📖

Les bases de Git

1. Créer un dépôt Git local

Un dépôt Git est un répertoire dans lequel Git peut suivre les modifications des fichiers.

Commande pour créer un dépôt :

```
git init
```

- Cette commande initialise un dépôt Git dans le répertoire courant.
- Un dossier caché `.git` est créé. Ce dossier contient toutes les informations nécessaires pour que Git suive les fichiers.

Exemple :

```
mkdir my_project
cd my_project
git init
```

2. Cloner un dépôt distant

Si un projet existe déjà sur une plateforme distante comme GitHub, GitLab ou Bitbucket, il est possible de le cloner pour obtenir une copie locale.

Commande pour cloner un dépôt :

```
git clone URL_REPOSITORY
```

- L'URL peut être HTTPS, SSH, ou un chemin local.
- Le dépôt distant est copié dans un répertoire local, et Git configure automatiquement un remote (`origin`) pour le dépôt cloné.

Exemple :

```
git clone https://github.com/user/project.git
```

3. Les trois zones de Git

Git fonctionne en trois zones principales :

1. **Working Directory** : Répertoire de travail contenant les fichiers.
2. **Staging Area** : Zone intermédiaire où on prépare les modifications pour le commit.
3. **Commit History** : Historique des modifications validées.

Cycle de base d'un fichier :

1. Modifier un fichier dans le répertoire de travail.
2. Ajouter ce fichier à la **staging area** avec `git add`.
3. Valider les modifications dans l'historique avec `git commit`.

4. Les commandes de base

4.1. Vérifier l'état des fichiers

```
git status
```

- Affiche les fichiers modifiés, ajoutés ou supprimés.
- Indique les fichiers suivis/non suivis et ceux prêts pour un commit.

Exemple de sortie :

```
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
        modified: file.txt
```

4.2. Ajouter des fichiers à la staging area

```
git add file
```

- Prépare un fichier pour le commit.
- Pour ajouter tous les fichiers modifiés :

  ```
  git add .
  ```

4.3. Faire un commit

```
git commit -m "commit message"
```

- Un commit enregistre les modifications de la staging area dans l'historique.
- Le message doit être descriptif pour expliquer les changements.

4.4. Afficher l'historique des commits

```
git log
```

- Affiche la liste des commits avec leur hash, l'auteur, la date et le message.

Options utiles :

- Format simplifié :

  ```
  git log --oneline
  ```

- Ajouter un graphe visuel pour voir les branches :

  ```
  git log --graph --oneline
  ```

Exemple d'un workflow simple

1. **Créer un fichier dans le répertoire local :**

```
echo "Hello Git :)" > file.txt
```

2. **Vérifier l'état des fichiers :**

```
git status
```

Sortie :

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      file.txt
```

3. **Ajouter le fichier à la staging area :**

```
git add fichier.txt
```

4. **Vérifier de nouveau l'état :**

```
git status
```

Sortie :

```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      new file:   fichier.txt
```

5. **Faire un commit :**

```
git commit -m "Adding test file"
```

6. **Afficher l'historique :**

```
git log --oneline
```

5. Suppléments pour les bases

5.1. Annuler des modifications non ajoutées

Pour restaurer un fichier à son état initial (avant modification) :

```
git restore fichier
```

5.2. Retirer un fichier de la staging area

Pour retirer un fichier de la staging area sans perdre les modifications :

```
git restore --staged fichier
```