Mini Project: Scientific Calculator With DevOps

Farman Ahmed
MT2023129

September 25, 2024

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE
CSE 816 Software Production Engineering

Contents

1 Problem Statement

Create a scientific calculator program with the following user menu-driven operations:

- Square root function - $\sqrt{x}$
- Factorial function - $x!$
- Natural logarithm (base e) - $\ln(x)$
- Power function - x^b

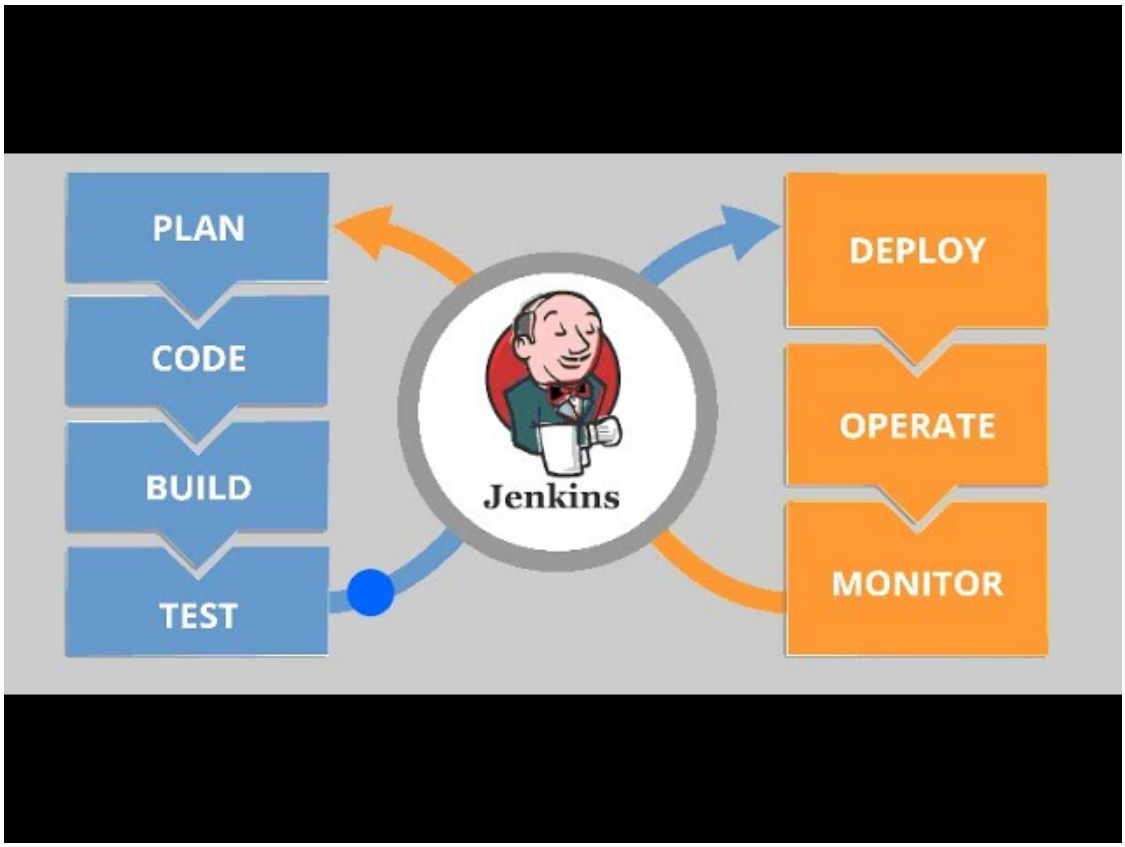
1.1 DevOps Tool Chain

You can use any set of DevOps tool chains you want, but the pipeline would be the same. The pipeline includes:

1. Using a source control management tool - like GitHub, GitLab, BitBucket, etc.
2. Testing - Test your code using either JUnit, Selenium, PyUnit, and many more.
3. Build - Build your code using tools like Maven, Gradle, Ant, and many more.
4. Continuous Integration - Continuously integrate your code using tools like Jenkins, GitLab CLI, Travis CLI, and many more.
5. Containerize - Containerize your code using Docker.
6. Push your created Docker image to Docker Hub.
7. Deployment - Do configuration management and deployment using Chef, Puppet, Ansible, Rundeck, etc. These tools will do configuration management, pull your Docker image, and run it on the managed hosts.
8. For deployment, you can either do it on your local machine or on Kubernetes cluster, OpenStack cloud, Amazon AWS, Google Cloud, or some other third-party cloud.

