CRÉER UNE BRANCHE, FUSIONNER ET RÉSOUDRE UN CONFLIT

1. COMPRENDRE AVANT DE MANIPULER

1.1 QU'EST-CE QU'UNE BRANCHE ?

Une branche est une ligne de travail parallèle à la branche principale du projet. Elle permet d'ajouter une fonctionnalité, corriger un bug ou tester quelque chose sans modifier la version officielle.

MAIN

version officielle du projet

AUTRES BRANCHES

zones de travail isolées

1.2 POURQUOI UTILISE-T-ON LES BRANCHES ?

Les branches permettent de :

- **TRAVAILLER EN SÉCURITÉ**
- **ÉVITER DE CASSER LE CODE PRINCIPAL**
- **COLLABORER SANS SE MARCHER DESSUS**
- **GARDER UN HISTORIQUE CLAIR**
- **TESTER SANS RISQUE**

1.3 QU'EST-CE QU'UN MERGE ?

Un merge est une fusion entre deux branches.

On fusionne généralement une branche de travail → vers main.

Deux cas existent :

FUSION AUTOMATIQUE

Git peut fusionner parce que les deux branches n'ont pas modifié la même zone.

CONFLIT

Les deux branches ont modifié la même ligne dans le même fichier. Git ne sait pas laquelle garder → c'est à l'utilisateur de décider.

1.4 COMMENT GIT AFFICHE UN CONFLIT DANS LE FICHIER ?

Lorsqu'un conflit apparaît, Git inscrit :

```
<<<<<<< HEAD
(Version actuelle sur main)
=======
(Version venant de l'autre branche)
>>>>>>> nom_de_branche
```

L'utilisateur doit choisir la bonne version et supprimer ces marques.

1.5 OBJECTIF DE L'EXERCICE

À la fin de cet exercice, tu sauras :

- CRÉER UNE BRANCHE
- TRAVAILLER DESSUS
- MODIFIER MAIN
- FUSIONNER
- PROVOQUER VOLONTAIREMENT UN CONFLIT
- LIRE LE CONFLIT
- RÉSOUDRE LE CONFLIT
- FINALISER CORRECTEMENT LA FUSION

1. PRÉPARATION

Assure-toi d'être dans ton projet Git.

VÉRIFIER LE DOSSIER ACTUEL :

```
pwd
```

VÉRIFIER L'ÉTAT GIT :

```
git status
```

Tu dois voir :

```
On branch main
nothing to commit, working tree clean
```

2. CRÉER ET UTILISER UNE BRANCHE

01

CRÉER LA BRANCHE :

```
git branch fonctionnalite-1
```

02

BASCULER SUR LA BRANCHE :

```
git checkout fonctionnalite-1
```

03

VÉRIFIER LES BRANCHES :

```
git branch
```

L'astérisque indique la branche active.

3. MODIFIER UN FICHIER SUR LA BRANCHE

Ouvre le fichier README.md et ajoute une ligne :

> Cette modification a été faite sur la branche fonctionnalite-1.

AJOUTER AU STAGING :

```
git add README.md
```

ENREGISTRER (COMMIT) :

```
git commit -m "Ajout de texte sur fonctionnalite-1"
```

4. MODIFIER LE MÊME FICHIER DANS MAIN

RETOURNER SUR MAIN :

```
git checkout main
```

Modifie la même zone dans README.md :

> Cette modification a été faite sur la branche main.

AJOUTER ET ENREGISTRER :

```
git add README.md
git commit -m "Ajout de texte sur main"
```

À ce stade, les deux branches ont modifié la même partie du même fichier → conflit garanti.

5. FUSIONNER ET PROVOQUER LE CONFLIT

Depuis main :

```
git merge fonctionnalite-1
```

Git affiche :

```
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Ce résultat est normal. Git te signale qu'il ne peut pas fusionner seul.

6. LIRE ET COMPRENDRE LE CONFLIT

Ouvre README.md. Tu verras :

