Mini Project Report

D V N P S M Manohar (IMT2019025)

Problem:

Creating a scientific calculator with the operations:

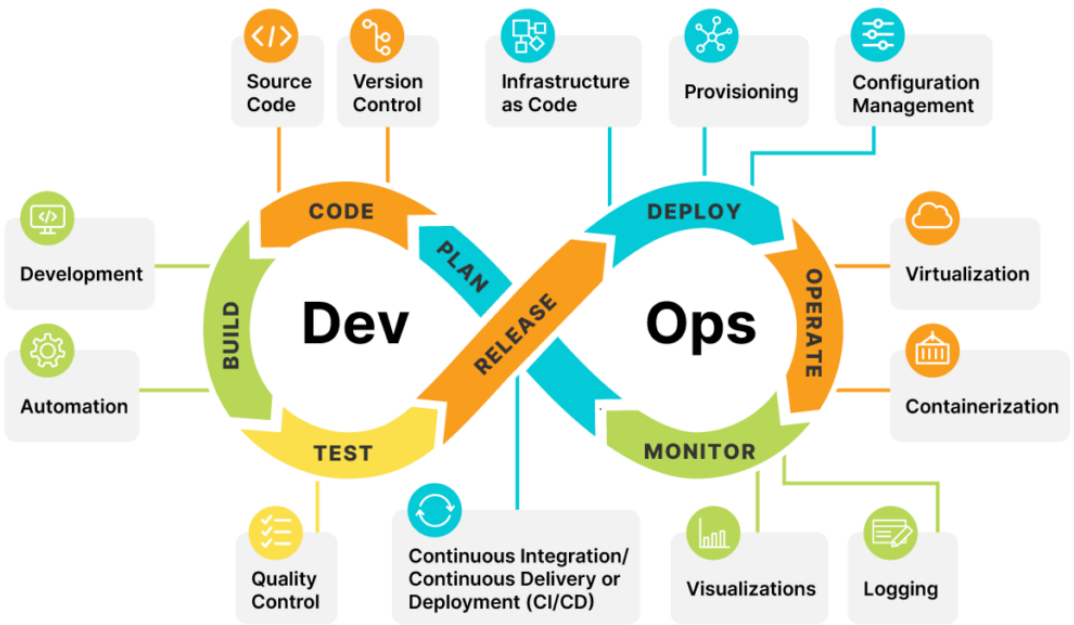
- Square Root Function
- Factorial Function
- Natural Log
- Power Function

Using jenkins pipeline for hands on with Devops concept of CI/CD.

Links:

- Github: https://github.com/manoharsuggula/ScientificCalculator
- Dockerhub: https://hub.docker.com/repository/docker/manoharsuggula/scientific_calculator/general

What is DevOps:



DevOps is a software development methodology that stresses collaboration and communication between software developers and IT operations teams. The aim of DevOps is to improve the speed, quality, and reliability of software delivery by automating processes and removing barriers between teams.

Traditionally, software development and IT operations were seen as separate and distinct disciplines, with different goals, priorities, and processes. In DevOps, Software creation, testing, and deployment are all integrated into a single, continuous process. This enables developers to test and release new code changes quickly while also providing insight and control over the entire software development lifecycle.

DevOps aims to remove these obstacles, resulting in a more collaborative and efficient software development process. Developers and operations teams collaborate instead of working in separate silos to create, test, and launch software, sharing knowledge and feedback along the way.

Automation tools for building, testing, and deploying software, as well as collaboration tools for sharing information and feedback between teams, are frequently used in DevOps methods. Continuous integration and continuous delivery (CI/CD) are other terms for automating the process of developing, testing, and delivering software in DevOps.

Why DevOps:

DevOps is "better" than other approaches to software development because it provides a number of advantages that make it well-suited to contemporary software development environments. One of the key principles of DevOps is automation. DevOps can greatly reduce the time to market for new features and applications by automating many parts of the software development and deployment process. DevOps enables developers to test and deploy changes to production more quickly, enabling organizations to react more quickly to changing business needs and customer demands.

Continuous improvement is another important DevOps concept. Teams can identify areas for growth and make iterative changes to improve speed, quality, and reliability by frequently monitoring and analyzing the software development process.

Another advantage of DevOps is dependability. DevOps can help ensure that software is fully tested and validated before it is released to production by integrating testing and deployment into a single, continuous process. This reduces the likelihood of bugs and errors affecting users and harming the organization's image.

Overall, DevOps is a flexible and adaptable strategy to software development that can assist organizations in rapidly and reliably building and deploying high-quality software. DevOps has demonstrated to be a highly effective approach for many modern software development environments, even if it is not the best fit for every company or project.

Tools Used:

1. **Maven:** It is a popular build automation and project management software, which is mainly used for Java-based tasks. It offers a standard method to arrange project files, manage dependencies, and automate the build process with the goal of making the process of creating and managing Java-based projects simpler.
    - **Installation**
        - sudo apt install maven
        - Check whether maven is installed or not by checking the version

```
chakradar@reddy:~/Desktop/ScientificCalculator$ mvn -v
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.18, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "5.19.0-32-generic", arch: "amd64", family: "unix"
chakradar@reddy:~/Desktop/ScientificCalculator$
```

2. **Git and Github:** Git is a version control system. GitHub is a famous web-based version control and collaborative software development platform.

3. **Docker:** Docker is a popular containerization platform for developers that enables them to package and distribute applications in isolated, lightweight containers.
    - **Installation**:
        - sudo apt-get install docker.io
        - Check whether docker is installed or not by checking the version

```
chakradar@reddy:~/Desktop/ScientificCalculator$ docker --version
Docker version 20.10.21, build 20.10.21-0ubuntu1~22.04.2
chakradar@reddy:~/Desktop/ScientificCalculator$
```

4. **Ansible:** Ansible is a popular open-source automation tool for automating IT infrastructure deployment, configuration, and administration. It is intended to provide a simple yet powerful means of automating repetitive tasks across a broad variety of systems and environments, such as server configuration, application deployment, and database management.
    - **Installation:**
        - sudo add-apt-repository ppa:ansible/ansible-2.9
        - sudo apt install ansible
        - Check whether ansible is installed or not by checking the version

```
chakradar@reddy:~/Desktop/ScientificCalculator$ ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/home/chakradar/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0]
chakradar@reddy:~/Desktop/ScientificCalculator$
```

5. **Jenkins:** Jenkins is a famous open-source automation server that is used to automate various aspects of the software development process, such as software development, testing, and deployment. It is intended to provide a continuous integration and continuous delivery (CI/CD) pipeline to assist developers in quickly and effectively delivering high-quality software.
    - **Installation**:
        - wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

- sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
- sudo apt install ca-certificates
- sudo apt-get install jenkins
- Check the jenkins status as shown

```
chakradar@reddy:~/Desktop/ScientificCalculator$ sudo service jenkins status
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-03-21 15:20:02 IST; 4h 15min ago
   Main PID: 7562 (java)
      Tasks: 64 (limit: 9012)
     Memory: 1.5G
        CPU: 5min 14.729s
     CGroup: /system.slice/jenkins.service
             └─7562 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Mar 21 15:46:30 reddy jenkins[7562]: 2023-03-21 10:16:30.096+0000 [id=261]        INFO        o.j.p.g.w.s.DefaultPushGHEventSubscriber$1#run:>
Mar 21 15:46:30 reddy jenkins[7562]: 2023-03-21 10:16:30.970+0000 [id=310]        INFO        c.c.jenkins.GitHubPushTrigger$1#run: SCM change>
Mar 21 15:47:48 reddy python3[65143]: ansible-ansible.legacy.setup Invoked with gather_subset=['all'] gather_timeout=10 filter=* fact_path=/e>
Mar 21 15:47:49 reddy python3[65235]: ansible-docker_image Invoked with name=manoharsuggula/scientific_calculator source=pull docker_host=uni>
Mar 21 15:47:49 reddy python3[65281]: ansible-ansible.legacy.command Invoked with _raw_params=docker run -id manoharsuggula/scientific_calcul>
Mar 21 18:46:01 reddy jenkins[7562]: 2023-03-21 13:16:00.905+0000 [id=20]        WARNING        hudson.security.csrf.CrumbFilter#doFilter: Fo>
Mar 21 18:46:01 reddy jenkins[7562]: 2023-03-21 13:16:01.023+0000 [id=20]        WARNING        hudson.security.csrf.CrumbFilter#doFilter: No>
Mar 21 18:49:11 reddy python3[157914]: ansible-ansible.legacy.setup Invoked with gather_subset=['all'] gather_timeout=10 filter=* fact_path=/>
Mar 21 18:49:14 reddy python3[158099]: ansible-docker_image Invoked with name=manoharsuggula/scientific_calculator source=pull docker_host=un>
Mar 21 18:49:15 reddy python3[158211]: ansible-ansible.legacy.command Invoked with _raw_params=docker run -id manoharsuggula/scientific_calcu>
chakradar@reddy:~/Desktop/ScientificCalculator$
```

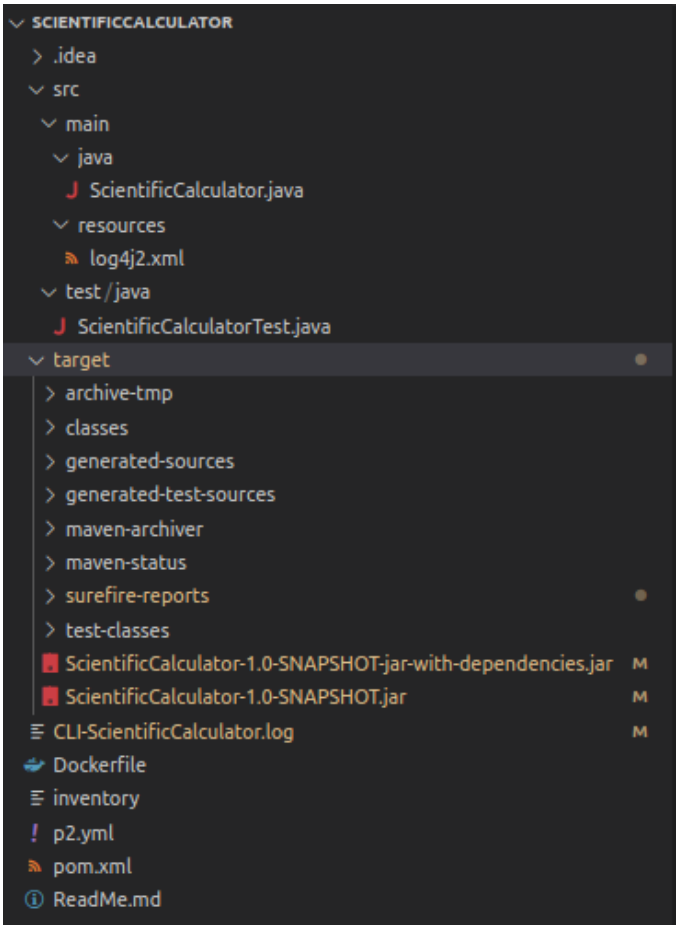
6. **Webhooks:** Webhooks help in triggering a job or an event in response to an action. When certain events take place, like when a new issue is created in a bug tracker or when a customer makes a purchase on an e-commerce website, they enable developers to receive real-time notifications. We used ngrok for this.
   - **Installation**:
     - Download the zip file from https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz
     - Extract it using
       - sudo tar xvzf ngrok-v3-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 8080; copy the public ip address for your local host.

7. **Elk Stack:** ELK stack is a collection of three open-source tools: Elasticsearch, Logstash, and Kibana, used for centralising and analysing log data.

Project Flow:

- Install all the required packages (explained above).
- Create a maven project using IntelliJ.
- Write the java code for Calculator and write test cases in the test file. Also adding the dependencies in the pom.xml.
- Creating a jar file using mvn clean install.
- Pushing the repository to github.
- Creating a repository in dockerhub, building the docker image and pushing the code along with Dockerfile.
- Pulling the docker image and deploying using ansible.
- In this project, we are automating these steps using the jenkins pipeline.

Final Directory Structure:



- This is the final structure of the project.
- The ScientificCalculator.java file contains the code for the calculator.
- The pom.xml file contains the dependencies.
- The log4j2.xml file of resources contains the format on how to write the log files.
- The ScientificCalculatorTest.java file contains the test cases for each of the functions.

```java
public class ScientificCalculatorTest {

    ScientificCalculator sc = new ScientificCalculator();

    @Test
    public void testSquareRoot() {
        assertEquals(2.0, sc.squareRoot(4),0.0f);
        assertEquals(3.0, sc.squareRoot(9),0.0f);
        assertEquals(4.0, sc.squareRoot(16),0.0f);
        assertNotEquals(15.0, sc.squareRoot(25),0.0f);
        assertNotEquals(9.0, sc.squareRoot(36),0.0f);
    }

    @Test
    public void testFactorial() {
        assertEquals(1, sc.factorial(0),0.0f);
        assertNotEquals(10, sc.factorial(1),0.0f);
        assertEquals(6, sc.factorial(3),0.0f);
        assertNotEquals(204, sc.factorial(4),0.0f);
        assertEquals(720, sc.factorial(6),0.0f);
    }

    @Test
    public void testNaturalLogarithm() {
        assertEquals(1.0, sc.naturalLogarithm(Math.E),0.2f);
        assertEquals(0.0, sc.naturalLogarithm(1),0.2f);
        assertNotEquals(5.6931, sc.naturalLogarithm(2),0.2f);
        assertNotEquals(11.3862, sc.naturalLogarithm(4),0.2f);
        assertEquals(2.3026, sc.naturalLogarithm(10),0.2f);
    }

    @Test
    public void testPower() {
        assertEquals(4.0, sc.power(2, 2),0.0f);
        assertNotEquals(108.0, sc.power(2, 3),0.0f);
        assertEquals(9.0, sc.power(3, 2),0.0f);
        assertNotEquals(89.0, sc.power(3, 4),0.0f);
        assertEquals(100.0, sc.power(10, 2),0.0f);
```

