**Introduction à Git**

Table des matières

[I. En local sur la console git bash 2](#_Toc94597250)

[1. Introduction 2](#_Toc94597251)

[2. Commandes de base 2](#_Toc94597252)

[3. Git init, git add et git push 3](#_Toc94597253)

[II. Connexion à un nouveau repository 5](#_Toc94597254)

[1. création d’un nouveau repository 5](#_Toc94597255)

[2. Travailler sur une branche 6](#_Toc94597256)

[3. Git merge 7](#_Toc94597257)

[4. Git stash 8](#_Toc94597258)

[III. Modèle de branche git efficace 9](#_Toc94597259)

[1. Organisation des branches pour gérer les versions d’application 9](#_Toc94597260)

[2. Raccourcis Git : alias 10](#_Toc94597261)

# 

# En local sur la console git bash

## 1. Introduction

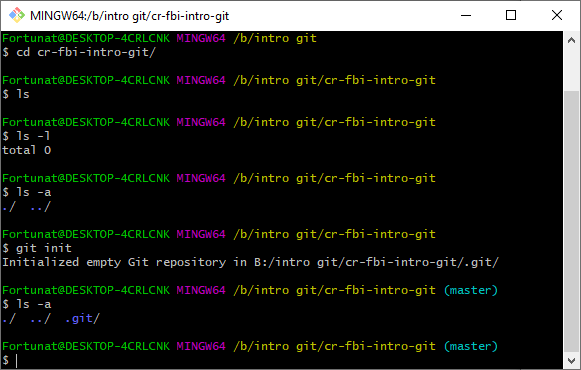
• Git est un logiciel de gestion de version distribué.

• Les différentes versions des fichiers d’un projet seront stockées sur l’ordinateur de chacun des contributeurs.

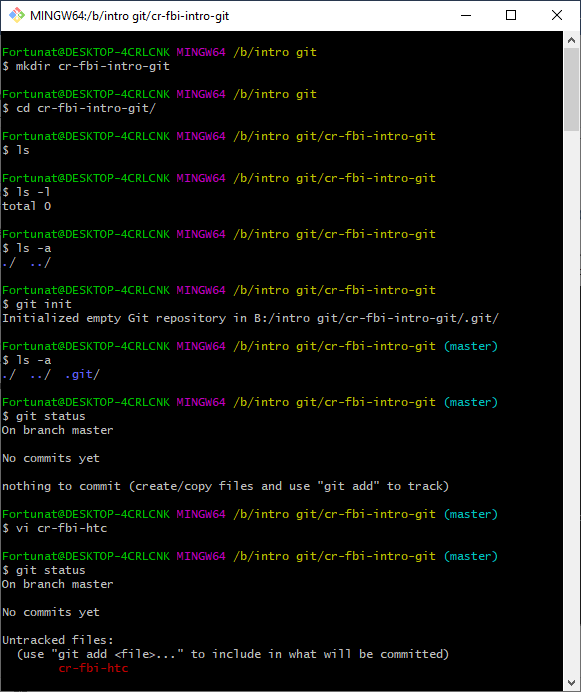
• pour faciliter la transmission des modifications, les fichiers sont également stockés sur un serveur

• Git est un projet open source avancé, qui est activement maintenu.

