GIT

LOGICIEL DE GESTION DE VERSIONS

Yann Rotella

IN100, Université de Versailles Saint Quentin en Yvelines

Décembre 2022

Travailler en groupe

On veut éviter les problèmes suivant :

- Modifications simultanées ;
- Bugs : "hier ça marchait, maintenant j'ai des bugs partout" ;
- Travailler hors connexion ou avec une mauvaise connexion.

Travailler en groupe

On veut éviter les problèmes suivant :

- Modifications simultanées ;
- Bugs : "hier ça marchait, maintenant j'ai des bugs partout" ;
- Travailler hors connexion ou avec une mauvaise connexion.

Une bonne solution :

Un logiciel de gestion de versions

Qu'est ce que ça fait ?

- On garde en mémoire **toutes** les anciennes versions (historique).
- **Qui** a modifié **quoi** et **pourquoi** ?
- Possibilité de **fusionner** intelligemment un travail simultané.

Qu'est ce que ça fait ?

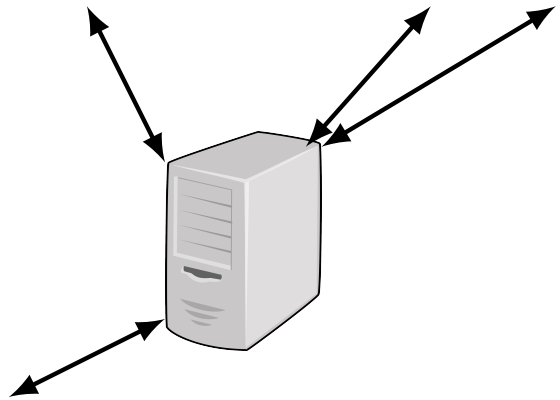
- On garde en mémoire **toutes** les anciennes versions (historique).
- **Qui** a modifié **quoi** et **pourquoi** ?
- Possibilité de **fusionner** intelligemment un travail simultané.

On utilisera donc Git pour travailler à plusieurs et/ou un projet long.

CHOIX DU LOGICIEL DE GESTION DE VERSIONS

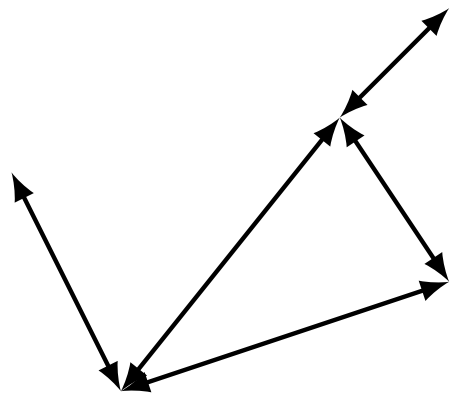
Centralisé (SVN) :

- Un serveur conserve tout et chaque utilisateur s'y connecte.

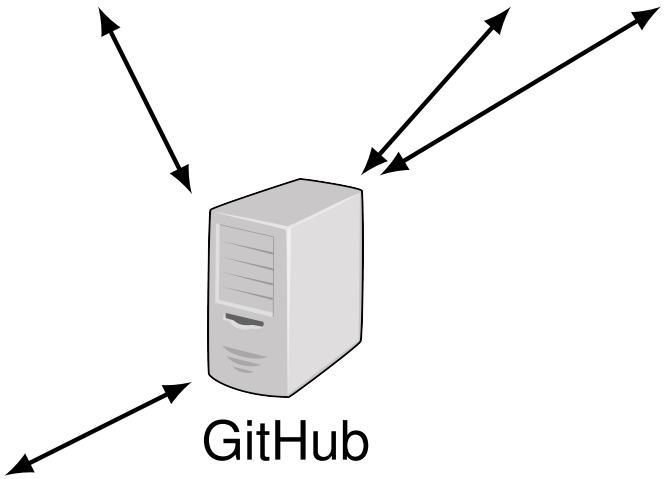


Décentralisé (Mercurial, Git) :

