

Rapport LOG430 - Labo 0 : Infrastructure (Git, Docker, CI/CD)



ÉCOLE DE
TECHNOLOGIE
SUPÉRIEURE

Université du Québec

Rédacteur : Damien Sudre - SUDD91370200 - LOG430

Ecriture des Tests

On a **5 fonctions** à tester dans la classe `Calculator` de l'application `calculator.py` :

- `get_hello_message`
- `addition`
- `subtraction`
- `multiplication`
- `division`

Les tests sont rédigés dans le fichier `src/tests/test_calculator.py`. Il existe déjà le cas de test pour la fonction `get_hello_message`, nous nous concentrerons alors sur les 4 autres fonctions.

On garde a l'esprit que le but du labo0 n'est pas de rediger des tests pour une application complexe.

On a alors les 4 cas de tests suivants :

```
12
13  def test_addition():
14      calc = Calculator()
15      assert calc.addition(5, 3) == 8
16
17  def test_subtraction():
18      calc = Calculator()
19      assert calc.subtraction(205, 115) == 90
20
21  def test_multiplication():
22      calc = Calculator()
23      assert calc.multiplication(35, 43) == 1505
24
25  def test_division():
26      calc = Calculator()
27      assert calc.division(5, 2) == 2.5
```

Modification due a des erreurs d'imports

Avec le code et la structure de projet fournis, `pytest` ne parvenait pas à résoudre l'import `from calculator import Calculator`. En effet, l'exécution de `pytest` donnait systématiquement l'erreur suivante, et ce quelque soit le répertoire depuis lequel j'appelais `pytest`.

```
(labo0) PS C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0\src> python --m pytest
=====
test session starts =====
platform win32 -- Python 3.12.3, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0\src
collected 0 items / 1 error

=====
ERRORS =====
-- ERROR collecting tests/test_calculator.py --
ImportError while importing test module 'C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0\src\tests\test_calculator.py'.
Hint: make sure your test modules/packages have valid Python names.
Traceback:
C:\Users\exter\AppData\Local\Programs\Python\Python312\Lib\importlib\__init__.py:90: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
tests\test_calculator.py:7: in <module>
    from testcalculator import Calculator
E   ModuleNotFoundError: No module named 'testcalculator'
=====
short test summary info =====
ERROR tests/test_calculator.py
!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!
===== 1 error in 0.07s =====
(labo0) PS C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0\src>
```

En souhaitant modifier le moins de code que possible, j'ai alors choisi de rajouter la section suivante dans le fichier `src/tests/test_calculator.py` afin de résoudre l'erreur en ajoutant le chemin absolu au `sys.path`:

```
3  SPDX - License - Identifier: LGPL - 3.0 - or -later
4  Auteurs : Gabriel C. Ullmann, Fabio Petrillo, 2025
5  """
6
7  import os
8  import sys
9
10 sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(__file__), "src")))
11
12 from calculator import Calculator
13
14 def test_app():
15     calc = Calculator()
16     assert calc.get_hello_message() == "== Calculatrice v1.0 =="
```

un fichier `__init__.py` a également été ajouté dans `src/tests` pour considérer le module de test.

On obtient alors bien l'exécution des tests :

```
(labo0) PS C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0> p
ytest
=====
test session starts =====
platform win32 -- Python 3.12.3, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0
collected 5 items

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

===== 5 passed in 0.01s =====
(labo0) PS C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0> _
```

La méthode utilisée pour résoudre ce problème n'est pas recommandé en situation réelle, cependant une résolution plus poussée sortirait du cadre du laboratoire.

Question 1 : Consequence de la rédaction d'un test éronné avec `pytest`

On modifie légèrement un cas de test pour forcer une erreur :

```
25
26     def test_multiplication():
27         calc = Calculator()
28         assert calc.multiplication(35, 43) == 1506
29
```

ici, on a modifier la valeur attendu à 1506 (au lieu de 1505)

On obtient alors la sortie suivante en lancant `pytest` :

```
(labo0) PS C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0\src> pytest
=====
test session starts =====
platform win32 -- Python 3.12.3, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0\src
collected 5 items

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

===== FAILURES =====
----- test_multiplication -----
def test_multiplication():
    calc = Calculator()
>     assert calc.multiplication(35, 43) == 1506
E     assert 1505 == 1506
E     +  where 1505 = multiplication(35, 43)
E     +  where multiplication = <calculator.Calculator object at 0x0000020C0D5D0110>.multiplication

tests\test_calculator.py:28: AssertionError
===== short test summary info =====
FAILED tests\test_calculator.py::test_multiplication - assert 1505 == 1506
===== 1 failed, 4 passed in 0.04s =====
(labo0) PS C:\Users\exter\Desktop\Etudes\Projet ETS\7 - Automne 2025\LOG430\Labo 0\log430-a25-labo0\src>
```

