Scientific Calculator with DevOps

Submitted by Aditya Vardhan - IMT2019003

Links

Github Repo:- https://github.com/vardhan2000/scientificCalculator

Docker Repo:- https://hub.docker.com/repository/docker/av2000/scientific-calculator/general

Problem Statement

Creating a calculator with the following operations:

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

The primary goal of the project is to become familiar with the CI/CD idea of DevOps, which is accomplished by building a Jenkins pipeline.

DevOps

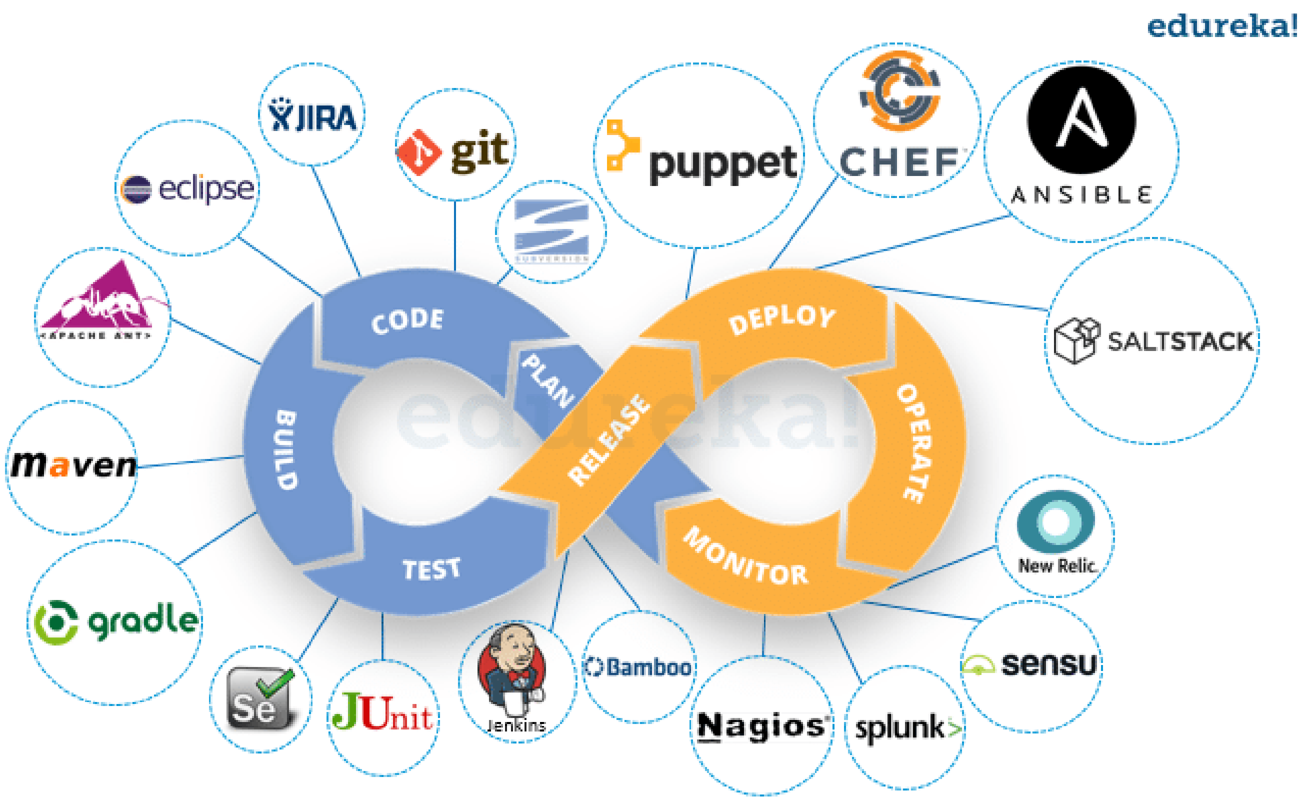
➢ What is DevOps?

  ○ DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to improve the speed and quality of software delivery. The goal of DevOps is to break down the traditional silos between development and operations teams and create a more collaborative and agile environment.

- DevOps practices include continuous integration and continuous delivery (CI/CD), infrastructure as code (IaC), monitoring and logging, and automated testing. By automating and streamlining these processes, DevOps teams can reduce the time and effort required to deliver software, while also improving the reliability and quality of the software being delivered.

- DevOps also emphasises a culture of collaboration, transparency, and continuous improvement. This involves creating cross-functional teams that include developers, operations engineers, and other stakeholders, and encouraging open communication and feedback. DevOps teams also embrace a "fail fast, learn fast" mentality, which involves testing and iterating on software quickly and continuously in order to identify and fix issues before they become major problems.

- Overall, DevOps is a key enabler of digital transformation and has become increasingly important in today's fast-paced, cloud-based software development landscape. By embracing DevOps practices, organisations can improve their agility, reduce their time to market, and deliver higher-quality software that meets the needs of their customers.



➢ Why DevOps?

- DevOps should be used because it offers several benefits to organizations that can help them to become more agile, efficient, and competitive in today's fast-paced digital landscape. Here are some of the key reasons why DevOps should be used:

    a. Faster Time-to-Market: By automating and streamlining software delivery processes, DevOps enables organizations to release software faster, more frequently, and with fewer errors. This can give organizations a competitive edge by allowing them to deliver new features and functionality to customers more quickly.

    b. Improved Collaboration: DevOps promotes a culture of collaboration between development and operations teams, which helps to break down silos and improve communication. This can lead to better alignment between different parts of the organisation, which can result in faster and more efficient problem-solving.

    c. Increased Efficiency: DevOps practices like automation, continuous integration, and continuous delivery can help organisations to reduce the time and effort required to deliver software. This can lead to greater efficiency and productivity, and can free up resources to focus on other areas of the business.

    d. Improved Quality: By emphasising automated testing, monitoring, and logging, DevOps helps organisations to identify and fix issues more quickly, before they become major problems. This can lead to higher-quality software that meets the needs of customers more effectively.

    e. Greater Agility: DevOps enables organisations to respond more quickly to changing market conditions and customer needs. By automating and streamlining software delivery processes, organisations can become more agile and responsive, which can help them to stay ahead of the competition.

Tools Used

1. **Maven:** a build automation tool used primarily for Java projects that manages dependencies, builds, and releases.
2. **Git:** a source code management tool that allows developers to collaborate on code and track changes.
3. **Github:** a web-based platform for software development that provides tools for version control, collaboration, and project management.
4. **Jenkins:** an open-source automation server that helps to automate parts of the software development process, such as building, testing, and deploying code. It is widely used for continuous integration and delivery (CI/CD) and can be integrated with a variety of other tools and technologies, such as Git, Docker, and AWS. Jenkins allows developers to automate repetitive tasks and streamline their workflows, which can improve efficiency and reduce errors in the software development process.
5. **Github webhooks:** detects commits and automatically triggers the pipeline stages on every commit.
6. **Ngrok:** a tool that provides secure tunnels to localhost environments, allowing developers to expose web servers running on their local machines to the internet. It creates a public URL that can be used to access a local server, making it easier to test webhooks, APIs, and other web applications without deploying them to a public server.
7. **Docker:** a platform that enables developers to automate the deployment of applications inside containers, providing an isolated environment for running software.
8. **Ansible:** an open-source automation tool that simplifies IT configuration management and application deployment across servers, networks, and cloud environments using a simple, human-readable language.
9. **Elastic search:** a distributed search and analytics engine that allows users to store, search, and analyse large volumes of data in real-time. It is commonly used for log analytics, application monitoring, and full-text search scenarios.
10. **Kibana:** an open-source data visualisation and exploration platform that is used with Elasticsearch to provide a user interface for analysing data. It allows users to create and share visualisations, dashboards, and reports based on data stored in Elasticsearch.

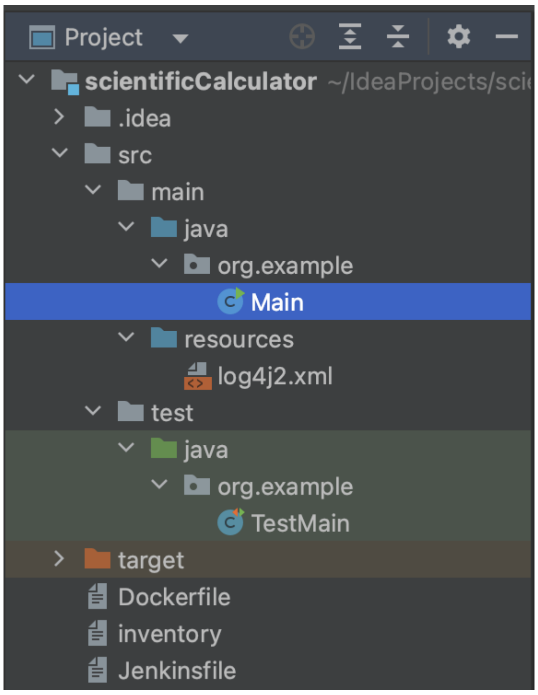
Steps

1. Make sure to have a Java-11/Java-17 on your system as Jenkins requires Java 11 or 17 since Jenkins 2.357 and LTS 2.361. 1. I used Java-17.
2. Make a maven project and write code for your calculator (CLI) app.
3. Write your test cases using a unit testing framework like JUnit.
4. Make a log file and log your app’s inputs, outputs and exceptions in it.

5. Push your code on Github.
6. Write Pipeline Script in Jenkins for the following:
    a. Git Pull
    b. Maven build
    c. Build a docker image.
    d. Push docker image to Docker Hub
    e. Ansible Deploy
7. Pull the image.
8. Run the image.
9. You can use ELK stack to generate reports for your log file and use it to monitor your app.

Development, Software Build, and Test

- The code is written using Java-17 in the IntelliJ-IDE development environment. Unit testing is done using JUnit, while log tracking is done using Log4j2.

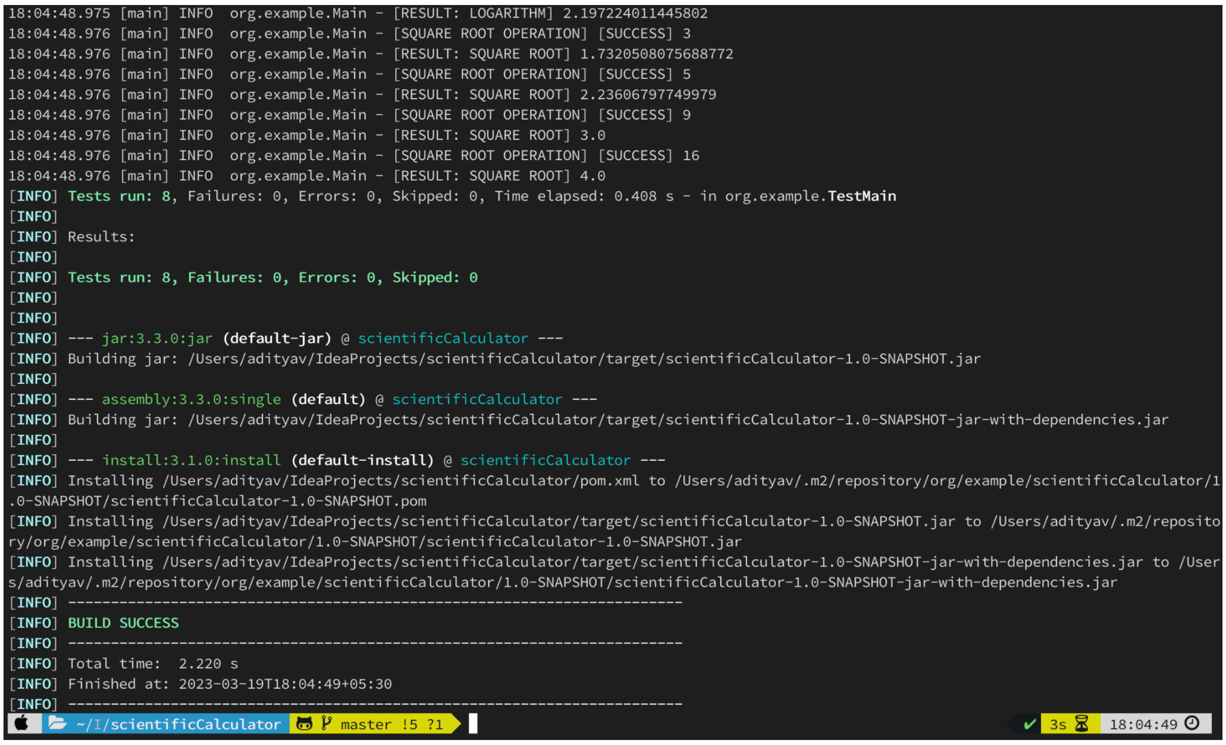


```
Project
scientificCalculator ~/IdeaProjects/sci
  .idea
  src
    main
      java
        org.example
          Main
      resources
        log4j2.xml
    test
      java
        org.example
          TestMain
  target
  Dockerfile
  inventory
  Jenkinsfile
```

- **Main.java** contains the main code of the project. which contains the following functions:
    - root(): computes the square root of a given number
    - factorial(): computes the factorial of a given number
    - ln(): computes the natural log of a given number
    - power(): computes the power of a given number

