CS 816 - Software Production Engineering

Mini Project Report - Scientific Calculator

Tanay Nagarkar IMT2022083

1. Introduction

This project implements a scientific calculator using Python, integrated with a DevOps toolchain for continuous integration, testing, containerization, and deployment. The calculator supports the following operations:

- **Square root (√x)**
- **Factorial (!x)**
- **Natural logarithm (ln(x))**
- **Power function (x^b)**

The aim of this project is to demonstrate software production engineering concepts, particularly the automation of build, test, and deployment processes using modern DevOps tools.

What and Why of DevOps

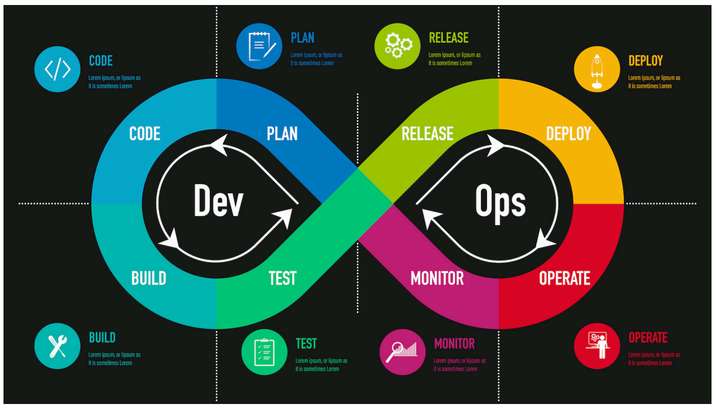
DevOps is a strategic software engineering paradigm that unifies software development (Dev) and IT operations (Ops) to enable rapid, reliable, and continuous software delivery. By breaking down traditional silos between development and operations teams, DevOps fosters a culture of collaboration, accountability, and efficiency.

Key advantages of adopting DevOps include:

1. **Continuous Integration and Continuous Delivery (CI/CD):**
    - Facilitates automated building, testing, and deployment of code changes, ensuring that software is always in a releasable state.
    - Reduces integration conflicts and accelerates time-to-market for new features.
2. **Automation of Repetitive Processes:**
    - Streamlines critical software lifecycle tasks such as dependency installation, unit testing, containerization, and deployment.
    - Minimizes human error and frees engineering teams to focus on higher-value development activities.
3. **Enhanced Collaboration and Transparency:**
    - Encourages cross-functional communication and shared ownership of software quality.
    - Provides clear visibility into code changes, pipeline status, and deployment progress.
4. **Accelerated Feedback Loops:**

- Enables rapid detection of defects, performance issues, and integration errors through automated testing and monitoring.
- Promotes proactive problem-solving, reducing downtime and improving overall software reliability.

Adopting DevOps practices not only optimizes the software development lifecycle but also aligns technical execution with organizational goals, resulting in higher-quality, resilient, and scalable software delivery.



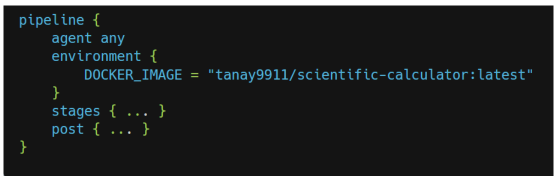
Tools used in this project:

- **GitHub –** Source control management
- **Python UnitTest –** Automated testing
- **Jenkins –** Continuous integration and pipeline automation
- **Docker –** Containerization
- **Docker Hub –** Docker image repository
- **Ansible –** Configuration management and deployment

3. Jenkins Pipeline

The Jenkinsfile defines the CI/CD pipeline for this project. The pipeline stages are as below:

3.1 Pipeline Overview



```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = "tanay9911/scientific-calculator:latest"
    }
    stages { ... }
    post { ... }
}
```

- agent any: Runs the pipeline on any available Jenkins agent.
- environment: Defines environment variables, including the Docker image name.