On constate alors que l'on obtiens un rapport d'erreur qui nous indique quelle assertion de quel test est concerné par l'échec. On a aussi un résumé en première ligne indiquant un "F" lorsqu'un test échoue (pratique pour figurer où se trouve le cas de test dans le fichier).

Pipeline CI + Versionnage du projet

Question 2 : Que font les étapes `Setup` et `Checkout` de la CI ainsi paramétrée :

On remarque que la question doit faire references aux étapes suivante :

```
4
5   jobs:
6     build:
7       runs-on: ubuntu-latest
8       environment: Labo0
9
10    steps:
11      - name: Checkout dépôt
12        uses: actions/checkout@v3
13
14      - name: Installer Python
15        run: python -m pip install -r requirements.txt
```

1

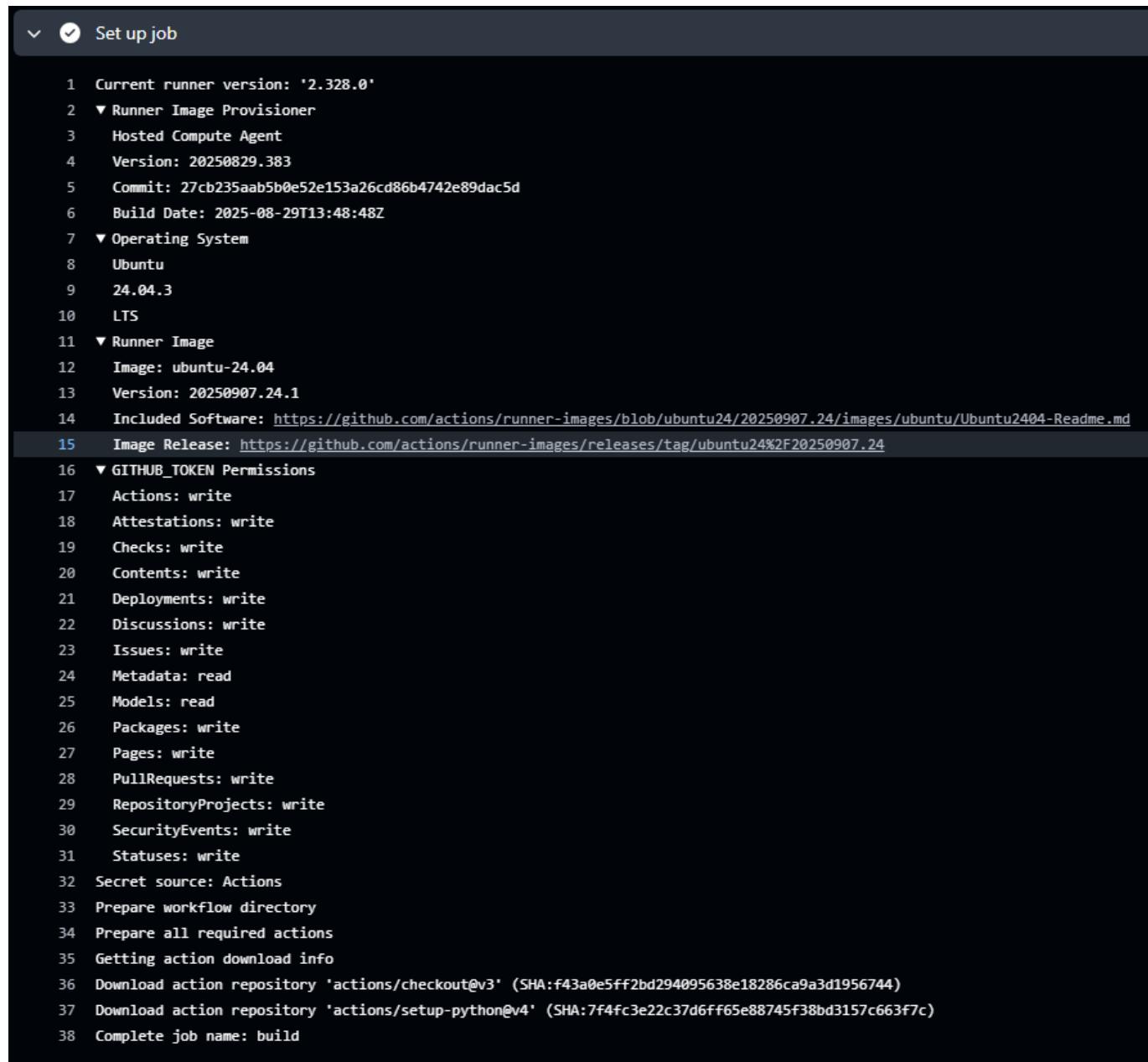
2

A propos du Setup :

runs-on: ubuntu-latest initie la création d'une machine virtuel utilisant une distribution recente d'Ubuntu.

environment : Labo0 permet de lier le runner du job **build** (celui qui utilise la VM Ubuntu) à l'environnement GitHub 'Labo0' pour lui attribuer des autorisations ou autre paramètres.

On a l'exécution suivante sur GitHub Actions pour le Setup :



The screenshot shows a terminal window with the title 'Set up job'. The log output is as follows:

```
1 Current runner version: '2.328.0'
2 ▼ Runner Image Provisioner
3   Hosted Compute Agent
4     Version: 20250829.383
5     Commit: 27cb235aab5b0e52e153a26cd86b4742e89dac5d
6     Build Date: 2025-08-29T13:48:48Z
7   ▼ Operating System
8     Ubuntu
9     24.04.3
10    LTS
11   ▼ Runner Image
12     Image: ubuntu-24.04
13     Version: 20250907.24.1
14     Included Software: https://github.com/actions/runner-images/blob/ubuntu24/20250907.24/images/ubuntu/Ubuntu2404-Readme.md
15     Image Release: https://github.com/actions/runner-images/releases/tag/ubuntu24%2F20250907.24
16   ▼ GITHUB_TOKEN Permissions
17     Actions: write
18     Attestations: write
19     Checks: write
20     Contents: write
21     Deployments: write
22     Discussions: write