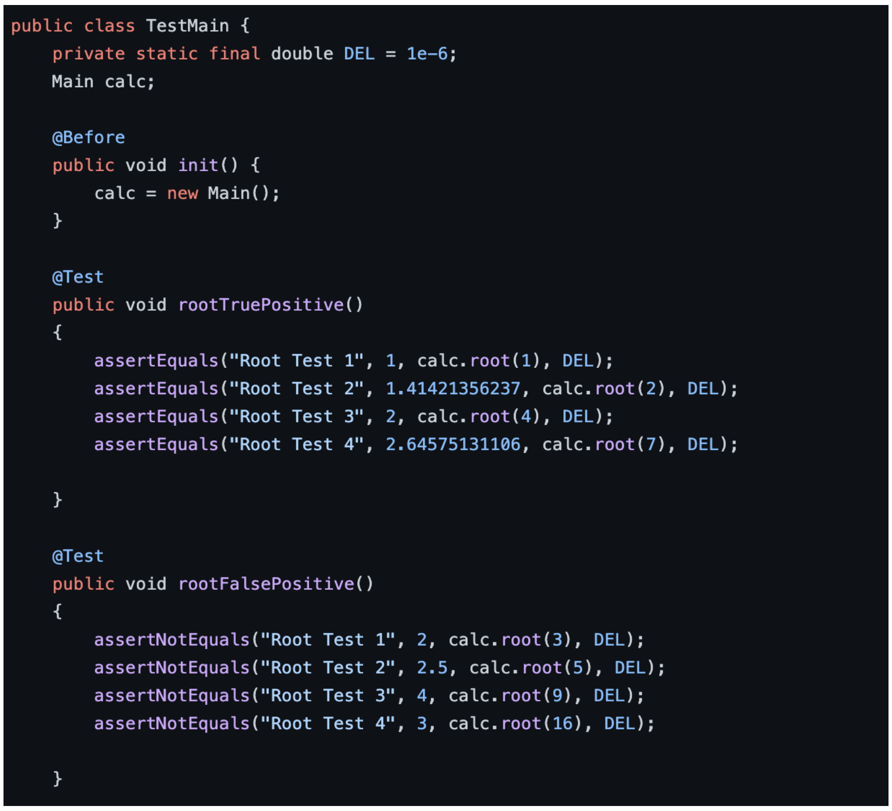
- **TestMain.java** contains true and false positive test cases used to test the code when we build the project. It is performed using JUnit.

Output:



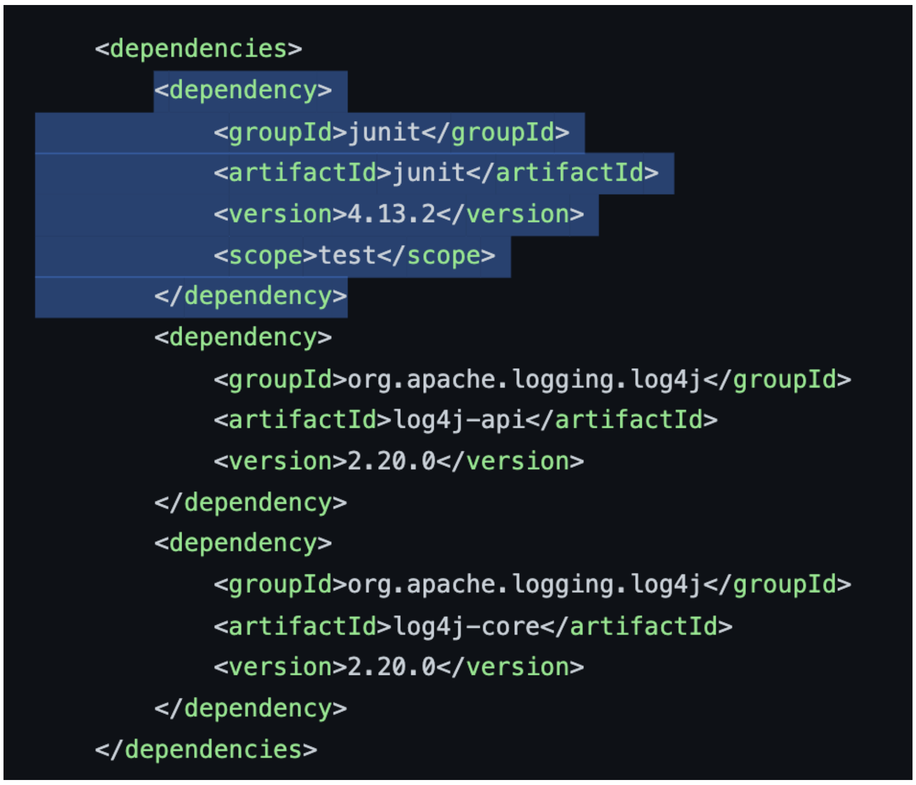
```
18:04:48.975 [main] INFO  org.example.Main - [RESULT: LOGARITHM] 2.1972240114458O2
18:04:48.976 [main] INFO  org.example.Main - [SQUARE ROOT OPERATION] [SUCCESS] 3
18:04:48.976 [main] INFO  org.example.Main - [RESULT: SQUARE ROOT] 1.7320508075688772
18:04:48.976 [main] INFO  org.example.Main - [SQUARE ROOT OPERATION] [SUCCESS] 5
18:04:48.976 [main] INFO  org.example.Main - [RESULT: SQUARE ROOT] 2.23606797749979
18:04:48.976 [main] INFO  org.example.Main - [SQUARE ROOT OPERATION] [SUCCESS] 9
18:04:48.976 [main] INFO  org.example.Main - [RESULT: SQUARE ROOT] 3.0
18:04:48.976 [main] INFO  org.example.Main - [SQUARE ROOT OPERATION] [SUCCESS] 16
18:04:48.976 [main] INFO  org.example.Main - [RESULT: SQUARE ROOT] 4.0
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.408 s - in org.example.TestMain
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ scientificCalculator ---
[INFO] Building jar: /Users/adityav/IdeaProjects/scientificCalculator/target/scientificCalculator-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- assembly:3.3.0:single (default) @ scientificCalculator ---
[INFO] Building jar: /Users/adityav/IdeaProjects/scientificCalculator/target/scientificCalculator-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO]
[INFO] --- install:3.1.0:install (default-install) @ scientificCalculator ---
[INFO] Installing /Users/adityav/IdeaProjects/scientificCalculator/pom.xml to /Users/adityav/.m2/repository/org/example/scientificCalculator/1
.0-SNAPSHOT/scientificCalculator-1.0-SNAPSHOT.pom
[INFO] Installing /Users/adityav/IdeaProjects/scientificCalculator/target/scientificCalculator-1.0-SNAPSHOT.jar to /Users/adityav/.m2/reposito
ry/org/example/scientificCalculator/1.0-SNAPSHOT/scientificCalculator-1.0-SNAPSHOT.jar
[INFO] Installing /Users/adityav/IdeaProjects/scientificCalculator/target/scientificCalculator-1.0-SNAPSHOT-jar-with-dependencies.jar to /User
s/adityav/.m2/repository/org/example/scientificCalculator/1.0-SNAPSHOT/scientificCalculator-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.220 s
[INFO] Finished at: 2023-03-19T18:04:49+05:30
[INFO] ------------------------------------------------------------------------
 ~/I/scientificCalculator   master !5 ?1                                                        ✔  3s   18:04:49
```

- **Test-Cases**: A True Positive test case and a False Positive test case are both utilised for every functionality.



```java
public class TestMain {
    private static final double DEL = 1e-6;
    Main calc;

    @Before
    public void init() {
        calc = new Main();
    }

    @Test
    public void rootTruePositive()
    {
        assertEquals("Root Test 1", 1, calc.root(1), DEL);
        assertEquals("Root Test 2", 1.41421356237, calc.root(2), DEL);
        assertEquals("Root Test 3", 2, calc.root(4), DEL);
        assertEquals("Root Test 4", 2.64575131106, calc.root(7), DEL);

    }

    @Test
    public void rootFalsePositive()
    {
        assertNotEquals("Root Test 1", 2, calc.root(3), DEL);
        assertNotEquals("Root Test 2", 2.5, calc.root(5), DEL);
        assertNotEquals("Root Test 3", 4, calc.root(9), DEL);
        assertNotEquals("Root Test 4", 3, calc.root(16), DEL);

    }
```

- **Project Dependencies**: We must include particular jar files in the pom.xml file in order to use JUnit and log4j2. Maven will thus include those dependencies.



```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.20.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.20.0</version>
    </dependency>
</dependencies>
```

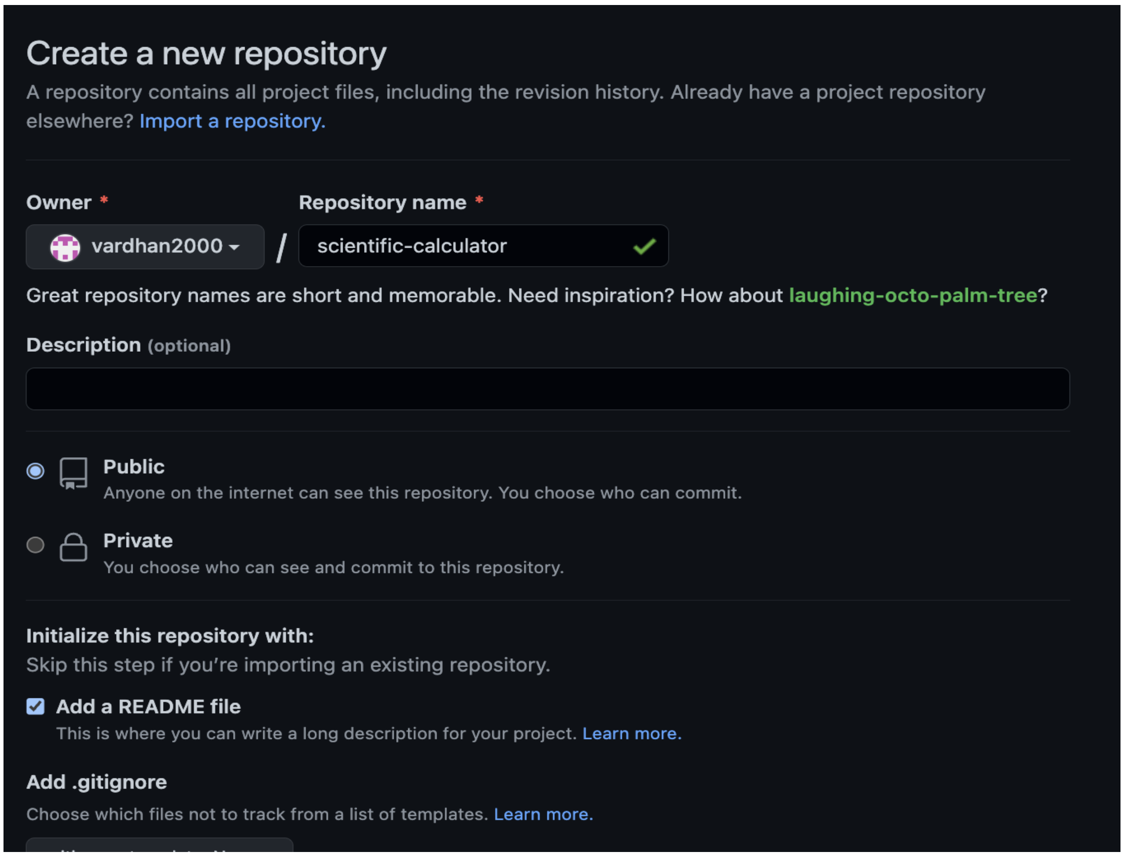
- In order to proceed, type **$ mvn clean install**. This will build the whole project, run all tests, and create a new subdirectory called "target" in the current directory where **.jar** files will be generated.

Source Code Management - GitHub

➢ SCM is the process of managing and tracking changes to source code over time. SCM tools allow developers to collaborate on code changes, track revisions, and manage code versions. It is a critical part of software development, enabling teams to work together effectively, manage code quality, and ensure that changes are tracked and documented. In our project we use Github as our SCM tool.

➢ To begin, create a new repository at https://github.com. By giving a brand-new repository a special name associated with the user, we are able to create it. Our code will be managed by the SCM, which will be connected to Jenkins as an input.

- **Steps:**
    - Create a public repository.
    - Open terminal in your project folder and enter the following commands:
        i. $ git init
        ii. $ git add .
        iii. $ git remote add origin <github repo URL>
        iv. $ git commit -m “<commit message>”
        v. $ git remote add origin < GitHub repo URL >
        vi. $ git push


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * vardhan2000 / Repository name * scientific-calculator

Great repository names are short and memorable. Need inspiration? How about laughing-octo-palm-tree?