- Chacun possède l'historique
- Fonctionnement pair à pair



CHOIX DU LOGICIEL DE GESTION DE VERSIONS



- ✓ Rapide
- ✓ Branches (plus tard)
- ✗ Un peu complexe
- ✗ Pas d'interface graphique
- ✓✓ GitHub
- ✓✓ Grande communauté

Installation de Git

Normalement déjà fait pour récupérer les TD et le cours

- **Linux** :

```
$ sudo apt-get install git-core gitk
```

- **Windows** : msysgit.github.io
- **MAC OS** : git-osx-installer (par exemple)

Installation de Git

Normalement déjà fait pour récupérer les TD et le cours

- **Linux** :

```
$ sudo apt-get install git-core gitk
```

- **Windows** : msysgit.github.io
- **MAC OS** : git-osx-installer (par exemple)

Dans tous les cas, on utilisera la console. C'est l'occasion de se familiariser avec le bash !

CONFIGURATION

```
$ git config --global color.diff auto
$ git config --global color.status auto
$ git config --global color.branch auto
$ git config --global user.name "mon_pseudo"
$ git config --global user.email my_email
```

NAVIGUER DANS LE TERMINAL

```
$ cd
$ ls
$ ls -a
$ mkdir
$ cd ..
$ git init
```

Cloner un dépôt Git

Dans le dossier où on veut travailler :

```
$ git clone https://github.com/depot_git
```

Cette solution demandera le nom d'utilisateur/ mot de passe à chaque fois.

Cloner un dépôt Git

Dans le dossier où on veut travailler :

```
$ git clone https://github.com/depot_git
```

Cette solution demandera le nom d'utilisateur/ mot de passe à chaque fois.

Utiliser SSH (avancé) :

```
$ git clone git@github.com:depot_git
```

Cette solution nécessite de créer des clefs SSH (RSA).

.git

fichier1.py

fichier2.py

Les informations du dépôt

Connaître les fichiers modifiés :

```
$ git status
```

Connaître les modifications du ou des fichiers :

```
$ git diff nom_fichier
$ git diff
```

LES COMMITS

- Les commits servent à modifier les fichiers au sein du dépôt git, afin de garder l'historique.

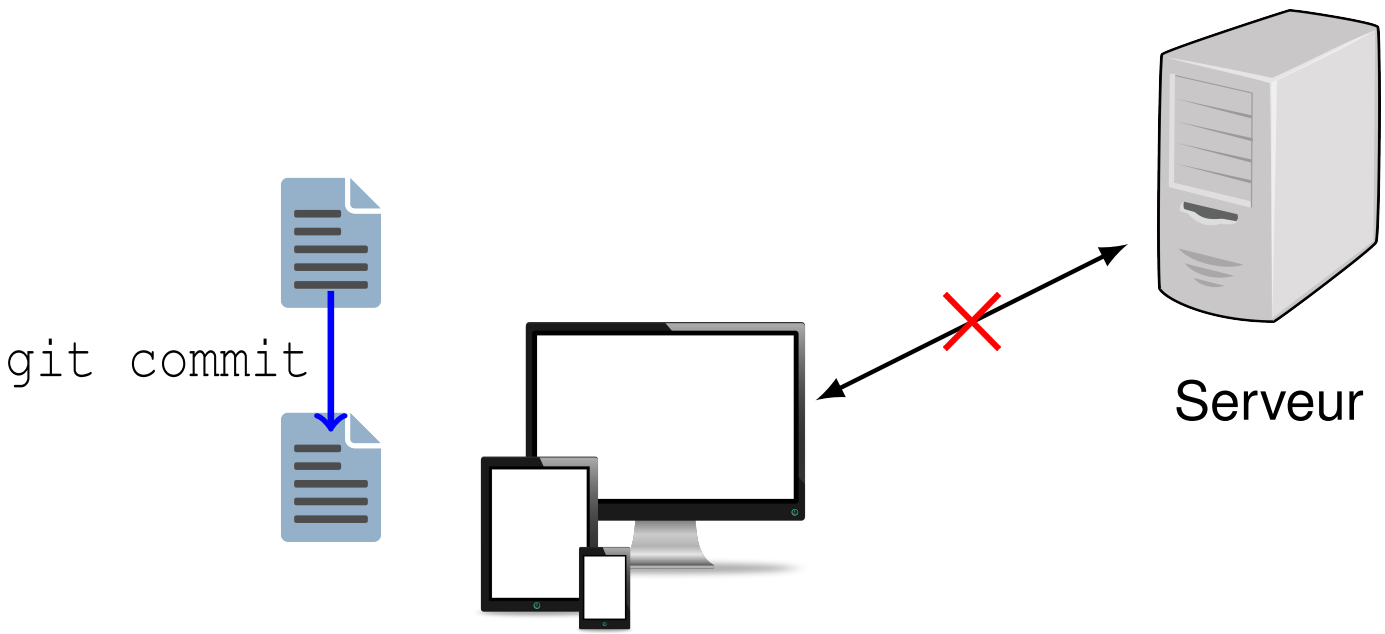
```
$ git add fichier1.py fichier2.py
$ git commit -a -m 'mon message'
$ git commit fichier1.py -m 'mom message'
```

