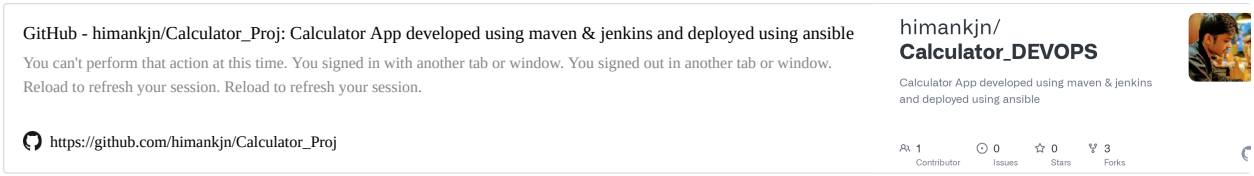




Summary:

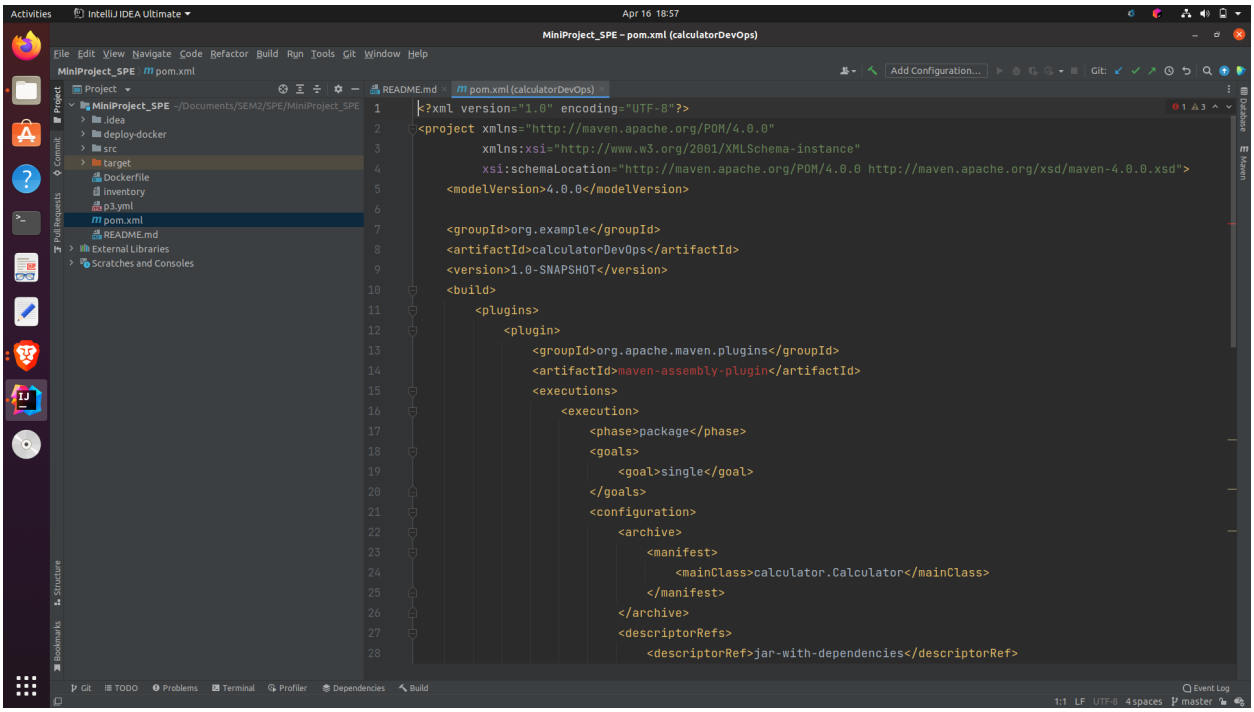
- Download and install required tools in host and server machines.
- Generate & Exchange ssh keys so that host and remote server can communicate without password.
- Create Maven Project containing mainly the following files:
  - **p3.yml:** Playbook for ansible remote deployment
  - **inventory:** file for ansible connection to remote server
  - **pom.xml:** Project Object Model i.e. maven file for dependencies.
  - **Dockerfile:** docker file containing instructions to build the image with final jar file and run it.
  - **src/main/java/calculator/Calculator.java:** Java file containing code for scientific calculator
  - **src/test/java/CalculatorTest.java:** Java file containing testing code for scientific calculator
- Push your code to **Github**
- Create **DockerHub** Repository
- Create Jenkins Pipeline Project, write the pipeline script containing the following steps:
  1. **Git** Pull
  2. **Maven** build
  3. **Docker** Image creation
  4. Pushing Image to **Docker Hub**
  5. **Ansible** Deploy
- Setup Jenkins Configuration for **Github** Webhook using **ngrok.**
- Build the project.
- Run the docker image as a container in remote server.

