Agile Software Engineering

for

DevOps

Draft

Prepared by Pranav P Kulkarni

Group Name: BTech Section C

Pranav P Kulkarni	1RVU22CSE119	pranavpk.btech22@rvu.edu.in

Instructor:	**Prof. CVSN Reddy**
Course:	**Agile Software and Engineering**
Lab Section:	*Project 2 – DevOps*
Teaching Assistant:	*Divya M*
Date:	**03 November 2023**

o

1 SDLC Life Cycle

1.1 Phases –

- **SoW**: Single line statement of what the customer wants.
- **Requirements**: Analyze the different functional requirements of the product.
- **Requirements Analysis**: Break down high level requirements into its lower counterpart.
- **Design**: Identify the functions/methods of the product.
- **Coding**: Write code as per coding guidelines.
- **UT**: Unit Testing which is line-by-line testing of the code.
- **IT**: Integration Testing which is integrating code of different developers and testing how they mesh with each other.
- **ST**: System Testing, done by system testers where we test the functionality of the product.
- **Deployment**: Delivering the product built as per the agreement.
- **Maintenance**: Providing software updates to fix bugs or add more features to the existing product.

Software Development Life Cycle (SDLC) Phases

Software Development Life Cycle (SDLC) Phases

1. Planning & Analysis
Gather all relevant information from stakeholders and analyze this information to determine what will be feasible.

2. Design
Create a design specification. This should answer the question, "How will we build this?".

3. Implement (or Code)
Building the software using a programming language by the development team.

4. Testing & Integration
Release the application to a testing environment, test all areas of the application to spot any defects, bugs, or other problems.

5. Deployment
All defects have been fixed and the software has been built to the agreed-upon goals and specifications.

6. Maintenance
Updating and supporting the software after it has been delivered to the market.

SDLC Phases

3 | SlideSalad.com

slidesalad

2 Unit Testing (UT) – Line by Line Testing

Library For Testing - unittest, pytest (optional)
Syntax for Testing - self.assertEqual(a, self.calc.add(b, c), "Error Message")
- 'a' is the expected output

- 'b' and 'c' are parameters to add function
- 'Error message' is printed if the calculated value is not equal to the expected value.

Invoking Unittest - unittest.main()

Full Code is in GitHub, python calc.py or pytest calc.py works for testing.

○ **2.1 PyUnit - Passed**

Ran 4 tests in 0.003s
OK

```
✓ Tests passed: 4 of 4 tests – 0 ms
C:\Misc\Documents\Career\Languages\miniconda3\python.exe C:/Misc/Documents/Career/Languages/PyCharm/plugins/python-ce/helpers/pycharm/_jb_ur
Testing started at 15:33 ...


Ran 4 tests in 0.003s

OK
Launching unittests with arguments python -m unittest C:\Misc\Documents\Career\Sem3\ASE\GitActions, PyLint, PyUnit\calc.py in C:\Misc\Docume


Process finished with exit code 0
```

○ **2.2 PyUnit - Failed**

Ran 4 tests in 0.015s
FAILED (failures = 1)
30 != 21 (expected = 21, calculated = 30)

```
⊗ Tests failed: 1, passed: 3 of 4 tests – 13 ms
C:\Misc\Documents\Career\Languages\miniconda3\python.exe C:/Misc/Documents/Career/Languages/PyCharm/plugins/python-ce/helpers/pycharm/_jb_ur
Testing started at 15:31 ...
Launching unittests with arguments python -m unittest C:\Misc\Documents\Career\Sem3\ASE\GitActions, PyLint, PyUnit\calc.py in C:\Misc\Docume



Ran 4 tests in 0.015s

FAILED (failures=1)

Multiplication is wrong
30 != 21

Expected :21
Actual   :30
<Click to see difference>

Traceback (most recent call last):
  File "C:\Misc\Documents\Career\Sem3\ASE\GitActions, PyLint, PyUnit\calc.py", line 94, in test_multiply
    self.assertEqual(21, self.calc.multiply(5, 6), "Multiplication is wrong")
AssertionError: 21 != 30 : Multiplication is wrong


Process finished with exit code 1
```

○ **2.3 PyLint - Static Code Analysis Tool**

- Can detect syntax errors, code style violations and potential bugs.
- Run **pylint calc.py** in cmd to get code analysis.
- Gives suggestions on how to improve the code and gives a score out of 10 based on how neat the code is written.
- We can use suggestions by PyLint with ChatGPT to improve our code further.

