EX 1 : Mon premier workflow GitHub Actions

Objectifs

- Créer un workflow GitHub Actions, puis l'envoyer sur GitHub avec `git push`.
- Découvrir GitHub Actions **progressivement**, en ajoutant une commande par étape : `echo`, `date`, `pwd`, `ls`, etc.
- Comprendre que **chaque modification** du workflow est déclenchée par un **push**.

Préparation du dépôt local

1. Sur GitHub, créez un nouveau dépôt **vide** (sans template, sans fichier README, sans workflow) nommé : `ex-01-githubactions`

2. Sur votre poste local (terminal Git Bash) :

```
git clone git@github.com:<votre_user>/ex-01-githubactions.git
```

3. Ouvrez dans VS Code le répertoire de repo:

> Dans la barre de votre Explorateur de document rendez-vous dans le répertoire `ex-01-githubactions`, dans la bare de l'explorateur tapez `cmd`, puis dans le terminal :

```
code .
```

Étape 1 – Workflow minimal avec `echo`

1. Créez le fichier `.github/workflows/main.yml` :

Créer le répertoire en ligne de commande, puis le fichier `main.yml` et ouvrez VS Code :

```
cd ex-01-githubactions
mkdir -p .github/workflows
touch .github/workflows/main.yml
code .
```

ou depuis votre interface sous Windows (explorateur de document et VS Code).

Puis ajouter le contenu suivant dans le fichier `main.yml`:

```
name: Exercice 1 - mon premier workflow GitHub Actions
```

```
on: [push] #

jobs:
  demo:
    runs-on: ubuntu-latest
    steps:
      - name: 1) Message de bienvenue
        run: echo "Bienvenue dans mon premier workflow GitHub Actions !"
```

1. Vérifiez l'état du dépôt :

```
git status
```

3. Ajoutez et validez les fichiers :

```
git add .github/workflows/main.yml
git commit -am "Étape 1: workflow minimal avec echo"
git push
```

4. Sur GitHub, allez dans l'onglet **Actions** et vérifiez :

   - que le workflow s'est lancé,
   - que la sortie de la commande `echo` apparaît dans les logs.

> Question : *À quel moment le workflow s'est-il déclenché ?*

... votre réponse ici ... Push

Étape 2 – Ajouter la commande `date`

Nous allons maintenant **modifier** le workflow **en local**, puis refaire un `push`.

> La commande `date` permet de retourner la date actuel (au moment de l'execution de la commande).

1. Ouvrez `.github/workflows/main.yml` et **ajoutez une nouvelle étape** sous la première, comme ci-dessus :

```
    steps:
      - name: 1) Message de bienvenue
        run: echo "Bienvenue dans mon premier workflow GitHub Actions !"

      - name: 2) Afficher la date du système
        run: date
```

2. Enregistrez le fichier, puis dans le terminal :

```
git commit -am "Étape 2: ajout de la commande date"
git push
```

1. Allez dans **Actions** sur GitHub, ex: `https://github.com/<votre-identifiant>/ex-01-githubactions/actions`, et ouvrez la nouvelle exécution du workflow, et observez la sortie de la commande `date`.

> Question :
> *Pourquoi voit-on maintenant deux étapes dans le job ?* car il y a 2 name
> *Quelle est le résultat de la commande `date` ?* date de de aujourd'hui

... votre réponse ici ...

Étape 3 – Ajouter `pwd` pour voir le répertoire courant

> La commande `pwd` sous Linux retourne le répertoire où l'on se trouve lors de son exécution.

1. Ajoutez une nouvelle étape dans le job executant le commande `pwd`, avec comme **name** = `3) Afficher le répertoire courant`:

2. Puis appliquez ces modifications à votre repo local avec le message **Étape 3: ajout de la commande pwd** et publier les changement sur votre repo GitHub distant. Et regardez à nouveau dans les logs GitHub Actions.

> Question: *Quel est le chemin affiché par `pwd` ?

... votre réponse ici ... /home/runner/work/ex-01-githubactions/ex-01-githubactions

Étape 4 – Ajouter `ls -al` pour lister les fichiers du repo

> Cette commande `ls -al` permette de lister tout les fichiers du répertoire courant.

> Pour que le runner puisse afficher le contenu de votre dépôt, il faut d'abord faire un checkout du code du repo. Sans cette étape, le runner démarre dans un environnement vide.

1. Ajoutez une nouvelle étape faisant un checkout du repo dans le runner, puis la commande `ls -al` avec comme nom d'étape **4) Lister les fichiers du projet**.

```
- name: 4.a) Checkout du code source
  uses: actions/checkout@v5
- name: 4.b) Lister les fichiers du projet
  run: ls -al
```

2. Puis commitez avec le message **Étape 4: ajout de l'action checkout et de la commande ls -al** & push :

> Question :
> *Pourquoi voit-on le fichier du workflow dans la sortie de `ls -al` ?* car le ls -al montre tout les fichier meme ceux qui sont cacher

> *Quel fichiers et répertoire sont listés?*

total 20
drwxr-xr-x 4 runner runner 4096 Nov 21 13:13 .
drwxr-xr-x 3 runner runner 4096 Nov 21 13:13 ..
drwxr-xr-x 7 runner runner 4096 Nov 21 13:13 .git
drwxr-xr-x 3 runner runner 4096 Nov 21 13:13 .github
-rw-r--r-- 1 runner runner 21 Nov 21 13:13 README.md

... votre réponse ici ...

Étape 5 – Créer et lire un fichier dans le runner

1. Ajoutez maintenant 3 nouvelles étapes :

```
      - name: 5.a) Créer un fichier de test
        run: echo "Fichier créé par GitHub Actions" > testfile.txt
      - name: 5.b) Vérifier que le fichier existe
        run: ls -al
      - name: 5.c) Afficher le contenu du fichier
        run: cat testfile.txt
```

2. Commit avec le message **Étape 5: création et lecture d'un fichier** & push :

3. Dans GitHub Actions, ouvrez l'exécution la plus récente :

   - Vérifiez que `testfile.txt` apparaît bien dans la sortie de `ls -al`.
   - Vérifiez que son contenu est bien affiché par `cat`.

> Question : *Où est stocké ce fichier ? Persiste-t-il après la fin du job ?*

-rw-r--r-- 1 runner runner 34 Nov 21 13:16 testfile.txt

non

... votre réponse ici ...

Étape 6 – Ajouter vos réponses de l'exercice dans votre repo

Ajoutez ce fichier Markdown dans le repo et commitez & push sur le repo distant.