Calculator - Devops Mini Project

Software Production Engineering

Nachiappan S K - IMT2018047

April 17, 2022

Goal:

The goal of the project is to build a calculator with the help of automation tools for integrating and deployment. The automation includes source code management(SCM), continuous integration, continuous deployment and configuration management for the whole source code.This is done by implementing some of the devops technologies that help in the automation of the activities listed above.The project involves using devops tools such as github, maven, junit, jenkins, ansible and docker to create a scientific calculator programme that includes arithmetic operations such as addition, subtraction, multiplication, division and Log operations and exponential operations. The project's main goal is to learn the devops concepts of CI/CD/CM through the creation of a Jenkins pipeline.

What is DevOps?

DevOps is a method which is used to reduce the conflicts between the developers(dev) and the operations(ops) team. It's a software engineering methodology that aims to combine the activities of software development and software operations teams by encouraging collaboration and shared responsibility.

Why DevOps?

Often there are problems between the developers and the operations teams due to the nature of their work. As the developers team strive to make changes for introducing new features to the application and the operations team doesn't want change as change might cause harm to the system if failed and thus introducing downtime for the application. Organizations that use DevOps to evolve and improve software products far faster than those that use traditional software development methods. DevOps is a bridge between development and operations in a company with the goal of increasing overall productivity. The gap between the development and the operations is filled by the devops team. Thus enabling the path for the testers and operators to work more efficiently. Advantages of using DevOps:

- Faster and continuous delivery.
- Stable operations

- Higher quality
- Early detection of errors
- Reduced general costs
- Less failure rate in implementation
- Greater communication between teams
- Better traceability

Tools Used:

- Git: version control system
- Maven: Build tool
- JUnit: Testing
- Jenkins: CI/CD pipeline
- Ansible: Configuration management and infrastructure as code
- Docker: Deployment/ containerisation

Installation of tools:

Prerequisite: Java-11

- Sudo apt-get update
- Sudo apt install openjdk-11-jre-headless
- Java -version

```
NachiappanSK@g3-3579[18:29:01]:~$ java -version
openjdk version "11.0.14.1" 2022-02-08
OpenJDK Runtime Environment (build 11.0.14.1+1-Ubuntu-0ubuntu1.18.04)
OpenJDK 64-Bit Server VM (build 11.0.14.1+1-Ubuntu-0ubuntu1.18.04, mixed mode, sharing)
NachiappanSK@g3-3579[18:29:03]:~$
```

- If it doesn't work java --version works

```
NachiappanSK@g3-3579[18:30:27]:~$ java --version
openjdk 11.0.14.1 2022-02-08
OpenJDK Runtime Environment (build 11.0.14.1+1-Ubuntu-0ubuntu1.18.04)
OpenJDK 64-Bit Server VM (build 11.0.14.1+1-Ubuntu-0ubuntu1.18.04, mixed mode, sharing)
NachiappanSK@g3-3579[18:30:31]:~$
```

- Since some other version of java has been active the java version has to be changed using update-alternatives command
- Sudo update-alternatives --config java

```
NachiappanSK@g3-3579[18:32:38]:~$ sudo update-alternatives --config java
There are 3 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                          Priority   Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-17-openjdk-amd64/bin/java      1711      auto mode
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111      manual mode
* 2            /usr/lib/jvm/java-17-openjdk-amd64/bin/java      1711      manual mode
  3            /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java   1081      manual mode

Press <enter> to keep the current choice[*], or type selection number: 1
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/java to provide /usr/bin/java (java) in manual mode
NachiappanSK@g3-3579[18:32:41]:~$
```

Git:

- Sudo apt-get update
- Sudo apt-get install git
- git --version

```
NachiappanSK@g3-3579[18:34:43]:~$ git --version
git version 2.17.1
NachiappanSK@g3-3579[18:34:47]:~$
```

Maven:

- Sudo apt-get update
- Sudo apt-get install maven
- mvn -v

```
NachiappanSK@g3-3579[18:34:47]:~$ mvn -v
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 1.8.0_312, vendor: Private Build, runtime: /usr/lib/jvm/java-8-openjdk-amd64/jre
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-175-generic", arch: "amd64", family: "unix"
NachiappanSK@g3-3579[18:37:26]:~$
```

Jenkins:

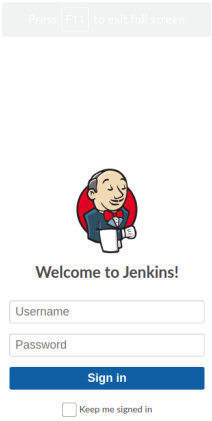
- wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
- sudo sh -c 'echo deb https://pkg.jenkins.io/debian binary/ > /etc/apt/sources.list.d/jenkins.list'
- sudo apt install ca-certificates
- sudo apt-get update
- sudo apt-get install jenkins
- sudo service jenkins status

```
NachiappanSK@g3-3579[21:34:39]:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: failed (Result: exit-code) since Sun 2022-04-17 21:28:17 IST; 6min ago
  Process: 3158 ExecStart=/usr/bin/jenkins (code=exited, status=1/FAILURE)
 Main PID: 3158 (code=exited, status=1/FAILURE)

Apr 17 21:28:17 g3-3579 systemd[1]: jenkins.service: Service hold-off time over, scheduling restart.
Apr 17 21:28:17 g3-3579 systemd[1]: jenkins.service: Scheduled restart job, restart counter is at 6.
Apr 17 21:28:17 g3-3579 systemd[1]: Stopped Jenkins Continuous Integration Server.
Apr 17 21:28:17 g3-3579 systemd[1]: jenkins.service: Start request repeated too quickly.
Apr 17 21:28:17 g3-3579 systemd[1]: jenkins.service: Failed with result 'exit-code'.
Apr 17 21:28:17 g3-3579 systemd[1]: Failed to start Jenkins Continuous Integration Server.
```

