

Gestion de branches avec flow

26 JUIN

Concepteur Développeur D'Application

Créé par : Jonathan Jeanniard & Aurélien BOUDIER

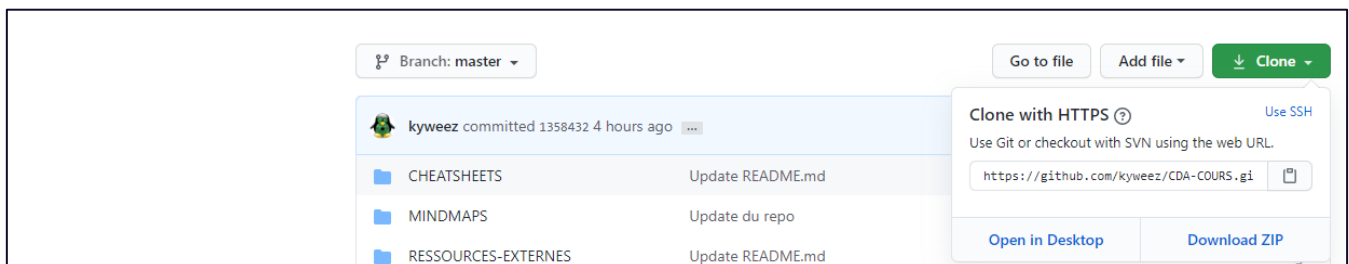


1. A quoi sert git flow ?

Git Flow est une méthode, une architecture Git permettant de séparer au maximum le travail et de toucher le moins possible à la branche Master. Cette méthode représente donc une architecture en branches. Pour commencer, on aura la branche Master au moment de l'initial commit. Ensuite, on va créer une branche develop, qui sera la branche principale. C'est par ici que vont passer tous les commits. La branche master ne récupérera le code de la branche develop qu'au moment d'une release, et à ce moment elle sera tag afin de différencier une version.

2. Récupération du dépôt

Dans un premier temps il faut récupérer un dépôt quel qu'il soit avec git clone.



3. Vérification git

Ouvrez l'invite de commande Windows (cmd) ou un terminal Linux. Naviguez jusqu'au répertoire où se trouvent le dépôt cloné avec la commande "cd" pour les deux systèmes.

```
Invite de commandes
D:\jonat\Documents\GitHub\CDA_2005_LABY>dir /AH /a
Le volume dans le lecteur D s'appelle Data
Le numéro de série du volume est EA59-6B6E

Répertoire de D:\jonat\Documents\GitHub\CDA_2005_LABY
16/06/2020  15:26    <DIR>          .
16/06/2020  15:26    <DIR>          ..
26/06/2020  07:07    <DIR>          .git
16/06/2020  15:26             6,352 .gitignore
16/06/2020  15:26             1,088 LICENSE
16/06/2020  15:26              15 README.md
               3 fichier(s)              7,455 octets
               3 Rép(s)  759,082,725,376 octets libres

jonas@JONAS-PC: ~/CDA_2005_LABY
jonas@JONAS-PC:~/CDA_2005_LABY$ ls -a
.  ..  .git  .gitignore  LICENSE  README.md
jonas@JONAS-PC:~/CDA_2005_LABY$
```

Une fois dans le répertoire du dépôt vérifiez que le dossier caché de git est bien présent :

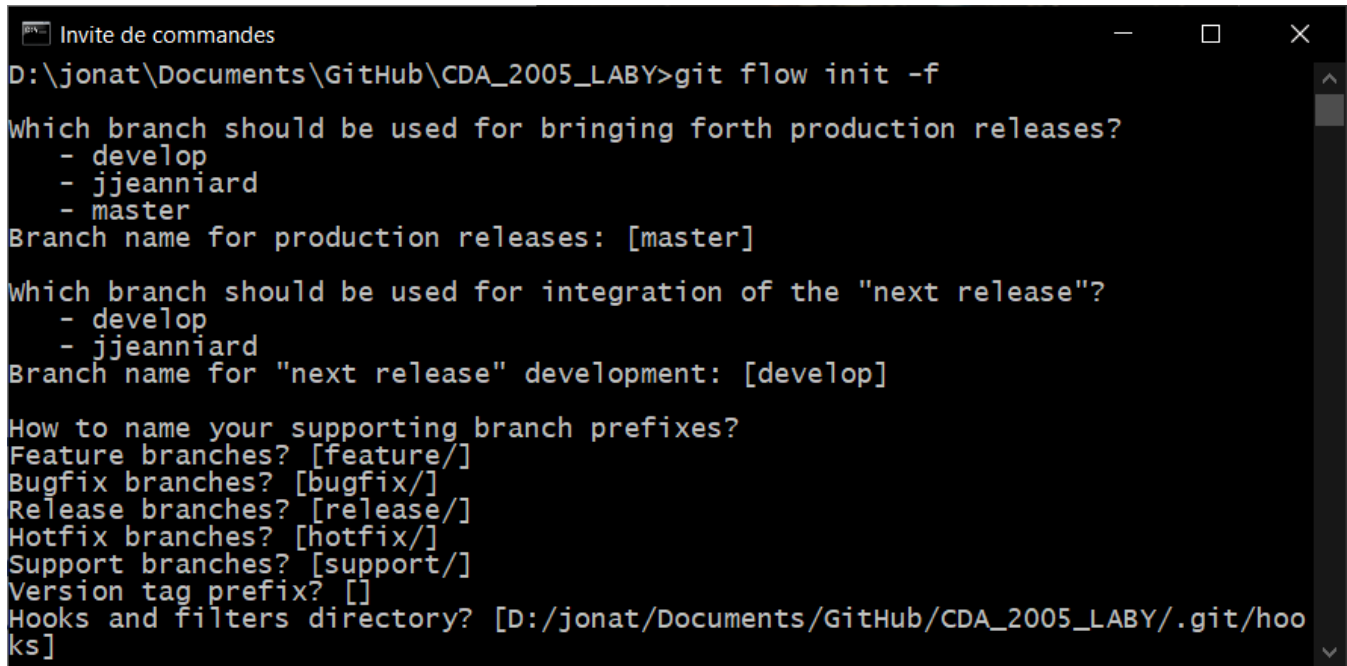
Windows : **dir /AH/a**

Linux/Mac : **ls -a**

4. Initialisation de flow

A partir de là, les commandes de git sont communes au 2 systèmes, nous utiliserons le cmd de Windows pour la suite pour plus de facilité.

Donc pour initialiser la gestion des branches, on tape `git flow init -f`. (L'option -f permet de forcer l'initialisation pour le cas où nous ayons déjà initialiser un flow précédemment).



```
D:\jonat\Documents\GitHub\CDA_2005_LABY>git flow init -f

Which branch should be used for bringing forth production releases?
- develop
- jjeanniard
- master
Branch name for production releases: [master]

Which branch should be used for integration of the "next release"?
- develop
- jjeanniard
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/jonat/Documents/GitHub/CDA_2005_LABY/.git/hooks]
```

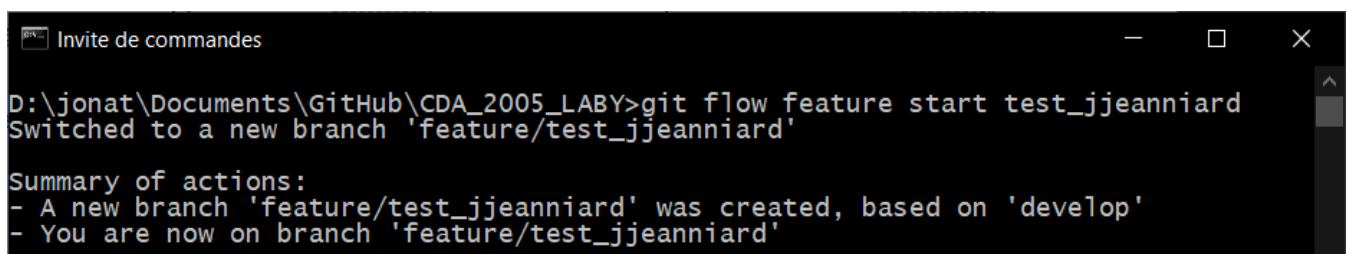
De nombreuses questions vous sont posées, vous pouvez tout mettre par default en appuyant sur Entrée (si un problème survient au moment de "next release", il faut au préalable créer la branche "develop"). Chaque branche correspond à une fonction donnée, comme on peut le remarquer.

