

# TP1 : Installation et utilisation de git sur Windows

## Objectifs :

- ❖ Installer git sur windows
- ❖ Tester les commandes `git init`, `git add`, `git commit`, `git log`, `git status`, `git diff`, `git reset`, `git reflog`, `git restore`,...

## GIT :

- + Git Qu'est-ce que git et pourquoi voudriez-vous l'utiliser?
- + Si vous supprimez accidentellement votre code, l'utilisation de git peut vous aider à le récupérer.
- + Si vous modifiez accidentellement votre code et cassez quelque chose, git peut le rétablir.
- + Git vous permet de partager et d'échanger facilement du code avec d'autres développeurs.
- + Si vous voulez savoir quelles modifications récentes vous avez apportées à votre code, git vous montrera.
- + Git vous permet de sauvegarder facilement votre code sur un serveur distant. Beaucoup plus de choses! Git est un logiciel qui vous permet d'effectuer un contrôle de version.
- + Les grands projets logiciels nécessitent un logiciel pour suivre tous les différentes modifications apportées à une base de code afin de suivre des éléments tels que: qui a édité un certain fichier, ce qu'ils ont changé, et comment revenir au code d'origine si nécessaire. Git fait toutes ces choses et plus encore, donc c'est pas étonnant que la plupart des grands éditeurs de logiciels utilisent git!
- + Git est un système de contrôle de version distribué et succède aux systèmes centralisés, tels que CVS et SVN. Les systèmes centralisés exigeaient tout aux utilisateurs de se connecter à un serveur centralisé avant de modifier un fichier. Comme un système distribué, chaque utilisateur a tout le code et l'historique des révisions sur l'ordinateur sur lequel il travaille.

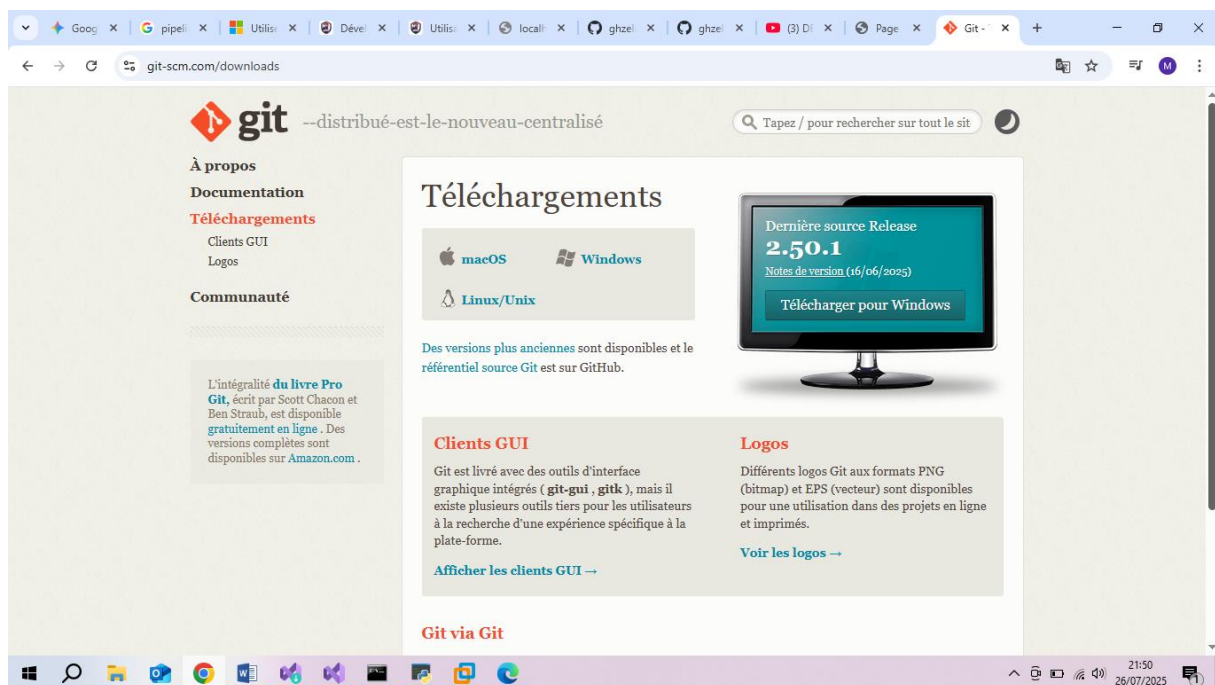
## GIT SVN

- + Git VS Svn Le logiciel Git est installé sur un poste de travail et agit comme un client et un serveur.
  - + Avec SVN, il existe un serveur et un client séparés. Avec Git, chaque développeur dispose d'une copie locale de l'historique complet des versions du projet sur sa machine individuelle.
  - + Avec SVN, seuls les fichiers sur lesquels un développeur travaille sont conservés sur la machine locale, et le développeur doit être en ligne et travailler avec le serveur. Les utilisateurs de SVN extraient les fichiers et valident les modifications sur le serveur.

