SPE Mini-Project Report

Simple Calculator Web-App

Name: Riddhi Chatterjee
Roll number: IMT2020094

Repositories:

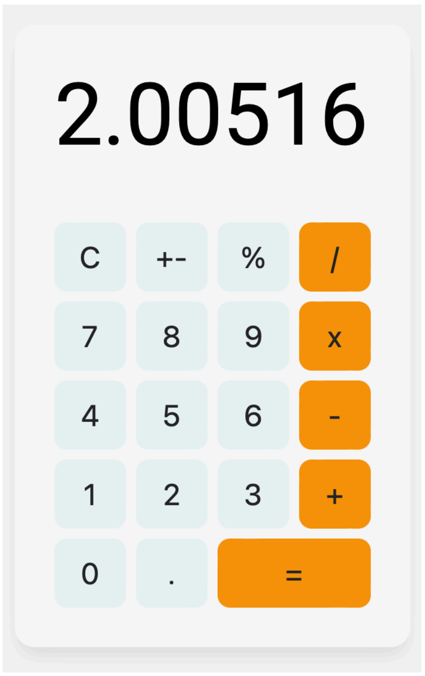
- **GitHub repository:**
  https://github.com/Riddhi-Chatterjee/calculator
- **Docker Hub repository:**
  https://hub.docker.com/repository/docker/riddhich/calculator/general

Project description:

A user-friendly, calculator web application is developed using react and the npm software registry. All the stages of the software development lifecycle have been automated using Jenkins as the CI/CD tool.
The calculator app has the following functionalities:

- Capable of handling simple calculations using the +,-,x,/ operators.
- Can perform floating point computations. Inputs can also be floating point numbers.
- Sign toggle button ("+-") can be used to toggle the sign of the number currently being displayed in the calculator.
- Percentage button ("%") can be used to divide the number being displayed by 100.
- Clear screen button ("C") can be used to clear the display of the calculator.

The user interface is as follows:

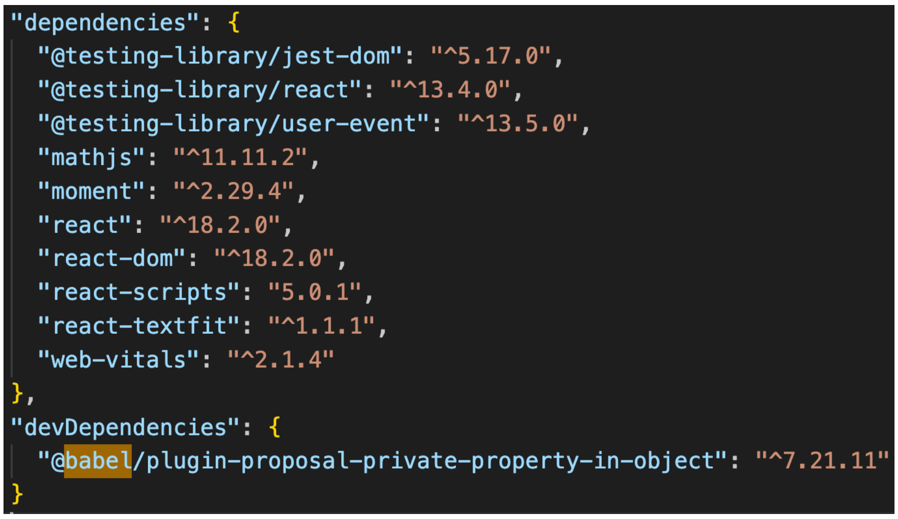


Tech Stack used:

- **Development:** react, npm
  A simple calculator web-app is developed using react and the npm software registry.
- **Testing:** Jest
  Jest is the testing framework used for testing react apps.
- **Source Code Management:** GitHub
  GitHub is used as the SCM tool, for version control.
- **Continuous Integration:** Jenkins
  Jenkins is used as the CI/CD tool for this project. It helps us to automate the development process by allowing us to write pipeline scripts.
- **Containerisation:** Docker, Docker Hub
  In this project, Docker was employed to containerise the application, simplifying its deployment and making it readily transferable to various environments. The Docker images that were generated were pushed to Docker Hub.
- **Deployment:** Ansible
  Ansible was used as an Infrastructure as Code (IaC) tool, for performing configuration management, pulling the docker image from Docker Hub, and running the container on localhost.

Calculator app dependencies:

The Calculator app has the following dependencies:



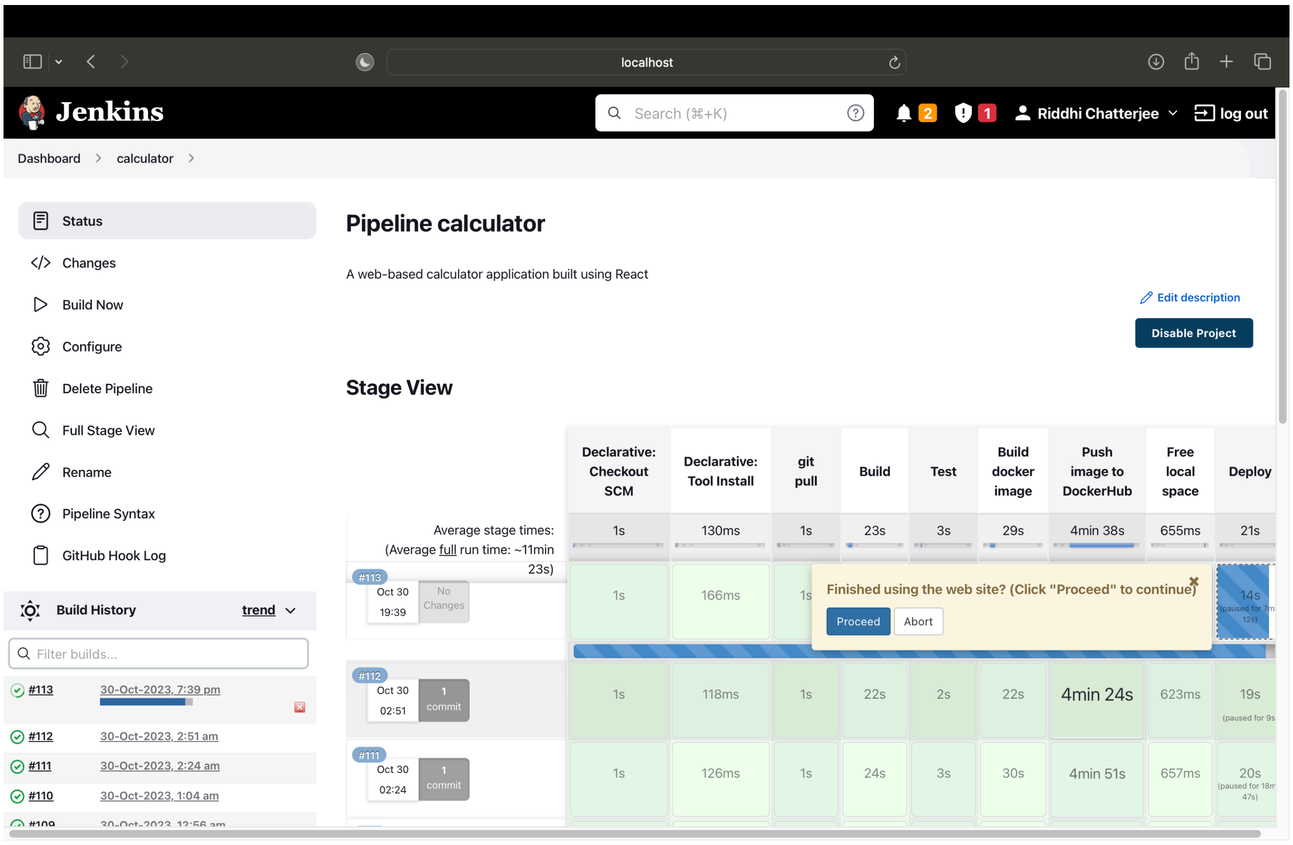
```
"dependencies": {
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "mathjs": "^11.11.2",
  "moment": "^2.29.4",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-scripts": "5.0.1",
  "react-textfit": "^1.1.1",
  "web-vitals": "^2.1.4"
},
"devDependencies": {
  "@babel/plugin-proposal-private-property-in-object": "^7.21.11"
}
```

