SPE REPORT (calculatorsOps)

What and Why of DevOps:

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the software development life cycle and deliver high-quality software continuously. It aims to automate and integrate the processes of software development, testing, deployment, and infrastructure management to achieve faster delivery, improved collaboration, and increased reliability.

Why DevOps for this project?

Integrating DevOps practices into the development of the Scientific Calculator application offers several benefits:

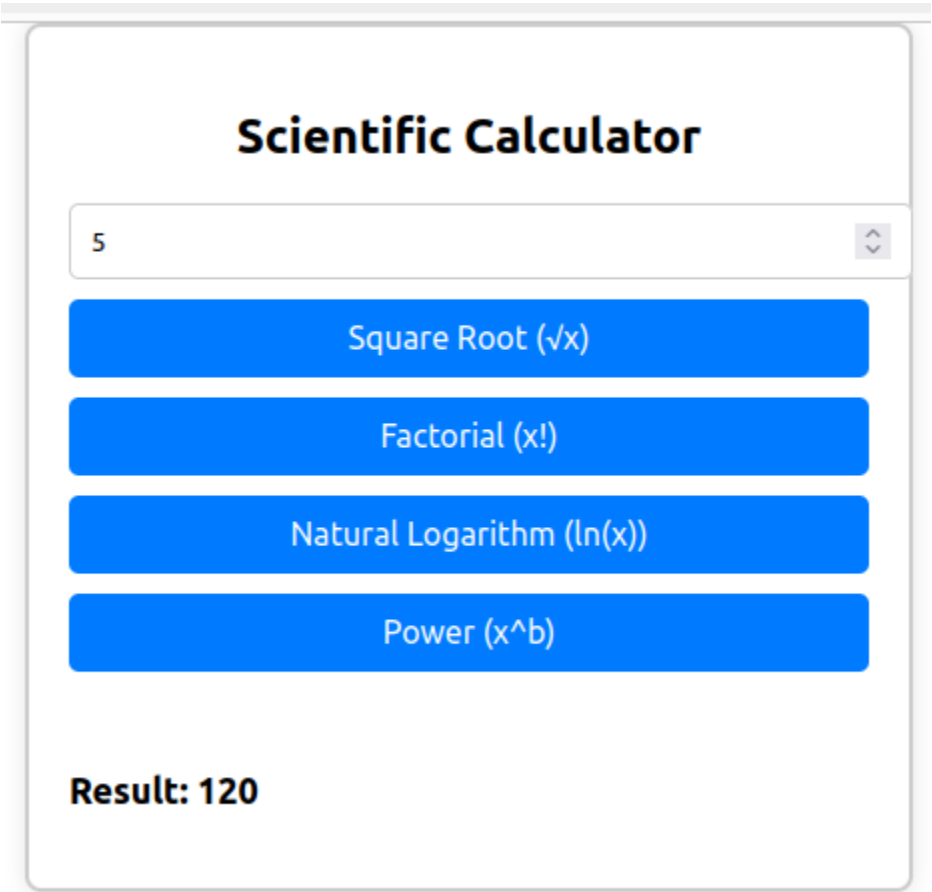
- Faster Delivery: DevOps automation streamlines the software delivery process, allowing for quicker iterations and updates.
- Improved Collaboration: DevOps encourages closer collaboration between development and operations teams, leading to better communication and alignment of goals.
- Continuous Integration and Deployment: Automated testing and deployment pipelines ensure that changes are thoroughly tested and deployed consistently, reducing the risk of errors and downtime.
- Scalability and Reliability: DevOps practices such as containerization and infrastructure as code (IaC) enhance scalability and reliability by enabling the application to be deployed consistently across different environments.

Tools Used:

Source Control Management: GitHub
Testing: JUnit
Build: npm
Continuous Integration: Jenkins
Containerization: Docker
Deployment and Configuration Management: Ansible

Step Explanations:

1. Create a React Project for a Calculator



2. Write Unit Tests Using Jest

- Developed comprehensive unit tests to ensure the functionality of the calculator.
- Tested arithmetic operations, input validation, and edge cases.

```
PASS src/App.test.js
  ✓ renders scientific calculator header (23 ms)
  ✓ calculates square root correctly (41 ms)
  ✓ calculates factorial correctly (19 ms)
  ✓ calculates natural logarithm correctly (11 ms)
  ✓ calculates power correctly (14 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        1.014 s
Ran all test suites.
```

3. Create Dockerfile

Create a Dockerfile in the root of your project with the following content:

Dockerfile

```
1   FROM node:alpine
2
3   WORKDIR '/app'
4
5   COPY package.json .
6   RUN npm install
7
8   COPY . .
9
10  CMD ["npm", "run", "start"]
```