- Upload the generated log file in **ELK Stack.**



Dependencies & Code:

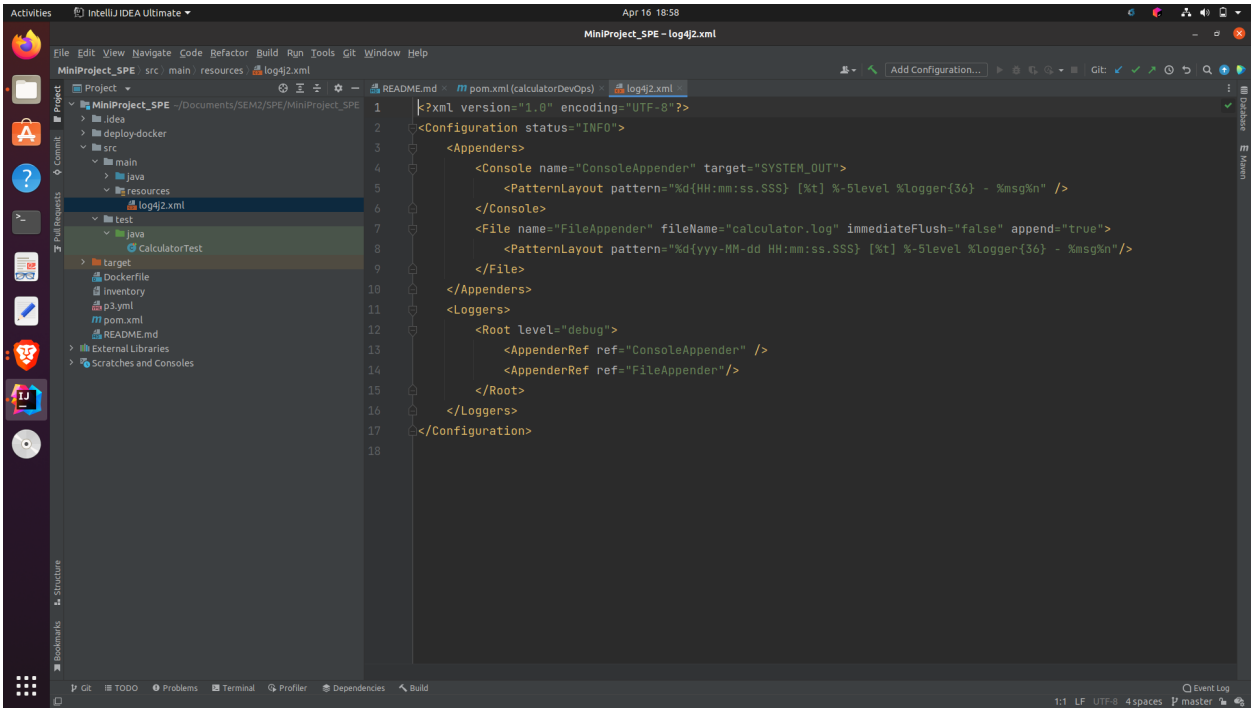
Dependencies in the code can be resolved using Maven. Dependencies are the external libraries which we use to integrate it within our code. All of the dependencies can be searched from https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-dependency-plugin and these can be included in POM.xml. Maven will automatically download the dependencies and will also indicate if there are any more requirements.