5. Première branche

La première fonctionnalité de notre programme sera "test", donc la commande pour réaliser notre espace de travail est `git flow feature start test_jjeanniard`.

Par convention nous mettrons le pseudo/nom, précédé d'un underscore après le nom de la fonctionnalité.

Cette commande va créer une nouvelle branche tout en récupérant ce qui se trouve dans la branche "develop" et puis changera automatiquement de branche en nous plaçant dans la nouvelle.



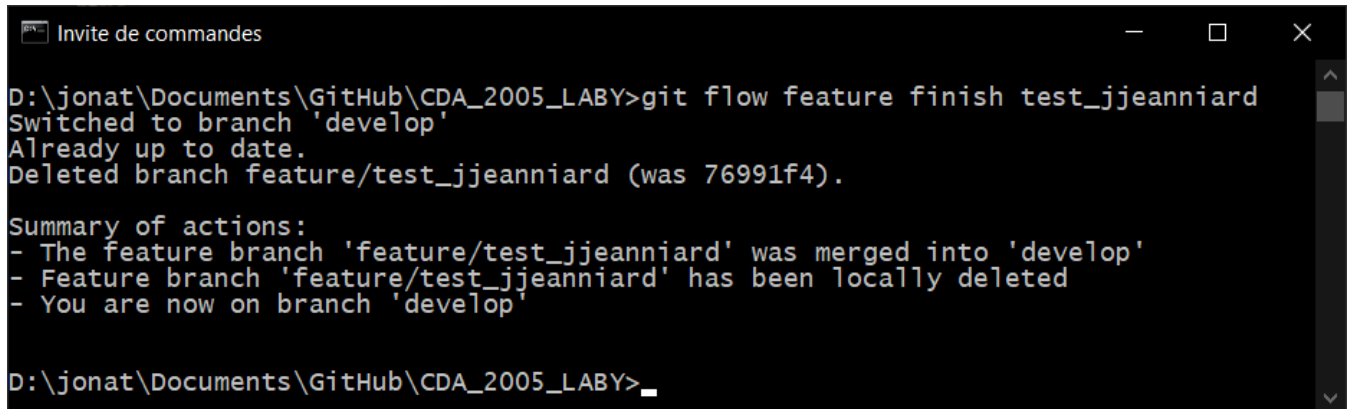
```
D:\jonat\Documents\GitHub\CDA_2005_LABY>git flow feature start test_jjeanniard
Switched to a new branch 'feature/test_jjeanniard'

Summary of actions:
- A new branch 'feature/test_jjeanniard' was created, based on 'develop'
- You are now on branch 'feature/test_jjeanniard'
```

NB : Dans les livres, ils préconisent de commencer dans la branche "feature" avec pour nom "init_projet" pour la réalisation d'un nouveau projet.

6. Fin de la branche

Après de nombreux commits, pushes, du "code review" sur la fonctionnalité (cycle sans fin) et un "pull request" réussi sur Github. Il est temps de changer de branche, mais avant d'en créer une nouvelle, il est préférable de « rendre la liberté » à votre précédente branche avec la commande "git flow feature finish test_jjeanniard".



```
Invite de commandes

D:\jonat\Documents\Github\CDA_2005_LABY>git flow feature finish test_jjeanniard
Switched to branch 'develop'
Already up to date.
Deleted branch feature/test_jjeanniard (was 76991f4).

Summary of actions:
- The feature branch 'feature/test_jjeanniard' was merged into 'develop'
- Feature branch 'feature/test_jjeanniard' has been locally deleted
- You are now on branch 'develop'

D:\jonat\Documents\Github\CDA_2005_LABY>_
```

Un merge de votre branche se fait automatiquement, avant d'être supprimé et puis vous êtes de nouveau switch dans la branche "develop". Une nouvelle vie de branche peu commencer (feature, hotfix, bugfix...).

Une chose à savoir, il n'est pas interdit de travailler sur plusieurs branches. A vous de savoir celles-ci et de ne pas modifier le mauvais fichier avec la mauvaise branche.

Crédit :

Action	Auteur	Descriptif	Date
Réalisation du document	Jonathan JEANNIARD	Découverte de la commande git flow	26/06/20
Agrémentation du document	Aurélien BOUDIER	Correction, reprise de la forme et ajout d'une section	26/06/20