2 DevOps Introduction

DevOps is a combination of software development (dev) and operations (ops). It involves continuous development, testing, integration, deployment, and monitoring of software throughout the development cycle.



Coninuous Development involves planning and coding the software application's functionality. The vision of the project is decided in this phase. Code can be done in any language but must be maintained by a version control system.

2.1 Why Version Control System is Important?

Some of the major advantages of a version control system are:

1. It saves us from creating multiple backups of our files.
2. It allows multiple people to work on the same file.
3. It tracks the changes and also records who made those changes.
4. It is easy to switch to older versions as and when required.
5. It makes us more productive.

3 Tools Used

1. **Git and GitHub**: Git is a distributed version control tool that allows developers to track changes in the code. It is a DevOps tool used for source code management. It allows multiple developers to work together on a single project. It supports non-linear development through thousands of parallel branches. GitHub is a web-based platform that provides hosting for Git repositories. It serves as a centralized hub where developers can store, share, and collaborate on code.
2. **Maven**: Maven is an open-source build automation and project management tool widely used for Java applications. It automates source code compilation, dependency management, assembles binary codes into packages, and executes test scripts. Maven translates and packages your source code so that it becomes an executable application.
3. **Jenkins**: Jenkins is a widely-used automation server that plays a vital role in modern software development. It's like having a reliable assistant that takes care of repetitive tasks, allowing developers to focus on coding and innovation. Jenkins automates the process of building, testing, and deploying software applications, ensuring that code changes are integrated smoothly and deployed efficiently.
4. **Docker**: Docker is a tool that allows you to package up your applications and all their dependencies into tiny, portable containers. These containers are like little virtual environments that can run anywhere: on your laptop, in the cloud, or even on someone else's computer. Docker containers ensure that the application runs smoothly, consistently, and avoids issues like "it works on my machine" syndrome because the container has everything it needs to run regardless of the environment.
5. **Ansible**: Ansible is a software tool that provides simple but powerful automation for cross-platform computer support. It is primarily intended for IT professionals who use it for application deployment, updates on workstations and servers, cloud provisioning, configuration management, and intra-service orchestration. Ansible doesn't depend on agent software and has no additional security infrastructure, making it easy to deploy. Since Ansible relies on written instructions, it is easy to implement version control, supporting the "infrastructure as code" movement.

4 Source Code Management

The remote repository is created under the name `scientific_calculator`. Clone this repository using:

```
git clone https://github.com/farmanahmed888/scientific_calculator.git
```

5 Software Build and Testing

The `pom.xml` file includes dependencies for JUnit testing. Use Maven to build and test:

`mvn clean install`

Few Tests listed below:

```
@Test
    public void testCalSquareRoot() {
        ByteArrayInputStream in = new ByteArrayInputStream("4\n".getBytes());
        System.setIn(in);
        Scanner scanner = new Scanner(System.in);
        Main.calSquareRoot(scanner);
        assertEquals(2.0, Math.sqrt(4));
    }
```

```
@Test
    public void testCalFactorial() {
        ByteArrayInputStream in = new ByteArrayInputStream("5\n".getBytes());
        System.setIn(in);
        Scanner scanner = new Scanner(System.in);
        Main.calFactorial(scanner);
        assertEquals(120, factorial(5));
    }
```

```
@Test
    public void testCalNaturalLogarithm() {
        ByteArrayInputStream in = new ByteArrayInputStream("10\n".getBytes());
        System.setIn(in);
        Scanner scanner = new Scanner(System.in);
        Main.calNaturalLogarithm(scanner);
        assertEquals(2.302585092994046, Math.log(10));
    }
```

```
@Test
    public void testCalPowerFunction() {
        ByteArrayInputStream in = new ByteArrayInputStream("2\n3\n".getBytes());
        System.setIn(in);
        Scanner scanner = new Scanner(System.in);
        Main.calPowerFunction(scanner);
        assertEquals(8.0, Math.pow(2, 3));
    }
```

