SPE

Mini-Project Report

-Dhanvi Medha Beechu (IMT2020529)

Introduction

The point of this project is to get a hands-on understanding of the Dev-Ops pipeline by implementing a simple calculator application and go over all the steps involved in the pipeline.The calculator application referenced above will be able to implement the following functions.

- Addition
- Subtraction
- Division
- Multiplication

Approach and Tools used

I have used Debian as my Operating System and Java as the programming language to implement the simple command line calculator app.The version control system used in the project is git and the remote repository is maintained by github.We then used Docker to create image of the project with all the dependencies and pushed the image to DockerHub. We then pulled the image from dockerhub to our local system and containerised it. Finally, we used Ansible to deploy the container. Once deployed, we can run the container on our system. For hosting and triggering the automatic builds,we have used github webhooks and ngrok.

Github Repo Link: https://github.com/unbalancedvariance/SPE_Mini_Project

Docker Hub Link: https://hub.docker.com/repository/docker/unbalancedvariance/calc

Tools used:

- ***Maven***: a build automation tool used primarily for Java projects that manages dependencies, builds, and releases.
- ***Git***: a source code management tool that allows developers to collaborate on code and track changes.
- ***Github***: a web-based platform for software development that provides tools for version control, collaboration, and project management.
- ***Jenkins***: an open-source automation server that helps to automate parts of the software development process, such as building, testing, and deploying code. It is widely used for continuous integration and delivery (CI/CD) and can be integrated with a variety of other tools and technologies, such as Git, Docker, and AWS. Jenkins allows developers to automate repetitive tasks and streamline their workflows, which can improve efficiency and reduce errors in the software development process.
- ***Github webhooks***: a github feature that detects commits to a repository and automatically triggers the pipeline stages on every commit.
- ***Ngrok***: a tool that provides secure tunnels to localhost environments, allowing developers to expose web servers running on their local machines to the internet. It creates a public URL that can be used to access a local server, making it easier to test webhooks, APIs, and other web applications without deploying them to a public server.
- ***Docker***: a platform that enables developers to automate the deployment of applications inside containers, providing an isolated environment for running software.
- ***Ansible***: an open-source automation tool that simplifies IT configuration management and application deployment across servers, networks, and cloud environments using a simple, human-readable language.

Approach and Tools used:

I will now discuss all the steps involved in the project in detail and will include screenshots wherever required.

Installation of the tools mentioned above:

I have installed the latest jdk toolkit from the oracle website and also included maven as one of the dependencies in the installation.

The first step to running a java program is to create a .jar file first.

To build the project, I used:

maven clean will remove previously created jar files and other output files.

maven compile will compile the project.

maven install will create the jar file.

After setting up git and adding the relevant details in the git config file.I have run the following commands to initialize my repository and connect it to the remote repository in github.

