Document Code : **SP201902PGFA2**

Présentation de Git Flow

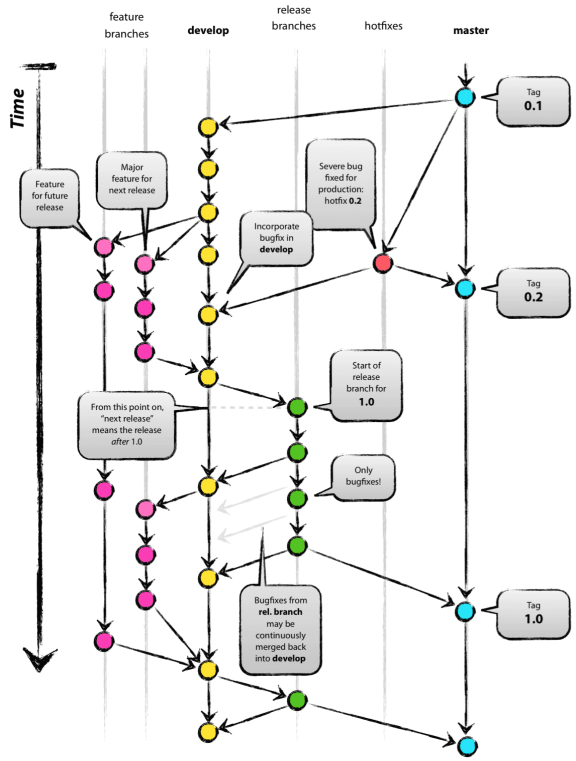
Historique des modifications

Version	Date	Auteur	Fonction	Comment
1.0	14/02/19	TEMA GILDAS	INGÉNIEUR	Création

Table des Matières

Introduction

git-flow est un modèle de branche, qui est fourni avec de la documentation, et un plugin git pour ajouter des commandes qui facilitent le travail.



Gardez en tête qu'il s'agit de standardiser ; des commandes git standard sont utilisées an arrière plan, vous pourriez obtenir le même résultat « manuellement » qu'avec git-flow. C'est juste plus simple à utiliser, et ça évite d'utiliser la mauvaise branche, ou d'oublier de merger quelque part.

Le but de la présente documentation n'est pas de lister les pour et les ocntre de ce modèle, on notera simplement qu'il n'est pas prévu pour fournir des branches de support à long terme, c'est quelque chose qui avait été évoqué mais qui n'a finalement jamais été implémenté.

D'après les règles de visionnage sémantiques :

- vous ajouterez des features pour les versions *majeures* et *mineures* uniquement,
- vous créerez des release pour des versions *majeures* ou *mineures*,
- vous créerez des hotfix pour les versions *patch*.

Pré-requis

Pour que les commandes soient disponibles, vous devrez installer le plugin git git-flow.

La majorité des distributions linux le fournissent dans leur dépôts (donc `yum install git-flow` ou `apt-get install git-flow` devrait faire l'affaire) ou vous pouvez suivre les instructions d'installation depuis le wiki du projet.

Beaucoup de logiciels GIT supportent git-flow, ou le peuvent par le biais de plugins ; consultez les documentations respectives.

Si vous utilisez la ligne de commande, il existe de nombreux moyens pour afficher des informations utiles dans le prompt. Bien que ce ne soit pas un pré-requis, cela peut vous faire gagner du temps !

Prise en main

Initialisation du projet

git init

Pour initialiser le projet git.

```
→  gitflow git init
Dépôt Git vide initialisé dans /home/gildas/Documents/Lab/gitflow/.git/
```

git flow int

Initialisation du projet avec flow

```
→  gitflow git:(master) git flow init
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
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? [/home/gildas/Documents/Lab/gitflow/.git/hooks]
```

Comme on peut le constater l'architecture git flow offre plusieurs branches à savoir :

- feature: pour de nouvelle fonctionnalités
- master : la branche principale
- hotfix: pour les corrections en production
- bugfix: pour la correction des bug en dev
- releases : pour le différentes releases

A présent, nous allons voir comment créer et gérer chacune des ses branches.

