ÉTAPE 1 – Exercice 1 Créer un dépôt Git et faire trois commits

Ce guide vous accompagne pas à pas dans la création de votre premier dépôt Git et la réalisation de trois commits. Vous apprendrez les commandes essentielles et comprendrez ce qui se passe à chaque étape.

1. Vérifier que Git est installé

Commande :

```
git --version
```

Ce qui se passe :

Cette commande demande à l'ordinateur : "Est-ce que Git est installé, et si oui, en quelle version ?"

Résultat attendu :

- Si Git est installé : `git version 2.xx.x`
- Si Git n'est pas installé : message du type `git : le terme « git » n'est pas reconnu...`

> **Si Git n'est pas installé :** il faut l'installer (voir fiche installation).

INSTALLATION

1. Télécharger Git pour Windows

Ouvre la page officielle :
https://git-scm.com/download/win

Windows va te proposer de télécharger un fichier `.exe`.

Télécharge-le dans ton dossier "Téléchargements".

2. Installer Git

Double-clique sur le fichier `.exe` téléchargé.

Pendant l'installation :

- laisse **toutes les options par défaut**
- clique sur *Next* jusqu'à *Install*
- clique sur *Finish*

L'installation prend environ 30 secondes.

3. Vérifier que Git fonctionne

Retourne dans VS Code → ouvre un terminal et tape :

```
git --version
```

Si tu vois une version (ex : `2.45.0`) :

Git est bien installé.

Configuration de Git (obligatoire)

Git a besoin de savoir qui fait les commits. Sinon, vos commit seront anonymes ou rejetés.

1. Configurer ton nom

Tape (remplace Prénom Nom) :

```
git config --global user.name "Prénom Nom"
```

2. Configurer ton email

Tape (remplace ton adresse) :

```
git config --global user.email "ton_mail@example.com"
```

Pourquoi c'est important ? Parce que Git signe chaque commit avec ton identité. C'est ce qui permet de savoir qui a fait quoi, surtout en équipe.

3.vérifier que ton nom, ton prénom et ton email sont bien configurés dans Git.

tu tapes juste cette commande dans ton terminal VS Code :

```
git config --global --list
```

Git va te renvoyer **toutes** les configurations globales enregistrées sur ton PC.

Tu dois voir quelque chose comme :

```
user.name= Alexandre Djordjevic
user.email=tonmail@example.com
```

Si les deux lignes apparaissent

Tout est nickel, Git sait qui tu es.

Si une des deux lignes n'apparaît pas

Ça veut dire que Git n'a pas encore l'info.
Tu refais alors :

```
git config --global user.name "Prénom Nom"
git config --global user.email "mail@example.com"
```

Astuce encore plus rapide

Tu peux tester individuellement :

```
git config --global user.name
```

Et :

```
git config --global user.email
```

Git affichera uniquement la valeur configurée pour chacun.

2. Se placer dans un dossier propre

Commande :

```
cd C:\Users\ton_identifiant
```

Ce qui se passe :

`cd` signifie ***change directory*** : tu te déplaces dans un autre dossier. Ici, tu vas dans ton dossier utilisateur, un endroit stable pour travailler. Ex: **C:\Users\acidj**

Pourquoi c'est important :

Git fonctionne mieux dans un dossier simple, sans synchro cloud ou droits bizarres.

3. Créer un dossier de projet

Commande :

```
mkdir mon-projet-git
```

Ce qui se passe :

`mkdir` crée un nouveau dossier vide nommé `mon-projet-git`. C'est dans ce dossier que tu vas faire tout le travail Git.

4. Entrer dans le dossier du projet

Commande :

```
cd mon-projet-git
```

Tu es maintenant dans ton projet. Tu peux vérifier ce qu'il contient :

```
ls
```

Résultat attendu : aucun fichier pour l'instant (dossier vide).

5. Initialiser Git dans ce dossier

Commande :

```
git init
```

Ce qui se passe :

Tu demandes à Git de transformer ce dossier en "projet Git". Git crée un dossier caché `.git` qui contient tout l'historique.

Pourquoi :

Sans `git init`, Git ne "voit" pas ton projet, il ne suivra aucune modification.