Les commits

- Les commits servent à modifier les fichiers au sein du dépôt git, afin de garder l'historique.

```
$ git add fichier1.py fichier2.py
$ git commit -a -m 'mon message'
$ git commit fichier1.py -m 'mom message'
```

- Les commits modifient **localement** le dépôt.



NAVIGUER DANS L'HISTORIQUE

```
$ git log
$ git log --stat
$ git log -p
```

À quoi ressemblent les log (stat exemple)

NAVIGUER DANS L'HISTORIQUE

```
$ git log
$ git log --stat
$ git log -p
```

À quoi ressemblent les log (stat exemple)

```
commit cd7c6ddbf84d02a8e2ae6c47f9c10665de977ee9
Author: Yann Rotella <yann.rotella@uvsq.fr>
Date:   Sun Oct 25 19:07:56 2020 +0100

    mon message

 presentations/fichier.py | 31 ++++++++++---
 1 file changed, 23 insertions(+),
 8 deletions(-)
```

NAVIGUER DANS L'HISTORIQUE

```
$ git log
$ git log --stat
$ git log -p
```

À quoi ressemblent les log (stat exemple)

```
commit cd7c6ddbf84d02a8e2ae6c47f9c10665de977ee9
Author: Yann Rotella <yann.rotella@uvsq.fr>
Date:   Sun Oct 25 19:07:56 2020 +0100

    mon message

 presentations/fichier.py | 31 ++++++++++---
 1 file changed, 23 insertions(+),
 8 deletions(-)
```

Flèches directionnelles + PageUp / PageDown

ANNULER LES BÊTISES

- Annuler le dernier commit, mais ne modifie pas le fichier (soft) :

```
$ git reset HEAD^
```

- Annuler et effacer les modifications (hard) :

```
$ git reset --hard HEAD^
```

- Revenir au dernier commit :