Log4j2.xml:

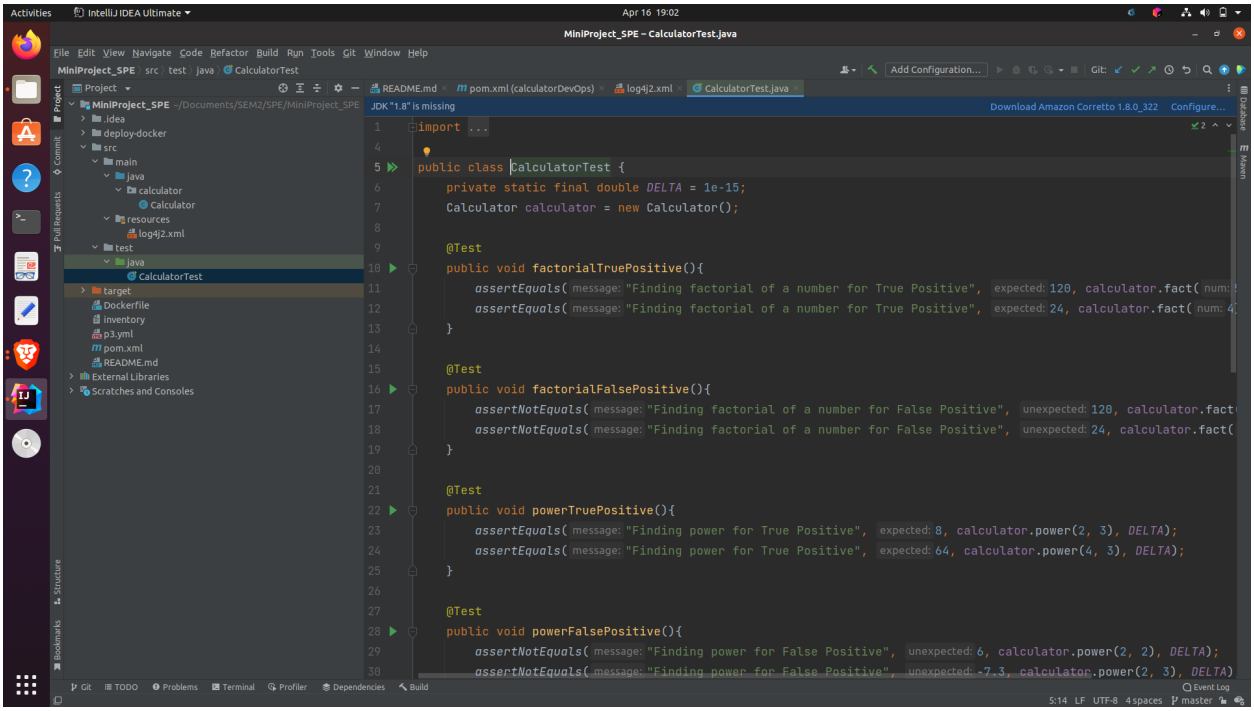
We have added lo4j2.xml which will create a log file for test cases. For creation of the same, some build properties also have to be added in pom.xml.

The log4j2.properties handles projects log functionalities.



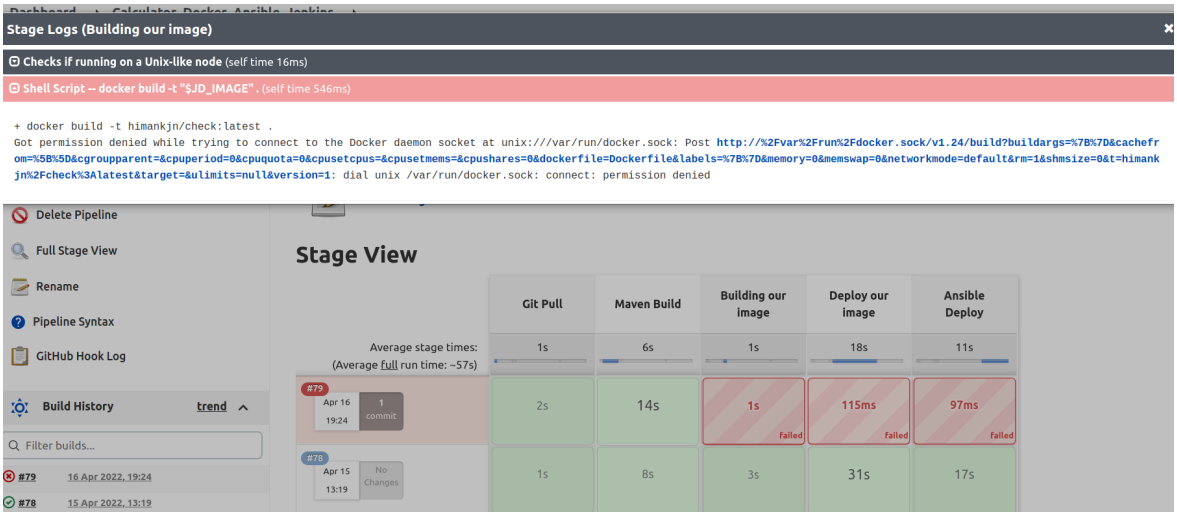
Unit Testing:

For unit testing Junit dependencies have been added and test cases have been written for false as well as true assertions.