PyLint - 10/10 Score

```
C:\Windows\System32\cmd.exe - conda  activate tools

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Misc\Documents\Career\Sem3\ASE\GitActions, PyLint, PyUnit>conda activate tools

(tools) C:\Misc\Documents\Career\Sem3\ASE\GitActions, PyLint, PyUnit>pylint calc.py


-----------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)


(tools) C:\Misc\Documents\Career\Sem3\ASE\GitActions, PyLint, PyUnit>
```

PyLint - Below 10 score and suggestions on how to improve

```
C:\Windows\System32\cmd.exe - conda  activate tools

(tools) C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>pylint calc.py
************* Module calc
calc.py:70:0: C0115: Missing class docstring (missing-class-docstring)


-----------------------------------------------------------------
Your code has been rated at 9.60/10 (previous run: 10.00/10, -0.40)


(tools) C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>
```

3 System Testing (ST) - Functionality Testing

ST is also known as blackbox testing and is done by system testers to test the functionality of the product.

○   **3.1 UiPath RPA - Test Script Automation**

● Open UiPath Studio.
● Under New Project, Select Process.
● Launch Calculator App.
● Click on App/Web recorder.
● Hover over the calculator app till a green boundary comes over the entire app.
● Perform a calculation like 7 + 8 = 15.
● Save the recording.
● Under Debug File, Select Run File.
● UiPath will automatically perform the operation that was recorded earlier without human input.
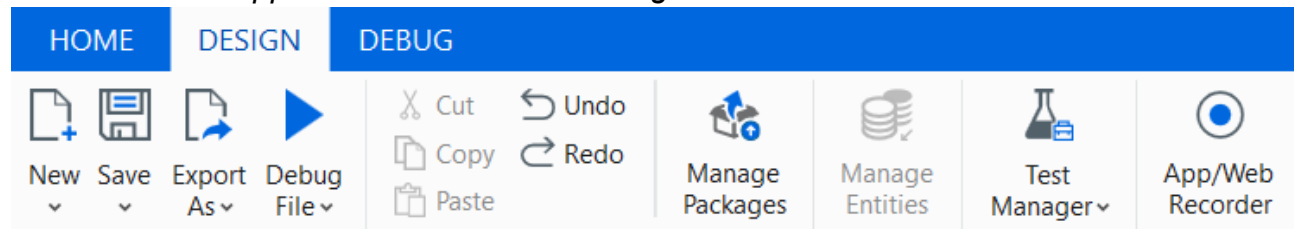● Save the project.

Below is a snippet on how to create a new process.

Our focus is on App/Web Recorder and Debug File as shown below.

4 Version Control, Git and GitHub

● Keeping track of all changes to a given file so that we can go back to previous versions if the current file is not desirable.
● Git is one of the most popular version control systems, GitHub is git in the cloud.

○   **Git Commands**

● **git init -** initialise repository (a directory where all files and their changes are saved).
● **git add -** add files to the staging area (in between local and git repository).

- **git commit** - save the files and changes to the git repository (from the staging area).
- **git status** - display current status of repository and staging area.
- **git clone** - clone a repository locally.
- **git push** - push files from git to github.
- **git pull** - pull files from github to git (local system).

```
C:\Windows\System32\cmd.exe                                          —    □    ✕

Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>git init
Initialized empty Git repository in C:/Misc/Documents/Career/Sem3/ASE/Project2 - DevOps/.git/

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>git add *

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>git commit -m o
[master (root-commit) 4cea94f] o
 6 files changed, 113 insertions(+)
 create mode 100644 Devops-projectreport.txt
 create mode 100644 GitActions-Status.png
 create mode 100644 PyLint.png
 create mode 100644 PyUnit-Failed.png
 create mode 100644 PyUnit-Passed.png
 create mode 100644 calc.py

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>
```

Git to GitHub
Now, in GitHub the pushed files will be reflected.

```
C:\Windows\System32\cmd.exe                                          —    □    ✕

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>git remote add origin https://github.com/AKxy4321/Project-2---DevOps.git

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>git branch -M main

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>git push -u origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 224.52 KiB | 11.23 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AKxy4321/Project-2---DevOps.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps>
```

5 CI/CB/CT/CD and GitActions

CI - Continuous Integration
Regularly integrating code changes into a shared repository.

CB - Continuous Build

Automated compilation of a software project's source code.

CT - Continuous Testing
Automated testing of code changes as they are integrated and deployed.

CD - Continuous Deployment
Automatic deployment of every code change that passes automated tests and quality checks directly to production.

