

Introduction à Git et GitHub

Contenu inspiré de <https://github.com/Pontelneptique/cours-git> (© Thibault Clérice, publié sous [licence CC BY](#))



Contenu

- **Git**

- C'est quoi?
- Les bases
- Les branches
- Travailler avec un serveur distant

- **GitHub**

- C'est quoi?
- Les Pull Requests

Git >> C'est quoi?

Git, c'est quoi?

- **Git = Logiciel de gestion de versions**
- **Logiciel libre et gratuit**
- **Permet de**
 - Conserver l'historique des modifications aux fichiers d'un projet
 - Revenir en arrière (annuler des modifications)
 - Gérer en parallèle plusieurs versions d'un projet et les fusionner (idéal pour la collaboration)



Historique

- **Créé en 2005 par Linus Torvalds (créateur du noyau Linux)**
- **Présentement à la version 2**



Photo par Krd et Von Sprat sur Wikimedia
(Licence CC BY-SA 4.0)

Quelques logiciels concurrents

- **SVN**
- **CVS**
- **Mercurial**

Git >> Les bases

Utilisation

- **Traditionnellement utilisé en ligne de commande**
- **Il existe aussi des outils graphiques, ex:**
 - GitKraken
 - GitHub Desktop
 - SourceTree

Les dépôts

- **Un projet Git est appelé un **dépôt** (*repository* en anglais, *repo* pour les intimes)**
- **Concrètement, un dépôt est un répertoire**
 - contenant des fichiers (typiquement de code source)
 - dont l'historique des modifications est géré par Git

Créer un dépôt

La commande « **git init** » permet d'initialiser un dépôt dans un répertoire (vide ou non)

```
plbrault@PORTABLE: ~/exemple-git
plbrault@PORTABLE: ~/exemple-git 68x7
→ exemple-git git init
Initialized empty Git repository in /home/plbrault/exemple-git/.git/
→ exemple-git git:(master) ls -a
.  ..  .git
→ exemple-git git:(master) █
```

Les commits

- **Pour qu'une modification soit « sauvegardée » par Git, elle doit être ajoutée à l'historique par un « **commit** »**
- **Un commit...**
 - Est identifié par un message composé par l'utilisateur (**commit message**)
 - Peut comporter des modifications à plusieurs fichiers

États des fichiers

- **On distingue trois « états » des fichiers**

- un état de travail: le fichier a subi des modifications, mais n'a pas encore été ajouté à un futur commit
- un état de futur enregistrement (**staging**): le fichier a été identifié comme devant être ajouté au prochain commit
- un état archivé : toutes les modifications apportées au fichier ont été « commitées »

État de travail	Staging	État archivé
Nouvelles modifications non ajoutées à un futur commit	Nouvelles modifications sélectionnées pour le prochain commit	Toutes les modifications ont été « commitées »

Ajouter une modification

La commande « **git add** » permet d'ajouter un fichier au prochain commit (donc de mettre le fichier en staging)

```
plbrault@PORTABLE: ~/exemple-git
plbrault@PORTABLE: ~/exemple-git 68x7
→ exemple-git git:(master) echo "Hello World" > nouveauFichier.txt
→ exemple-git git:(master) X git add nouveauFichier.txt
→ exemple-git git:(master) X
```

Consulter l'état du dépôt

La commande « **git status** » affiche l'état des modifications non « **commitées** » (« **stagées** » ou non)

```
plbrault@PORTABLE: ~/exemple-git
plbrault@PORTABLE: ~/exemple-git 78x19
→ exemple-git git:(master) echo "Hello World" > nouveauFichier.txt
→ exemple-git git:(master) X git add nouveauFichier.txt
→ exemple-git git:(master) X echo "Bonjour le monde" > nouveauFichier2.txt
→ exemple-git git:(master) X git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   nouveauFichier.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        nouveauFichier2.txt

→ exemple-git git:(master) X █
```

Ajouter un commit

La commande « **git commit** » permet de créer un nouveau commit. Il faut utiliser l'option **-m** pour indiquer le message à associer au commit.

```
plbrault@PORTABLE: ~/exemple-git
plbrault@PORTABLE: ~/exemple-git 81x14
→ exemple-git git:(master) X git commit -m "Ajout du nouveau fichier"
[master (root-commit) e7c58c7] Ajout du nouveau fichier
 1 file changed, 1 insertion(+)
 create mode 100644 nouveauFichier.txt
→ exemple-git git:(master) X git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        nouveauFichier2.txt

nothing added to commit but untracked files present (use "git add" to track)
→ exemple-git git:(master) X █
```

