CSE 816

Software Production Engineering

Mini Project on

Scientifc Calculator with DevOps

By

Deepansh Pandey
MT2024038

Repository link:-

https://github.com/deepanshpandey/SPE_MiniProject

PREFACE

In today's fast-paced software development landscape, the demand for agility, reliability, and continuous innovation has led to the widespread adoption of DevOps. As organizations strive to bridge the gap between software development and IT operations, DevOps has emerged as a transformative approach, fostering collaboration, automation, and efficiency across the entire software lifecycle.

This report explores the significance of DevOps in modern software projects and provides a comprehensive understanding of its core principles, methodologies, and benefits. It delves into why DevOps is crucial for project success, emphasizing how it enhances deployment speed, scalability, and system stability while reducing time-to-market and operational overhead.

By integrating continuous integration (CI), continuous delivery (CD), infrastructure as code (IaC), automated testing, and monitoring, DevOps empowers teams to build and maintain high-quality software with greater agility and resilience. This report will highlight key reasons why DevOps is essential for project execution, detailing how it improves collaboration, automation, security, and efficiency.

INDEX

Setup:..

Calculator code:...

Configuring Pipeline:...

Setup Webhook:...

TOOLS USED

1. Github
2. Docker
3. Jenkins
4. Ansible
5. Python

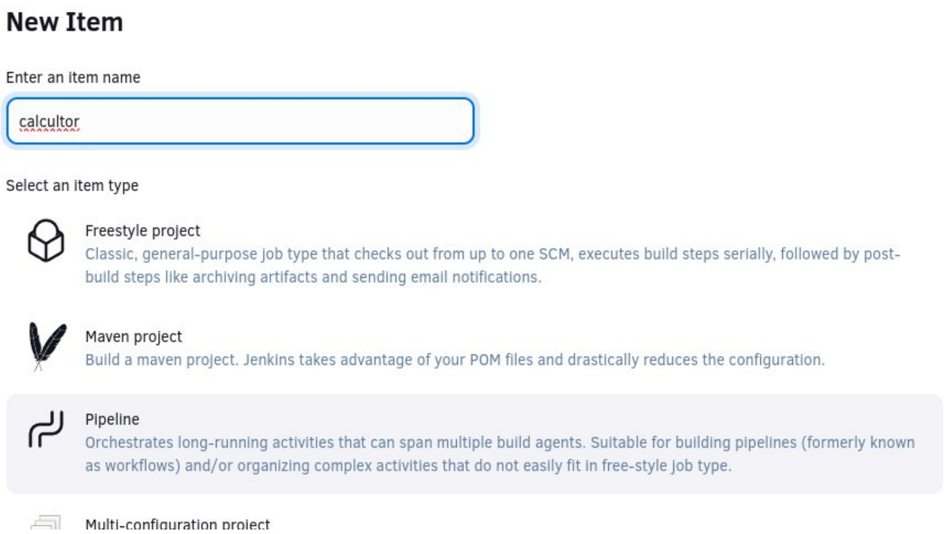
Setup

1. **Create Jenkins Project**

   Create a github repository Named after your choice for us its https://github.com/deepanshpandey/**SPE_MiniProject**

2. **Create Jenkins Project**

   Create a New item in Jenkins, with pipeline Selected



New Item

Enter an item name

calcultor

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project

3. **Install Plugins in Jenkins**
   - Docker: Docker Plugin, Pipeline, Docker-Build-step
   - Ansible: Ansible Plugin
   - SSH: SSH build Agent, SSH credentials plugin
4. **Docker HUB Account**

   Go to https://hub.docker.com/ and create an account. Note the username in there.

5. **Install Python**

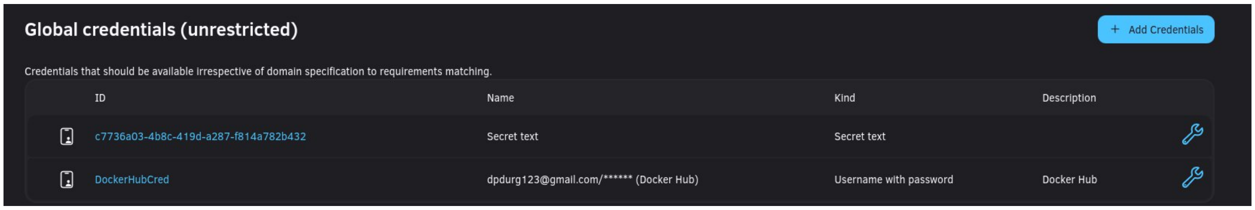
   Open terminal and install python using sudo apt install python3

6. **Install Docker, Ansible and SSH**

   sudo apt install –y docker.io ansible openssh-server

7. **Add credentials to jenkins**
   - Navigate to: manage Jenkins-> Credentials

- Click on Global
- Add the credentials (put DockerHubCred in ID field)
- Repeat and add ansible credntials (put ansible_ssh in ID field)



Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
c7736a03-4b8c-419d-a287-f814a782b432	Secret text	Secret text	
DockerHubCred	dpdurg123@gmail.com/****** (Docker Hub)	Username with password	Docker Hub

Calculator code

1. **Create Calculator Program**

#In your choice of IDE put the following code and give filename calculator.py

```
import math


def square_root(x):
    if x < 0:
        raise ValueError("Cannot compute square root of a negative number")
    return math.sqrt(x)


def factorial(n):
    if n < 0:
        raise ValueError("Cannot compute factorial of a negative number")
    return math.factorial(n)
```

```python
def natural_log(x):
    if x <= 0:
        raise ValueError("Cannot compute natural logarithm of non-positive number")
    return math.log(x)


def power(x, b):
    return x ** b


def main():
    options = {
        1: ("Square Root (√x)", square_root),
        2: ("Factorial (x!)", factorial),
        3: ("Natural Logarithm (ln(x))", natural_log),
        4: ("Power Function (x^b)", power)
    }

    print("Options:")
    for key, (description, _) in options.items():
        print(f"{key}. {description}")

    try:
        option = int(input("Enter your option: "))
        if option not in options:
            raise ValueError("Invalid option")
        value = float(input("Enter the value: "))
        if option == 4:
            exponent = float(input("Enter the exponent: "))
            result = options[option][1](value, exponent)
        else:
            result = options[option][1](value)

        print(f"Result: {result}")

    except ValueError as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    main()
```

2. Design Test Cases with Unittest

Create a file for caltest.py with following code.

```python
import unittest
import calculator

class TestCalculator(unittest.TestCase):
```

```
def test_square_root(self):
    self.assertAlmostEqual(calculator.square_root(9), 3.0)
    self.assertAlmostEqual(calculator.square_root(36), 6.0)
    self.assertAlmostEqual(calculator.square_root(49), 7.0)
    self.assertAlmostEqual(calculator.square_root(64), 8.0)

def test_factorial(self):
    self.assertEqual(calculator.factorial(2), 2)
    self.assertEqual(calculator.factorial(3), 6)
    self.assertEqual(calculator.factorial(4), 24)
    self.assertEqual(calculator.factorial(7), 5040)

def test_natural_log(self):
    self.assertAlmostEqual(calculator.natural_log(1), 0.0)
    self.assertAlmostEqual(calculator.natural_log(10), 2.3025850929)
    self.assertAlmostEqual(calculator.natural_log(7), 1.9459101491)
    self.assertAlmostEqual(calculator.natural_log(20), 2.9957322736)

def test_power(self):
    self.assertAlmostEqual(calculator.power(3, 3), 27.0)
    self.assertAlmostEqual(calculator.power(2, 4), 16.0)
    self.assertAlmostEqual(calculator.power(6, 2), 36.0)
    self.assertAlmostEqual(calculator.power(3, 4), 81.0)
```

```
if __name__ == '__main__':
    unittest.main()
```

3. **Build and Test**
    - In Terminal type “python3 ./calculator.py”.
    - In another terminal “python3 ./caltest.py”.

Configuring Pipeline

1. **Push to GitHub**

    Follow the below commands in Project directory to push it to github

    - git init && git remote add origin https://github.com/deepanshpandey/SPE_MiniProject/
    - git add . && git commit –m “first commit”

2. **Create a DockerFile**

    Create a file named Dockerfile and put following in it

```
FROM python:3.11-slim
# Set working directory
WORKDIR /app
# Copy application files
COPY calculator.py caltest.py
# Default command to run the application
CMD ["python3", "calculator.py"]
```

3. Create Deploy.yml

Create a file named deploy.yml

```yaml
---
- name: Deploy Python Calculator
  hosts: localhost
  remote_user: deepanshpandey
  become: false
  environment:
    DOCKER_HOST: "unix:///var/run/docker.sock"
  tasks:
    - name: Pull the latest Docker image
      community.general.docker_image:
        name: coffeeinacafe/calpy
        source: pull
      register: docker_pull_result
    - name: Display Docker Pull Result
      debug:
        var: docker_pull_result
    - name: Stop and remove existing container if running
      shell: docker stop calpy && docker rm calpy
    - name: Start Docker service
      service:
        name: docker
        state: started
    - name: Running container
      shell: docker run -it -d --name calpy coffeeinacafe/calpy
```

4. Create Inventory

Create a file named Inventory and put the following

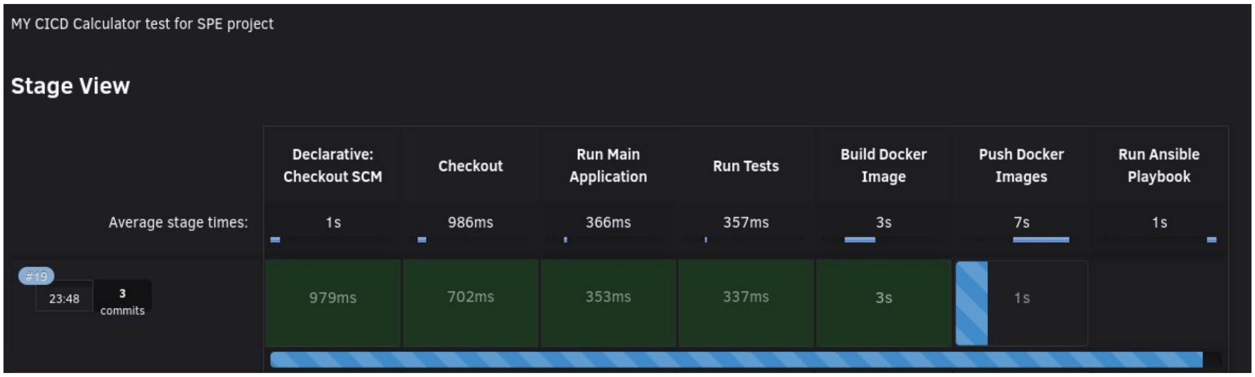
```
localhost ansible_connection=local
```

5. Create JenkinsFile

Create a file named Jenkinsfile and put following in it.

```
pipeline {
    agent any


    environment {
        DOCKER_IMAGE_NAME = 'calpy'
        GITHUB_REPO_URL = 'https://github.com/deepanshpandey/SPE_MiniProject.git'
    OPTION = 1
    NUMBER = 2
    EXP = 3
    }


    stages {
        stage('Checkout') {
            steps {
                script {
                    git branch: 'main', url: "${GITHUB_REPO_URL}"
                }
            }
        }
        stage('Run Main Application') {
            steps {
                script {
                    sh "echo '${OPTION}\n${NUMBER}\n${EXP}' | python3 calculator.py"
                }
            }
        }
        stage('Run Tests') {
            steps {
                script {
                    sh 'python3 caltest.py'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    // Build Docker image
                    docker.build("${DOCKER_IMAGE_NAME}", '.')
                }
            }
        }

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

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

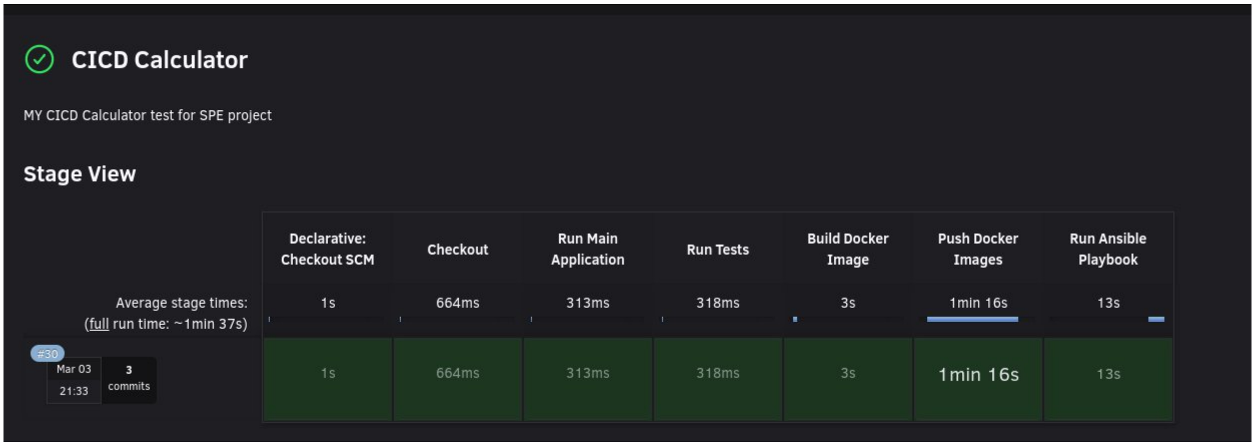
6. **Execution**
    - Goto jenkins->Calculator and click build now
    - The pipeline should build successfully now



MY CICD Calculator test for SPE project

Stage View

	Declarative: Checkout SCM	Checkout	Run Main Application	Run Tests	Build Docker Image	Push Docker Images	Run Ansible Playbook
Average stage times:	1s	986ms	366ms	357ms	3s	7s	1s
#19 23:48 3 commits	979ms	702ms	353ms	337ms	3s	1s	



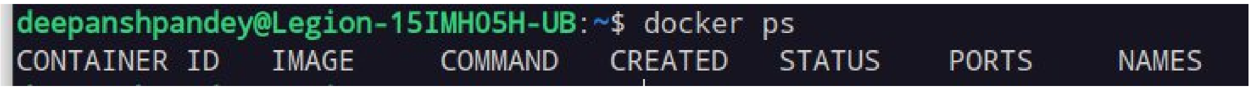
CICD Calculator

MY CICD Calculator test for SPE project

Stage View

	Declarative: Checkout SCM	Checkout	Run Main Application	Run Tests	Build Docker Image	Push Docker Images	Run Ansible Playbook
Average stage times: (full run time: ~1min 37s)	1s	664ms	313ms	318ms	3s	1min 16s	13s
#30 Mar 03 21:33 3 commits	1s	664ms	313ms	318ms	3s	1min 16s	13s

- Once done check the container is runing by typing “docker ps” in terminal



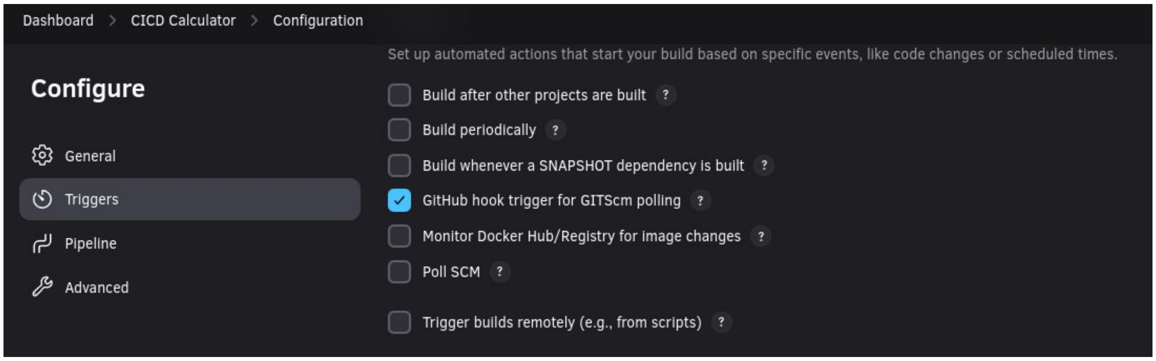
```
deepanshpandey@Legion-15IMH05H-UB:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

- Then you can run the program by

  docker run -it [dockerHubUsername]/[filename]

Setup Webhook

1. **Configure Build Trigger in Jenkins**
    - Navigate to Jenkins->Calculator->Configure
    - Then to General->Triggers->
    - Enable an option named “Github hook trigger for GITScm Pooling”



Set up automated actions that start your build based on specific events, like code changes or scheduled times.

☐ Build after other projects are built ?
☐ Build periodically ?
☐ Build whenever a SNAPSHOT dependency is built ?
☑ GitHub hook trigger for GITScm polling ?
☐ Monitor Docker Hub/Registry for image changes ?
☐ Poll SCM ?
☐ Trigger builds remotely (e.g., from scripts) ?

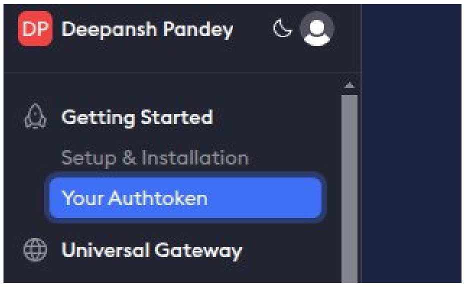
- Save the changes.

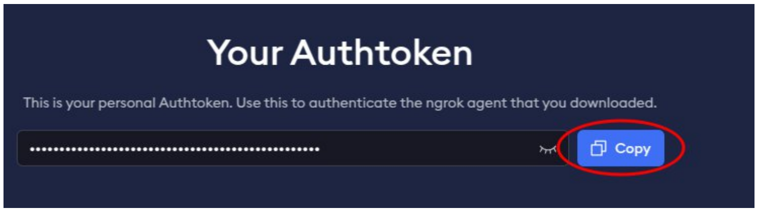
2. Setup NGROK

- Install ngrok with the following command

```
curl -sSL https://ngrok-agent.s3.amazonaws.com/ngrok.asc   | sudo tee
/etc/apt/trusted.gpg.d/ngrok.asc >/dev/null    && echo "deb https://ngrok-
agent.s3.amazonaws.com buster main"  | sudo tee
/etc/apt/sources.list.d/ngrok.list     && sudo apt update      && sudo apt
install ngrok
```

- Setup your account on dashboard.ngrok.com/signup
- Navigate to “Your AuthToken” and copy it.