“react-textfit” might cause some issues if it’s being installed by the “npm install” command. Thus, use the command “npm install --force” to install the above mentioned dependencies.

Jenkins pipeline overview:

- **Deployed state (container running) (Calculator useable at localhost:3000):**
  At the end of the Jenkins pipeline, in the Deploy stage, a container containing our application code is run in our local machine. The application code is executed there after installing all the required dependencies, and our Calculator web-app becomes usable at localhost:3000.

  The execution of the Jenkins pipeline pauses until we are done using the Calculator app. After we are done using the app, we can click the “Proceed” button, thereby allowing Jenkins to proceed with rest of the pipeline execution.


localhost

Jenkins | Search (⌘+K) | 2 | 1 | Riddhi Chatterjee | log out

Dashboard > calculator >

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax
- GitHub Hook Log

Build History — trend

Filter builds...

#113	30-Oct-2023, 7:39 pm
#112	30-Oct-2023, 2:51 am
#111	30-Oct-2023, 2:24 am
#110	30-Oct-2023, 1:04 am
#109	30-Oct-2023, 12:56 am

Pipeline calculator

A web-based calculator application built using React

Edit description

Disable Project

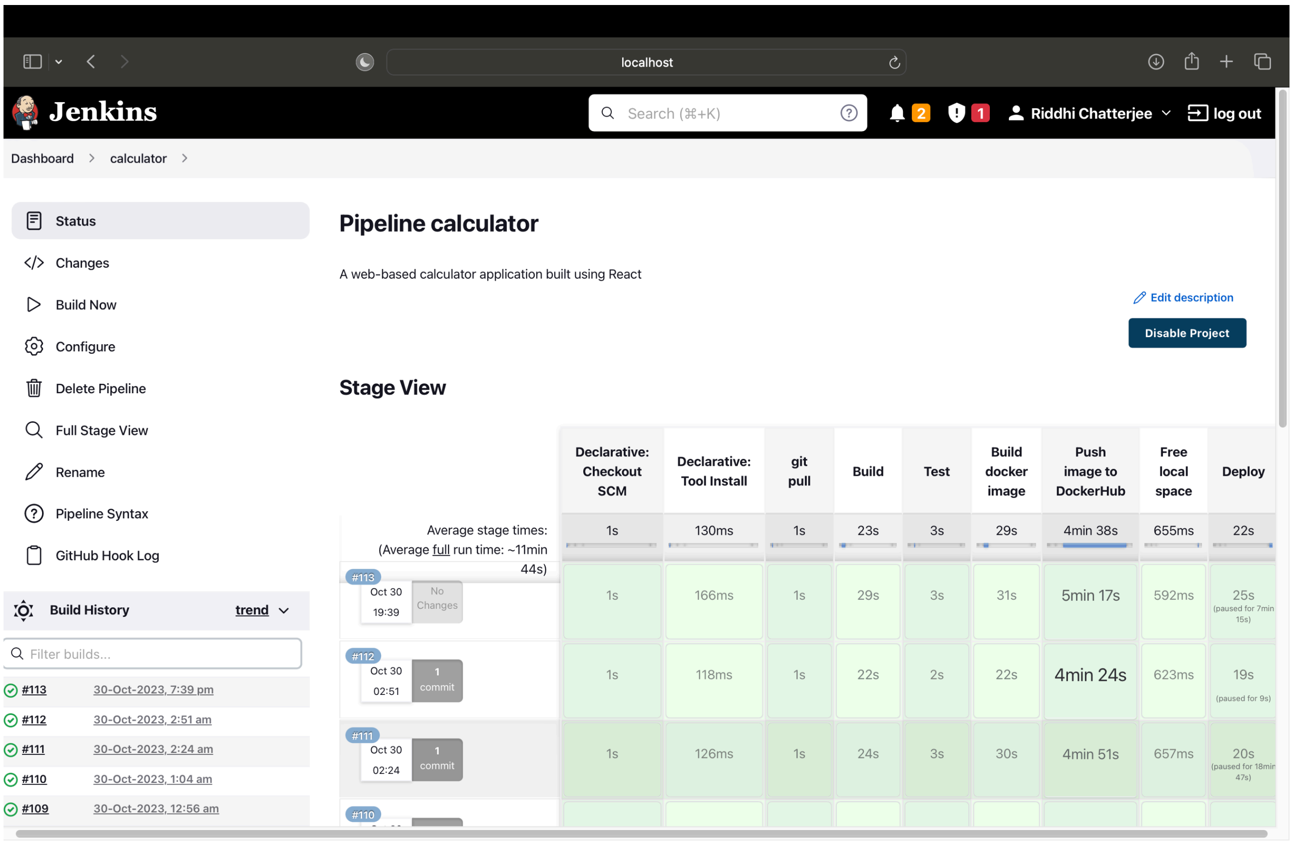
Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	git pull	Build	Test	Build docker image	Push image to DockerHub	Free local space	Deploy
Average stage times: (Average full run time: ~11min 23s)	1s	130ms	1s	23s	3s	29s	4min 38s	655ms	21s
#113 Oct 30 19:39 No Changes	1s	166ms	1s						14s (paused for 7min 12s)
#112 Oct 30 02:51 1 commit	1s	118ms	1s	22s	2s	22s	4min 24s	623ms	19s (paused for 9s)
#111 Oct 30 02:24 1 commit	1s	126ms	1s	24s	3s	30s	4min 51s	657ms	20s (paused for 18min 47s)

Finished using the web site? (Click "Proceed" to continue)

Proceed | Abort

- **Container stopped after using the Calculator:**
After we click the “Proceed” button, the rest of the Jenkins pipeline is executed and the container running our application code is stopped, thereby shutting down our Calculator application.


localhost

Jenkins | Search (⌘+K) | 2 | 1 | Riddhi Chatterjee | log out

Dashboard > calculator >

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax
- GitHub Hook Log

Build History — trend

Filter builds...