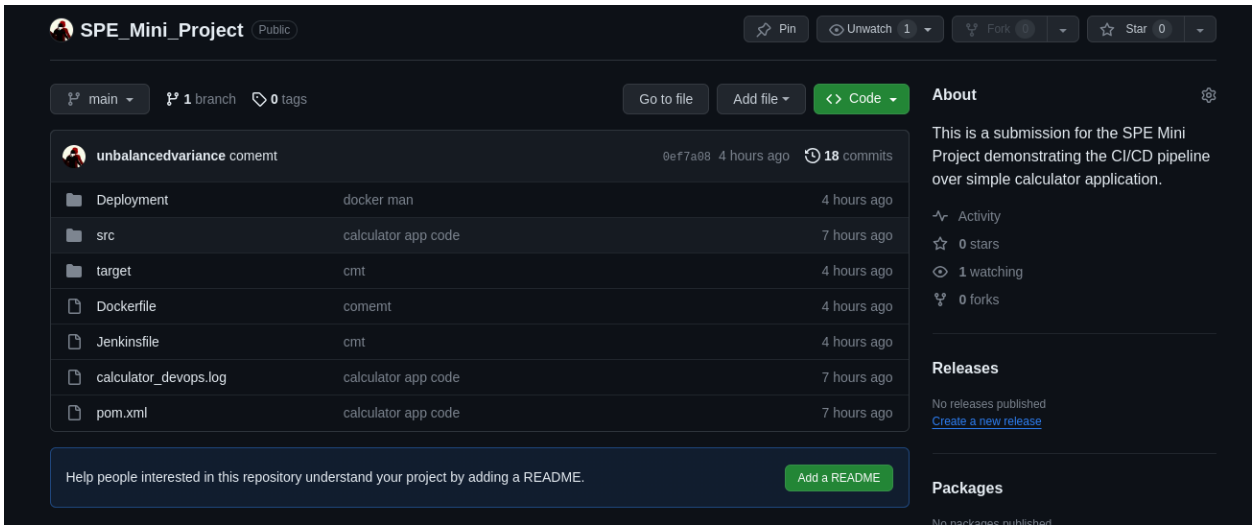
git init

git add

git remote add origin <url>

git commit -m <commit_msg>

git push origin branch_name



We can see that my repo is up and running in github.

Run the following commands to install Jenkins on your system.

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt install jenkins

sudo systemctl start jenkins

sudo systemctl enable jenkins

Make sure to use the initial password to login and change your password later after logging in using the initial admin credentials to maintain security.

Before moving any further, we need to install some important plugins that we will need to complete the pipeline.

Install the following plugins in Jenkins:

1. Ansible

2. Docker

3. Docker Pipeline

4. JUnit

Jenkins Pipeline Script:

This Jenkins pipeline script automates the Continuous Integration (CI) and Continuous Deployment (CD) process for a project hosted on GitHub. It consists of several stages, each performing specific tasks:

Agent Setup: The pipeline can run on any available agent. Environment variables CI and dockerimage are initialized.

Stage 1: Git Pull: This stage checks out the latest source code from a GitHub repository (unbalancedvariance/SPE_Mini_Project).The main branch is specified, and credentials can be configured to access the repository securely.

Stage 2: Build:The project is built using Maven with the command mvn clean install.

Stage 3: Build Docker Image:A Docker image is created using the project output.The image is tagged as unbalancedvariance/calc:latest.

Stage 4: Push Image to DockerHub:DockerHub credentials (DockerCred) are configured.The Docker image created in the previous stage is pushed to DockerHub under the unbalancedvariance/calc:latest repository.

Stage 5: Clean Docker Images:This stage cleans up unused Docker containers and images to free up resources on the build machine.

Stage 6: Pull the Docker Image:The Docker image from DockerHub (unbalancedvariance/calc) is pulled to prepare for deployment.

Stage 7: Ansible Deployment:Ansible is used for deploying the application.The ansiblePlaybook step is configured with Ansible-related settings, such as inventory and playbook.The deployment playbook (deploy.yml) is executed to deploy the application.

Summary:

This Jenkins pipeline automates the entire CI/CD process, starting from code retrieval from GitHub to deploying the application using Ansible. It incorporates best practices for Docker image management and ensures a smooth and reliable deployment process. By using Jenkins pipelines, the project can achieve efficient and consistent automation of software delivery.

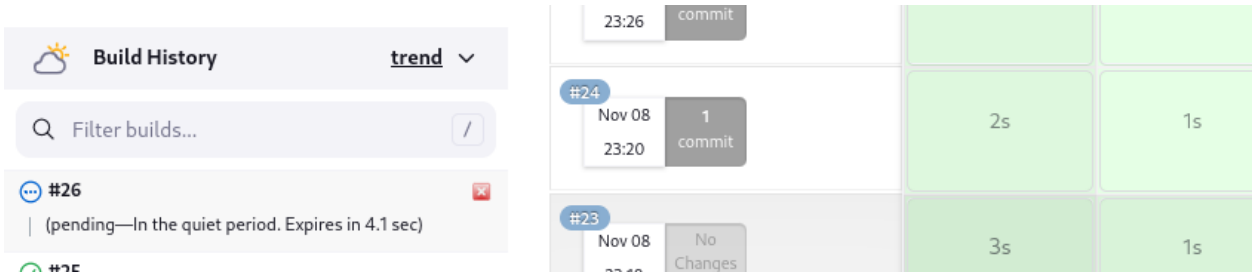
GitSCM Polling and Hosting the project using ngrok:

Enable the GitSCM Polling in Jenkins configure tab. We then need to add GitHub Credentials as Jenkins' global credentials. This will trigger the pipeline every time a change is pushed to the github repo.

Run ***ngrok 8080*** and It will create a forwarding URL, which we can copy and add as an observer in Github webhooks. So, whenever there is any commit in the github repo, all the webhooks will get triggered. We now need to make Jenkins recognise this forwarding URL by changing Jenkins' localhost to this forwarding URL.

