
git

INTRODUCTION

GIT est un logiciel de version dématérialisé, libre et gratuit (aussi appelé VCS : Versionning Control System)

Il est apparu en 2005 par Linus Torvalds, en 2010 il est devenu le logiciel de gestion de versions le plus populaire dans le développement de logiciel et web.

À quoi sert un logiciel de version ?
Il conserve l'historique des modifications de chaque fichier, ainsi il est facile de pouvoir revenir sur une version de code antérieure, permettant ainsi de corriger les éventuels problèmes.

Il facilite aussi le travail collaboratif en permettant à chaque développeur de travailler indépendant sur des « **branches** ».

FONCTIONNEMENT

GIT est hébergé en ligne, c'est là tout l'intérêt : les changements de code sont envoyés sur le serveur distant, communément appelé « **ORIGIN** ».

Ainsi il est facile de pouvoir partager du code et le récupérer entre plusieurs développeurs.

Autre intérêt : avoir toujours une sauvegarde de notre code en ligne, au cas où un incident arrive sur notre machine locale…

Dans GIT, il est possible de conserver plusieurs projets, on les appelle des « **repository** », il s'agit de l'équivalent d'un « super dossier » content votre projet, sur le serveur distant.

LES BRANCHES

Le principe de branche dans GIT, c'est comme pour un arbre :

- Le « tronc » représente la branche « principale » du code
- Les développeurs créés des « branches » depuis la branche principale
- Une branche est simplement une « révision du code » avec un nom, ici vous avez les noms des branches comme « feature », « main » ou « bug »

LES COMMIT

Un commit est représenté par les cercles au niveau des branches, ils représentent les moments où les développeurs vont « **sauvegarder** » leur code en ajoutant un message pour expliquer les changements qui on été réalisé sur la **branche locale**. Chaque commit a un « **identifiant** », permettant de revenir précisément à un commit antérieur.
Au préalable, vous devez « **ajouter** » les fichiers à envoyer dans le commit.

PUSH & PULL

Un commit ne sauvegarde les changements que sur votre **machine locale** ! Afin que le serveur distant les possède il faut les « **push** », pousser les changements locaux vers le serveur distant.
Ces changements sont envoyés vers le serveur distant, sur la branche en question : par défaut, c'est le même nom que la branche locale.

« **pull** » permet de récupérer les changements du serveur distant sur notre machine locale.

UTILISATION

Pour utiliser GIT sur notre machine locale, on doit déjà l'installer :

Pour windows : https://git-scm.com/download/win

Tout se passera en ligne de commande, vous trouverez un récapitulatif dans ce cours.

PLATEFORMES

Parmi les plateformes courantes, vous avez « GitHub », « GitLab » et « GitBucket » :

GitHub

GitLab

Bitbucket

COMMANDES

- « **git init** » : initialiser le dossier courant comme dépôt git (sur votre machine locale)
- « **git branch XXX** » : créer une branche de nom XXX
- « **git branch -l** » : affiche la liste des branches « trackées » en local
- « **git checkout XXX** » : se déplace de la branche courante vers la branche XXX
- « **git checkout -b XXX** » : créer une branche de nom XXX et se déplace dessus
- « **git add .** » : ajoute les fichiers au prochain commit
- « **git commit -m XXX** » : créer un commit de label XXX
- « **git push** » : envoie vos changements locaux sur le serveur distant
- « **git push --set-upstream origin XXX**» : si votre branche locale n'existe pas sur le dépôt distant, indique que l'on créé la branche « XXX » sur le distant et que l'on pousse les changements dessus
- « **git pull** » : récupère les changements distants sur votre local
- « **git status** » : affiche l'état de votre branche locale (fichiers modifiés etc)
- « **git clone XXX** » : permet de récupérer le repository distant sur l'URL XXX
- « **git merge XXX** » : fusionne la branche XXX sur la branche actuelle
- « **git remote add origin XXX** » : définit que le dépôt local est lié au dépôt distant de l'URL XXX