#113	30-Oct-2023, 7:39 pm
#112	30-Oct-2023, 2:51 am
#111	30-Oct-2023, 2:24 am
#110	30-Oct-2023, 1:04 am
#109	30-Oct-2023, 12:56 am

Pipeline calculator

A web-based calculator application built using React

Edit description

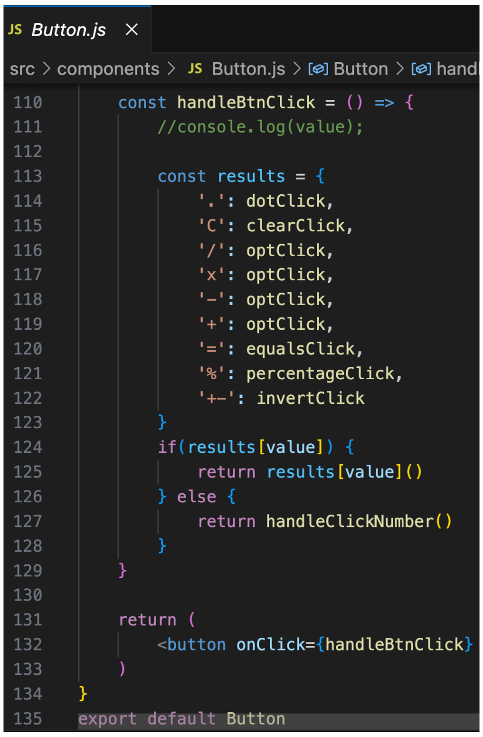
Disable Project

Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	git pull	Build	Test	Build docker image	Push image to DockerHub	Free local space	Deploy
Average stage times: (Average full run time: ~11min 44s)	1s	130ms	1s	23s	3s	29s	4min 38s	655ms	22s
#113 Oct 30 19:39 No Changes	1s	166ms	1s	29s	3s	31s	5min 17s	592ms	25s (paused for 7min 15s)
#112 Oct 30 02:51 1 commit	1s	118ms	1s	22s	2s	22s	4min 24s	623ms	19s (paused for 9s)
#111 Oct 30 02:24 1 commit	1s	126ms	1s	24s	3s	30s	4min 51s	657ms	20s (paused for 18min 47s)

Calculator app source code (handling of button clicks):

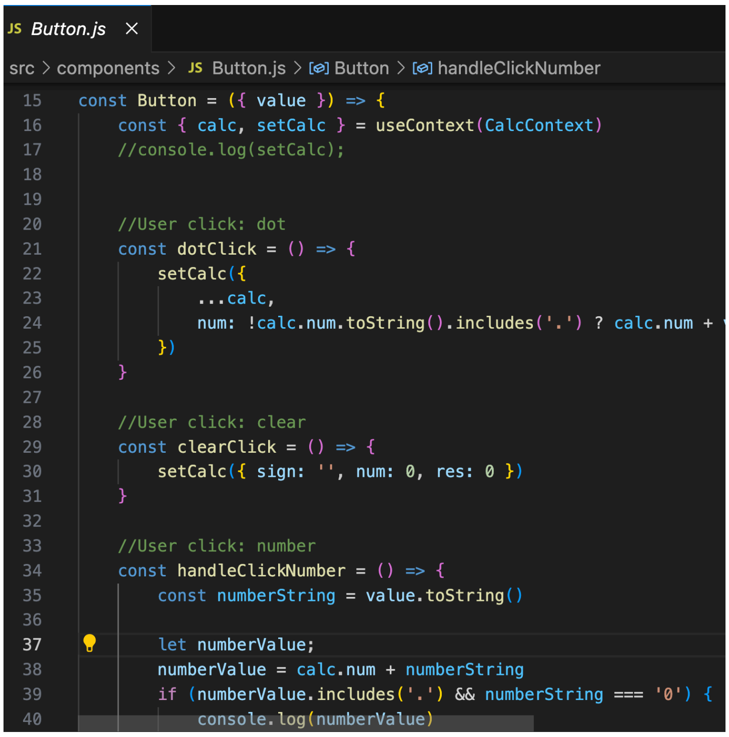
Several functions have been written to handle button clicks corresponding to the various buttons of the calculator:



Button.js

src > components > Button.js > Button > hand

```javascript
    const handleBtnClick = () => {
        //console.log(value);

        const results = {
            '.': dotClick,
            'C': clearClick,
            '/': optClick,
            'x': optClick,
            '-': optClick,
            '+': optClick,
            '=': equalsClick,
            '%': percentageClick,
            '+-': invertClick
        }
        if(results[value]) {
            return results[value]()
        } else {
            return handleClickNumber()
        }
    }

    return (
        <button onClick={handleBtnClick}
    )
}
export default Button
```



Button.js

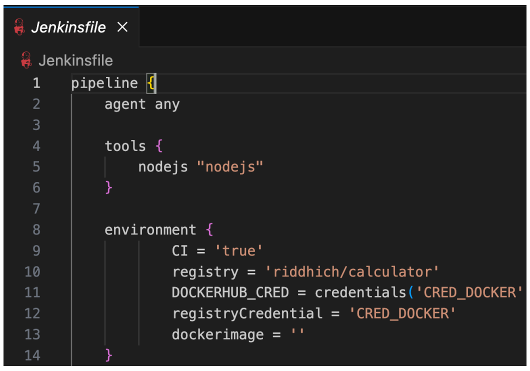
src > components > Button.js > Button > handleClickNumber

```javascript
const Button = ({ value }) => {
    const { calc, setCalc } = useContext(CalcContext)
    //console.log(setCalc);


    //User click: dot
    const dotClick = () => {
        setCalc({
            ...calc,
            num: !calc.num.toString().includes('.') ? calc.num +
        })
    }

    //User click: clear
    const clearClick = () => {
        setCalc({ sign: '', num: 0, res: 0 })
    }

    //User click: number
    const handleClickNumber = () => {
        const numberString = value.toString()

        let numberValue;
        numberValue = calc.num + numberString
        if (numberValue.includes('.') && numberString === '0') {
            console.log(numberValue)
```

Steps followed:

- **Setting up agent, tools and environment in the pipeline script:**

  At first, we specify the agent, tools and environment variables in our Jenkins pipeline script. We have to install the NodeJS Plugin in Jenkins and set up nodejs as a tool in Jenkins global tool configuration.