## 2. Commandes de base (Shell)



**Mkdir** *nom\_nouveau\_dossier* pour créer un nouveau dossier.



**Cd** *nom\_dossier* change directory : aller à …

**touch** *nom\_fichier* pour créer un fichier.

**ls** pour lister les dossier du répertoire. Ici il est vide car le répertoire vient d’être créé.

**ls -l** pour compter le nombre de dossier

**ls -a** : **.** **/** dossier actuel **.. /** dossier parent **.git/** répertoire git.

**vi** *nom\_fichier.txt* pour ouvrir et modifier un fichier via la console :



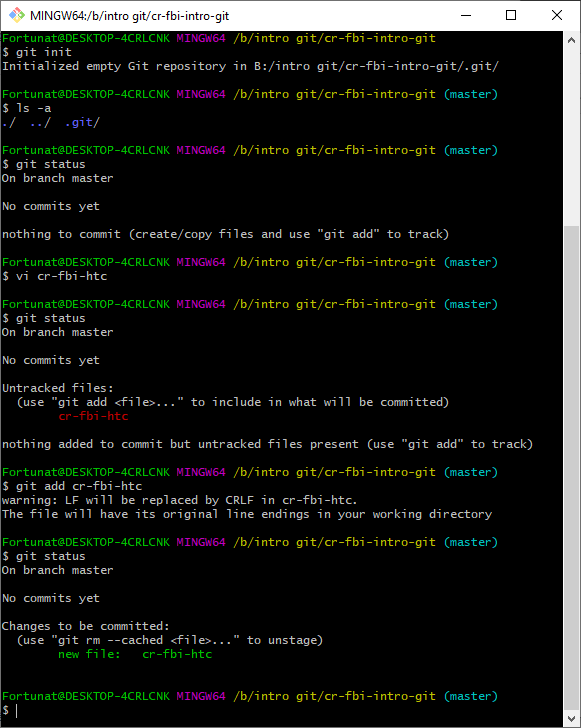
* **i** pour insérer dans le fichier
* **[échap]** pour quitter le mode insert
* **:wq** pour enregistrer et quitter
* **q!** pour sortir

## 3. Git init, git add et git push

La première chose à faire après l’installation de Git est de renseigner le nom d’utilisateur et l’adresse de courriel. C’est une information importante car toutes les validations dans Git utilisent cette information et elle est indélébile dans toutes les validations que vous pourrez réaliser :

$ **git config --global user.name** "John Doe"

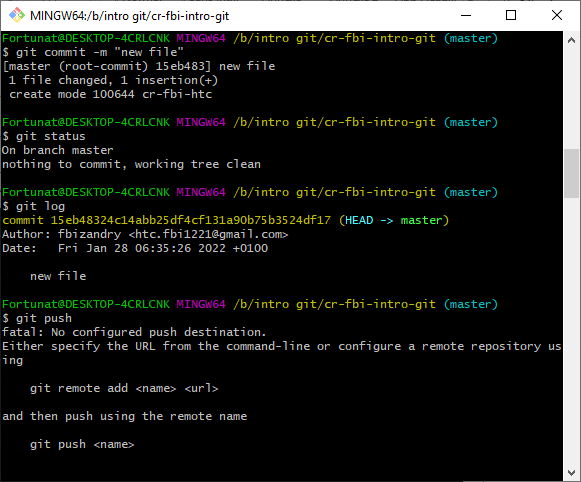
$ **git config --global user.email** johndoe@example.com



**git init** sert à créer un dépôt .Git vide ou réinitialiser un dépôt existant.

**git status** (à faire quasi systematiquement!) La commande git status affiche l'état du répertoire de travail et de la zone de staging.

**Git add .** met à jour l’index en utilisant le contenu actuel trouvé dans le répertoire de travail, pour préparer le contenu de la prochaine validation.



**git commit *nom\_fichier* -m ‘’message’’** crée un nouveau commit du fichier contenant le contenu actuel de l’index et avec le message de validation décrivant la modification. La commande git commit -m ’’msg’’ va commit-er tous les changements.