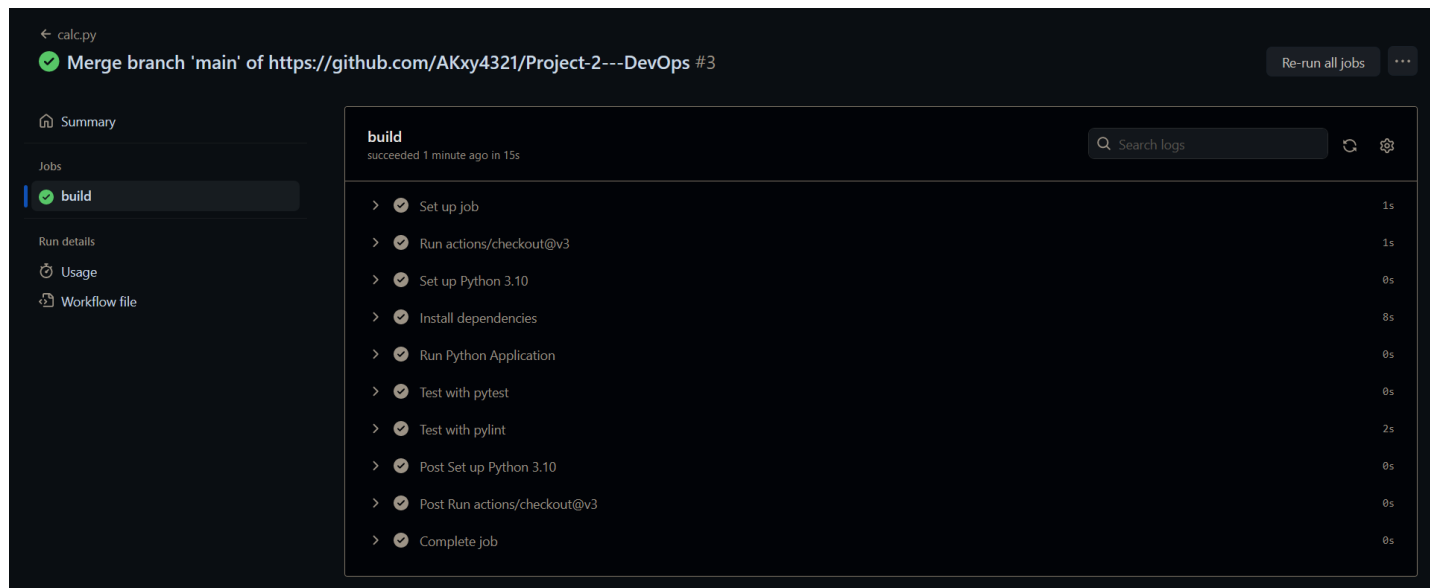Process of going through CI, CB, CT, CD is called the DevOps pipeline.

○   **GitActions**

- Go to your repository in GitHub.
- Go to Actions Tab.
- Select Configure Python application option.
- Edit the yaml file (python calc.py, pytest calc.py and pylint calc.py - optional) and commit the changes.
- Pull the repository to your local machine with git pull.
- Add a dummy text file or make some changes to the code to make it different from the GitHub Repository.
- Push the repository from git to github.
- In the yaml file, the commands configured are run automatically and we can check their status in the Actions Tab.

GitActions Summary

Actions taken by GitActions shown below.

6 Docker and DockerHub

- **Docker** is a tool that allows applications to be run anywhere using containers.
- **Docker Image** contains VM, OS, Software and any other additional libraries required to make the application run.
- **Docker container** is a running instance of a docker image.
- **Dockerfile** is a plain text configuration file used to build a Docker image.
  It contains instructions on how to create the docker image.
- **DockerHub** is docker in the cloud.

Docker Commands

- *docker build -t username/imagename:tag .* - Build docker image
- *docker run username/imagename:tag* - Run container
- *docker push username/imagename:tag* - Push to DockerHub