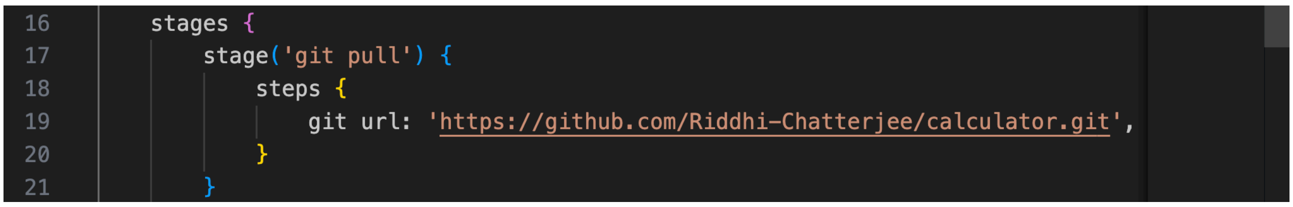


Jenkinsfile

```groovy
pipeline {
    agent any

    tools {
        nodejs "nodejs"
    }

    environment {
            CI = 'true'
            registry = 'riddhich/calculator'
            DOCKERHUB_CRED = credentials('CRED_DOCKER'
            registryCredential = 'CRED_DOCKER'
            dockerimage = ''
    }
```

- **Pipeline Stage 1 (Git Pull):**

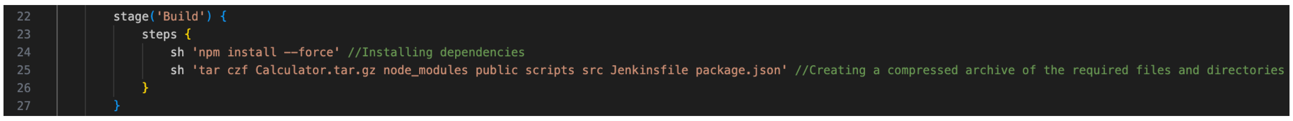
  The first stage of our pipeline script involves pulling the code from our GitHub repository (specified using the git url). The script corresponding to this stage is as follows:



```groovy
    stages {
        stage('git pull') {
            steps {
                git url: 'https://github.com/Riddhi-Chatterjee/calculator.git',
            }
        }
```

- **Pipeline Stage 2 (Build):**

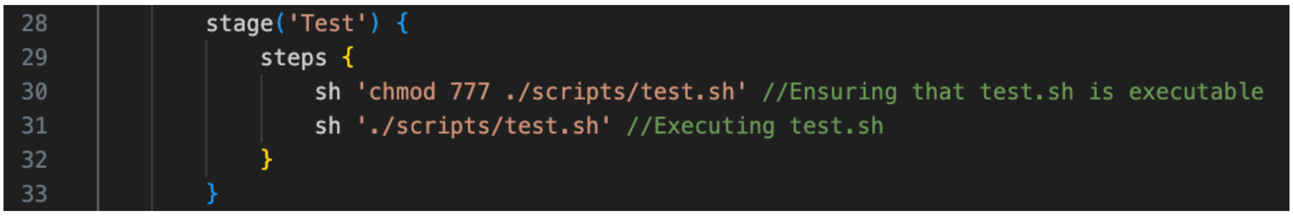
  The 'npm install --force' command executed with the 'sh' command installs all the dependencies listed in the package.json file.
  The 'tar czf' command is employed to compress and package the essential files and directories for the React application, including the Jenkinsfile and other required files, into a single archive for deployment.



```
        stage('Build') {
            steps {
                sh 'npm install --force' //Installing dependencies
                sh 'tar czf Calculator.tar.gz node_modules public scripts src Jenkinsfile package.json' //Creating a compressed archive of the required files and directories
            }
        }
```