**git log** affiche l’historique des actions effectuées sur le répertoire et a comme option les commandes suivantes :

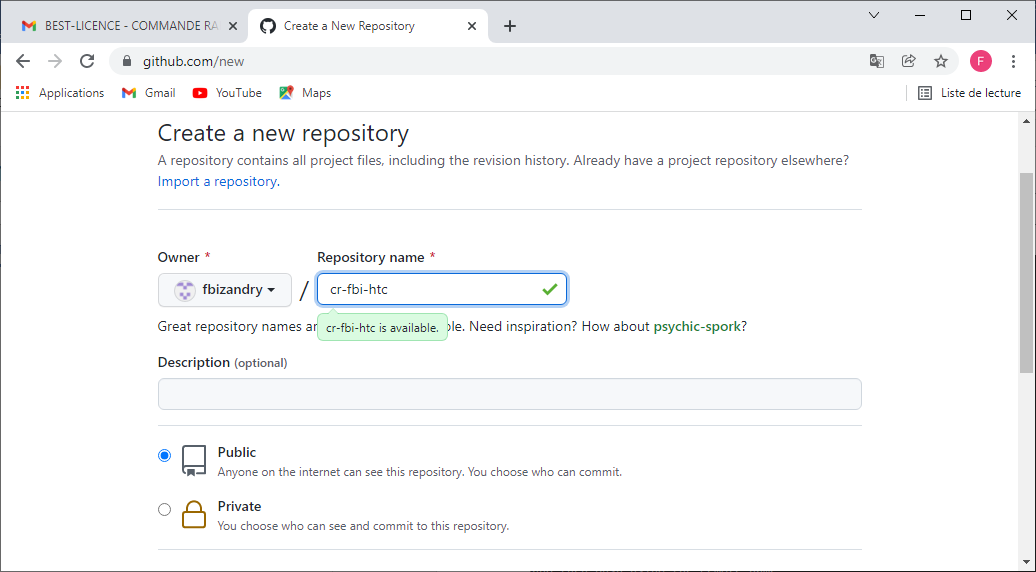
* **git rev-list** pour contrôler ce qui est montré et comment,
* **git diff**' pour contrôler la façon dont les modifications que chaque validation introduit sont affichées.

**git push** permet de transférer les commits du dépôt local vers un dépôt distant.

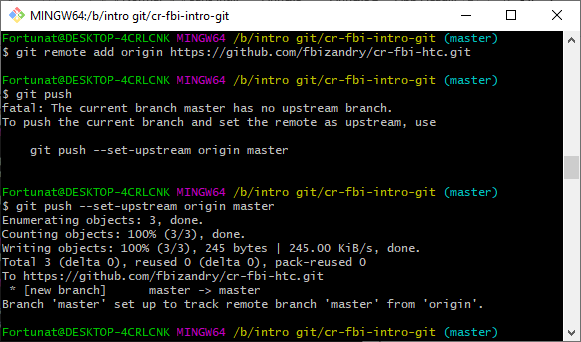
**Remarque :** ici ça ne marche pas car le répertoire local n’a pas de lien avec le repository à distance ni avec une branche.

# Connexion à un nouveau repository

## 1. création d’un nouveau repository



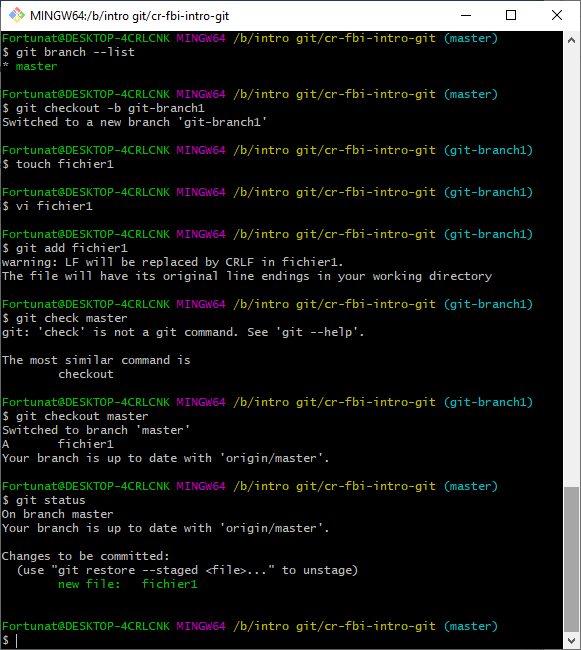
* Création de compte GitHub
* Création de nouveau dépôt (repository)



