DevOps Diploma in IT Year 3	Week 7 2 hours
Week 7 and 8 Exercises	

Part A.1 – Preparation

For the ease of understanding and accountability, the following shall be used for **group** work.

- **Repository name:**
  DevOps_Oct2024_Team<number>_Prac<Number>
- **Language of choice:** Python (file extension .py)
- **Test tool of choice:** Pytest / Python Behave / Robotframework
- **Extra Collaborator to add:** erpv-np

Please note that there may be a team shared repository so all members of the group would need to be added as collaborators.

All created repositories are defaulted as public repositories unless specifically stated.

Tools Installation (Completed in Previous Labs)

Ensure that the following are installed

1. Latest or at least a working Python installation
   a. https://www.python.org/downloads/
2. Pytest for python
   a. pip install pytest
   b. pip install pytest-cov
3. Python-behave for python
   a. pip install behave
4. Allure for Python
   a. pip install allure-behave
5. Selenium library for python
   a. pip install selenium
6. Robot framework
   a. pip install robotframework
   b. pip install robotframework-mqttlibrary
   c. pip install robotframework-seleniumlibrary
   d. pip install robotframework-requests
7. Visual Studio Code
   a. https://code.visualstudio.com/Download

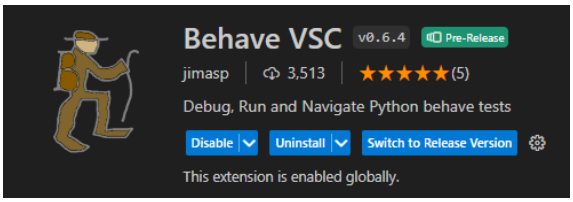
Commands related to tools installations

1. Note that you may need to use pip3 instead of pip for some setups of python3.
2. You may use ***pip list*** to check the list of libraries installed for python

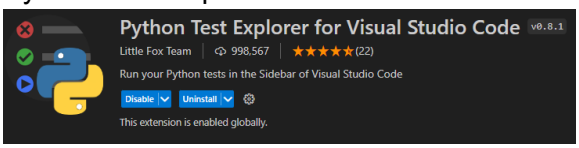
Plugin / Extensions installation

The following extensions are suggested for VSCode.

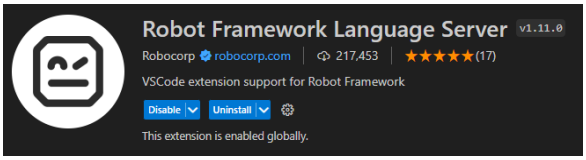
1. Behave VSC
    a. Install the pre-release version as there is no release version



2. Python Test Explorer for Visual Studio Code



3. Robot Framework Language Server



4. Related browser driver of choice for selenium
    a. https://www.selenium.dev/ecosystem/
    b. insert the downloaded driver into the following directory to enable python to detect selenium browser driver
        i. <python installation>\ Scripts\

Git Account Registration (Completed in Previous Labs)

Register for a Git account if you have not done so.

- https://www.github.com
- Please note that this is a personalized account that would be used potentially as a profile during employment.
- **Avoid using unnecessary names** that do not officially associate the account to you as a potential interview candidate. For example, MyHimePrecious2020, IForYou2020 or s01261313 etc.
- Refer to https://www.linkedin.com/learning/craft-a-great-github-profile/create-a-great-github-profile?autoplay=true&u=42538748

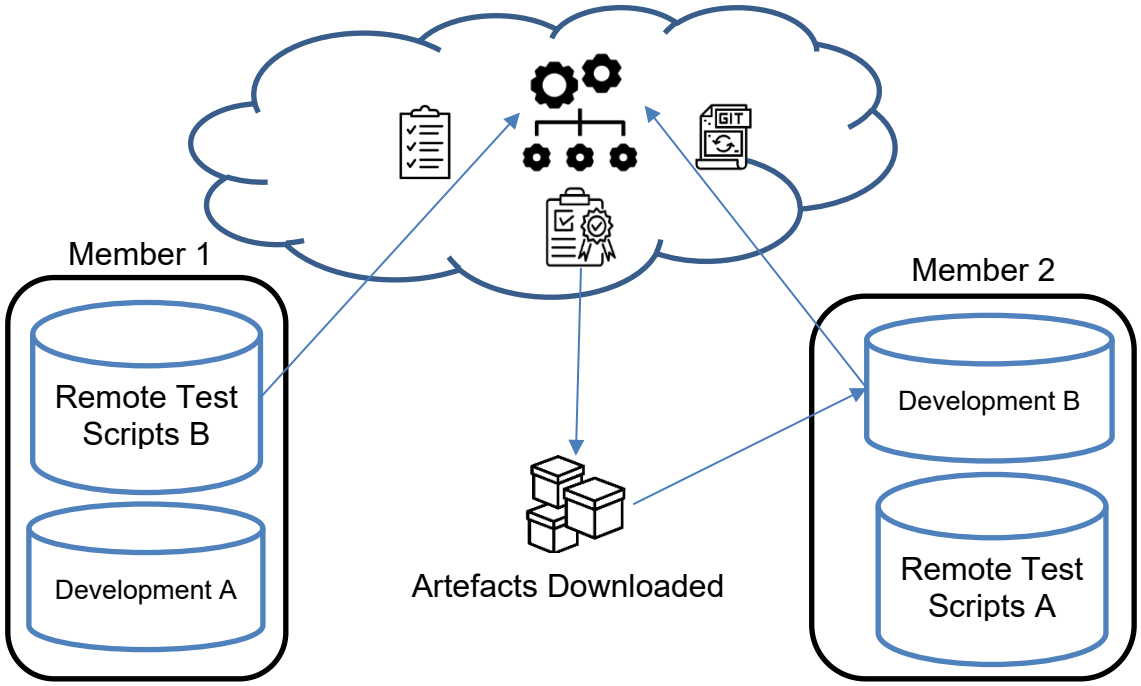
Peer Simulation of testing via remote repositories

Remote testing scripts would be simulated in this activity.