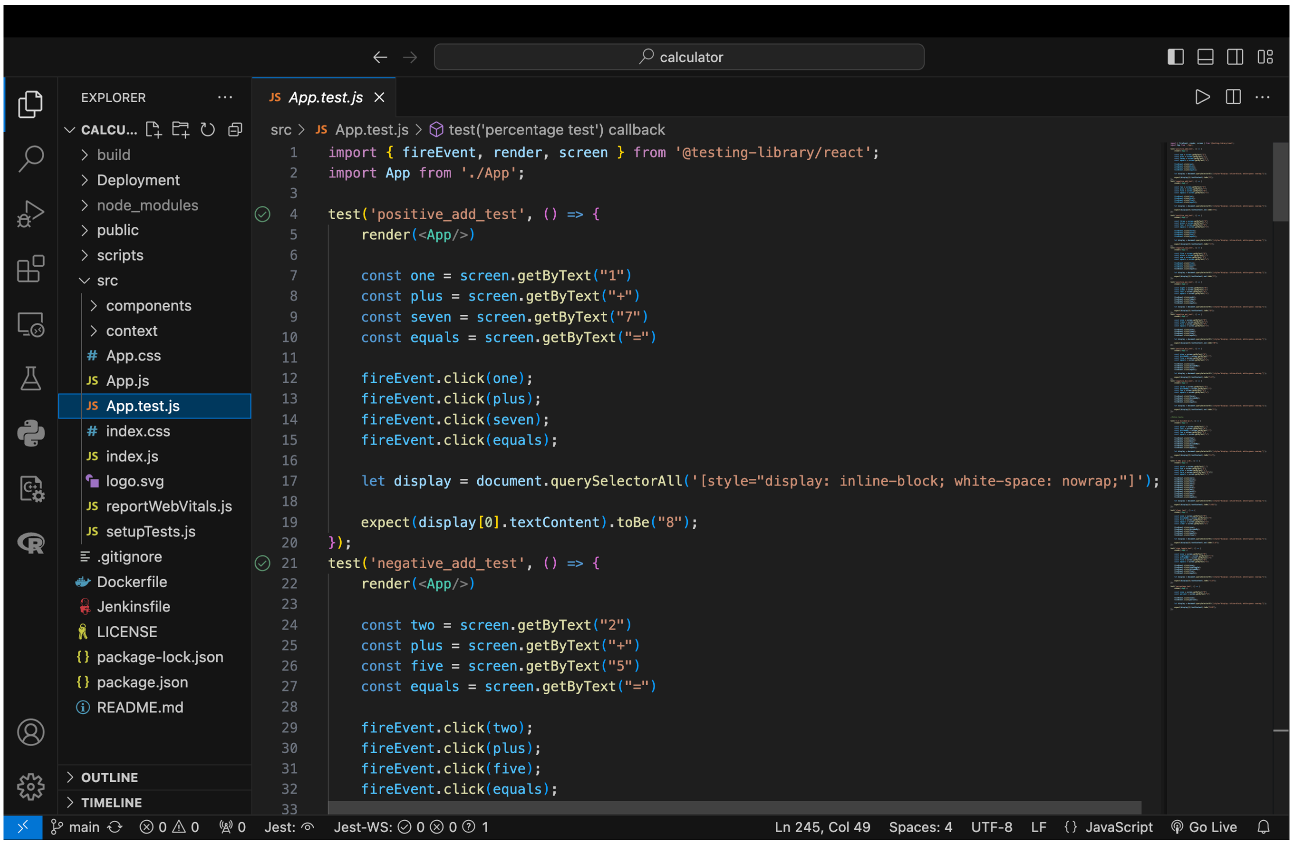
- **Pipeline Stage 3 (Test):**

  The third stage of our pipeline is to run the unit tests. The 'test.sh' file is responsible for running 'npm test'. The pipeline script snippet is as follows:



```
        stage('Test') {
            steps {
                sh 'chmod 777 ./scripts/test.sh' //Ensuring that test.sh is executable
                sh './scripts/test.sh' //Executing test.sh
            }
        }
```

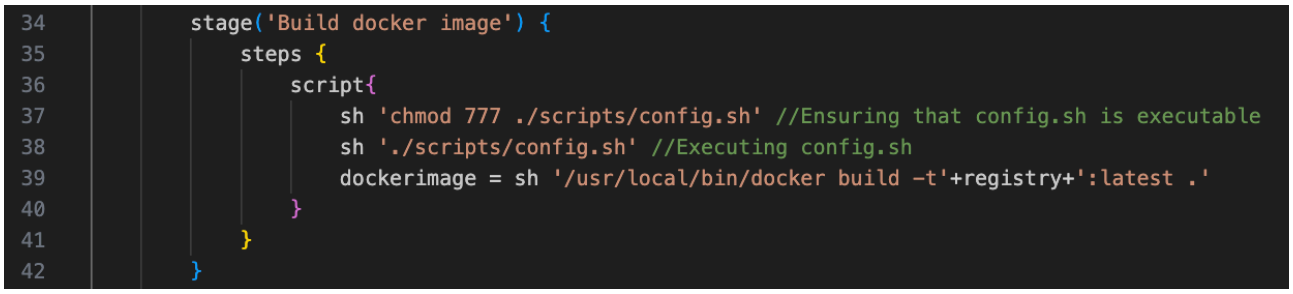
A total of 13 unit tests have been performed on our Calculator application. This includes positive and negative tests for addition, subtraction, multiplication and division operations, and 5 other miscellaneous tests. The test cases are written in the 'App.test.js' file as shown below:



```
import { fireEvent, render, screen } from '@testing-library/react';
import App from './App';

test('positive_add_test', () => {
    render(<App/>)

    const one = screen.getByText("1")
    const plus = screen.getByText("+")
    const seven = screen.getByText("7")
    const equals = screen.getByText("=")

    fireEvent.click(one);
    fireEvent.click(plus);
    fireEvent.click(seven);
    fireEvent.click(equals);

    let display = document.querySelectorAll('[style="display: inline-block; white-space: nowrap;"]');

    expect(display[0].textContent).toBe("8");
});
test('negative_add_test', () => {
    render(<App/>)

    const two = screen.getByText("2")
    const plus = screen.getByText("+")
    const five = screen.getByText("5")
    const equals = screen.getByText("=")

    fireEvent.click(two);
    fireEvent.click(plus);
    fireEvent.click(five);
    fireEvent.click(equals);

```

- **Pipeline Stage 4 (Build docker image):**

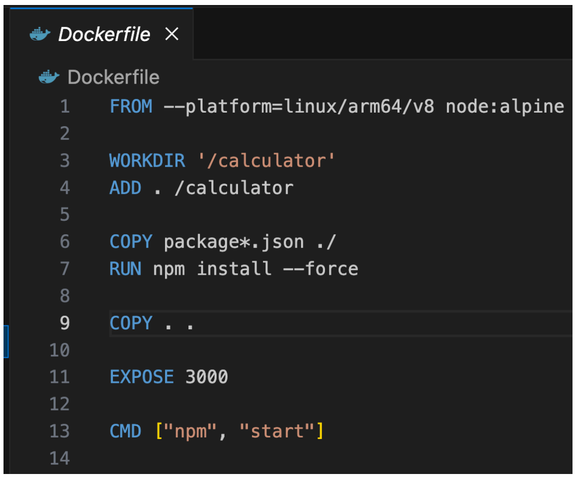
The pipeline script snippet corresponding to the creation of the docker image is as follows:



```
        stage('Build docker image') {
            steps {
                script{
                    sh 'chmod 777 ./scripts/config.sh' //Ensuring that config.sh is executable
                    sh './scripts/config.sh' //Executing config.sh
                    dockerimage = sh '/usr/local/bin/docker build -t'+registry+':latest .'
                }
            }
        }
```

At first some modifications have to be made to the '~/.docker/config.json' file because of a small typo in the original file. This is done using the 'config.sh' shell script.
We also have to ensure that we have installed the 'Docker Pipeline' and 'Docker plugin' plugins available in Jenkins.
We also have to set up docker credentials in Jenkins, which we will be using for authenticating ourselves to DockerHub.