Description (optional)

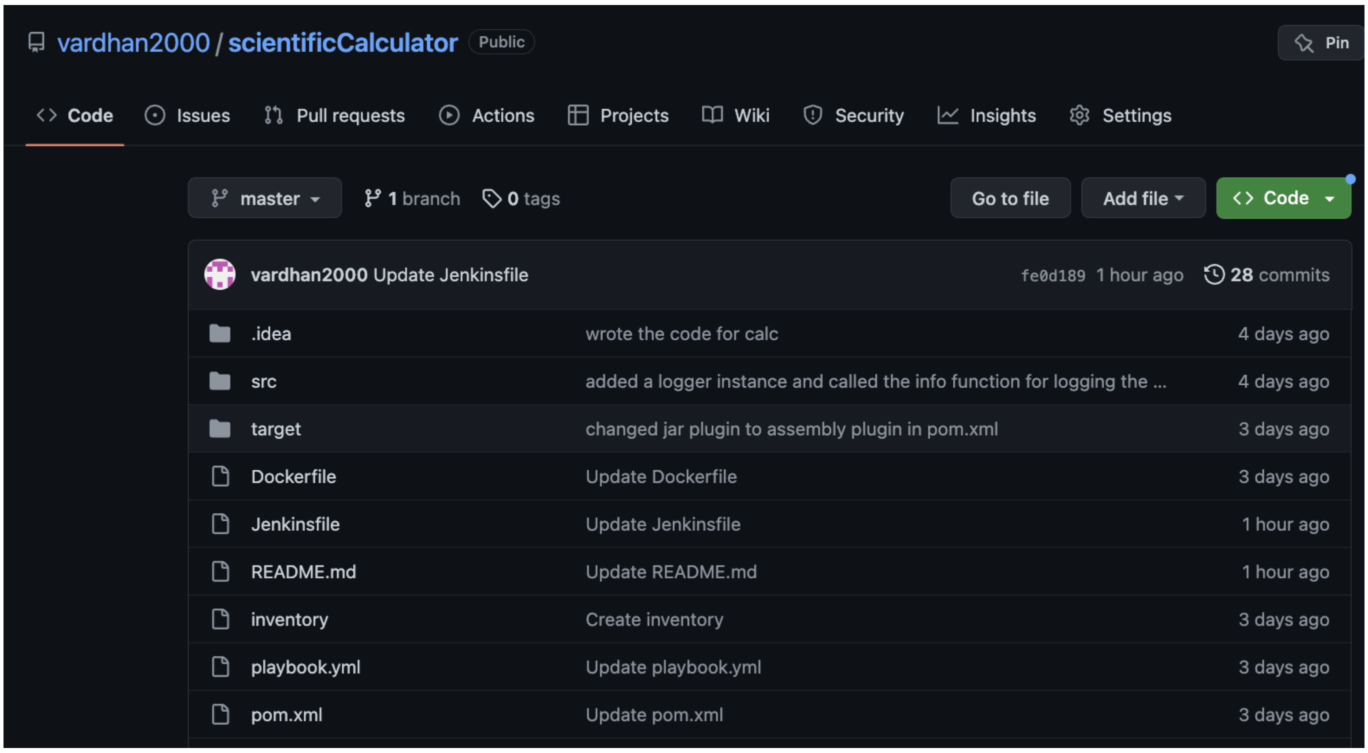
[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

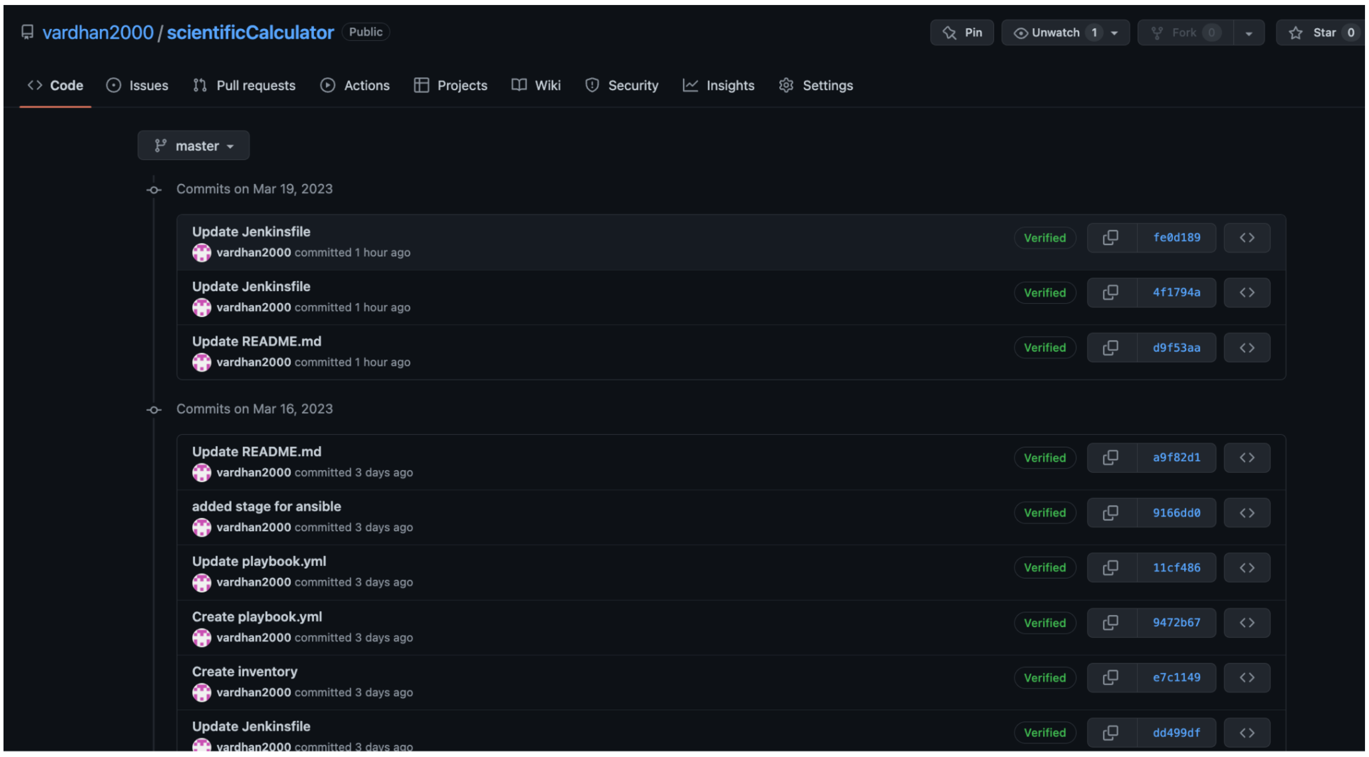
[✓ Add a README file] This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.


vardhan2000 / scientificCalculator Public

vardhan2000 Update Jenkinsfile fe0d189 1 hour ago 28 commits

.idea	wrote the code for calc	4 days ago
src	added a logger instance and called the info function for logging the ...	4 days ago
target	changed jar plugin to assembly plugin in pom.xml	3 days ago
Dockerfile	Update Dockerfile	3 days ago
Jenkinsfile	Update Jenkinsfile	1 hour ago
README.md	Update README.md	1 hour ago
inventory	Create inventory	3 days ago
playbook.yml	Update playbook.yml	3 days ago
pom.xml	Update pom.xml	3 days ago



Commits on Mar 19, 2023

Update Jenkinsfile
vardhan2000 committed 1 hour ago — Verified fe0d189

Update Jenkinsfile
vardhan2000 committed 1 hour ago — Verified 4f1794a

Update README.md
vardhan2000 committed 1 hour ago — Verified d9f53aa

Commits on Mar 16, 2023

Update README.md
vardhan2000 committed 3 days ago — Verified a9f82d1

added stage for ansible
vardhan2000 committed 3 days ago — Verified 9166dd0

Update playbook.yml
vardhan2000 committed 3 days ago — Verified 11cf486

Create playbook.yml
vardhan2000 committed 3 days ago — Verified 9472b67

Create inventory
vardhan2000 committed 3 days ago — Verified e7c1149

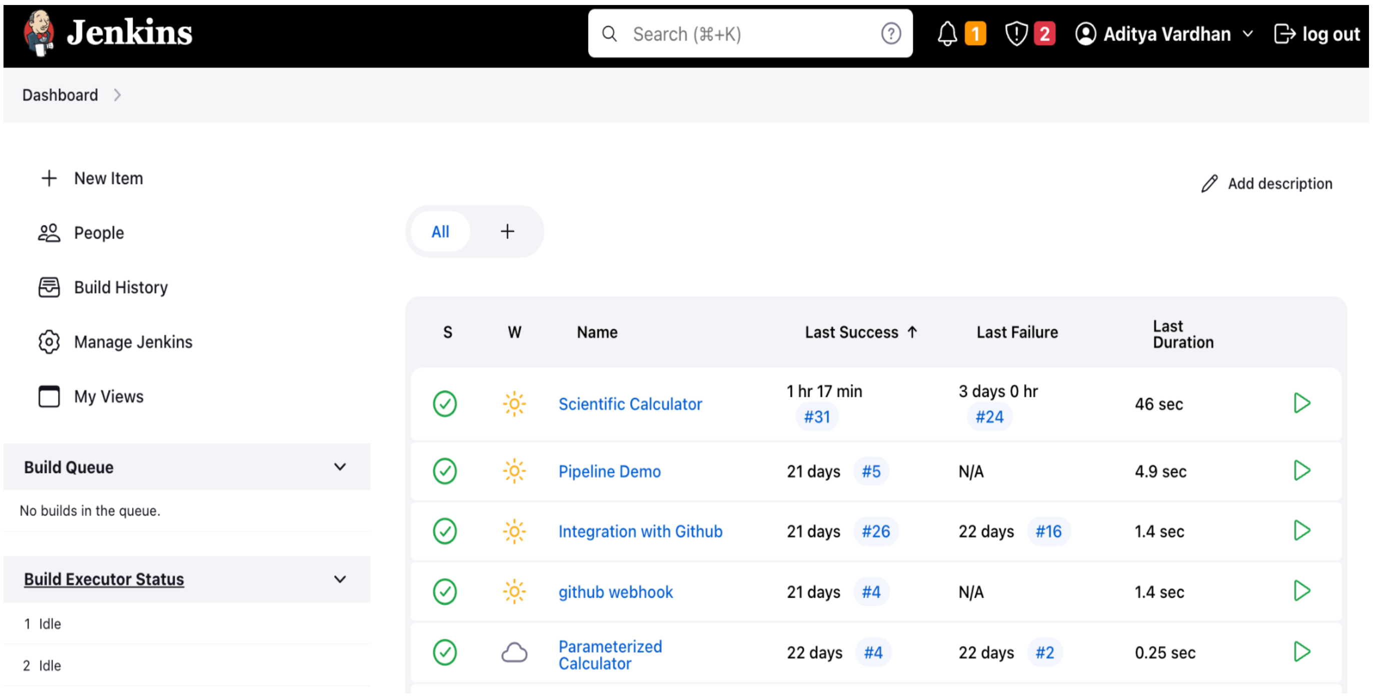
Update Jenkinsfile
vardhan2000 committed 3 days ago — Verified dd499df

Jenkins

➢ Jenkins is a Java-based Continuous Integration (CI) plugin-based open-source automation framework. It is simpler for developers to incorporate changes and for consumers to acquire a new build thanks to Jenkins, which is used to regularly generate and test software projects.

➢ In this project, continuous delivery, or till delivery, was handled via the Jenkins pipeline. The Jenkins service may be accessed via **http://localhost:8080**. Open a web browser and enter this URL into the address bar to access it.

➢ Setup steps: I used a macos and following are the steps to set up jenkins. If you are using any other OS, you can refer to jenkins documentation for the same.
  - Make sure you have a homebrew package manager installed on your system.
  - Run **$ brew install jenkins-lts** to install jenkins.
  - To start jenkins service, run: **$ brew services start jenkins-lts** and browse to the above mentioned URL.



Build Queue

No builds in the queue.

Build Executor Status

1 Idle

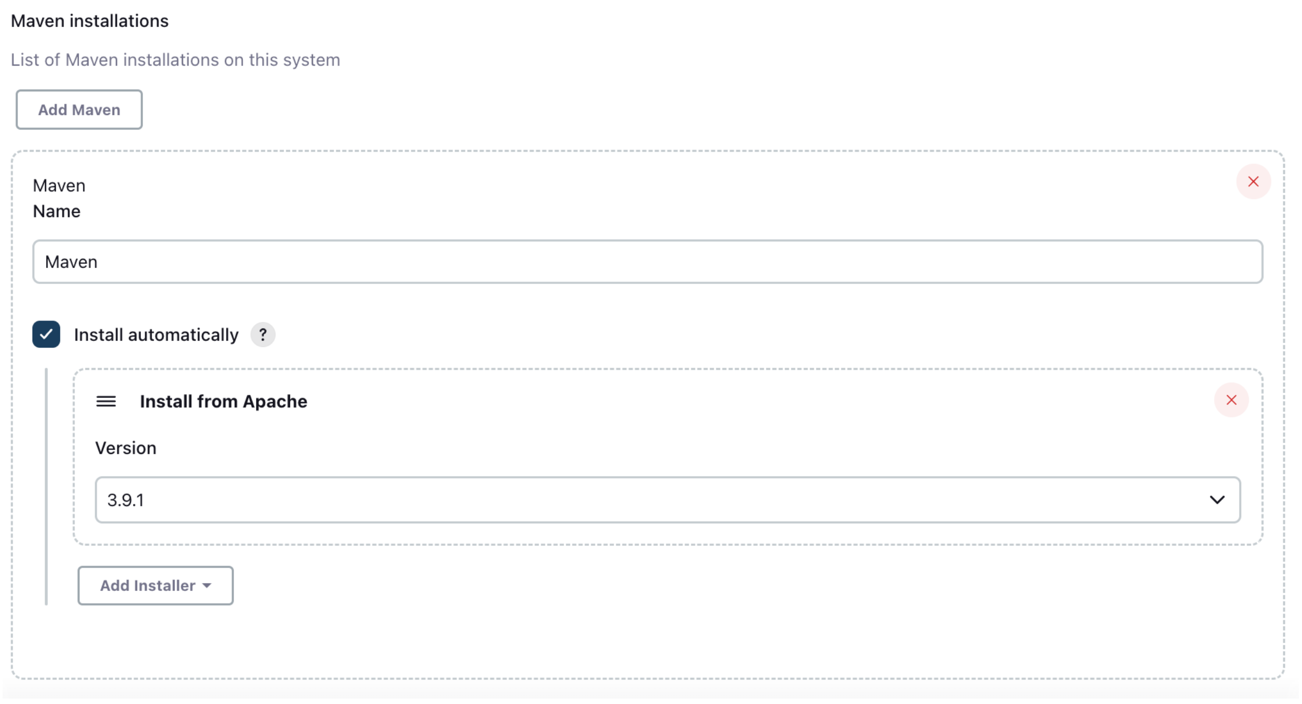
2 Idle

S	W	Name	Last Success ↑	Last Failure	Last Duration
✓	☀	Scientific Calculator	1 hr 17 min #31	3 days 0 hr #24	46 sec
✓	☀	Pipeline Demo	21 days #5	N/A	4.9 sec
✓	☀	Integration with Github	21 days #26	22 days #16	1.4 sec
✓	☀	github webhook	21 days #4	N/A	1.4 sec
✓	☁	Parameterized Calculator	22 days #4	22 days #2	0.25 sec

- ➢ Prerequisites for Jenkins:
  - ○ We will need a variety of plugins to utilise Jenkins Pipeline.

