TD Versioning
Prise en main de Git

L'ensemble de ce TD est inspiré des documents de Sébastien Mosser, Philippe Collet, Cyril Cecchinel et Simon Urli.

0. Modalités

Le travail est individuel et est à effectuer sur Github Classroom au cours de la séance.
Tout commit non pushé après la fin du temps imparti dans la séance ne sera pas considéré.

I. Mise en place

- Git bash devrait être installé sur les machines, dans le cas contraire il peut être téléchargé ici : https://git-scm.com/downloads

- Lancer 'Git Bash' et configurer Git pour qu'il associes vos commits à votre nom et adresse e-mail:

```
git config --global user.name "Votre Nom"
git config --global user.email supermail@gmail.com
```

Si un commit ne peut être identifié comme provenant de vous il est considéré comme anonyme lors des évaluations !

- Si ce n'est pas fait, créez-vous un compte Github.
- S'enregistrer pour l'assignement Github Classroom suivant : https://classroom.github.com/assignment-invitations/bddccbaed7b773edb6d03767ab8e1478

- Aller dans le répertoire que vous souhaitez sur votre ordinateur et effectuez la commande avec l'url fournie par Github : `git clone https://github.com/.../td-git-{githubUserName}.git`
Ceci va cloner le repository distant localement sur votre machine. Allez dans le dossier en question : `cd td-git-{githubUserName}`

II. Actions basiques

- Créez ensuite un fichier Menu.txt contenant des plats de restaurant, ligne par ligne :
    - Steak tartare
    - Salade norvégienne
    - Filet de dorade
    - Bar à la citronnelle
- Vérifiez le résultat en tapant la commande `git status`
- Utilisez les commandes `git add` et `git commit` pour faire votre premier commit contenant le fichier Menu.txt, **attention à ne pas oublier le message de commit !**

    Il existe plusieurs manières de commiter en laissant un message de commit :
    - git commit -m “le message de commit”,
    - git commit sans argument vous ouvre un éditeur (vi par défaut - echap et :wq pour sortir en écrivant le message)
    - et d'autres que vous pourrez retrouver dans la documentation

- Modifiez le fichier Menu.txt et commitez ces modifications.
- Affichez l'historique des modifications et enregistrez les dans un fichier log.txt que vous ajoutez au dépôt.
- Visualisez les modifications effectuées dans Menu.txt entre le premier et le deuxième commit grâce à `git diff` et enregistrer le résultat dans un fichier `diff.txt` que vous ajoutez au dépôt.

    Sous un shell linux vous pouvez utiliser > pour que la sortie standard soit redirigée sur un fichier. Par exemple ls -l > toto.txt écrit la liste des fichiers du répertoire courant dans le fichier toto.txt.

- Committez ces deux fichiers
- Poussez ensuite les commits effectués grâce à la commande `git push -u origin master`

III. Introduction aux branches (sans git flow)

- Créez une branche develop dans le dépôt : `git branch develop`
- Basculez sur cette branche : `git checkout develop`
- Vérifiez que vous êtes sur la bonne branche : `git branch`
- Ajoutez des plats dans Menu.txt et commitez au fur et à mesure vos modifications
- Basculez sur la branche principale (master)
- Observez le fichier Menu.txt et modifiez des plats, puis commitez
- Fusionnez la branche que vous aviez créée : `git merge develop`
- Résolvez les conflits et commitez
- Pushez les changements
- Créez un tag appelé v1.0 à ce commit : `git tag -a v1.0 -m “version 1.0”`
- Visualisez le tag : `git show v1.0`

- Pushez le tag : `git push origin v1.0`

IV. Introduction à Git flow

- Initialisez git flow dans votre dépôt : `git flow init`. Conservez les choix par défaut de git flow.
- Créez une nouvelle feature dans le dépôt : `git flow feature start platX` en choisissant un nom de plat
- Ajoutez des plats dans Menu.txt
- Commitez et faites un `git push` : que se passe-t-il ? Comment résoudre le problème ?
- Terminez la feature : `git flow feature finish platX`
- Faites à nouveau un push
- Préparez une nouvelle release : `git flow feature start v2 .0`
- Faites une modification au fichier Menu.txt
- Terminez la release : `git flow feature finish v2.0`
- Résolvez les conflits, committez, faites un tag v2.0 et pushez.

Pour aller plus loin…

http://pcottle.github.io/learnGitBranching/ : des exercices interactifs sur le branching avec Git
http://nvie.com/posts/a-successful-git-branching-model/ : article de référence sur Git Flow
http://danielkummer.github.io/git-flow-cheatsheet/ : page très claire présentant l'utilisation de Git Flow

http://git-scm.com/docs : et bien sûr la doc officielle !