
git

git

Dans ce cours, le code et les exercices seront diffusés via git. Git est un logiciel de gestion de version (VCS).
Il apporte les fonctionnalités suivantes lorsqu'un groupe travaille sur un même projet :

1. Synchroniser le travail entre les personnes qui programment
2. Enregistrer l'évolution du code au cours du temps
3. Stocker le code en lieu sûr

Git stocke ses données dans le répertoire de votre projet, dans le dossier *.git*, de manière indépendante pour chaque projet.
Git se contrôle par ligne de commande, le contenu de *.git* n'est jamais modifié manuellement.

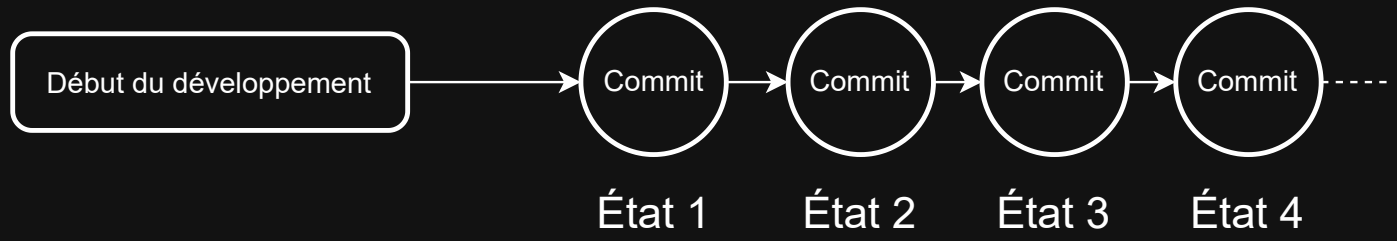
Références :

- Site officiel : https://git-scm.com/
- Téléchargement : https://git-scm.com/downloads
- Livre : https://git-scm.com/book/en/v2

git

Les commits

La notion fondamentale dans git est le *commit*.
Il s'agit d'un instantané de votre projet, un "point de sauvegarde" qui enregistre son état.
Git permet de revenir, de comparer, de mélanger différents commits.
Ce qui se passe entre les commit n'est pas sauvegardé.



git

Les commits

Dans un projet, pour indiquer à git qu'un fichier doit être *tracké* (pris en compte), la commande est :

```
git add <chemin_vers_le_fichier>
```

Pour faire un commit, la commande est :

```
git commit -am "Message"
```

Git créé alors un nouveau commit

- L'option *-a* indique de prendre tous les fichiers trackés modifiés
- L'option *-m* indique le message du commit. Il s'agit d'un texte décrivant ce qui a été modifié dans ce commit.

Si l'option *-m* n'est pas précisée, git ouvrira un éditeur de texte pour entrer le message de commit

git

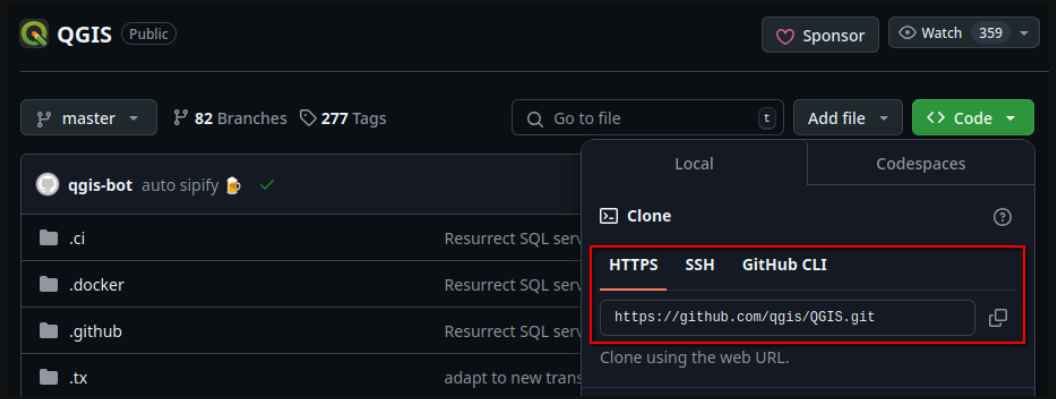
Le clonage

Pour travailler sur un projet existant, il faut d'abord le copier en local sur sa machine. On appelle cela le *clonage*. La commande est :

```
git clone <url_du_projet_distant>
```

Git créé alors un dossier contenant tout le projet.

