📖

Gestion de l’historique

1. Exploration de l’historique

1.1 Afficher l’historique des commits

- **Commande de base :**

  ```
  git log
  ```

  Affiche les commits dans l’ordre chronologique inverse.

- **Options utiles :**
  - `-oneline` : Affiche chaque commit sur une ligne.
  - `-graph` : Montre une représentation visuelle des branches et fusions.
  - `-since` et `-until` : Filtre les commits par date. Exemple :

    ```
    git log --since="2 weeks ago"
    ```

1.2. Visualiser les changements

- Voir les différences dans un fichier entre deux commits :

  ```
  git diff COMMIT1 COMMIT2 -- file_name
  ```

- Comparer un fichier avec sa version dans le dernier commit :

  ```
  git diff HEAD -- file_name
  ```

1.3. Identifier l’auteur d’une modification

On peut utiliser la commande `git blame` pour voir quel commit a modifié chaque ligne d’un fichier :

```
git blame file_name
```

2. Revenir à une version antérieure

2.1. Revenir à un commit spécifique

- Inspecter un commit sans modifier l’historique :

  ```
  git checkout SHA_COMMIT
  ```

- Pour revenir définitivement à cet état et supprimer les modifications ultérieures :

  ```
  git reset --hard SHA_COMMIT
  ```

2.2. Annuler un commit

1. **Annuler un commit tout en conservant les changements dans l'espace de travail** :

```
git reset --soft HEAD~1
```

2. **Annuler un commit et supprimer les modifications associées** :

```
git reset --hard HEAD~1
```

2.3. Annuler les modifications d'un commit

Pour créer un nouveau commit qui annule les effets d'un commit précédent sans modifier l'historique :

```
git revert SHA_COMMIT
```

3. Les tags pour organiser l'historique

Les tags sont des marqueurs utilisés pour identifier des versions spécifiques du code (souvent liés à des versions de production).

3.1. Créer un tag

- Tag léger :

```
git tag v1.0
```

- Tag annoté (avec un message) :

```
git tag -a v1.0 -m "Initial Version"
```

3.2. Lister les tags

```
git tag
```

3.3. Pousser les tags sur le dépôt distant

```
git push origin --tags
```

4. Nettoyer l'historique

4.1. Supprimer un commit spécifique

- Supprimer un commit intermédiaire (sans perdre les modifications) :

```
git rebase -i COMMIT^
```

Simplement supprimer la ligne associée au commit.

4.2. Suppression d'un tag

- En local :

```
git tag -d v1.0
```

- Sur le dépôt distant :

```
git push origin --delete v1.0
```

4.3. Nettoyage des branches

Pour supprimer les branches locales inutilisées :

```
git branch -d branch_name
```

5. Résumé des commandes principales

Action	Commande
Voir l'historique des commits	`git log`, `git log --oneline`
Voir les changements	`git diff`, `git blame`
Modifier un commit	`git commit --amend`
Inverser un commit	`git revert SHA_COMMIT`
Supprimer un commit	`git reset --hard` ou `git rebase -i`