**Git remote add origin *URL\_repository*** pour l’associer l’adresse du dépôt du répertoire à distance.

**Git push –set-upstream origin** master pour associer la branche correspondant à l’envoie.

## 2. Travailler sur une branche



**Git branch *nom\_nouvelle\_branche***pour créer une nouvelle branche

**Git branch list** sort la liste des branches et met en vert la branche actuelle.

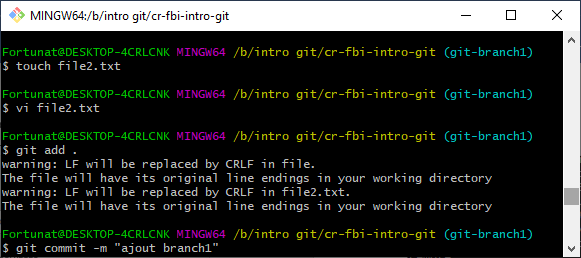
**Git checkout** est la commande pour changer de branche.

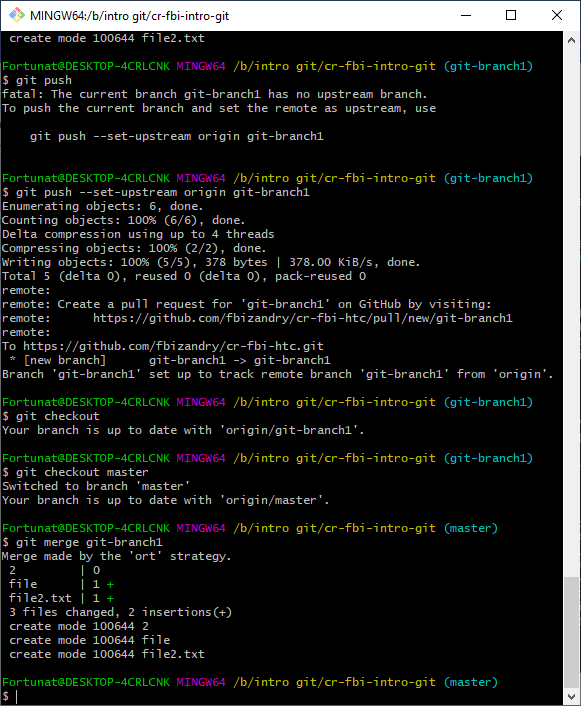
**Git checkout -b *nom\_nouvelle\_branche*** est la commande pour créer et entrer dans une nouvelle branche.

**ATTENTION**: un fichier ajouté à l’index par la commande git add doit être traité avant de changer de branche car il peut être commit-é de n’importe quelle autre branche. Dans l’exemple ci-dessus on peut ouvrir et modifier file1 change de branche de master à git-branch1.

## 3. Git merge

Dans cette partie on effectue des modifications sur la branche git-branch1 suivi d’un git add, git commit et git push et ensuite on veut fusionner cette branche à ma branche *principale* master.

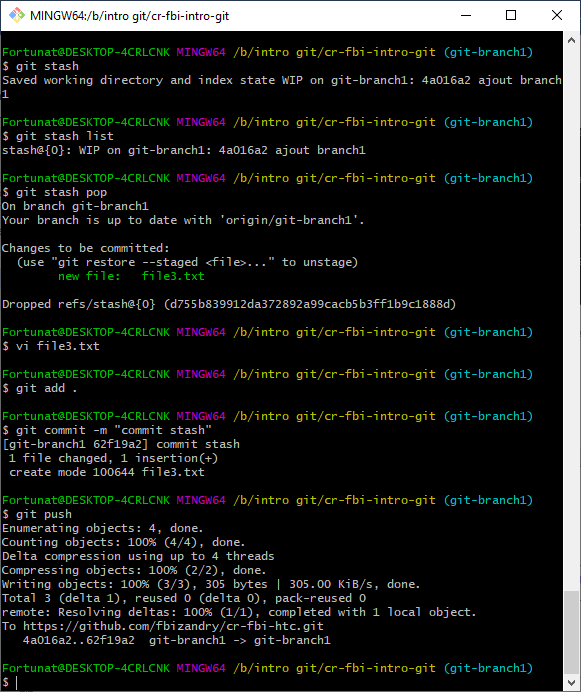




Pour faire cette merge il faut se placer dans la branche principale et on appelle l’autre branche par la commande **git merge *nom\_branche.***