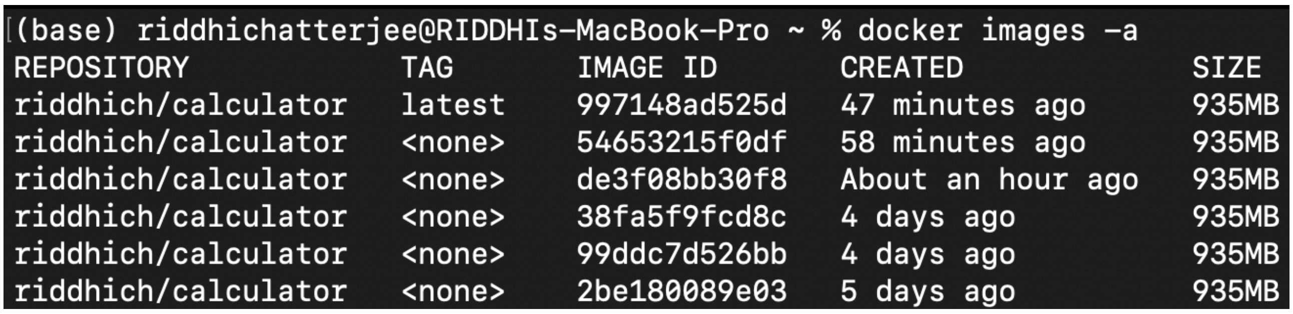
We require a Dockerfile to help in generating an image that includes our codebase and subsequently uploading it to DockerHub:



```
FROM --platform=linux/arm64/v8 node:alpine

WORKDIR '/calculator'
ADD . /calculator

COPY package*.json ./
RUN npm install --force

COPY . .

EXPOSE 3000

CMD ["npm", "start"]

```

- To begin, we specify the base image from which we want to build. We select "node:<version>:alpine," which is based on the popular Alpine Linux project.
- Next, we establish a directory within the image to store the application code. This directory will serve as the working directory for our application. Our working directory is set to "/calculator."
- Since this image already includes Node.js and npm, the next step is to install our application's dependencies using npm. This involves running the command "npm install --force". A wildcard is used to ensure both "package.json" and "package-lock.json" are copied.
- To include our application's source code in the docker image, we use the COPY instruction.
- Our application binds to port 3000, so we utilize the EXPOSE instruction to make sure it's accessible via the docker daemon.
- Finally, we define the command to run our application using CMD, specifying our runtime. In this case, we use "npm start" to start our application.

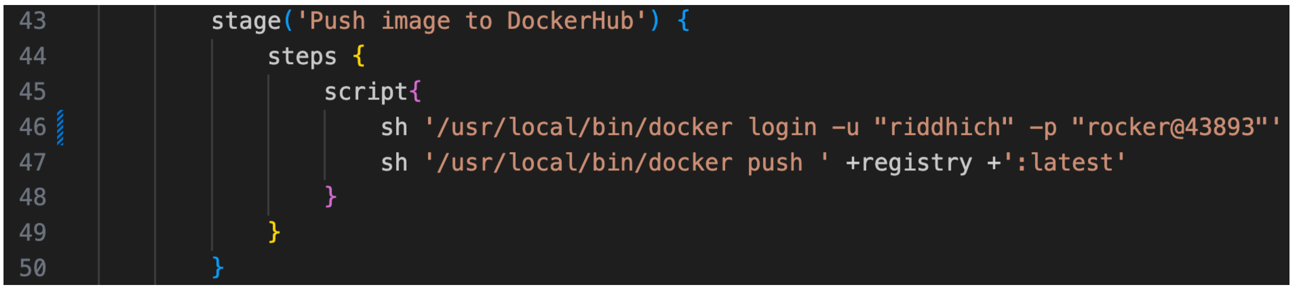
Finally we build our docker image using the docker build command. The newly created docker image can be viewed using the 'docker images -a' command as shown below:



```
(base) riddhichatterjee@RIDDHIs-MacBook-Pro ~ % docker images -a
REPOSITORY            TAG       IMAGE ID       CREATED            SIZE
riddhich/calculator   latest    997148ad525d   47 minutes ago     935MB
riddhich/calculator   <none>    54653215f0df   58 minutes ago     935MB
riddhich/calculator   <none>    de3f08bb30f8   About an hour ago  935MB
riddhich/calculator   <none>    38fa5f9fcd8c   4 days ago         935MB
riddhich/calculator   <none>    99ddc7d526bb   4 days ago         935MB
riddhich/calculator   <none>    2be180089e03   5 days ago         935MB
```

- **Pipeline Stage 5 (Push image to DockerHub):**

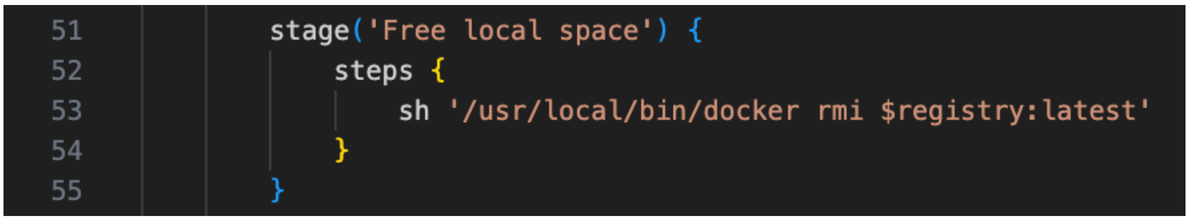
  The fifth stage of our pipeline involves pushing the newly created docker image to DockerHub. The pipeline script snippet is as follows:



```
        stage('Push image to DockerHub') {
            steps {
                script{
                    sh '/usr/local/bin/docker login -u "riddhich" -p "rocker@43893"'
                    sh '/usr/local/bin/docker push ' +registry +':latest'
                }
            }
        }
```

- **Pipeline Stage 6 (Free local space):**

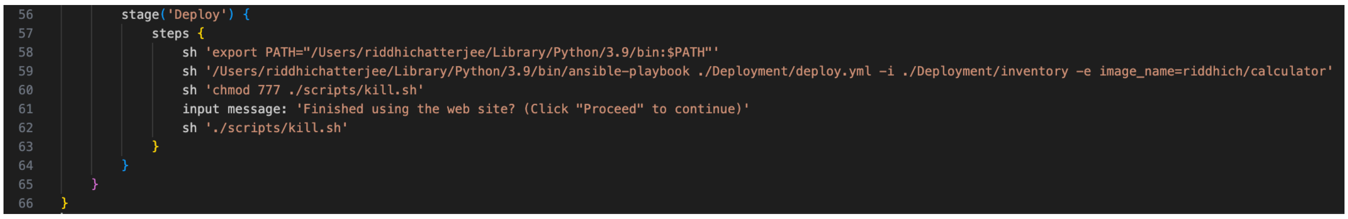
  The sixth stage of our pipeline is to delete the docker image on our local machine to save disk space. The pipeline script snippet is as follows:



```
        stage('Free local space') {
            steps {
                sh '/usr/local/bin/docker rmi $registry:latest'
            }
        }
```

