Labo 00 – Infrastructure (Git, Docker, CI/CD)

Nom: Simon-Olivier Vaillancourt

Code permanent: VAIS80330101

Question 1

Pytest utilise plusieurs moyens pour signaler les erreurs lors des tests. Premièrement, pour chaque test dans un fichier, Pytest imprime dans la console soit un '.' vert pour représenter un succès ou un 'F' rouge pour représenter un échec. Deuxièmement, Pytest imprime chaque test échoué dans la console avec l'instruction problématique, le numéro de ligne associé ainsi que le type d'erreur levée. Troisièmement, à la fin de son rapport, Pytest indique le nombre de tests qui ont réussi ainsi que le nombre de tests qui ont échoué. Ces trois indicateurs permettent aux développeurs de repérer rapidement et précisément les erreurs afin de les corriger aisément.

Exemple de sortie

(labo0) PS G:\ets\log430-a25-labo0\src> python -m pytest
============================= test session starts
============================= platform win32 -- Python 3.10.4, pytest-8.4.2, pluggy-1.6.0
rootdir: G:\ets\log430-a25-labo0\src collected 6 items

tests\test_calculator.py .....F [100%]

================================= FAILURES
=================================
_____ test_error
_____

```
def test_error():
    my_calculator = Calculator()
```

```
    assert my_calculator.division(10, 2) == 3
```

E assert 5.0 == 3 E + where 5.0 = division(10, 2) E + where division = <calculator.Calculator object at 0x000001A487A89DE0>.division

tests\test_calculator.py:31: AssertionError
=========================

========== short test summary info
==========
========== FAILED tests/test_calculator.py::test_error - assert 5.0 == 3
==========
========== 1 failed, 5 passed in 0.12s
==========
==========

Question 2

Set up

Pendant le set up, Github créer la machine virtuelle qui exécutera l'action avec l'OS spécifié, Ubuntu dans ce cas. Ensuite, Github prépare l'environnement de la machine virtuelle pour exécuter les étapes et télécharge les répertoire des actions utilisées dans celles-ci. Dans ce cas, checkout@v3 et setup-python@v4.

Exemple de sortie

Current runner version: '2.328.0' Runner Image Provisioner Operating System Runner Image GITHUB_TOKEN Permissions Secret source: Actions Prepare workflow directory Prepare all required actions Getting action download info Download action repository 'actions/checkout@v3' (SHA:f43a0e5ff2bd294095638e18286ca9a3d1956744) Download action repository 'actions/setup-python@v4' (SHA:7f4fc3e22c37d6ff65e88745f38bd3157c663f7c) Complete job name: build

Checkout

Pendant l'étape checkout, Github s'occupe d'installer le répertoire sur la machine virtuelle. Précisément, Github clone la branche actuelle (main par exemple) et met en place l'authentification pour récupérer les secrets nécessaire pour exécuter l'action. Dans notre cas, il n'y a pas de secret à récupérer.

Exemple de sortie

Run actions/checkout@v3 Syncing repository: s0punk/log430-a25-labo0 Getting Git version info Temporarily overriding HOME='/home/runner/work/_temp/3b66bd9c-dc1e-46f1-a5bb-c3b616693a49' before making global git config changes Adding repository directory to the temporary git global config as a safe directory /usr/bin/git config --global --add safe.directory /home/runner/work/log430-a25-labo0/log430-a25-labo0 Deleting the contents of '/home/runner/work/log430-a25-labo0/log430-a25-labo0' Initializing the repository Disabling automatic garbage collection Setting up auth Fetching the repository Determining the checkout info Checking out the ref /usr/bin/git log -1 --format='%H' '7e31d41c565f2c635a04100ba32a3974697ce463'

Question 3 & 4

Annulées