**Un peuple-Un but-Une foi**



**MINISTERE DE L’ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE**

**DIRECTION GENERALE DE L’ENSEIGNEMENT SUPERIEUR**



**COURS DE DEVOPS**

# Présentation de Git & GitHub

**Sommaire:**

1. Qu'est-ce que Git, GitHub ?

1. Son rôle dans le développement de logiciels
   1. Outils complémentaires
   2. Avantage d’utilisation de Git

1. Installation de git sur windows

1. Mise en place de l’environnement de travail et la découverte des commandes.

**Conclusion**

**I. Qu’est-ce que Git ? GuitHub ?**

**Git** est un système de contrôle de version ou VCS (Version Control System) qui nous permet de gérer nos projets, il est libre et open source.

Un VCS est un logiciel dont le rôle est d’enregistrer l’évolution d’un fichier ou d’un ensemble de fichier au cours du temps de manière à ce qu’on puisse rappeler les versions antérieures de ce fichier à tout moment.

Les projets de développement de logiciel sont en générale dans des dossiers ou répertoires dans lesquels se trouve un ensemble de fichiers contenant des instructions transcrites selon la syntaxe de langage utilisée. Git est donc un outil qui nous permettra d’avoir une trace sur les différents changements ou mise à jour apportés au cours d’un projet spécifique et nous aide à les envoyer sur le web ou l’héberger sur GitHub.

GitHub est service d’hébergement open source, permettant aux programmeurs et développeurs d’héberger et de partager leur code informatique afin de travailler de façon collaborative.

# Le rôle de git et github dans le développement de logiciel

Git gère les ajouts et changements apportés au code source de manière tracée.

Ainsi, si une erreur est commise, les développeurs peuvent revenir en arrière et comparer les versions antérieures du code, ce qui leur permet de corriger l'erreur tout en minimisant les perturbations pour tous les membres de l'équipe.

Git facilite donc le travail collaboratif en minimisant les risques de perte de travail et en permettant également aux développeurs de travailler chacun sur leurs branches et donc en autonomie sans empiéter sur le travail des autres.

En termes de sécurité, l’intégrité du code source géré par Git en fait un outil primordial. En effet, l'intégrité a été la priorité absolue lors de la conception de Git. Ainsi la traçabilité des changements ne peut jamais être remise en cause.

Les équipes de développement qui n'utilisent aucune forme de contrôle de version se heurtent souvent à des problèmes tels que le fait de ne pas savoir quels changements ont été mis à la disposition des utilisateurs ou l'apport de changements incompatibles entre deux éléments indépendants, qui doivent alors être minutieusement démêlés et retravaillés.

# a. Outils complémentaires

On ne peut parler de Git sans parler également des systèmes d’hébergement et de gestion de code.