Tu peux vérifier l'état :

```
git status
```

Résultat attendu :

- On branch main
- No commits yet
- nothing to commit

6. Créer le fichier README.md (dans le terminal)

Commande :

```
echo "# Mon premier projet Git" > README.md
```

Ce qui se passe :

- `echo` écrit le texte `# Mon premier projet Git`
- `>` crée un fichier (ou l'écrase s'il existe déjà)
- `README.md` est le nom du fichier

Tu crées donc un fichier README.md contenant une première ligne de texte.

Tu peux vérifier :

```
ls
```

Tu dois voir : `README.md`.

Et pour lire son contenu :

```
cat README.md
```

7. Regarder ce que Git voit

Commande :

```
git status
```

Résultat attendu :

- Untracked files:
- README.md en rouge.

Explication :

Rouge signifie que Git voit le fichier, mais qu'il ne le suit pas encore. À ce stade, le fichier existe, mais Git ne l'inclura dans aucune "photo" tant que tu ne lui demandes pas.

8. Ajouter le fichier au staging (git add)

C'est ici que tu EXPLIQUES le STAGING.

Commande :

```
git add README.md
```

Ce qui se passe :

Tu dis à Git : "Je veux que README.md fasse partie de la prochaine version enregistrée."

C'est le rôle de la **staging area** : c'est une zone intermédiaire où tu poses les fichiers que tu veux inclure dans le prochain commit. On peut la voir comme une table où tu poses les objets avant de prendre une photo.

Pourquoi :

Git ne prend jamais tout automatiquement. C'est toi qui choisis quels fichiers intégrer dans le prochain commit.

9. Vérifier que le fichier est en staging

Commande :

```
git status
```

Résultat attendu :

- Changes to be committed:
- new file: README.md en vert.

Explication :

Vert signifie que le fichier est prêt pour le commit. Il est maintenant dans la staging area, en attente d'enregistrement.

10. Faire le premier commit

C'est ici que tu EXPLIQUES le SHA.

Commande :

```
git commit -m "Création du fichier README"
```

Ce qui se passe :

- `git commit` enregistre une nouvelle version du projet
- `-m` permet d'ajouter un message
- `"Création du fichier README"` décrit ce que tu as fait

Git renvoie quelque chose comme :

```
[main (root-commit) a1b2c3d] Création du fichier README
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

Le SHA :

Dans `[main (root-commit) a1b2c3d]`, la partie `a1b2c3d` est le début du SHA du commit. Le SHA est une sorte d'empreinte unique pour ce commit, comme une plaque d'immatriculation. Il permet à Git d'identifier précisément cette version, même s'il y en a des centaines.

11. Faire le deuxième commit

1. Modifie le fichier README.md et ajoute dessous :

```
## Description

Ceci est mon premier projet versionné avec Git.
```

1. Ajoute les modifications au staging :

```
git add README.md
```

1. Crée un nouveau commit :

```
git commit -m "Ajout de la description du projet"
```

À chaque fois, Git crée une nouvelle "photo" avec une nouvelle empreinte (SHA).

12. Faire le troisième commit

1. Modifie encore README.md et ajoute :

```
## Tâches

- Initialiser Git
- Faire trois commits
- Apprendre les branches
```

1. Ajoute au staging :

```
git add README.md
```

1. Enregistre un nouveau commit :

```
git commit -m "Ajout de la liste des tâches"
```

13. Afficher l'historique des commits (et revoir les SHA)

Commande :

```
git log --oneline
```

Résultat attendu :

```
e31a12b Ajout de la liste des tâches
a22b901 Ajout de la description du projet
aa12bb3 Création du fichier README
```

Explication :

- chaque ligne = un commit
- à gauche : le début du SHA (identifiant unique)
- à droite : ton message de commit

Tu peux lire l'histoire de ton projet comme une liste d'actions.

14. Erreurs fréquentes

Problème : fatal: not a git repository

Cause : Tu n'es pas dans le bon dossier.

Solution :

```
cd C:\Users\ton_identifiant\mon-projet-git
```

Erreurs fréquentes (suite)

Problème : nothing to commit, working tree clean

Cause : Tu n'as rien modifié depuis le dernier commit, ou tu as oublié `git add`.

Solution : Modifier le fichier, puis :

```
git add README.md
git commit -m "Message"
```

Erreurs fréquentes (fin)

Problème : git : le terme « git » n'est pas reconnu

Cause : Git n'est pas installé ou mal installé.

Solution : Installer Git (voir la partie installation).