1. Install the following from Plugin Manager:
   i. Maven
   ii. Git
   iii. Ansible
   iv. Docker

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



Maven installations

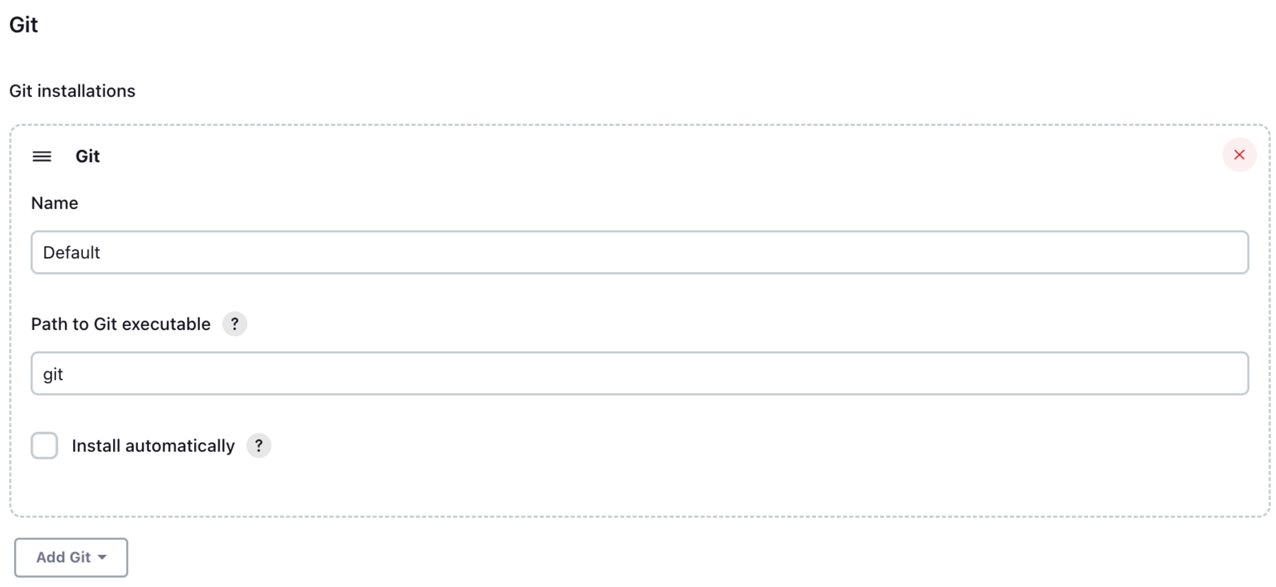
List of Maven installations on this system

Maven
Name: Maven

[✓] Install automatically ?

Install from Apache

Version: 3.9.1


Git

Git installations

Git

Name

Default

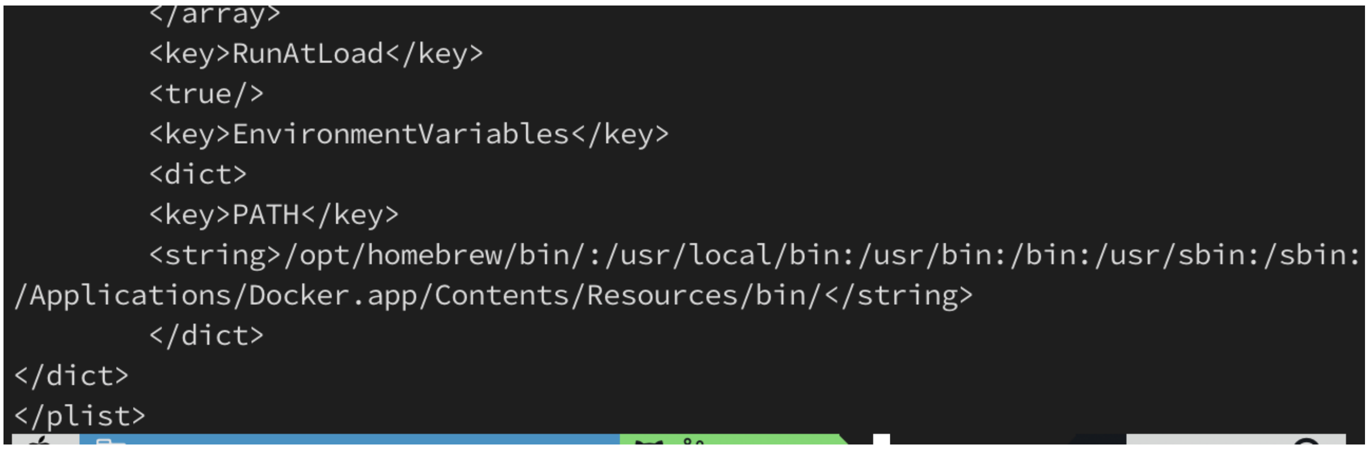
Path to Git executable

git

Install automatically

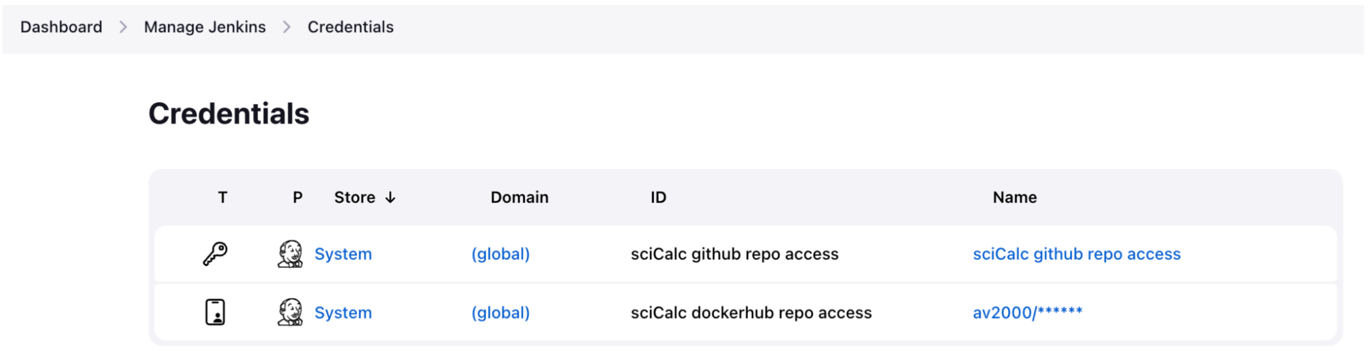
For docker and ansible, we need to give paths to the root directory but for security reasons macos doesn't allow direct access to those files. So, we need to configure the root file of jenkins on your system. Given below are the steps for the same:

i. Go to the folder: **/opt/homebrew/Cellar/jenkins-lts/<jenkins-version>**
ii. Open the file with the ".plist” extension and add the following lines of code:


```
        </array>
        <key>RunAtLoad</key>
        <true/>
        <key>EnvironmentVariables</key>
        <dict>
        <key>PATH</key>
        <string>/opt/homebrew/bin/:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:
/Applications/Docker.app/Contents/Resources/bin/</string>
        </dict>
</dict>
</plist>
```

This will let jenkins know the path to your docker service.

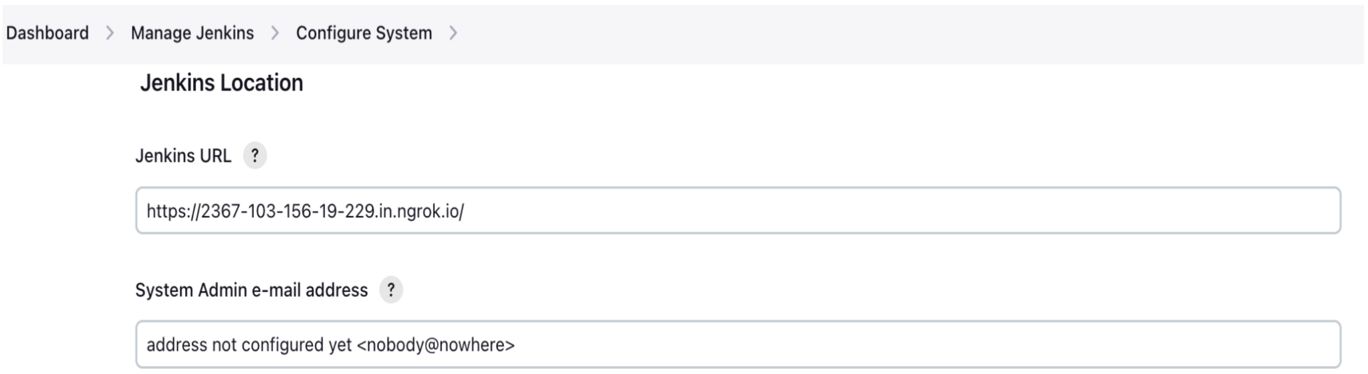
3. **Manage Jenkins → Manage Credentials**


Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	sciCalc github repo access	sciCalc github repo access
		System	(global)	sciCalc dockerhub repo access	av2000/******

4. **Manage Jenkins → Configure System**


Dashboard > Manage Jenkins > Configure System >

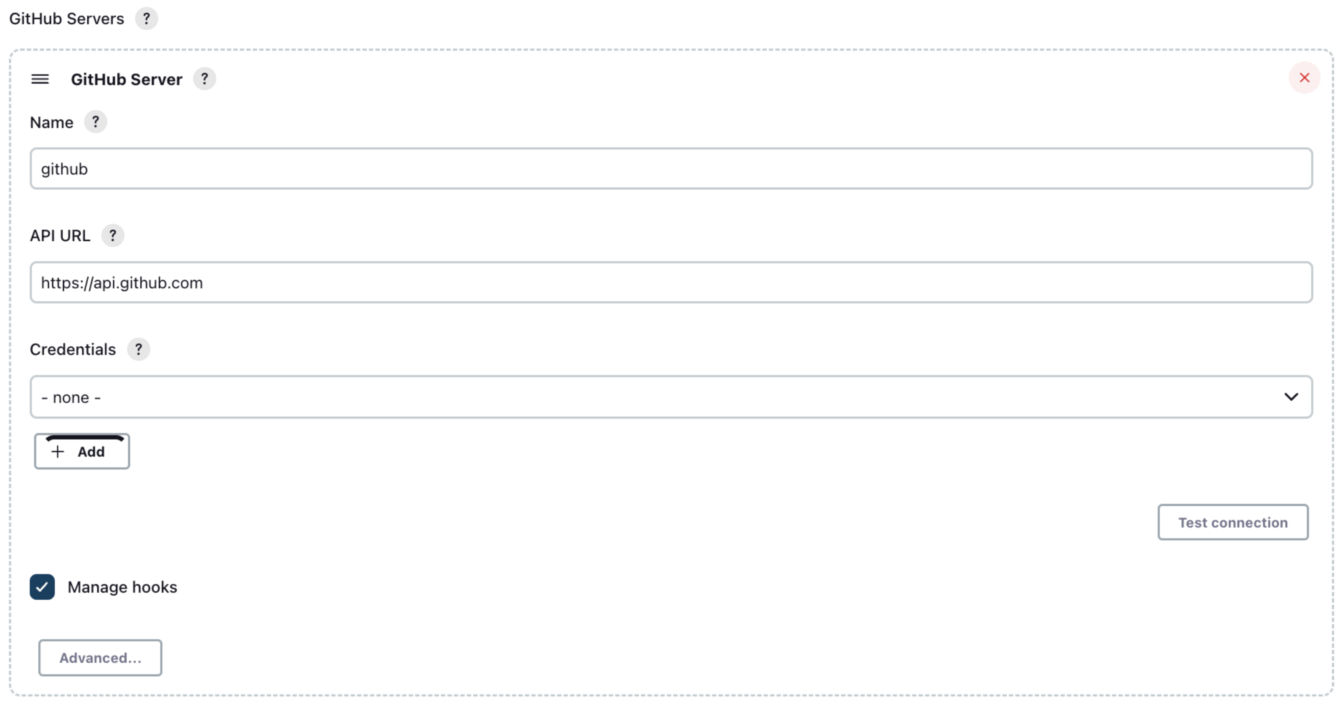
Jenkins Location

Jenkins URL ?

https://2367-103-156-19-229.in.ngrok.io/

System Admin e-mail address ?

address not configured yet <nobody@nowhere>


GitHub Servers ?