Steps
- Go to the desired directory, Alt+D and type cmd.
- In CMD, run **echo > Dockerfile**.
- Go to the Dockerfile and write the configurations.
- Go back to CMD, and run **docker build -t akxy4321/devops:devops .** .
- Push to DockerHub with **docker push akxy4321/devops:devops**.
- Run the container with **docker run -p 5000:5000 akxy4321/devops:devops**.
- Go to any browser and type **localhost:5000**.
- Hello World should be printed.

```
C:\Windows\System32\cmd.exe                                                              —    □    ✕

Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps\Docker>docker build -t akxy4321/devops:devops .
[+] Building 14.2s (10/10) FINISHED                                                      docker:default
 => [internal] load build definition from Dockerfile                                               0.0s
 => => transferring dockerfile: 194B                                                               0.0s
 => [internal] load .dockerignore                                                                  0.0s
 => => transferring context: 77B                                                                   0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim                               14.0s
 => [auth] library/python:pull token for registry-1.docker.io                                      0.0s
 => [1/4] FROM docker.io/library/python:3.10-slim@sha256:7b03387c44da8695400590becc6430d8a811b6e97838a740e56f5844  0.0s
 => [internal] load build context                                                                  0.0s
 => => transferring context: 102B                                                                  0.0s
 => CACHED [2/4] WORKDIR /app                                                                       0.0s
 => CACHED [3/4] RUN pip install flask==2.3                                                         0.0s
 => [4/4] COPY . /app                                                                               0.0s
 => exporting to image                                                                             0.0s
 => => exporting layers                                                                            0.0s
 => => writing image sha256:0f3b6bcd411eb4a994c15cc63c1c4f770bee5c6e4719dde5266b7089d6a23f19        0.0s
 => => naming to docker.io/akxy4321/devops:devops                                                  0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps\Docker>docker push akxy4321/devops:devops
The push refers to repository [docker.io/akxy4321/devops]
ee0fd56583ba: Pushed
efcb0c04aecf: Layer already exists
704d24fd22d4: Layer already exists
711b7097618f: Layer already exists
ec68ab009408: Layer already exists
82c3ec988e2a: Layer already exists
e76afd493130: Layer already exists
cb4596cc1454: Layer already exists
devops: digest: sha256:bb5c5409e2a5ce8335ccbeebd6ab10af2d833cff0f2da1405ce6790c24d02aa1 size: 1994

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps\Docker>
```

Docker build and docker push demonstrated above.

```
C:\Windows\System32\cmd.exe - docker run -p 5000:5000 akxy4321/devops:devops              —    □    ✕

C:\Misc\Documents\Career\Sem3\ASE\Project2 - DevOps\Docker>docker run -p 5000:5000 akxy4321/devops:devops
 * Serving Flask app 'app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [31/Oct/2023 13:21:08] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [31/Oct/2023 13:21:08] "GET /favicon.ico HTTP/1.1" 404 -
```

Docker run demonstrated above.
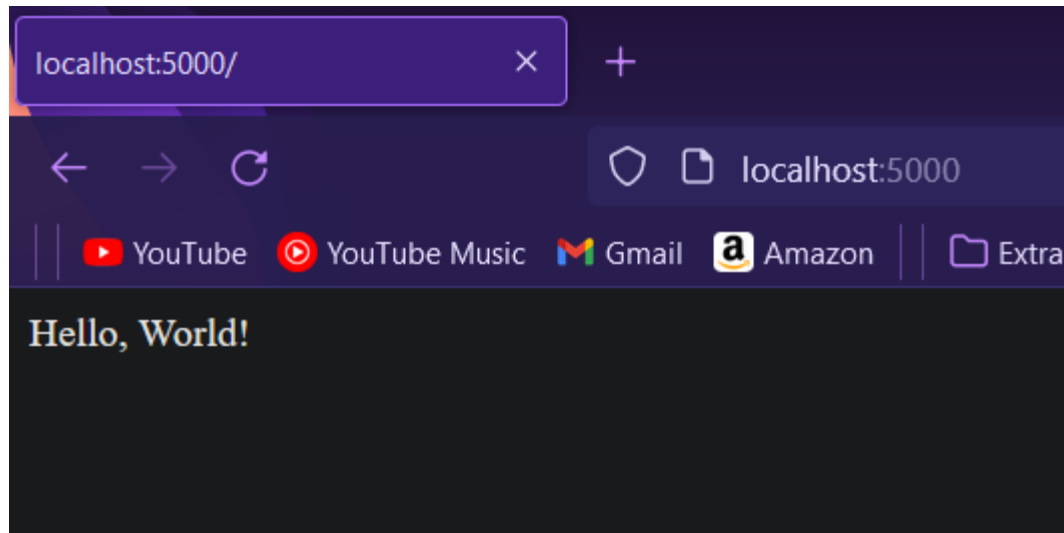
Expected output when localhost:5000 is typed in any browser after docker run command.

7 Conclusions

Through this project I have learnt about the following -
- **SDLC** and its different phases.
- **PyLint -** Static Code Analysis.
- **PyUnit -** Unit Testing.
- **UiPath RPA -** Test Case Automation.
- **Git/GitHub -** Version Control System.
- **GitActions -** CI/CB/CT/CD.
- **Docker/DockerHub -** Deployment of containers.