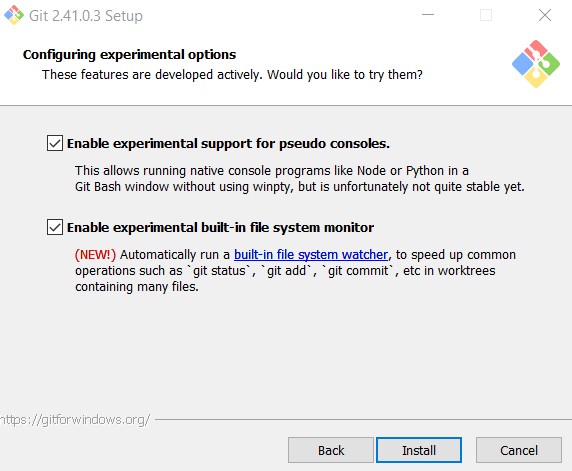
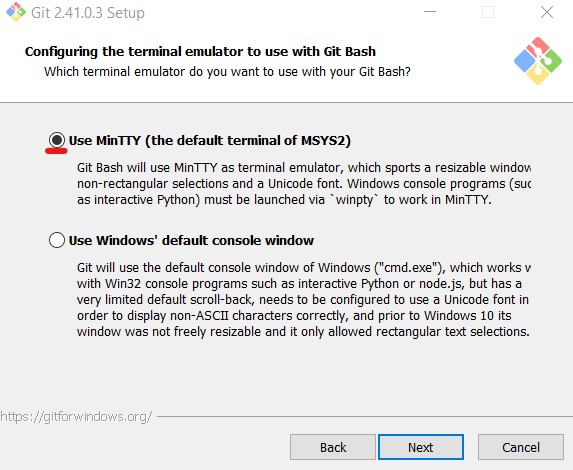
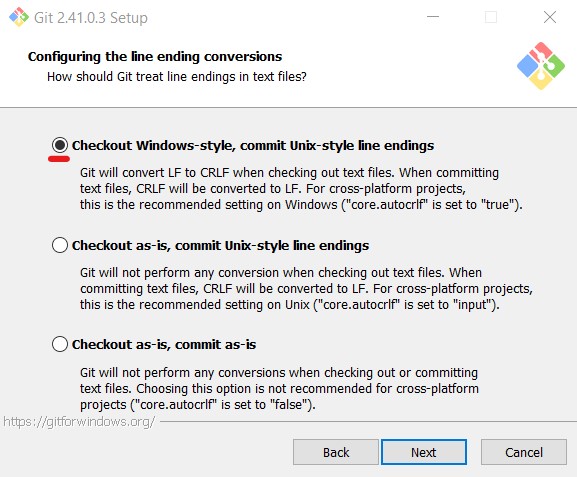
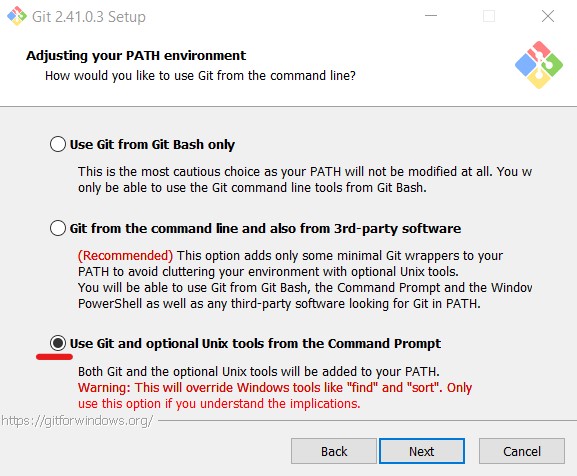
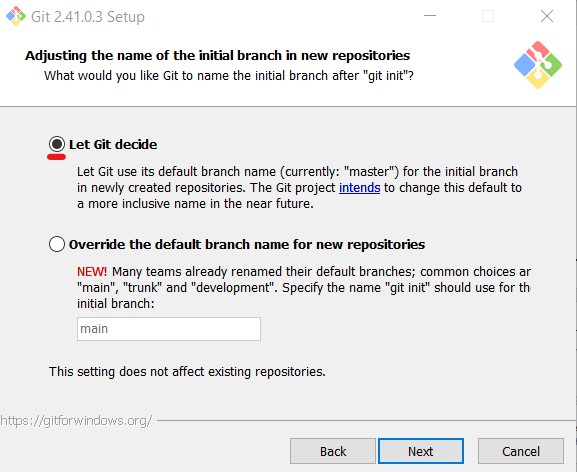
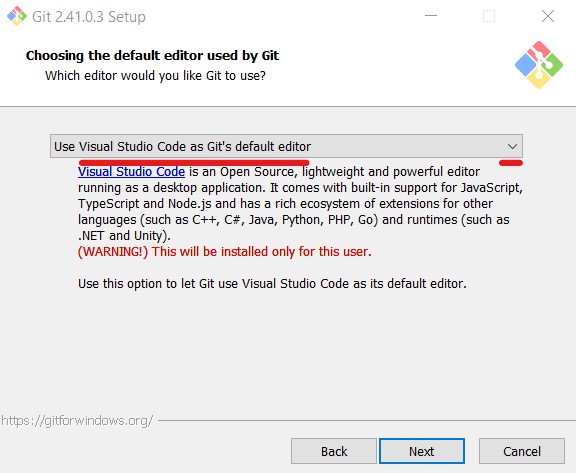
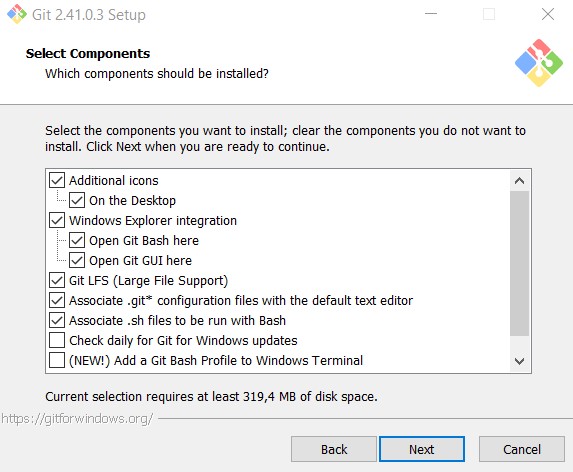
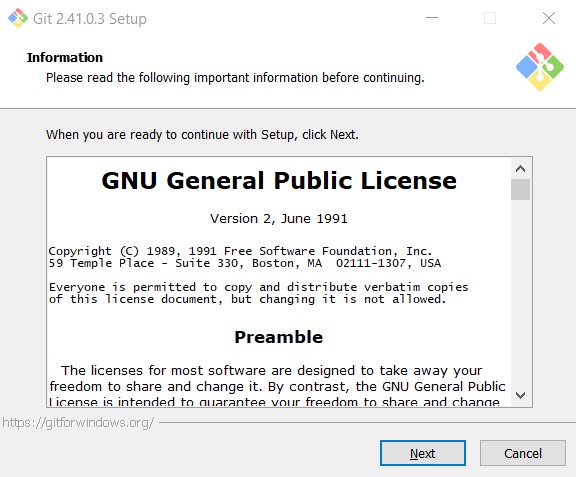
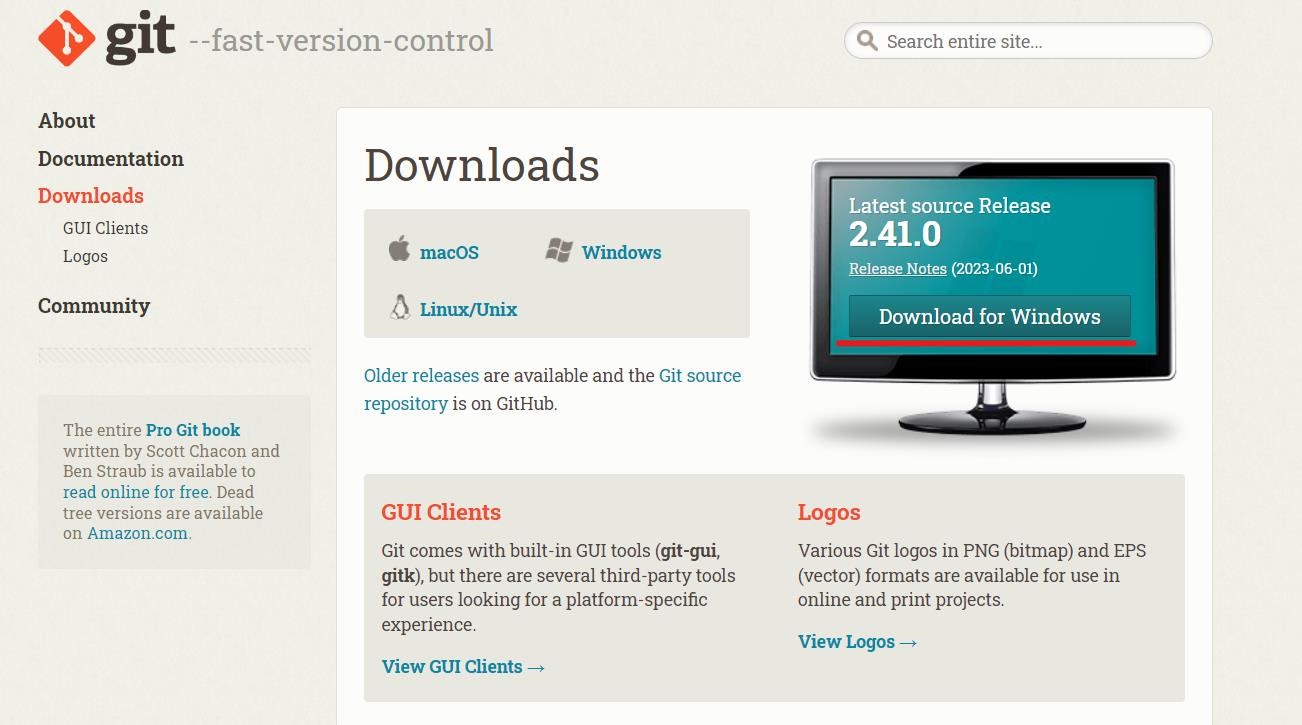
Les principaux sont **Github, Gitlab** ou bien **BitBucket**. Ces hébergeurs sont complémentaires à Git et viennent en extension de celui-ci.