Screenshot of the pipeline script describe above:

```
stages{
stage('Stage 1: Git Pull') {
  steps {
      git url: 'https://github.com/unbalancedvariance/SPE_Mini_Project.git',
      branch: 'main',
      credentialsId: ''
} }
stage('Stage 2:Build') {
  steps {
      sh 'mvn clean install'
} }

stage('Stage 3:Build Docker Image') {
  steps {
    script{
      dockerimage = docker.build "unbalancedvariance/calc:latest"
} }
}

stage('Stage 4:Push Image to dockerHub') {
    steps {
      script{
       docker.withRegistry('','DockerCred'){
       dockerimage.push()
       }
} }
}

stage('Stage 5: Clean docker images') {
  steps {
    script {
    sh 'docker container prune -f'
    sh 'docker image prune -f'
    }
  }
 }

 stage('Stage 6: Pull the docker image') {
  steps {
    script {
    sh 'docker pull unbalancedvariance/calc'
    }
  }
 }

stage('Step 7: Ansible Deployment') {
  steps {
    ansiblePlaybook(
```



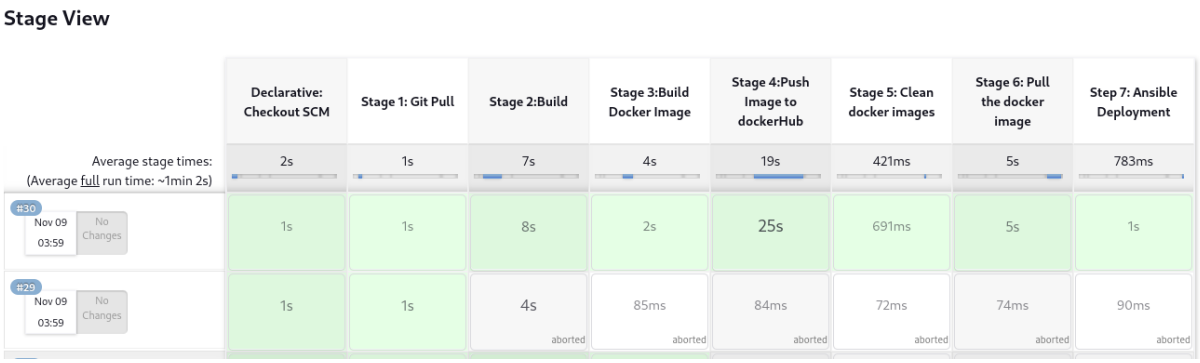
This is an example of a build job being triggered by Git SCM Polling.

ELK Stack:

I was getting ”ClassNotFoundException” in Java, while running the Project with Logger and LogManager. This part of the project is not implemented.

Screenshots of working of the project:

-> Jenkins Pipeline working



-

> Container Working

```
dhanvimedha@debian:~/Desktop/SPE_Mini_Project$ sudo docker container ls --all
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS                       PORTS     NAMES
35f0c1540057   unbalancedvariance/calc   "/bin/sh -c 'java -c…"   44 seconds ago   Exited (130) 9 seconds ago             calc
dhanvimedha@debian:~/Desktop/SPE_Mini_Project$ sudo docker container prune -f
Deleted Containers:
35f0c1540057ab5f852ceb8213f78db197d68b4322a18330b716ee40045b616b

Total reclaimed space: 75B
dhanvimedha@debian:~/Desktop/SPE_Mini_Project$ sudo docker container ls --all
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
dhanvimedha@debian:~/Desktop/SPE_Mini_Project$ sudo docker run -it -d --name calc unbalancedvariance/calc
dc7a44eefc1c4daa54a099ad3b57b3ba59009d244b01b29791fe466da7bce152
dhanvimedha@debian:~/Desktop/SPE_Mini_Project$ sudo docker container ls --all
CONTAINER ID   IMAGE                     COMMAND                  CREATED         STATUS         PORTS     NAMES
dc7a44eefc1c   unbalancedvariance/calc   "/bin/sh -c 'java -c…"   4 seconds ago   Up 2 seconds             calc
dhanvimedha@debian:~/Desktop/SPE_Mini_Project$ sudo docker start -a -i calc



1
Enter the 1st operand
2
Enter the 2nd operand
3
result = 5.0
Choose your operations
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
```