- sudo service jenkins start
- sudo service jenkins status

```
NachiappanSK@g3-3579[18:42:45]:~$ sudo service jenkins status
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; v
   Active: active (running) since Mon 2022-04-18 12:40:26 IST; 6h
 Main PID: 11800 (java)
    Tasks: 83 (limit: 4915)
   CGroup: /system.slice/jenkins.service
           └─11800 /usr/bin/java -Djava.awt.headless=true -jar /us

Apr 18 18:35:15 g3-3579 jenkins[11800]: 2022-04-18 13:05:14.856+00
Apr 18 18:35:15 g3-3579 jenkins[11800]: 2022-04-18 13:05:14.856+00
Apr 18 18:35:15 g3-3579 jenkins[11800]: 2022-04-18 13:05:14.856+00
Apr 18 18:35:15 g3-3579 jenkins[11800]: 2022-04-18 13:05:14.856+00
Apr 18 18:40:14 g3-3579 jenkins[11800]: 2022-04-18 13:10:14.855+00
Apr 18 18:40:15 g3-3579 jenkins[11800]: 2022-04-18 13:10:14.856+00
Apr 18 18:40:15 g3-3579 jenkins[11800]: 2022-04-18 13:10:14.856+00
Apr 18 18:40:15 g3-3579 jenkins[11800]: 2022-04-18 13:10:14.856+00
Apr 18 18:40:15 g3-3579 jenkins[11800]: 2022-04-18 13:10:14.856+00
Apr 18 18:40:15 g3-3579 jenkins[11800]: 2022-04-18 13:10:14.856+00
lines 1-18/18 (END)
```

- Go to http://localhost:8080/ to use jenkins
- To copy admin password: sudo cat /var/lib/jenkins/secrets/initialAdminPassword
- Go through the steps in the interface.



Ansible:

- Sudo apt-get update
- Sudo apt-get install ansible
- pip3 install docker
- pip3 install ansible
- Ansible --version

```
NachiappanSK@g3-3579[18:44:15]:~$ ansible --version
ansible [core 2.12.4]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/nachiappan-senthil-kumar
/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/nachiappan-senthil-kumar/
.anaconda3/lib/python3.8/site-packages/ansible
  ansible collection location = /home/nachiappan-senthil-kumar/.an
sible/collections:/usr/share/ansible/collections
  executable location = /home/nachiappan-senthil-kumar/.anaconda3/
bin/ansible
  python version = 3.8.3 (default, Jul  2 2020, 16:21:59) [GCC 7.3
.0]
  jinja version = 2.11.2
  libyaml = True
NachiappanSK@g3-3579[18:46:21]:~$
```

Docker:

- sudo apt install apt-transport-https ca-certificates curl
- Sudo apt-get install software-properties-common
- curl -fsSL https://get.docker.com -o get-socker.sh
- bash get-docker.sh
- docker -v

```
NachiappanSK@g3-3579[18:46:21]:~$ docker -v
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3
NachiappanSK@g3-3579[18:48:56]:~$
```

Ngrok:

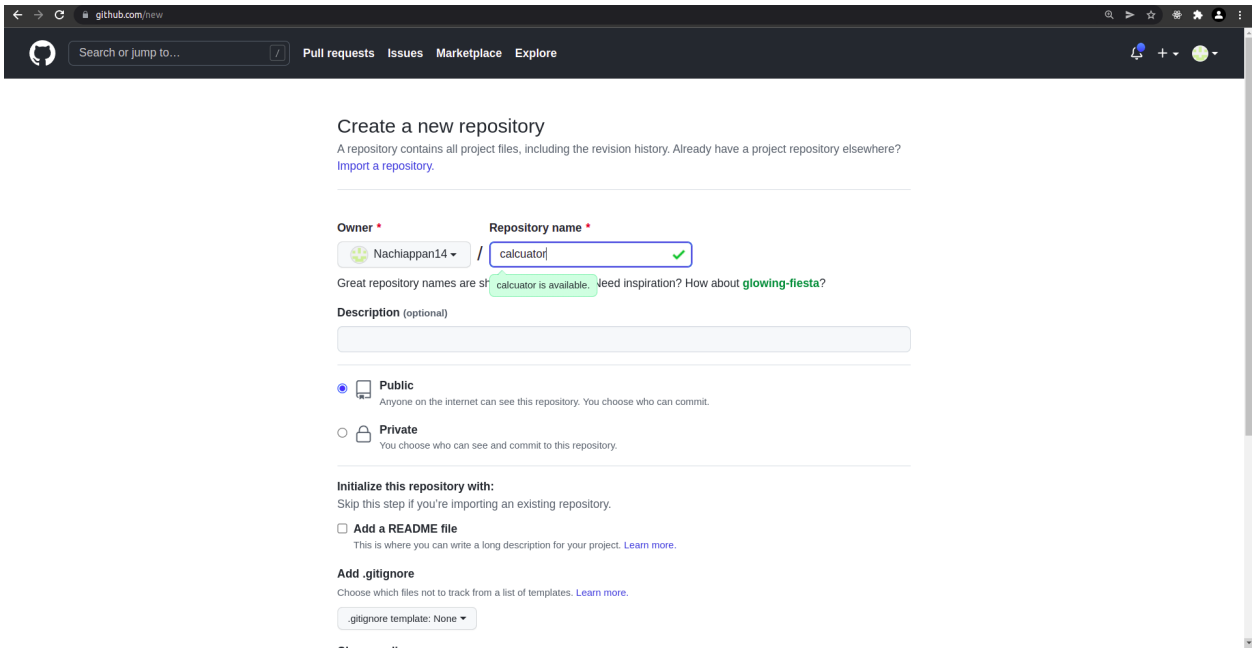
- Go to https://ngrok.com/ and download ngrok and extract in a folder
- Create a account in ngrok and add the key to the config
- Now use ngrok http 8080 to make your jenkins server public

```
nachiappan-senthil-kumar@g3-3579: ~/.ngrok
File  Edit  View  Search  Terminal  Help
ngrok                                                    (Ctrl+C to quit)

Session Status                online
Account                       Nachiappan (Plan: Free)
Version                       3.0.2
Region                        India (in)
Latency                       103.755146ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://45ce-2409-4071-211b-89a4-e820-2382-ea28-97

Connections                   ttl     opn     rt1     rt5     p50     p90
                              164     0       0.00    0.00    0.91    6.37

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

POST /github-webhook/                  200 OK
POST /github-webhook/                  200 OK
POST /github-webhook/                  200 OK
POST /github-webhook/                  200 OK
POST /github-webhook/                  200 OK
POST /job/Calculator/github-webhook/   403 Forbidden
GET  /job/Calculator/7/wfapi/changesets 200 OK
GET  /job/Calculator/wfapi/runs        200 OK
```