Gestion des Features

```
→  gitflow git:(develop) git flow feature start automatic_recharg
Basculement sur la nouvelle branche 'feature/automatic_recharg'

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

Now, start committing on your feature. When done, use:

     git flow feature finish automatic_recharg
```

On peut constater que cette branche est une sous branche de feature.

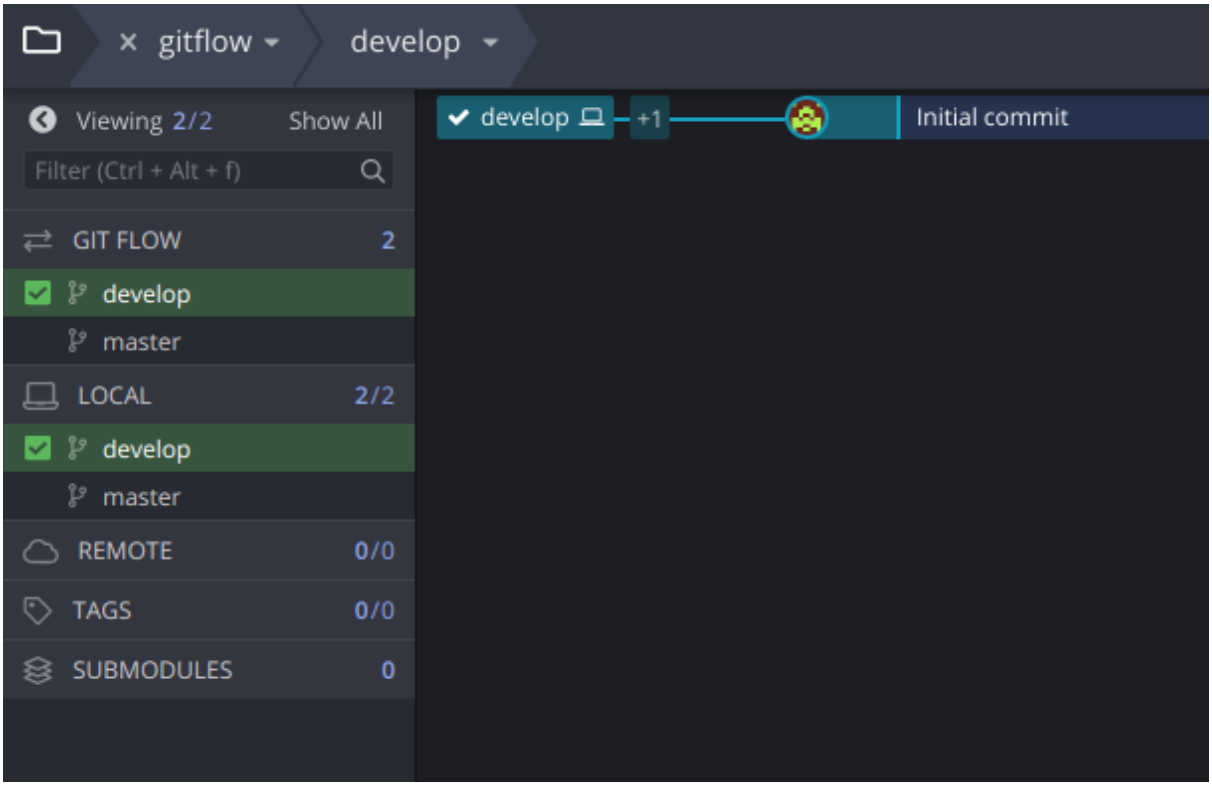
aprés modification pour finaliser la branche on peut finaliser la branche avec la commande .

git flow feature finish automatic_recharg

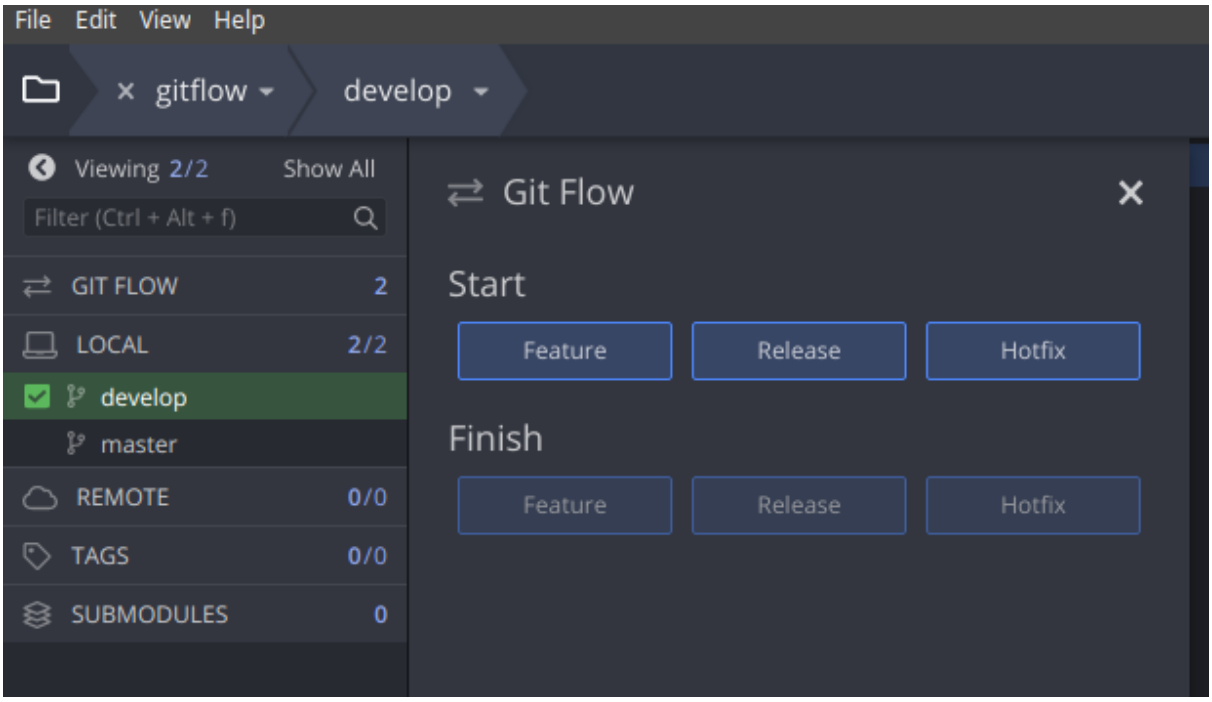
```
→  gitflow git:(feature/automatic_recharg) git flow feature finish  automatic_recharg
Basculement sur la branche 'develop'
Déjà à jour.
Branche feature/automatic_recharg supprimée (précédemment aececd6).

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