Form a group of at least 2 members or more.

Each person is to prepare

1. A test script for the other party in Git repository and an access token specifically created for other party to gain access to the test script Git repository.
2. Another Git repository demonstrates a full CICD with test results derived from a remote test script.



1. **Remote Test Script Preparation**
   a) Create a Git repository to be shared with the team for them to access.
   b) Create a BDD-based test cases for the following:

   A scientific calculator that could do calculations based on user input.
   - Addition
   - Subtraction
   - Multiplication
   - Division
   - Basic cosine
   - Basic sine
   - Basic tangent

   Note:
   You may refer to this example for BDD based test scripts with Python-Behave: https://semaphoreci.com/community/tutorials/getting-started-with-behavior-testing-in-python-with-behave

2. **Code Development Preparation**
   a) Create a Git repository for code development
   b) Prepare a simple CI with test using the Activity on YAML automation.
   c) Begin coding away ONLY after the CI YAML is created and verified working.

   You may refer to the following as starter codes
   https://www.digitalocean.com/community/tutorials/how-to-make-a-calculator-program-in-python-3#step-1-prompt-users-for-input

   Modifications are needed to the source to refactor and make good the code.

Activity #1 : YAML automation for CI

A YAML automated CI would need to include the following:

- Setup of simple OS of choice. i.e. Linux
- Setup of environment for test
- Setup tools necessary for testing
- Extract necessary repositories. i.e. Source code and Test scripts
- Setup commands for test
- Setup output of test results
- Setup issues found if necessary

You may refer to:

- Github Actions and Creating Workflows
  - https://docs.github.com/en/actions/about-github-actions/understanding-github-actions

- Github-hosted Runners
  - https://docs.github.com/en/actions/using-github-hosted-runners/using-github-hosted-runners/about-github-hosted-runners

- Sample Github Workflows
  - https://github.com/actions/starter-workflows/tree/main/ci

The following are suggested Github Actions for your considerations

- Create an issue if error occurs
  - https://github.com/dacbd/create-issue-action

- Publish results to github actions (Requires the use of XMLs as input to produce result output)
  - https://github.com/EnricoMi/publish-unit-test-result-action

Listing 1 is a sample listing of ci.yaml that is written to do BDD test for each new commit. Please analyze the provided .yaml file and complete Step 5 – 6.

ci.yaml

```yaml
name: CI Test (Behave) Pipeline

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  behave-tests:
    runs-on: ubuntu-latest  # Use Linux OS

    permissions:
      issues: write
      checks: write
      pull-requests: write

    steps:
      # Step 1: Checkout the repository
      - name: Checkout repository
        uses: actions/checkout@v4  #v3
      - name: print content of repo
        run: |
          ls

      # Step 2: Setup Python environment
      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.9'  # Specify the Python version

      # Step 3: Install dependencies
      - name: Install Dependencies
        run: |
          python -m pip install --upgrade pip
          pip install behave

      # Step 4: Run Tests from the features folder
      - name: Run Tests
        run: |
          mkdir -p test-results  # Ensure the directory exists
          behave features/ --junit --junit-directory=test-results # Specify the features folder and save results
          ls test-results/

      # Step 5a: Upload Test results
      - name: Upload Test Results
      # Step 5b: Publish Test results
      - name: Publish Test Results
      # Step 6a: Create Issue
      - name: Create an issue
      # Step 6b: Annotate Test Failures (optional)
      - name: Annotate Test Failures
```

Listing 1 – Sample YAML file

Activity #2: YAML automation for CICD

Delivery

A simple YAML automated CICD would need to include the following.

- Setup of simple OS of choice. i.e. Linux
- Setup of environment for test
- Setup tools necessary for testing
- Extract necessary repositories. i.e. Source code and Test scripts
- Setup commands for test
- Setup output of test results
- Setup issues found if necessary
- Upload targeted files to a determined location in cloud
- Download as zip file and store in current action run

Deployment

A simple YAML automated CICD would need to include the following.

- Setup of simple OS of choice. i.e. Linux
- Setup of environment for test
- Setup tools necessary for testing
- Extract necessary repositories. i.e. Source code and Test scripts
- Setup commands for test
- Setup output of test results
- Setup issues found if necessary
- Upload targeted files to a determined location in cloud
- Download as zip file and store in current action run
- Create instance in release location
- Upload and create associated files for client download

NOTE:

The team may wish to explore and add in a monitoring action where needed.

The following are suggested Github Actions for your considerations:

- Create an issue if error occurs
  - https://github.com/dacbd/create-issue-action
- Publish results to Github Actions (Requires the use of XMLs as input to produce result output)
  - https://github.com/EnricoMi/publish-unit-test-result-action
- Uploading of artifact created to temp (Requires a pre-determined folder to upload to temp space)
  - https://github.com/actions/upload-artifact
- Downloading of artifact created to temp (Requires artefacts to be uploaded to temp space)
  - https://github.com/actions/upload-artifact
- Zip artefacts to a zip file for ease of release
  - https://github.com/TheDoctor0/zip-release
- Release of artefacts to Git Release section
  - https://github.com/actions/upload-release-asset
  - Note that it is no longer maintained but usable.
  - May want to consider finding alternatives where possible.

NOTE:
The team may wish to explore and add in a monitoring action where needed.