3.2 Stage 1: Checkout

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = "tanay9911/scientific-calculator:latest"
    }
    stages { ... }
    post { ... }
}
```

- Pulls the latest code from GitHub.
- Requires a GitHub Personal Access Token for authentication.

3.3 Stage 2: Build / Install Dependencies

```
stage('Build / Install Dependencies') {
    steps {
        echo "Installing Python dependencies..."
        sh 'python3 -m pip install --upgrade pip'
        sh 'pip install -r requirements.txt'
    }
}
```

- Updates pip and installs project dependencies from requirements.txt.
- Ensures all libraries, including Flask and pytest, are available.

3.4 Stage 3: Run Tests

```
stage('Run Tests') {
    steps {
        echo "Running unit tests..."
        sh 'python3 -m unittest discover tests'
    }
}
```

- Runs unit tests defined in tests/test_calc.py.
- Validates all calculator functions (√x, !x, ln(x), x^b).

3.5 Stage 4: Docker Build

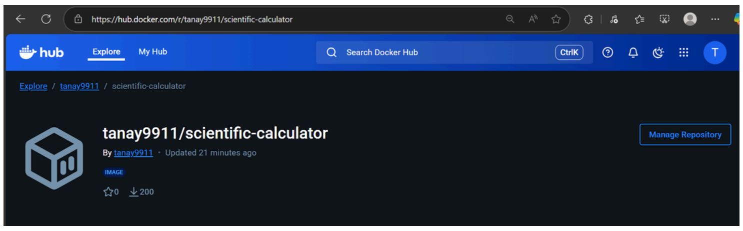
```
stage('Docker Build') {
    steps {
        echo "Building Docker image..."
        sh "docker build -t ${DOCKER_IMAGE} -f docker/Dockerfile ."
    }
}
```

- Builds Docker image using the docker/Dockerfile.
- Sets Python 3.8-slim as base image and copies app and templates.
- Exposes Flask port 5000.

3.6 Stage 5: Docker Push

```
stage('Docker Push') {
    steps {
        echo "Pushing Docker image to Docker Hub..."
        sh "docker push ${DOCKER_IMAGE}"
    }
}
```

- Pushes the Docker image to Docker Hub under tanay9911/scientific-calculator.


https://hub.docker.com/r/tanay9911/scientific-calculator

Explore / tanay9911 / scientific-calculator

tanay9911/scientific-calculator

By tanay9911 · Updated 21 minutes ago

IMAGE

☆0 ⤓200

3.7 Stage 6: Deploy via Ansible

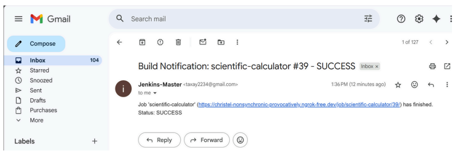
```
stage('Deploy via Ansible') {
    steps {
        echo "Running Ansible playbook to deploy Docker container..."
        sh 'ansible-playbook -i ansible/hosts.ini ansible/deploy.yml'
    }
}
```

- Uses Ansible to pull Docker image and run container on local machine.
- Removes existing container if present and ensures container restarts automatically.

3.8 Post Actions

```
post {
        always {
            script {
                echo "Sending notification email..."
                def status = currentBuild.currentResult
                def subject = "Build Notification: ${env.JOB_NAME} #${env.BUILD_NUMBER} - ${status}"
                def body = "Job '${env.JOB_NAME}' (${env.BUILD_URL}) has finished.\nStatus: ${status}"