4. Test the Dockerfile by Building the Image & Pushing It to the Docker Hub

- Built a Docker image using the Dockerfile.
- Tested the Docker image locally to ensure it functioned correctly.
- Pushed the Docker image to Docker Hub for accessibility.

```
vicky@Vicky:~/spe_mini_project$ docker build -t calculator .
[+] Building 68.8s (10/10) FINISHED                                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 145B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/node:alpine                                                                0.0s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [1/5] FROM docker.io/library/node:alpine                                                                                  0.0s
 => [internal] load build context                                                                                             5.8s
 => => transferring context: 299.50MB                                                                                         5.7s
 => CACHED [2/5] WORKDIR /app                                                                                                 0.0s
 => [3/5] COPY package.json .                                                                                                 0.4s
 => [4/5] RUN npm install                                                                                                    47.4s
 => [5/5] COPY . .                                                                                                            8.2s
 => exporting to image                                                                                                        6.8s
 => => exporting layers                                                                                                       6.8s
 => => writing image sha256:40acaeadb14eeb351ca3297eadfe471c0588beda9adfa46e8a5236a899d37d32                                  0.0s
 => => naming to docker.io/library/calculator                                                                                 0.0s
vicky@Vicky:~/spe_mini_project$
```

```
vicky@Vicky:~/spe_mini_project$ docker run calculator

> spe_mini_project@0.1.0 start
> react-scripts start

(node:25) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewa
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:25) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddle
Starting the development server...

One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintianed anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.

Compiled successfully!

You can now view spe_mini_project in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.17.0.2:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
Compiling...
Compiled successfully!
webpack compiled successfully
```

5. Created a Jenkins Pipeline with Git SCM

Step 1: Install Required Jenkins Plugins

1. Navigate to Jenkins Dashboard > Manage Jenkins > Manage Plugins.
2. Go to the "Available" tab and search for the following plugins:
    - Docker Pipeline
    - Ansible
    - GitHub Integration

- NodeJS

3. Select these plugins and click on "Install without restart."

Step 2: Add Docker Credentials

1. Navigate to Jenkins Dashboard > Credentials > System > Global credentials.
2. Click on "Add Credentials."
3. Select "Username with password" as the kind of credentials.
4. Enter your Docker Hub username and password.
5. Set an ID for your credentials.

Step 3: Configure Docker in Jenkins

1. Navigate to Jenkins Dashboard > Manage Jenkins > Global Tool Configuration.
2. Scroll down to the "Docker" section and click on "Add Docker."
3. Enter a name for the Docker installation.
4. Check the box for "Install automatically" and select a Docker version to install.

Step 4: Configure NodeJS in Jenkins

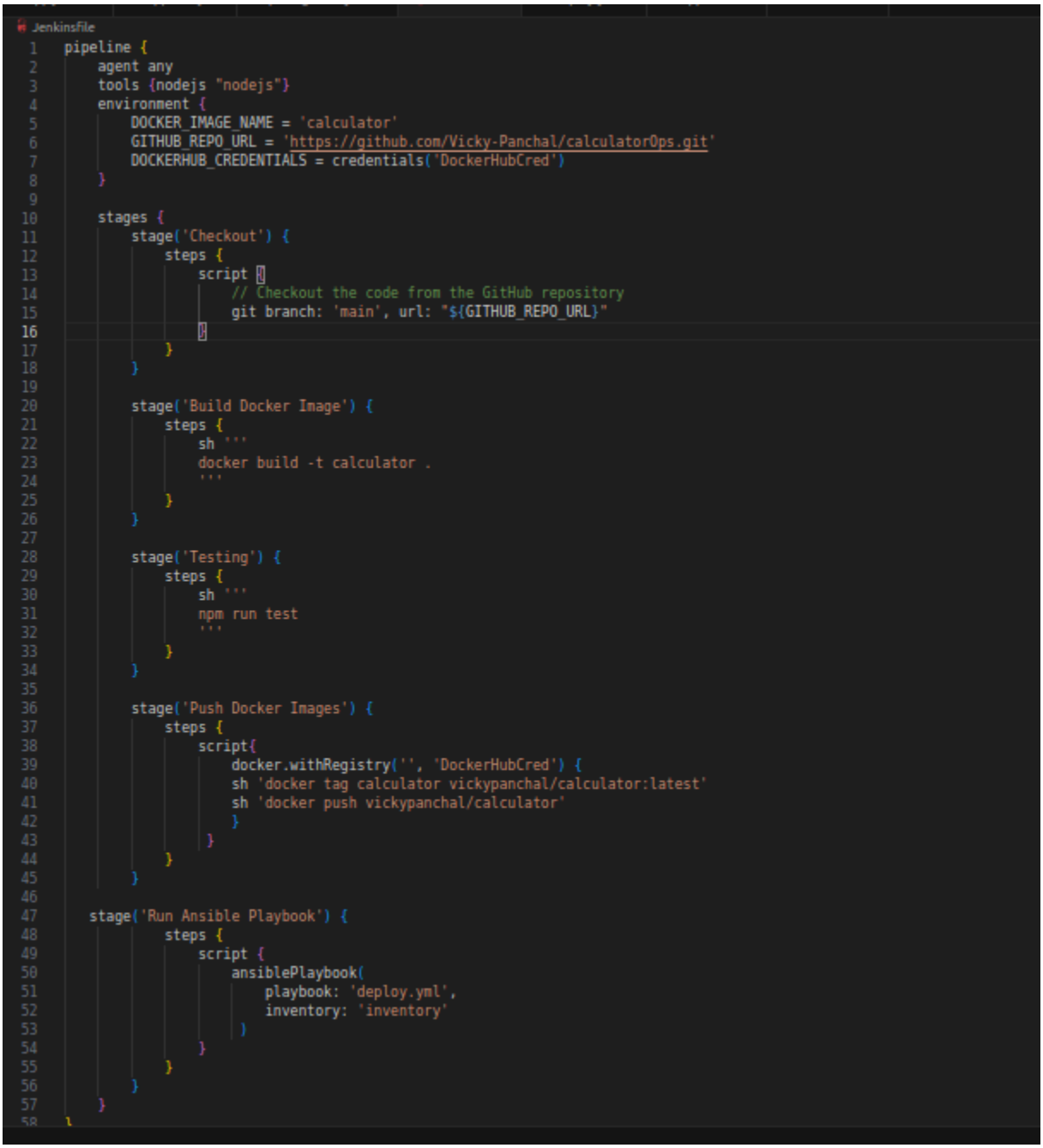
1. Navigate to Jenkins Dashboard > Manage Jenkins > Global Tool Configuration.
2. Scroll down to the "NodeJS" section and click on "Add NodeJS."
3. Enter a name for the NodeJS installation.
4. Check the box for "Install automatically" and select a NodeJS version to 18.18.2.

