Simplon Boulogne = {Git}

Utilisation basique de git

Qu’est-ce que Git & Github ?

Git est un logiciel de **gestion de versions**. C’est un outil incontournable pour tout développeur qui se respecte pour stocker, collaborer et connaitre dans les détails toutes les modifications de code d’un **projet commun**.

Tutoriel complet d’utilisation : https://try.github.io

Inscription sur Github : https://github.com/join

Initialiser .git

Un dossier .git (caché) est **créé localement** là où vous vous situez dans votre projet en local. De préférence il faut se placer à la racine d'un projet. Il contient toutes les informations concernant **les interactions entre git et votre projet**

$ git init

Status - Staging Area

Le staging area est un fichier situé dans votre dossier .git qui stocke toutes les informations de **votre prochain commit.**

$ git status

Add or remove & ...

$ git add <files or directory>

$ git rm <files or directory>

… Commit

On peut **encore ajouter ou supprimer** des fichiers dans notre Staging Area avant de **préparer** un commit (généralement accompagné d'un message)

$ git commit -m « Mon premier commit »

Remote repositories

Remote repositories est un **dossier** sur les serveurs en ligne de Github permettant la synchronisation entre vos dossiers locaux avec le serveur.

$ git remote add <nameRemote> <urlGitHub>

Push remotely

Cette commande indique à Git **d'envoyer notre commit** local (une fois prêt et contenant les changements) **vers les serveurs en ligne.** Nous devons spécifier un message accompagnant le commit.

$ git push -u <nameRemote> <branch>

Pull remotely

Cette commande indique à Git **de récupérer les changements depuis le serveur sur notre local**. Nous devons spécifier le **nom** de notre remote et la **branche** souhaitée.

$ git pull <nameRemote> <branch>

Créer une branche

Les branches sont des **copies** (ayant comme base la branche master) permettant un travail collaboratif : en effet chaque branche contient son propre code et peuvent être **synchronisées** entre elles.

Exemples : nouvelle fonctionnalité ou la résolution d'un bug

$ git branch <name_branch>

Aller dans une branche

$ git checkout <name_branch>

Merge une branche

Synchronise votre nouvelle branche avec la **branche de base** « **master** » pour ainsi valider votre nouveau code à tout le monde.

$ git merge <branch_name>

Supprimer une branche

Une fois votre branche terminée et synchronisé avec la branche master, vous pouvez la supprimer.

```
$ git branch -d <branch_name>
```