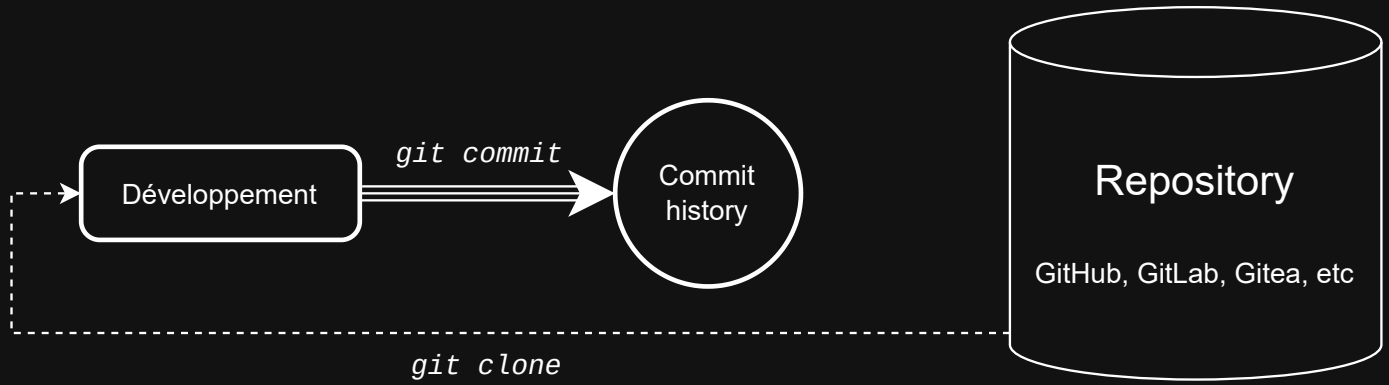
Les projets sont stockés dans un dépôt (*repository*) distant. Il existe plusieurs fournisseurs de stockage : GitHub, GitLab, ou des stockages auto-hébergés tels que Gitea.



git

Workflow de base

Le workflow est donc le suivant :



git

Push & Pull

Après plusieurs commits, il est possible d'envoyer (*push*) les changements au repository distant:

```
git push
```

Cela enverra tous les commits sur le repository distant. Ce qui n'a pas été commit ne sera pas pris en compte. Pour faire un `git push` , il faut avoir le droit de push sur le repository distant.

Pour récupérer (*pull*) la dernière version du code en ligne, si celui-ci a été modifié par quelqu'un d'autre:

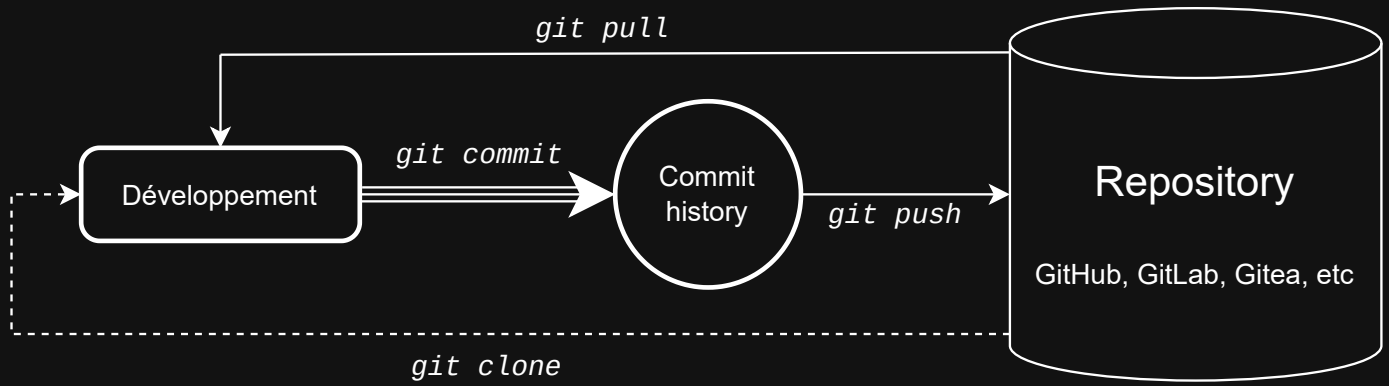
```
git pull
```

Faites toujours un commit de vos changements avant un pull

git

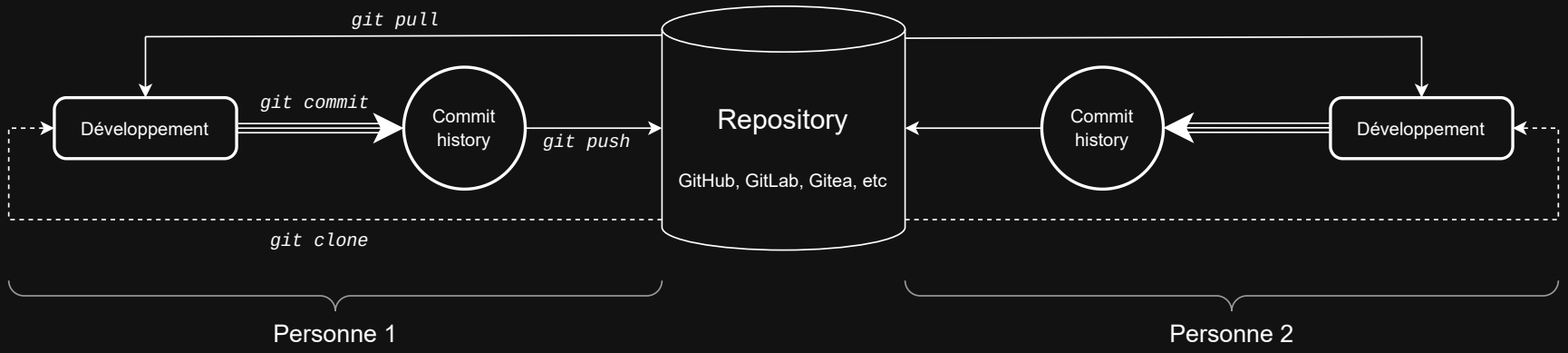
clone & commit & push & pull

Le workflow est donc le suivant :



git

En groupe



git

Les conflits

Supposons :

1. Alice et Bob clonent leur projet, chacun de son côté
2. Alice écrit `div{ color:blue; }`, puis fait un commit
3. Bob écrit `div{ color:green; }`, puis fait un commit
4. Bob push son code avec `git push`
5. Alice push son code avec `git push`

Que se passe-t-il chez Alice ?

```
! [rejected]        main → main (fetch first)
error: failed to push some refs to 'https://github.com/ ... '
```

Alice ne peut pas push, car elle ne possède pas la dernière version du code

git

Les conflits

La solution : Alice doit d'abord faire un `git pull` pour récupérer la dernière version du code. A ce moment, git va mélanger (*merge*) les deux versions du code, celle d'Alice et celle de Bob.

Deux cas peuvent se produire :

1. Les modifications ne sont pas contradictoires, git parvient à faire automatiquement le merge
   - Alice possède alors une version du code mélangée
   - Il lui suffit de faire un `git commit` et un `git push` pour push sa nouvelle version.

git

Les conflits

La solution : Alice doit d'abord faire un `git pull` pour récupérer la dernière version du code. A ce moment, git va mélanger (merge) les deux versions du code, celle d'Alice et celle de Bob.

Deux cas peuvent se produire :

2. Il y a conflit

Lors du pull Alice recevra le message :

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

Les conflits sont indiqués sous la forme :

```
<<<<<<< HEAD
div{ color:blue; }
=======
div{ color: green; }
>>>>>>> b6eeeaef7d4c17e8b7ad2b90968e2d17720ba319
```

Alice devra alors résoudre les conflits manuellement, puis `git commit` et `git push`

git

Astuces

La commande `git log` permet de voir l'historique des commits

La commande `git status` permet de voir la liste des fichiers modifiés depuis le dernier commit

Si un fichier nommé `.gitignore` est placé à la racine d'un projet, les dossier et fichiers listés à l'intérieur ne seront jamais trackés. Cela est très pratique pour directement exclure des fichiers et des dossiers entiers qu'on ne souhaite pas synchroniser. Il est possible d'utiliser des expressions génériques. Par exemple `*.txt` empêchera tous les fichiers avec l'extension `.txt` d'être trackés.

Ce chapitre est une introduction à git, il existe de nombreuses autres fonctions qui n'ont pas été mentionnées et que vous découvriez au travers des exercices et de la pratique.