Ils permettent notamment d'héberger le code sur internet et donc de le rendre accessible facilement aux équipes, tout en offrant une interface graphique simple et intuitive aux fonctionnalités prévues par Git.

# b. Quelques avantages d’utilisation Git

* La collaboration : si plus d’une personne travaille sur le même projet ou sur un même code, git nous permet de fusionner proprement les différents changements de manière logique.
* Historique des différentes versions
* Développement en parallèle avec le système de branche.

# Installation de Git



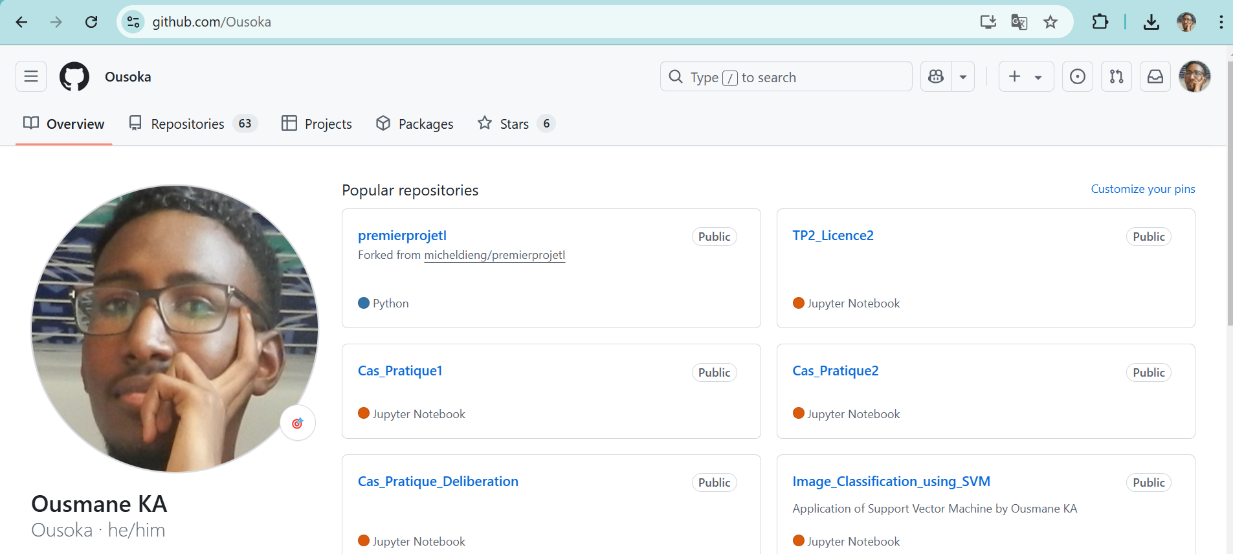
**IV. Mise en place de l’environnement de travail et la découverte des commandes.**

*Scenario :* Nous allons créer un dossier où sera contenu notre projet, y ajouter du code, faire des modifications avec différentes versions du projet et les envoyer en ligne sur GitHub.