- Les changements Git se produisent localement. L'avantage est que le développeur n'a pas besoin d'être connecté en permanence. Une fois tous les fichiers téléchargés sur le poste de travail du développeur, les opérations locales sont plus rapides.
- Dans le passé, les développeurs Git ayant chacun une copie de l'historique complet des versions signifiait qu'ils pouvaient facilement partager les modifications avant de les pousser vers un référentiel central.
- Désormais, tout le partage se fait dans des référentiels centraux, comme un GitHub. Et, dans le monde d'aujourd'hui, les entreprises ont des projets qui couvrent plusieurs référentiels comprenant de gros fichiers binaires. À mesure que les projets grandissent, le stockage local n'est ni vraiment réaliste ni souhaitable.
- En ce qui concerne les performances de Git par rapport à SVN, le modèle client-serveur de SVN surpasse les performances avec de fichiers et des bases de code plus volumineux.

## 1- Installer git sur windows

Pour installer git sur windows : <https://git-scm.com/downloads>



[Git pour Linux: <https://git-scm.com/download/linux-Ubuntu> (linux-mint)]

```
# sudo apt-get install git
```

La première chose à faire après l'installation de Git est de renseigner votre nom et votre adresse de courriel. C'est une information importante car toutes les validations dans Git utilisent cette information et elle est indélébile dans toutes les validations que vous pourrez réaliser :

```

user@DESKTOP-6PHHB8T MINGW64 ~
$ git config --global user.name "ghzelmarwa"

user@DESKTOP-6PHHB8T MINGW64 ~
$ git config --global user.email "ghzelmarwa@live.com"

user@DESKTOP-6PHHB8T MINGW64 ~
$ |

```

## 2- Initialisation d'un dépôt Git dans un répertoire local :

On crée un dossier tp-git c'est un repos local

Puis on crée un projet git

git init

```

user@DESKTOP-6PHHB8T MINGW64 ~ (master)
$ cd tp-git

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ ls

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ ls -la
total 8
drwxr-xr-x 1 user 197121 0 Jul 26 22:36 ./
drwxr-xr-x 1 user 197121 0 Jul 26 22:36 ../

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git init
Initialized empty Git repository in C:/Users/user/tp-git/.git/

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ ls -la
total 8
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 ./
drwxr-xr-x 1 user 197121 0 Jul 26 22:36 ../
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 .git/

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$

```

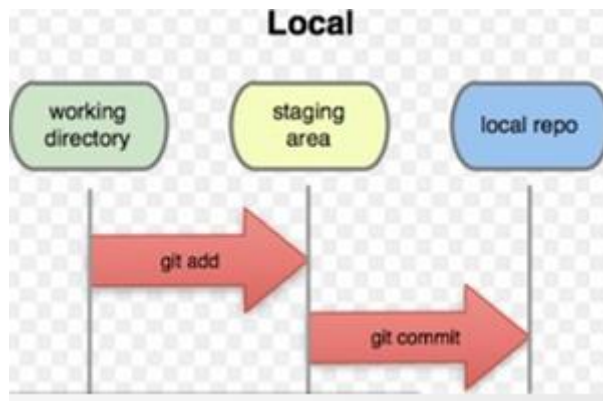
L'initialisation de la dépôt git sous le répertoire tp-git crée un nouveau sous-répertoire nommé .git qui contient tous les fichiers nécessaires au dépôt local.

```

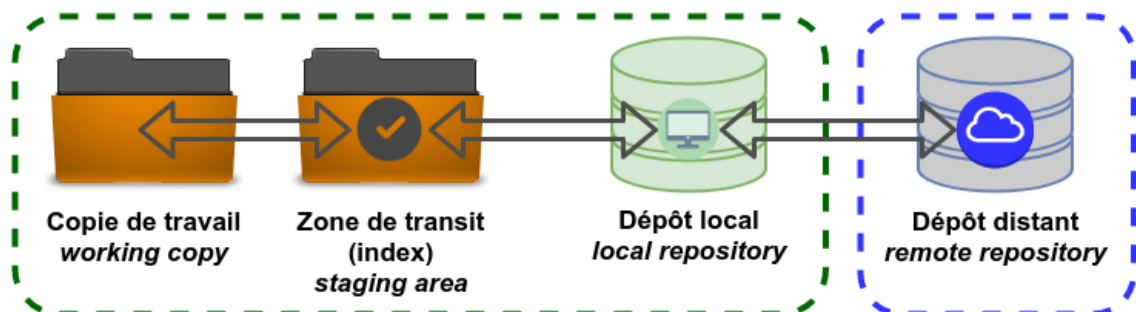
user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ ls -la .git/
total 7
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 ./
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 ../
-rw-r--r-- 1 user 197121 23 Jul 26 22:39 HEAD
-rw-r--r-- 1 user 197121 130 Jul 26 22:39 config
-rw-r--r-- 1 user 197121 73 Jul 26 22:39 description
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 hooks/
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 info/
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 objects/
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 refs/