6 Jenkins and Pipeline

Install Jenkins using:

```
brew install jenkins-lts
brew services start jenkins-lts
```

Set up a pipeline to automate:

- Install Necessary Plugins
  - GitHub
  - Maven
  - Ansible
  - Docker
- Add global tool configuration
  - Maven
    - `Name`: Maven
    - `MAVEN_HOME`:/opt/homebrew/opt/maven
  - Ansible
    - `Name`: Ansible
    - `ANSIBLE_HOME`:/opt/homebrew/bin
  - Git
    - `Name`: Default
    - `PATH`: git
- Add Credentials
  - Docker
    - `username`: farmanahmed888
    - `password`: *SECRET*
  - Ansible
    - `username`: farmanahmed
    - `password`: *SECRET*
- Checkout from GitHub
- Maven build
- Run tests
- Build Docker image
- Push Docker image to DockerHub
- Run Ansible playbook for deployment

Jenkins script:

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE_NAME = 'scientific_calculator'
        GITHUB_REPO_URL = 'https://github.com/farmanahmed888/scientific_calculator.
           git'
        MAVEN_HOME = '/opt/homebrew/opt/maven'

    }
    stages {
        stage('Checkout') {
            steps {
                script {
                    // Checkout code from GitHub repository
                    git branch: 'main', url: "${env.GITHUB_REPO_URL}"
                }
            }
        }
        stage('Maven Build') {
                    steps {
                        script {
                            env.PATH = "${env.MAVEN_HOME}/bin:${env.PATH}"
                            sh 'mvn clean install'
                        }
                    }
                }
        stage('Build Docker Image') {
            steps {
                script {
                    // Build Docker image
                    sh 'docker build -t scientific_calculator .'
                }
            }
        }
        stage('Push Docker Images') {
            steps {
                script {
                    // Tag and push Docker image to Docker Hub
                    docker.withRegistry('', 'DockerHubCred') {
                        sh "docker tag scientific_calculator farmanahmed888/
                        scientific_calculator:latest"
                        sh "docker push farmanahmed888/scientific_calculator:latest
                           "
                    }
                }
            }
        }
        stage('Run Ansible Playbook') {
            steps {
                script {
                    // Run Ansible playbook
                    sh "/opt/homebrew/bin/ansible-playbook -i inventory deploy.yml"
                }
            }
        }
    }
}
```

7 Containerization with Docker

Steps for Docker containerization:

```
FROM openjdk:21
COPY ./target/scientific_calculator-1.0-SNAPSHOT.jar ./
WORKDIR ./
CMD ["java", "-cp", "scientific_calculator-1.0-SNAPSHOT.jar", "org.example.Main"]
```

8 Continuous Deployment

Continuous deployment is automated using Ansible. Define an inventory file and playbook for deploying the Docker container on local or cloud machines. Inventory File:

```
localhost ansible_user=farmanahmed ansible_ssh_pass=Farman@@5
ansible_python_interpreter=/Users/farmanahmed/anaconda3/bin/python
# ansible_ssh_common_args = '-o StrictHostKeyChecking=no'
```

Deploy.yml File:

```
---
- name: Pull Docker Image from Docker Hub
  hosts: localhost
  remote_user: farmanahmed888
  become: false
  tasks:
    - name: Pull Docker Image
      docker_image:
        name: "scientific_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
      when: ansible_facts['os_family'] == "Debian"

    - name: Running container
      shell: /usr/local/bin/docker run -it -d
      --name scientific_calculates scientific_calculator
```

9 Repository Reference Links

- https://github.com/farmanahmed888/scientific_calculator.git