

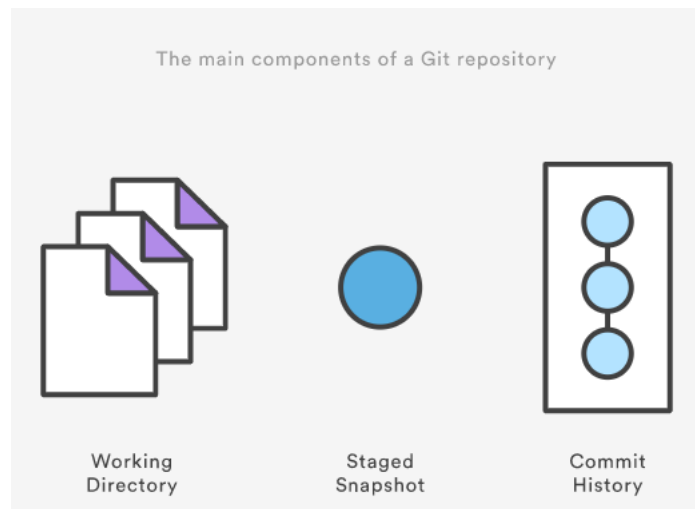
Module : M202 – Approche Agile

Filière : Développement digital – option web full stack

Activité 5.1

Objectifs :

- Manipuler les commandes git



Les principales commandes git sont :

- **git init** : crée un nouveau dépôt vide à l'emplacement courant.
- **git status** : affiche les différences entre le répertoire de travail, l'index et HEAD.
- **git log /git reflog** : affiche l'historique des commits effectués
- **git add** : ajoute des fichiers depuis le répertoire de travail vers l'index.
- **git commit** : ajoute des fichiers depuis l'index vers HEAD.
- **git config** : définit les valeurs de configuration Git au niveau local du projet (c'est la valeur par défaut, le fichier se trouve sous .git/config) ou global (au niveau de l'utilisateur du système d'exploitation sous etc/config).
- **git clone** : clone un dépôt existant local ou distant.
- **git pull** : récupère des modifications depuis un dépôt distant vers HEAD.
- **git push** : publie des modifications depuis HEAD vers un dépôt distant.

1. Créer un repository

git init crée un repository vide dans le répertoire courant

```
% mkdir git-tutorial
```

```
% cd git-tutorial
```

```
% git init
Initialized empty Git repository in /home/mh/git-tutorial/.git/
% ls -l .git
total 24
-rw-r--r--      1 mh      mh    23 Oct 26 09:14 HEAD
-rw-r--r--      1 mh      mh   111 Oct 26 09:14 config
-rw-r--r--      1 mh      mh    58 Oct 26 09:14 description
drwxr-xr-x     12 mh      mh   408 Oct 26 09:14 hooks
drwxr-xr-x      3 mh      mh   102 Oct 26 09:14 info
drwxr-xr-x      4 mh      mh   136 Oct 26 09:14 objects
drwxr-xr-x      4 mh      mh   136 Oct 26 09:14 refs
```

2. Ajouter des fichiers

git add ajoute des fichiers nouveaux ou récemment modifiés à l'index

```
% echo "Hello World" > file.txt
% git add .
```

3. Consulter l'état

Afficher la branche courante, et l'état de l'index :

```
% git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#
#       (use "git rm --cached <file>..." to unstage)
#
#       new file:   file.txt
#
```

4. Committer les modifications

```
% git commit -m 'message 1'
```

```
Created initial commit 0ba7bd8: Initial version
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 file.txt
```

```
% echo "Hello Matthieu" > file.txt
```

```
% git commit -a -m 'message 2'
```

```
Created commit 7fbf4cb: Modif
1 files changed, 1 insertions(+), 1 deletions(-)
```

5. Regarder en arrière

Plusieurs façons d'examiner l'historique des modifications

```
-
$ git log
commit cc12433d9a6ecdb721bb941a185e65e4c34b7c8a (HEAD -> master)
Author: AsmaeYouala <asmae.youala2@gmail.com>
```

Date: Thu Oct 20 15:47:07 2022 +0100

message1

commit fa7f6e0a8758d819e2adab2216a8cd23a024753e

Author: AsmaeYouala <asmae.youala2@gmail.com>

Date: Thu Oct 20 15:45:11 2022 +0100

Message2

```
-  
$ git log --oneline  
cc12433 (HEAD -> master) test  
fa7f6e0 message
```

6. Examiner les modifications

```
echo "Good bye" > file.txt  
$ git diff  
diff --git a/file.txt b/file.txt  
index 6bd8f3c..c0ee9ab 100644  
--- a/file.txt  
+++ b/file.txt  
@@ -1,1 @@  
-Hello Matthieu  
+Good bye
```

7. Corriger les erreurs, revenir à une version saine

Restaurer le working set à partir de la dernière version commitée, détruisant toutes les modifications locales :

```
$ git reset --hard  
HEAD is now at cc12433 test
```

Restaurer le working set à partir d'un commit précis

```
$ git reset --hard fa7f6e0
```