```

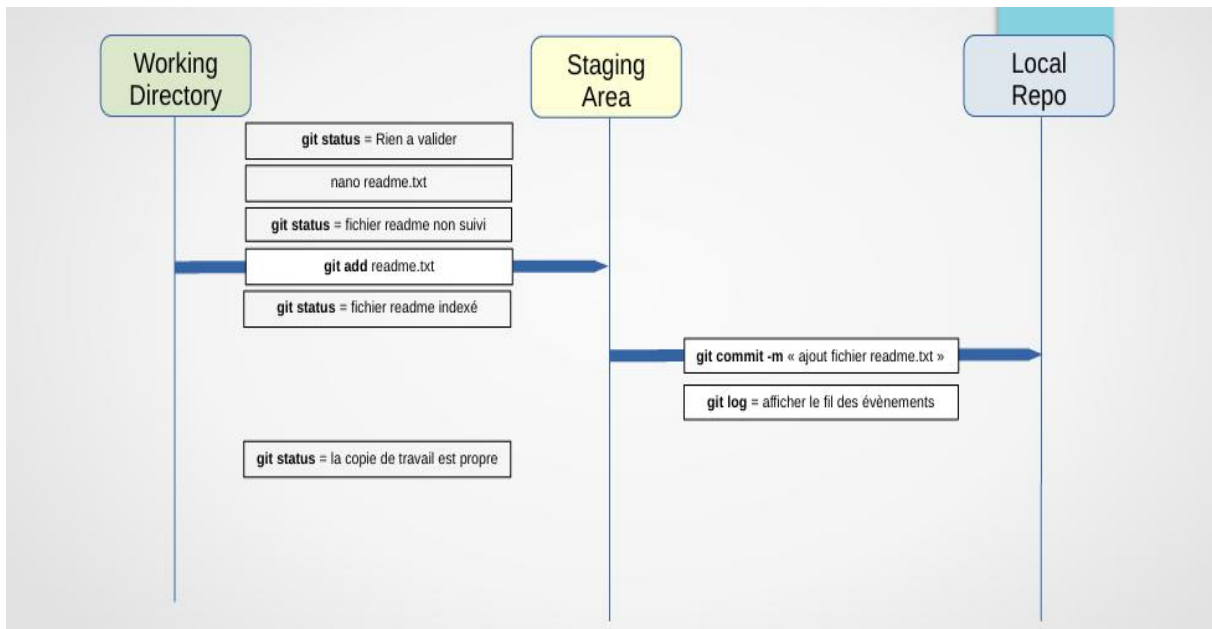
On peut travailler en individuel ou en groupe sur un repos en local (ajouter code, modifier projet, supprimer des fichiers). Si je valide une telle version je fais un push vers un dépôt distant (remote repos telque gitlab ou github) ou le projet peut être privé ou public partagé par un groupe ou tout le monde.



- ✚ **Répertoire de travail (working directory)** : contient la copie locale des sources du projet contient, à sa racine, le répertoire .git (gestion interne technique).
- ✚ **Index (staging area)** : espace temporaire utilisé pour préparer la transition de données entre le répertoire de travail et le dépôt local permet de choisir quel sous-ensemble de modifications, présentes dans le répertoire de travail, répercuter dans le dépôt local lors d'un commit
- ✚ **Dépôt local (local repos)** : contient la totalité de toutes les versions de tous les fichiers du projet, par l'intermédiaire des commits contient toutes les méta-informations : historique, logs... chaque contributeur possède son propre dépôt local



