Rapport de réponse aux questions de création de workflow

Encadrant : Josselyn Colombet

Solène Floydy OKOME NDONG TATY

Guillaume PETTA

Timothé BRUYERE

18 avril 2024

Table des matières

1 Créer un workflow simple qui exécute des tests à chaque push sur la branche principale.

```yaml
name: Python Tests

on:
  push:
    branches:
      - main
jobs:
  test:
    name: Run Tests
    runs-on: windows-latest

    steps:
    - name: Checkout Repository
      uses: actions/checkout@v2

    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: 3.x

   # - name: Run Tests
    #  run: pytest
    - name: Install Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14.x'

    - name: Install JavaScript Dependencies
      run: npm install

    - name: Run JavaScript Tests
      run: npm test
```

Ce workflow contient un job nommé Run test qui exécute des tests JavaScript. Les étapes du job incluent la configuration de l'environnement Node.js, l'installation des dépendances JavaScript à partir de package.json et l'exécution des tests JavaScript avec npm test.

2 Écrire un workflow pour déployer automatiquement une application sur un service cloud lorsque les tests passent sur la branche de production.

2.1 Version Azure.

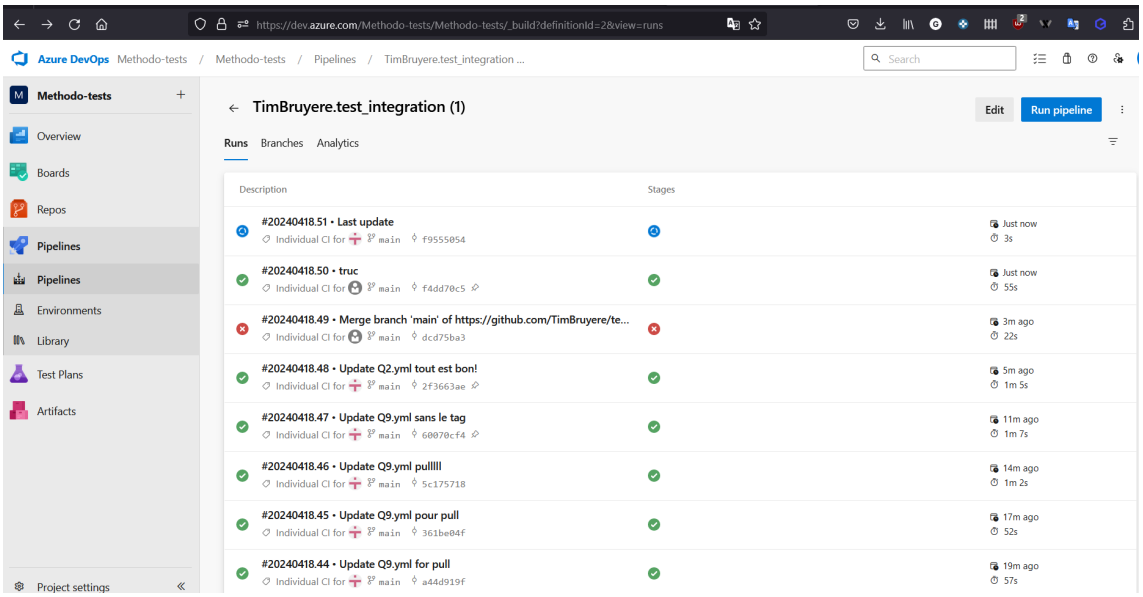

FIGURE 1 – Capture de Azure

```yaml
# Docker
# Build a Docker image
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
- main

resources:
- repo: self

variables:
  tag: '$(Build.BuildId)'

stages:
- stage: Build
  displayName: Build image
  jobs:
  - job: Build
    displayName: Build
    pool:
      vmImage: ubuntu-latest
    steps:
    - script: echo 'Début du déploiement sur Azure DevOps'
      displayName: 'Initialisation du déploiement'
    - task: Docker@2
      displayName: Build an image
      inputs:
        command: build
        dockerfile: '$(Build.SourcesDirectory)/dockerfile'
        tags: |
          $(tag)
```

Version Cloud : Ne fonctionne qu'avec une Vm ubuntu et pas windows.

