



Git & Github



Git

- **Version Control System**

- Suivre attentivement les modifications dans vos fichiers
- Collaborer plus facilement
- Tester des modifications sans perdre les versions originales
- Revenir à des versions antérieures si nécessaire/Rollback



Github



- **Service d'hébergement web pour Git**
 - Peut servir de « lieu distant » pour stocker vos espaces de travail Git.
 - Peut toujours être accessible même quand on perd notre ordi etc, de partout
 - Permet également de contribuer à des projets open source
 - Retrouver les codes sources de nombreux outils/packages









Git

- **Création de compte**
 - Création de compte sur github: <https://github.com/join>
- **Installation**
 - lien de téléchargement: <https://git-scm.com/downloads>

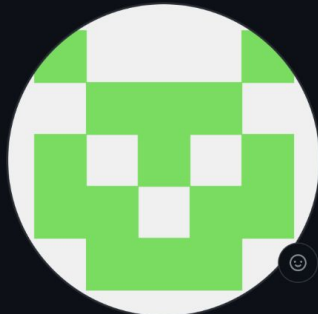

Github

 idrissa-mgs

Q Type  to search

Overview Repositories **23** Projects Packages Stars **1**



idrissa-mgs
idrissa-mgs
Passionate about Data-Engineering and
Programming

Popular repositories

Customize your pins

mnist_challenge Public
Forked from MadryLab/mnist_challenge
A challenge to explore adversarial robustness of neural networks on MNIST.
Python

paris-dauphine Public
Forked from Sabmit/paris-dauphine
Python

e-commerce_product_categorization Public
Python

Computer-Vision Public
Jupyter Notebook

simple_ml_projects Public
Jupyter Notebook

scraping Public
Python



Git

Repository: Emplacement distant de ton projet et qui recense tout l'historique de ce dernier.

On peut le créer depuis github ou depuis une commande git

Initialiser un repo: *git init <nom-du-repo>*

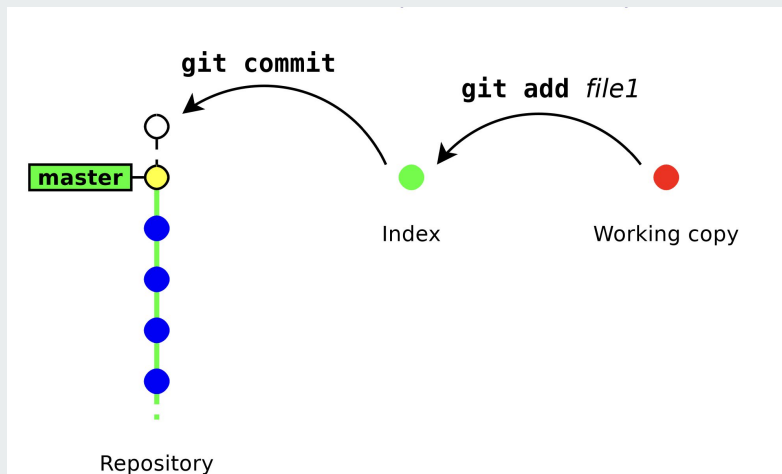
une fois la commande lancé un dossier `.git` sera créé et on y trackera tous les changements qu'on va faire sur le repo

Git

Staging area ou index:

On peut le voir comme une étape transitoire nécessaire où on ajoute les fichiers/dossier ayant subis des modifications qu'on souhaite pousser en local.

On le fait via la commande: `git add <chemin-du-fichier/dossier>`





Git - Commands

git clone <lien du repo>

- Pour récupérer un repository/projet distant en local

Git - Commands



git add <chemin du fichier ou dossier>

- Pour dire à git de tracker/prendre en compte les changements faits sur ce fichier ou ce dossier
- Les fichiers non ajoutés seront ignorés et ne seront pas tracés.
- On dit également que les fichiers ajoutés ont été ajoutés à la staging area de git

git add -A ou ***git add .*** (pas conseillé)

- Pour ajouter tous les fichiers

Exemple:

git add foo.py

On peut également retirer des fichiers/dossier de la staging area

git rm foo.py



Git - Commands

git commit -m "message about my last changes"

- Cette permet de commiter le staging area
- Prend une "capture instantanée" de tous les fichiers actuellement dans la zone de staging et les valide dans la mémoire de Git.
- Ajouter un message explicite qui décrit ce qui vous avez changé



Git - Commands

git diff

- Permet de voir la différence entre l'index et l'état courant du repos en local

git reset

- supprime/annule les changements ajouter au niveau de l'index

git reset --hard

- supprime tous à la fois les changements au niveau de l'index mais aussi en local



Git - Commands

git status

- Pour voir les récents changements faits



Git - Commands

git push

- Pour pousser les changements (en staging area) en local vers le repo distant (de mon ordi vers github)
-



Git - Commands

git pull

- Pour récupérer des changements faits sur le repo distant en local



Git - branches

Sur git, on travaille sur des branches

- Une branche peut-être vu comme comme une version parallèle du projet/repo qu'on va pouvoir faire évoluer sans "impacter" les autres devs
- Il existe toujours une branche principale souvent nommer master ou main
- Pour créer de nouvelles features, résoudre des bugs, etc **pensez à créer une nouvelle branche**



Git - branches

Pour créer une nouvelle branche

- `git branch <nom_nouvelle_branche>`

Pour supprimer une branche existante

- `git branch -d <nom_de_la_branche>`

Pour lister les branches

- `git branch`

Pour se positionner sur nouvelle branche

- `git checkout <ma_branche>`

Pour créer et se mettre sur une nouvelle branche

- `git checkout -b <ma_branche>`



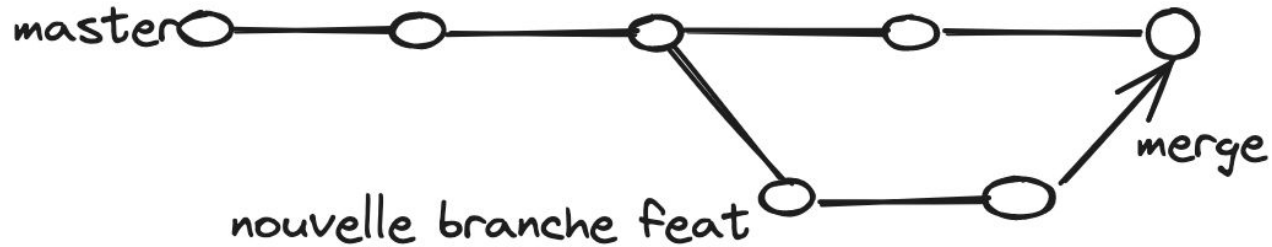
Git

Pour voir l'historique des différents commits faits

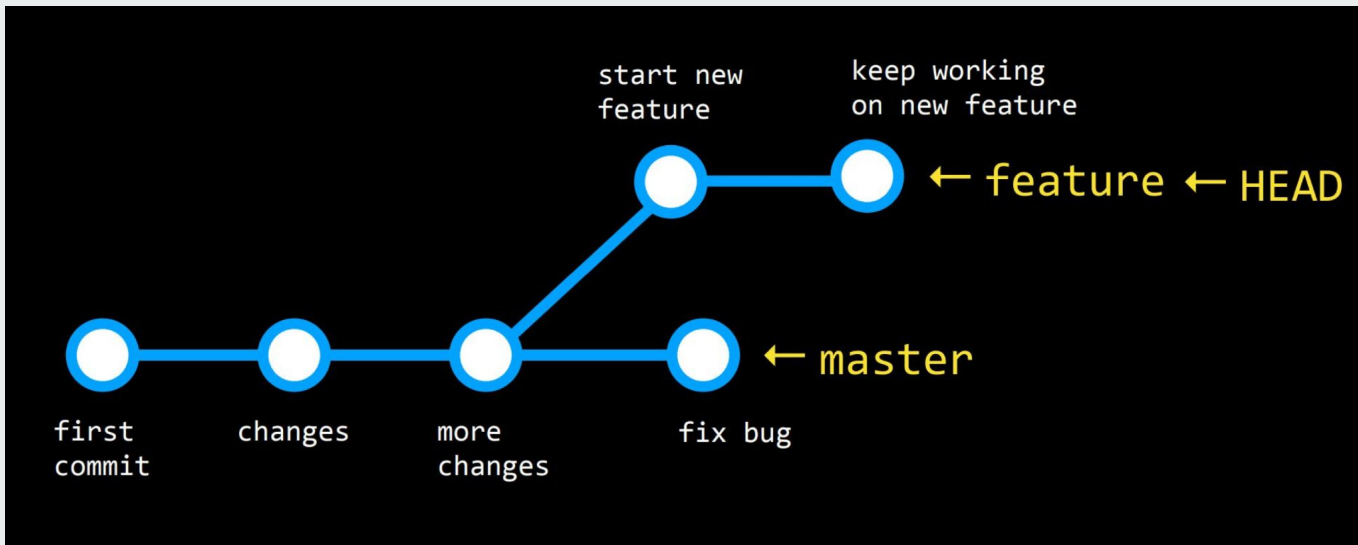
- *git log*

Git - merge

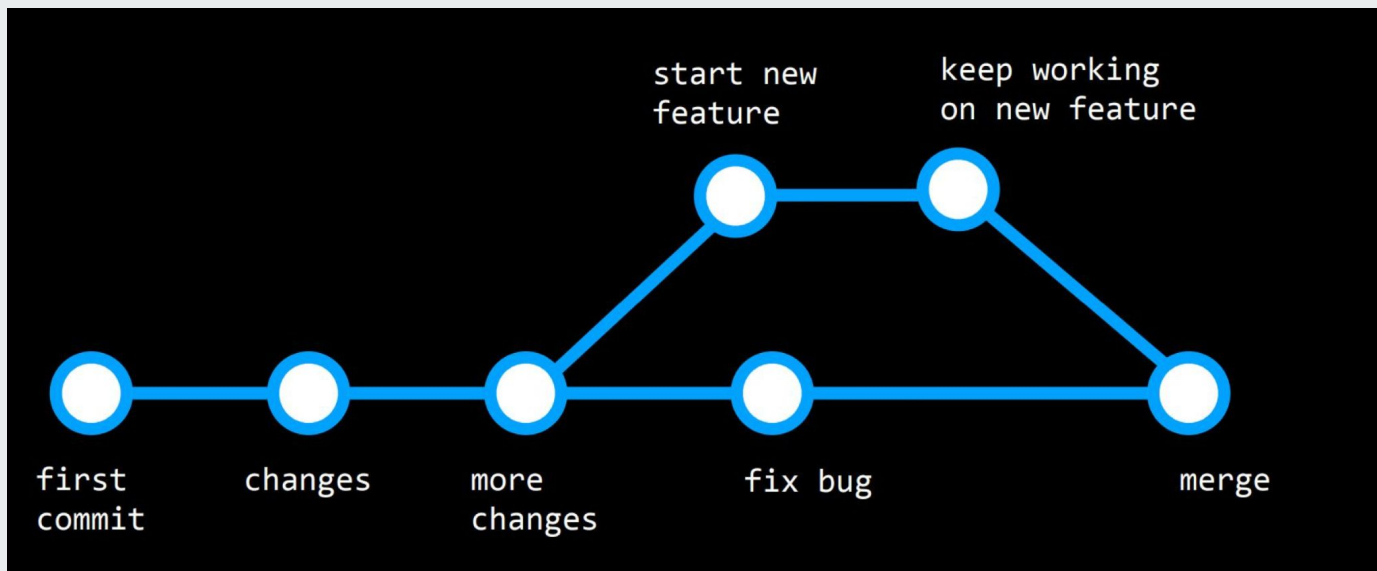
- Workflow



Git - merge



Git - merge





Git - merge

- On se positionne sur notre branche puis on fait
 - `git merge <master or another branch>`
- Le résultat du merge est automatiquement *commit* (sauf en cas de conflit)
-



Git - Quelques bonnes pratiques

- **Ne jamais pousser sur master/main**
- Créer une branche feature ou bug
 - `git checkout -b feat/ajout-feat`
 - `git checkout -b fix/bug33`
- Partir de la branche master
 - `git checkout master ; git pull & git checkout -b "ma-branche"`
- Ajouter des messages explicites au commit
- Créer des Pull Request pour permettre aux autres membres de l'équipe de review votre code



Git - TP

- cloner le repos <https://github.com/idrissa-mgs/intro-devops/tree/main>
- créer et se mettre sur une nouvelle branche (<feat/your-name>)
- créer un nouveau fichier simple en python
- pousser ce changement vers le repo distant et vérifie que le changement a été