Workshop Git/GitHub (première partie)
- Prise de notes -

Git & VCS Version Control System

Création de branches d'un fichier où chacune représente une version du fichier, et pour chaque version des branches de version représentant une évolution de chaque version et ainsi de suite...

Git est une sorte de VCS, qui est rapide, décentralisé (non relié à un seul serveur), libre et gratuit, approprié aux petits et aux grands projets, respecte la philosophie open source, et qu'on peut utiliser hors connexion.

Le workflow de Git comprend 3 étapes :
Working directory : répertoire local
Staging area : (index) intermédiaire entre le *working directory* et le repository. A partir de celui-ci, il peut y avoir une relation entre un fichier du *working directory* et un fichier sur un repository.
Repository : sorte de base de données de Git

Le but de la formation d'aujourd'hui : création d'un repository Git

Commandes de configuration de Git (à exécuter à la première utilisation)

```
git config --global user.name « name »      //pour que les commits soient associés à un nom d'utilisateur
git config -global user.email « email »     //pour définir un email. Utile si on veut travailler sur GitHub par la suite
git config -global core.editor nano         //pour définir l'éditeur par défaut qu'ouvre Git pour la modification de fichiers
git config -global color.ui true            //pour que les affichages soient colorés
```

Commandes de base

`git init` : relie le répertoire courant à Git en créant un dossier caché .git
`git status` : pour déterminer quels fichiers sont dans quel état
`git add` : ajouter un fichier à la *staging area*
`git commit -m "1st version"` : enregistrer une version avec un message explicatif
`git checkout "nom de fichier modifié"` : retour à la version précédente où on a exécuté commit
`git log` : historique des commits
`git commit -am "version..."` : *add* et *commit* en une seule commande (ne marque que sur les fichiers où l'on a déjà effectué un *add* (i.e. qui sont déjà dans la *staging area*)

Commandes se rapportant aux branches

`git checkout -b "nom branche"` : création d'une branche
`git branch` : pour montrer l'arborescence du projet
`git checkout "nom branche"` : se placer sur la branche en question
`git branch -D "nom de la branche"` : supprimer une branche

Pour fusionner deux branches :
- se placer sur la branche avec laquelle on veut fusionner
- exécuter : `git merge "nom branche à fusionner"`

Fichiers à ignorer

- Créer le fichier .gitignore
- Noter dedans les fichiers à ignorer (accepte les expressions régulières)
- Site utile pour générer les .gitignore automatiquement : http://gitignore.io

Partie GitHub

- D'abord, créer un compte sur http://github.com
- Confirmer votre compte via mail
- Se connecter sur GitHub
- Cliquer sur + (*new repository*) en haut à droite.
- Choisir un nom à votre repository et mettre une description (facultative)
- Choisir le mode public (les dépôts privés sont payants, mais GitHub en offre 5 gratuitement aux étudiants, les contacter ici pour ce faire)
- Garder décochée l'option « README...etc »

Note : Le fichier `README.md` sert à donner une description du contenu du dépôt, et des instructions de configuration et d'installation (si applicable). Il est sous le format .md (ou *markdown* : un langage simple de formatage de texte).

Connecter Git à GitHub

```
git push remote add origin https://github.com/[nom_utilisateur ]/[nom_repo].git
git push -u origin "name of branch"
```

Reprendre la commande `git push -u origin "name of branch"` à chaque fois que je veux répercuter les changements de Git à GitHub

Autres commandes

`git remote` : relie notre répertoire local avec celui sur GitHub
`git remote add "name" "url repository"` : créer un *fork* (copie) du répertoire chez soi
`git clone "url repository"` : cloner un repository en local

*** Tâche à faire ***

- Cloner (*fork*) le projet de Walid et lui envoyer une *pull request*
- Se documenter sur les *pull requests*