Nous apprendrons les commandes de bases de git qui nous permettront de mener à bien notre projet.

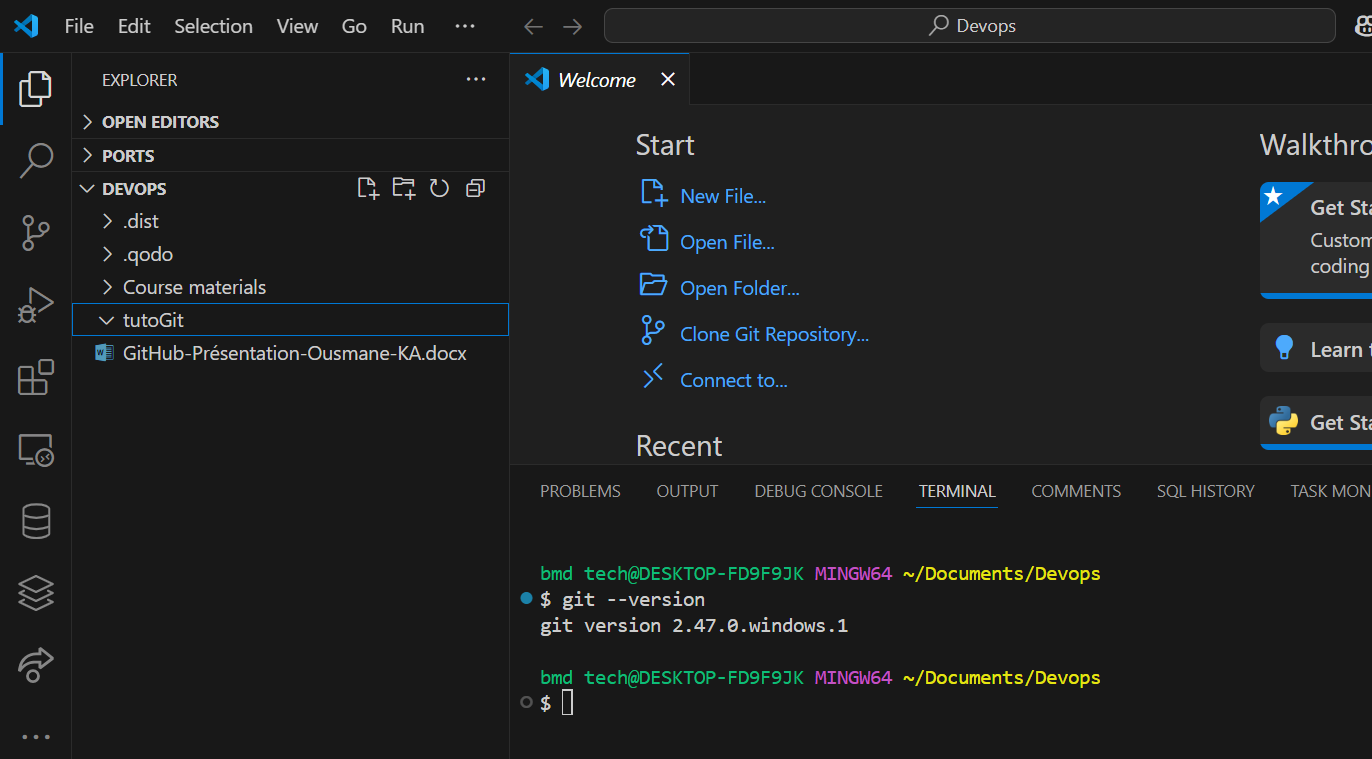
Si vous n’avez pas de compte GitHub vous pouvez en créer un rapidement sur la plateforme.

A notre niveau nous avons déjà un compte. Voir ci-dessous:

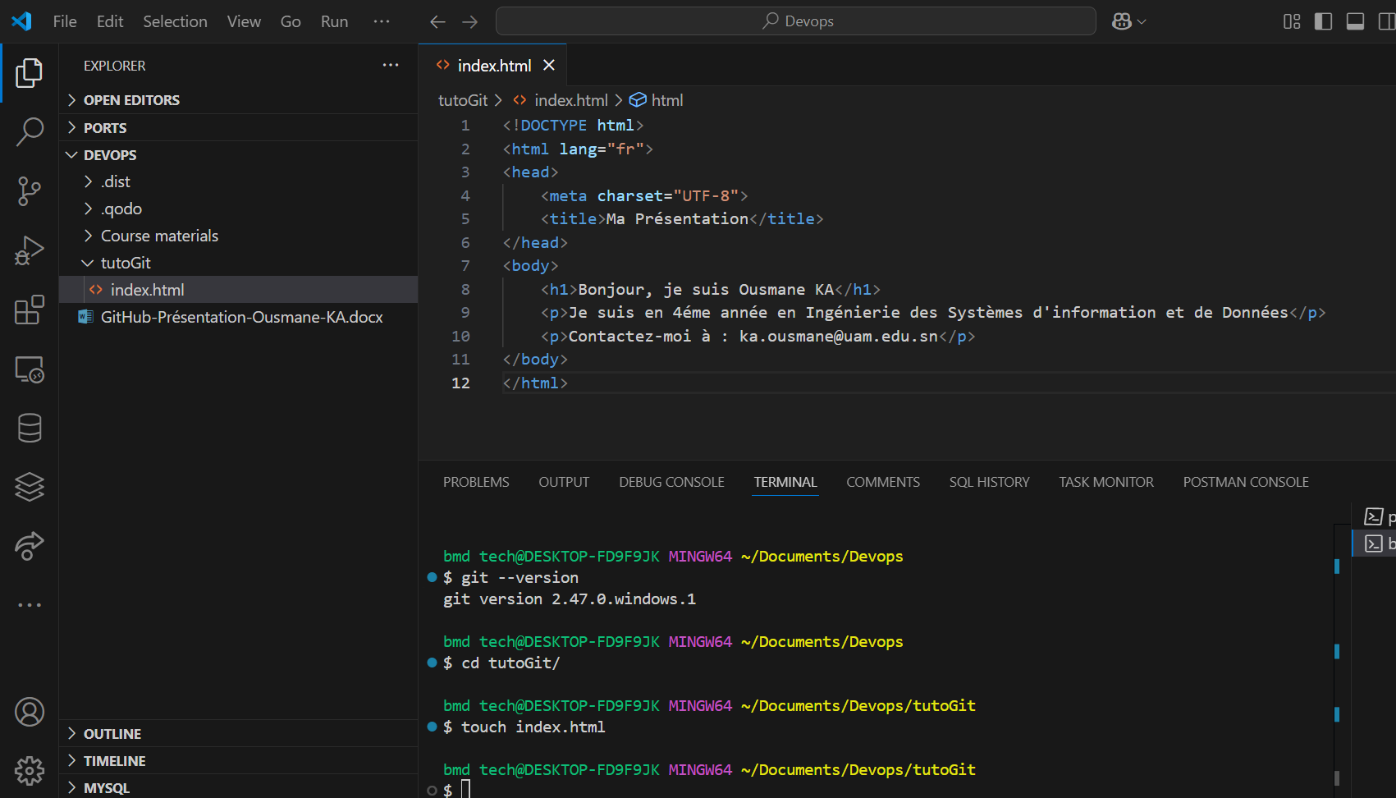


Nous allons créer notre répertoire de projet sur le bureau avec comme nom ***tutoGit*** puis nous allons l’ouvrir sur notre éditeur de texte **VSCode**.

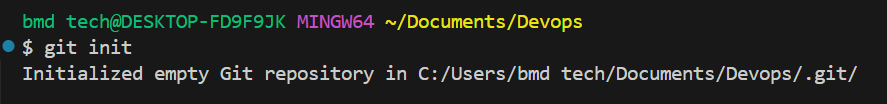
Ouvrons maintenant le dossier dans ***Git bash***, là où se fera les commandes.



Nous allons créer un fichier index.html où on mettra un bout de code en html. Le fichier est créé sur git bash avec la commande ***touch***et on peut remarquer que ça se crée automatiquement dans le dossier tutoGit.

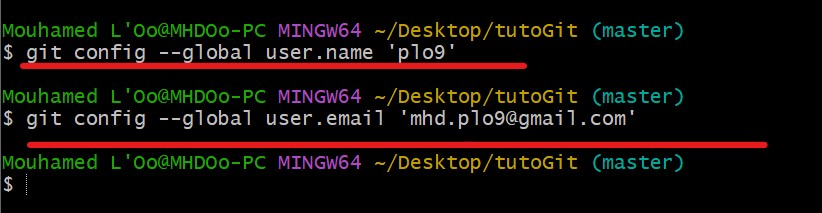


Nous allons initialiser notre code sur git avec la commande git init



Nous allons à présent nous identifier sur git bash avec la commande ***git config***. Ici j’ai utilisé les mêmes identifiants que sur GitHub.