23     Issues: write
24     Metadata: read
25     Models: read
26     Packages: write
27     Pages: write
28     PullRequests: write
29     RepositoryProjects: write
30     SecurityEvents: write
31     Statuses: write
32     Secret source: Actions
33     Prepare workflow directory
34     Prepare all required actions
35     Getting action download info
36     Download action repository 'actions/checkout@v3' (SHA:f43a0e5ff2bd294095638e18286ca9a3d1956744)
37     Download action repository 'actions/setup-python@v4' (SHA:7f4fc3e22c37d6ff65e88745f38bd3157c663f7c)
38     Complete job name: build
```

A propos du Checkout :

Cette première étape utilise une action officiel de github (avec **uses: actions/checkout@v3**) qui a pour rôle de cloner (ou plus précisément, d'initialiser) le repo de code sur la VM déployée (sinon il n'y a qu'une VM vide). C'est une étape essentielle pour ensuite pouvoir lancer des tests sur notre code (puisque pour tester le code, on a besoin du dit code).

On a la sortie suivante sur GitHub Actions :

```

✓ Checkout dépôt
1 ► Run actions/checkout@3
14 Syncing repository: DamienSud/LOCAL-log430-a25-labo0
15 ▼ Getting Git version info
16 Working directory is '/home/runn.../LOCAL-log430-a25-labo0/LOCAL-log430-a25-labo0'
17 /usr/bin/git version
18 git version 2.51.0
19 Temporarily overriding HOME='/home/runn.../_temp/d17b63e0-ae4c-40be-8427-9a46c647bdc0' before making global git config changes
20 Adding repository directory to the temporary git global config as a safe directory
21 /usr/bin/git config --global --add safe.directory '/home/runn.../LOCAL-log430-a25-labo0/LOCAL-log430-a25-labo0'
22 Deleting the contents of '/home/runn.../LOCAL-log430-a25-labo0/LOCAL-log430-a25-labo0'
23 ► Initializing the repository
39 ► Disabling automatic garbage collection
41 ► Setting up auth
47 ► Fetching the repository
133 ► Determining the checkout info
134 ► Checking out the ref
138 /usr/bin/git log -1 --format='%H'
139 'c93a9d8ab4b07722d5f9a401bfe6d13a83e18c00'

> ⓘ Installer Python