GitHub Server ?

Name ?

github

API URL ?

https://api.github.com

Credentials ?

- none -

[✓] Manage hooks

Jenkins Pipeline

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

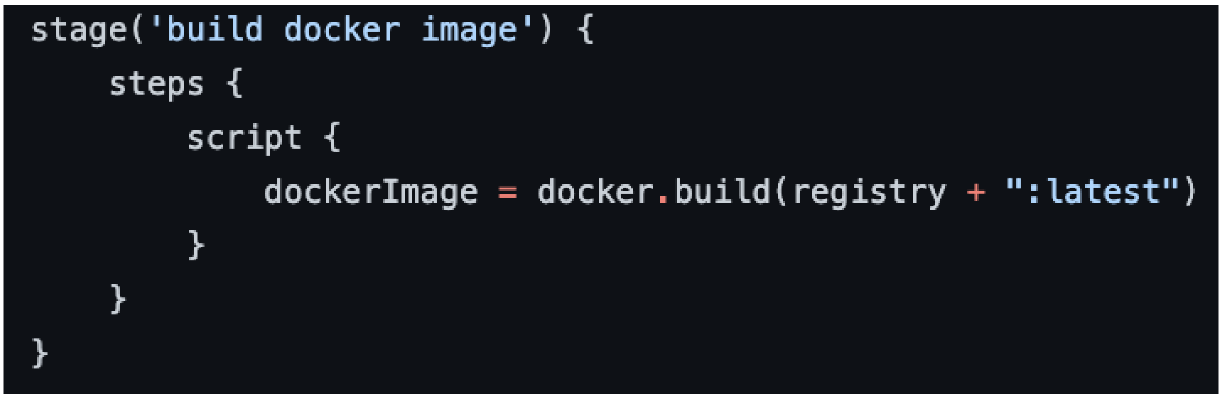
```
stage('Git pull') {
    steps {
        // Get some code from a GitHub repository
        git 'https://github.com/vardhan2000/scientificCalculator.git'
    }
}
```

2. **Maven Build:** perform a complete build of the project, resolving any dependencies and generating necessary build artifacts. If the build is successful, the resulting jar file will be placed in the target directory of the project.

```
stage('Build') {
    steps {
        // Run Maven on a Unix agent.
        sh "mvn clean install"
    }
}
```

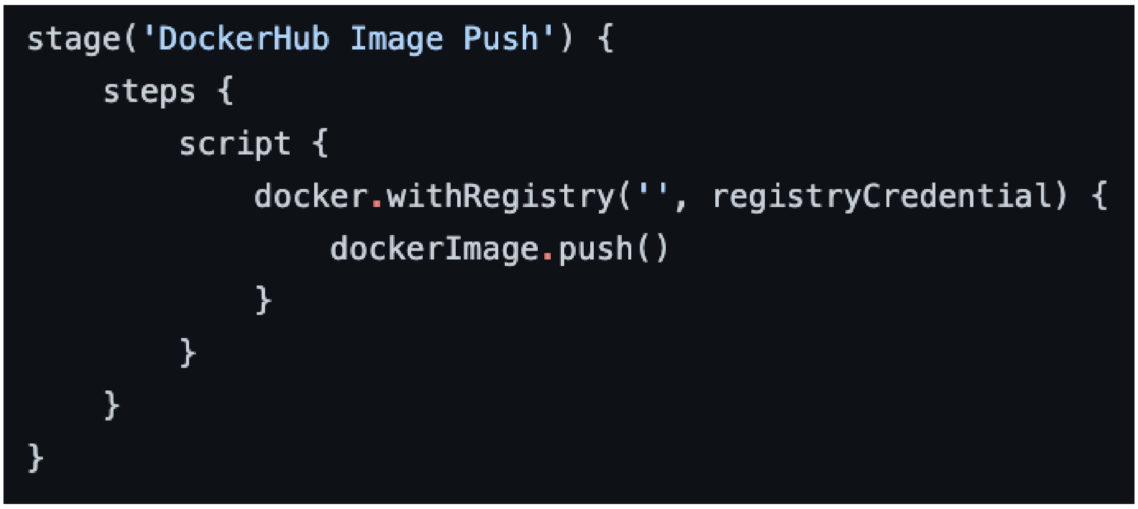
3. **Docker Image Creation:** It is used to create images on our local system that are then uploaded to our Docker hub, allowing us to fetch the image and execute the application on other servers. Environment only creates variables that can be utilised later. The tag name for the image is “latest”.

```
environment
{
    registry = "av2000/scientific-calculator"
    registryCredential = "sciCalc dockerhub repo access"
    dockerImage = ""
}
```



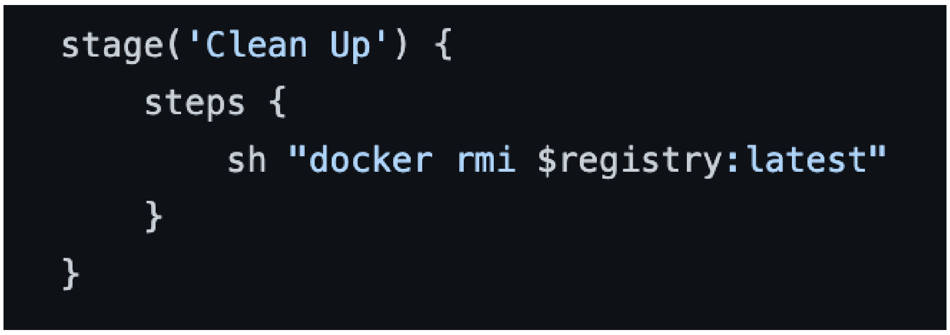
```
stage('build docker image') {
    steps {
        script {
            dockerImage = docker.build(registry + ":latest")
        }
    }
}
```

4. **Push docker image:** deploys the image into DockerHub so that anyone can pull the image.



```
stage('DockerHub Image Push') {
    steps {
        script {
            docker.withRegistry('', registryCredential) {
                dockerImage.push()
            }
        }
    }
}
```

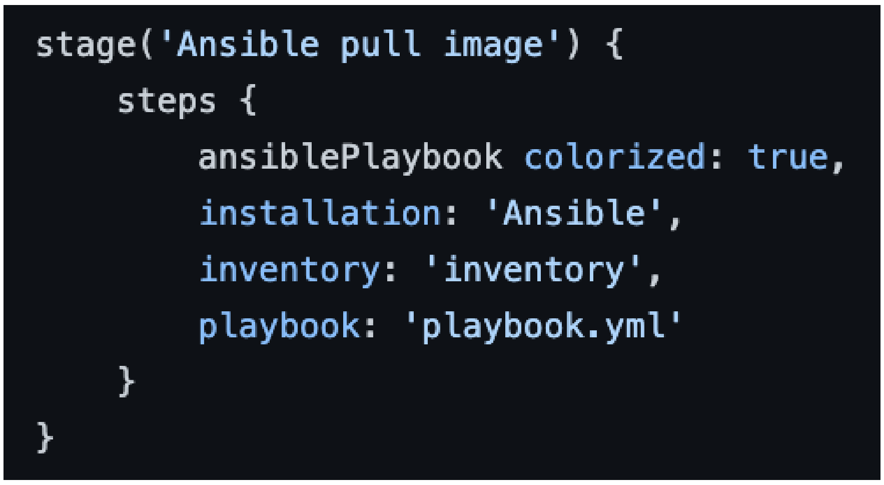
5. **Removing Docker Image:**



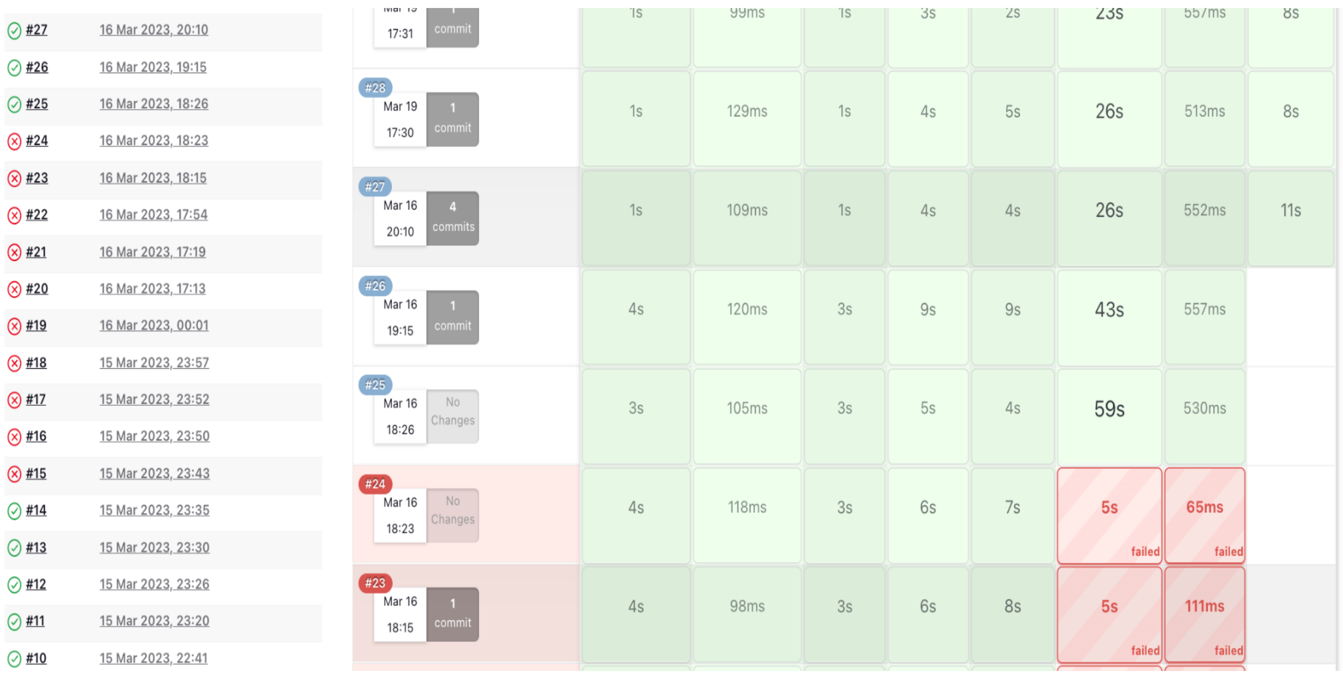
```
stage('Clean Up') {
    steps {
        sh "docker rmi $registry:latest"
    }
}
```

The **docker rmi** command removes one or more images from the host node.

6. **Ansible pull image:**



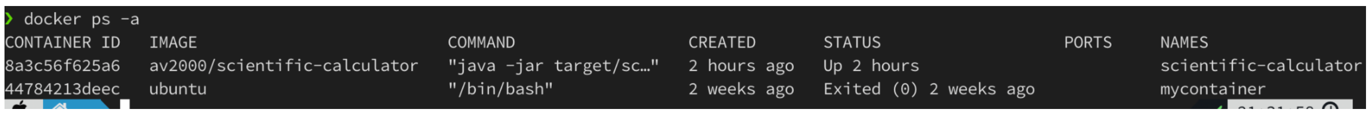
```
stage('Ansible pull image') {
    steps {
        ansiblePlaybook colorized: true,
        installation: 'Ansible',
        inventory: 'inventory',
        playbook: 'playbook.yml'
    }
}
```



✓	#27	16 Mar 2023, 20:10
✓	#26	16 Mar 2023, 19:15
✓	#25	16 Mar 2023, 18:26
✗	#24	16 Mar 2023, 18:23
✗	#23	16 Mar 2023, 18:15
✗	#22	16 Mar 2023, 17:54
✗	#21	16 Mar 2023, 17:19
✗	#20	16 Mar 2023, 17:13
✗	#19	16 Mar 2023, 00:01
✗	#18	15 Mar 2023, 23:57
✗	#17	15 Mar 2023, 23:52
✗	#16	15 Mar 2023, 23:50
✗	#15	15 Mar 2023, 23:43
✓	#14	15 Mar 2023, 23:35
✓	#13	15 Mar 2023, 23:30
✓	#12	15 Mar 2023, 23:26
✓	#11	15 Mar 2023, 23:20
✓	#10	15 Mar 2023, 22:41