## 4. Git stash

La commande git stash permet de sauvegarder des modifications locales indexées (add .) d’une branche sans commit ni push le fichier pour ne pas polluer répertoire de travail. git stash stocke (ou stashe) temporairement les changements apportés à la copie de travail pour pouvoir effectuer d'autres tâches, puis revenir et les réappliquer par la suite.



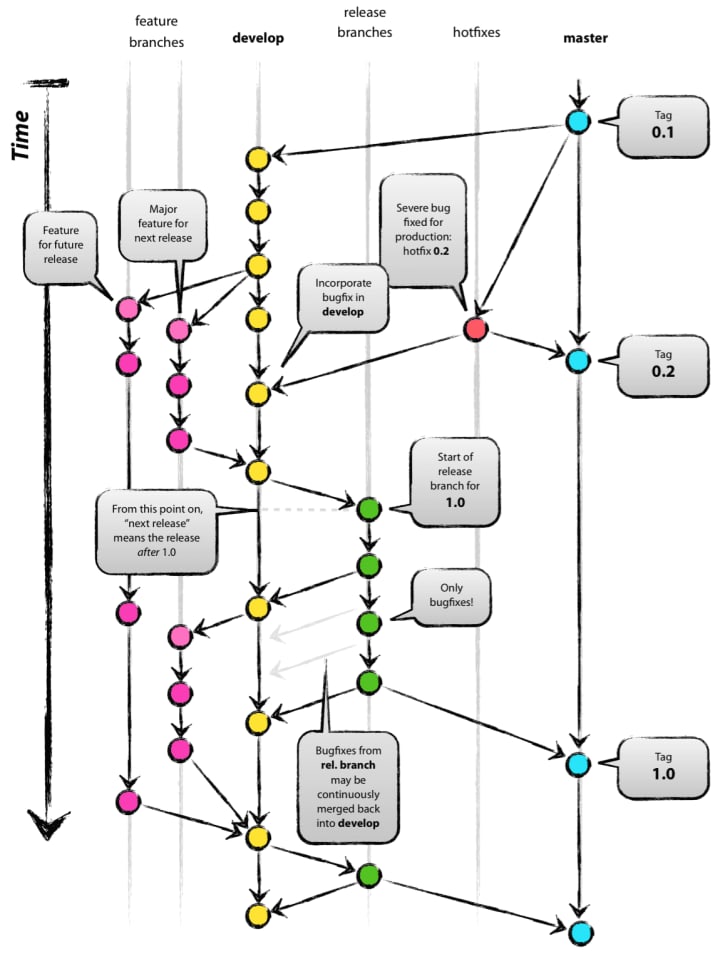
**git stash** est la commande qui permet de planquer les modifications après un git add .

**git stash list** est la commande pour lister les fichiers en stash

**git stash pop** est la commande pour sortir les fichiers des planques de n’importe quelle branche. Attention : Last In First Out.

# Modèle de branche git efficace

## 1. Organisation des branches pour gérer les versions d’application



https://frank.taillandier.me/2014/12/14/un-modele-de-branches-git-efficace/

## 2. Raccourcis Git : alias

Il y a deux manières de créer des raccourcis de commandes git :

* Sur la console avec la commande : git config --global alias.<short> <long>
* Chercher le fichier .gitconfig dans : disque local ( :C) > utilisateur > .gitconfig. En suite ajouter [alias]

<short> <long>