**git** 



```

MINGW64:/c/Users/user/tp-git
user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ ls -la
total 12
drwxr-xr-x 1 user 197121 0 Jul 26 22:39 ./
drwxr-xr-x 1 user 197121 0 Jul 26 22:36 ../
drwxr-xr-x 1 user 197121 0 Jul 26 22:48 .git/

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ nano readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  readme.txt

nothing added to commit but untracked files present (use "git add" to track)

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$

```

```

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git add readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git commit -m "ajouter readme.txt au projet"
[master (root-commit) 70011b4] ajouter readme.txt au projet
 1 file changed, 1 insertion(+)
 create mode 100644 readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master
nothing to commit, working tree clean

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)

```

- ✚ **git status** : pour déterminer quels fichiers sont dans quel état (répondre aux questions: qu'est-ce qui a été modifié mais pas encore indexé ? Quelle modification a été indexée et est prête pour la validation ?
- ✚ **git add** « fichier » : pour commencer à suivre la version (ajouter le fichier au projet .git)
- ✚ **Git commit – m** « message » pour valider les modifications

```

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git log
commit 70011b4a1d46153a1b8fd1b3afca2ccf15522bc5 (HEAD -> master)
Author: ghzelmarwa <ghzelmarwa@live.com>
Date:   Sun Jul 27 22:47:49 2025 +0100

    ajouter readme.txt au projet

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git log --oneline
70011b4 (HEAD -> master) ajouter readme.txt au projet

```

- ✚ **git log** : revoir le fil des événements
- ✚ **git log --oneline** : format du log en raccourci

Maintenant on va modifier le fichier readme.txt sans indexer le nouveau fichier.

Ecrire dans le fichier readme.txt la ligne : c'est la deuxième modification dans le fichier readme.txt

```

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git restore readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master
nothing to commit, working tree clean

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ cat readme.txt
bonjour c'est mon premier projet git

```

Le fichier readme.txt apparaît sous la section nommée « Modifications qui ne seront pas validées » ce qui signifie que le fichier sous suivi de version a été modifié dans la copie de travail mais n'est pas encore indexé. Pour l'indexer, il faut lancer la commande **git add**

**Git restore** : pour annuler la modification

Maintenant on va valider la modification du fichier readme.txt ( nano, git add, git commit)

```

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ nano readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ cat readme.txt
bonjour c'est mon premier projet git
hello rsi 31-2025

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git add readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git commit -m "ajout d'une deuxieme ligne dans readme.txt"
[master b8efcd1] ajout d'une deuxieme ligne dans readme.txt
1 file changed, 1 insertion(+)

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master
nothing to commit, working tree clean

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ cat readme.txt
bonjour c'est mon premier projet git
hello rsi 31-2025

```

```

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git log
commit b8efcd1272e29b474641a29d3fb2b74e20c1fe8c (HEAD -> master)
Author: ghzelmarwa <ghzelmarwa@live.com>
Date: Sun Jul 27 23:10:57 2025 +0100

    ajout d'une deuxieme ligne dans readme.txt

commit 70011b4a1d46153a1b8fd1b3afca2ccf15522bc5
Author: ghzelmarwa <ghzelmarwa@live.com>
Date: Sun Jul 27 22:47:49 2025 +0100

    ajouter readme.txt au projet

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git log --oneline
b8efcd1 (HEAD -> master) ajout d'une deuxieme ligne dans readme.txt
70011b4 ajouter readme.txt au projet

```

Partie suivante : je vais ajouter un deuxième fichier dans mon projet git.

```

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
ls
readme.txt

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
nano fichier1.txt

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
ls
fichier1.txt  readme.txt

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    fichier1.txt

nothing added to commit but untracked files present (use "git add" to track)

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
git add .
warning: in the working copy of 'fichier1.txt', LF will be replaced by CRLF the next time Git touches it

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   fichier1.txt

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
git commit -m "ajout du fichier fichier1.txt au repos local"
[master dbd4b28] ajout du fichier fichier1.txt au repos local
1 file changed, 1 insertion(+)
create mode 100644 fichier1.txt

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
git status
On branch master
nothing to commit, working tree clean

ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
|

```

Supprimer le fichier1.txt du Working Directory et le récupérer à l'aide de commande git restore

```

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ ls
fichier1.txt  readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master
nothing to commit, working tree clean

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git log --oneline
1bd4b28 (HEAD -> master) ajout du fichier fichier1.txt au repos local
08efcd1 ajout d'une deuxieme ligne dans readme.txt
70011b4 ajouter readme.txt au projet

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ rm fichier1.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ ls
readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    fichier1.txt

no changes added to commit (use "git add" and/or "git commit -a")

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git restore fichier1.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ ls
fichier1.txt  readme.txt

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ git status
On branch master
nothing to commit, working tree clean

user@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
$ |

