Atelier 3 — Introduction à Git, GitHub et Intégration avec Docker

Objectifs

À la fin de cet atelier, l'étudiant doit être capable de :

- Comprendre les notions de **contrôle de version**, de **dépôt distant** et de **collaboration via GitHub**.
- Créer un **dépôt Git** local et le synchroniser avec **GitHub**.
- Versionner un projet Dockerisé (comme l'API Flask).
- Automatiser le **build et le push Docker** grâce à **GitHub Actions (CI/CD)**.

1. Comprendre les outils

Outil	Définition	Utilité principale
Git	Système de **contrôle de version** distribué.	Sauvegarder et suivre l'évolution du code.
GitHub	Plateforme collaborative basée sur Git.	Héberger le code, collaborer et exécuter des workflows (CI/CD).
Docker	Technologie de conteneurisation.	Exécuter l'application dans un environnement isolé et portable.
GitHub Actions	Service d'intégration continue (CI/CD) intégré à GitHub.	Automatiser les tests, builds et déploiements.

2. Notions clés de Git

a. Commit

Une *photo du code* à un instant T.

Exemple : `git commit -m "Ajout du fichier app.py"`

b. Push / Pull

- **Push** : envoi des changements vers le dépôt distant (GitHub).
- **Pull** : récupération des dernières modifications du dépôt distant.

c. Branches

Les branches permettent de travailler sur plusieurs versions du code sans tout casser.

Exemple : une branche `main` stable, et une branche `dev` pour expérimenter.

d. Tags

Les *tags* marquent une version du projet (ex. : `v1.0.0` pour une première release).

3. Étapes pratiques

Étape 1 — Initialiser le projet

```
git init
git add .
git commit -m "Initialisation du projet Flask avec Docker"
```

Étape 2 — Ajouter un .gitignore

Fichier `.gitignore` (pour éviter d'ajouter les fichiers inutiles) :

```
__pycache__/
*.pyc
.venv/
.env
*.log
```

4. Créer un dépôt GitHub

1. Connecte-toi à GitHub.
2. Clique sur **New Repository → Nom : flask-ann-api**.
3. Ne coche **aucune case** (pas de README, ni .gitignore).
4. Copie la commande indiquée par GitHub :
5. `git remote add origin https://github.com/<ton_nom>/flask-ann-api.git`
6. `git branch -M main`
7. `git push -u origin main`

5. Créer une première version Dockerisée

Assure-toi que ton projet contient :

- `app.py`
- `requirements.txt`
- `Dockerfile`

Puis lance :

```
docker build -t flask-ann-api .
docker run -p 5000:5000 flask-ann-api
```

Quand tout marche :

```
git add .
git commit -m "Ajout du Dockerfile et test réussi"
git push
```

6. Mettre en place une Intégration Continue (CI/CD)

a) Créer les secrets Docker Hub

Sur GitHub :

- **Settings → Secrets and variables → Actions → New secret**
  - `DOCKERHUB_USERNAME` = ton nom Docker Hub
  - `DOCKERHUB_TOKEN` = ton *access token Docker Hub*

b) Créer un fichier de workflow CI

Chemin : `.github/workflows/docker-image.yml`

```yaml
name: Build & Push Docker Image

on:
  push:
    branches: [ "main" ]

jobs:
  docker:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout le code
        uses: actions/checkout@v4

      - name: Connexion à Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}

      - name: Build et Push de l'image
        uses: docker/build-push-action@v6
        with:
          context: .
          push: true
          tags: ${{ secrets.DOCKERHUB_USERNAME }}/flask-ann-api:latest
```

c) Explication du fichier CI/CD

Élément	Signification
`on: push`	Le pipeline s'exécute automatiquement à chaque `git push` sur la branche `main`.
`runs-on: ubuntu-latest`	GitHub exécute le workflow sur une VM Linux.
`docker/login-action`	Permet de se connecter à Docker Hub.
`docker/build-push-action`	Construit l'image et la pousse sur Docker Hub.

7. Vérifier l'exécution du workflow

1. Sur GitHub → onglet **Actions**
2. Clique sur le dernier workflow → tu verras toutes les étapes :
   checkout → login → build → push

Si tout est vert, ton image est maintenant sur Docker Hub :
https://hub.docker.com/repository/docker/<ton_user>/flask-ann-api

8. Sur une autre machine

Télécharger et exécuter l'image :

```
docker pull <ton_user>/flask-ann-api:latest
docker run -p 5000:5000 <ton_user>/flask-ann-api:latest
```

Tester avec Postman :

```
GET http://127.0.0.1:5000/ping
```

9. Extensions possibles

- Ajouter **des tests unitaires** exécutés avant le build.
- Taguer automatiquement les images selon la version (`v1.0.0`).
- Ajouter un déploiement vers le **Cloud Run** ou **Kubernetes** (prochain atelier !).

Résumé pédagogique

Étape	Compétence acquise
1	Initialiser un projet Git
2	Publier un code sur GitHub
3	Dockeriser une application
4	Configurer un pipeline CI/CD
5	Publier et déployer une image Docker
6	Tester et automatiser un déploiement