# Introduction à Git


EL HAMMOUDI HAMZA

# Introduction

- Il existe de nombreux logiciels de gestion de version, qui peuvent être basés sur différents modèles :

  - **Modèle centralisé** : un serveur central contrôle toute la base de code du logiciel.
    *Exemples de logiciels de versioning utilisant un modèle centralisé :* SVN, CVS.

  - **Modèle distribué** : toutes les machines ont accès à la base de code, pas besoin de passer par un serveur central.
    *Exemples de logiciels de versioning utilisant un modèle distribué :* Git, Mercurial, Bazaar.

# Introduction

- Le modèle distribué présente plusieurs avantages :
    - Moins de risques de perdre son code puisqu'il est accessible par plusieurs sources.
    - On peut travailler plus rapidement et sans être connecté à Internet puisqu'il n'y a pas besoin de se connecter à un serveur central.

# Introduction

git

- logiciel de gestion de versions décentralisé.
- C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU.
- En 2016, il s’agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes.

# Installer Git

- Télécharger Git

https://git-scm.com/download/

- Lancer les commandes sur la console
  *#git config --global user.name "Votre nom ou pseudo"*
  *#git config --global user.email "Votre@email.com"*

# Créer un repository

- Sur la console, se positionner sur le répertoire à activer comme repository Git
- Lancer la commande
  *#git init*

- Pour versionner un fichier sur le répertoire l'ajouter par la commande
  *#git add nomDeVotreFichier.extension.*
- Pour ajouter tous les fichiers d'un répertoire
  *#git add .*

# Premier commit

- Faire une modification d’un fichier sur le repository
  *#git commit -m "première modification"*

# Voir l’historique

- Commande console
  *#git log*

- Dans la liste de cet historique, chaque commit est répertorié avec:
  - son **SHA** : son identifiant unique
  - son auteur : qui a fait le commit
  - sa date
  - sa description : vous vous rappelez, c'est le message de description que vous indiquez avec l'option  **-**m

# Mettre à jour un fichier

- lorsque vous mettez à jour un fichier dans votre repository, vous devez procéder en deux étapes :
  - Ajouter votre fichier à l'index avec la commande  git add,
    *#git add fichier.extension*
  - Faire un commit qui décrit la mise à jour de votre fichier avec la commande  git commit.
    *#git commit -m  " commentaire "*
- Si vous ne faites que mettre à jour un fichier, vous pouvez condenser ces deux étapes

  *#git commit -a -m  " commentaire "*

  ou

  *#git commit -am  " commentaire "*

# Se positionner sur un commit donné

- Commande

  *#git checkout SHADuCommit*

- Pour revenir à votre branche principale (au commit le plus récent), on utilise la même commande :

  *#git checkout master*

# Annuler un commit

- Vous pouvez "revert" un commit, c'est-à-dire créer un nouveau commit qui fait l'inverse du précédent, avec la commande :

  *#git revert SHADuCommit*

- Attention, ça crée un nouveau commit dans l'historique du coup ! (le commit "inverse" du précédent)

# Annuler les changements non commités

- Je n'ai pas encore fait mon nouveau commit, mais je veux annuler tous les changements que je n'ai pas encore commités. Possible avec un reset !

```
#git reset --hard
```

# Les remotes

- Lorsque vous travaillez sur un projet sur votre machine, il est important d'avoir un backup de votre code sur une autre machine, au cas où la vôtre tombe en panne par exemple.
- Utiliser un remote externe va aussi vous permettre de travailler sur des projets à plusieurs, pour que tout le monde ait accès aux dernières modifications de chacun sur un remote partagé.

# Les remotes

- Une fois que vous avez travaillé sur votre code et effectué vos commits, vous allez donc les envoyer sur un **remote**, c'est-à-dire une autre machine qui peut être :
  - interne (si vous avez la chance d'avoir plusieurs ordinateur)
  - ou externe (grâce à des services comme GitHub ou BitBucket).

# GitHub

- GitHub est un service en ligne qui permet d'héberger ses repositories de code.
- GitHub est un outil gratuit pour héberger du code open source, et propose également des plans payants pour les projets de code privés.
- C'est le numéro 1 mondial et il héberge plus d'une dizaine de millions de repositories !