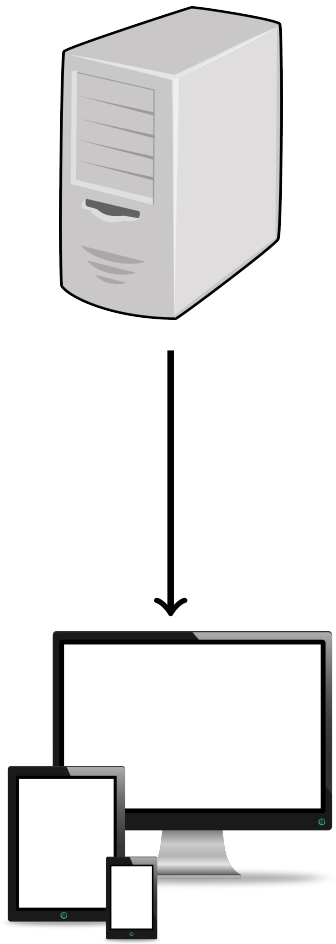
```
$ git checkout
```

- Annuler le `git add` :

```
$ git reset HEAD -- fichier_a_supprimer
```

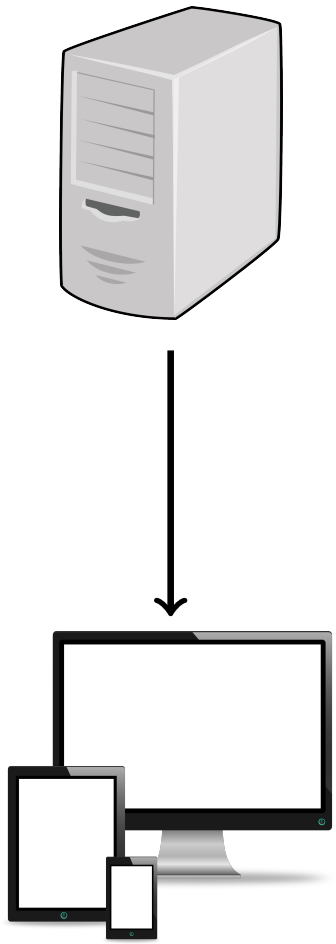
TRAVAILLER AVEC LES AUTRES

```
$ git pull
```



Travailler avec les autres

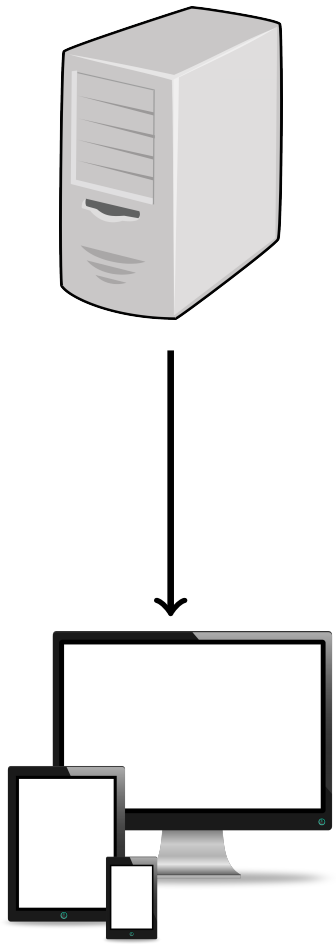
```
$ git pull
```



1. Pas de changement depuis la dernière fois : aucun soucis (Fast-Forward).

TRAVAILLER AVEC LES AUTRES

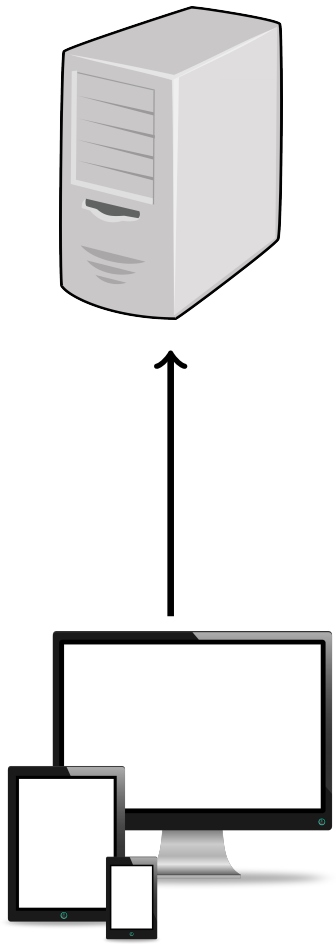
```
$ git pull
```



1. Pas de changement depuis la dernière fois : aucun soucis (Fast-Forward).
2. Vous avez fait des commits : les changements sont fusionnés intelligemment ou **il y a des conflits**, indiqués par «««< (cf TD)

