SPE Mini Project

Scientific Calculator

Akshat Abhishek Lal
MT2024015
March 1, 2025

Quick Links

- Github Repo: https://github.com/Akshat2920/ScientificCalcMiniProject
- DockerHub Repo: https://hub.docker.com/repository/docker/akshat2911/scientific-calculator/general
- Problem Statement: https://github.com/Akshat2920/ScientificCalcMiniProject/blob/master/Mini-Project-Scientific-Calculator.pdf

What is DevOps

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

Under a DevOps model, development and operations teams are no longer "siloed." Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

Key Principles of DevOps

- Automation
- Continuous Integrations
- Continuous Delivery
- Monitoring
- Version and Configuration Management

Project Components :

- git : Version Control
- Jenkins : CI/CD Pipleine and automation
- Docker : Containerization
- Ansible : Deployment

Other tools used:

- Python : source code and test cases
- Homebrew : necessary installation like docker and ansible to macOS
- Shell Scripting : commands to execute and run the automations
- Flask : Linking frontend and backend system
- Github webhook : Auto trigger of Jenkins builds via GitSCM
- NgRok : Provide an IP to link localhost, Jenkins and GitHub web-hooks
- Virtual Environment : A virtual environment is created where all the necessary installation for ansible can be fetched. It is done so as to avoid any version conflicts of dependencies.

Project Structure:

- Source Code
  - ScientificCalulator.py - Python Script containing mathematical functions to be executed,
  - TestCases.py - Unit test cases to test ScientificCalculator functions.
- Front End File
  - index.html - front end which connect to port:5000 interacting with backend
- Root files
  - Dockerfile - Defines instructions for building the docker image, which is the backed source code
  - Pilepline - Pipeline to autmomate entire process, i.e, fetching source code, building docker image, pushing it to dockerhub, using ansible to deploy.
  - deploy.yml - Ansible playbook for automated deployment
  - inventory.ini - Ansible inventory defining the targets

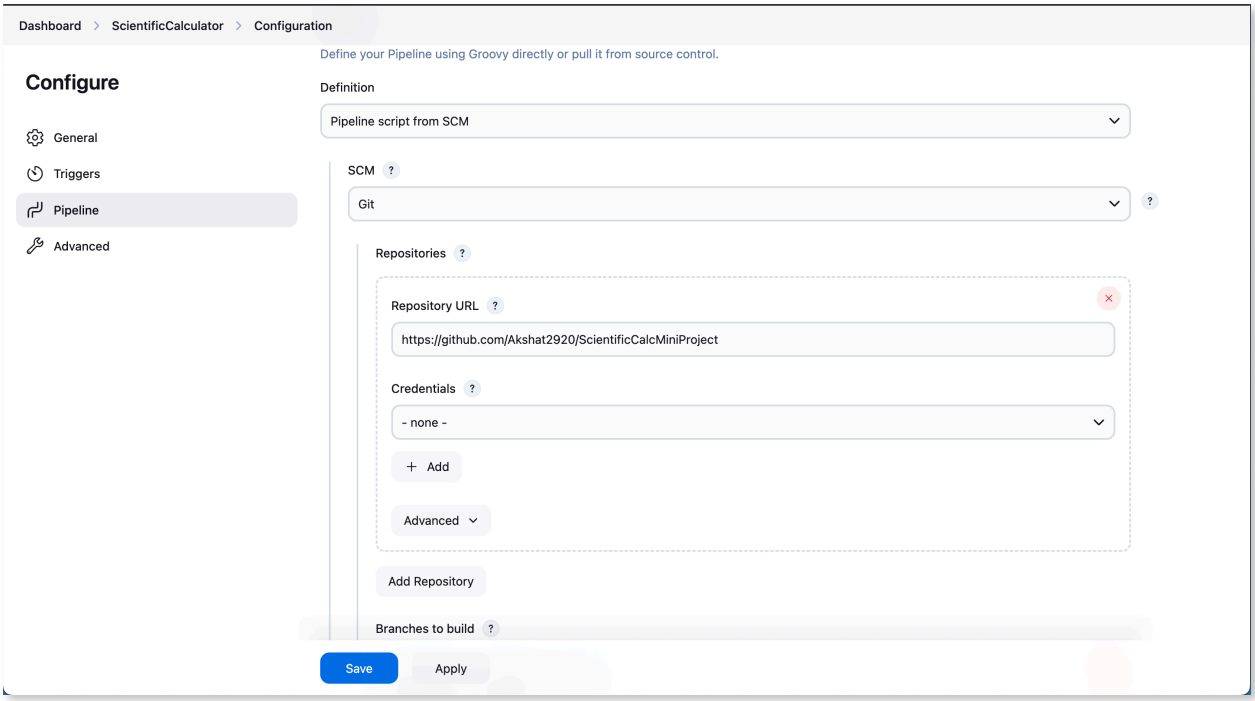
```python
from flask import Flask, request, jsonify
from flask_cors import CORS
import math

app = Flask(__name__)
CORS(app)  # Enable CORS for all routes

class ScientificCalculator:
    @staticmethod
    def square_root(x):
        if x < 0:
            raise ValueError("Square root of negative number is not allowed")
        return math.sqrt(x)
```

Pipleline Overview (via Jenkinss)

Jenkins fetch the Pipleine with following stages from the GitHub Repo.



