Scientific Calculator DevOps Pipeline Report

Name: Anirudh Pathaneni
ID: IMT2022505
Course: CS 816 SPE
Date: October 10, 2025

What and Why of DevOps?

DevOps integrates development and operations to automate and streamline the software delivery process, ensuring faster, reliable, and consistent deployments. For the Scientific Calculator project, DevOps enables automated testing, containerization, and deployment, reducing manual errors and ensuring the application is always production-ready. The pipeline automates code pushes to GitHub, triggers build via Jenkins, tests the code, builds and pushes Docker images, and deploys locally using Ansible, aligning with CI/CD principles.

Tools Used

- **Jenkins**: Automates the CI/CD pipeline with a Jenkinsfile.
- **Docker**: Containerizes the application for consistent deployment.
- **Docker Hub**: Stores Docker images.
- **Ansible**: Manages local deployment of containers.
- **ngrok**: Exposes Jenkins for GitHub webhooks.
- **GitHub**: Hosts the source code and triggers the pipeline.

Setup and Configurations

- **Jenkins** :
    - Installed via Homebrew: brew install jenkins-lts.
    - Configured jenkins-lts.plist for Docker (/usr/local/bin) and Ansible (/Users/anirudhpathaneni/.pyenv/shims).
    - **Jenkins Pipeline Configuration**:

    - Created a pipeline job in Jenkins (scalculator-pipeline).
    - Set SCM to https://github.com/anirudh-pathaneni/ScientificCalculator-DevOps.git, branch main.
    - Enabled "GitHub hook trigger for GITScm polling" under Build Triggers.
    - Installed plugins: GitHub Integration, Docker Pipeline, Email Extension.
    - Configured credentials for Docker Hub (docker-hub-credentials) in Manage Jenkins > Manage Credentials.
- **ngrok Setup**:
    - Ran ngrok http 8080 --url=https://tegminal-enjambed-coreen.ngrok-free.dev.
    - Configured GitHub webhook with /github-webhook/.
- **Ansible**:
    - Installed via pip3 install ansible and ansible-galaxy collection install community.docker.
    - Created ansible/deploy.yml to pull and run the Docker image locally.
    - Configured inventory (localhost ansible_connection=local) for local deployment.
- **Docker**: Installed Docker Desktop.
- **GitHub Webhook**:
    - Added webhook in GitHub: Settings > Webhooks > Add webhook, set Payload URL to https://tegminal-enjambed-coreen.ngrok-free.dev/github-webhook/, selected "Just the push event".

Workflow

The pipeline, defined in the Jenkinsfile, automates the build, test, and deployment process for the Scientific Calculator. Below is a detailed explanation of each step, including setup, commands, and results, with placeholders for screenshots to demonstrate the process.

1. Push to GitHub

- **Description**: Code changes are committed and pushed to the main branch of the GitHub repository.
- **Purpose**: Initiates the CI/CD pipeline via a webhook.
- **Command**: git commit -m "Update code" && git push origin main
- **Result**: Triggers the webhook, notifying Jenkins.

```
anirudhpathaneni@Anirudhs-MacBook-Pro ScientificCalculator-DevOps % git add .
anirudhpathaneni@Anirudhs-MacBook-Pro ScientificCalculator-DevOps % git commit -m "Update code" && git push origin main
[main 170e82c] Update code
 1 file changed, 1 insertion(+), 1 deletion(-)
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 10 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:anirudh-pathaneni/ScientificCalculator-DevOps.git
   9594304..170e82c  main -> main
```

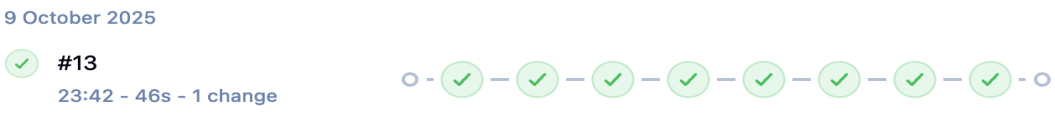
2. Trigger Webhook

- **Description**: A GitHub webhook, configured with ngrok, sends a POST request to Jenkins when the code is pushed.
- **Setup**: ngrok exposes Jenkins (http://localhost:8080) at https://tegminal-enjambed-coreen.ngrok-free.dev/github-webhook/. Webhook added in GitHub repo settings.
- **Command**: ngrok http 8080 --url=https://tegminal-enjambed-coreen.ngrok-free.dev
- **Result**: Webhook delivers a 200 OK response, triggering a Jenkins build.

```
ngrok

Decouple policy and sensitive data with vaults: https://ngrok.com/r/secrets

Session Status                online
Account                       Anirudh Pathaneni (Plan: Free)
Version                       3.30.0
Region                        India (in)
Latency                       37ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://tegminal-enjambed-coreen.ngrok-free.dev -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              1       0       0.01    0.00    30.09   30.09

HTTP Requests
-------------

23:42:00.916 IST POST /github-webhook/          200 OK
```

3. Trigger Jenkins via Jenkinsfile (for Stage View)

- **Description**: Jenkins processes the webhook and executes the Jenkinsfile, visualized in Stage View.
- **Setup**: Created a pipeline job in Jenkins, set SCM to https://github.com/anirudh-pathaneni/ScientificCalculator-DevOps.git, enabled "GitHub hook trigger for GITScm polling".
- **Result**: Build#13 runs all stages, visible in Stage View.