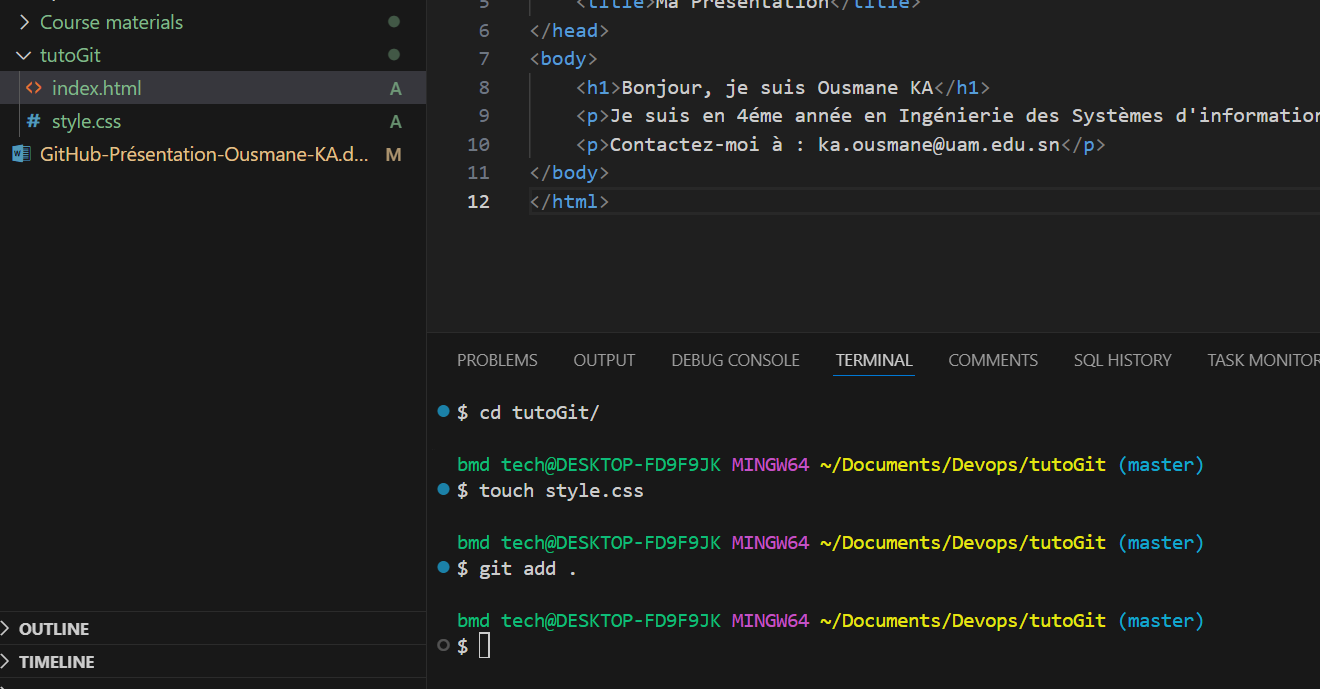
Cette étape a été déjà faite à notre niveau et voici une illustration de comment y parvenir.



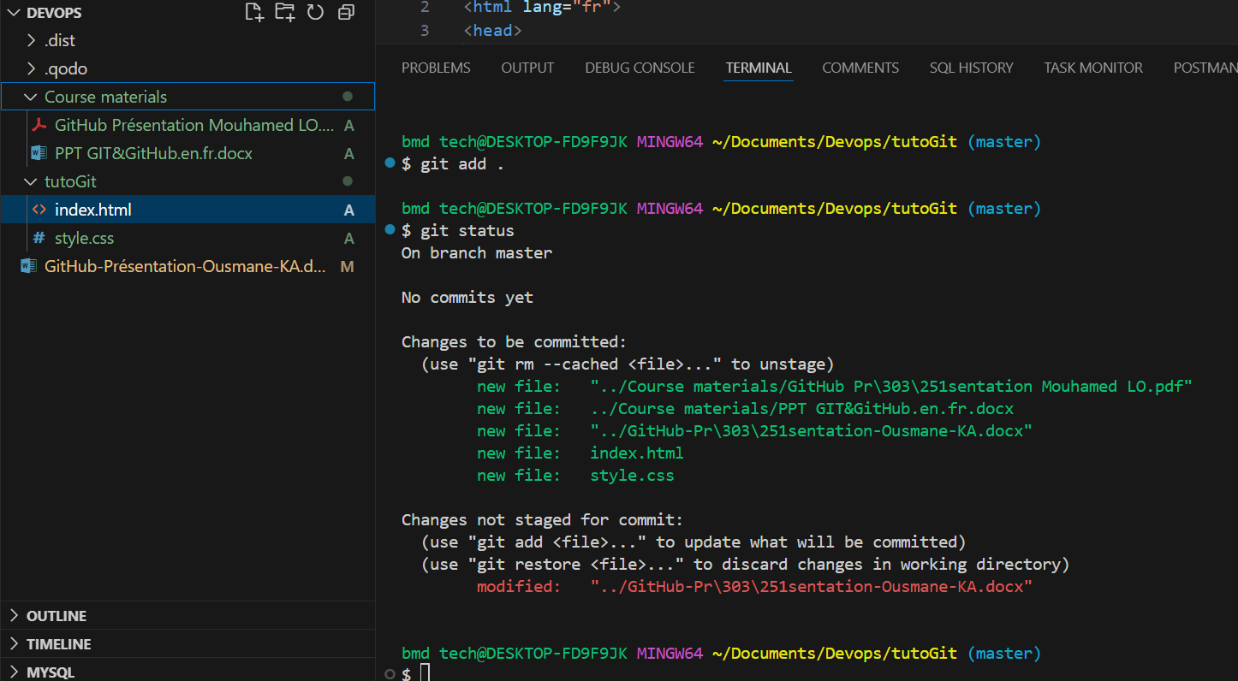
Maintenant que git bash nous a identifié nous pouvons utiliser toutes les commandes. On ajoute notre fichier *index.html* à la liste des fichiers à sauvegarder (ou au pré sauvegarde) avec la commande ***git add***.