# GitHub – Créer un compte

- Pour créer votre compte GitHub
  https://github.com/
- Une fois votre compte créé, vous aurez accès à votre dashboard et découvrirez toutes les fonctionnalités de GitHub.

# Récupérez du code d'un autre repository

- À partir de GitHub, vous pouvez copier un repository sur votre machine. Pour cela, il vous suffit de rechercher le repository qui vous intéresse sur GitHub, de vous y placer, puis d'utiliser l'option "clone URL" en bas à droite de l'écran.

- Cette option vous propose un lien SSH, HTTPS ou Subversion. Ici, nous allons choisir un lien HTTPS, le copier, puis coller ce lien en utilisant la commande **git clone** dans le dossier que vous aurez choisi sur votre machine :

  *#git clone lienFourniParGitHub*

# Créez votre premier repository

- Connectez-vous à votre compte GitHub. Cliquez sur le bouton "Create new" symbolisé par un signe "+" en haut à droite de votre écran, puis "New repository".

- GitHub vous demandera alors de préciser quelques détails sur votre repository:
  - son nom
  - sa description
  - son statut public ou privé. Ici, nous partageons du code open-source, c'est donc public et gratuit

# Envoyez votre code sur GitHub

- Faire un clone de votre repo GitHub sur votre machine.

  *#git clone lienFourniParGitHub*

- Comment faire pour synchroniser les modifications que vous faites dans votre repo sur votre machine avec votre repo sur GitHub ?
  - Ouvrez votre terminal et placez-vous dans votre repo local.
  - Faites un/des commit(s) des modifications que vous avez ajoutées sur ce repo.
  - *Pushez!* Envoyez ces modifications sur votre repo GitHub en utilisant la commande git push:

    *#git push origin master*

# Envoyez votre code sur GitHub

- La branche **master** est la branche qui contient le code courant de votre repo GitHub.
- Le remote sur lequel vous envoyez votre code est appelé **origin** par défaut. Ici, ce remote est GitHub.

- Une fois que vous avez "pushé" votre code en ligne, vous pouvez aller consulter votre repo sur GitHub.
- Vous y retrouverez les derniers commits effectués en cliquant sur l'option "Commits" dans votre repo.

# Récupérez des modifications

- Pour récupérer en local les dernières modifications du repo GitHub, il vous faut utiliser la commande **git pull** :

  *#git pull origin master*

# Créez des branches

- Pour voir les branches présentes dans votre repo, utilisez la commande

  *#git branch*

- Elle vous retournera les branches présentes, et ajoutera une étoile devant la branche dans laquelle vous êtes placés.

# Créez des branches

- Pour créer une nouvelle branche, il vous suffit d'ajouter le nom de la branche à créer à la suite de la commande précédente :

  *#git branch nouvelle-branche*

- Pour vous placer dans une autre branche à l'intérieur de votre repo, vous allez avoir besoin d'un nouveau mot-clé : **checkout** :

  *#git checkout nouvelle-branche*

- vous pouvez regrouper ces deux opérations en une seule commande :

  *#git checkout -b ma-branche*

# Fusionnez des branches

- Lorsque vous travaillez sur plusieurs branches, il va souvent vous arriver de vouloir ajouter  dans une branche A les mises à jour que vous avez faites dans une autre branche B. Pour cela, on se place dans la branche A :

  *#git checkout brancheA*

- Puis on utilise la commande **git merge** :

  *#git merge brancheB*

# Retrouvez qui a fait une modification

- Pour retrouver qui a modifié une ligne précise de code dans un projet: la commande **git blame**.

  *#git blame nomdufichier.extension*

- Cette commande liste toutes les modifications qui ont été faites sur le fichier ligne par ligne. À chaque modification est associé le début du **sha** du commit correspondant. Par exemple :

  *ba92e12f (khalid 2017-11-26 01:21:24 +0000 1) tata rata*

- Utiliser la commande **git show** qui vous renvoie directement les détails du commit recherché en saisissant le début de son sha :

  *#git show ba92e12f*