4. Pull GitHub Repo

- **Description**: Jenkins clones the repository to fetch the latest code.
- **Jenkinsfile Step**: stage('Pull Repo') {
  steps {
  git url: 'https://github.com/anirudh-pathaneni/ScientificCalculator-DevOps',
  branch: 'main'
  }
  }

- **Result**: Cloned commit (e.g., 43d5bf049d5c...) with message "Update code".

```
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Pull Repo)
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /Users/anirudhpathaneni/.jenkins/workspace/scalculator-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/anirudh-pathaneni/ScientificCalculator-DevOps # timeout=10
Fetching upstream changes from https://github.com/anirudh-pathaneni/ScientificCalculator-DevOps
 > git --version # timeout=10
 > git --version # 'git version 2.50.1 (Apple Git-155)'
 > git fetch --tags --force --progress -- https://github.com/anirudh-pathaneni/ScientificCalculator-DevOps +refs/heads/*:ref
timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 170e82c20f6c8ef3f98969aa306d11e149315179 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 170e82c20f6c8ef3f98969aa306d11e149315179 # timeout=10
 > git branch -a -v --no-abbrev # timeout=10
 > git checkout -b main 170e82c20f6c8ef3f98969aa306d11e149315179 # timeout=10
Commit message: "Update code"
```

5. Run Test Case

- **Description**: Executes unit tests for the calculator.
- **Jenkinsfile Step**:stage('Run Tests') {
  steps {
  sh 'python3 -m unittest test_scalculator.py'
  }
  }

- **Result**: Ran 4 tests in 0.000s, all passed (OK).

```
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Run Tests)
[Pipeline] sh
+ python3 -m unittest test_scalculator.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.000s

OK
```

6. Build Image

- **Description**: Builds a Docker image for the calculator.
- **Jenkinsfile Step**:stage('Build Image') {
  steps {
  script {
  dockerImage =
  docker.build("${DOCKER_HUB_REPO}:${env.BUILD_NUMBER}")
  }
  }
  }

- **Result**: Built 5unnysunny/scalculator:13 using Dockerfile.

```
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Image)
[Pipeline] script
[Pipeline] {
[Pipeline] isUnix
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ docker build -t 5unnysunny/scalculator:13 .
#0 building with "desktop-linux" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 156B 0.0s done
#1 DONE 0.0s

#2 [internal] load metadata for docker.io/library/python:3.12-slim
#2 ...

#3 [auth] library/python:pull token for registry-1.docker.io
#3 DONE 0.0s

#2 [internal] load metadata for docker.io/library/python:3.12-slim
#2 DONE 2.3s

#4 [internal] load .dockerignore
#4 transferring context: 2B done
#4 DONE 0.0s

#5 [internal] load build context
#5 transferring context: 2.63kB done
#5 DONE 0.0s

#6 [1/4] FROM docker.io/library/python:3.12-slim@sha256:47ae396f09c1303b8653019811a8498470603d7ffefc29cb07c88f1f8cb3d19f
#6 resolve docker.io/library/python:3.12-slim@sha256:47ae396f09c1303b8653019811a8498470603d7ffefc29cb07c88f1f8cb3d19f 0.0s done
#6 DONE 0.0s

#7 [2/4] WORKDIR /app
#7 CACHED

#8 [3/4] COPY scalculator.py .
#8 CACHED

#9 [4/4] COPY test_scalculator.py .
#9 DONE 0.0s

#10 exporting to image
#10 exporting layers 0.0s done
#10 exporting manifest sha256:4528e0edf108123e2fd8d807d04c33d10622e2f612f73f777b07eb041a393931
#10 exporting manifest sha256:4528e0edf108123e2fd8d807d04c33d10622e2f612f73f777b07eb041a393931 done
#10 exporting config sha256:080655961982402cc020afb3dcd149f779f7f4b66c1c571d4fb6e047e9f75c7c done
#10 exporting attestation manifest sha256:67a1d8ee851d217fc5b97d29fc373139f694cdacae4a0b94c72aace8b378c529 done
#10 exporting manifest list sha256:7ada6b553f9964db5f86344611fcba7507532250f6f502ee1b4dc773d453f9e4 done
#10 naming to docker.io/5unnysunny/scalculator:13 done
#10 unpacking to docker.io/5unnysunny/scalculator:13 0.0s done
#10 DONE 0.1s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ouxu54rzof4ncwkuph5pmd47n
```

7. Login to Docker Hub

- **Description**: Authenticates with Docker Hub to push the image.