```

Deploiement du code manuellement sur une VM

En utilisant docker, on peut **build** & **run** sur notre machine virtuelle (fournis par l'ETS) :

Build the Image :

```

(lab0) log430@log430-etudiante-100:~/labo0/LOCAL-log430-a25-labo0$ ls
Dockerfile LICENSE README.md docker-compose.yml docs rapport requirements.txt src
(lab0) log430@log430-etudiante-100:~/labo0/LOCAL-log430-a25-labo0$ docker build -t labo0-calculator .
[+] Building 9.5s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 128B
=> [internal] load metadata for docker.io/library/python:3.11-slim
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/library/python:3.11-slim@sha256:a0939570b38cddeb861b8e75d20b1c8218b21562b18f301171904b54 4.4s
=> => resolve docker.io/library/python:3.11-slim@sha256:a0939570b38cddeb861b8e75d20b1c8218b21562b18f301171904b54 0.0s
=> sha256:11b89692b2085631f6e2407edd8545b033c8e6945837103875d6db484e945b6f 1.29MB / 1.29MB 0.3s
=> sha256:764e05fe66b6768e40fa2a21d5108eceb8f3f8f2c32463d72c109c54dde0d5c1 14.60MB / 14.60MB 0.7s
=> sha256:a0939570b38cddeb861b8e75d20b1c8218b21562b18f301171904b54#4e8cf228 10.37kB / 10.37kB 0.0s
=> sha256:316d89b74c4d467565864be703299878ca97893ed4iae45f6acba5af09d154 1.75kB / 1.75kB 0.0s
=> sha256:c4640ec09886fe463924abb53b1694191eeff91ce3cfea2137e0ed81b6cb88194 5.38kB / 5.38kB 0.0s
=> sha256:ce1261c6d567efa8e3b4576736eeeb474a0a8066df6bb95ca9a6a94a31e219dd3 29.77MB / 29.77MB 1.2s
=> sha256:a4aefcec16c5bdc01af2ad1c5341b420d4179f3b825c0dc866367fb43f0d50ac 250B / 250B 0.0s
=> => extracting sha256:ce1261c6d567efa8e3b4576736eeeb474a0a8066df6bb95ca9a6a94a31e219dd3 1.6s
=> => extracting sha256:11b89692b2085631f6e2407edd8545b033c8e6945837103875d6db484e945b6f 0.1s
=> => extracting sha256:764e05fe66b6768e40fa2a21d5108eceb8f3f8f2c32463d72c109c54dde0d5c1 1.0s
=> => extracting sha256:a4aefcec16c5bdc01af2ad1c5341b420d4179f3b825c0dc866367fb43f0d50ac 0.0s
=> [internal] load build context
=> => transferring context: 2.53kB
=> [2/3] WORKDIR /app
=> [3/3] COPY src/ ./src/
=> exporting to image
=> => exporting layers
=> => writing image sha256:75fb87e31894ce1d9fab210f27alacl6800020d1f6596427b0e423abfa3bce86 0.0s
=> => naming to docker.io/library/lab0-calculator 0.0s

```

Run the Container :

```

(lab0) log430@log430-etudiante-100:~/labo0/LOCAL-log430-a25-labo0$ docker run -it labo0-calculator
== Calculatrice v1.0 ==
Operation : additionner deux valeurs
Saisissez la valeur 1 : 25
Saisissez la valeur 2 : 16
V1 + V2 = 41
Voulez-vous faire une autre addition ? [1 = Oui | 2 = Non] : 2
Au revoir :)
test
(lab0) log430@log430-etudiante-100:~/labo0/LOCAL-log430-a25-labo0$ 

```

Mise en place d'un self hosted Runner pour la CD

Cette étape sert à outrepasser les restrictions de connections à la VM de l'école en hébergeant un self hosted runner sur la VM qui écoute le repo.