                // Send mail using basic Email Notification plugin
                mail to: 'taxay2234@gmail.com',
                     subject: subject,
                     body: body
            }
        }
        success {
            echo "Build succeeded!"
        }
        failure {
            echo "Build failed!"
        }
    }
```

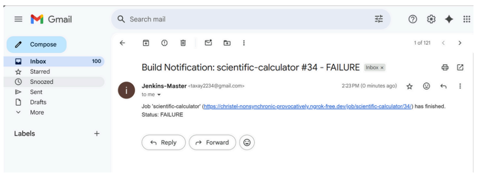
- Always: Sends email notification about build status.
- Success: Logs build success.
- Failure: Logs build failure.


Build Notification: scientific-calculator #39 - SUCCESS

Jenkins-Master <taxay2234@gmail.com> 1:36 PM (12 minutes ago)
to me

Job 'scientific-calculator' (https://christel-nonsynchronic-provocatively.ngrok-free.dev/job/scientific-calculator/39/) has finished.
Status: SUCCESS

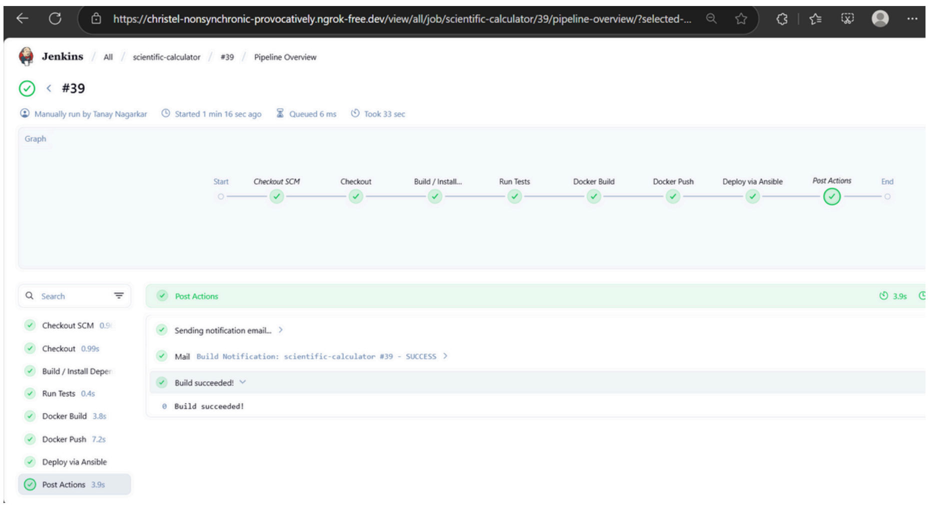
I created some errors myself to test for failure notification:


Build Notification: scientific-calculator #34 - FAILURE

Jenkins-Master <taxay2234@gmail.com> 2:23 PM (0 minutes ago)
to me

Job 'scientific-calculator' (https://christel-nonsynchronic-provocatively.ngrok-free.dev/job/scientific-calculator/34/) has finished.
Status: FAILURE

Full jenkins pipeline:


https://christel-nonsynchronic-provocatively.ngrok-free.dev/view/all/job/scientific-calculator/39/pipeline-overview/?selected-...

Jenkins / All / scientific-calculator / #39 / Pipeline Overview

#39

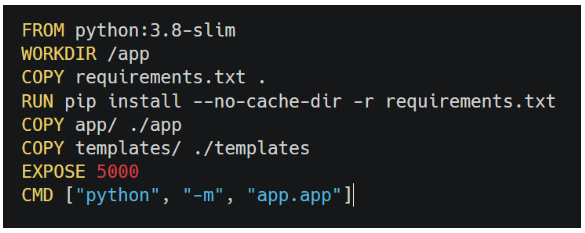
Manually run by Tanay Nagarkar Started 1 min 16 sec ago Queued 6 ms Took 33 sec

Post Actions 3.9s

Sending notification email...
Mail Build Notification: scientific-calculator #39 - SUCCESS
Build succeeded!