```
<<<<<<< HEAD
Cette modification a été faite sur la branche main.
=======
Cette modification a été faite sur la branche fonctionnalite-1.
>>>>>>> fonctionnalite-1
```

Ces zones représentent :

- partie haute = ce qui est dans main
- partie basse = ce qui vient de la branche fonctionnalite-1

7. RÉSOUDRE LE CONFLIT

Tu dois choisir ce que tu veux garder.

POSSIBLE CHOIX A : GARDER LA VERSION MAIN

Cette modification a été faite sur la branche main.

POSSIBLE CHOIX B : GARDER LA VERSION FONCTIONNALITE-1

Cette modification a été faite sur la branche fonctionnalite-1.

POSSIBLE CHOIX C : COMBINER LES DEUX

Cette modification a été faite sur main et fonctionnalite-1.

Après avoir choisi :

- supprime les lignes :

```
<<<<<<< HEAD
=======
>>>>>>> fonctionnalite-1
```

- enregistre le fichier

8. INFORMER GIT QUE LE CONFLIT EST RÉSOLU

```
git add README.md
```

Ce geste indique à Git que :

> « le fichier est propre, je valide ma résolution ».

Finaliser la fusion :

```
git commit -m "Résolution du conflit entre main et fonctionnalite-1"
```

9.VÉRIFIER LA FUSION

Afficher l'historique en graphe :

```
git log --oneline --graph --decorate --all
```

Ce graph montre :

- les branches
- leur séparation
- leur fusion
- le commit final de résolution

LES 6 FAÇONS DE GÉRER UN CONFLIT GIT

1. MÉTHODE CLASSIQUE (CELLE QUE TU AS DÉJÀ) – RÉSOUDRE DANS LE FICHIER

Quand utiliser ?

Toujours. C'est la méthode de base.

Étapes :

Ouvrir le fichier en conflit

Lire les marques :

Choisir la version que tu souhaite

Supprimer les marques

.git add

.git commit

C'est la méthode universelle.

2. GARDER ENTIÈREMENT LA VERSION ACTUELLE (MAIN)

Commande :

git checkout --ours chemin/du/fichier

Quand utiliser ? Quand tu sais que ce qui vient de main est la version à garder.

Ex : "Ours" = notre version = main.

Ensuite :

git add fichier git commit

3. GARDER ENTIÈREMENT LA VERSION DE L'AUTRE BRANCHE

Commande :

git checkout --theirs chemin/du/fichier

Quand utiliser ? Quand tu veux remplacer ce que tu as dans main par ce qui vient de fonctionnalite-1.

Explication simple : "Theirs" = la version de l'autre branche.

Puis :

git add fichier git commit

4. ANNULER COMPLÈTEMENT LE MERGE (RECULER EN ARRIÈRE)

Commande :

git merge --abort

Quand utiliser ? Quand tu veux arrêter le merge parce que :

tu t'es trompé de branche

tu veux recommencer

tu veux d'abord corriger autre chose

Git remet l'état exact avant le merge.

5. UTILISER UN OUTIL VISUEL DE RÉSOLUTION (VS CODE INTÉGRÉ)

Quand utiliser ? Quand plusieurs fichiers sont en conflit ou que les élèves paniquent.

Dans VS Code :

Le conflit affiche 3 boutons :

Accept Current Change

Accept Incoming Change

Accept Both Changes

Compare Changes

Cette méthode est la plus intuitive pour les débutants.

Ensuite :

git add fichier git commit

6. PRENDRE LES DEUX VERSIONS AUTOMATIQUEMENT (SANS LES TAPER À LA MAIN)

Git peut fusionner les deux versions sans avoir besoin d'écrire toi-même.

Commande :

git merge -X ours fonctionnalite-1

→ Git prend systématiquement la version de main (et ignore les conflits de texte).

Ou :

git merge -X theirs fonctionnalite-1

→ Git prend systématiquement la version de l'autre branche.

Attention : à utiliser en équipe uniquement quand on sait ce qu'on fait. Mais c'est très utile en formation pour montrer des cas avancés.

RÉSUMÉ DE L'EXERCICE

01

CRÉER UNE BRANCHE :

```
git checkout -b fonctionnalite-1
```

02

MODIFIER, AJOUTER, COMMITTER

03

MODIFIER LA MÊME LIGNE SUR MAIN

04

FUSIONNER :

```
git merge fonctionnalite-1
```

05

RÉSOUDRE LE CONFLIT DANS LE FICHIER

06

AJOUTER ET ENREGISTRER LA RÉSOLUTION :

```
git add fichier
git commit -m "Résolution du conflit"
```

07

VÉRIFIER :

```
git log --oneline --graph --decorate --all
```