CI-CD

Greetings to you all !
Petit document pour expliquer le fonctionnement des Actions/Runners Github avec pour objectif de déployer une solution CI-CD.

On commence par le basique :

Créer son environnement de travaille

```
git init

git remote set-url origin git@github.com:User/UserRepo.git

mkdir .github && cd .github
mkdir workflows && touch workflows/basic_action.yml

cd ../..
```

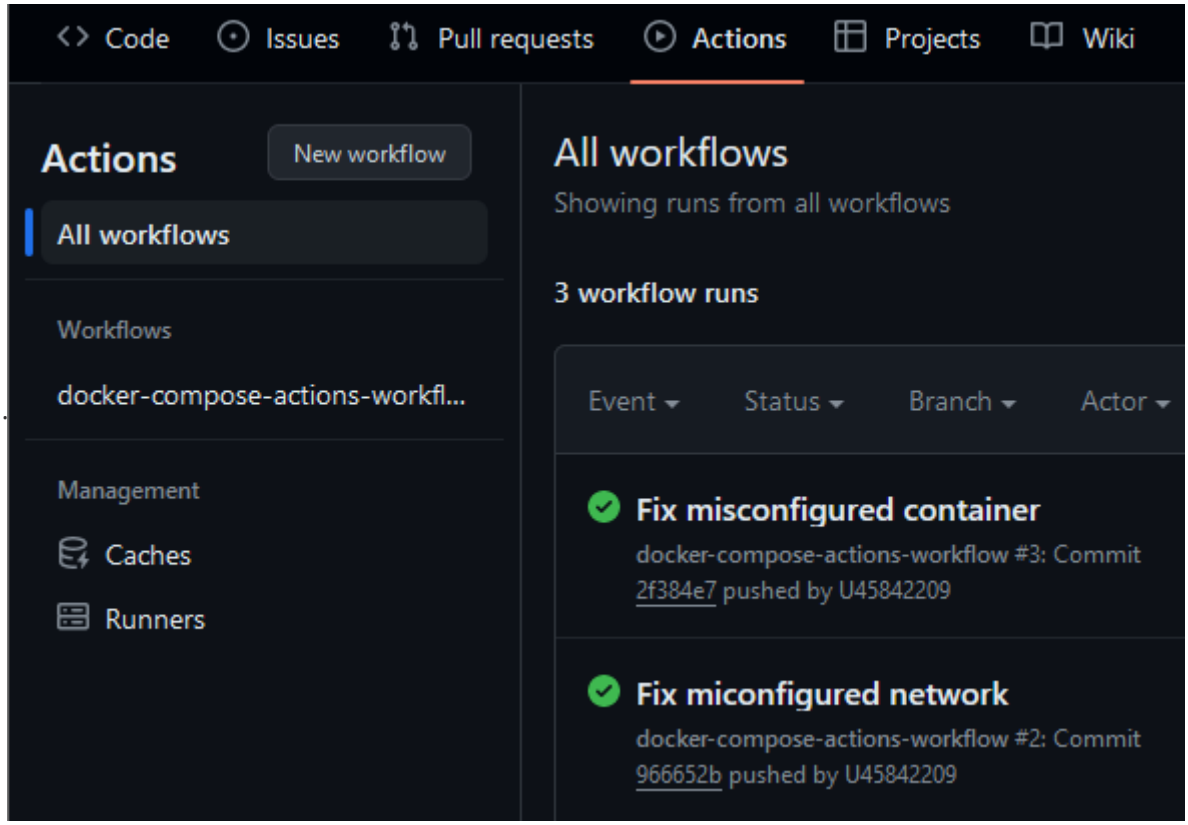
Dans cette première étape, on initie la création d'un repo Github et on le relie a un repo existant crée au préalable sur **github.com**.

On crée directement à la suite les fichiers/dossiers nécessaire pour la suite du projet.

> On oublie pas de pull/push les modifications ;)

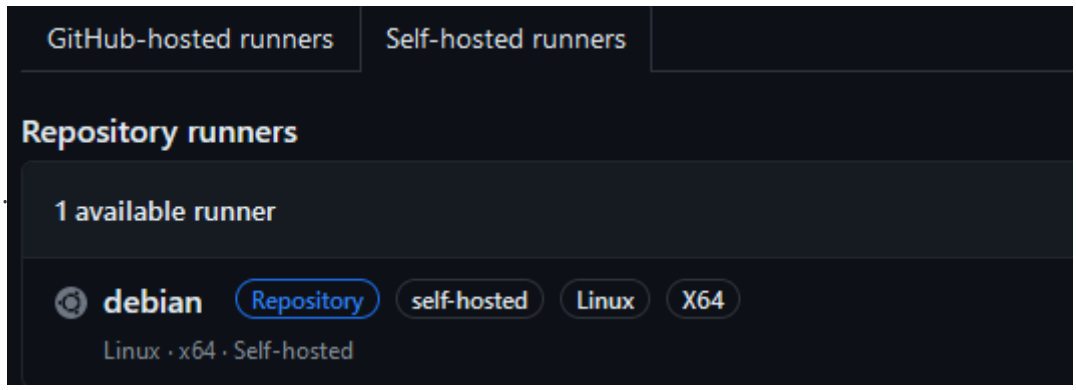
Vérification

On vérifie les actions, normalement elles n'existent pas encore, mais bon.



> Une fois le projet terminé vous retrouverez vos test de push/merge dans **Actions**.

Dans l'onglet **Runners -> Self-hosted runners**, on va venir créer une entité avec le bouton "New runner".



Le code pour configurer devrait ressembler à ceci :

```
# Download the latest runner package
curl -o actions-runner-linux-x64-2.315.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.315.0/actions-runner-
linux-x64-2.315.0.tar.gz

# Optional: Validate the hash
echo "6362646b67613c6981db76f4d25e68e463a9af2cc8d16e31bfeabe39153606a0  actions-
runner-linux-x64-2.315.0.tar.gz" | shasum -a 256 -c

# Extract the installer
tar xzf ./actions-runner-linux-x64-2.315.0.tar.gz

# Create the runner and start the configuration experience
./config.sh --url https://github.com/U45842209/CI-CD --token token

# Last step, run it!
nohup ./run.sh &
```

Configuration

Pour cet environnement, on va utiliser docker-compose :

```
version: '3.7'

services:
  ghost-server:
    image: ghost:5.70.1-alpine
    cap_add:
      - SYS_NICE
    security_opt:
      - seccomp:unconfined
    restart: always
    ports:
      - '2369:2368'
    depends_on:
      - ghost-db
    environment:
      database__client: mysql
      database__connection__host: ghost-db
```

```
      database__connection__user: ${MYSQLUSR}
      database__connection__password: ${MYSQLPWD}
      database__connection__database: ghost
    volumes:
      - 'ghost_conf:/var/lib/ghost/content'
    networks:
      admin:
        ipv4_address: 172.70.20.4

  ghost-db:
    image: mysql:8
    security_opt:
    - seccomp:unconfined
    restart: always
    command: --default-authentication-plugin=mysql_native_password
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQLPWD}
    volumes:
      - 'mysql_conf:/var/lib/mysql'
    networks:
      admin:
        ipv4_address: 172.70.20.3
```

> Ne pas oublier le .env pour les variables du docker-compose ;)

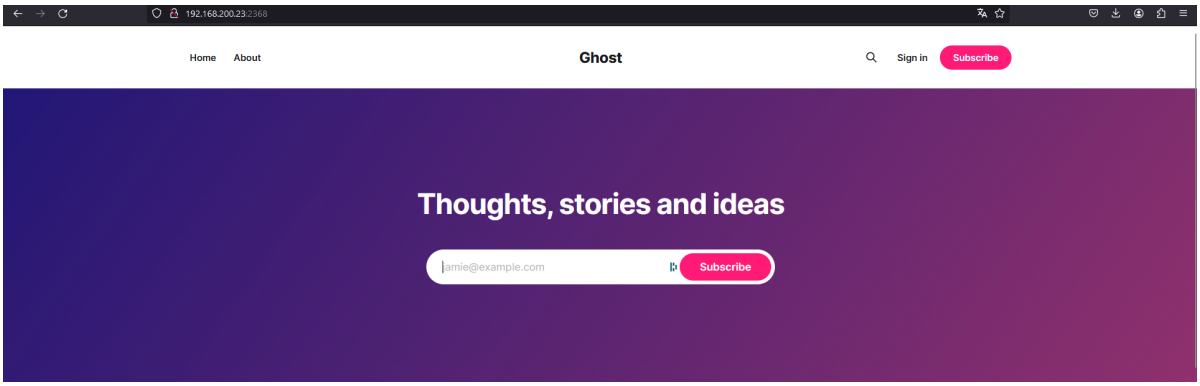
Le fichier workflow qui devra être dans le fichier crée initialement "basic_action.yml" :

```
name: docker-compose-actions-workflow

on:
  push:
    branches: [ "main" ]

permissions:
  contents: read

jobs:
  test:
    runs-on: self-hosted
    steps:
      - uses: actions/checkout@v2
      - name: Build the stack
        run: docker-compose up -d
```

Pour une petite explication : Ce workflow initie un test a chaque fois que un push est fait sur la branch "main".

Une fois que ces changements sont appliqués en **LOCAL** : add, commit, push !

Et vous devriez voir que les tests sont crée et que la solution est déployée sur le runners configuré en partie 2.



Et voilà, ça devrait fonctionner ! 😄

Le plus !

Les workflow s'active aussi en cas de pull request sur la branch main !