Git

Cyril Barrelet

January 2026

1 Git : Qu'est-ce que c'est ?

Git est un logiciel de contrôle distribué de version. Le contrôle de version (*versionning*), est une façon de sauvegarder les changements apportés à un projet au cours du temps sans effacer les versions précédentes.

Le contrôle "distribué" veut dire que tous les dévelopeurs travailant sur dépôt de Git (*Git repository*) ont une copie du repository entier. Cela veut dire que chaque dévlopeur a accès à tous les **commit**, **branch**, dossiers et fichiers.

2 Comment utiliser Git ?

Les commandes les plus importantes et utiles sont :

- `git clone [repo url]` : Télécharger un repo GitHub.
- `git status` : Vérifier l'état du repo, voir si un fichier a été ajouté, modifié ou supprimé. Permet de vérifier quoi ajouter avant un commit.
- `git add [ file ]` : Ajouter un fichier au commit. Possibilité de faire `git add .` pour ajouter tous les fichiers modifiés, mais mieux vaut ajouter fichier par fichier pour éviter d'envoyer des fichiers non désirés.
- `git commit −m "[Commit message]"` : Créer un commit. Le message est très important, c'est le seul moyen de tracker les changements facilement.
- `git push` : Envoyer les commits vers le repo distant. A faire régulièrement pour partager son travail. Il faut faire attention d'être sur la bonne branche avant de push.
- `git pull` : Récupérer les modifications du repo distant. Cela met à jour le repo local avec les changements des autres d
- `git fetch` : Récupérer les branches et commits.
- `git branch` : Lister les branches locales. Utile pour savoir sur quelle branche on travaille.
- `git branch [branch−name]` : Créer une nouvelle branche. L'utilisation en mémoire est optimisé, c'est ok d'avoir beaucoup de branches.
- `git checkout [branch−name]` : Passer sur une autre branche. Il faut vérifier si les changements ont bien été commités avant de changer de branche pour éviter de les perdre.
- `git checkout −b [branch−name]` : Créer une nouvelle branche et bascule directement sur la nouvelle branche.
- `git merge [branch−name]` : Fusionner la branche spécifiée dans la branche courante. Tous les commits de la branche cible sont intégrés à votre branche actuelle.

Petit lien super utile : Git Cheat sheet

3 Comment créer un repo Git ?

3.1 Créer un projet localement

```
# Créer un dossier pour le projet
mkdir my_projet
cd my_projet

# Initialiser un repo Git vide
git init
```

3.2 Créer les fichiers du projet

```
# Exemple : créer un fichier Python
echo "print('Hello world')" > main.py

# Ajouter un README
echo "# Mon Projet" > README.md
```

3.3 Ajouter les fichiers au suivi Git

```
git status # Petite vérification (pas obligatoire)

# Placement des fichiers dans la staging area
git add main.py README.md

git status # Petite vérification (pas obligatoire)
```

3.4 Faire un commit

```
git commit -m "Initial commit"
```

3.5 Créer un repo distant (GitHub, GitLab, etc.

- Aller sur GitHub
- Cliquer sur **New repository**
- Nommer le repo, laisser public ou privé.
- Copier l'URL du repo

3.6 Lier le projet local au repo distant

```
git remote add origin https://github.com/username/mon_projet.git

git remote -v # Petite vérification
```

3.7 Envoyer le projet sur GitHub

```
git push -u origin main
```

3.8 Créer une nouvelle branche pour une feature

```
git checkout -b feature_xyz
```

3.9 Ajouter/modifier des fichiers

```
git add .
git commit -m "Add feature XYZ"
```

3.10 Envoyer la branche sur GitHub

```
git push origin feature_xyz
```

3.11 Fusionner dans la branche principale (main)

```
# Retour sur la branche principale
git checkout main

# Récupération des derniers changements
git pull

# Fusion de la branche feature_xyz dans la branche courante (ici main)
git merge feature_xyz

git push
```

3.12 Conseils pratiques

4 Comment gérer les conflits ?

- Toujours pull avant de push pour éviter des conflits.
- Committez souvent avec des messages clairs.
- Ne mettez jamais vos mots de passe dans le code (Clé API, etc.).
- ORganisez vos branches : `main` pour le code stable, `dev` pour le code en dévloppement, `feature_xyz` pour chaque fonctionnalité.

5 Pour aller plus loin

- Comprendre Git en 5 minutes (5 mn video en français) : Youtube
- Git and GitHub Crash Course (50 mn video en anglais): Youtube
- Git Tutorial for Beginners (1 hour video en anglais) : Youtube