Project Steps:

Creating Repository with Version Control:

- Git is a tool which is highly useful for enabling version control systems, i.e., being able to keep all versions and go back to the required version of code when needed. To utilize git with any project folder, it must first be initialized as a git repository. git is used on the local system, but it is preferable to push the code to common cloud based management platform such as github, where anyone can grab the code, depending on the repository's privacy settings, modify and contribute.
- “git init” to initialize the project repository
- The other useful git commands used while making the project are
    - Git status - to see the status of the repo
    - Git add - to add the code to the staging area
    - Git commit - to commit the current version of the code
    - Git log - to see who made what changes to the code
    - Git remote - to add the remote repo link to the current repo
- Now create a repository in github



- add the link to the current local repo with the use of git remote add, example) git remote add origin <link to github repo>
- Git remote add origin https://github.com/Nachiappan14/Calculator.git

Creating a java(Maven) project:

- Using Eclipse IDE/other IDE, create a maven project
- The structure will have a pom.xml and src folder

- Create a package called calculator and code the java files as required
- The package created will be located under src/main/java
- Put the test code in the src/test/java
- Pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://
  <modelVersion>4.0.0</modelVersion>
  <groupId>calculator</groupId>
  <artifactId>calculator</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Calculator</name>
  <description>A simple terminal based calculator used for demonstration of pipeline</description>

  <dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.1</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.14.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.14.0</version>
    </dependency>
</dependencies>


  <properties>
     <maven.compiler.source>11</maven.compiler.source>
     <maven.compiler.target>11</maven.compiler.target>
</properties>
</project>
```

Building the project:

- Maven is a project management and comprehension tool that provides a full build lifecycle framework to developers. Because Maven employs an uniform directory layout and a default build lifecycle, the development team may automate the project's build infrastructure in nearly no time.
- The Project Object Model (POM), which is the essential element of the Maven system, declares the structure and contents of a Maven project in an xml file called pom.xml.
- Maven simplifies dependency management and helps the user in converting a project into a JAR file that can be moved and run on other systems.
- All the dependencies for the project are given in the pom.xml file
- Now “maven clean install” is used to build the project

```
NachiappanSK@g3-3579[19:39:09]:~/sem8/SPE/Calculator$ mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< calculator:calculator >-----------------------
-
[INFO] Building Calculator 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
-
Downloading from central: https://repo.maven.apache.org/maven2/junit/junit/4.1
3.1/junit-4.13.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/junit/junit/4.13
.1/junit-4.13.1.pom (25 kB at 5.0 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/hamcrest/ha
mcrest-core/1.3/hamcrest-core-1.3.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/hamcrest/ham
crest-core/1.3/hamcrest-core-1.3.pom (766 B at 670 B/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/hamcrest/ha
mcrest-parent/1.3/hamcrest-parent-1.3.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/hamcrest/ham
crest-parent/1.3/hamcrest-parent-1.3.pom (2.0 kB at 2.4 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/logg
ing/log4j/log4j-api/2.14.0/log4j-api-2.14.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/loggi
ng/log4j/log4j-api/2.14.0/log4j-api-2.14.0.pom (14 kB at 18 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/logg
```

- **JUnit** - Junit is a easy to use unit test tool which is used in the project. To indicate that a method is a test method, @test tag is used for the annotation.You use an assert method provided by JUnit or similar assert framework to compare an expected and actual result. The most popular names for these calls are assertions or assert statements. When using assert statements, you may normally give messages that will be displayed if the test fails. Such messages help in identifying and resolving the failures quickly
- Maven runs the JUnit tests while installing and the results are also logged as shown below.

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running calculator.TestArithmetic

40.0

14.0

7.0

-30.0

30.0
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.038 sec
Running calculator.TestLog

2.0794415416798357

1.6094379124341003
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
Running calculator.TestExponent

-8.0

3.0

125.0

64.0
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec

Results :

Tests run: 11, Failures: 0, Errors: 0, Skipped: 0
```

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  7.090 s
[INFO] Finished at: 2022-04-18T19:52:09+05:30
[INFO] ------------------------------------------------------------------------
```

- The build jar will be stored in the target folder
- To run the build jar file we use the following command
- Java -cp <jar-name> <class-path>

```
NachiappanSK@g3-3579[20:08:54]:~/sem8/SPE/Calculator$ java -cp target/calculator-0.0.1-SNAPSHOT.jar calculator.App