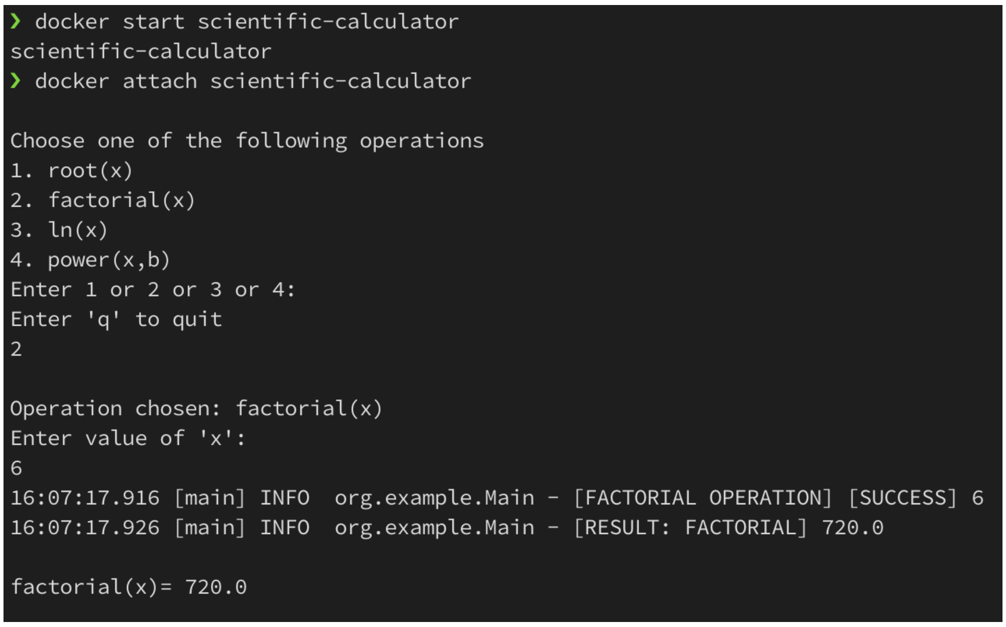
	17:31	commit	1s	99ms	1s	3s	2s	23s	557ms	8s
#28	Mar 19 17:30	1 commit	1s	129ms	1s	4s	5s	26s	513ms	8s
#27	Mar 16 20:10	4 commits	1s	109ms	1s	4s	4s	26s	552ms	11s
#26	Mar 16 19:15	1 commit	4s	120ms	3s	9s	9s	43s	557ms	
#25	Mar 16 18:26	No Changes	3s	105ms	3s	5s	4s	59s	530ms	
#24	Mar 16 18:23	No Changes	4s	118ms	3s	6s	7s	5s failed	65ms failed	
#23	Mar 16 18:15	1 commit	4s	98ms	3s	6s	8s	5s failed	111ms failed	

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



```
❯ docker ps -a
CONTAINER ID   IMAGE                           COMMAND                  CREATED        STATUS                   PORTS     NAMES
8a3c56f625a6   av2000/scientific-calculator   "java -jar target/sc…"   2 hours ago    Up 2 hours                         scientific-calculator
44784213deec   ubuntu                          "/bin/bash"              2 weeks ago    Exited (0) 2 weeks ago             mycontainer
```

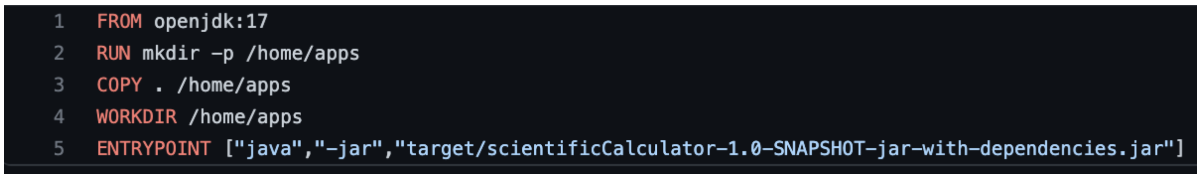
Start the container and then attach to it to execute the jar



```
❯ docker start scientific-calculator
scientific-calculator
❯ docker attach scientific-calculator

Choose one of the following operations
1. root(x)
2. factorial(x)
3. ln(x)
4. power(x,b)
Enter 1 or 2 or 3 or 4:
Enter 'q' to quit
2

Operation chosen: factorial(x)
Enter value of 'x':
6
16:07:17.916 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [SUCCESS] 6
16:07:17.926 [main] INFO  org.example.Main - [RESULT: FACTORIAL] 720.0

factorial(x)= 720.0
```

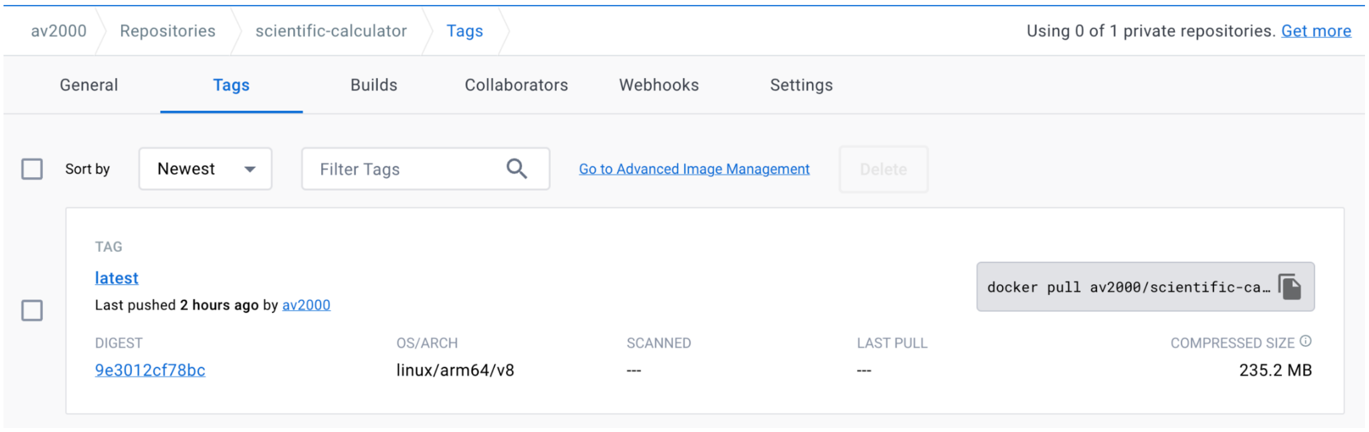
Containerize

- Docker is a platform that simplifies the process of building, deploying, and running applications inside containers. Containers are lightweight, portable, and self-contained environments that provide an isolated environment for running software. Docker allows developers to package their applications and dependencies into a single container that can be deployed across different environments, making it easier to manage and scale applications in a consistent and repeatable manner.
- In order to generate a docker image, we will use open-JDK 17 and the calculator jar file. The image will then be published on the Docker Hub following that (we need to create a public repository on the docker hub before pushing the image). The image will then be downloaded from Docker Hub by Ansible and installed on several computers.



```
FROM openjdk:17
RUN mkdir -p /home/apps
COPY . /home/apps
WORKDIR /home/apps
ENTRYPOINT ["java","-jar","target/scientificCalculator-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

- To build a docker image, a docker file is used in which the script is written. In the above mentioned docker file. The Dockerfile code above performs the following steps:
    - Specifies the base image to be used (OpenJDK version 17).
    - Creates a new directory at /home/apps.
    - Copies the current directory into the /home/apps directory in the container.
    - Sets the working directory to /home/apps.
    - Specifies the command to run when the container starts, which is to execute the jar file located in the target directory with all its dependencies.

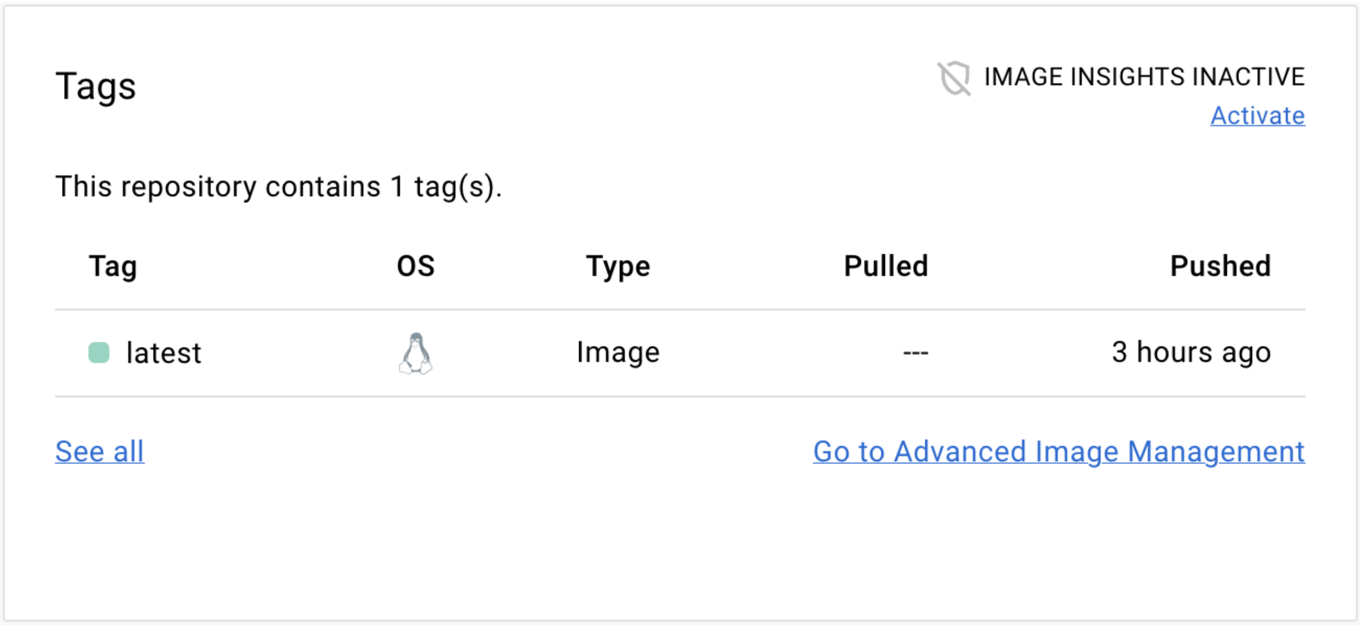


av2000 › Repositories › scientific-calculator › Tags   Using 0 of 1 private repositories. Get more

TAG
latest
Last pushed 2 hours ago by av2000

docker pull av2000/scientific-ca…

DIGEST	OS/ARCH	SCANNED	LAST PULL	COMPRESSED SIZE
9e3012cf78bc	linux/arm64/v8	---	---	235.2 MB



Tags

IMAGE INSIGHTS INACTIVE
Activate

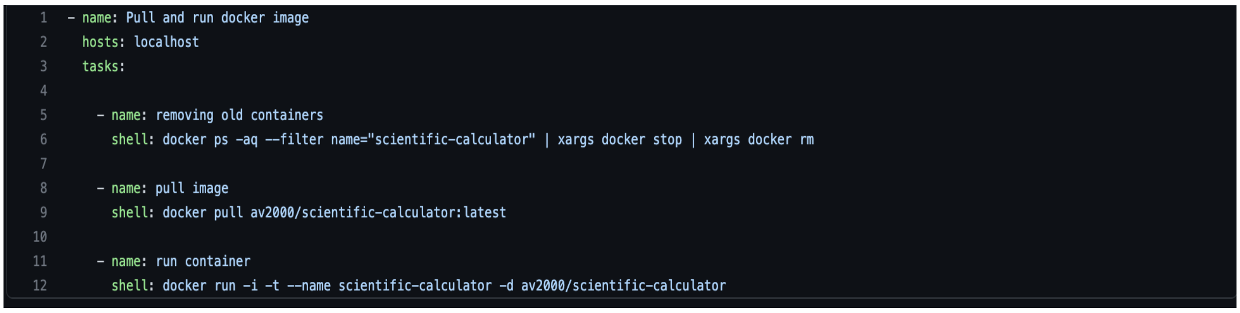
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	---	3 hours ago

Ansible

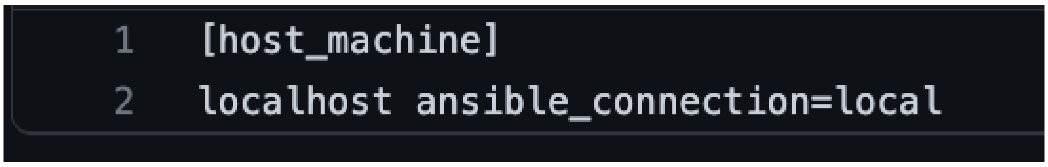
- Ansible is an open-source automation tool that simplifies IT configuration management and application deployment across servers, networks, and cloud environments using a simple, human-readable language. It allows users to automate tasks across multiple systems, making it easier to manage and scale infrastructure. Ansible uses a push-based model, where the control machine pushes configurations and commands to remote machines, and supports a wide range of modules and plugins for managing different technologies and platforms. Ansible is used by IT teams of all sizes to automate repetitive tasks, improve efficiency, and reduce errors in the IT environment.

- **Playbook**



```
- name: Pull and run docker image
  hosts: localhost
  tasks:

    - name: removing old containers
      shell: docker ps -aq --filter name="scientific-calculator" | xargs docker stop | xargs docker rm