```
0 Build succeeded!
```

4. Dockerfile

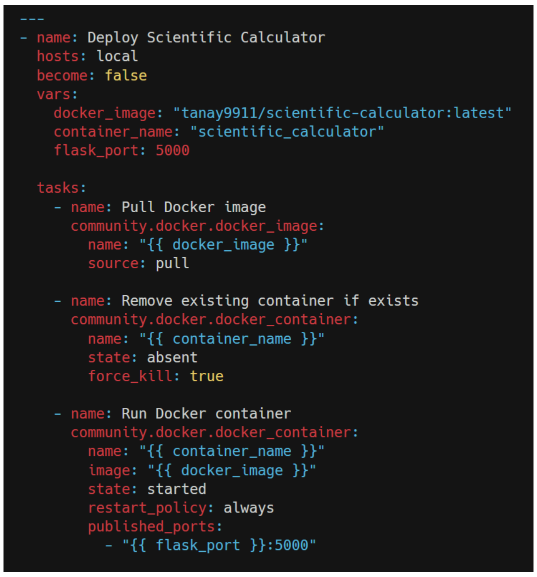


```
FROM python:3.8-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app/ ./app
COPY templates/ ./templates
EXPOSE 5000
CMD ["python", "-m", "app.app"]
```

- Base image: Python 3.8 slim.
- Workdir: /app.
- Dependencies: Installed from requirements.txt.
- Application copy: app and templates folders.
- Port: 5000 exposed for Flask app.
- Command: Runs Flask application**.**

5. Ansible Playbook

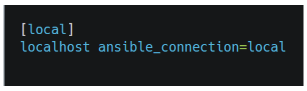
deploy.yml :



```
---
- name: Deploy Scientific Calculator
  hosts: local
  become: false
  vars:
    docker_image: "tanay9911/scientific-calculator:latest"
    container_name: "scientific_calculator"
    flask_port: 5000

  tasks:
    - name: Pull Docker image
      community.docker.docker_image:
        name: "{{ docker_image }}"
        source: pull

    - name: Remove existing container if exists
      community.docker.docker_container:
        name: "{{ container_name }}"
        state: absent
        force_kill: true

    - name: Run Docker container
      community.docker.docker_container:
        name: "{{ container_name }}"
        image: "{{ docker_image }}"
        state: started
        restart_policy: always
        published_ports:
          - "{{ flask_port }}:5000"
```

- Pulls Docker image from Docker Hub.
- Removes previous container if exists.
- Starts container on port 5000.

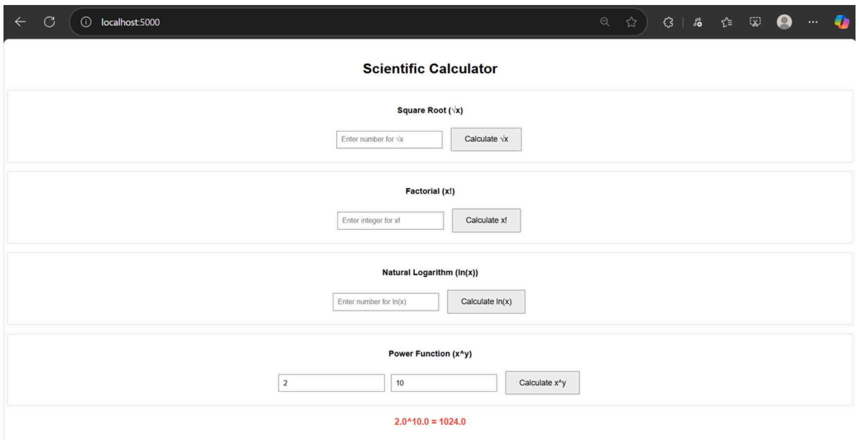
hosts.ini :



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

hosts.ini configures Ansible to run tasks locally on the same machine.

6. **Web Application** running on http://localhost:5000



```
2.0^10.0 = 1024.0
```

7. Links

- **GitHub Repository:** https://github.com/tanay9911/scientific-calculator
- **Docker Hub:** https://hub.docker.com/r/tanay9911/scientific-calculator

8. Conclusion

This mini project demonstrates:

- Development of a scientific calculator in Python.
- Integration of DevOps pipeline for CI/CD using Jenkins.
- Automated testing with unittest.
- Docker containerization and deployment using Ansible.
- Full cycle from code commit to automated deployment.