What would you like to do? Input a number and press enter.
1. Arithmetic
2. Exponents
3. Logarithms
4. Exit

2
Pick an operation by entering a number and pressing enter.
1. Square
2. Cube
3. Variable Exponent
4. Square Root
5. Return to Exponent Menu
6. Return to Main Menu

3
format: a power (b)
Enter a
2
Enter b
10

1024.0

What would you like to do? Input a number and press enter.
1. Arithmetic
2. Exponents
3. Logarithms
4. Exit

4
Bye!
NachiappanSK@g3-3579[20:09:11]:~/sem8/SPE/Calculator$
```

Adding github to local repository:

- Use git remote add origin

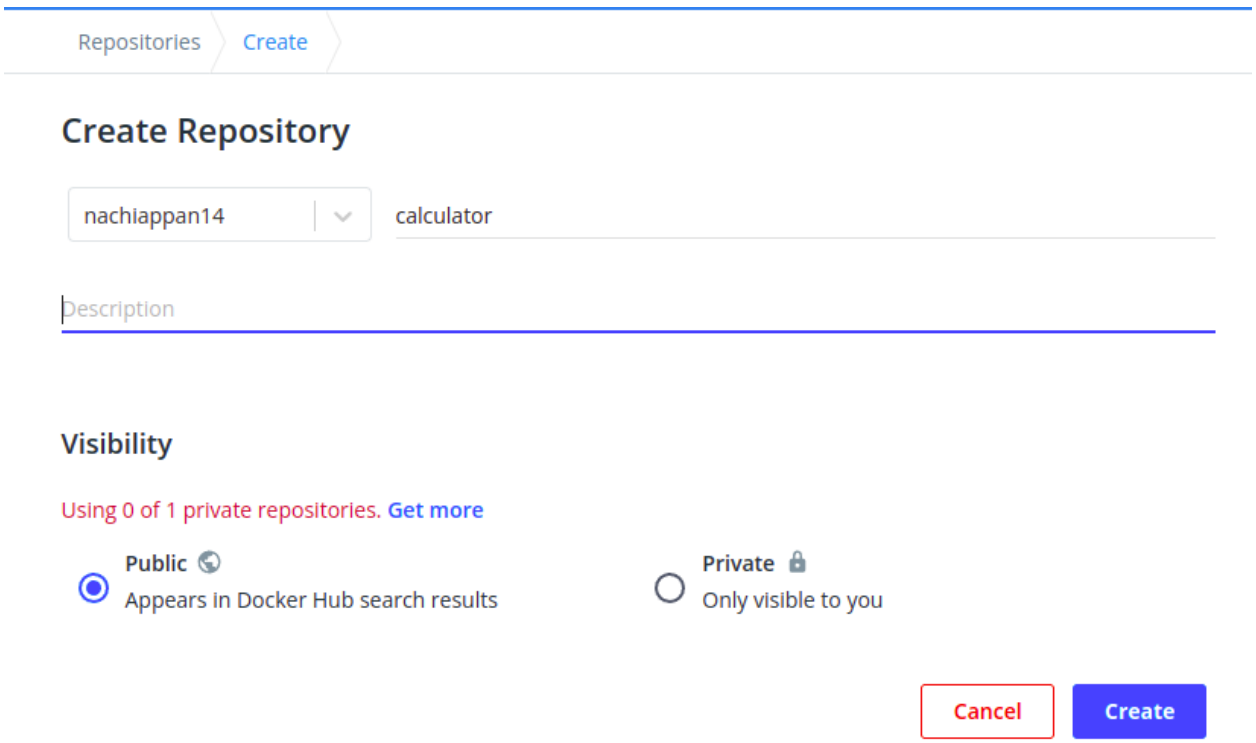
```
NachiappanSK@g3-3579[14:49:18]:~/sem8/SPE/Calculator$ git init .
Initialized empty Git repository in /home/nachiappan-senthil-kumar/sem8/SPE/Calculator/.git/
NachiappanSK@g3-3579[14:49:24]:~/sem8/SPE/Calculator$ git remote add origin https://github.com/Nachiappan14/Calculator.git
NachiappanSK@g3-3579[14:49:27]:~/sem8/SPE/Calculator$ git remote -v
origin  https://github.com/Nachiappan14/Calculator.git (fetch)
origin  https://github.com/Nachiappan14/Calculator.git (push)
NachiappanSK@g3-3579[14:49:32]:~/sem8/SPE/Calculator$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .classpath
        .project
        .settings/
        pom.xml
        src/
        target/

nothing added to commit but untracked files present (use "git add" to track)
```

- git push -u origin master to push the code.
- We can also set up the upstream for the current branch using
  git push --set-upstream origin master

```
NachiappanSK@g3-3579[14:54:37]:~/sem8/SPE/Calculator$ git push --set-upstream origin master
Username for 'https://github.com': Nachiappan14
Password for 'https://Nachiappan14@github.com':
Counting objects: 41, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (31/31), done.
Writing objects: 100% (41/41), 16.77 KiB | 5.59 MiB/s, done.
Total 41 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Nachiappan14/Calculator.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Docker

- When the software is delivered, there will be a number of challenges to address because there will be multiple dependencies and versions of dependencies to support. It will be extremely tough for the clients to resolve. Docker comes in handy here because it allows us to install all of the dependencies with the correct version and then compress them into an image.
- The image is pushed to the dockerhub, reason being similar using the github, now anyone can pull the docker image from docker hub and use the image for further developments.