2.2 Version Docker.

```yaml
name: deploy-docker

on:
  push:
    branches:
      - main  # Adjust branch as needed

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout repository
      uses: actions/checkout@v2

    - name: Build Docker image
      run: |
        docker build -t my-docker-image .

    - name: Log in to Docker Hub
      uses: docker/login-action@v1
      with:
        username: ${{ secrets.DOCKER_USERNAME }}
        password: ${{ secrets.DOCKER_PASSWORD }}

    - name: Push Docker image to Docker Hub
      run: |
        docker tag my-docker-image ${{ secrets.DOCKER_USERNAME }}/my-docker-image:latest
        docker push ${{ secrets.DOCKER_USERNAME }}/my-docker-image:latest
```

Ne fonctionne qu'avec un runs-on sur ubuntu et pas windows.

3 Modifier un workflow existant pour intégrer des notifications Discord sur le statut du déploiement.

```yaml
trigger:
- main

resources:
- repo: self

variables:
  tag: '$(Build.BuildId)'

stages:
- stage: Build
  displayName: Build image
  jobs:
  - job: Build
    displayName: Build
    pool:
      vmImage: ubuntu-latest
    steps:
    - script: echo 'Début du déploiement sur Azure DevOps'
      displayName: 'Initialisation du déploiement'
    - task: Docker@2
      displayName: Build an image
      inputs:
        command: build
        dockerfile: '$(Build.SourcesDirectory)/dockerfile'
        tags: |
          $(tag)
    - script: |
        # curl -X POST -H "Content-Type: application/json" -d "{\"content\": \"Le déploiement a été
↪  effectué : $(Build.Reason).\"}"
↪  https://discord.com/api/webhooks/1229740540178731099/ktCP8ziVjamWyZq5_Q7BPzQdlGQnNDwsihuvjE43B8O64sC4DVgyUej6G
      displayName: 'Send Discord Notification'

# <@UserID> pour notifier un utilisateur
```

Adapté au workflow 2.2.

4 Configurer un workflow pour exécuter des tests sur plusieurs versions d'un environnement de programmation (ex. différentes versions de Node.js).

```yaml
    name: Tests sur différentes versions de Node.js

on:
  push:
    branches:
      - main

jobs:
  test:
    name: Run Tests
    strategy:
      matrix:
        node-version: [ 14.x, 16.x, 20.x]
    runs-on: windows-latest

    steps:
    - name: Checkout Repository
      uses: actions/checkout@v2

    - name: Set up Node.js ${{ matrix.node-version }}
      uses: actions/setup-node@v2
      with:
        node-version: ${{ matrix.node-version }}

    - name: Install Dependencies
      run: npm install

    - name: Run Tests
      run: npm test
```

Les tests de différentes versions de nodes révèlent que les versions 12.x ne sont plus compatibles.

5 Créer un workflow qui utilise des matrices pour exécuter des tests parallèles.

```yaml
name: Parallel Tests

on:
  push:
    branches:
      - main

jobs:
  test:
    name: Run Parallel Tests
    runs-on: windows-latest
    strategy:
      matrix:
        node-version: [14.x, 16.x, 20.x]
    steps:
    - name: Checkout Repository
      uses: actions/checkout@v2

    - name: Set up Node.js ${{ matrix.node-version }}
      uses: actions/setup-node@v2
      with:
        node-version: ${{ matrix.node-version }}

    - name: Install Dependencies
      run: npm install

    - name: Run Tests
      run: npm test
```

La question 5 reprend la matrice de la question 4 (sauf 12.x) mais en permettant de lancer le test de toutes les versions d'environnement en parralèle plutôt que de manière séquentielle.