Ajout automatique des fichiers modifiés (git commit -a)

```
plbrault@PORTABLE: ~/exemple-git
plbrault@PORTABLE: ~/exemple-git 81x29
→ exemple-git git:(master) X
→ exemple-git git:(master) X echo '!' >> nouveauFichier.txt
→ exemple-git git:(master) X git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

       modified:   nouveauFichier.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

       nouveauFichier2.txt

no changes added to commit (use "git add" and/or "git commit -a")
→ exemple-git git:(master) X git commit -am "Ajout d'un point d'exclamation à la
  fin de nouveauFichier"
[master 025776d] Ajout d'un point d'exclamation à la fin de nouveauFichier
 1 file changed, 1 insertion(+)
→ exemple-git git:(master) X git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

       nouveauFichier2.txt

nothing added to commit but untracked files present (use "git add" to track)
→ exemple-git git:(master) X
```


Consulter l'historique du dépôt

La commande « **git log** » permet d'afficher l'historique du dépôt

```
plbrault@PORTABLE: ~/exemple-git
plbrault@PORTABLE: ~/exemple-git 81x2
→ exemple-git git:(master) X git log
```

```
git log
git log 81x15

commit 025776d27ef7cdea10a6041f5b5bf9f913a7aef2 (HEAD -> master)
Author: Pier-Luc Brault <plbrault@gmail.com>
Date:   Wed Jan 15 21:19:17 2020 -0500

    Ajout d'un point d'exclamation à la fin de nouveauFichier

commit e7c58c7e7ce2646bcfa184af20f243465b3e0050
Author: Pier-Luc Brault <plbrault@gmail.com>
Date:   Wed Jan 15 21:14:29 2020 -0500

    Ajout du nouveau fichier
(END)
```

Tableau-synthèse des commandes de base

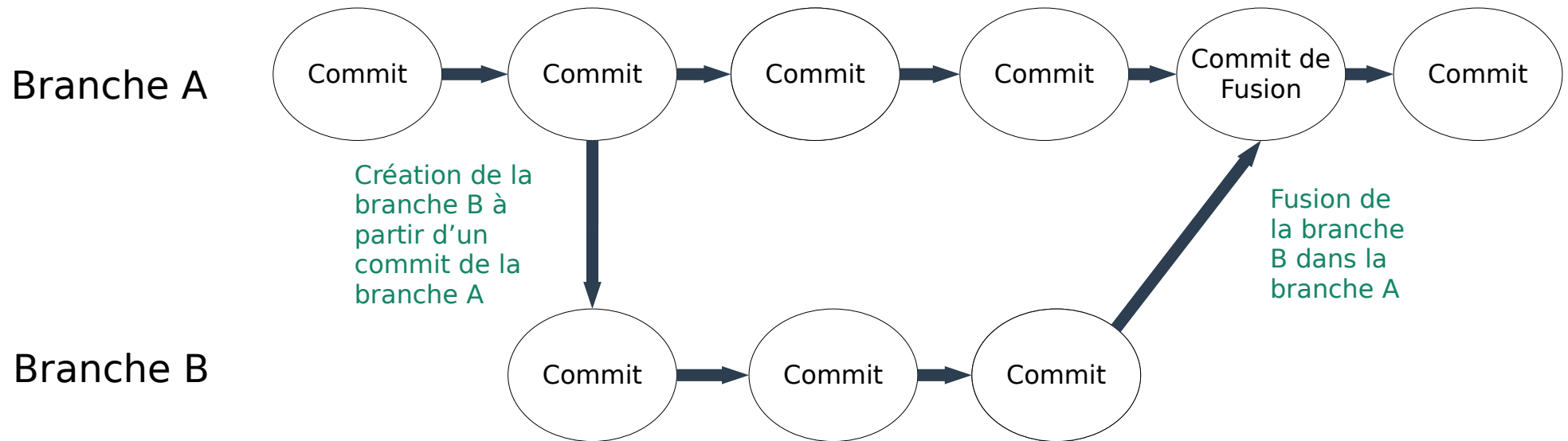
Description	Commande
Créer un dépôt	git init
Ajouter une modification au prochain commit	git add <NOM DU FICHIER> (Astuce: utiliser git add --all pour ajouter toutes les modifications non « stagées »)
Consulter l'état du dépôt	git status
Ajouter un commit	git commit -m "MESSAGE" L'option -a permet d'ajouter toutes les nouvelles modifications automatiquement (mais pas les nouveaux fichiers)
Consulter l'historique du dépôt	git log

Git >> Les branches

Les branches

- Les **branches** permettent de gérer en parallèle plusieurs versions d'un dépôt
- Créer une branche, c'est donc un peu comme utiliser la fonction « Sauvegarder sous » d'un logiciel
- Sauf qu'on peut plus facilement fusionner deux versions (branches) par la suite!

Exemple visuel

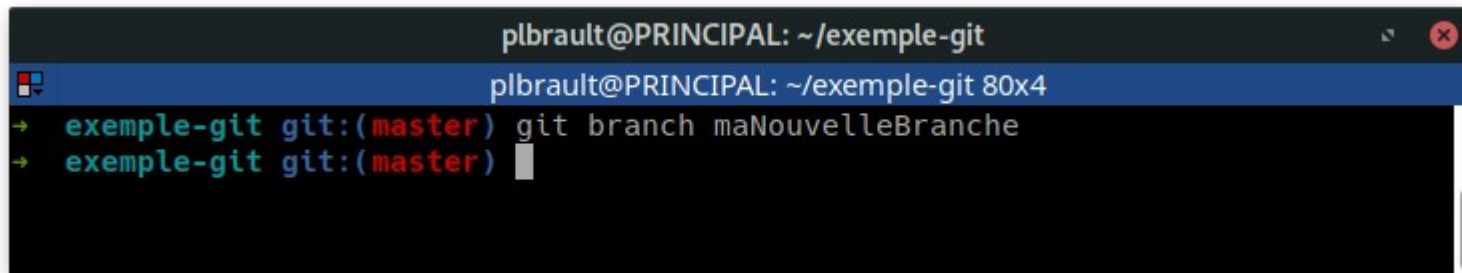


Branche principale et utilité des branches

- **La branche par défaut s'appelle **master****
 - Tendance depuis 2020 à renommer cette branche en « **main** » pour éviter la référence à l'esclavage
- **Les branches permettent de travailler sur différents problèmes en parallèle**
 - Possibilité de travailler sur des problèmes différents en même temps et de changer de tâche rapidement
 - Ex:
 - La branche **main** qui contient les fonctionnalités stables du logiciel
 - La branche **fonctionnalite1** pour travailler sur le développement de la fonctionnalité 1
 - La branche **bogue1** pour travailler sur la résolution du bogue 1

Créer une branche

La commande « **git branch** » permet de créer une nouvelle branche à partir du dernier commit de la branche courante



```
plbrault@PRINCIPAL: ~/exemple-git
plbrault@PRINCIPAL: ~/exemple-git 80x4
→ exemple-git git:(master) git branch maNouvelleBranche
→ exemple-git git:(master) █
```

Se déplacer dans une branche

La commande « **git checkout** » permet de se déplacer dans une branche existante

```
plbrault@PORTABLE: ~/exemple-git
plbrault@PORTABLE: ~/exemple-git 64x8
→ exemple-git git:(master) git branch maNouvelleBranche
→ exemple-git git:(master) git checkout maNouvelleBranche
Switched to branch 'maNouvelleBranche'
→ exemple-git git:(maNouvelleBranche) █
```


Fusionner une branche

La commande « **git merge** » permet de fusionner une autre branche dans la branche courante

```
plbrault@PORTABLE: ~/exemple-git
plbrault@PORTABLE: ~/exemple-git 56x7
→ exemple-git git:(master) git merge maNouvelleBranche
Updating 025776d..cba12ac
Fast-forward
 nouveauFichier.txt | 1 +
 1 file changed, 1 insertion(+)
→ exemple-git git:(master) █
```

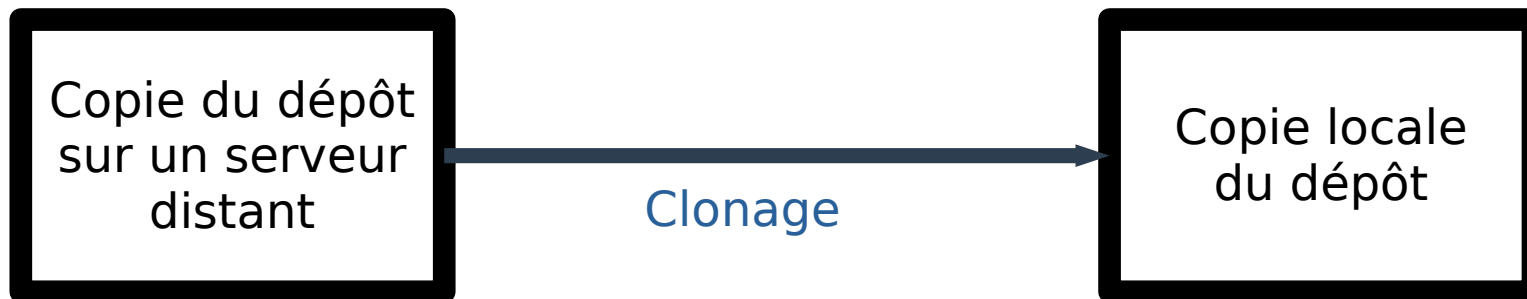
Git >> Travailler avec un serveur distant

Travailler avec un serveur distant

- **Git est décentralisé**
 - C'est-à-dire que tous les utilisateurs contribuant à un dépôt possèdent une copie complète de celui-ci sur leur poste, avec tout l'historique des commits
- **Pour que plusieurs utilisateurs puissent collaborer ensemble, il faut qu'un serveur distant maintienne lui aussi une copie du dépôt**

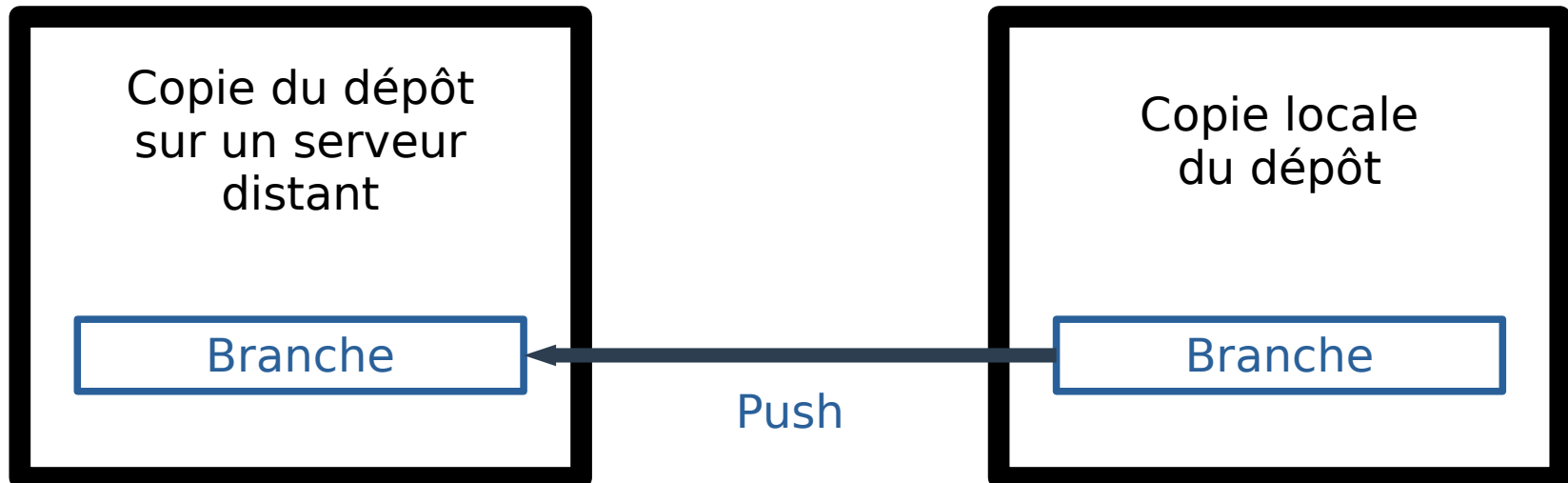
Récupérer un dépôt distant

- Pour contribuer à un dépôt, on doit d'abord **cloner** celui-ci depuis le serveur qui l'héberge
- On travaille ensuite localement en créant des branches et des commits à volonté