- While building the image dockerfile is used to create docker image.
- Dockerfile

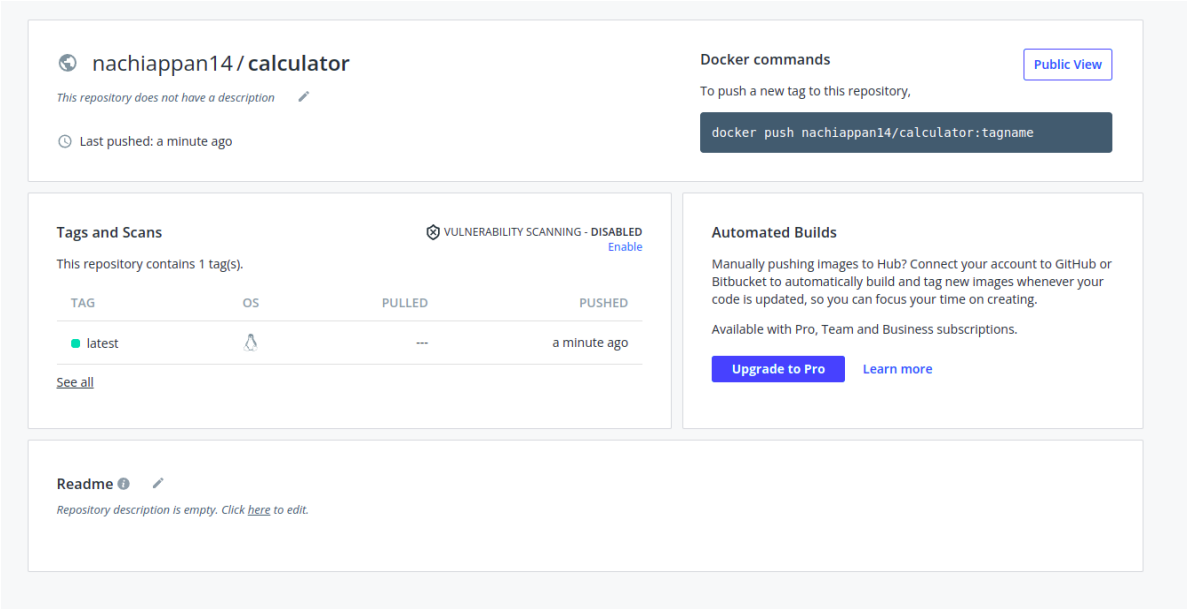
```
FROM openjdk:11
COPY ./target/calculator-0.0.1-SNAPSHOT.jar ./
COPY ./calculator.log ./
WORKDIR ./
CMD ["java","-cp","calculator-0.0.1-SNAPSHOT.jar","calculator.App"]
~
```

- docker build -t <docker-username/reponame:tagname><folderpath>

```
NachiappanSK@g3-3579[21:27:32]:~/sem8/SPE/Calculator$ docker build -t nachiappan14/calculator .
Sending build context to Docker daemon  268.8kB
Step 1/5 : FROM openjdk:11
 ---> 8c5fc4518cc2
Step 2/5 : COPY ./target/calculator-0.0.1-SNAPSHOT.jar ./
 ---> 350c869073ec
Step 3/5 : COPY ./calculator.log ./
 ---> 55cf4b6331d0
Step 4/5 : WORKDIR ./
 ---> Running in 90c37e4ba0b6
Removing intermediate container 90c37e4ba0b6
 ---> 2777781fb444
Step 5/5 : CMD ["java","-cp","calculator-0.0.1-SNAPSHOT.jar","calculator.App"]
 ---> Running in 7ebc74bf770e
Removing intermediate container 7ebc74bf770e
 ---> 96c76d388473
Successfully built 96c76d388473
Successfully tagged nachiappan14/calculator:latest
NachiappanSK@g3-3579[21:29:42]:~/sem8/SPE/Calculator$
```

- To push the image to the docker hub use the below command
- docker push <docker-username/repo-name:tagname>

```
NachiappanSK@g3-3579[21:34:49]:~/sem8/SPE/Calculator$ docker push nachiappan14/calculator:latest
The push refers to repository [docker.io/nachiappan14/calculator]
658def26eefe: Pushed
757a1c39b522: Pushed
0816d1f73744: Layer already exists
84f2cb0fc541: Layer already exists
b0dc1a441986: Layer already exists
7a7698da17f2: Layer already exists
d59769727d80: Layer already exists
348622fdcc61: Layer already exists
4ac8bc2cd0be: Layer already exists
latest: digest: sha256:df2527630d12d27a9e9e1e48389ea2b57ad040bdd1af01b0fa0fc7f701693197 size: 2210
NachiappanSK@g3-3579[21:36:40]:~/sem8/SPE/Calculator$
```



- To run the container use the below command
- docker run -it <docker-username/repo-name:tagname>

```
NachiappanSK@g3-3579[20:09:11]:~/sem8/SPE/Calculator$ docker run -it nachiappa
n14/calculator:latest

What would you like to do? Input a number and press enter.
1. Arithmetic
2. Exponents
3. Logarithms
4. Exit
```

Ansible

- Continuous deployment is a method of automating the distribution of software to a large number of client workstations. Ansible is a tool for managing configurations. But, it can be used to send anything to a large number of "controlled hosts." Ansible is installed and managed on the control node. A control node stores all copies of your Ansible project files and configuration information. Computers that deliver the application/infrastructure as code are known as managed hosts. Ansible acts as a central distributor from the control node to all managed hosts. A list of managed hosts can be seen in the inventory file.
- Here instead of using other VM or machine as host, I am using my localhost as host.
- Inventory file:

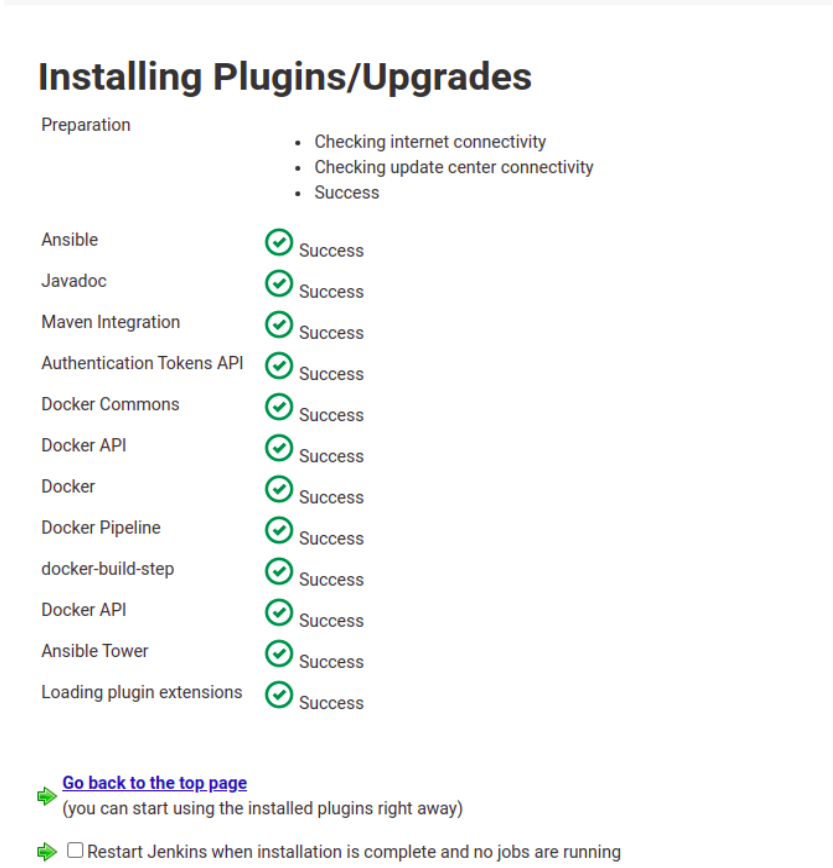
```
localhost ansible_user=nachiappan-senthil-kumar
~
```

- p2.yml

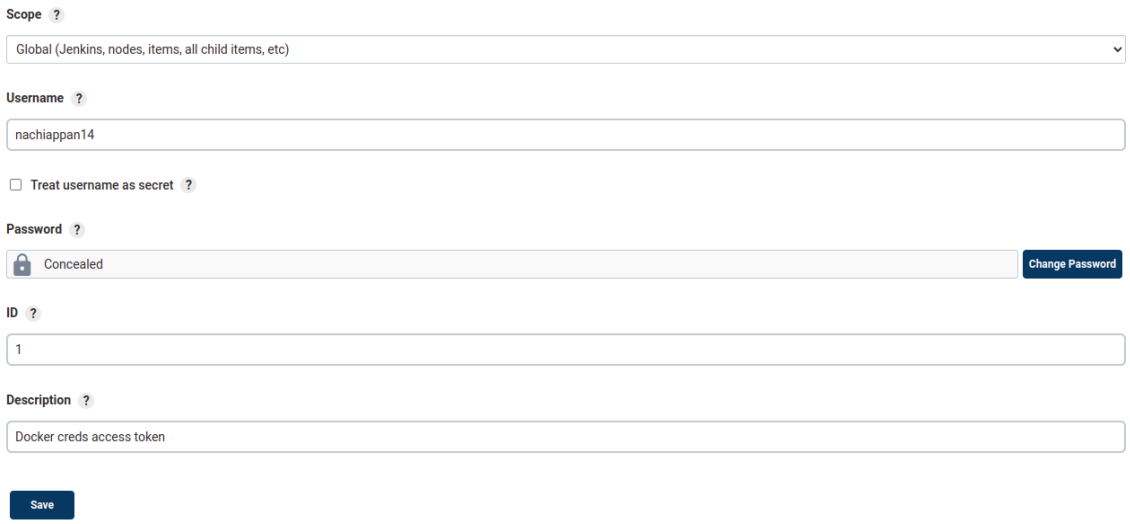
```
- name: Pull and Run docker image
  hosts: localhost
  connection: local
  vars:
    ansible_python_interpreter: /home/nachiappan-senthil-kumar/.anaconda3/bin/
python
  tasks:
    - name: Pull image
      docker_image:
        name: nachiappan14/calculator
        pull: yes
```

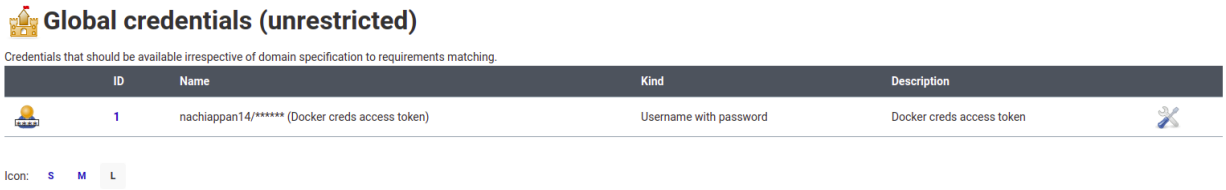
Jenkins

- Jenkins is an open source automation server that is free to use. It aids continuous integration and delivery by automating the elements of software development related to building, testing, and deploying. This provides a fantastic user interface that allows us to simply construct and customize projects or pipelines.
- We use plugins to manage the different functionality in Jenkins. We need docker, github, maven, ansible and build pipeline plugins. Thus installing all the plugins in jenkins