Maven Build

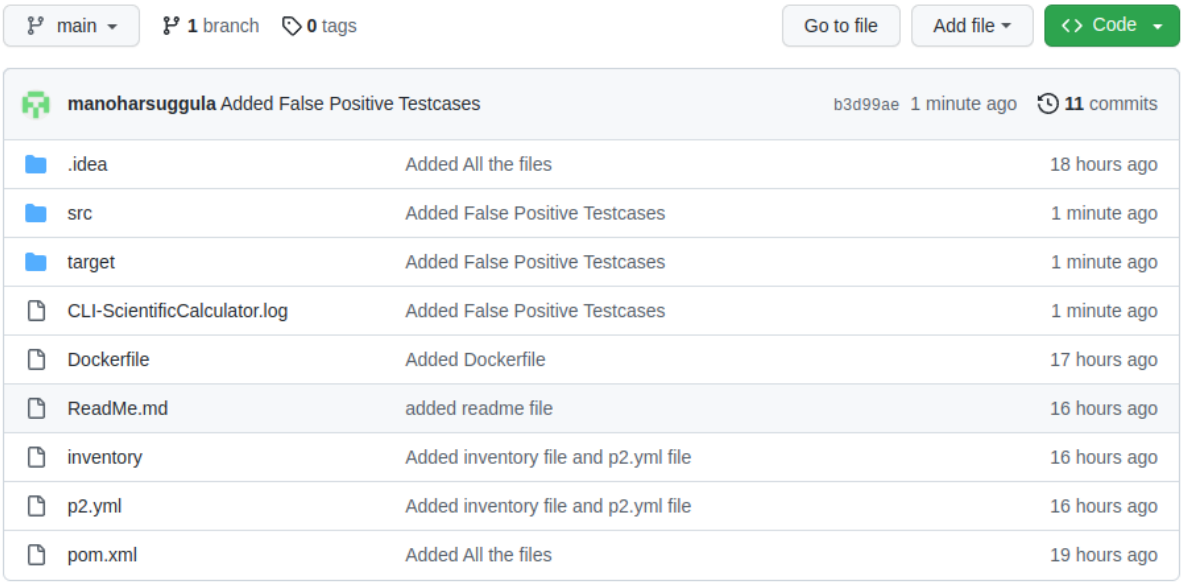
- After the code is written, we create a jar file from the code using
  - $mvn clean install.

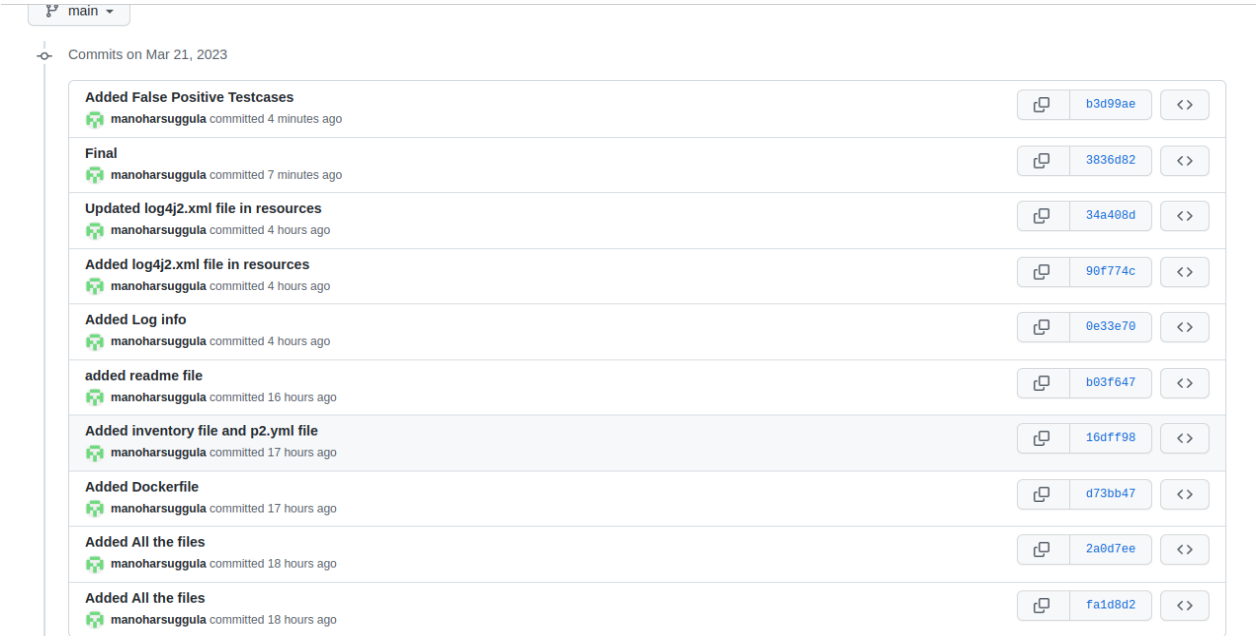
```
chakradar@reddy:~/Desktop/ScientificCalculator$ mvn clean install
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for org.example:ScientificCalculator:jar:1.0-SNAPSHOT
[WARNING] 'dependencies.dependency.version' for org.junit.jupiter:junit-jupiter:jar is either LATEST or RELEASE (both of them are being deprec
ated) @ line 20, column 22
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] ------------------< org.example:ScientificCalculator >------------------
[INFO] Building ScientificCalculator 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ ScientificCalculator ---
[INFO] Deleting /home/chakradar/Desktop/ScientificCalculator/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ ScientificCalculator ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ ScientificCalculator ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/chakradar/Desktop/ScientificCalculator/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ ScientificCalculator ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/chakradar/Desktop/ScientificCalculator/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ ScientificCalculator ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/chakradar/Desktop/ScientificCalculator/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ ScientificCalculator ---
```