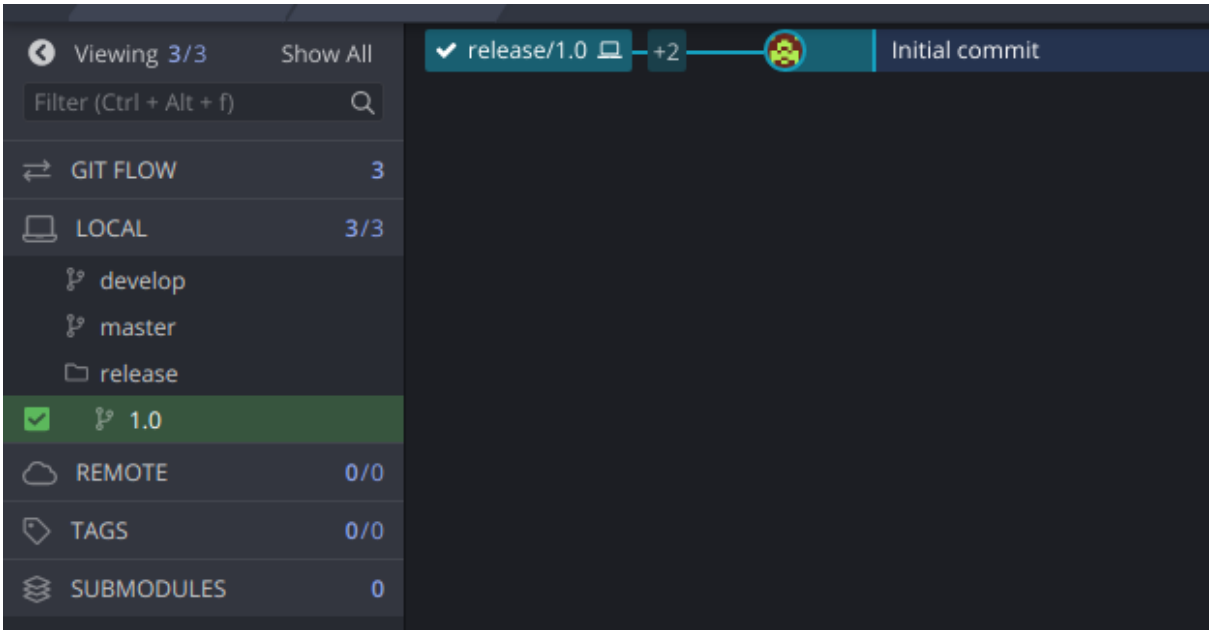
Branche develop

La branche develop est la branche de travail, c'est à partir de cette branche qu'on crée les branches. Pour continuer nous allons servir nous allons vous présenter un outil très intéressant pour gérer un projet git ou git flow. **GitKraken**

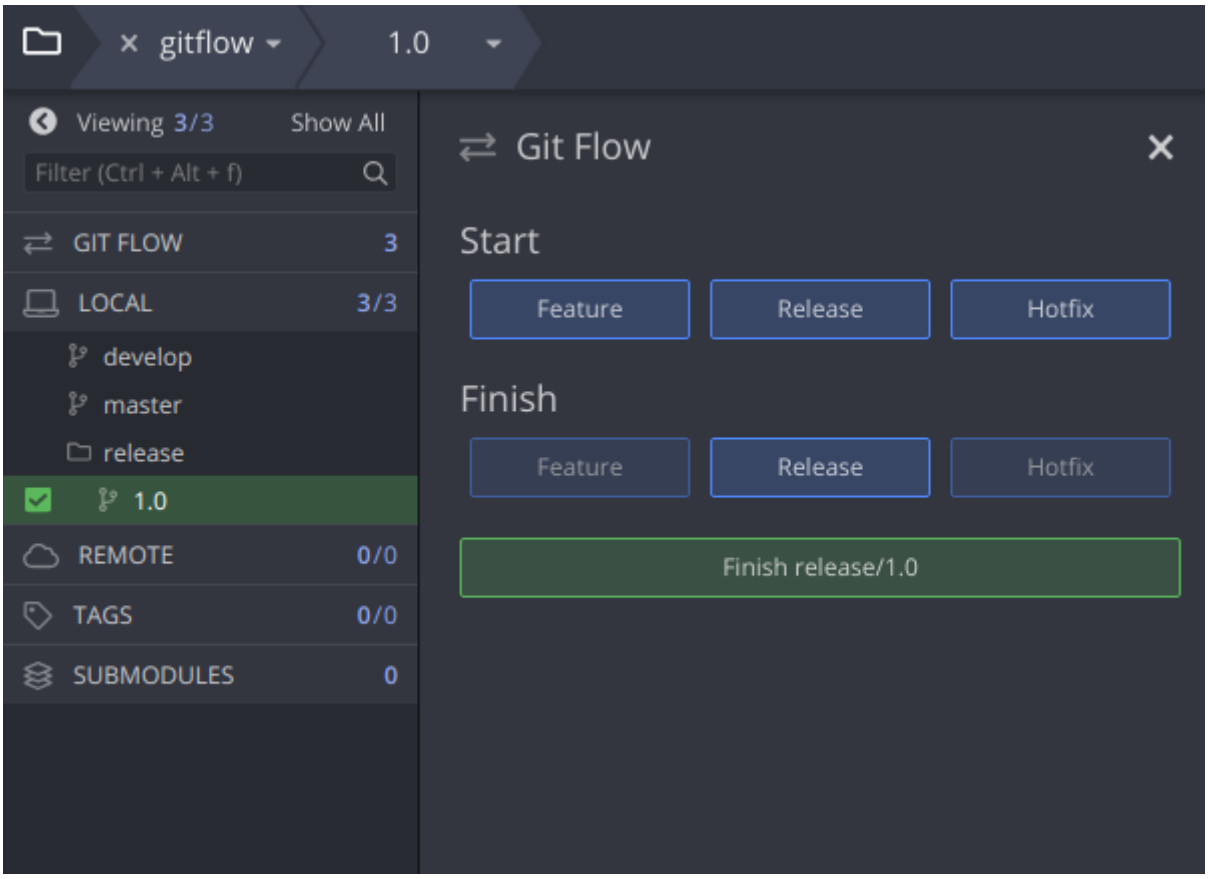


Nous allons à présent créer une realease avec GitKraken.