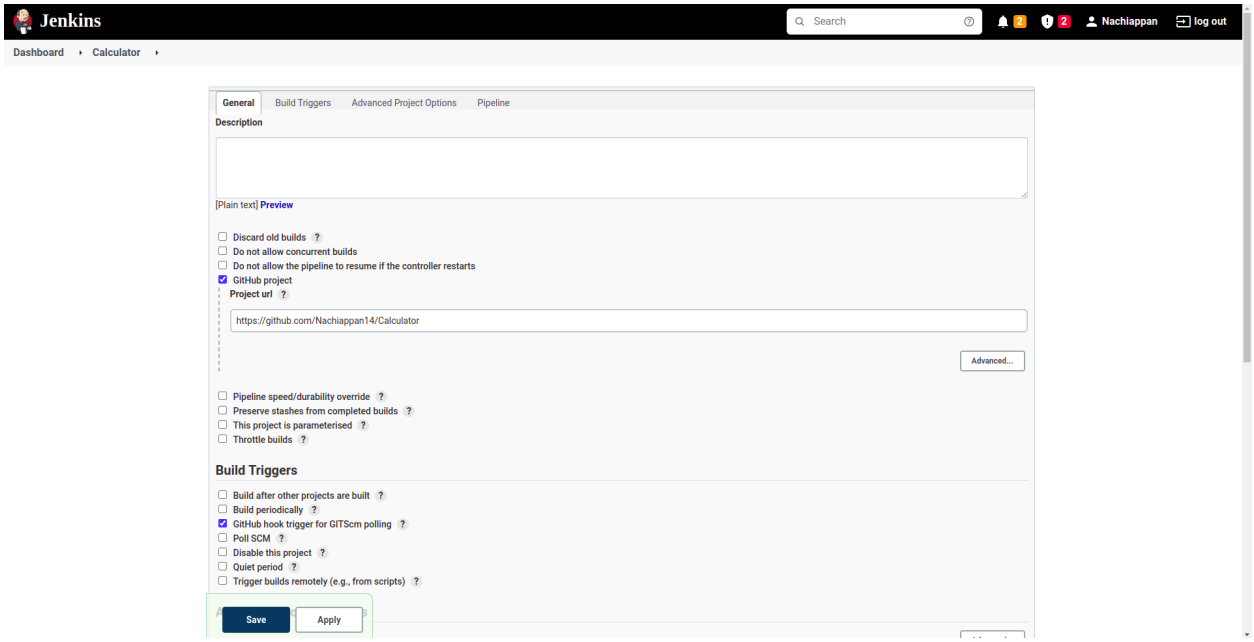


- we need to configure the docker credentials in jenkins global credentials, thus we can use dockerhub in the pipeline
- Create an access token in dockerhub and add the access token in the the jenkins credentials

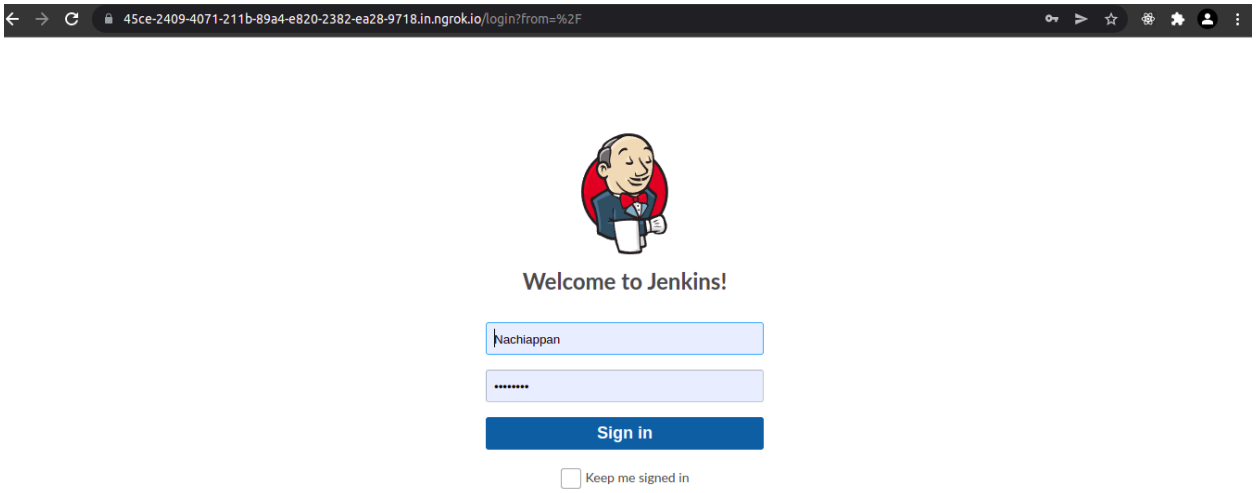




- Jenkins helps in the proper flow of the devops, when a code is changed the code is pulled from the github and continuously built and deployed in the cloud. The build can also be triggered manually. I tried poll SCM method, but since it was not efficient, a github webhook has been set up for the triggering.
- The Method to setup github hook has been given below



- Also go to github settings of the repository and add a webhook, use ngrok to host your website and give that link as the delivery link followed by a /web_hook
- To create a ngrok host use ngrok http <port>, make sure that you have signed in before creating an host
- The output is displayed in the installation section



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://45ce-2409-4071-211b-89a4-e820-2382-ea28-9718.in.ngrok.

Content type

application/json

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.

Send me **everything**.

Let me select individual events.