- After the jar file is created, we upload the whole repo to github to the main branch.

```
chakradar@reddy:~/Desktop/ScientificCalculator$ git init
Reinitialized existing Git repository in /home/chakradar/Desktop/ScientificCalculator/.git/
chakradar@reddy:~/Desktop/ScientificCalculator$ git add .
chakradar@reddy:~/Desktop/ScientificCalculator$ git commit -m "Added False Positive Testcases"
[main b3d99ae] Added False Positive Testcases
 7 files changed, 54 insertions(+), 14 deletions(-)
 rewrite target/test-classes/ScientificCalculatorTest.class (79%)
chakradar@reddy:~/Desktop/ScientificCalculator$ git push origin main
Username for 'https://github.com': manoharsuggula
Password for 'https://manoharsuggula@github.com':
Enumerating objects: 47, done.
Counting objects: 100% (47/47), done.
Delta compression using up to 8 threads
Compressing objects: 100% (27/27), done.
Writing objects: 100% (29/29), 23.09 KiB | 259.00 KiB/s, done.
Total 29 (delta 17), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (17/17), completed with 10 local objects.
To https://github.com/manoharsuggula/ScientificCalculator.git
   34a408d..b3d99ae  main -> main
chakradar@reddy:~/Desktop/ScientificCalculator$
```

- The github repo looks like this

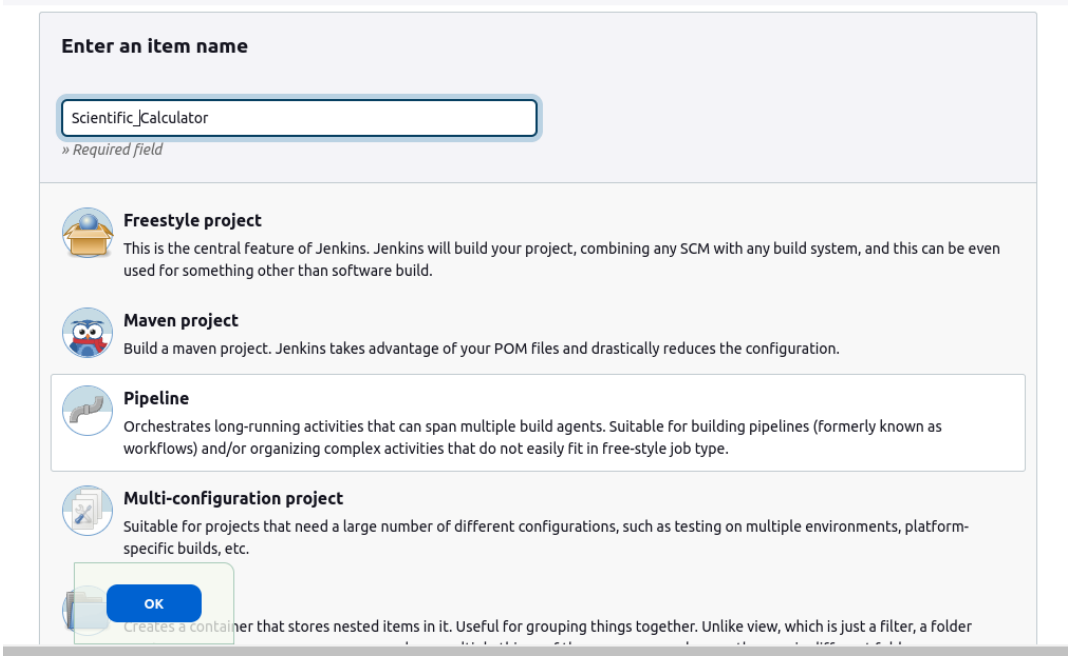




The above are the list of the commits.

Jenkins:

- Now that we created the jar file from the code, we can now start the jenkins pipeline.
- First create a job (pipeline) in jenkins as shown



- We then install all the required plugins.
- We then write the pipeline script for testing and building using maven.

Script ?

```
pipeline {
    environment{
        imageName = ""
    }
    agent any

    stages {
        stage('Git Pull stage') {
            steps {
                // Get some code from a GitHub repository
                git url:'https://github.com/manoharsuggula/ScientificCalculator',
                    branch:"main"
            }
        }
        stage('Maven Test'){
            steps{
                script{
                    sh 'mvn test'


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

                }
            }
        }
```

- The build output is

#2 Mar 21 02:21 No Changes	7s	8s	9s
#1 Mar 21 01:31 No Changes	4s	29s	21s

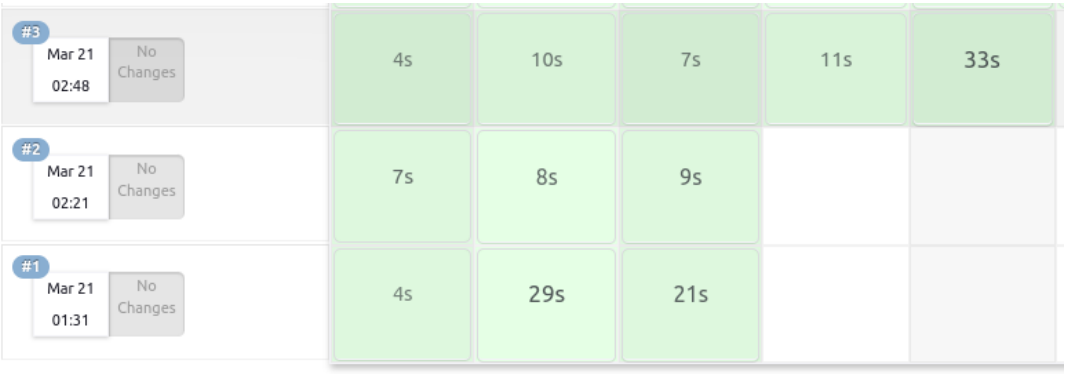
Here, the pipeline will pull the code from github, test using mvn test and build the jar file using mvn clean install.