installation du runner (etape fournis sous **Settings > Actions > Runners > New self-hosted runner**):

```
(labo0) log430@log430-etudiante-100:~/labo0$ mkdir actions-runner && cd actions-runner
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ curl
curl: try 'curl --help' or 'curl --manual' for more information
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ curl -o actions-runner-linux-x64-2.328.0.t
ar.gz -L https://github.com/actions/runner/releases/download/v2.328.0/actions-runner-linux-x64-2.328.0
.tar.gz
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
          Dload  Upload   Total Spent  Left Speed
0       0     0     0       0      0  --:--:--  --:--:--  --:--:--   0
100  216M  100  216M  0     0  97.4M  0  0:00:02  0:00:02  --:--:--  111M
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ ^C
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ echo '-----'
----- actions-runner-linux-x64-2.328.0.tar.gz" | shasum -a 256 -c
actions-runner-linux-x64-2.328.0.tar.gz: OK
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ tar xzf ./actions-runner-linux-x64-2.328.0
.tar.gz
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ ls
actions-runner-linux-x64-2.328.0.tar.gz  env.sh
bin                                     externals
config.sh                                run-helper.cmd.template
                                         run-helper.sh.template
                                         run.sh
                                         safe_sleep.sh
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ cd ..
(lab0) log430@log430-etudiante-100:~/labo0$ ls
LOCAL-log430-a25-labo0  actions-runner
(lab0) log430@log430-etudiante-100:~/labo0$ cd actions-runner/
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ ls
actions-runner-linux-x64-2.328.0.tar.gz  env.sh
bin                                     externals
config.sh                                run-helper.cmd.template
                                         run-helper.sh.template
                                         run.sh
                                         safe_sleep.sh
(lab0) log430@log430-etudiante-100:~/labo0/actions-runner$ ./config.sh --url https://github.com/DamienSud/LOCAL-log430-a25-labo0 --token -----
```

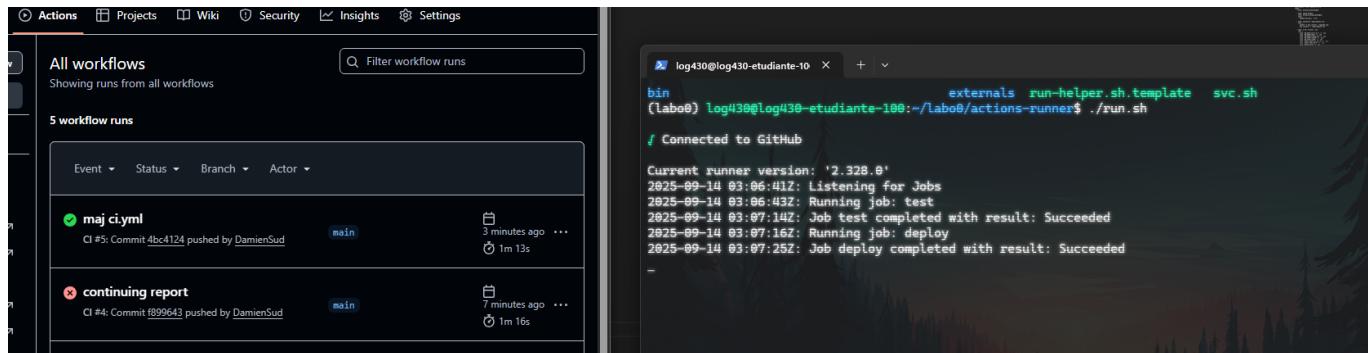


a titre d'information, j'ai decider de nommer mon runner **custom_runner** (si jamais le nom apparait plus loin dans ce rapport).

The screenshot shows two parts of the GitHub interface. On the left, the 'Runners' page lists a single runner named 'custom_runner' which is 'Idle'. On the right, a terminal window titled 'Actions > Runners > New self-hosted runner**' shows the command-line steps to download and extract the runner, followed by a message indicating it is connected to GitHub and listening for jobs.

Le contenu du fichier **ci.yml** a aussi été éditer pour passer les jobs en **runs-on: self-hosted** afin d'utiliser le hosted-runner configurer precedentement.

On voit sur la capture ci dessous qu'avec le runner installé sur la VM et le fichier **ci.yml** adapté, notre pipeline passe avec succès :



The screenshot shows two windows side-by-side. On the left is the GitHub Actions interface under the 'Actions' tab, displaying 'All workflows' and '5 workflow runs'. The first run, 'maj ci.yml', is successful ('green checkmark') and triggered by a commit from 'main'. The second run, 'continuing report', is failing ('red X'). On the right is a terminal window titled 'log430@log430-etudiante-10 ~' showing deployment logs:

```
bin externals run-helper.sh.template svc.sh
(lab0) log430@log430-etudiante-10:~/Labo0/actions-runner$ ./run.sh
Connected to GitHub

Current runner version: '2.328.0'
2025-09-14 03:06:41Z: Listening for Jobs
2025-09-14 03:06:43Z: Running job: test
2025-09-14 03:07:18Z: Job test completed with result: Succeeded
2025-09-14 03:07:18Z: Running job: deploy
2025-09-14 03:07:25Z: Job deploy completed with result: Succeeded
-
```

Question 3 : Comment est fait le déploiement

Une fois l'étape du self-hosted runner réalisé, on utilise le fichier `ci.yml` nouvellement créé ayant le profil suivant :

```
.github/workflows/.github/workflows/cl.yml
1 name: CI
2
3 on:
4   push:
5     branches: [main, master]
6   pull_request:
7
8 jobs:
9   test:
10    runs-on: self-hosted
11    environment: Test Pipeline CI/CD
12    steps:
13      - uses: actions/checkout@v4
14
15      - name: Setup Python
16        uses: actions/setup-python@v4
17        with:
18          python-version: '3.11'
19
20      - name: Installer requirements.txt
21        run: |
22          python -m pip install --upgrade pip
23          pip install -r requirements.txt
24
25      - name: Exécuter les tests
26        working-directory: src
27        run: |
28          python3 -m pytest
29
30 deploy:
31   runs-on: self-hosted
32   needs: test
33   steps:
34     - uses: actions/checkout@v4
35
36     - name: Déploiement local
37       run: |
38         cd $GITHUB_WORKSPACE
39         echo "Déploiement sur la VM en cours..."
40
41         docker compose down || true
42         docker compose up -d --build
```

Principalement, on a séparer la logique d'installation et de Test (orienté CI) dans un job et celle de déploiement dans un autre qui ne se lancerait qu'a la condition de la réussite du premier.

On peut alors constater que le projet est bien déployer dans le repertoire `$GITHUB_WORKSPACE` (dans notre cas, il s'agit du dossier `_work`, il a été défini lors de la création du runner)

```
(labo0) log430@log430-etudiante-100:~/labo0/actions-runner$ cd _work
(labo0) log430@log430-etudiante-100:~/labo0/actions-runner/_work$ ls
LOCAL-log430-a25-labo0 _PipelineMapping _actions _temp _tool
(labo0) log430@log430-etudiante-100:~/labo0/actions-runner/_work$ cd LOCAL-log430-a25-labo0/
(labo0) log430@log430-etudiante-100:~/labo0/actions-runner/_work/LOCAL-log430-a25-labo0$ ls
LOCAL-log430-a25-labo0
(labo0) log430@log430-etudiante-100:~/labo0/actions-runner/_work/LOCAL-log430-a25-labo0$ cd LOCAL-log430-a25-labo0/
(labo0) log430@log430-etudiante-100:~/labo0/actions-runner/_work/LOCAL-log430-a25-labo0/LOCAL-log430-a25-labo0$ ls
Dockerfile LICENSE README.md docker-compose.yml docs rapport requirements.txt src
```

Question 4 : Que fait la Commande top

La commande top permet d'avoir une vue en temps réel des processus qui tournent sur une machine ainsi que des ressources qu'elles utilisent.

```
top - 03:27:16 up 10 days, 5:44, 1 user, load average: 0.00, 0.02, 0.01
Tasks: 129 total, 1 running, 128 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.2 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3622.6 total, 671.1 free, 641.2 used, 2625.2 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 2981.4 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
31404 log430 20 0 15128 7088 5120 S 0.3 0.2 0:04.73 sshd
33670 root 20 0 0 0 0 I 0.3 0.0 0:00.56 kworker/0:2-events
  1 root 20 0 22756 13780 9428 S 0.0 0.4 0:38.64 systemd
  2 root 20 0 0 0 0 S 0.0 0.0 0:00.26 kthreadd
  3 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pool_workqueue_release
  4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-rCU_g
  5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-rCU_p
  6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-slub_
  7 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-netns
  9 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-events_highpri
 12 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-mm_pe
 13 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_kthread
 14 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_rude_kthread
 15 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace_kthread
 16 root 20 0 0 0 0 S 0.0 0.0 0:03.45 ksoftirqd/0
 17 root 20 0 0 0 0 I 0.0 0.0 1:32.94 rcu_preempt
 18 root rt 0 0 0 0 S 0.0 0.0 0:07.97 migration/0
 19 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
 20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
 21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1
```