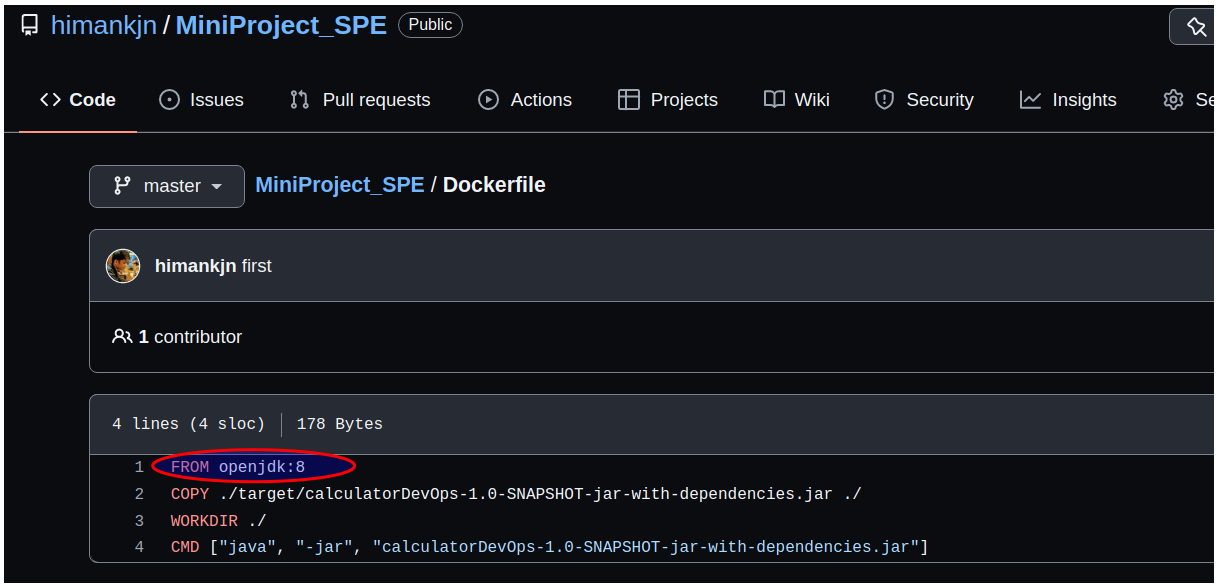


CHALLENGES AND SOLUTIONS:

1. Docker Daemon Permission
   - **Error**: permission denied while trying to connect to Docker daemon socket at unix:///var/run/docker.sock
   - **Solution**: `sudo chmod 666 varrun/docker.sock` on both server and local machine



2. JDK version mismatch
   - **Error**: Image creation using java 8 base image but installing java 11 in remote server
   - **Solution**: changing the base image in DockerFile from openjdk:8 to openjdk:11



3. Docker Hub default image tag
   - **Error**: The default image tag being pulled by remote server was :latest but the one we pushed through pipeline was the :$BUILD_NUMBER

- **Solution**: Changing the image tag in pipeline to :latest

4. pip and pip-ansible in remote server:
    - **Error**: remote server was unable to pull the image using ansible and suggested to install ansible through pip
    - **Solution:** Install python-pip and pip install ansible
5. Webhook:
    - **Error:** ngrok webhook through github threw 404 error.
    - **Solution**: Ensure that ngrok is running on localhost ip:8080
6. Remote Host Connectivity:
    - **Error**: Ansible was unable to connect to remote server using ssh.
    - **Solution**: Ensure SSH keys generated and copied to remote serverusing ssh-copy-id
7. Virtual Machines IP address:
    - **Error**: Virtual Box assigned same IP to both machines.
    - **Solution**: Use bridge adapter in network settings of vitual box to ensure different ip addresses