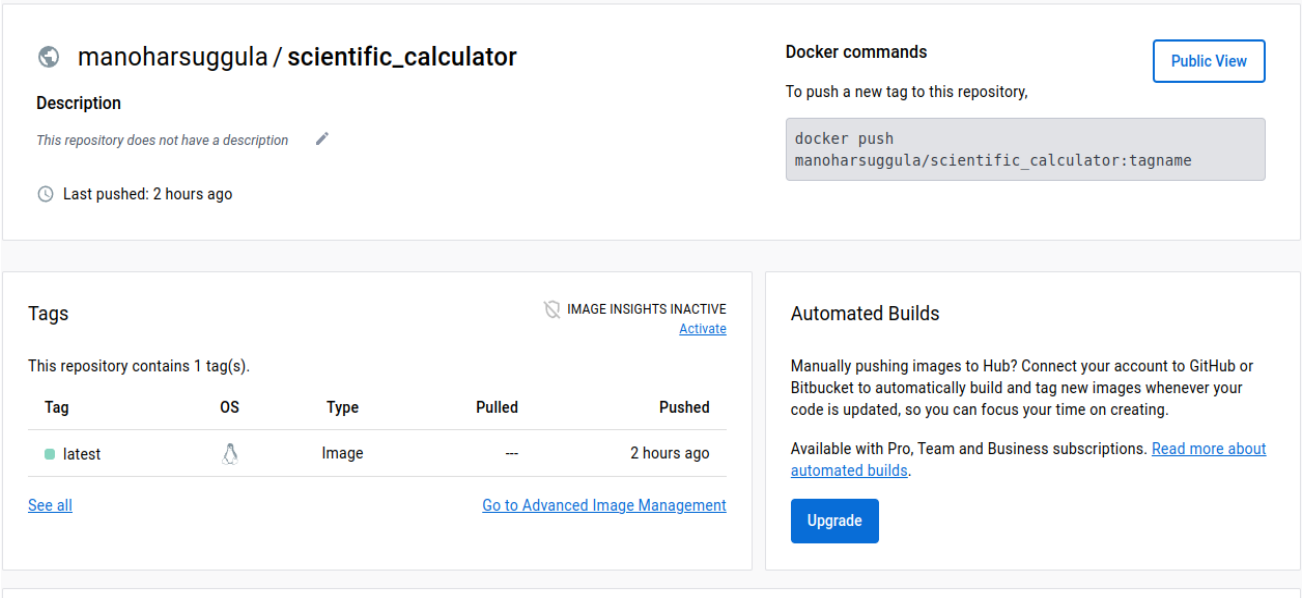
Docker:

- Create an account in dockerhub and create a repository.
- Locally using the jar file from mvn clean install, we create a docker image using the following commands after logging in
    - sudo docker login
    - sudo docker start service
    - sudo docker build -t manoharsuggula/scientific_calculator:latest .
    - sudo docker run -it manoharsuggula/scientific_calculator:latest
    - sudo docker push manoharsuggula/scientific_calculator:latest
- It will create the docker image and push it to dockerhub.
- In jenkins we do this in the jenkins pipeline.
- The pipeline script of docker code is

```
            script{
                sh 'mvn clean install'

            }
        }
    }
    stage('Docker Build Image')
    {
        steps{
            script{
                imageName = docker.build "manoharsuggula/scientific_calculator:latest"
            }
        }
    }
    stage('Push Docker Image')
    {
        steps{
            script{
                docker.withRegistry("", 'docker_credentials' ){
                    imageName.push()
                }
            }
        }
    }
```

The build output is





Docker pipeline console output

```
+ docker build -t manoharsuggula/scientific_calculator:latest .
Sending build context to Docker daemon  9.742MB

Step 1/4 : FROM openjdk:11
 ---> 47a932d998b7
Step 2/4 : COPY ./target/ScientificCalculator-1.0-SNAPSHOT-jar-with-dependencies.jar ./
 ---> 8061efee35b0
Step 3/4 : WORKDIR ./
 ---> Running in 4e5c98b09d65
Removing intermediate container 4e5c98b09d65
 ---> 4dac22eda26d
Step 4/4 : CMD ["java", "-jar", "ScientificCalculator-1.0-SNAPSHOT-jar-with-dependencies.jar"]
 ---> Running in d3be4ca6e31c
Removing intermediate container d3be4ca6e31c
 ---> de2c2aa9e6ea
Successfully built de2c2aa9e6ea
Successfully tagged manoharsuggula/scientific_calculator:latest
```

Docker push console output

```
+ docker push manoharsuggula/scientific_calculator:latest
The push refers to repository [docker.io/manoharsuggula/scientific_calculator]
cf525867f615: Preparing
7b7f3078e1db: Preparing
826c3ddbb29c: Preparing
b626401ef603: Preparing
9b55156abf26: Preparing
293d5db30c9f: Preparing
03127cdb479b: Preparing
9c742cd6c7a5: Preparing
293d5db30c9f: Waiting
03127cdb479b: Waiting
9c742cd6c7a5: Waiting
826c3ddbb29c: Layer already exists
7b7f3078e1db: Layer already exists
b626401ef603: Layer already exists
9b55156abf26: Layer already exists
293d5db30c9f: Layer already exists
03127cdb479b: Layer already exists
9c742cd6c7a5: Layer already exists
cf525867f615: Pushed
latest: digest: sha256:717c7c1544a47a2d81287f370400581c132558376671f54d9fba49ecb6183284 size: 2006
```

The above images show the pipeline script, build outputs and console outputs after adding docker stages to the pipeline. In the pipeline script, we first build the docker image from the jar file. We then push the docker image.

This is done using theDockerfile.

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

It copies the jar file in the target directory and creates a docker image to the working directory.

The list of docker images.

```
chakradar@reddy:~/Desktop/ScientificCalculator$ sudo docker images
REPOSITORY                              TAG       IMAGE ID       CREATED        SIZE
manoharsuggula/scientific_calculator    latest    de2c2aa9e6ea   3 hours ago    656MB
manoharsuggula/scientific_calculator    <none>    5dfd0e902b1b   6 hours ago    656MB
manoharsuggula/scientific_calculator    <none>    14b541341d71   7 hours ago    656MB
manoharsuggula/scientific_calculator    <none>    1cf7b1dd4ec6   7 hours ago    656MB
manoharsuggula/scientific_calculator    <none>    d7aa1f532700   19 hours ago   656MB
manoharsuggula/scientific_calculator    <none>    148215e8ab05   19 hours ago   656MB
manoharsuggula/scientific_calculator    <none>    c3adb8b1113b   19 hours ago   656MB
manoharsuggula/scientific_calculator    <none>    f7170a87b758   20 hours ago   656MB
ubuntu                                  latest    08d22c0ceb15   13 days ago    77.8MB
openjdk                                 11        47a932d998b7   7 months ago   654MB
chakradar@reddy:~/Desktop/ScientificCalculator$
```

We can run any image using

```
chakradar@reddy:~/Desktop/ScientificCalculator$ sudo docker images
REPOSITORY                                  TAG       IMAGE ID       CREATED        SIZE
manoharsuggula/scientific_calculator        latest    de2c2aa9e6ea   4 hours ago    656MB
manoharsuggula/scientific_calculator        <none>    5dfd0e902b1b   7 hours ago    656MB
manoharsuggula/scientific_calculator        <none>    14b541341d71   7 hours ago    656MB
manoharsuggula/scientific_calculator        <none>    1cf7b1dd4ec6   7 hours ago    656MB
manoharsuggula/scientific_calculator        <none>    d7aa1f532700   19 hours ago   656MB
manoharsuggula/scientific_calculator        <none>    148215e8ab05   19 hours ago   656MB
manoharsuggula/scientific_calculator        <none>    c3adb8b1113b   19 hours ago   656MB
manoharsuggula/scientific_calculator        <none>    f7170a87b758   20 hours ago   656MB
ubuntu                                      latest    08d22c0ceb15   13 days ago    77.8MB
openjdk                                     11        47a932d998b7   7 months ago   654MB
chakradar@reddy:~/Desktop/ScientificCalculator$ sudo docker run -it manoharsuggula/scientific_calculator:latest
Select an operation:
1. Square root
2. Natural log
3. Factorial
4. Power function
0. Exit
```