    - name: pull image
      shell: docker pull av2000/scientific-calculator:latest

    - name: run container
      shell: docker run -i -t --name scientific-calculator -d av2000/scientific-calculator
```

- **Inventory**

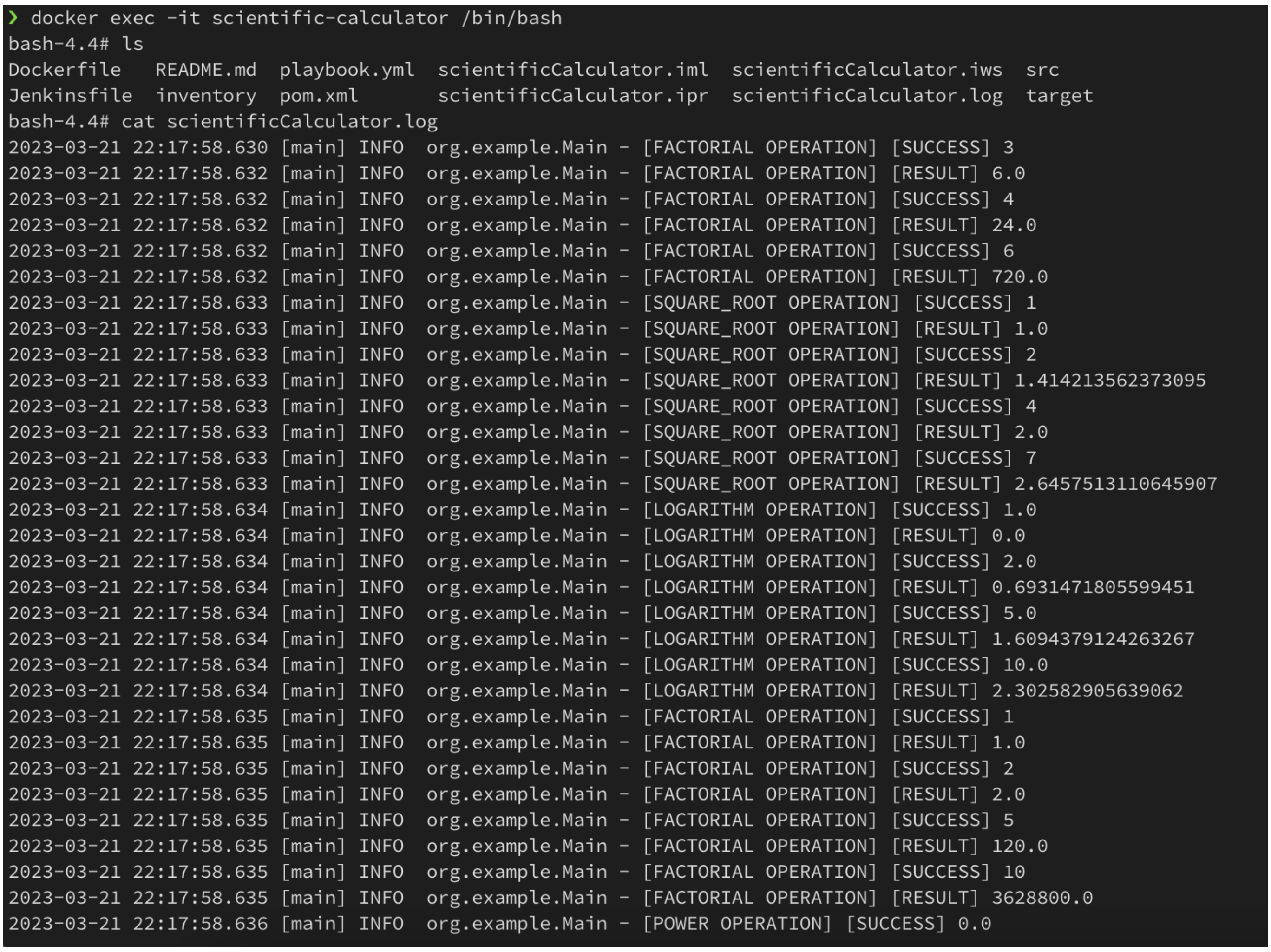


```
[host_machine]
localhost ansible_connection=local
```

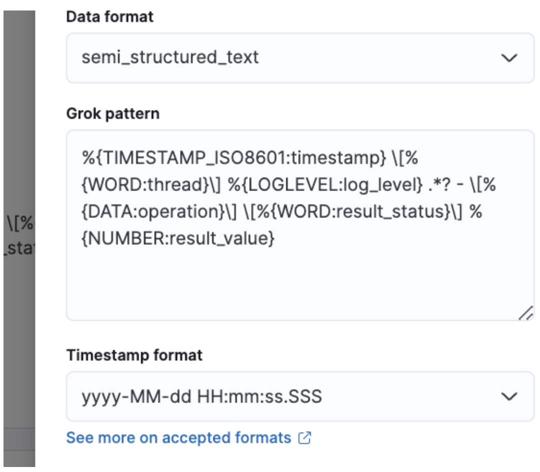
Continuous Monitoring

➢ The next step is to monitor your system once your deployment is complete. Monitoring involves checking to see if the programme is operating as planned.

➢ ELK stack is the monitoring tool that can be used for any software that has been deployed. It does an analysis of the logs, and then that analysis can be viewed on the kibana dashboard.



```
❯ docker exec -it scientific-calculator /bin/bash
bash-4.4# ls
Dockerfile   README.md  playbook.yml  scientificCalculator.iml  scientificCalculator.iws  src
Jenkinsfile  inventory  pom.xml       scientificCalculator.ipr  scientificCalculator.log  target
bash-4.4# cat scientificCalculator.log
2023-03-21 22:17:58.630 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [SUCCESS] 3
2023-03-21 22:17:58.632 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [RESULT] 6.0
2023-03-21 22:17:58.632 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [SUCCESS] 4
2023-03-21 22:17:58.632 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [RESULT] 24.0
2023-03-21 22:17:58.632 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [SUCCESS] 6
2023-03-21 22:17:58.632 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [RESULT] 720.0
2023-03-21 22:17:58.633 [main] INFO  org.example.Main - [SQUARE_ROOT OPERATION] [SUCCESS] 1
2023-03-21 22:17:58.633 [main] INFO  org.example.Main - [SQUARE_ROOT OPERATION] [RESULT] 1.0
2023-03-21 22:17:58.633 [main] INFO  org.example.Main - [SQUARE_ROOT OPERATION] [SUCCESS] 2
2023-03-21 22:17:58.633 [main] INFO  org.example.Main - [SQUARE_ROOT OPERATION] [RESULT] 1.414213562373095
2023-03-21 22:17:58.633 [main] INFO  org.example.Main - [SQUARE_ROOT OPERATION] [SUCCESS] 4
2023-03-21 22:17:58.633 [main] INFO  org.example.Main - [SQUARE_ROOT OPERATION] [RESULT] 2.0
2023-03-21 22:17:58.633 [main] INFO  org.example.Main - [SQUARE_ROOT OPERATION] [SUCCESS] 7
2023-03-21 22:17:58.633 [main] INFO  org.example.Main - [SQUARE_ROOT OPERATION] [RESULT] 2.6457513110645907
2023-03-21 22:17:58.634 [main] INFO  org.example.Main - [LOGARITHM OPERATION] [SUCCESS] 1.0
2023-03-21 22:17:58.634 [main] INFO  org.example.Main - [LOGARITHM OPERATION] [RESULT] 0.0
2023-03-21 22:17:58.634 [main] INFO  org.example.Main - [LOGARITHM OPERATION] [SUCCESS] 2.0
2023-03-21 22:17:58.634 [main] INFO  org.example.Main - [LOGARITHM OPERATION] [RESULT] 0.6931471805599451
2023-03-21 22:17:58.634 [main] INFO  org.example.Main - [LOGARITHM OPERATION] [SUCCESS] 5.0
2023-03-21 22:17:58.634 [main] INFO  org.example.Main - [LOGARITHM OPERATION] [RESULT] 1.6094379124263267
2023-03-21 22:17:58.634 [main] INFO  org.example.Main - [LOGARITHM OPERATION] [SUCCESS] 10.0
2023-03-21 22:17:58.634 [main] INFO  org.example.Main - [LOGARITHM OPERATION] [RESULT] 2.302582905639062
2023-03-21 22:17:58.635 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [SUCCESS] 1
2023-03-21 22:17:58.635 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [RESULT] 1.0
2023-03-21 22:17:58.635 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [SUCCESS] 2
2023-03-21 22:17:58.635 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [RESULT] 2.0
2023-03-21 22:17:58.635 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [SUCCESS] 5
2023-03-21 22:17:58.635 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [RESULT] 120.0
2023-03-21 22:17:58.635 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [SUCCESS] 10
2023-03-21 22:17:58.635 [main] INFO  org.example.Main - [FACTORIAL OPERATION] [RESULT] 3628800.0
2023-03-21 22:17:58.636 [main] INFO  org.example.Main - [POWER OPERATION] [SUCCESS] 0.0
```



Data format

semi_structured_text

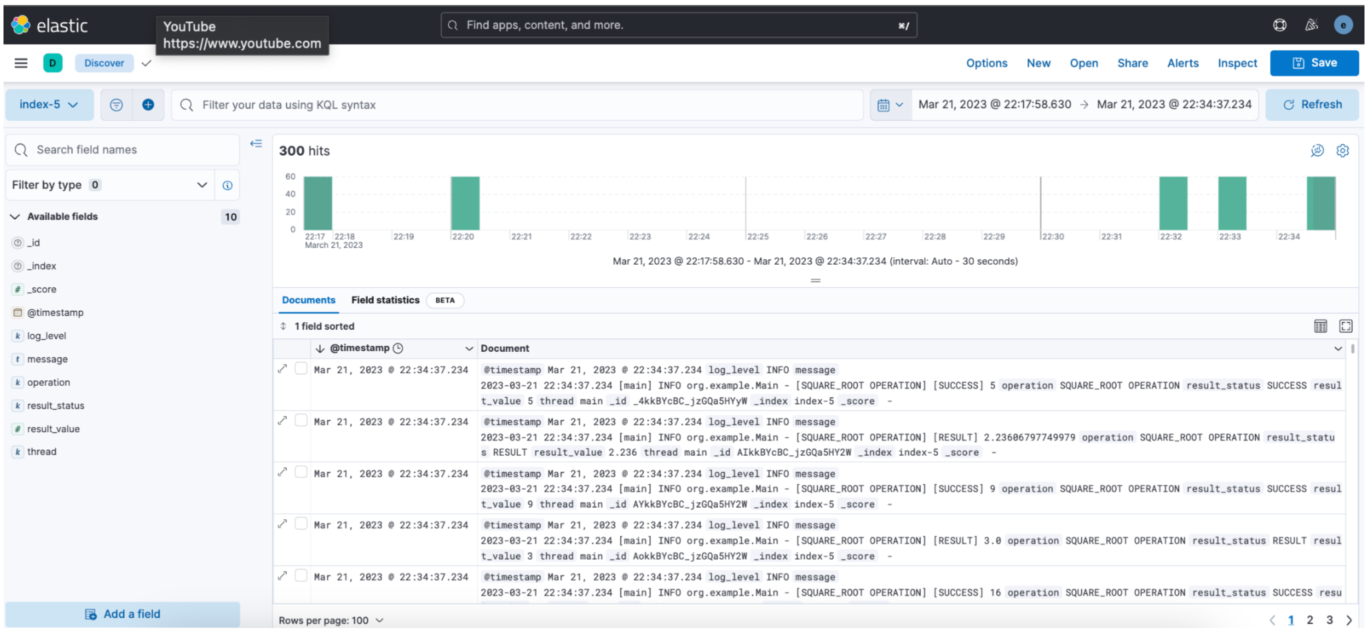
Grok pattern

```
%{TIMESTAMP_ISO8601:timestamp} \[%{WORD:thread}\] %{LOGLEVEL:log_level} .*? - \[%{DATA:operation}\] \[%{WORD:result_status}\] %{NUMBER:result_value}
```

Timestamp format

yyyy-MM-dd HH:mm:ss.SSS

See more on accepted formats

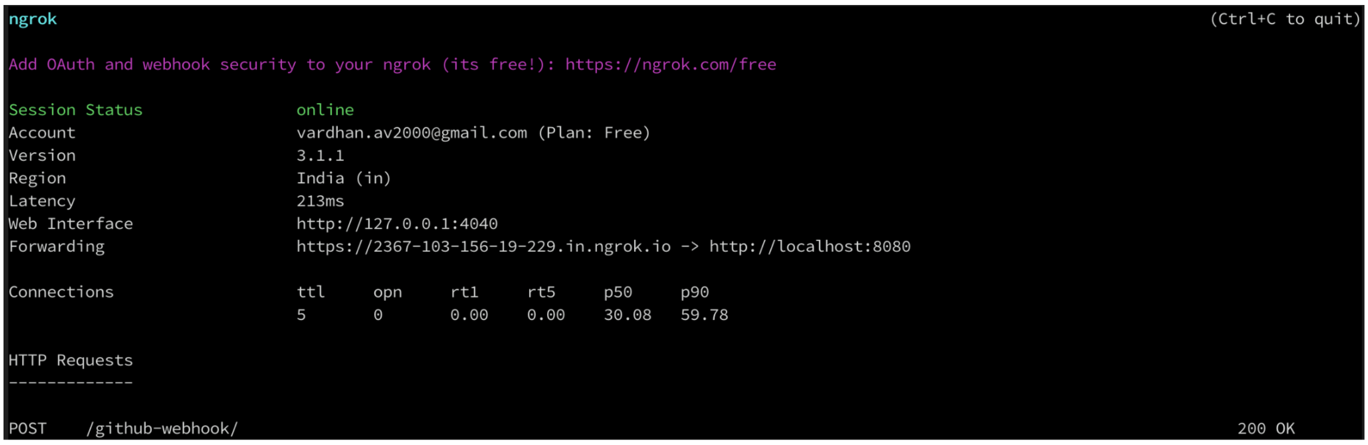


Mar 21, 2023 @ 22:17:58.630 - Mar 21, 2023 @ 22:34:37.234 (interval: Auto - 30 seconds)