Nous allons créer un nouveau fichier style.css

La commande ***git add .*** permet d’ajouter tous les fichiers présents dans un dossier



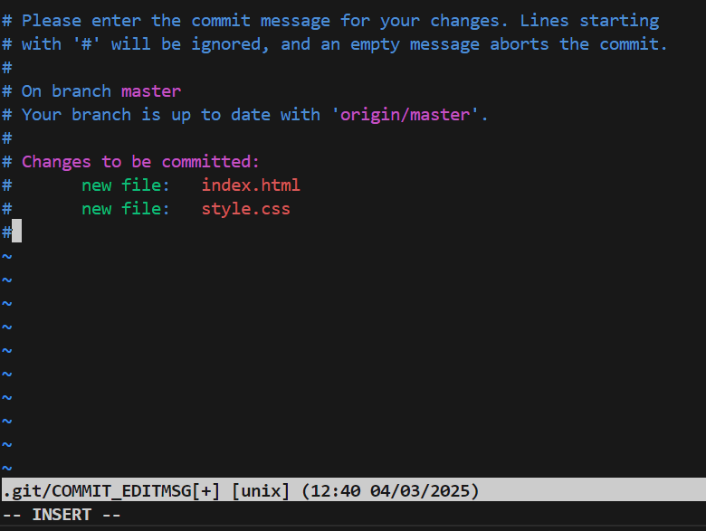
On peut voir à l’aide de la commande git status la liste des fichiers disponible au pré sauvegarde. Nous allons maintenant les sauvegarder avec la commande ***git commit***.



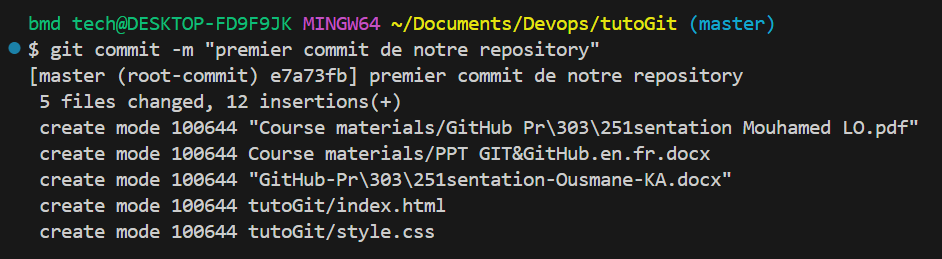
Quand la commande ***git commit*** est exécutée, un éditeur de texte s’ouvre et nous demande de commenter la sauvegarde de façon à avoir une traçabilité. Il peut s’agir de l’éditeur vim ou n’importe quel autre éditeur défini lors de l’installation de git bash. Dans notre cas c’est *VS code*.



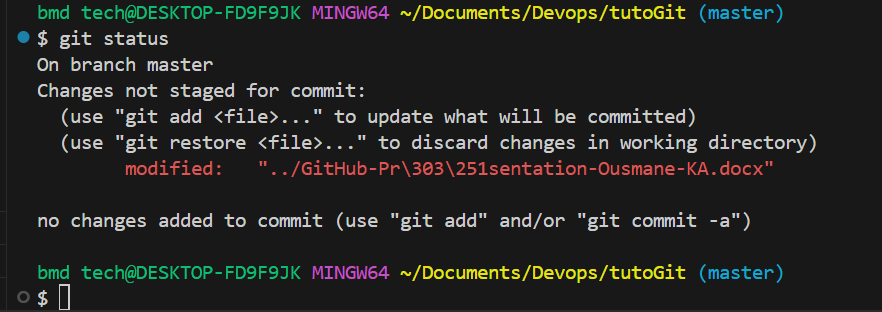
Ajouter le message et faire Ctrl+C :q



Ou directement utiliser la commande ***git commit -m ‘message du commit’***



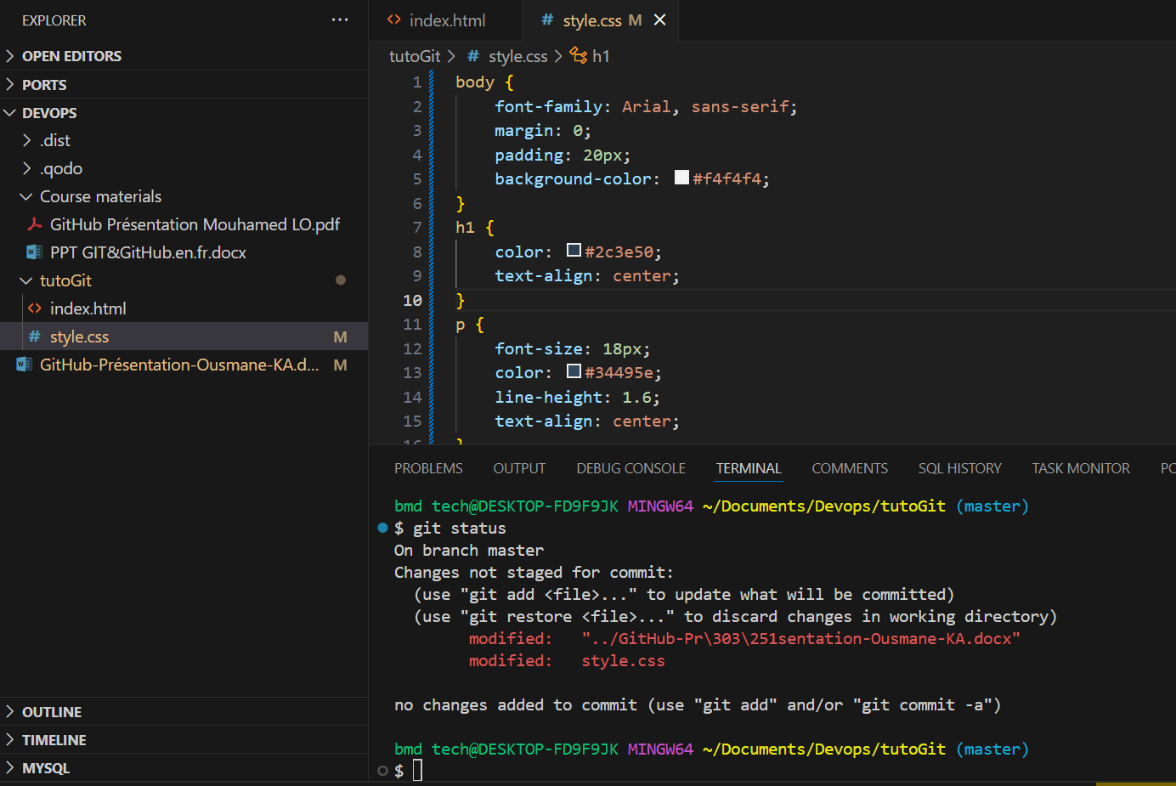
Et si on refait un ***git status*** on verra que les fichiers ne sont plus à l’étape de pré sauvegarde mais plutôt sauvegardés.