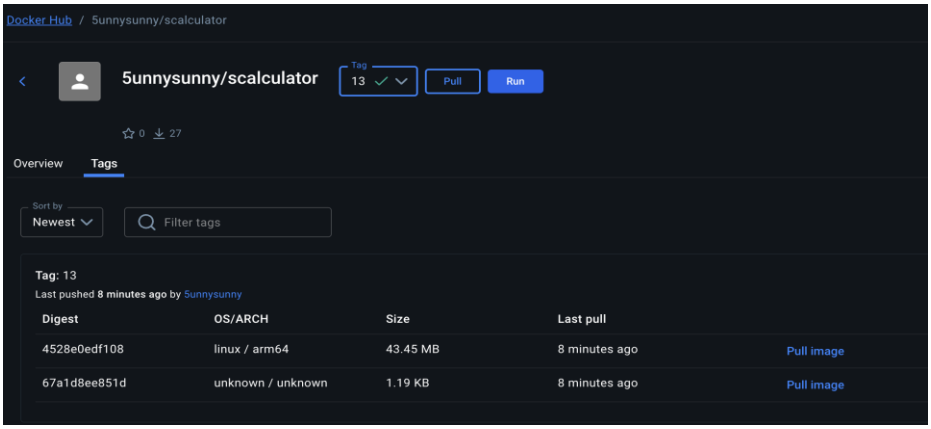
- **Jenkinsfile Step**:stage('Login to Docker Hub') {
  steps {
    withCredentials([usernamePassword(credentialsId: "${DOCKER_CREDENTIALS_ID}", usernameVariable: 'DOCKER_USERNAME', passwordVariable: 'DOCKER_PASSWORD')]) {
      sh "echo ${DOCKER_PASSWORD} | docker login -u ${DOCKER_USERNAME} --password-stdin"
    }
  }
}

- **Result**: Successful login with username.

8. Push Image

- **Description**: Pushes the built image to Docker Hub.
- **Jenkinsfile Step**:stage('Push Image') {
  steps {
    sh "docker push ${DOCKER_HUB_REPO}:${env.BUILD_NUMBER}"
  }
}

- **Result**: Pushed 5unnysunny/scalculator:13 (digest sha256:4ec885e3...).

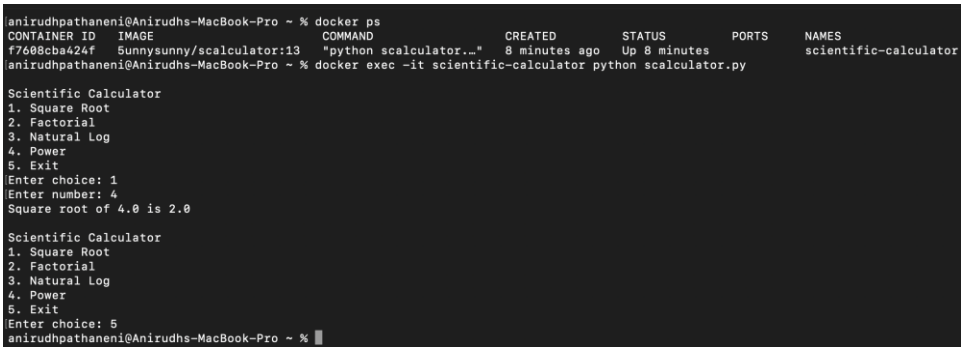


9. Deploy on Local System

- **Description**: Deploys the container locally using Ansible.
- **Jenkinsfile Step**:stage('Deploy') {
  steps {
  sh 'ansible-playbook -i ansible/inventory.ini, ansible/deploy.yml --extra-vars "image_tag=${BUILD_NUMBER}"'
  }
  }

- **Result**: Container scientific-calculator running, verified with docker ps.

```
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy) (hide)
[Pipeline] sh
+ ansible-playbook -i ansible/inventory.ini ansible/deploy.yml --extra-vars image_tag=14

PLAY [Deploy Scientific Calculator Container] **********************************

TASK [Gathering Facts] *********************************************************
[WARNING]: Host 'localhost' is using the discovered Python interpreter at
'/Users/anirudhpathaneni/.pyenv/versions/3.11.11/bin/python3.11', but future installation of another Python interpreter
could cause a different interpreter to be discovered. See https://docs.ansible.com/ansible-
core/2.19/reference_appendices/interpreter_discovery.html for more information.
ok: [localhost]

TASK [Pull Docker image] *******************************************************
ok: [localhost]

TASK [Run container] ***********************************************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```



Links

- **GitHub**: https://github.com/anirudh-pathaneni/ScientificCalculator-DevOps
- **Docker Hub**: https://hub.docker.com/r/5unnysunny/scalculator

Conclusion

The Scientific Calculator DevOps pipeline exemplifies the ease of CI/CD by automating the entire software delivery lifecycle—from code push to deployment—with minimal manual intervention. By leveraging Jenkins for orchestration, Docker for consistent environments, Ansible for seamless deployment, and ngrok for reliable webhook integration, the pipeline ensures rapid feedback, consistent builds, and error-free deployments. Challenges like tool path errors and webhook misconfigurations were resolved, resulting in a robust, scalable workflow. This automation reduces development cycles, enhances reliability, and allows developers to focus on coding rather than manual processes.