📖

Travailler avec des fichiers

1. Suivre et ignorer des fichiers

1.1 Suivre un fichier

Pour commencer à suivre un fichier avec Git, on doit l'ajouter à la **staging area** à l'aide de la commande :

```
git add file_name
```

1.2 Ignorer des fichiers avec `.gitignore`

> 💡 Voir section GitIgnore pour plus de détails.

Certains fichiers ne doivent pas être suivis par Git, comme les fichiers de configuration locaux, les fichiers temporaires ou les dépendances compilées.

Étapes :

1. Créer un fichier nommé `.gitignore` à la racine du dépôt.
2. Ajouter les règles pour ignorer des fichiers ou des répertoires.

Exemples de règles :

```
# Ignorer tous les fichiers .log
*.log

# Ignorer le dossier node_modules
node_modules/

# Ignorer un fichier spécifique
config.local.json
```

> Les fichiers déjà suivis ne seront pas ignorés même s'ils sont dans .gitignore. Dans ce cas, nous devons les retirer du suivi avec :

```
git rm --cached file_name
```

2. Renommer et déplacer des fichiers

Git permet de renommer ou de déplacer des fichiers tout en conservant leur historique.

2.1. Renommer un fichier :

```
git mv old_name new_name
```

- Cette commande effectue à la fois le déplacement du fichier et l'ajout à la staging area.

Exemple :

```
git mv file.txt new_name.txt
```

2.2. Déplacer un fichier dans un répertoire :

```
git mv file.txt folder/
```

- Si le répertoire n'existe pas, Git le crée automatiquement.

3. Supprimer des fichiers du dépôt

Git offre deux façons de supprimer un fichier :

3.1. Supprimer un fichier du suivi (mais le conserver localement) :

Si nous voulons que Git cesse de suivre un fichier mais que celui-ci reste dans le répertoire local :

```
git rm --cached file_name
```

3.2. Supprimer un fichier complètement (localement et dans le dépôt) :

Pour supprimer un fichier du dépôt et du système de fichiers :

```
git rm file_name
```

Exemple :

```
git rm file_to_delete.txt
git commit -m "Deleting file_to_delete.txt"
```

4. Restaurer des fichiers

Git permet de restaurer des fichiers supprimés ou modifiés à partir de l'historique.

4.1. Restaurer un fichier supprimé (avant un commit) :

Si on a supprimé un fichier par erreur et que l'on n'a pas encore validé (commit) :

```
git restore file_name
```

4.2. Restaurer un fichier supprimé (après un commit) :

Si le fichier a été supprimé et que la suppression a été validée :

1. Trouver le dernier commit contenant le fichier avec :

   ```
   git log -- file_name
   ```

2. Restaurer le fichier avec :

   ```
   git checkout COMMIT_SHA -- file_name
   ```

5. Vérifier l'état des fichiers

Pour surveiller les modifications dans le dépôt, utiliser :

```
git status
```

Statuts possibles :

- **Untracked files** : Fichiers non suivis (pas encore ajoutés à Git).
- **Changes not staged for commit** : Fichiers modifiés mais pas encore ajoutés à la staging area.
- **Changes to be committed** : Fichiers prêts à être validés.

6. Historique et différentiel des fichiers

6.1. Voir les modifications apportées à un fichier

Pour voir les différences entre la version actuelle d'un fichier et sa dernière version commitée :

```
git diff file_name
```

6.2. Voir les commits affectant un fichier spécifique

Pour afficher l'historique des modifications d'un fichier :

```
git log -- file_name
```

7. Résumé des commandes principales liées aux fichiers

Action	Commande
Ajouter un fichier	`git add fichier`
Ignorer un fichier	Ajouter une règle dans `.gitignore`
Renommer un fichier	`git mv ancien_nom nouveau_nom`
Déplacer un fichier	`git mv fichier dossier/`
Supprimer un fichier (local)	`git rm fichier`
Retirer un fichier du suivi	`git rm --cached fichier`
Restaurer un fichier supprimé	`git restore fichier`
Voir les modifications	`git diff fichier`