Fiche d’utilisation de Git :

À la fin de ce document, vous serez capable de :

- installer et configurer Git et GitHub
- utiliser les commandes de base de Git
- corriger les erreurs courantes sur GitHub

1/ Gestionnaire de versions et explication :

Contrôle de version :

Git à un contrôle de versions qui permet de suivre les modifications, quelle modification a été faite, par qui et pourquoi. Il le fait automatiquement et s’il y a un problème, on peut revenir facilement sur une version précédente.

Le contrôle de version permet donc :

- Revenir à une version précédente en cas de problème
- Suivre l’évolution de votre code étape par étape
- Travailler à plusieurs sans risquer de supprimer les modifications des autres collaborateurs

Git est un gestionnaire de versions, vous pouvez l’utilisez pour créer un dépôt local et gérer les versions de vos fichiers.

Un dépôt :

Un dépôt est comme un dossier qui conserve un historique des versions et des modifications d’un projet. Il est essentiel pour travailler en équipe ou collaborer à un projet open source.

- Un dépôt local est l’endroit où l’on stocke, sur sa machine, une copie d’un projet, ses différentes versions et l’historique des modifications.

- Un dépôt distant est une version dématérialisée du dépôt local, que ce soit sur Internet ou sur un réseau. Il permet de centraliser le travail des développeurs dans un projet collectif.

2/ Créez un compte GitHub

Aller sur https://github.com pour s'inscrire : clicker sur Sign up puis renseigner un email, un mot de passe et un nom d'utilisateur.

Vous pouvez consulter votre tableau de bord personnel pour :

- suivre les problèmes et extraire les demandes sur lesquelles vous travaillez ou que vous suivez
- accéder à vos principaux repositories et pages d'équipe
- rester à jour sur les activités récentes des organisations et des repositories auxquels vous êtes abonné

- Pour créer un projet, il suffit de cliquer sur "Start a project"
- L'onglet Pull requests permet de faire des demandes de modifications réalisées sur le code
- Via Explore, vous pourrez trouver de nouveaux projets open source sur lesquels travailler, en parcourant les projets recommandés, en vous connectant à la communauté GitHub et en recherchant des repositories par sujet ou par libellé

Pour mettre vos projet sur GitHub, vous devez créer un repository.

Pour créer votre propre dépôt, vous devez créer un repository :

Choisissez un nom de dépôt simple

Choisissez si le dépôt est publique ou privé

README est un fichier qui indique les informations clés de votre projet : description, environnement à utiliser, dépendances possibles et droits d'auteurs. C'est un peu comme le mode d'emploi de votre projet.

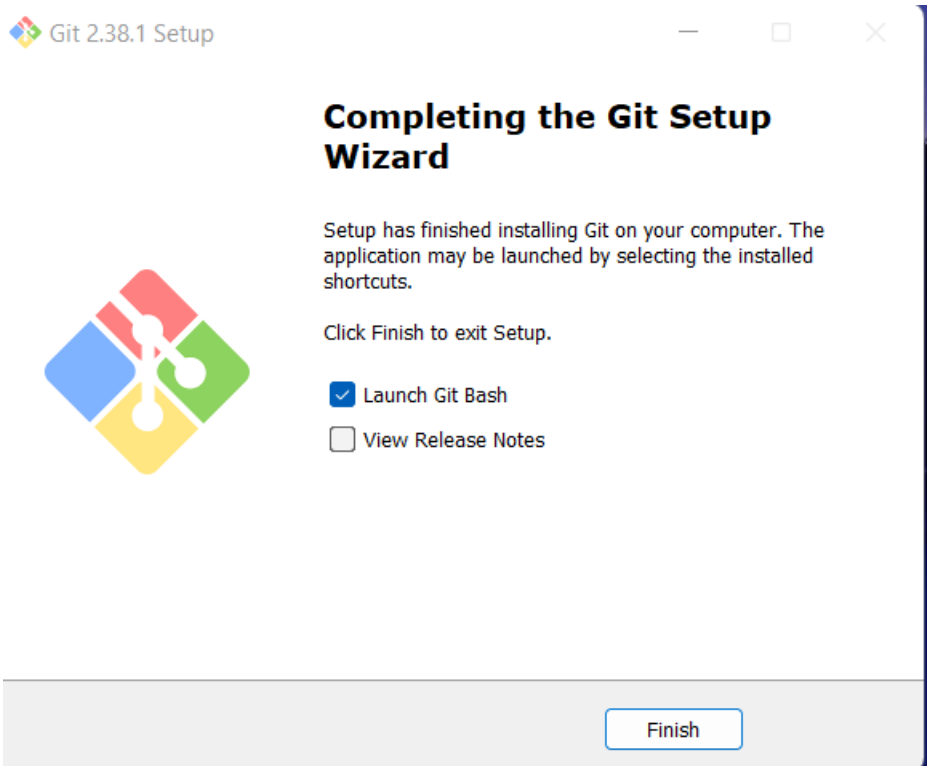
Gitignore est un fichier qui permet d'ignorer certains fichiers de votre projet Git

Sommaire :

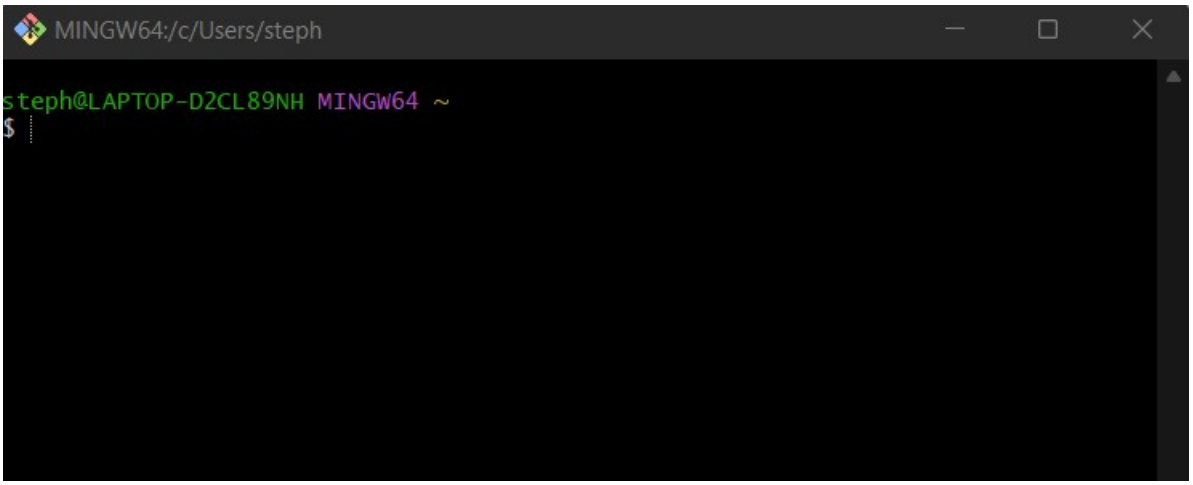
3/ Télécharger, installer et utiliser Git

Pour télécharger Git allez sur https://git-scm.com/downloads

Faite Next et laisser les paramètres par défaut puis cocher Launch Git Bash

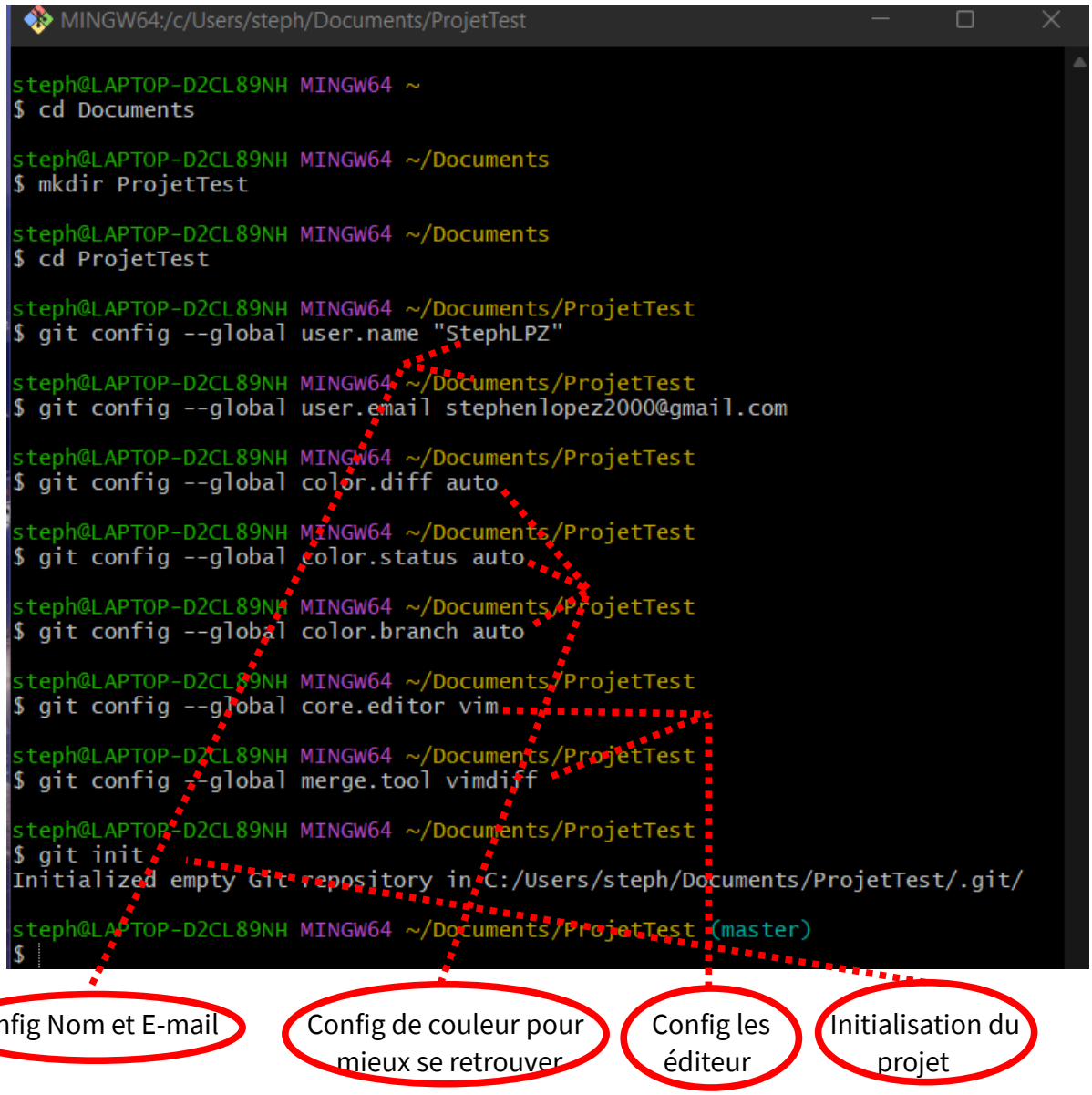


Puis Git Bash se lance



Initialisation d'un nouveau dépôt Git :

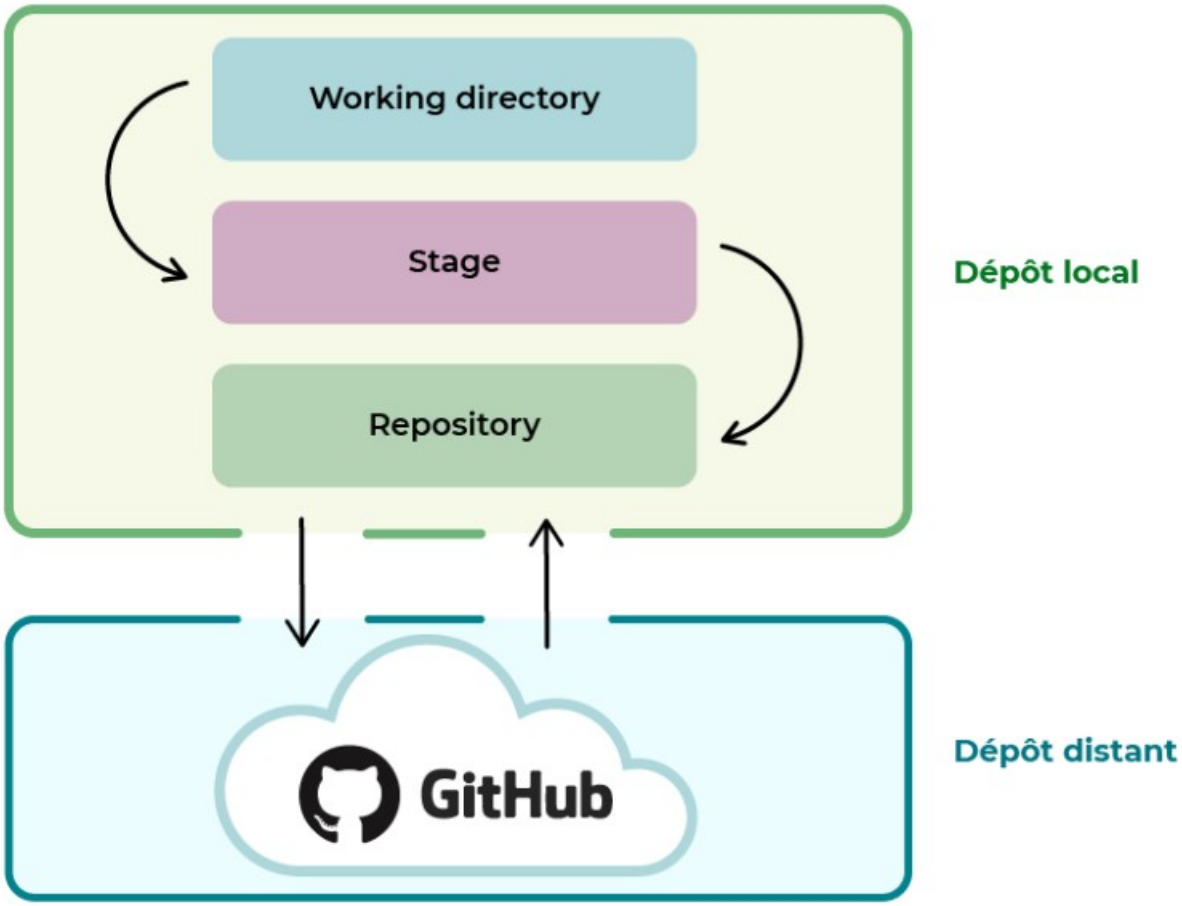
Commande utile :
cd : permet de ce déplacer dans les dossiers de votre ordinateur
mkdir : permet de créer un dossier
git config –-global : permet de configurer tout les prochain projet Git
si vous ne mettez pas – devant le global ca va configurer que le projet sur lequel vous travaillez



Pour vérifier que tout les paramètres sont bien passés faite : git config --list

Fin de la configuration et de l'initialisation du premier projet/dépôt Git.

Travaillez depuis votre dépôt local Git :



Le Working directory : Cette zone correcpond au dossier du projet sur votre ordinateur.

Le Stage : Cette zone est un intermédiaire entre le working dorectory et le repository. Elle représente tous les fichiers modifiés que vous souhaitez voir apparaître dans votre prochaine version de code.

Le Repository : Lorsque l'on crée de nouvelles versions d'un projet, on les stockes dans cette zone.

Exemple :

un projet est composé de 3 fichiers : fichier1, fichier2, fichier3
nous faisons une modification sur le fichier1 et le fichier2 depuis le working dorectory et on veux sauvegarder la nouvelle version donc on la stocke dans le repository, on va faire cela avec un fichier HTML et un fichier CSS.

Créer des fichiers et les enregistrer :

Pour créer un fichier HTML faite : **touch index.html et/ou touch index.css**
On peut voir que votre index html est en rouge, ça veut dire c'est un fichier qui a été créé ou modifié ou supprimé mais pas encore indexé, pour ce faire, faite **git add index.html**

```
steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ touch index.html

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git status
On branch master

No commits yet

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

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

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$
```

Le fichier index.html est maintenant en vert, il est maintenant dans index ou stage.

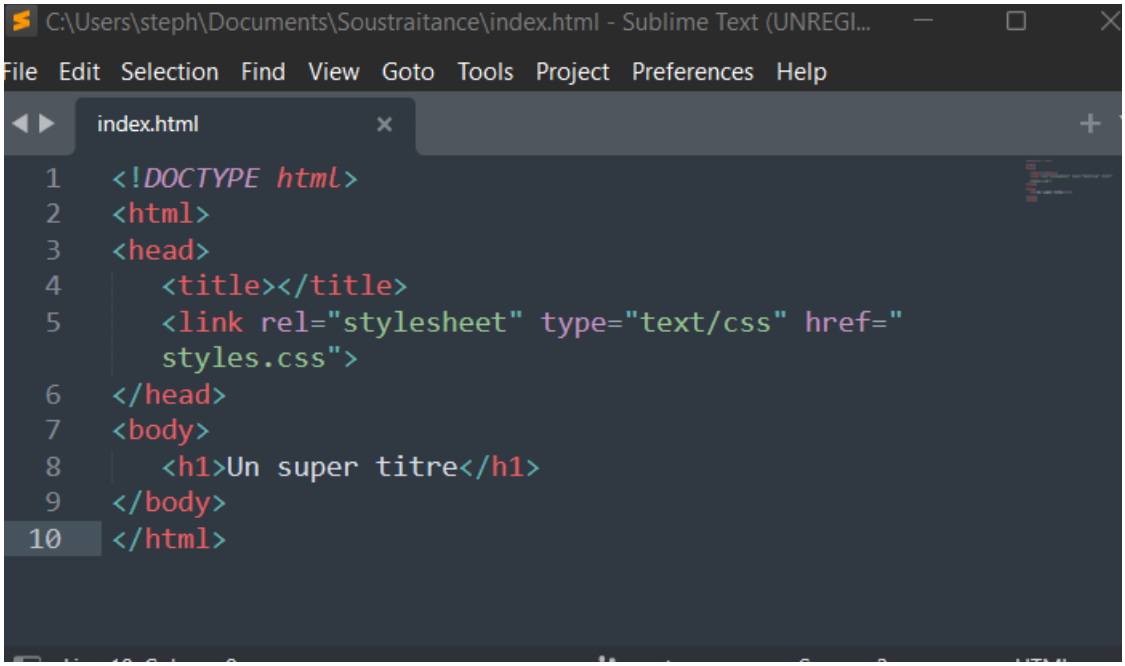
```
steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git add index.html

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git status
On branch master

No commits yet

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


steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
```

A chaque fois que vous modifierai votre dossier vous devrez faire un **git add** pour prendre en compte les modifications. Maintenant que le fichier est dans le stage, vous devez le commiter. C'est pour enregistrer, ça va faire passer votre fichier du stage vers le repository :
git commit -m "Ajout du fichier index.html" .
Vous avez créé la première version de votre projet. Si vous avez écrit **git commit**, ecrivez votre message et faite Echap puis rentrer **:x**.

```
steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git commit -m "Ajout du fichier index.html"
[master (root-commit) 6623153] Ajout du fichier index.html
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
```

Vous vous situez maintenant sur la branche principal de votre projet soit la branche master.

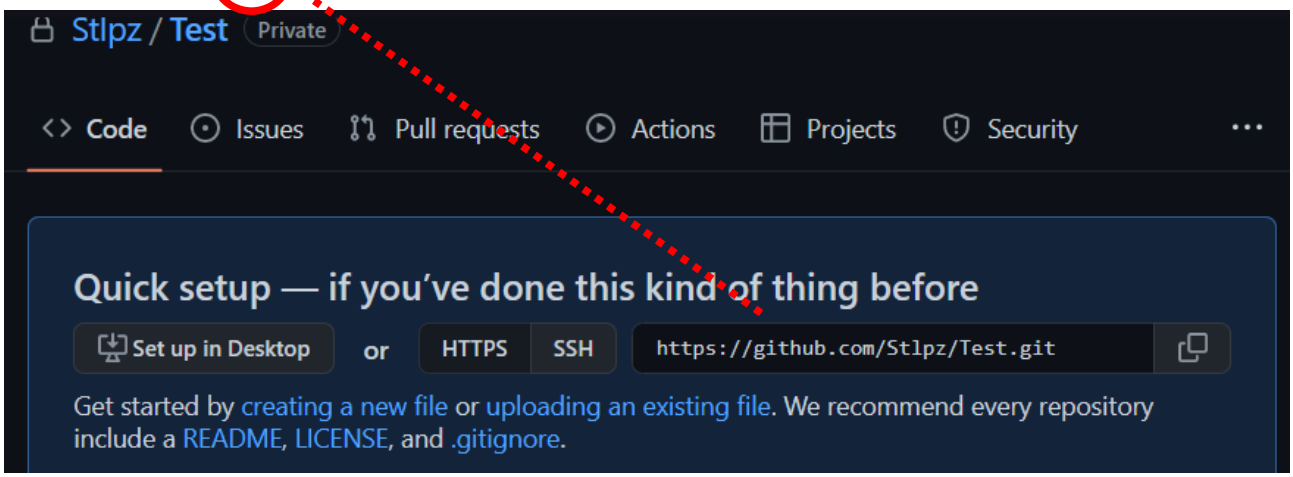
Vous pouvez créer un fichier htlm ou css via Sublime Text par exemple.
Ensuite quand le fichier htlm est terminé vous devez le add et le commit pour l'enregistrer correctement dans votre repository.




```
steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (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:   index.html

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

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git add index.html

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html


steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git commit "Debut d ecriture"
error: pathspec 'Debut d ecriture' did not match any file(s) known to git

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git commit -m "Debut d ecriture"
[master f667a75] Debut d ecriture
 1 file changed, 10 insertions(+)
```

Relier le dépôt local et le dépôt distant :

git remote add "...." le lien est sur le repository que vous aurez créé sur GitHub



git branch -M main
git push -u origin

Après c'est deux commande le dépôt local et relié au dépôt distant, les commits seront envoyés sur le repository vers le dépôt distant GitHub

```
steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git remote add
usage: git remote add [<options>] <name> <url>

    -f, --fetch           fetch the remote branches
    --tags                import all tags and associated objects when fetching
                          or do not fetch any tag at all (--no-tags)
    -t, --track <branch>  branch(es) to track
    -m, --master <branch>
                          master branch
    --mirror[=(push|fetch)]
                          set up remote as a mirror to push to or fetch from


steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git remote add origin https://github.com/Stlpz/Test.git

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (master)
$ git branch -M main

steph@LAPTOP-D2CL89NH MINGW64 ~/documents/Soustraitance (main)
$ git push -u origin main
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (13/13), 1.29 KiB | 659.00 KiB/s, done.
Total 13 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Stlpz/Test.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

La commande **git push origin main** envoie la nouvelle version sur le dépôt distant.

Comment utiliser les branches :

Pour créer une nouvelle version de votre projet sur une autre branche :

git branch : permet de voir la liste de toute vos branche (l'étoile signifie que c'est la branche sur laquelle vous vous situez actuellement et que c'est là que vous faites vos modifications.
git branch "nom_de_la_branch" : permet de créer une branche
git checkout "nom_de_la_branch" : permet de changer de branche
touch "nom_de_la_branch"."nomfichier" : permet de créer un fichier dans la nouvelle branche faite
git add "nom_de_la_branch"."nomfichier" : permet d'indexer le fichier
git commit -m "message" : permet de commit le fichier et de l'enregistrer
git push -u origin "nom_de_la_branch" : permet de créer la branche "nom_de_la_branch" sur votre dépôt distant

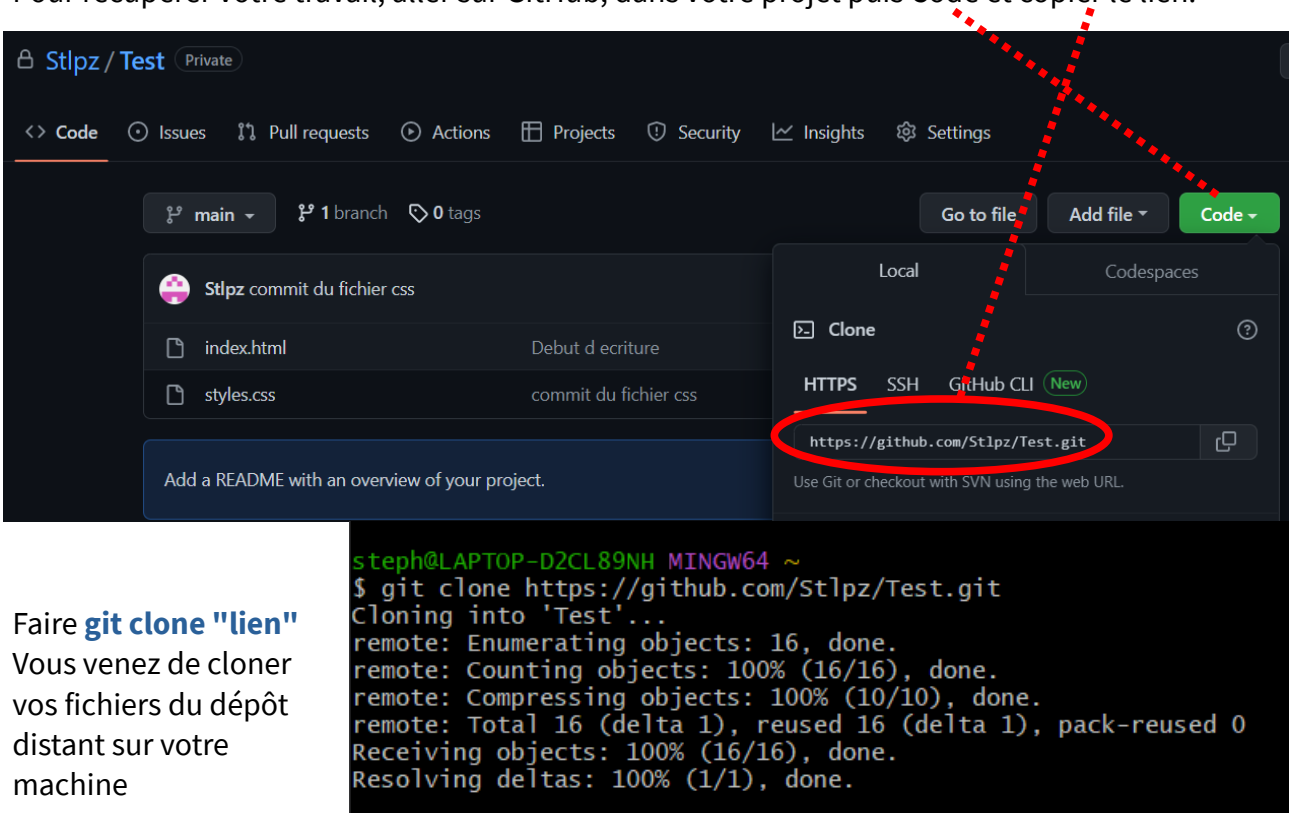
Comment fusionner la branche créer avec la branche master :

Placez vous sur la branche sur laquelle vous voulez fusionner (la master) :

Git merge "nom_de_la_branch" : permet de fusionner les branches

Comment accéder à un dépôt distant et le copier en local :

Pour récupérer votre travail, aller sur GitHub, dans votre projet puis Code et copier le lien.



Faire **git clone "lien"**
Vous venez de cloner vos fichiers du dépôt distant sur votre machine

```
steph@LAPTOP-D2CL89NH MINGW64 ~
$ git clone https://github.com/Stlpz/Test.git
Cloning into 'Test'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 16 (delta 1), reused 16 (delta 1), pack-reused 0
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (1/1), done.
```

git pull origin main : permet de récupérer le travail que quelqu'un a fait sur la branche.

git branch -d "nom_de_la_branch" : permet de supprimer une branche déjà créée.

git stash : permet de mettre de coté les modifications faite avant d'avoir commit pour les réutiliser plus tard (pour vérifier si la branche master est propre, faite git status)

Si vous voulez récupérer vos modifications sur une nouvelle branche, créer cette branche, placez vous dessus et faite :

git stash list : pour voir la liste des remises
git stash apply stash@{,,,} : pour appliquer la remise qui nous intéresse

Si vous avez fait des modifications et que vous avez commit avec des erreurs, pour récupérer vos données faite :

git log : permet d'analyser les derniers commit

git reset --hard HEAD^ : supprime le dernier commit de la branche principale, le HEAD^ indique que c'est bien le dernier commit que nous voulons supprimer

Il faut créer une nouvelle branch (git branch **"nom_de_la_branch"**), basculer sur cette branche (**git checkout"nom_de_la_branch"**)

git reset --hard "8 premiers caractères du git log" : applique le commit sur la nouvelle branche

Comment modifier un mauvais message de commit :

git commit --amend -m"Nouveau message" : modifie le message du dernier commit

Oublie d'un fichier dans le dernier commit :

git commit --amend --no-edit : envoie le dernier fichier ajouté dans le dernier commit envoyé
Comment supprimer le dernier commit :

git revert HEAD^ : revert le dernier commit en créant un nouveau commit d'annulation

L'accès à distance ne fonctionne pas :

Il faut créer une paire de clés SSH, on vous demandera un nom de clef puis un mot de passe :
ssh-keygen -t rsa -b 4096 -C "votre adresse mail rentrez au début de l'autentification"
aller dans votre dossier, afficher les fichiers cachés puis aller dans ssh (2 fichiers, une clef privée et une clef publique)

ouvrir la clef publique dans un éditeur de texte et copier là, aller dans votre compte GitHub → Setting → ssh and CPG keys → News Ssh key, choisissez un titre et copier votre clef publique dans key

Les différents type de git reset :

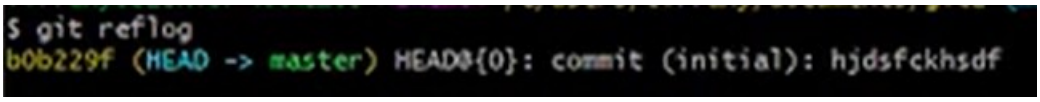
git reset --soft : commande qui permet de revenir à vos fichiers avant le commit et ne supprime aucun fichier

git reset --mixed : commande qui permet de revenir en arrière jusqu'à votre dernier commiter aucun suppression effectuée

git reset "Commit ciblé" --hard : commande qui fait des modifications qui ne peuvent pas être annuler et qui supprime le dernier commiter

Comment faire pour revenir à un commit particulier :

git reflog : permet de récupérer la list des logs



git checkout "b0b229f" : permet de revenir à un précédent commit