Git

Dans ce document, nous allons définir le contenu de la partie pratique du cours sur GIT.

Initier un projet GIT

- Nous allons tout d'abord initier un nouveau projet GIT
- Configurez le nom et l'email qui doit être utilisé par GIT
- Trouvez le fichier de configuration dans lequel les informations ci-dessus sont sauvegardées. Ou se trouve-t-il ? quel est son contenu

Créer nos premiers fichiers

- Créez deux nouveaux fichiers : [README.md](http://readme.md) et LICENCE
- Commitez ces deux fichiers dans deux commits différents
- Quel est le `sha1` généré pour ces commits
- Modifiez le fichier [README.md](http://readme.md) avec le texte suivant :

```
Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
when an unknown printer took a galley of type and scrambled it to make a type
specimen book. It has survived not only five centuries, but also the leap into
electronic typesetting, remaining essentially unchanged. It was popularised in the
 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and
more recently with desktop publishing software like Aldus PageMaker including
versions of Lorem Ipsum.
```

- Listez les modifications réalisées
- Commitez cette modification en deux commit différents : un commit par paragraphe
- Exécutez une commande permettant de voir les éléments modifiés entre le dernier commit réalisé et l'avant dernier.
- Exécutez une commande permettant de lister les deux derniers commits réalisés.
- Créez un fichier `.gitignore`, afin d'ignorer tous les fichiers se trouvant dans répertoire `bin` . Confirmez que cette configuration est fonctionnelle.

Créer des branches

- Créez une nouvelle branche `develop`

- Ajoutez un fichier `TODO.txt` avec le contenu suivant :

```
- Faire les TPs
- Envoyer le document avec toutes les réponses aux TP
```

- Commitez cet ajout
- Montrez la différence entre la branche `develop` et la branche `master`
- Mergez ces ajouts dans la branche `master`

Revenir dans le passé et gestion des conflits

- Modifiez le fichier LICENCE avec le contenu suivant :

```
    Licence APACHE
```

- Commitez cette modification
- Revenez à l'état de votre projet avant cette modification en utilisant la commande `git checkout`
- Créez une nouvelle branche `licence`
- Dans cette nouvelle branche, le fichier LICENCE soit etre vide. Ajoutez le contenu suivant :

```
    Licence MIT
```

- Commitez cette modification
- Revenez sur le dernier commit réalisé, en utilisant les référence `HEAD` ou `master`
- Mergez la branche `licence` avec la branche `master`. Vous devez avoir des conflits
- Gérez ces conflits afin de garder la version du fichier LICENCE avec le contenu `Licence MIT`

Créer des tags

- Créez un nouveau tag `v1`
- Listez les tags afin de s'assurer que le tag a bien été créé
- Dans le répertoire .git, trouvez-vous une référence de ce nouveau tag ? Si oui, quel est le contenu de ce fichier ? A quoi correspond ce contenu ?

Utiliser Gitlab

Nous allons dans cette partie versionner notre projet sur gitlab.

- Créez un nouveau projet Gitlab, sur l'instance de l'IMT

- Configurez votre repository local pour pointer vers ce repository créé sur Gitlab. Nous nommerons ce repository `origin`
- Exécutez une commande permettant de vérifier que le repository distant est bien configuré.
- Synchronisez toutes vos modifications en local sur le repository Gitlab
- Vérifiez que l'ensemble de votre historique est présent sur Gitlab
- Synchronisez également le `tag` créé précédemment.
- Faites une modification sur l'un de vos fichiers directement depuis l'interface de Gitlab, et récupérez la en local.
- Créez une `issue` quelconque, et ajoutez la référence de cette issue dans un nouveau commit. est-ce que la page dédiée à l'issue est bien mise à jour avec cette référence.