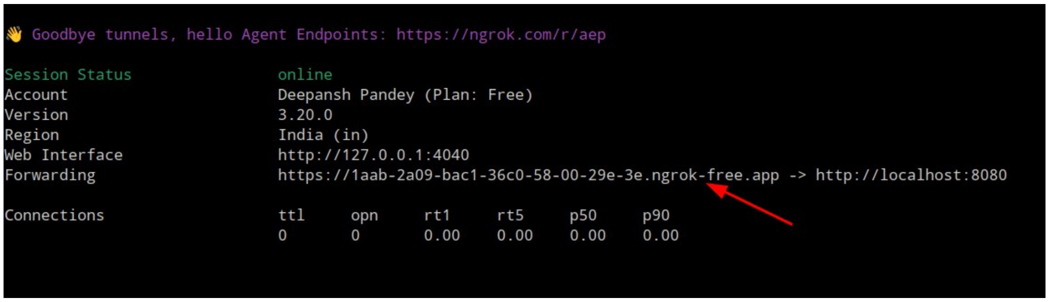


Your Authtoken

This is your personal Authtoken. Use this to authenticate the ngrok agent that you downloaded.

••

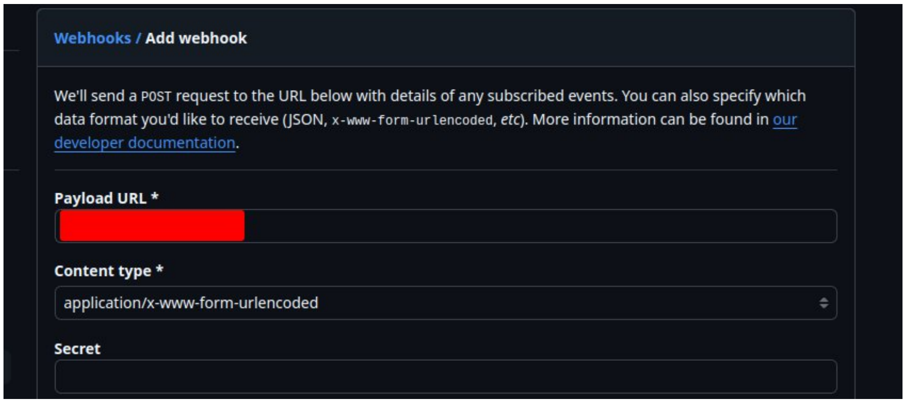
- In terminal execute “ngrok config add-authtoken [your AuthToken]”
- Run “ngrok http 8080”
- Copy the link under forwarding



```
👋 Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

Session Status                online
Account                       Deepansh Pandey (Plan: Free)
Version                       3.20.0
Region                        India (in)
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://1aab-2a09-bac1-36c0-58-00-29e-3e.ngrok-free.app -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

3. Add Webhook to Repository

- Go to GitHub Repo->settings->Webhooks
- Add “[copiedforwadinglink]/github-webook/” in Payload URL Field.



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *

[redacted]

Content type *

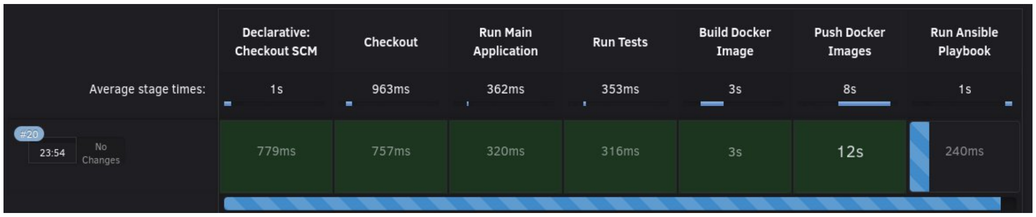
application/x-www-form-urlencoded

Secret

- Save the changes.

4. Execution

- Make any minor change in the repository then commit and push.
- Check Jenkins Tab and the build should have been started automatically.
- The build should be successfull now.



	Declarative: Checkout SCM	Checkout	Run Main Application	Run Tests	Build Docker Image	Push Docker Images	Run Ansible Playbook
Average stage times:	1s	963ms	362ms	353ms	3s	8s	1s
#20 23:54 No Changes	779ms	757ms	320ms	316ms	3s	12s	240ms

- You can check if the code is running properly by “docker run –it [DockerHubUsernaem]/[Filename]” the run should reflect newly made changes.