Exercice Pratique : Git & GitHub pour le Travail Collaboratif

Année : 2025-2026

Objectifs de l'Exercice

À la fin de cet exercice pratique, vous serez capable de :

1. **Cloner** et **mettre à jour** un dépôt Git.
2. Travailler sur une **branche** dédiée pour une fonctionnalité.
3. Effectuer des **validations (commits)** avec des messages clairs.
4. Soumettre votre travail à un dépôt principal via une **Pull Request (PR)** sur GitHub.
5. Gérer le flux de travail collaboratif de base (branchement, développement, fusion).

Pré-requis

1. Avoir un compte **GitHub**.
2. Avoir **Git** installé sur votre machine.
3. Avoir configuré votre identité Git :

```
$ git config --global user.name "Votre Nom"
$ git config --global user.email votre.email@example.com

```

1 Partie 1 : Préparation et Clonage du Projet

Étape 1 : Clonage du Dépôt du Workshop

L'animateur fournira l'URL d'un dépôt GitHub de base (ex: `Workshop-Git-Base`).

1. Ouvrez votre terminal et clonez le dépôt. Remplacez `[URL_DU_DEPOT]` par l'URL fournie.

```
$ git clone [URL_DU_DEPOT] workshop-git

```

2. Accédez au répertoire cloné.

```
$ cd workshop-git

```

3. Vérifiez l'état de votre copie de travail et la branche courante.

```
$ git status

```

2 Partie 2 : Développer une Nouvelle Fonctionnalité

Étape 2 : Créer et Basculer vers une Nouvelle Branche

Pour isoler vos modifications, vous travaillerez sur une branche thématique.

1. Créez une nouvelle branche et basculez-y immédiatement. Utilisez le format `feat-[vos-initiales]`.

```
$ git checkout -b feat-yl

```

2. Vérifiez que vous êtes sur la nouvelle branche.

```
$ git status

```

Étape 3 : Modification, Indexation et Validation (Commit)

Le projet contient un fichier `collaborateurs.txt`. Votre tâche est d'ajouter votre nom à cette liste.

1. Ouvrez le fichier `collaborateurs.txt` et ajoutez votre nom et votre rôle sur une nouvelle ligne (ex: `Jane Doe, Developpeuse`).

2. Après la modification, vérifiez l'état de votre fichier.

```
$ git status

```

3. Indexez le fichier modifié.

```
$ git add collaborateurs.txt

```

4. Validez les modifications avec un message de commit clair et descriptif.

```
$ git commit -m "feat: Ajout de mon nom    la liste des collaborateurs"

```

3 Partie 3 : Partage et Collaboration avec GitHub

Étape 4 : Pousser la Branche sur GitHub

Partagez votre travail local avec le dépôt distant.

1. Poussez votre nouvelle branche vers GitHub.

```
$ git push --set-upstream origin feat-votre-branche

```

Étape 5 : Créer une Pull Request (PR)

La Pull Request est la demande de fusion de votre fonctionnalité dans la branche principale (`main`).

1. Ouvrez le dépôt sur **GitHub** dans votre navigateur.
2. Créez une nouvelle **Pull Request** de votre branche (`feat-votre-branche`) vers la branche de base (`main`).
3. Remplissez le titre et la description de la PR, puis soumettez-la.
4. L'animateur (ou le propriétaire du projet) examinera et **fusionnera (Merge)** votre Pull Request.

Étape 6 : Mettre à Jour votre Branche Principale

Une fois la PR fusionnée, votre branche locale `main` doit être synchronisée avec les dernières modifications du dépôt distant.

1. Retournez à la branche principale locale.

```
$ git checkout main

```

2. Récupérez et fusionnez les modifications distantes.

```
$ git pull origin main

```

3. Vérifiez le contenu de `collaborateurs.txt` pour vous assurer que votre nom est présent.

4 Exercices Supplémentaires et Défi (Si le temps le permet)

Défi 1 : La Rectification de Commit

Simulez l'oubli d'un fichier dans une validation.

1. Créez un nouveau fichier `notes.txt`.
2. Modifiez le fichier `collaborateurs.txt`.
3. Indexez et validez **uniquement** `collaborateurs.txt`.

```
$ git add collaborateurs.txt
$ git commit -m "feat: Mise a jour mineure"

```

4. Vous réalisez que vous avez oublié `notes.txt`. Indexez-le.

```
$ git add notes.txt

```

5. Utilisez la commande de rectification pour ajouter `notes.txt` au **dernier commit** sans créer un nouveau commit séparé.

```
$ git commit --amend

```

6. Vérifiez votre historique avec `git log`. Combien de nouveaux commits voyez-vous ? [cite$_s$*tart*](*Rponseattendue* : 1*nouveaucommitquiremplacel'ancien*[*cite* : 250]).

Défi 2 : Annulation d'Indexation

1. Modifiez le fichier `main.c` (ou un autre fichier de code).
2. Indexez la modification : `git add main.c`.
3. Réalisez que vous ne vouliez pas inclure cette modification dans le prochain commit.
4. Utilisez une commande pour annuler l'indexation de `main.c` (le ramener à l'état de "Modifications qui ne seront pas validées" / "Changes not staged for commit").

```
$ git reset HEAD main.c

```

5. Vérifiez l'état.

```
$ git status

```