Introduction pratique à Git

📌 **Objectif** : Découvrir et manipuler les bases de Git en 2 heures à travers un support écrit structuré et des exercices pratiques.

---

1. Introduction

🔹 **Qu'est-ce que Git ?**

Git est un outil de gestion de versions (ou VCS - Version Control System) est un logiciel permettant de suivre l'évolution des fichiers d'un projet au fil du temps. Il est principalement utilisé en développement logiciel pour :

✅ **Sauvegarder** différentes versions d'un fichier ou d'un projet.
✅ **Tracer les modifications** (qui a changé quoi et quand).
✅ **Collaborer** à plusieurs sans écraser le travail des autres.
✅ **Revenir en arrière** en cas d'erreur ou de problème.

Git est aussi un systèmes distribués **(DVCS - Distributed Version Control System)**

- Chaque utilisateur possède une copie complète du projet et de son historique.
- Permet de travailler hors ligne et d'effectuer des modifications en local avant de les synchroniser.
- Avantages : robustesse, rapidité et meilleure gestion du travail en équipe.
- Suivi des modifications et collaboration sur du code.

🔹 **Installation et configuration initiale rapide**

- Installation (Git-SCM.com) **apt install git**
- Obtenir la version installée **git --version**
- Configuration initiale :

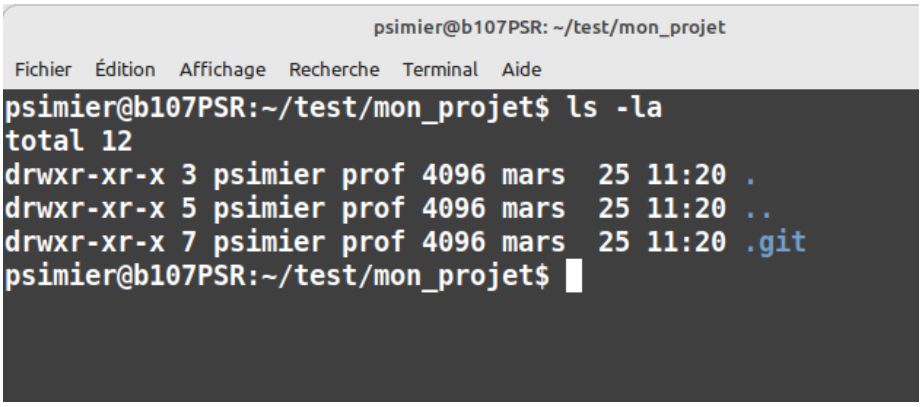
```
git config --global user.name "Votre Nom"
git config --global user.email "votre@email.com"
git config --global init.defaultBranch main
```

2. Premiers Pas avec Git

🔹 Initialisation et premiers commits

1️⃣ Créer un dépôt Git

```
git init mon_projet
cd mon_projet
ls -la
```



2️⃣ Ajouter un fichier et le suivre

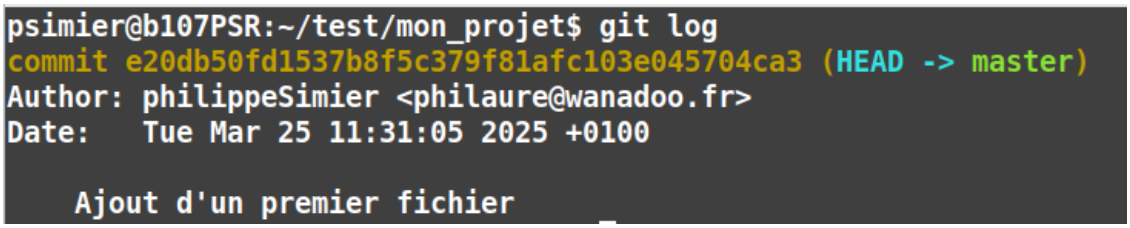
```
echo "Hello Git!" > fichier.txt
git status
```

```
psimier@b107PSR:~/test/mon_projet$ git status
Sur la branche master

Aucun commit

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
        fichier.txt

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez
"git add" pour les suivre)
psimier@b107PSR:~/test/mon_projet$
```

```
git add fichier.txt
git commit -m "Ajout du premier fichier"
```

3️⃣ Revoir l’état du dépôt

```
git status
git log
```



✏️ **Exercice** : Créez un dépôt, ajoutez un fichier `readme.txt` et validez-le.

---

3. Travail en équipe : GitHub

🔹 **Connexion avec un dépôt distant**

1️⃣ Cloner un projet existant

```
git clone https://github.com/utilisateur/projet.git
```

2️⃣ Envoyer ses modifications

```
git add .
git commit -m "Modification du fichier"
git push origin main
```

3️⃣ Récupérer les mises à jour du projet

```
git pull origin main
```

✏️ **Exercice** : Forkez un dépôt GitHub, clonez-le, modifiez un fichier, puis poussez votre modification.

---

4. Gestion des Branches

🔹 **Création et fusion de branches**
1️⃣ **Créer et basculer sur une nouvelle branche**

```
git branch dev
git checkout dev
```

2️⃣ **Fusionner les modifications**

```
git checkout main
git merge dev
```

3️⃣ **Gérer un conflit (si nécessaire)**

```
git status
git diff
```

✏️ **Exercice** : Créez une branche "feature" et ajoutez un nouveau fichier. Fusionnez-la dans `main`.

---

5. Correction et Historique (15 min)

🔹 **Annuler une modification**

```
git checkout -- fichier.txt
```

🔹 **Revenir à un état précédent**

```
git reset --hard <commit_id>
```

✏️ **Exercice** : Faites un commit, puis essayez de l’annuler.

---

6. Conclusion & Ressources (15 min)

✅ **Récapitulatif** : Commandes essentielles de Git
✅ **Bonnes pratiques** :

- Faire des commits clairs et fréquents
- Utiliser des branches pour séparer les fonctionnalités
- Toujours récupérer les mises à jour (`git pull`) avant de pousser (`git push`)
  ✅ **Ressources complémentaires** :
- Documentation Git
- GitHub Learning Lab