Step 5: Update Jenkins Pipeline with Docker and NodeJS Configurations

1. Update your Jenkinsfile to include Docker and NodeJS configurations in the pipeline stages where they are required.

6. Wrote Jenkins Pipeline in Jenkinsfile

- Defined stages in the Jenkins pipeline for building, testing, and deploying the React application.
- Ensured proper error handling and notifications throughout the pipeline.



```
Jenkinsfile
pipeline {
    agent any
    tools {nodejs "nodejs"}
    environment {
        DOCKER_IMAGE_NAME = 'calculator'
        GITHUB_REPO_URL = 'https://github.com/Vicky-Panchal/calculatorOps.git'
        DOCKERHUB_CREDENTIALS = credentials('DockerHubCred')
    }

    stages {
        stage('Checkout') {
            steps {
                script {
                    // Checkout the code from the GitHub repository
                    git branch: 'main', url: "${GITHUB_REPO_URL}"
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                sh '''
                docker build -t calculator .
                '''
            }
        }

        stage('Testing') {
            steps {
                sh '''
                npm run test
                '''
            }
        }

        stage('Push Docker Images') {
            steps {
                script{
                    docker.withRegistry('', 'DockerHubCred') {
                    sh 'docker tag calculator vickypanchal/calculator:latest'
                    sh 'docker push vickypanchal/calculator'
                    }
                }
            }
        }

    stage('Run Ansible Playbook') {
            steps {
                script {
                    ansiblePlaybook(
                        playbook: 'deploy.yml',
                        inventory: 'inventory'
                    )
                }
            }
        }
    }
}
```

7. Wrote Ansible Script in deploy.yml

- Created an Ansible playbook (deploy.yml) to deploy the React application to a target server.
- Included tasks for copying files, installing dependencies, and restarting the application.

deploy.yml

```yaml
---
- name: Pull Docker Image from Docker Hub
  hosts: localhost
  remote_user: vickypanchal
  become: false
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Pull Docker Image
      docker_image:
        name: "vickypanchal/calculator"
        source: pull
      register: docker_pull_result

    - name: Display Docker Pull Result
      debug:
        var: docker_pull_result

    - name: Start Docker service
      service:
        name: docker
        state: started

    - name: Remove existing container if running
      shell: docker rm -f calculator
      ignore_errors: true

    - name: Running container
      shell: docker run -it -d --name calculator vickypanchal/calculator
```

8. Pushed Project to GitHub

- Committed and pushed the React project code to a GitHub repository.
- Used Git branching and version control best practices.

9. Tested the Jenkins Pipeline with "Build Now"

- Triggered the Jenkins pipeline manually to test the entire CI/CD process.
- Verified that the pipeline executed successfully and deployed the application as expected.