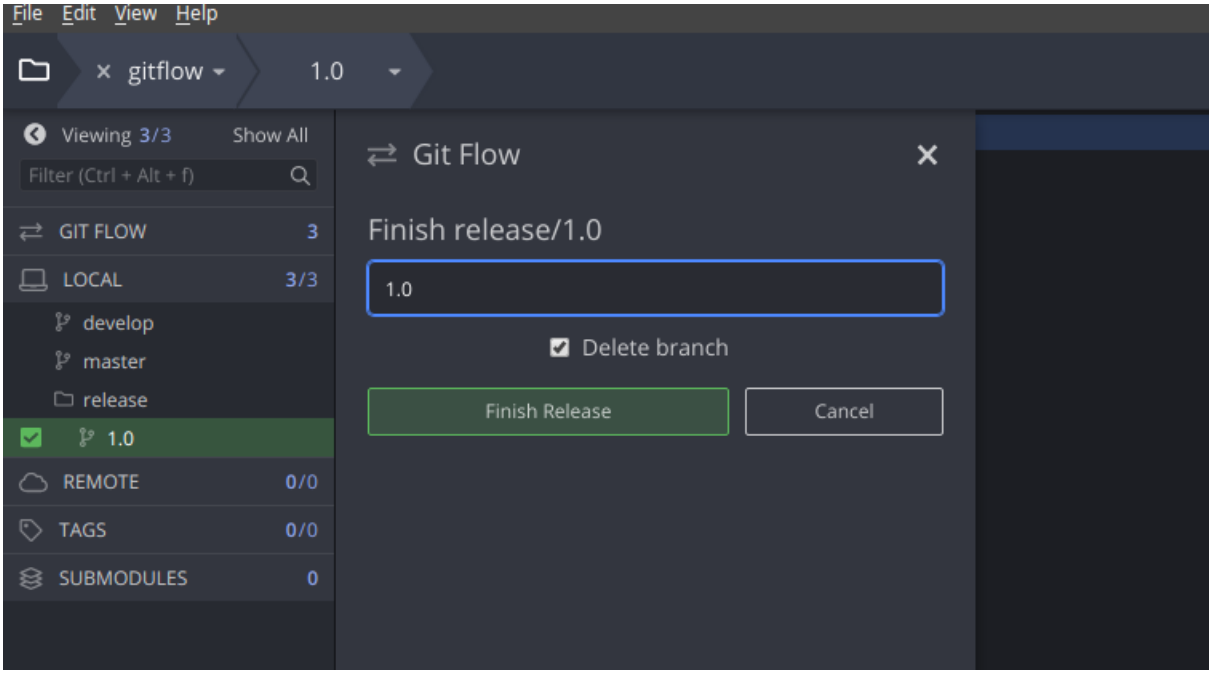


pour finaliser la release,



Mettre sur la branche master

Master


File Edit View Help
gitflow
1.0
Viewing 3/3
Show All
Filter (Ctrl + Alt + f)
GIT FLOW 3
LOCAL 3/3
develop
master
release
1.0
REMOTE 0/0
TAGS 0/0
SUBMODULES 0
Git Flow
Finish release/1.0
1.0
Delete branch
Finish Release
Cancel