Git: les branches (branching)

Quand tu développes un projet, tu ne travailles pas toujours directement sur la version principale (`main` ou `master`). Parfois, tu veux tester une nouvelle idée, corriger un bug ou développer une nouvelle fonctionnalité **sans casser** ce qui fonctionne déjà. 👉 C'est exactement à ça que servent les **branches** en Git : **elles permettent de travailler en parallèle sur différentes versions d'un projet**.

6TTR 5TTR 6TQ Découverte

Le branching en Git et GitHub

Objectifs du cours

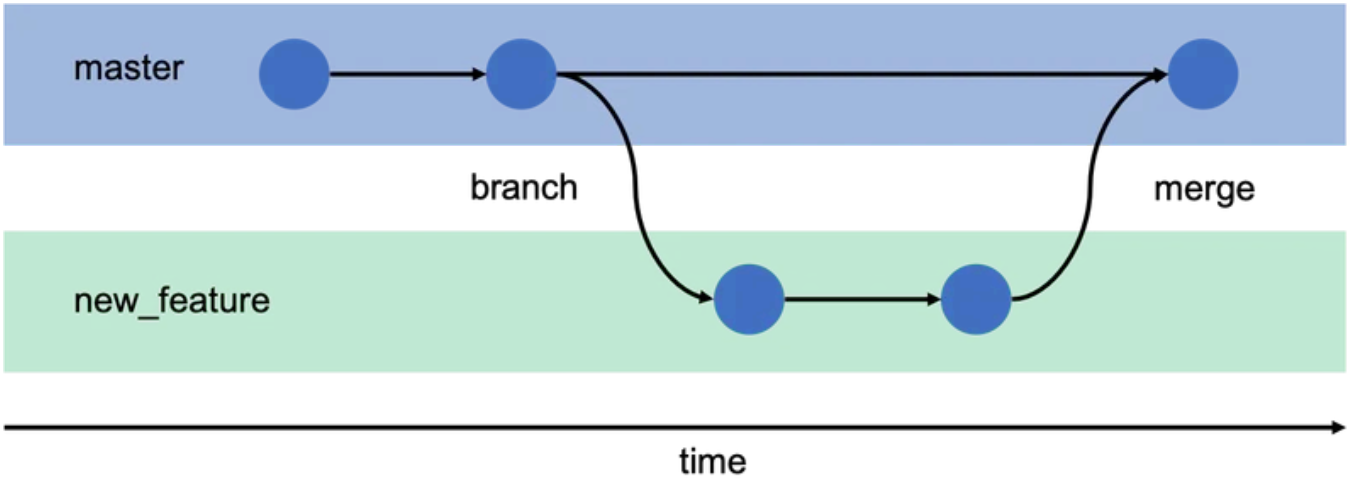
- Comprendre ce qu'est une branche dans Git.
- Savoir pourquoi et quand utiliser des branches.
- Savoir créer, changer et fusionner une branche.
- Découvrir les bonnes pratiques de travail avec GitHub.

Notion essentielle

Une branche, c'est quoi ?

Une branche est simplement un **pointeur vers un état particulier de ton code**. Il ne s'agit donc pas d'une copie complète de ton code. Créer une branche est une opération *très légère* en Git. Cela ne demande pas beaucoup de ressources (espace disque).

La branche principale s'appelle généralement `main` ou `master`. Quand tu crées une nouvelle branche, tu obtiens une **copie indépendante** du projet à partir de laquelle tu peux travailler.



Ainsi :

- Tu peux coder sans risque de casser le code principal.
- Plusieurs personnes peuvent travailler en même temps sur des fonctionnalités différentes.
- Tu peux tester des idées avant de les valider.

Pourquoi utiliser des branches ?

1. **Expérimenter sans risque** Tu veux tester/créer une nouvelle fonctionnalité → tu crées une branche. Si ça ne marche pas, tu la supprimes, et ton projet principal reste propre.
2. **Travailler en équipe** Chacun peut développer sa partie du projet sur une branche différente, puis les fusionner quand elles sont prêtes.
3. **Organisation claire** On sait directement où se trouvent les développements en cours : branche `feature-login`, branche `fix-bug-42`, etc.

Les commandes principales

Créer une nouvelle branche

```
git branch ma-branche
```

Aller sur une branche

```
git checkout ma-branche
```

ou en version moderne :

```
git switch ma-branche
```

Créer et aller sur une branche en une seule commande

```
git checkout -b ma-branche
```

Voir toutes les branches

```
git branch
```

Fusionner une branche dans `main`

Tu dois être sur `main` :

```
git checkout main
git merge ma-branche
```

Supprimer une branche devenue inutile

```
git branch -d ma-branche
```

Avec GitHub

Quand tu as fini de travailler sur ta branche en local :

- Tu l'envoies sur GitHub pour la sauvegarder :

```
git push -u origin ma-branche
```

- Tu peux alors continuer ton travail depuis un autre ordinateur, ou partager la branche avec d'autres.
- Une fois fusionnée localement avec `main`, pense aussi à pousser `main` :

```
git push origin main
```

Exemple concret

Tu travailles sur un site web :

1. Le projet en ligne est sur `main`.
2. Tu veux ajouter une page de contact → tu crées une branche `feature-contact`.
3. Tu développes et testes sans casser le reste du site.
4. Quand tout est bon, tu fusionnes `feature-contact` dans `main`.
5. Tu supprimes la branche devenue inutile.

Bonnes pratiques

- Donne des noms clairs à tes branches (`feature-login`, `fix-typo-header`...).
- Une branche = une idée ou une fonctionnalité.
- Supprime les branches inutiles une fois fusionnées.
- **Toujours garder** `main` **fonctionnel** (elle doit toujours pouvoir être utilisé).

À retenir

1. Une **branche** = une version/copie parallèle de ton projet.
2. Ça sert à travailler **sans casser** la version principale.
3. Les commandes clés : `git branch`, `git checkout`, `git merge`.
4. Tu peux créer, changer, fusionner et supprimer des branches facilement.
5. Sur GitHub, les branches sont utilisées avec les **Pull Requests** pour collaborer.
6. Toujours garder un `main` **propre et stable**.

Pour toi

👉 Essaie ces étapes :

1. Crée un projet Git (`git init`).
2. Ajoute un fichier `index.html` et fais un commit.
3. Crée une branche `test`.
4. Change le fichier dans `test` et commit.
5. Reviens sur `main` et regarde : les changements ne sont pas là !
6. Fusionne la branche et observe que ton code est mis à jour.

© 2025 Mr. Lambrechts https://v2.ttrinfo.be 5 | 5