The jenkins project is also hooked to the github, hence any changes in repo will automatically trigger a a new built, staring from fetching new version files to deployment.

Declarative: Checkout SCM	Checkout Code	Install Dependencies	Build Python Code	Run Python Tests	Build Docker Image	Login to Docker Hub	Push Docker Image	Cleanup Docker Images	Deploy using Ansible	Declarative: Post Actions
1s	986ms	411ms	456ms	454ms	1min 30s	3s	41s	2s	24s	128ms
1s	956ms	398ms	484ms	397ms	2min 11s	3s	37s	2s	1min 55s	156ms

Stages detail:

- Checkout Code : Pulls all the modules from GitHub Repo
- Install Dependences : Installs all the necessary dependencies if not present
- Build Python Code : Complie ScientificCalculator.py
- Run Python Tests : Checks for unit tests usinf TestCases.py
- Build Docker Image : Build docker image as defined by Dockerfile
- Login to Docker Hub : Login to push image
- Push Docker Image : Tag and push the image to DockerHub
- Cleanup Docker Images : Remove docker image from local
- Deploy using Ansible : Runs ansible deploy.yml file
- Post Stage - It's execution is independent of success/ failure of any of the above stages.
  - Success : Print success in the console
  - Failure : Print build failed in the console

```
post {
    success {
        echo "Build, Tests, and Docker Push Successful!"
    }
    failure {
        echo "Build or Tests Failed! Check logs for details."
    }
}
```

Deployment Overview (via ansible)

- Install necessary ansible-collections
- Login to DockerHub
- Pull the latest Docker image
- Stop and remove any existing container
- Stop and remove any existing container
- Run the Docker container
- Ensure that the container is running

```yaml
- name: Deploy Scientific Calculator Docker Container
  hosts: all
  become: yes
  tasks:
    - name: Install required Ansible collections
      ansible.builtin.command: ansible-galaxy collection install community.docker

    - name: Log in to Docker Hub
      docker_login:
        username: "aks
        password: "Aks

    - name: Pull the latest Docker image
      community.docker.docker_image:
        name: "akshat2911/scientific-calculator"
        tag: latest
        source: pull

    - name: Stop and remove any existing container
      docker_container:
        name: scientific_calculator
        state: absent

    - name: Run the Docker container
      docker_container:
        name: scientific_calculator
        image: "akshat2911/scientific-calculator:latest"
        state: started
        restart_policy: always
        published_ports:
          - "5000:5000"

    - name: Ensure the container is running
      command: /usr/local/bin/docker ps
```

“become: yes” will ensure sudo access to all the commands in the playbook. Without sudo, there is chances of restricted access based on system settings. More elaborating the deploy.yml file, “hosts: all” points to the inventory where networks with the label all are target.

```
[all]
localhost ansible_connection=local ansible_python_interpreter=/Users/akshatlal/.ansible-env/bin/python3
```

Docker Image and container:

Docker is the core of entire project. It is where the backend system, that is the functional components of the project will run.

The docker image will contain all the project files. As the process is automated, the files fetched from github repo will be copied from jenkins workspace to docker image. Post that, the necessary requirements and ssh will be installed in the docker image.

```
FROM python:3.11

WORKDIR /app

COPY . .

RUN pip install --no-cache-dir -r requirements.txt

RUN apt-get update && apt-get install -y openssh-server && \
    mkdir /var/run/sshd && \
    echo 'root:root' | chpasswd && \
    sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config && \
    sed -i 's@session required pam_loginuid.so@session optional pam_loginuid.so@g' /etc/pam.d/sshd && \
    echo "export VISIBLE=now" >> /etc/profile

EXPOSE 5000 22

CMD service ssh start && python ScientificCalculator.py
```

Ports are also exposed to connect via ssh. CMD will start ssh and execute source program.

Name	Tag	Image ID	Created	Size	Actions
akshat2911/scientific-ca	latest	6ea34096b307	11 seconds ag	1.85 GB	
scientific-calculator	latest	6ea34096b307	11 seconds ag	1.85 GB	