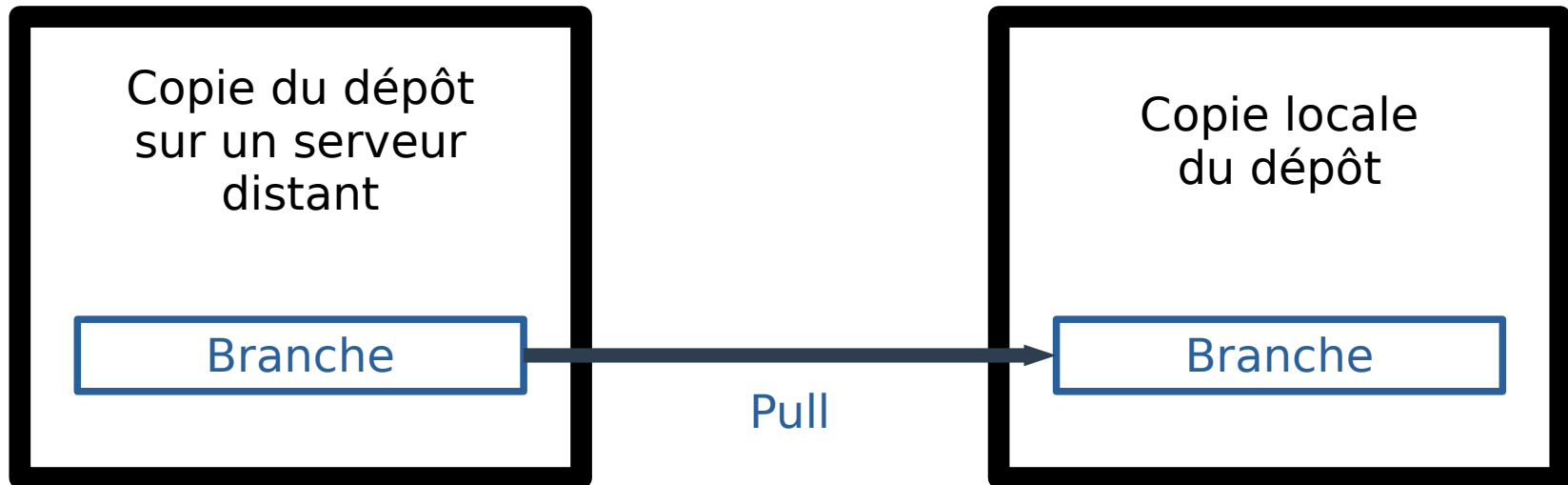
Mettre à jour une branche distante

- Pour partager ses modifications, on doit effectuer un **push** d'une branche locale vers le serveur distant



Mettre à jour une branche locale

- Pour récupérer les modifications effectuées par autrui, on doit effectuer un **pull** d'une branche du serveur distant vers le dépôt local



Commandes

- **Les commandes permettant d'effectuer ces opérations seront couvertes dans l'exercice**

GitHub, c'est quoi?

GitHub, c'est quoi?

- Hébergeur de dépôts Git (serveur distant)
- Interface Web permettant de visualiser les dépôts
- Permet de gérer l'accès aux dépôts (utilisateurs autorisés en lecture et en écriture)
- Gestion de bogues et de tickets (GitHub Issues)
- Gestion de fusion de branches et de **revue de code** (*code review*)



Principaux concurrents

- **GitLab**
- **Bitbucket**

GitHub >> Les Pull Requests

Les Pull Requests

- Les bonnes pratiques veulent qu'on ne travaille jamais directement sur la branche principale
- Lorsqu'on est prêt à fusionner notre branche dans la branche principale, on effectue un **Pull Request** via GitHub

Les Pull Requests

- **Les autres contributeurs du dépôt sont alors en mesure d'effectuer une revue de code**
- **Une fois le *Pull Request* approuvé, la branche correspondante peut être fusionnée directement via l'interface Web**

Fin de la présentation

Des questions?



Photo par Emily Morter sur Unsplash