Ansible:

- Developers and IT teams can automate difficult tasks with the help of Ansible, which uses a declarative language to describe infrastructure as code. This boosts operational efficiency and consistency.
- Here, we write an ansible playbook, which contains the set of instructions, which are written in YAML format. It specifies the series of tasks to be executed in series on remote hosts. Playbooks are the primary mechanism used in Ansible to orchestrate complex IT automation workflows.

```
! p2.yml
1   ---
2   - name: pulling scientific_calculator docker image
3     hosts: all
4     tasks:
5             - name: Pulling junit devops image
6               docker_image:
7                     name: manoharsuggula/scientific_calculator
8                     source: pull
9
10            - name: creating an updated container
11              shell: docker run -id manoharsuggula/scientific_calculator
```

- Then we write an inventory file to determine which hosts to target when running playbooks or executing commands. An inventory file is typically written in INI or YAML format and contains information such as the hostname, IP address, and connection details of each host.

```
inventory
1   localhost ansible_user=chakradar ansible_connection=local ansible_python_interpreter=/usr/bin/python3
```

- In the Jenkins pipeline, we add a stage for ansible deployment.

```
        steps{
            script{
                docker.withRegistry("", 'docker_credentials' ){
                    imageName.push()
                }
            }
        }
    }
    stage('Ansible pull docker image')
    {
        steps{

            sh "ansible-playbook p2.yml -i inventory"
        }
    }

}
```

Here, it pulls the docker image (specified in playbook), runs the ansible playbook and the inventory specifies the hosts the ansible will target.

- The build output is

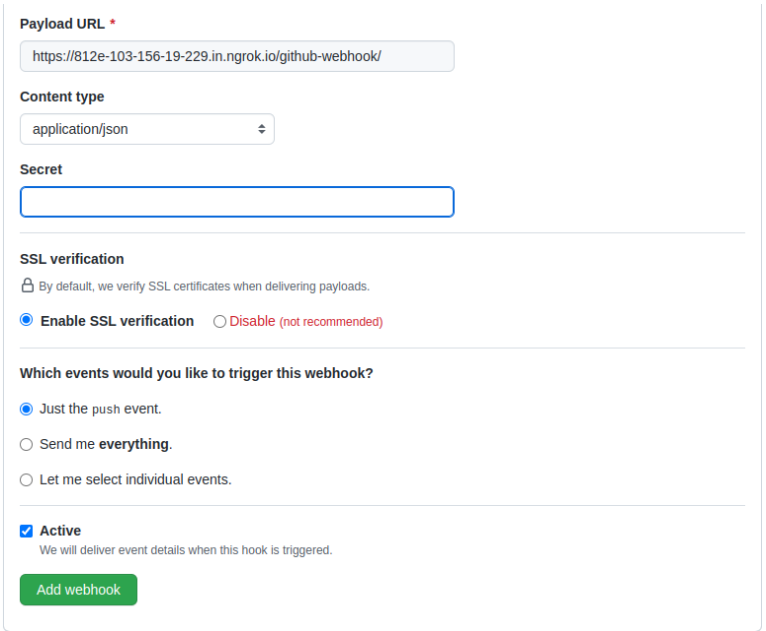
#5 Mar 21 03:23 1 commit	3s	10s	9s	10s	28s	10s
#4 Mar 21 03:03 1 commit	2s	8s	9s	10s	28s	11s
#3 Mar 21 02:48 No Changes	4s	10s	7s	11s	33s	
#2 Mar 21 02:21 No Changes	7s	8s	9s			
#1 Mar 21 01:31 No Changes	4s	29s	21s			

Ansible console output

```
+ ansible-playbook p2.yml -i inventory

PLAY [pulling scientific_calculator docker image] ****************************

TASK [Gathering Facts] *******************************************************
ok: [localhost]

TASK [Pulling junit devops image] ********************************************
ok: [localhost]

TASK [creating an updated container] *****************************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Webhooks:

- We use webhooks to trigger the jenkins pipeline whenever there is an update in the code(git push).
- We use GitHub hook triggers for GITScm polling. WE used ngrok to trigger the pipeline.
- First we need to run ngrok in local, then we create a webhook in github for the current repository with ngrok forwarding link.



```
ngrok

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

Session Status                online
Account                       manoharsuggula (Plan: Free)
Version                       3.2.1
Region                        India (in)
Latency                       24ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://812e-103-156-19-229.in.ngrok.io -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       1       0.00    0.00    0.00    0.00

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

POST /github-webhook/          200 OK
```

ELK Stack:

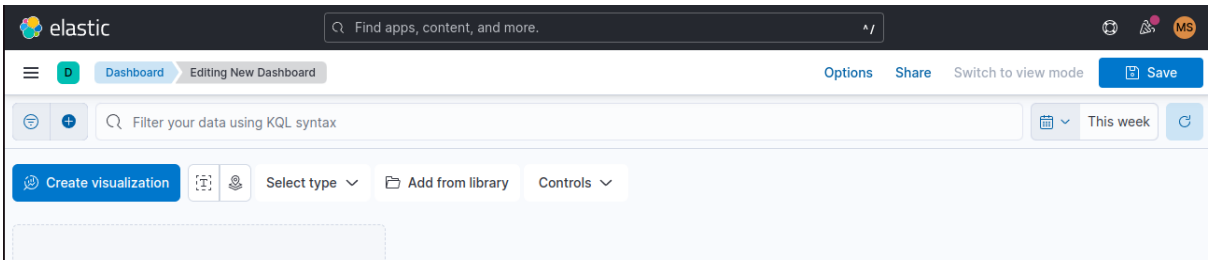
ELK Stack is a comprehensive platform for developers and IT teams to gather, process, and analyse log data, making it simpler to troubleshoot problems, spot patterns, and enhance system performance.

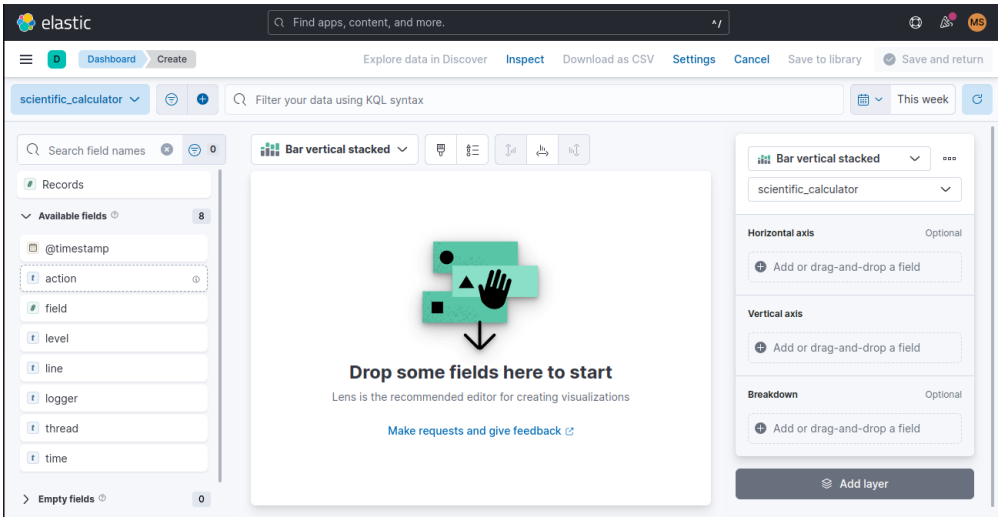
This is the log file

