# Git town □□

> Git town est un outil pour simplifier l'utilisation de Git CLI, permettant de garder facilement toutes ses branches à jour, même lorsqu'on a une arborescence de branches complexe.

## Créer une branche

Le première commande a connaître est celle qui permet de créer une nouvelle branche : **hack**

```
git town hack 'feature/my-feature'
```

Git town va d'abord fetch les derniers changements du repository, puis mettre à jour la branche main, avant de créer la branche de feature à partir de main. De cette façon, on est sûr d'avoir une branche de feature à jour avec la dernière version de main.

## Publier sa branche

Une fois que j'ai fini et commité le travail sur ma branche, je peux publier celle-ci et ouvrir une pull request grâce à une simple commande : **propose**

```
git town propose
```

Avant de pousser la branche locale, git town va mettre à jour la branche parente locale (main par exemple) et rebase la branche de feature sur celle-ci. Ensuite, il push la branche et ouvre directement un onglet de navigateur sur la page de création d'une PR. Plus qu'à mettre à jour le titre/la description et valider !

## Créer une branche enfant

En ma plaçant sur la branche dont je veux créer un enfant, je peux le faire simplement avec **append**

```
git town append 'feature/children-feature'
```

Encore une fois, git town va mettre à jour toutes les branches parentes avant de créer la nouvelle.

## Mettre à jour sa branche

Si j'ai besoin de pull ou push des changements de ma branche, je peux le faire très simplement avec **sync**

```
git town sync
```

Git town va mettre à jour **toutes** les branches parentes, puis la branche actuelle, puis pousser la branche locale. S'il y a des modifications non commitées, il effectue un stash avant toute chose, puis un pop en dernière étape, de façon à reprendre facilement de là où on en était.

## Afficher les branches locales

La commande **branch** permet de lister les branches locales et d'afficher ça de façon hiérarchique, ce qui peut être plutôt pratique :


## Configurer les branches pérennes (perennial branches)

Par défaut, git town considère uniquement `main` comme branche permanente. Si l'on souhaite ajouter la branche `dev` comme branche permanente afin de pouvoir utiliser git town de la même façon, il faut l'indiquer explicitement pour que git town la traite comme une racine valide et ne tente pas de la rebase sur `main` :

```
git town config perennial-branches dev
```

Cette commande est à exécuter une fois par repo. Après ça, git town sait que `dev` est une branche permanente, et `sync` la mettra à jour sans chercher à la rattacher à une branche parente.

## Changer la branche parente (set-parent)

Il peut arriver qu'une branche ait été créée depuis `main` mais qu'on veuille finalement la rattacher à `dev` , ou simplement que l'on ait créé une branche de façon classique et que l'on souhaite la rattacher pour pouvoir `sync` ensuite. Dans ce cas :

```
git town set-parent dev
```

Git town va alors considérer `dev` comme branche parente, ce qui a deux effets concrets : `sync` mettra d'abord `dev` à jour avant de mettre à jour la branche courante, et `propose` ouvrira la PR vers `dev` plutôt que vers `main`.

## Autre informations

Il faut penser à configurer la stratégie désirée pour la mise à jour des branches (merge ou rebase). Il faudra indiquer quelle est la branche main sur le repo lors de la première utilisation.

En cas de conflit lors d'une opération, git town s'interrompt. Il suffit alors de résoudre les conflits et de taper la commande **git town continue**

Liens utiles : Documentation officielle | Github

---

Révision #9
Créé 10 mars 2026 09:36:50 par Nils Lapotre
Mis à jour 12 mars 2026 10:52:55 par Romain Lacits