- Now enter the pipeline script that will be used for running the pipeline

Pipeline

Definition

Pipeline script

Script

```
pipeline
{
    environment
    {
        registry = "nachiappan14/calculator"
        registryCredential = "1"
        dockerImage = ''
    }
    agent any

    stages
    {
        stage("GitHub Repository")
        {
            steps
```

Use Groovy Sandbox

- Steps in the pipeline:
    - Cloning Git repository
    - Build Executable Jar
    - Building image (docker)
    - Push Image onto docker hub
    - Execute ansible script to pull docker image

Pipeline Calculator

Recent Changes

Stage View

	GitHub Repository	Maven	Build Docker image	Push docker image	Ansible Deploy
Average stage times: (Average full run time: ~1min 52s)	4s	10s	26s	30s	10s
#19 Apr 18 19:55 1 commit	2s	6s	42s	37s	14s
#18 Apr 18 19:53 1 commit	4s	11s failed	4s failed	1s failed	4s failed
#17 Apr 18 19:43 No Changes	10s	15s	34s	50s	22s
#16 Apr 18 19:33 1 commit	3s	25s	34s	46s	25s
#15 Apr 18 19:31 1 commit	3s	6s failed	1s failed	891ms failed	454ms failed
#14 Apr 18 1	4s	3s	28s	31s	3s

- After the pipeline stages are over you can do docker run to use the application.
- While using docker run we can mount a log file in the host machine to the log file of the app inside docker
- As and when code gets pushed the pipeline script gets executed and stages are built
- sudo docker run -it --mount type=bind, source=$(pwd)/calculator.log, target=/calculator.log nachiappan14/calculator:latest

```
NachiappanSK@g3-3579[22:35:42]:~/sem8/SPE/Calculator$ sudo docker run -it --mo
NachiappanSK@g3-3579[22:35:42]:~/sem8/SPE/Calculator$ sudo docker run -it --mo
unt type=bind,source=$(pwd)/calculator.log,target=/calculator.log nachiappan14
/calculator:latest

What would you like to do? Input a number and press enter.
1. Arithmetic
2. Exponents
3. Logarithms
4. Exit
```

Challenges Faced:

- Used target as localhost for building because the RAM was not sufficient to run a virtual machine in my computer and system froze many times and I had to restart my system.

Going through the full cycle:

- Make changes to the code and push it

```
NachiappanSK@g3-3579[23:44:13]:~/sem8/SPE/Calculator$ git push
Counting objects: 42, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (33/33), done.
Writing objects: 100% (42/42), 9.90 KiB | 4.95 MiB/s, done.
Total 42 (delta 17), reused 0 (delta 0)
remote: Resolving deltas: 100% (17/17), completed with 11 local objects.
To https://github.com/Nachiappan14/Calculator.git
   20e6ef7..e8526b2  master -> master
NachiappanSK@g3-3579[23:44:20]:~/sem8/SPE/Calculator$
```

- Jenkins pipeline automatically pulls the code, builds and deploys it

Recent Changes

Stage View

	GitHub Repository	Maven	Build Docker image	Push docker image	Ansible Deploy
Average stage times: (Average full run time: ~1min 52s)	5s	9s	26s	30s	10s
#20 Apr 18 23:45 No Changes					
	almost complete				
#19 Apr 18 19:55 1 commit	2s	6s	42s	37s	14s

Recent Changes

Stage View

	GitHub Repository	Maven	Build Docker image	Push docker image	Ansible Deploy
Average stage times: (Average full run time: ~1min 52s)	5s	14s	26s	30s	10s
#20 Apr 18 23:45 1 commit	12s	57s			
	almost complete				

- Since localhost is the target machine the docker gets built in the localhost

```
NachiappanSK@g3-3579[23:53:19]:~/sem8/SPE/Calculator/target$ docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
nachiappan14/calculator     latest    11951f1b08a8   14 minutes ago   660MB
nachiappan14/calculator     <none>    96c76d388473   2 hours ago      660MB
nachiappan14/calculator     <none>    00ff07d1c312   4 hours ago      660MB
nachiappan14/calculator     <none>    ac05af8c4a77   4 hours ago      660MB
nachiappan14/calculator     <none>    c45c86440670   4 hours ago      660MB
nachiappan14/calculator     <none>    26405d786363   6 hours ago      660MB
nachiappan14/calculator     <none>    bc9e0419e8a1   6 hours ago      660MB
nachiappan14/calculator     <none>    2cf52bcf9681   6 hours ago      660MB
nachiappan14/calculator     <none>    553c9af03c69   6 hours ago      660MB
nachiappan14/calculator     <none>    d67e87b85d53   7 hours ago      660MB
nachiappan14/calculator     <none>    1decd605b1ab   7 hours ago      660MB
nachiappan14/calculator     <none>    800345105f4d   9 hours ago      660MB
<none>                      <none>    ad0a8046b0e6   9 hours ago      660MB
openjdk                     11        8c5fc4518cc2   2 weeks ago      660MB
NachiappanSK@g3-3579[23:53:22]:~/sem8/SPE/Calculator/target$
```

- We can run the application with the following command
  - Docker run -it nachiappan14/calculator:latest

```
NachiappanSK@g3-3579[23:54:08]:~/sem8/SPE/Calculator/target$ docker run -it nachiappan14/calculator:latest

What would you like to do? Input a number and press enter.
1. Arithmetic
2. Exponents
3. Logarithms
4. Exit

1
Pick an operation by entering a number and pressing enter.
1. Add
2. Subtract
3. Multiply
4. Divide
5. Return to Arithmetic Menu
6. Return to Main Menu

4
format: a / b
Enter a
12
Enter b
7

1.7142857142857142

What would you like to do? Input a number and press enter.
1. Arithmetic
2. Exponents
3. Logarithms
4. Exit

4
Bye!
NachiappanSK@g3-3579[23:54:47]:~/sem8/SPE/Calculator/target$
```

Links:

Github: https://github.com/Nachiappan14/Calculator
Docker: https://hub.docker.com/repository/docker/nachiappan14/calculator