6 Développer un workflow qui utilise des artifacts pour passer des fichiers entre jobs.

```yaml
name: artifact-passing

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: windows-latest
    steps:
      - name: Create text file
        run: echo "Hello, world!" > example.txt

      - name: Upload artifact
        uses: actions/upload-artifact@v2
        with:
          name: my-artifact
          path: example.txt

  deploy:
    runs-on: windows-latest
    needs: build
    steps:
      - name: Download artifact
        uses: actions/download-artifact@v2
        with:
          name: my-artifact
          path: ./downloaded-artifact

      - name: Print artifact contents
        run: cat ./downloaded-artifact/example.txt
```

7 Écrire un script pour gérer les variables d'environnement de manière sécurisée dans GitHub Actions.

```yaml
name : Gestion de variable

on:
  push:
    branches:
      - main

jobs:
  test:
    name: Gestion de variable
    runs-on: windows-latest
    steps:
      - name: Set up environment
        run: echo "Super_variable_secrète=${{ secrets.variable_secrete }}" >> $GITHUB_ENV

      - name: Magnifique variable secrète
        run: echo $Super_variable_secrète
```

Création puis utilisation d'une variable d'environnement. Attention le echo montre des "*"

8 Configurer un job qui s'exécute seulement quand un changement a lieu dans un dossier spécifique du répertoire.

```yaml
name: Workflow avec déclenchement par changement dans un dossier

on:
  push:
    paths:
      - 'Exercice/*'  # Spécifiez le chemin du dossier à surveiller
  pull_request:
    paths:
      - 'Exercice/*'  # Spécifiez le chemin du dossier à surveiller

jobs:
  mon_job:
    name: Mon Job
    runs-on: windows-latest

    steps:
    - name: Checkout du code
      uses: actions/checkout@v2
```

Jobs se lançant bien uniquement suite à la modification du dossier "Exercice".

9 Créer un workflow pour automatiser le rollback en cas d'échec du déploiement.

```yaml
    name: Check Deployment Status and Trigger Rollback

on:
  workflow_run:
    workflows: [deploy-docker]
    types: [completed]

jobs:
  on-success:
    runs-on: ubuntu-latest
    if: ${{ github.event.workflow_run.conclusion == 'success' }}
    steps:
      - run: echo 'The triggering workflow passed'
  on-failure:
    runs-on: ubuntu-latest
    if: ${{ github.event.workflow_run.conclusion == 'failure' }} # Vérification si le workflow
↪   déclencheur a échoué
    steps:
      - name: Fail
        run: echo 'The triggering workflow failed'

      - name: Checkout code
        uses: actions/checkout@v2

      - name: Log in to Docker Hub
        uses: docker/login-action@v1
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Redéployer la version précédente
        run: |
          docker pull ${{ secrets.DOCKER_USERNAME }}/my-docker-image:latest
          docker push ${{ secrets.DOCKER_USERNAME }}/my-docker-image:latest
```

Lancement du workflow se déclenchant bien selon succes ou failure de workflow 2.

10 Réaliser un workflow pour intégrer l'analyse de code statique et les rapports de couverture de tests.

```yaml
name: Tests et couverture de code

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

jobs:
  test:
    name: Exécution des tests
    runs-on: windows-latest

    steps:
    - name: Checkout du code
      uses: actions/checkout@v2

    - name: Installer les dépendances
      run: npm install

    - name: Exécuter les tests Jest
      run: npm test

  coverage:
    name: Rapport de couverture
    runs-on: windows-latest

    steps:
    - name: Checkout du code
      uses: actions/checkout@v2

    - name: Installer les dépendances
      run: npm install

    - name: Exécuter les tests Jest avec la couverture
      run: npm test -- --coverage

    - name: Sauvegarder le rapport de couverture
      uses: actions/upload-artifact@v2
      with:
        name: coverage-report
        path: ./coverage
```

Résultats visibles lors de l'exécution. le rapport est disponible à la fin de la page action du workflow.