```
@timestamp              Document
Mar 21, 2023 @ 22:34:37.234  @timestamp Mar 21, 2023 @ 22:34:37.234 log_level INFO message
2023-03-21 22:34:37.234 [main] INFO org.example.Main - [SQUARE_ROOT OPERATION] [SUCCESS] 5 operation SQUARE_ROOT OPERATION result_status SUCCESS result_value 5 thread main _id _4kkBYcBC_jzGQa5HYyW _index index-5 _score -
Mar 21, 2023 @ 22:34:37.234  @timestamp Mar 21, 2023 @ 22:34:37.234 log_level INFO message
2023-03-21 22:34:37.234 [main] INFO org.example.Main - [SQUARE_ROOT OPERATION] [RESULT] 2.23606797749979 operation SQUARE_ROOT OPERATION result_status RESULT result_value 2.236 thread main _id AIkkBYcBC_jzGQa5HY2W _index index-5 _score -
Mar 21, 2023 @ 22:34:37.234  @timestamp Mar 21, 2023 @ 22:34:37.234 log_level INFO message
2023-03-21 22:34:37.234 [main] INFO org.example.Main - [SQUARE_ROOT OPERATION] [SUCCESS] 9 operation SQUARE_ROOT OPERATION result_status SUCCESS result_value 9 thread main _id AYkkBYcBC_jzGQa5HY2W _index index-5 _score -
Mar 21, 2023 @ 22:34:37.234  @timestamp Mar 21, 2023 @ 22:34:37.234 log_level INFO message
2023-03-21 22:34:37.234 [main] INFO org.example.Main - [SQUARE_ROOT OPERATION] [RESULT] 3.0 operation SQUARE_ROOT OPERATION result_status RESULT result_value 3 thread main _id AokkBYcBC_jzGQa5HY2W _index index-5 _score -
Mar 21, 2023 @ 22:34:37.234  @timestamp Mar 21, 2023 @ 22:34:37.234 log_level INFO message
2023-03-21 22:34:37.234 [main] INFO org.example.Main - [SQUARE_ROOT OPERATION] [SUCCESS] 16 operation SQUARE_ROOT OPERATION result_status SUCCESS resu
```

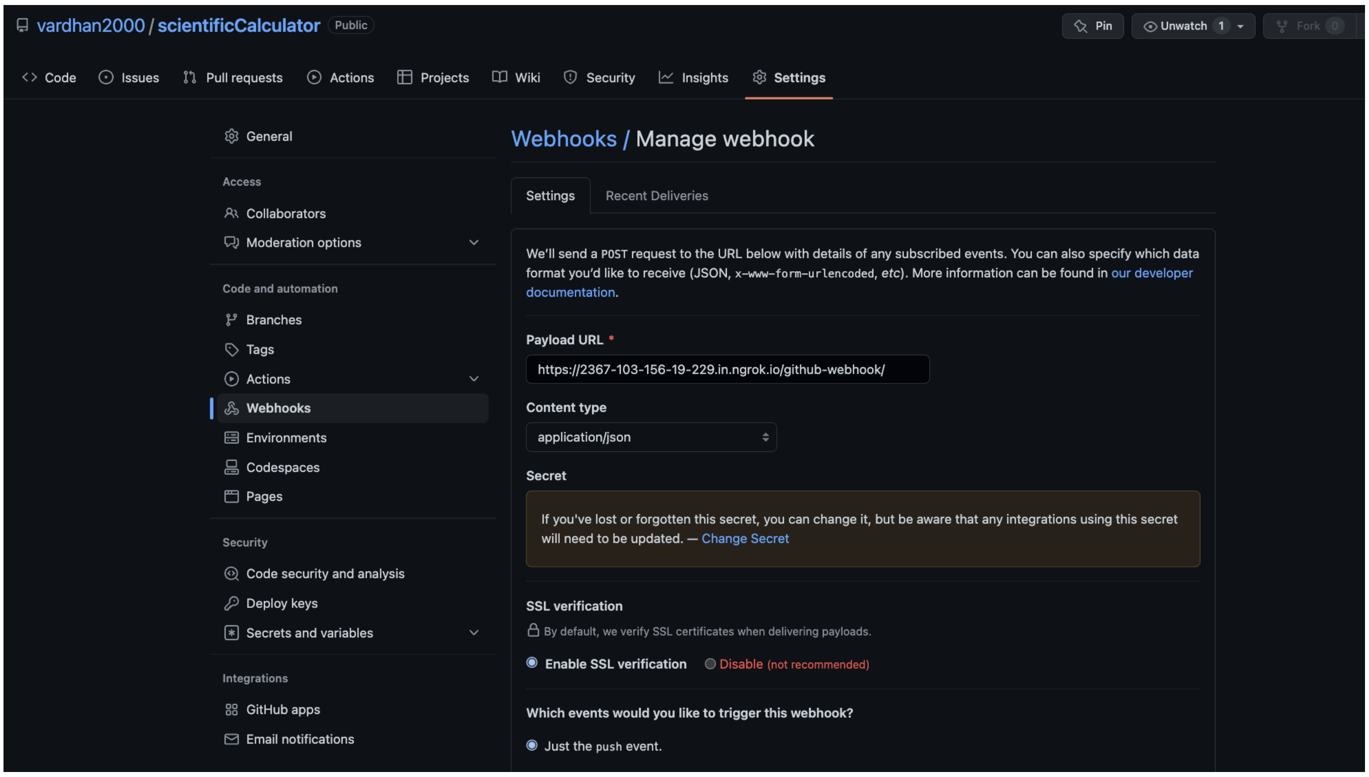
WebHooks

- Webhooks are messages that are transmitted immediately whenever there is a change in the environment. If we make any changes to the GitHub repository, the webhook will activate the Jenkins pipeline immediately, and it will do so automatically. Ngrok is able to link local servers that are protected by NATs (Network Address Translation) and firewalls to the public internet by utilising encrypted tunnels. It is equipped with a real-time web interface that gives you the ability to view any HTTP traffic that is moving through your tunnels. It gives you the capability of connecting to the internet through a web server that is operating on your local machine. Just provide ngrok with the port number on which your web server is actively listening.



```
ngrok                                                                                         (Ctrl+C to quit)

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

Session Status                online
Account                       vardhan.av2000@gmail.com (Plan: Free)
Version                       3.1.1
Region                        India (in)
Latency                       213ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://2367-103-156-19-229.in.ngrok.io -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              5       0       0.00    0.00    30.08   59.78

HTTP Requests
-------------

POST   /github-webhook/                                                                          200 OK
```

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



Webhooks / Manage webhook

Settings | Recent Deliveries

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

https://2367-103-156-19-229.in.ngrok.io/github-webhook/

Content type

application/json

Secret

If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated. — Change Secret

SSL verification

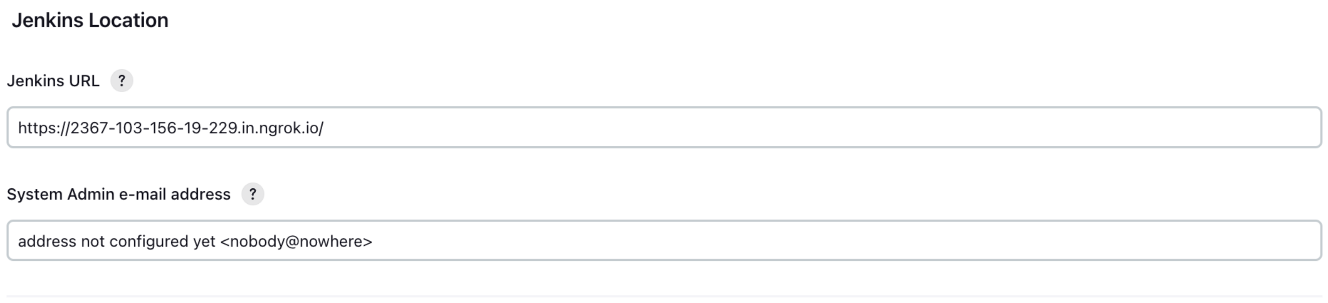
By default, we verify SSL certificates when delivering payloads.

[✓ Enable SSL verification] Disable (not recommended)

Which events would you like to trigger this webhook?

[✓ Just the `push` event.]

In Jenkins → Configure System



Jenkins Location

Jenkins URL ?

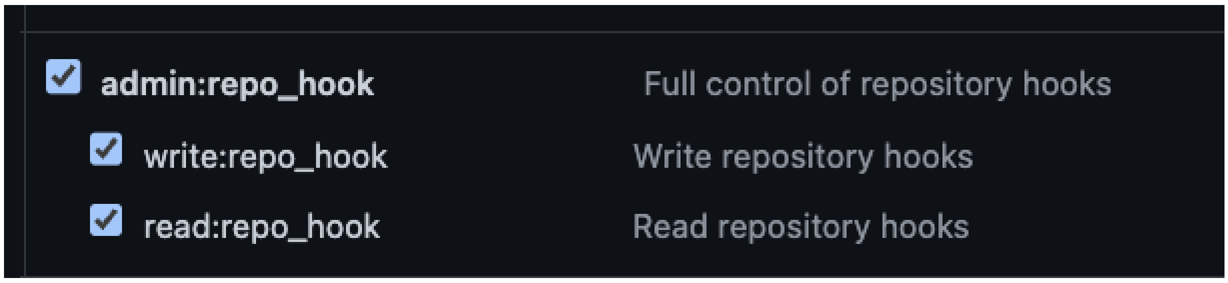
https://2367-103-156-19-229.in.ngrok.io/

System Admin e-mail address ?

address not configured yet <nobody@nowhere>

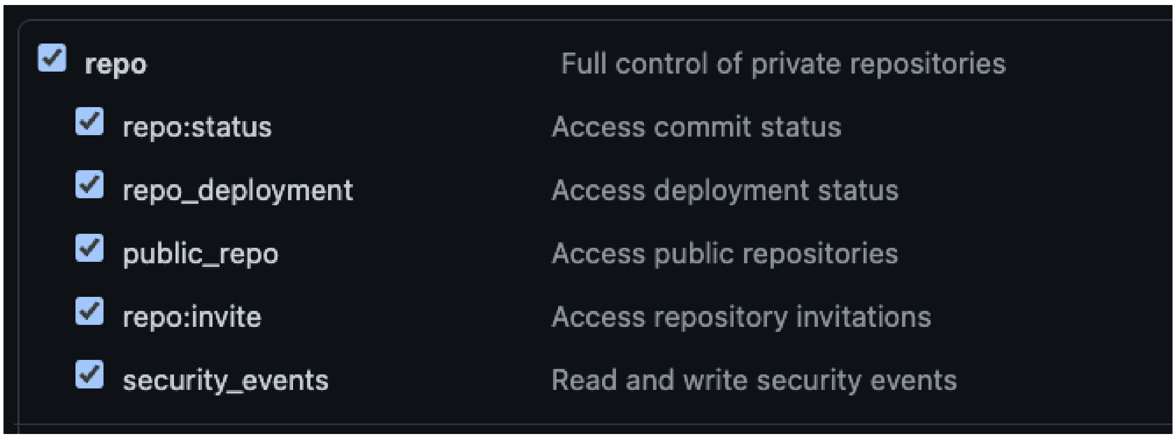
Challenges

➢ Initially, the webhook was not working, as Jenkins was not detecting the commits. At the time of github token generation, I selected the following scopes:

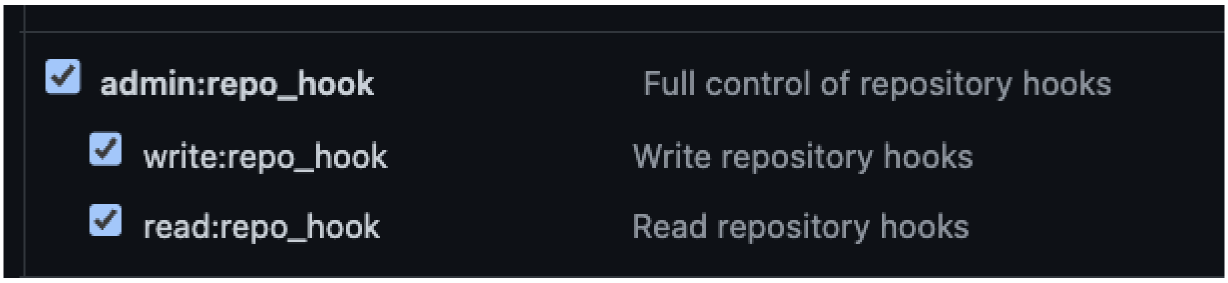


✓ admin:repo_hook	Full control of repository hooks
✓ write:repo_hook	Write repository hooks
✓ read:repo_hook	Read repository hooks

Then I increased the scope of the token to the following:



✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events



✓ admin:repo_hook	Full control of repository hooks
✓ write:repo_hook	Write repository hooks
✓ read:repo_hook	Read repository hooks

And then it worked.

➢ I had trouble integrating Jenkins with docker. MacOS was not giving access to Jenkins to read the root directory of docker. I finally figured out that it could be done by adding the path to the root directory of docker in the **.plist** file present in Jenkins' root directory.