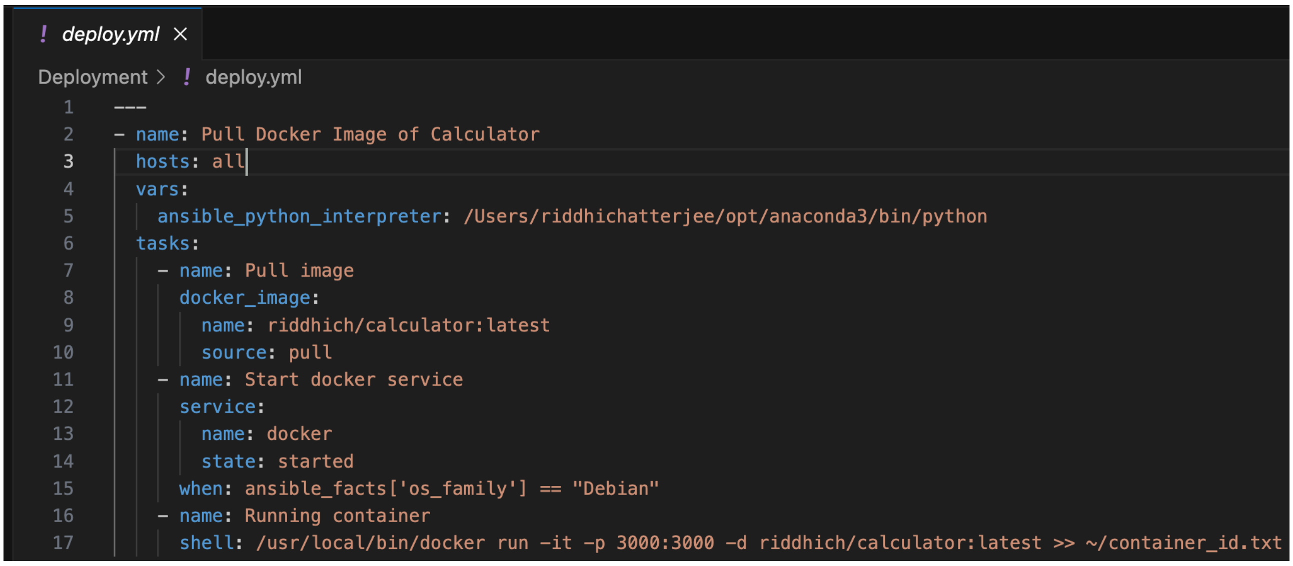
- **Pipeline Stage 7 (Deploy):**

  The seventh stage of our pipeline is the deployment stage, where we run a container on our local machine, which in turn gets our calculator application up and running. When we are done using the application, we stop the container, thereby shutting down our calculator application. The pipeline script snippet is as follows:



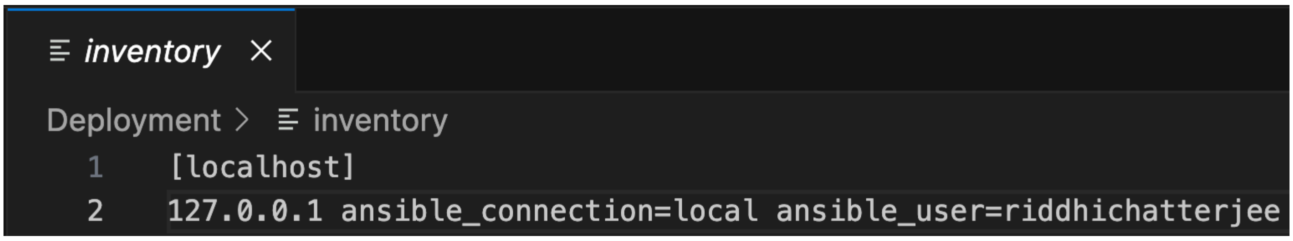
```
        stage('Deploy') {
            steps {
                sh 'export PATH="/Users/riddhichatterjee/Library/Python/3.9/bin:$PATH"'
                sh '/Users/riddhichatterjee/Library/Python/3.9/bin/ansible-playbook ./Deployment/deploy.yml -i ./Deployment/inventory -e image_name=riddhich/calculator'
                sh 'chmod 777 ./scripts/kill.sh'
                input message: 'Finished using the web site? (Click "Proceed" to continue)'
                sh './scripts/kill.sh'
            }
        }
    }
}
```

The Ansible playbook ‘deploy.yml’ is as follows:



```
---
- name: Pull Docker Image of Calculator
  hosts: all
  vars:
    ansible_python_interpreter: /Users/riddhichatterjee/opt/anaconda3/bin/python
  tasks:
    - name: Pull image
      docker_image:
        name: riddhich/calculator:latest
        source: pull
    - name: Start docker service
      service:
        name: docker
        state: started
      when: ansible_facts['os_family'] == "Debian"
    - name: Running container
      shell: /usr/local/bin/docker run -it -p 3000:3000 -d riddhich/calculator:latest >> ~/container_id.txt
```

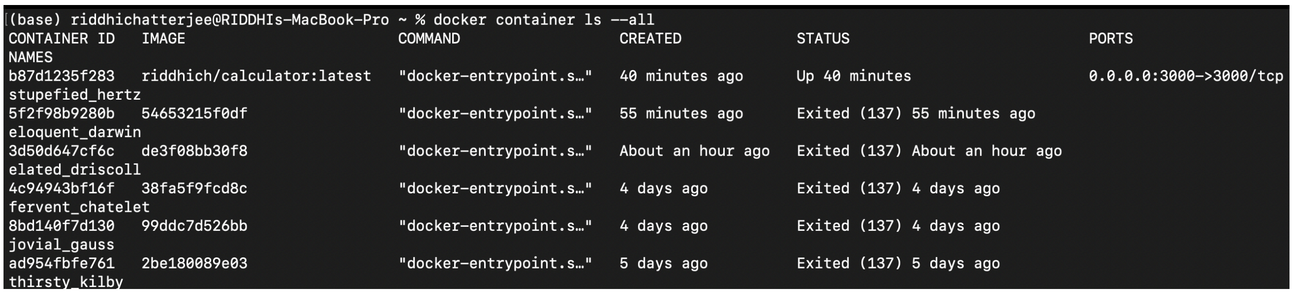
The inventory file is as follows:



```
[localhost]
127.0.0.1 ansible_connection=local ansible_user=riddhichatterjee
```

