Compte Rendu TP CI/CD

Exercice 1 : Configuration de l'environnement local

Installation des différents outils nécessaires pour la CI/CD.

- Docker
- gitlab-runner
- hatch

Exercice 2 : Configuration du projet

Nom du projet : my-arithmetic-oscarb

fonction pgcd créée

```
def pgcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a
```

test de la fonction créée

```
def test_valid_pgcd(self):
    self.assertEqual(34, pgcd(40902, 24140))

def test_invalid_pgcd(self):
    self.assertNotEqual(0, pgcd(40902, 24140))
```

Exercice 3 : Création d'un projet GitLab

3.1 Run automatique des tests

Contenu du fichier .gitlab-ci.yml pour le run automatique des tests :

```
stages:
  - test

image: python:3.13-alpine

# Étape de test
test:
  stage: test
  script:
    - echo "Hello, $GITLAB_USER_LOGIN!"
    - echo "Running Test"
    - apk add gcc python3-dev musl-dev linux-headers
    - pip install --upgrade pip setuptools hatchling
    - pip install hatch
    - hatch shell
    - hatch test
```

3.2 Calcul de la couverture de code et affichage du badge

Contenu du fichier .gitlab-ci.yml mis à jour ajouter le calcul de la couverture de code et affichage du badge :

```
stages:
  - test
  - coverage

image: python:3.13-alpine

# Étape de test
test:
  stage: test
  script:
    - echo "Hello, $GITLAB_USER_LOGIN!"
    - echo "Running Test"
    - apk add gcc python3-dev musl-dev linux-headers
    - pip install --upgrade pip setuptools hatchling
    - pip install hatch
    - hatch shell
    - hatch test

# Étape de couverture
coverage:
  stage: coverage
  script:
    - echo "Collecting Code Coverage"
    - pip install pytest-cov
    - pytest --cov=src.my_arithmetic_oscarb.function --cov-report=xml --cov-report=term-missing
  artifacts:
    expire_in: 1 hour
    reports:
      coverage_report:
        coverage_format: cobertura
        path: coverage.xml
  coverage: '/TOTAL.*? (\d+%)$/'
```

Ligne ajoutée dans le README.md pour afficher le badge de couverture de code :

```
[![coverage report](https://gitlab.univ-lr.fr/oblais/my-arithmetic-oscarb/badges/main/coverage.svg)](https://gitlab.univ-lr.fr/oblais
```

Rendu final du badge :

J'ai également testé d'ajouter une deuxieme fonction à mon projet pour voir si la couverture de code se mettait à jour automatiquement. Le test a été concluant.

Badge coverage non complet:

Exercice 4 : Déploiement automatique

Premierement, le pyproject.toml doit être mis à jour pour ajouter hatch-vcs.

4.1 Mise à jour du fichier pyproject.toml pour ajouter hatch-vcs

Element ajouté/modifié dans le fichier pyproject.toml afin d'ajouter hatch-vcs :

```
requires = ["hatchling", "hatch-vcs"]

[tool.hatch.version]
source = "vcs"

[tool.hatch.version.raw-options]
version_scheme = "no-guess-dev"
local_scheme = "no-local-version"

[tool.hatch.metadata.hooks.vcs]
version = "tag"
```

4.2 Création du build lors de la création d'un nouveau tag

J'ai ensuite créé 2 nouveaux jobs, un pour le cas de la création d'un nouveau tag et l'autre pour le cas d'un push sur develop (partie 4.3).

Contenu du fichier .gitlab-ci.yml mis à jour pour ajouter le build lors de la création d'un nouveau tag :

```
stages:
  - test
  - coverage
  - deploy

image: python:3.13-alpine

# Étape de test
test:
  stage: test
  script:
    - echo "Hello, $GITLAB_USER_LOGIN!"
    - echo "Running Test"
    - apk add gcc python3-dev musl-dev linux-headers git
    - pip install hatch
    - hatch shell
    - hatch test

# Étape de couverture
coverage:
  stage: coverage
  script:
    - echo "Collecting Code Coverage"
    - pip install pytest-cov
    - pytest --cov=src.my_arithmetic_oscarb.function --cov-report=xml --cov-report=term-missing
  artifacts:
    expire_in: 1 hour
    reports:
      coverage_report:
        coverage_format: cobertura
        path: coverage.xml
  coverage: '/TOTAL.*? (\d+%)$/'

# Étape de déploiement (tag)
deploy-tag:
  stage: deploy
  script:
    - apk add git
    - pip install hatch hatch-vcs
    - hatch build
    - echo "my-arithmetic-oscarb deployment on stable servers"
    - echo "Contenu du dossier dist :"
    - ls dist/
  only:
    - tags
  artifacts:
    paths:
      - dist/*.whl
```

4.3 Création du build lors d'un push sur develop

Contenu du fichier .gitlab-ci.yml mis à jour pour ajouter le build lors d'un push sur develop :

```yaml
stages:
  - test
  - coverage
  - deploy

image: python:3.13-alpine

# Étape de test
test:
  stage: test
  script:
    - echo "Hello, $GITLAB_USER_LOGIN!"
    - echo "Running Test"
    - apk add gcc python3-dev musl-dev linux-headers git
    - pip install hatch
    - hatch shell
    - hatch test

# Étape de couverture
coverage:
  stage: coverage
  script:
    - echo "Collecting Code Coverage"
    - pip install pytest-cov
    - pytest --cov=src.my_arithmetic_oscarb.function --cov-report=xml --cov-report=term-missing
  artifacts:
    expire_in: 1 hour
    reports:
      coverage_report:
        coverage_format: cobertura
        path: coverage.xml
  coverage: '/TOTAL.*? (\d+%)$/'

# Étape de déploiement (tag)
deploy-tag:
  stage: deploy
  script:
    - apk add git
    - pip install hatch hatch-vcs
    - hatch build
    - echo "my-arithmetic-oscarb deployment on stable servers"
    - echo "Contenu du dossier dist :"
    - ls dist/
  only:
    - tags
  artifacts:
    paths:
      - dist/*.whl

# Étape de déploiement (develop)
deploy-develop:
  stage: deploy
  script:
    - apk add git
    - pip install hatch hatch-vcs
    - hatch build
    - echo "my-arithmetic-oscarb deployment on develop servers"
    - echo "Contenu du dossier dist :"
    - ls dist/
  rules:
    - if: $CI_COMMIT_BRANCH == "develop"
  artifacts:
    paths:
      - dist/*.whl
```

Exercice 5 : Mirroir GitLab et GitHub

L'opération de mirroring entre GitLab et GitHub a été effectuée avec succès. Le projet est maintenant disponible sur GitHub à l'adresse suivante : Repo my-arithmetic-oscarb sur GitHub (https://github.com/oblais23/my-arithmetic-oscarb/tree/develop)