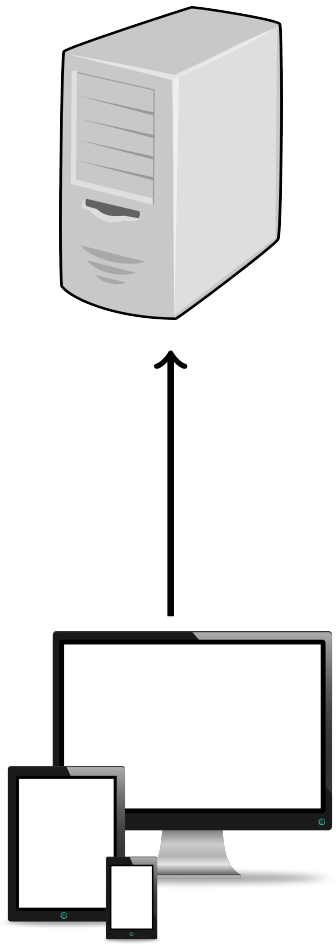
TRAVAILLER AVEC LES AUTRES

```
$ git push
```



TRAVAILLER AVEC LES AUTRES

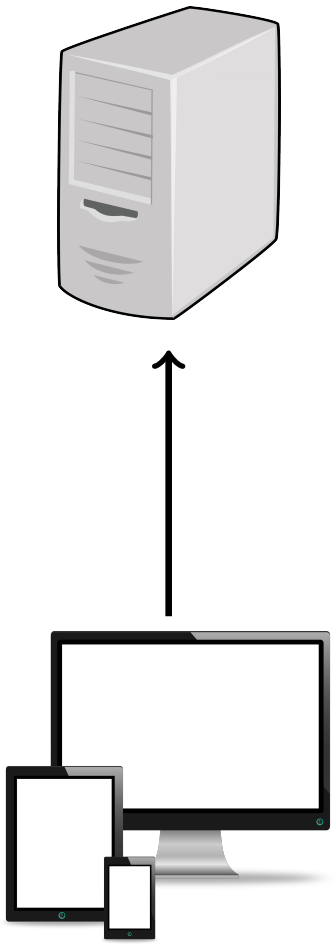
```
$ git push
```



1. Toujours de type Fast-Forward

Travailler avec les autres

```
$ git push
```



1. Toujours de type Fast-Forward
2. Personne ne doit avoir fait push, **on doit donc toujours pull avant de push**

ANNULER LES COMMITS QUI ONT ÉTÉ PUSH

1. Récupérer l'Identifiant du commit :

```
$ git log
```

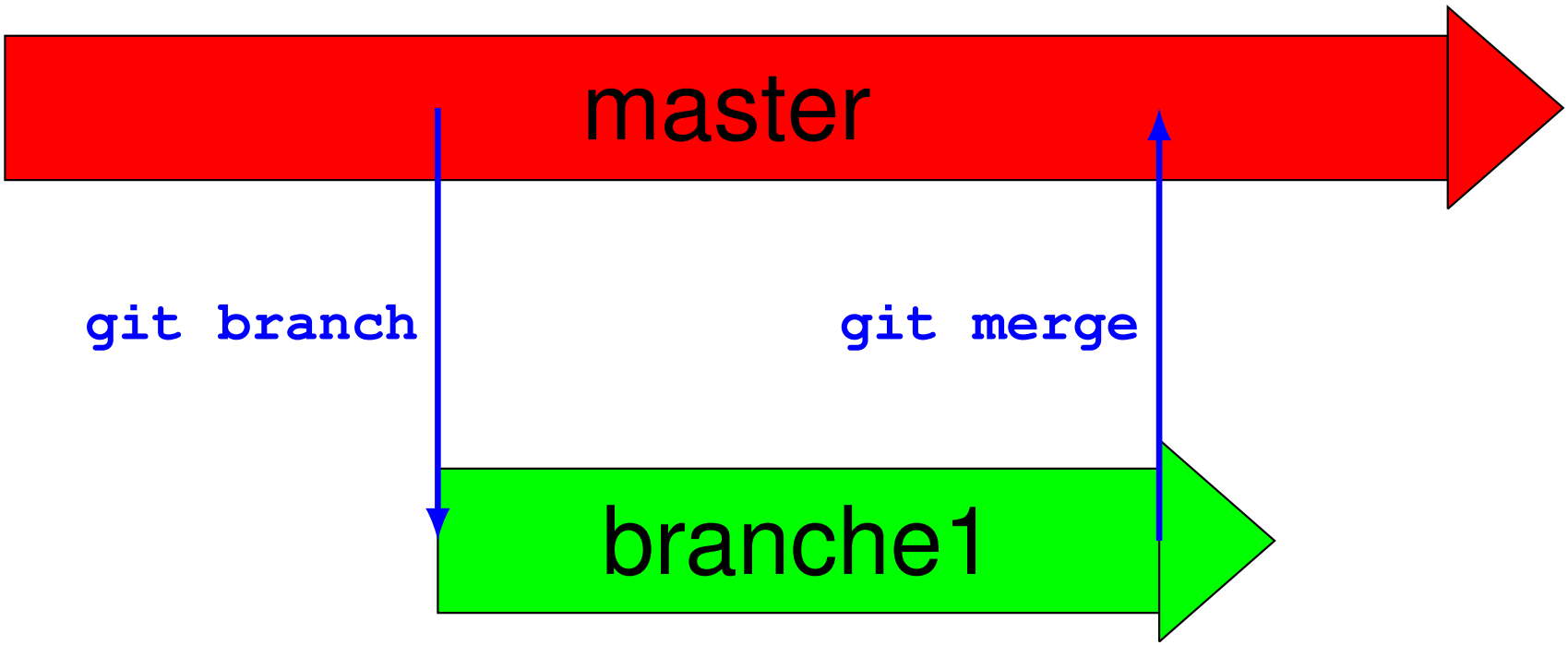
2. Annuler le commit :

```
$ git revert ID
```

pour l'Identifiant, il suffit de taper les 5-6 premiers caractères, et assurer l'unicité sur l'ensemble des commits.

Les branches (Git avancé)

On utilise les branches pour des sous-projets qui vont être longs.



Les branches (Git avancé)

- Créer la branche :

```
$ git branch nom_branche
```

- Travailler sur la branche :

```
$ git checkout nom_branche
```

- Fusionner la branche avec master :

```
$ git merge nom_branche
```

- Supprimer la branche :

```
$ git branch -d nom_branch
```

Autres

Si on veut pull en urgence, stash permet de sauvegarder les modifications locales non-commitées.

```
$ git stash
```

Autres

Si on veut pull en urgence, stash permet de sauvegarder les modifications locales non-commitées.

```
$ git stash
```

Très avancé : la gestion des branches vues précédemment peut devenir complexe, mais permet de couper un très gros projet en plein de branches, et d'intégrer les branches seulement quand le code est stable.

Les bonnes pratiques

- On commit fréquemment, mais uniquement du code qui **fonctionnne**.

Les bonnes pratiques

- On commit fréquemment, mais uniquement du code qui **fonctionnne**.
- On teste donc son code **avant** de commit.

Les bonnes pratiques

- On commit fréquemment, mais uniquement du code qui **fonctionnne**.
- On teste donc son code **avant** de commit.
- On push rarement (à la fin de la journée / demi-journée de travail).

Les bonnes pratiques

- On commit fréquemment, mais uniquement du code qui **fonctionnne**.
- On teste donc son code **avant** de commit.
- On push rarement (à la fin de la journée / demi-journée de travail).
- **On utilise Git !**