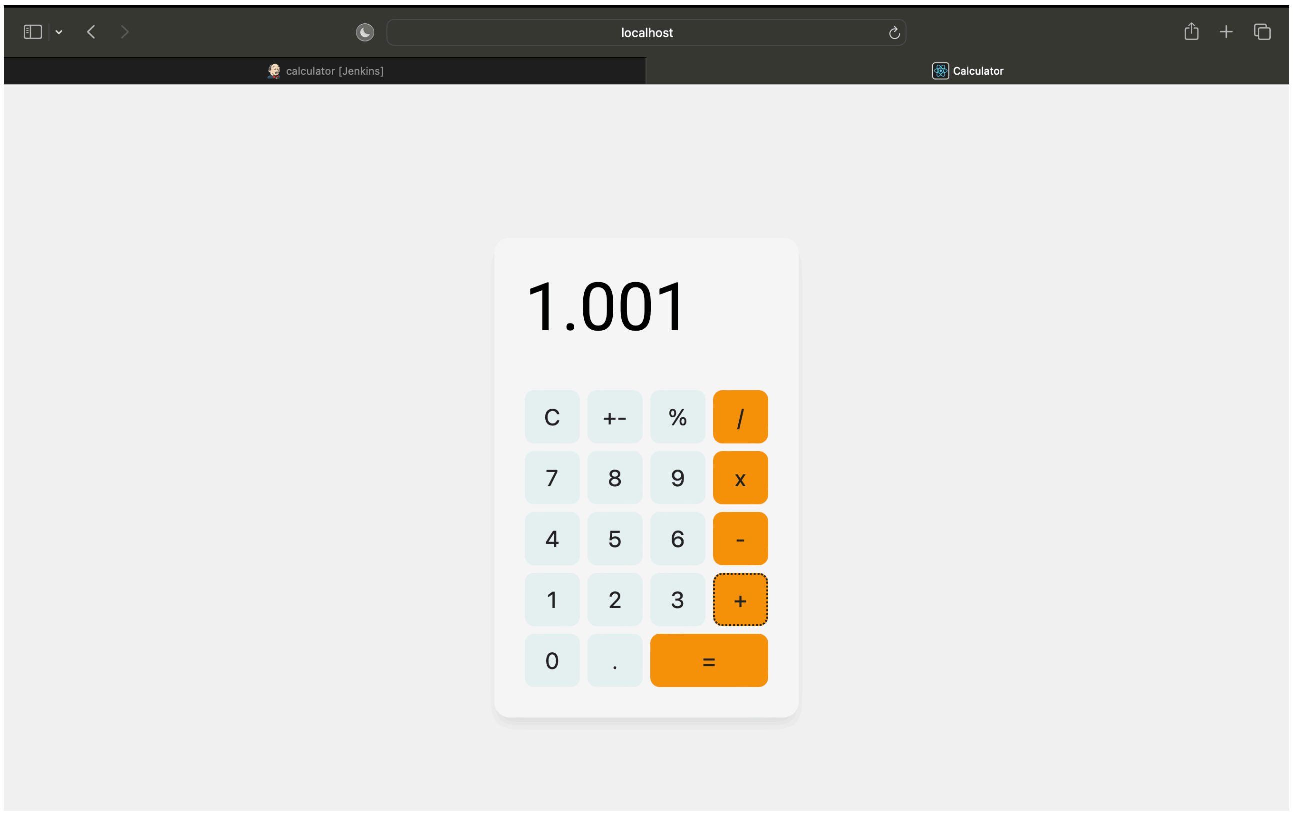
We need to make sure that we have installed the Ansible plugin in Jenkins.
After executing the playbook, a container would start running, which in turn would run our application. We store the ID of the container we ran just now for future use.

We can view the container we created and ran just now using the command ‘docker container ls --all’ as shown below:

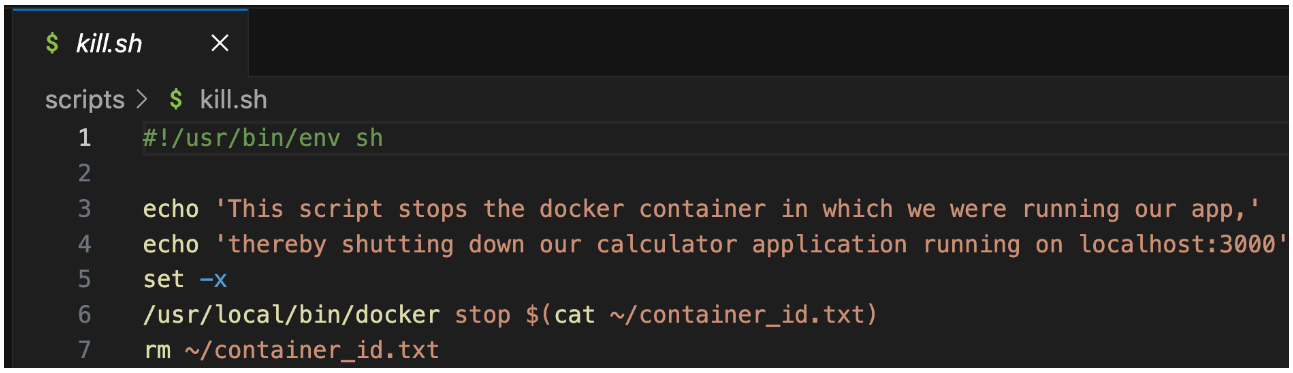


```
(base) riddhichatterjee@RIDDHIs-MacBook-Pro ~ % docker container ls --all
CONTAINER ID   IMAGE                        COMMAND                  CREATED             STATUS                           PORTS
NAMES
b87d1235f283   riddhich/calculator:latest   "docker-entrypoint.s…"   40 minutes ago      Up 40 minutes                    0.0.0.0:3000->3000/tcp
stupefied_hertz
5f2f98b9280b   54653215f0df                 "docker-entrypoint.s…"   55 minutes ago      Exited (137) 55 minutes ago
eloquent_darwin
3d50d647cf6c   de3f08bb30f8                 "docker-entrypoint.s…"   About an hour ago   Exited (137) About an hour ago
elated_driscoll
4c94943bf16f   38fa5f9fcd8c                 "docker-entrypoint.s…"   4 days ago          Exited (137) 4 days ago
fervent_chatelet
8bd140f7d130   99ddc7d526bb                 "docker-entrypoint.s…"   4 days ago          Exited (137) 4 days ago
jovial_gauss
ad954fbfe761   2be180089e03                 "docker-entrypoint.s…"   5 days ago          Exited (137) 5 days ago
thirsty_kilby
```

Our calculator application is now usable at: http://localhost:3000



After using the calculator application, we stop the running container using the ‘kill.sh’ shell script, with the help of the ID of the running container that we had stored earlier:



```
#!/usr/bin/env sh

echo 'This script stops the docker container in which we were running our app,'
echo 'thereby shutting down our calculator application running on localhost:3000'
set -x
/usr/local/bin/docker stop $(cat ~/container_id.txt)
rm ~/container_id.txt
```

This shuts down our calculator application gracefully.