Voici quelques commandes Git :

- **git init :** permet de créer un nouveau référentiel Git dans un dossier local.

- **git clone :** permet de cloner un référentiel distant existant sur votre machine locale.

- **git fetch :** permet de récupérer les dernières modifications du référentiel distant sans les fusionner avec votre référentiel local

- **git pull :** permet de récupérer les dernières modifications du référentiel distant sur votre branche locale et de les fusionner.

- **git add <fichier>:** permet d'ajouter des fichiers à l'index pour les inclure dans le prochain commit.
- **Git reset <fichier>:** retirer le fichier du staging (prochain commit)

- **git branch <nom-de-la-nouvelle-branche> <nom-de-la-branche-de-depart> :** création d'une nouvelle branche à partir d'une autre branch

- **git commit :** permet de valider les modifications apportées aux fichiers qui ont été ajoutés à l'index. Elle crée un nouveau commit contenant les modifications et un message décrivant les changements.
- **Git commit -m <message de commit> :** fait un commit en incluant un message
- **Git commit -a :** add & commit tous les fichiers modifiés ou supprimés

- **git push :** permet de pousser les commits locaux vers un référentiel distant.

- **Git merge <branche> :** pour fusionner l'historique de la branche spécifiée dans la branche actuelle

- **git tag :** est utilisée pour marquer des points spécifiques dans l'historique des commits de votre projet Git.

Les tags sont généralement utilisés pour identifier les versions stables ou les versions importantes de votre projet

Git switch "nom de la branche" : changer de branche de travail

Config de git :
Git config --list
Git config --global user.mail "adresse mail de l'utilisateur git" : modifie l'adresse mail de l'utilisateur git
Git config --global user.name "nom de l'utilisateur git" : modifie le nom d'utilisateur qui sera utilisé pour la journalisation de git

Journalisation :
Git log : journalisation des évènements sur git
Git log --oneline : condensé plus lisible rapidement

Git log --oneline --graph : visualisation avec les branches

```
ludov@Pc-Ludo MINGW64 ~/OneDrive/Bureau/Solyti/Vscode/Git/hello-world (main)
$ git log --oneline
131517a (HEAD -> main, origin/main, origin/HEAD, test) Modif readme + ajout test
6d13cae (origin/test) Merge branch 'test' into 'main'
1610698 Update README.md
41546bb Merge branch 'test' into 'main'
8367ccf Update README.md
975b438 Initial commit
```

Comparaison :
Git diff : permet de comparer des différents commits et d'afficher les modifications qui ont eu lieu
Git diff ID commit 1 ID commit 2
Git diff branche1..branche2

```
ludov@Pc-Ludo MINGW64 ~/OneDrive/Bureau/Solyti/Vscode/Git/hello-world (main)
$ git diff 6d13cae 131517a
diff --git a/README.md b/README.md
index 2aeb155..fe75cb9 100644
--- a/README.md
+++ b/README.md
@@ -2,6 +2,7 @@

 test
 test2
+test3

 ## Getting started

diff --git a/test.txt b/test.txt
new file mode 100644
index 0000000..e69de29
```

git tag : Cette commande affiche tous les tags existants dans votre référentiel Git.
git tag v1.0.0 : Cette commande crée un tag léger appelé v1.0.0 pour le commit actuel.
git tag -a v1.0.0 -m "Version 1.0.0" : Cette commande crée un tag annoté appelé v1.0.0 avec un message de tag Version 1.0.0 pour le commit actuel.
git push origin v1.0.0 : Cette commande pousse un tag spécifique (v1.0.0 dans cet exemple) vers le référentiel distant.
git tag -d v1.0.0 : supprimer un tag local
git push --tags : pour les tags vers le dépôt distant

Ressources : https://www.commentcoder.com/commandes-git/