```

Création d'un nouveau dossier projet 2 ( undo changes ) : **git reset --hard**

Undo changes : retour en arrière après avoir effectué un commit sur un fichier. Comment git réinitialiser les commits locaux ? Ce tutoriel Git se concentre sur la capacité d'annuler les modifications, d'annuler un commit et restaurer l'environnement de développement à une version antérieure plus stable. Pour parcourir l'historique des commit et retour à un état précédent, vous aurez besoin de la commande **git reset hard**

**Git reflog** : montre le numéro et la position du Header

Le header d'un commit Git est une partie essentielle de chaque enregistrement de modification dans l'historique du projet. Il contient des informations clés sur le commit, notamment l'auteur, la date, le message de commit et un identifiant unique.

```
ser@DESKTOP-6PHHB8T MINGW64 ~/tp-git (master)
cd ..

ser@DESKTOP-6PHHB8T MINGW64 ~ (master)
mkdir projet2git

ser@DESKTOP-6PHHB8T MINGW64 ~ (master)
cd projet2git/

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
ls -l
total 0

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
git init
initialized empty Git repository in C:/Users/user/projet2git/.git/

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
ls -ll
total 0

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
ls -al
total 8
-rwxr-xr-x 1 user 197121 0 Aug  5 22:14 ./
-rwxr-xr-x 1 user 197121 0 Aug  5 22:13 ../
-rwxr-xr-x 1 user 197121 0 Aug  5 22:14 .git/

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
touch f1.html

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
git add .

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
git commit -m "commit numero 1 ajout du fichier f1.html"
master (root-commit) 05b0bf8] commit numero 1 ajout du fichier f1.html
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 f1.html

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
touch f2.html

ser@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
git add .
```

```

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git commit -m "commit numero 2 ajout du fichier f2.html"
[master 4be6bac] commit numero 2 ajout du fichier f2.html
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 f2.html

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ touch f3.html

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git add .

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git commit -m "commit numero 3 ajout du fichier f3.html"
[master 79c3f44] commit numero 3 ajout du fichier f3.html
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 f3.html

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git status
On branch master
nothing to commit, working tree clean

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ ls -l
total 0
-rw-r--r-- 1 user 197121 0 Aug  5 22:14 f1.html
-rw-r--r-- 1 user 197121 0 Aug  5 22:15 f2.html
-rw-r--r-- 1 user 197121 0 Aug  5 22:16 f3.html

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git reflog
79c3f44 (HEAD -> master) HEAD@{0}: commit: commit numero 3 ajout du fichier f3.h
tml
4be6bac HEAD@{1}: commit: commit numero 2 ajout du fichier f2.html
05b0bf8 HEAD@{2}: commit (initial): commit numero 1 ajout du fichier f1.html

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git reset --hard 4be6bac
HEAD is now at 4be6bac commit numero 2 ajout du fichier f2.html

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git reflog
4be6bac (HEAD -> master) HEAD@{0}: reset: moving to 4be6bac
79c3f44 HEAD@{1}: commit: commit numero 3 ajout du fichier f3.html
4be6bac (HEAD -> master) HEAD@{2}: commit: commit numero 2 ajout du fichier f2.h
tml
05b0bf8 HEAD@{3}: commit (initial): commit numero 1 ajout du fichier f1.html

```

On remarque que le commit 3 ne figure pas comme si on a enlevé le header de commit3.

```

$
user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git log --oneline
4be6bac (HEAD -> master) commit numero 2 ajout du fichier f2.html
05b0bf8 commit numero 1 ajout du fichier f1.html

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ |

```

Le fichier **.gitignore** pour empêcher le tracking du projet3. On ajoute le nom de fichier qu'on ne veut pas le suivre par git add. Nous pouvons ajouter des patterns au fichier **.gitignore** à savoir \*.pdf

```
user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ touch .gitignore

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ nano .gitignore

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git commit -m "ajout du fichier .gitignore"
[master 6f39111] ajout du fichier .gitignore
1 file changed, 1 insertion(+)
create mode 100644 .gitignore

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git status
On branch master
nothing to commit, working tree clean

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ cat .gitignore
projet3.txt

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ touch website

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    website

nothing added to commit but untracked files present (use "git add" to track)
```

```
user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    website

nothing added to commit but untracked files present (use "git add" to track)

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ touch projet3.txt

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    website

nothing added to commit but untracked files present (use "git add" to track)

user@DESKTOP-6PHHB8T MINGW64 ~/projet2git (master)
$
```