```
CLI-ScientificCalculator.log
21/03/2023:18:24:44 852 [ScientificCalculator.java] [INFO] ScientificCalculator [SQRT] - 4.0
21/03/2023:18:24:44 859 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - SQRT] - 2.0
21/03/2023:18:24:44 864 [ScientificCalculator.java] [INFO] ScientificCalculator [SQRT] - 9.0
21/03/2023:18:24:44 865 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - SQRT] - 3.0
21/03/2023:18:24:44 865 [ScientificCalculator.java] [INFO] ScientificCalculator [SQRT] - 16.0
21/03/2023:18:24:44 866 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - SQRT] - 4.0
21/03/2023:18:24:44 867 [ScientificCalculator.java] [INFO] ScientificCalculator [SQRT] - 25.0
21/03/2023:18:24:44 868 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - SQRT] - 5.0
21/03/2023:18:24:44 868 [ScientificCalculator.java] [INFO] ScientificCalculator [SQRT] - 36.0
21/03/2023:18:24:44 869 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - SQRT] - 6.0
21/03/2023:18:24:44 870 [ScientificCalculator.java] [INFO] ScientificCalculator [LOG] - 2.718281828459045
21/03/2023:18:24:44 872 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - LOG] - 1.0
21/03/2023:18:24:44 872 [ScientificCalculator.java] [INFO] ScientificCalculator [LOG] - 1.0
21/03/2023:18:24:44 873 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - LOG] - 0.0
21/03/2023:18:24:44 874 [ScientificCalculator.java] [INFO] ScientificCalculator [LOG] - 2.0
21/03/2023:18:24:44 874 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - LOG] - 0.6931471805599453
21/03/2023:18:24:44 875 [ScientificCalculator.java] [INFO] ScientificCalculator [LOG] - 4.0
21/03/2023:18:24:44 876 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - LOG] - 1.3862943611198906
21/03/2023:18:24:44 877 [ScientificCalculator.java] [INFO] ScientificCalculator [LOG] - 10.0
21/03/2023:18:24:44 878 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - LOG] - 2.302585092994046
21/03/2023:18:24:44 879 [ScientificCalculator.java] [INFO] ScientificCalculator [FACTORIAL] - 0
21/03/2023:18:24:44 881 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - FACTORIAL] - 1
21/03/2023:18:24:44 882 [ScientificCalculator.java] [INFO] ScientificCalculator [FACTORIAL] - 1
21/03/2023:18:24:44 883 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - FACTORIAL] - 1
21/03/2023:18:24:44 884 [ScientificCalculator.java] [INFO] ScientificCalculator [FACTORIAL] - 3
21/03/2023:18:24:44 885 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - FACTORIAL] - 6
21/03/2023:18:24:44 886 [ScientificCalculator.java] [INFO] ScientificCalculator [FACTORIAL] - 4
21/03/2023:18:24:44 887 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - FACTORIAL] - 24
21/03/2023:18:24:44 887 [ScientificCalculator.java] [INFO] ScientificCalculator [FACTORIAL] - 6
21/03/2023:18:24:44 889 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - FACTORIAL] - 720
21/03/2023:18:24:44 903 [ScientificCalculator.java] [INFO] ScientificCalculator [POWER] - 2.0, 2.0
21/03/2023:18:24:44 905 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - POWER] - 4.0
21/03/2023:18:24:44 906 [ScientificCalculator.java] [INFO] ScientificCalculator [POWER] - 2.0, 3.0
21/03/2023:18:24:44 906 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - POWER] - 8.0
21/03/2023:18:24:44 907 [ScientificCalculator.java] [INFO] ScientificCalculator [POWER] - 3.0, 2.0
21/03/2023:18:24:44 908 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - POWER] - 9.0
21/03/2023:18:24:44 908 [ScientificCalculator.java] [INFO] ScientificCalculator [POWER] - 3.0, 4.0
21/03/2023:18:24:44 909 [ScientificCalculator.java] [INFO] ScientificCalculator [RESULT - POWER] - 81.0
21/03/2023:18:24:44 910 [ScientificCalculator.java] [INFO] ScientificCalculator [POWER] - 10.0, 2.0
```

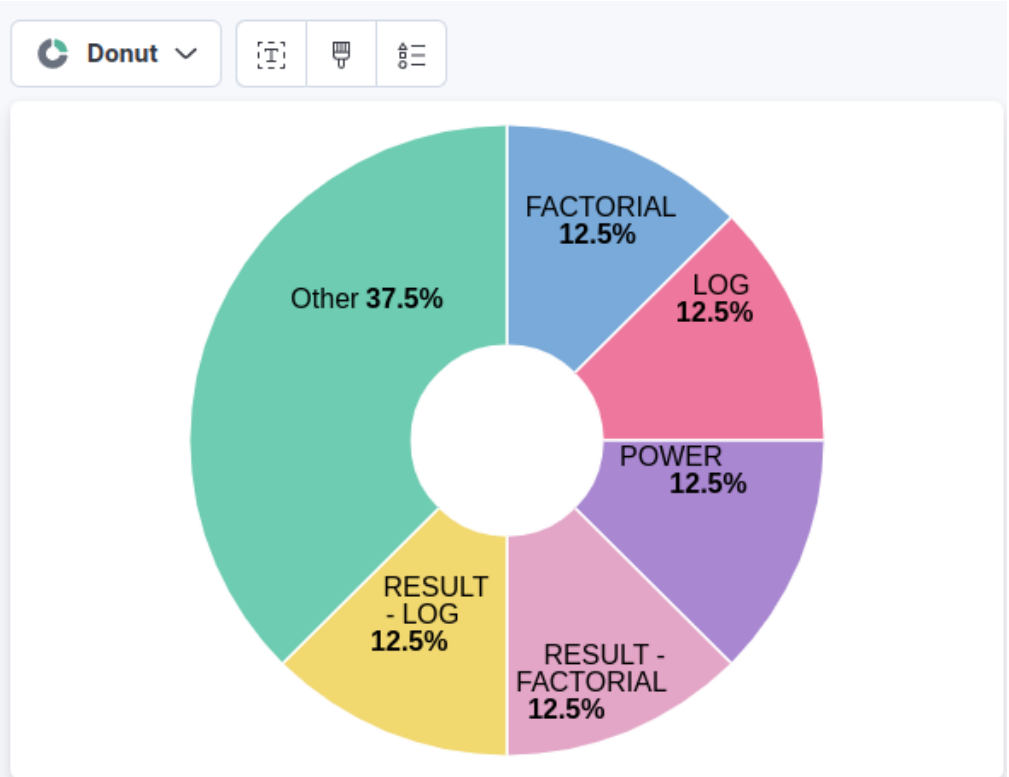
We can create different visualisations as follows

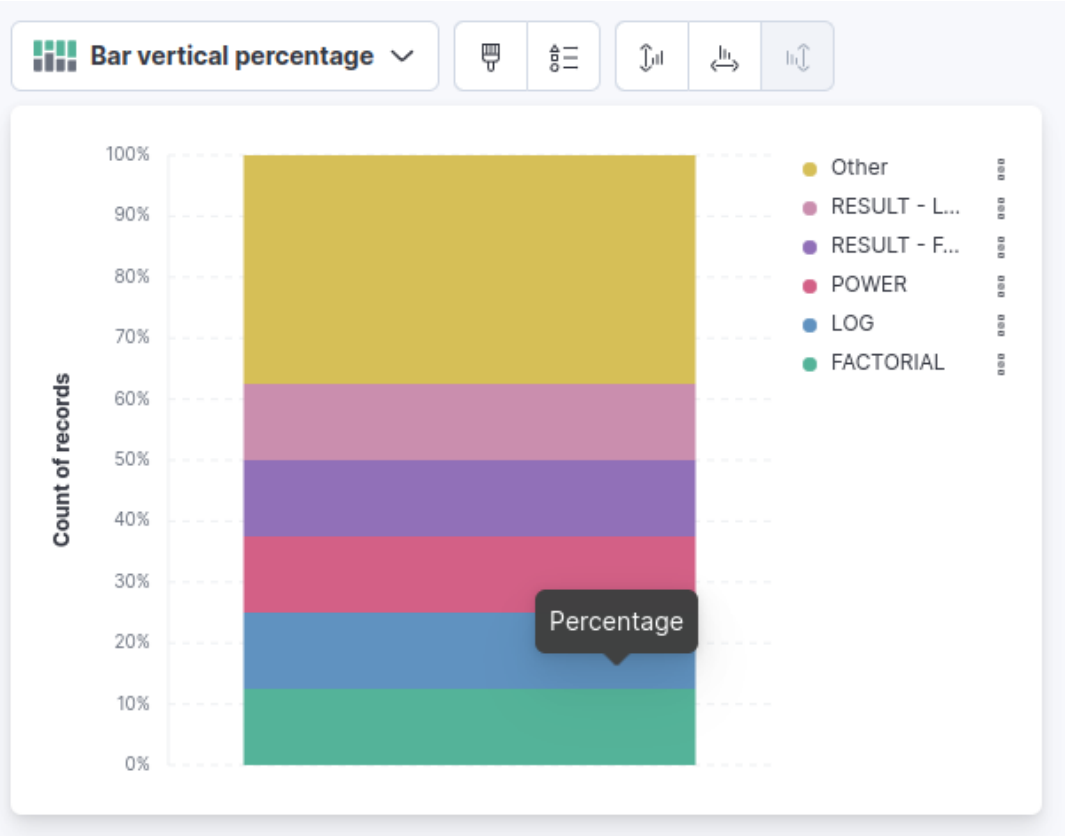
Create Visualization -> Drag and Drop Action

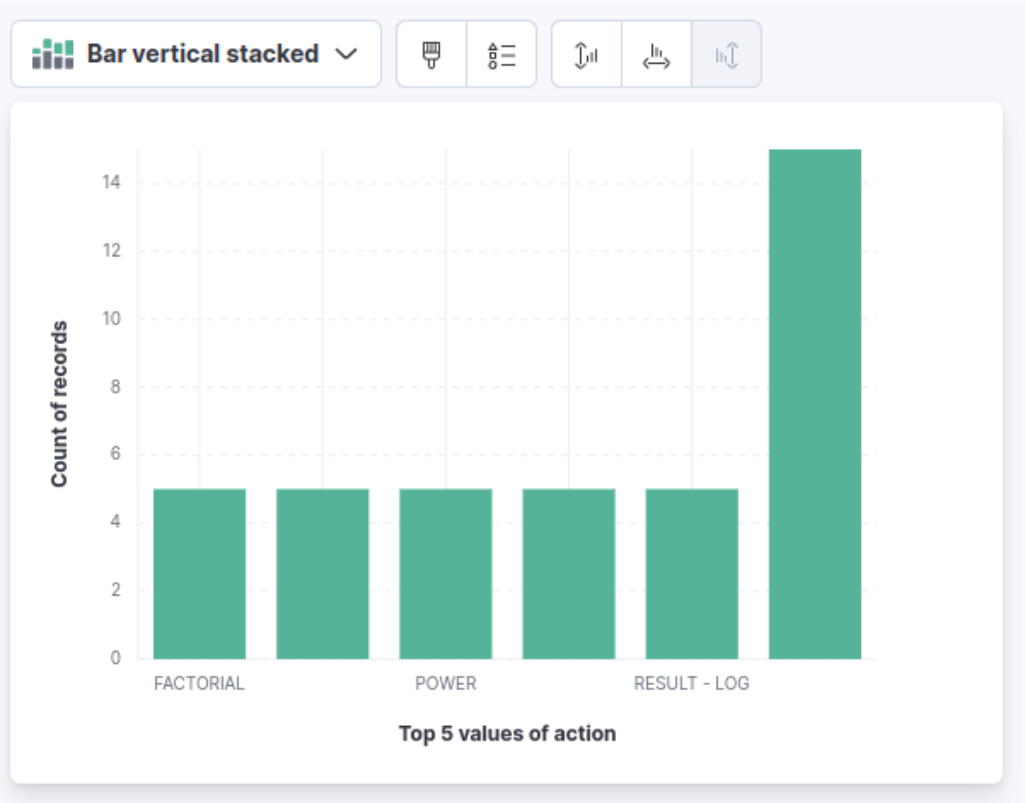




Then we get the visualizations as follows




Bar vertical percentage
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Count of records
Percentage
Other
RESULT - L...
RESULT - F...
POWER
LOG
FACTORIAL


Bar vertical stacked
14
12
10
8
6
4
2
0
Count of records
FACTORIAL
POWER
RESULT - LOG
Top 5 values of action