Git - Comprendre et utiliser les tags

Un **tag** est une étiquette attachée à un commit spécifique dans Git. 👉 Les tags servent à **marquer un point important dans l'historique**, par exemple une version stable (`v1.0`), une release (`2.3.5`), ou un jalon important du projet.

5TTR 6TTR 6TQ Découverte

Qu'est-ce qu'un tag ?

Un **tag** est une étiquette attachée à un commit spécifique dans Git.

Contrairement à une branche, un tag ne bouge pas : il reste toujours associé au même commit.

👉 Les tags servent à **marquer un point important dans l'historique**, par exemple une version stable (`v1.0`), une release (`2.3.5`), ou un jalon important du projet.

Pourquoi utiliser des tags ?

- **Marquer des versions stables** : par exemple `v1.0` pour la première version distribuée d'un logiciel.
- **Déploiement et support** : savoir précisément quel commit correspond à une version livrée chez un client.
- **Navigation facile** : plutôt que de retenir un hash compliqué (`a3f9c1d`), on utilise un nom clair (`v2.0`).
- **Collaboration** : partager des repères communs dans l'équipe.

Types de tags

Tag léger (lightweight tag)

```
git tag v1.0
```

- Juste un nom collé à un commit.
- Rapide à créer, mais sans information supplémentaire.

- Utilisé pour des repères rapides ou des versions internes.

Tag annoté (annotated tag)

```
git tag -a v1.0 -m "Version 1.0 stable"
```

- Contient un message, l'auteur, la date.
- Plus riche et recommandé pour marquer des versions officielles.
- C'est le format attendu pour les releases publiques.

Git : `checkout` d'un tag

Quand tu tapes :

```
git checkout v1.0
```

Git replace ton **working directory** exactement comme il était au moment du commit associé au tag `v1.0`. 👉 Tu retrouves **les fichiers tels qu'ils étaient lors de cette version**.

Important : mode *detached HEAD*

> En faisant `checkout` sur un tag, tu n'es pas sur une branche mais sur un **commit fixe**.
> - Ton HEAD est "détaché" (*detached HEAD*).
> - Tu peux lire, tester ou compiler le projet à cet état précis.
> - Mais si tu modifies des fichiers et commits, ils ne seront pas liés à une branche → risque de perdre ce travail.
>
> Pour pouvoir modifier "un tag", tu devras créer une nouvelle branche sur base de ce tag (c'est pour plus tard).

Partager des tags sur GitHub

Par défaut, `git push` **n'envoie pas les tags**. Il faut les pousser explicitement :

- Pousser un seul tag :

```
git push origin v1.0
```

- Pousser tous les tags d'un coup :

```
git push --tags
```

⚠️ Pas recommandé !

- Cela envoie aussi des tags temporaires, de test ou obsolètes.
- Risque de polluer le dépôt distant avec des tags inutiles.

👉 Bonne pratique : **POUSSER UNIQUEMENT LES TAGS IMPORTANTS** (versions officielles).

Astuce : `--follow-tags`

Quand tu pousses une branche, tu peux ajouter l'option :

```
git push --follow-tags
```

👉 Cela pousse automatiquement les **tags annotés** associés aux commits que tu envoies.

- Utile quand tu fais une release (`git tag -a v1.2 -m "Release 1.2"`) puis que tu pousses la branche : le tag partira aussi.
- Mais cela n'envoie **pas** tous les tags du dépôt (contrairement à `--tags`).

En résumé

- Un **tag** = une étiquette fixe sur un commit.
- Deux types : **léger** (rapide) et **annoté** (recommandé pour les versions).
- Utiliser des tags pour marquer des versions stables, releases, jalons.
- Sur GitHub :
  - `git push origin v1.0` pour un tag précis.

- `git push --tags` existe mais est à éviter.
- `git push --follow-tags` pousse uniquement les tags annotés liés aux commits envoyés.

> 👉 Bonne pratique : utilise des **TAGS ANNOTÉS** pour tes versions officielles, et pousse-les un par un (ou avec `--follow-tags`) pour garder ton dépôt propre.