The build docker images and tag which is to be pushed to be pushed in dockerhub. As seen in the pipeline, post the push, both image as well as tag will be removed from the local system.

```
stage('Build Docker Image') {
    steps {
        sh "/usr/local/bin/docker build -t ${DOCKER_IMAGE_NAME} ."
    }
}

stage('Login to Docker Hub') {
    steps {
        withCredentials([usernamePassword(credentialsId: env.DOCKER_CREDENTIALS_ID, usernameVariable: 'DOCKER_USERNAME', passwordVariable: 'DOCKER_PASSWORD')]) {
            sh '''
                echo "$DOCKER_PASSWORD" | /usr/local/bin/docker login -u "$DOCKER_USERNAME" --password-stdin
            '''
        }
    }
}

stage('Push Docker Image') {
    steps {
        sh "/usr/local/bin/docker tag ${DOCKER_IMAGE_NAME} ${DOCKER_TAG}"
        sh "/usr/local/bin/docker push ${DOCKER_TAG}"
    }
}

stage('Cleanup Docker Images') {
    steps {
        sh "/usr/local/bin/docker rmi ${DOCKER_IMAGE_NAME} || true"
        sh "/usr/local/bin/docker rmi ${DOCKER_TAG} || true"
        sh "/usr/local/bin/docker system prune -f || true"
    }
}
```

After the deploy stage, the image will be fetched from dockerhub, and will be run in a container.

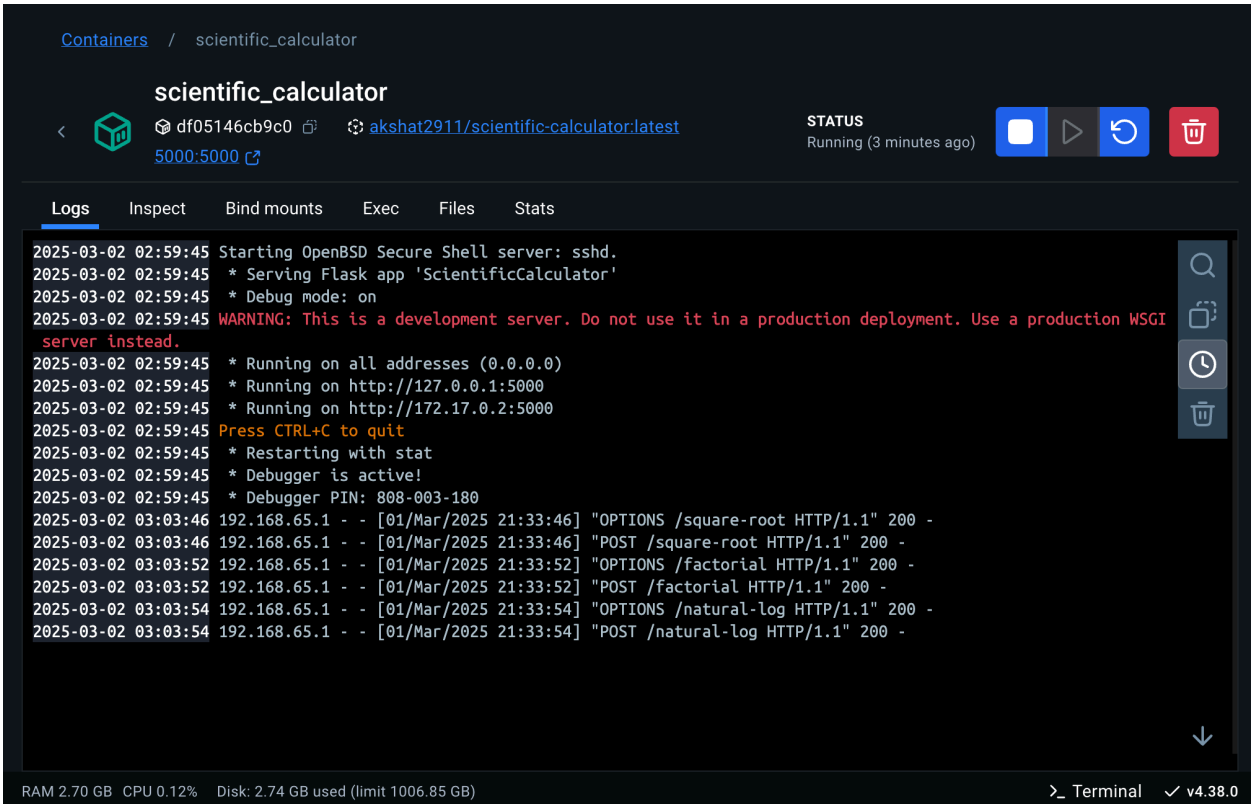
```yaml
- name: Pull the latest Docker image
  community.docker.docker_image:
    name: "akshat2911/scientific-calculator"
    tag: latest
    source: pull

- name: Stop and remove any existing container
  docker_container:
    name: scientific_calculator
    state: absent

- name: Run the Docker container
  docker_container:
    name: scientific_calculator
    image: "akshat2911/scientific-calculator:latest"
    state: started
    restart_policy: always
    published_ports:
      - "5000:5000"
```

Notice the port is 5000:5000, it means port 5000 of system will connect to port 5000 of container.

It is necessary that the localhost port be same as the target of frontend.



The logs of a running container. Notice the status “200” for POST and OPTION request by the frontend

```
const response = await fetch(`http://localhost:5000/${operation}`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(data)
});
```

Project Snapshots: