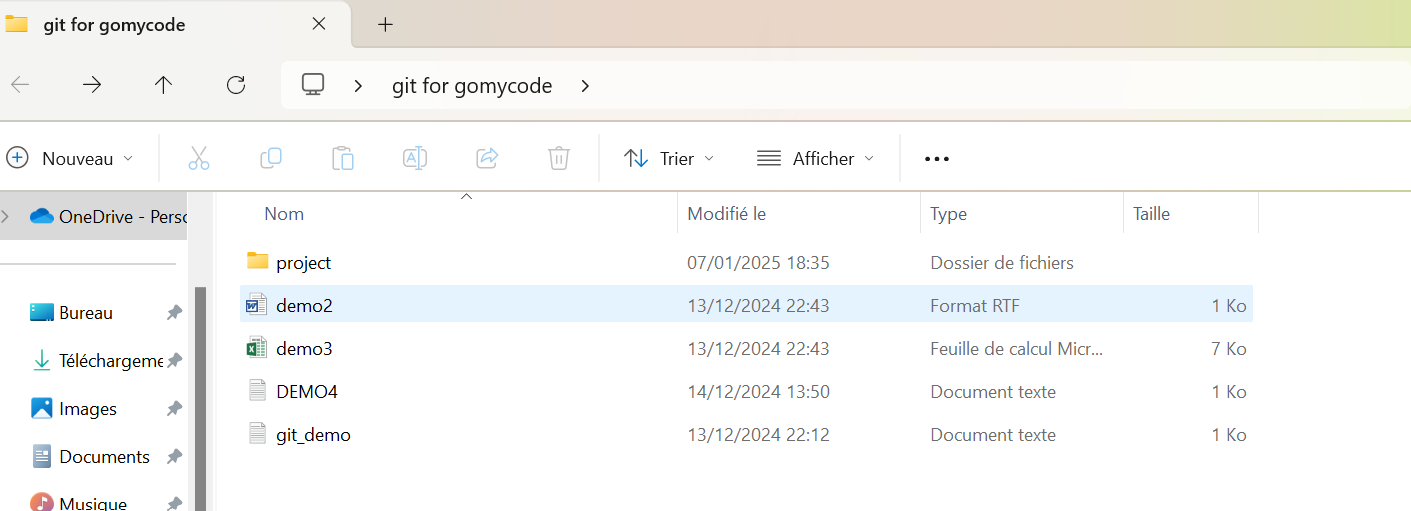
**Git Ajouter de nouveaux fichiers**

Vous venez de créer votre premier repo Git local.

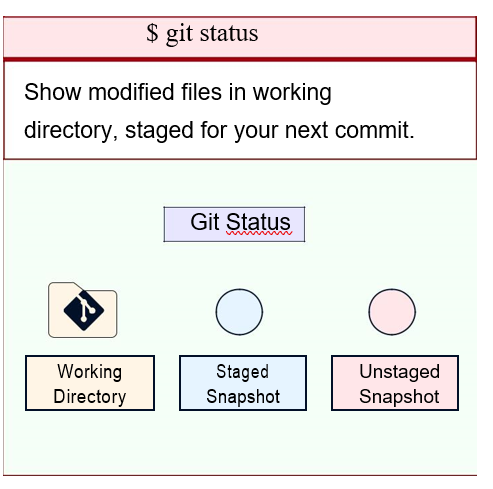
Ajoutons donc quelques fichiers, ou créons un nouveau fichier.



**Git Status**

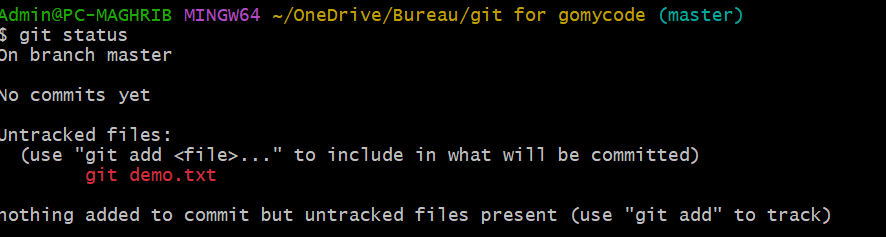
La commande git status affiche l'état du répertoire de travail et de la zone de transit.

Après avoir créé des fichiers sur votre repo local, nous vérifions maintenant l'état de Git et voyons s'il fait partie de notre repo:



Les fichiers dans votre dossier de dépôt Git peuvent être dans l'un des 2 états suivants:

* **Tracked**- les fichiers que Git connaît et qui sont ajoutés au dépôt
* **Untracked**- les fichiers qui sont dans votre répertoire de travail, mais qui n'ont pas été ajoutés au dépôt



Lorsque vous ajoutez pour la première fois des fichiers à un dépôt vide, ils ne sont pas suivis. Pour que Git les suive, vous devez les mettre en scène, ou les ajouter à l'environnement de mise en scène.

Nous aborderons l'environnement de mise en scène dans les prochaines étapes.

**Environnement de transit Git : Ajouter, commettre, pousser**

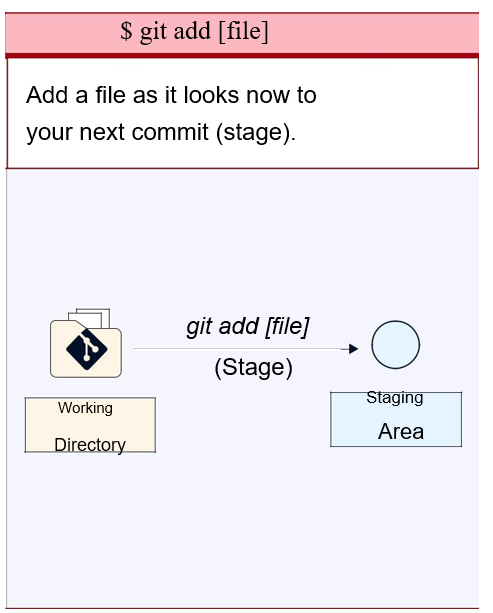
L'une des fonctions principales de Git est le concept d'environnement de mise à disposition et de commettre.

**Les fichiers mis à disposition** sont des fichiers qui sont prêts à être **commis** dans le dépôt sur lequel vous travaillez.

Les commandes git add, git commit et git push sont fondamentales pour gérer les modifications dans Git, vous permettant de mettre en place des modifications locales et de pousser des mises à jour de dépôts distants.

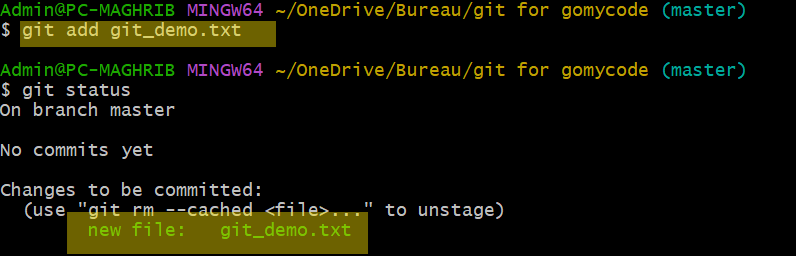
**git add**

git add est la commande utilisée pour commencer à suivre les modifications effectuées dans votre dépôt. Elle met en scène les changements pour le prochain commit, ce qui signifie qu'elle marque les modifications dans votre répertoire de travail pour les stocker dans un instantané avec le prochain commit.



Utilisez cette commande:

Git add filename.extention



Maintenant le fichier est ajouté à l'environnement de mise en scène, et nous sommes prêts à faire notre premier commit.

**Focus !**L'utilisation de --all au lieu de noms de fichiers individuels mettra en scène tous les changements (nouveaux, modifiés et supprimés) des fichiers.

Note : La commande abrégée pour git add --all est git add -A

**Git Commit**

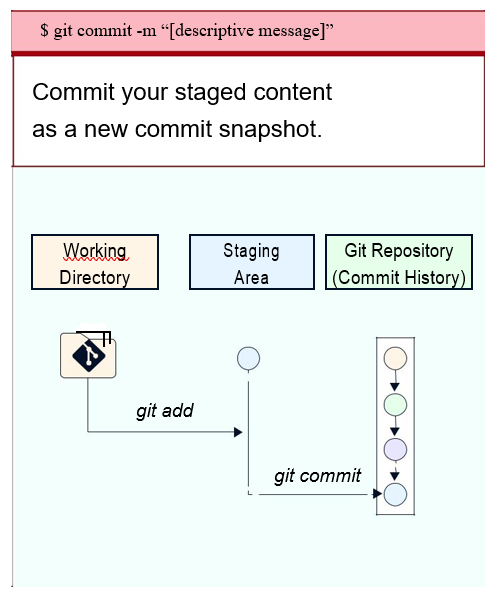
Puisque nous avons terminé notre travail, nous sommes prêts à passer de l'étape au commit pour notre repo.

L'ajout de commits permet de suivre nos progrès et nos changements au fur et à mesure que nous travaillons. Git considère chaque livraison comme un point de changement ou "point de sauvegarde". C'est un point du projet auquel vous pouvez revenir si vous trouvez un bogue ou si vous voulez faire une modification.

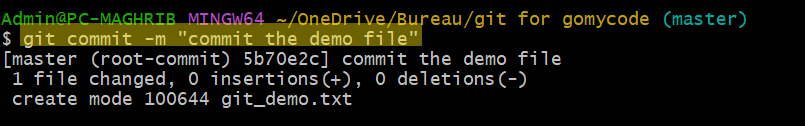
Lorsque nous effectuons un commit, nous devrions **toujours** inclure un **message**.

En ajoutant des messages clairs à chaque livraison, il est facile pour vous (et les autres) de voir ce qui a changé et quand.

Une fois que les changements sont mis en page avec git add, git commit est utilisé pour enregistrer des instantanés de ces changements dans l'historique du projet. Cet instantané vous permet d'enregistrer des versions de la base de code à des moments incrémentiels.

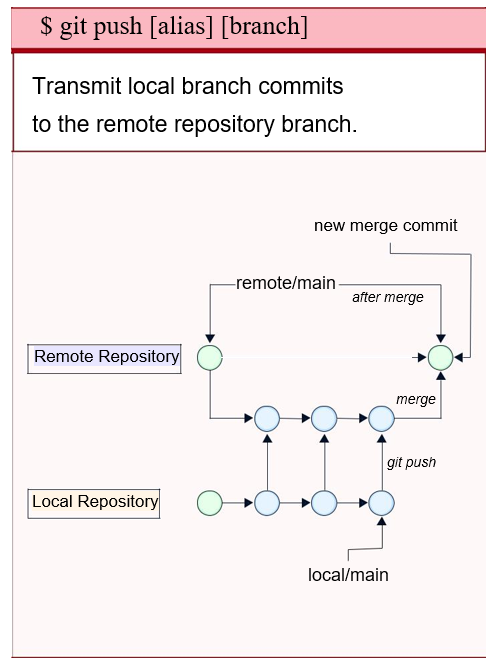


Utilisez cette commande : Git commit -m "votre message"



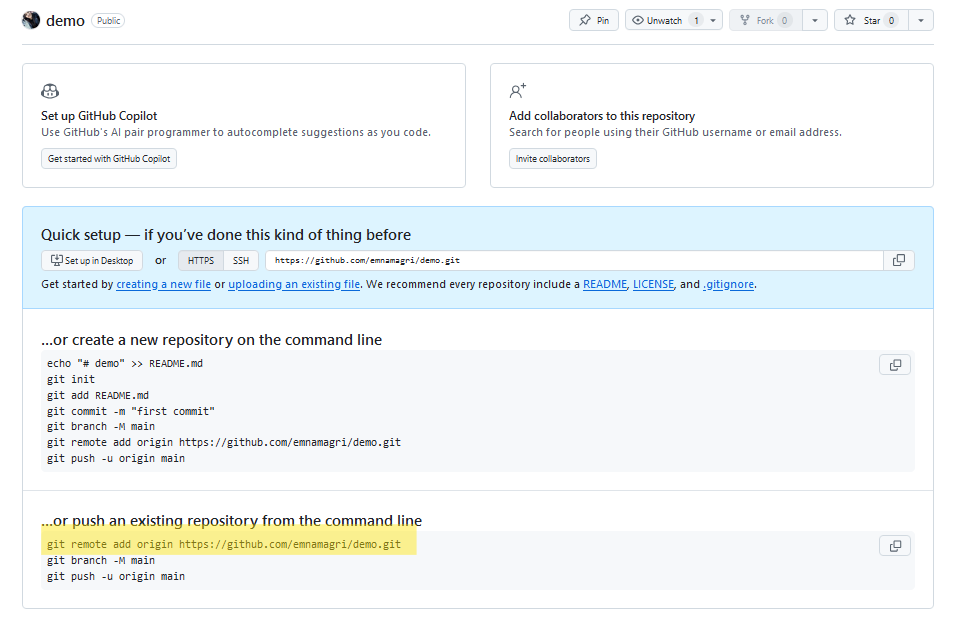
**Git push**

Après avoir effectué des modifications localement, git push télécharge vos modifications vers un dépôt distant, en les stockant sur un serveur en ligne, en assurant la redondance et en les mettant à la disposition d'autres développeurs.



**Pousser le dépôt local vers GitHub**

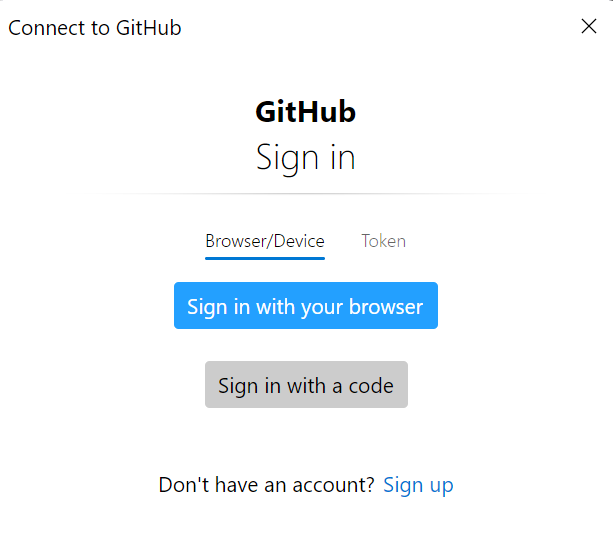
Puisque nous avons déjà mis en place un dépôt Git local, nous allons le pousser vers GitHub:



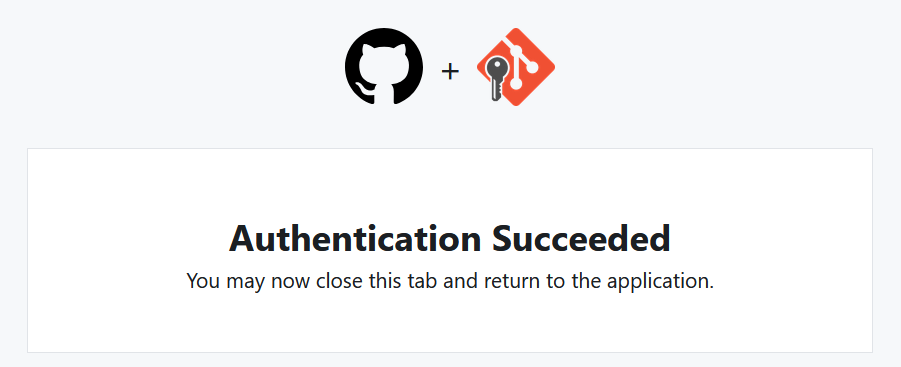
Copiez l'URL, ou cliquez sur le presse-papiers marqué dans l'image ci-dessus.

Collez maintenant la commande suivante dans git Bash.

La première fois, il vous sera demandé de vous connecter à votre compte GitHub.



Connectez-vous pour réussir votre authentification



Maintenant, retournez sur GitHub et voyez que le dépôt a été mis à jour:

