

## Git en local

Comme vous devriez le savoir, dans un projet informatique, il est indispensable d'utiliser un outil de gestion de versions (versionning). Plusieurs outils existent, et ici on a choisi Git. Le but de cette partie est donc de vous montrer comment créer un dépôt Git local (i.e., sur votre machine) et comment l'utiliser.

En utilisant Git, vous allez entendre et exploiter les termes suivants :

- Dépôt (repository) : c'est un répertoire qui contient toutes les données nécessaires pour gérer l'historique des modifications. Son contenu est modifiable via des commandes.
- Commit : Un commit concerne une sorte d'image des fichiers à un instant donné. Cela correspond à des modifications signalées. Il est caractérisé par une date, un auteur, une description, et un lien vers les précédents commit.

### 1. Mise en place d'un fichier git local:

Avant de commencer, vérifiez que Git est installé. En principe, ça doit être installé par défaut. Pour vérifier, lancez via un terminal la commande : `git --version`

Allez dans le répertoire où vous avez enregistré vos fichiers du TD précédent. En utilisant un terminal, allez dans ce répertoire et lancez les commandes suivantes :

- **git init** : Cette commande va permettre la création d'un répertoire (caché) qui s'appelle `.git`. Ce dossier va contenir toutes les modifications effectuées
- **git config -- global user.name "Toto"** : Cette commande va configurer le nom d'utilisateur à Toto. En ajoutant `global`, cette configuration s'applique à tout dépôt Git. En l'enlevant, ça s'applique sur le projet en cours.
- **git config -- global user.email [git.demo@gmail.com](mailto:git.demo@gmail.com) :** Cette commande va configurer le mail
- **git add .** : Cette commande permet d'examiner le répertoire de travail et cherche les fichiers qui ont été modifiés, ajoutés, et supprimés. Cela permet de « prendre une image » du répertoire.
- **git status** : permet de visualiser l'état de la future sauvegarde. Autrement dit, ce que vous avez capturé et que vous allez par la suite sauvegarder via un commit.
- **git commit -m "message explicitant les modifications"** : Cette commande permet de sauvegarder, via un **commit**, l'image capturée.

Donc, pour résumer les étapes que vous devriez faire sont :


- Faire vos modifications et les sauvegarder
- Faire une capture via `git add`.

- Faire la sauvegarde de la capture avec git commit. Le commit ne pourra pas se faire si une capture n'a pas été préalablement faite avec git add .
- **git log** : Cette commande permet de voir tous les commits avec leur identifiant (suite de caractères), leur auteur et leur date

```
commit 78e5234d77c978aec80b0974de0051500988626a (HEAD -> master)
Author: NG <ng.git.demo@gmail.com>
Date: Mon Oct 18 01:13:43 2021 +0200
    ajout d'un document JSON

commit ec197a4e1c341d6adda795ce49967ef919080b62
Author: NG <ng.git.demo@gmail.com>
Date: Mon Oct 18 01:10:38 2021 +0200
    ajout d'un document JSON

commit 5850e430a165c42ec2fbcea6ddc015f03d2477c8
Author: NG <ng.git.demo@gmail.com>
Date: Mon Oct 18 00:36:46 2021 +0200
    ajout d'un document XML
```



- **git checkout id\_commit** : permet de voir vos sources au commit *id\_commit* où *id\_commit* est l'identifiant du commit souhaité. Cependant, cela ne permet pas d'écraser les modifications qui ont eu lieu après le commit retrouvé. Le but est seulement de pouvoir visualiser par exemple ce qui avait lors de ce commit.
- Pour revenir à votre dernier commit, il suffit de taper la commande **git checkout master**
- **git revert id\_commit**: permet de retourner à l'état d'un commit antérieur. En exécutant cette commande, Git va vous demander de saisir un message. Par défaut, vous verrez un message Revert "message du commit à retrouver". Vous pouvez le modifier ou le garder tel quel. Sauvegarder et continuer. Vous pouvez vérifier l'état de votre projet et l'état des commit avec un git log.

### Travail demandé (1) :

En appliquant ce que vous venez de voir, étendez le document XML précédemment créé. Faites des commits.

Faites une bêtise volontaire (exemple une erreur) dans votre document. Faites un commit.

Retourner à la version qui précède la bêtise volontaire.

## Les branches :

Avec git, on a la possibilité de créer différentes branches. Une branche peut être vue comme une ligne qui reçoit des commits. L'intérêt d'utiliser différentes branches est par exemple de dédier une branche par catégorie ou fonctionnalité pour ensuite la faire rapatrier (merger) avec la branche principale. Cette branche principale est **master**.

- **git branch** : permet de visualiser les branches existantes. Par défaut la branche master est créé systématiquement (la branche principale).
- **git branch nomDeLaBrancheACréer** : cette commande permet de créer une branche nommée nomDeLaBrancheACréer
- **git checkout nomDeLaBranche** : cette commande permet de passer à la branche nomDeLaBranche. Une fois vous êtes sur la branche, vous pouvez faire vos commits. Une fois vous avez fini tous les commits de cette branche, vous pouvez la faire rapatrier à la branche master. Pour faire cela, il faut d'abord se positionner sur la branche principal (i.e., master)
- **git checkout master** : cela permet de se repositionner sur la branche master
- **git merge nomDeLaBrancheACréer** : cela permet de rapatrier (merger) la branche créée
- **git branch -d nomDeLaBranche** : permet de supprimer la branche. Cela est à faire une fois que la branche n'est plus utile

## Travail demandé (2) :

En appliquant ce que vous venez de voir, ajoutez le document JSON réalisé mais en utilisant une autre branche. Faites des modifications. A la fin, rapatrier la branche à la branche principale.