Git : mode d'emploi pour un usage seul, sans dépôt distant, avec une à plusieurs branches

Florian Legendre

Université de Poitiers

Année 2020 - 2021

Table of Contents

Université de Poitiers

Table of Contents

Commandes : git branch

Il est important de connaître la branche sur laquelle on travaille. Pour cela, il suffit d'entrer dans le terminal "git branch" sans argument :

```
crex@crex:~/projects/avlTreesL3Project$ git branch
  dev
* main
```

Listing 1 – Exemple d'affichage de la commande "git branch"

Commandes : git branch [-d] <nom_branche>

Pour créer une branche on entre "git branch" suivi du nom de la branche qu'on souhaite donner. L'option -d permet au contraire de supprimer la branche désigné par le nom donné en paramètre :

```
crex@crex:~/projects/avlTreesL3Project$ git branch
  dev
* main
crex@crex:~/projects/avlTreesL3Project$ git branch exp
crex@crex:~/projects/avlTreesL3Project$ git branch
  dev
  exp
* main
crex@crex:~/projects/avlTreesL3Project$ git branch -d exp
Deleted branch exp (was 9985cdb).
crex@crex:~/projects/avlTreesL3Project$ git branch
  dev
* main
```

Listing 2 – Exemple de création et de suppression de branche

Commandes : git checkout <nom_branche>

Pour changer de branch on entre "git checkout" suivi du nom de la branche :

```
crex@crex:~/projects/avlTreesL3Project$ git branch
  dev
* main
crex@crex:~/projects/avlTreesL3Project$ git checkout dev
Switched to branch 'dev'
crex@crex:~/projects/avlTreesL3Project$ git branch
* dev
  main
```

Listing 3 – Exemple de création et de suppression de branche

Remarque : Les fichiers sont automatiquement changés pour correspondre à l'état de la version la plus récente de la branche sur laquelle on bascule.

Commandes : git checkout <nom_branche>

ATTENTION : Le changement de branche vous sera refusé si des modifications ont été détectées mais n'ont pas encore été "commitées". Deux solutions s'offrent alors à vous : 1) commiter vos changements puis changer de branche ou 2) utiliser la remise dont nous détaillerons le fonctionnement ultérieurement.

Commandes : git merge <nom_branche>

La commande "git merge" suivi du nom de la branche permet d'incorporer les changements d'une branche dans une autre branche.

ATTENTION : git merge <nom_branche> incorpore les changements de la branche désignée **dans la branche courante**.

Ce qui signifie, par exemple dans le cas où vous voulez incorporer les changements d'une branche de développement dans une branche "main" ou "master" que vous devez d'abord changer de branche pour aller sur la branche "main" ou "master" puis exécuter la commande.

Table of Contents

Intérêt de la remise

La remise permet de revenir au dernier commit tout en sauvegardant votre travail en cours. Ceci est très utile dans certains cas, par exemple :

- Quand on veut changer de branche rapidement sans "commiter" des modifications en cours
- Quand on a commencé un travail mais qu'on s'est trompé de branche
- Et probablement d'autres cas auxquels je n'ai pas encore pensé...

Commandes : git stash [pop/list/drop]

La commande "git stash" fonctionne de la manière suivante :

- git stash + aucun argument : Revient au dernier commit en sauvegardant le travail en cours
- git stash list : Liste les différents WIP ("Work In Progress") sauvegardés dans la remise
- git stash apply <numéro> : Applique les modifications de <numéro> de la remise
- git stash pop : Applique les modifications du dernier stash de la remise puis le supprime
- git stash drop <numéro> : Retire le stash <numéro> de la remise

Commandes : git stash [pop/list/drop]

```
crex@crex:~/projects/test$ git st
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working
     directory)
     modified:   test.txt

crex@crex:~/projects/test$ git stash
Saved working directory and index state WIP on master: 1a0b2eb Dé
     but du suivi de test.txt

crex@crex:~/projects/test$ git stash list
stash@{0}: WIP on master: 1a0b2eb Début du suivi de test.txt

crex@crex:~/projects/test$ git stash apply 0
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working
     directory)
     modified:   test.txt
```

Listing 4 – Exemple de fonctionnement du stash

Table of Contents

La boucle de travail local multibranches