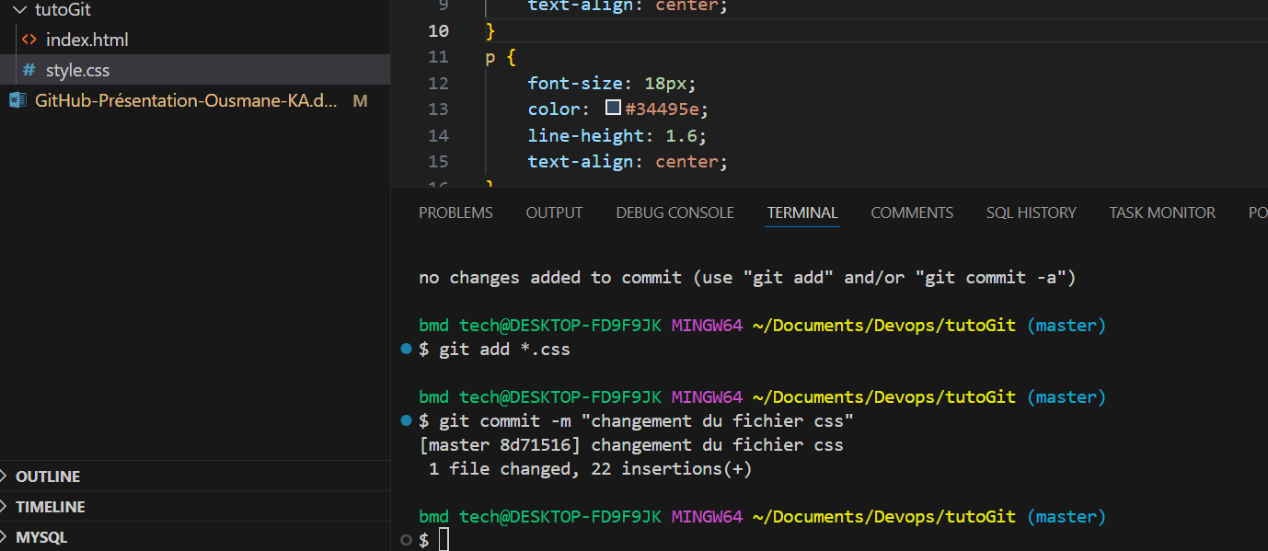
Là, nous allons modifier notre fichier css en y ajoutant une ligne de code et voir ce qui se passe.

On effectue un ***git status*** et on nous signale que le fichier style.css a été modifié et qu’il faut l’ajouter à nouveau.

On effectue à nouveau un ***git add*** et cette fois ci avec comme commande ***git add \*.css*** pour n’ajouter que le fichier css.



Et maintenant nous devons faire un deuxième commit mais cette fois ci nous voulons sauter l’étape où nous allons écrire notre commentaire sur l’éditeur de texte. On mettra donc le commentaire au niveau de la commande : voir capture ci-dessous

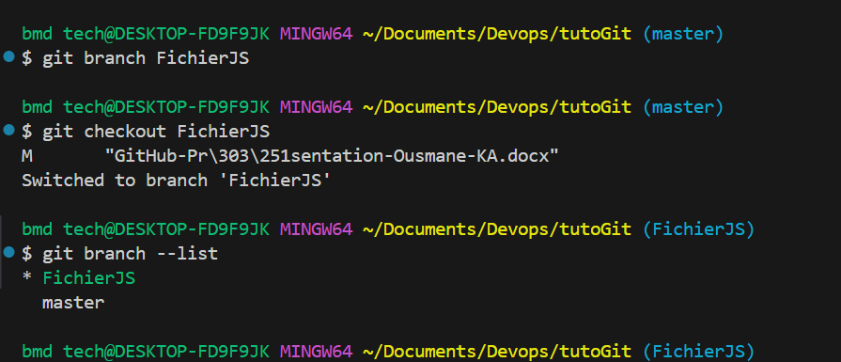


Comme expliquer plus haut il est possible de faire du développement en parallèle. Cela se fait avec les branches.

Git status nous montre que nous sommes sur la branche **master** et nous voulons créer un fichier JavaScript sur une branche à part puis lier le fichier html et le fichier JavaScript

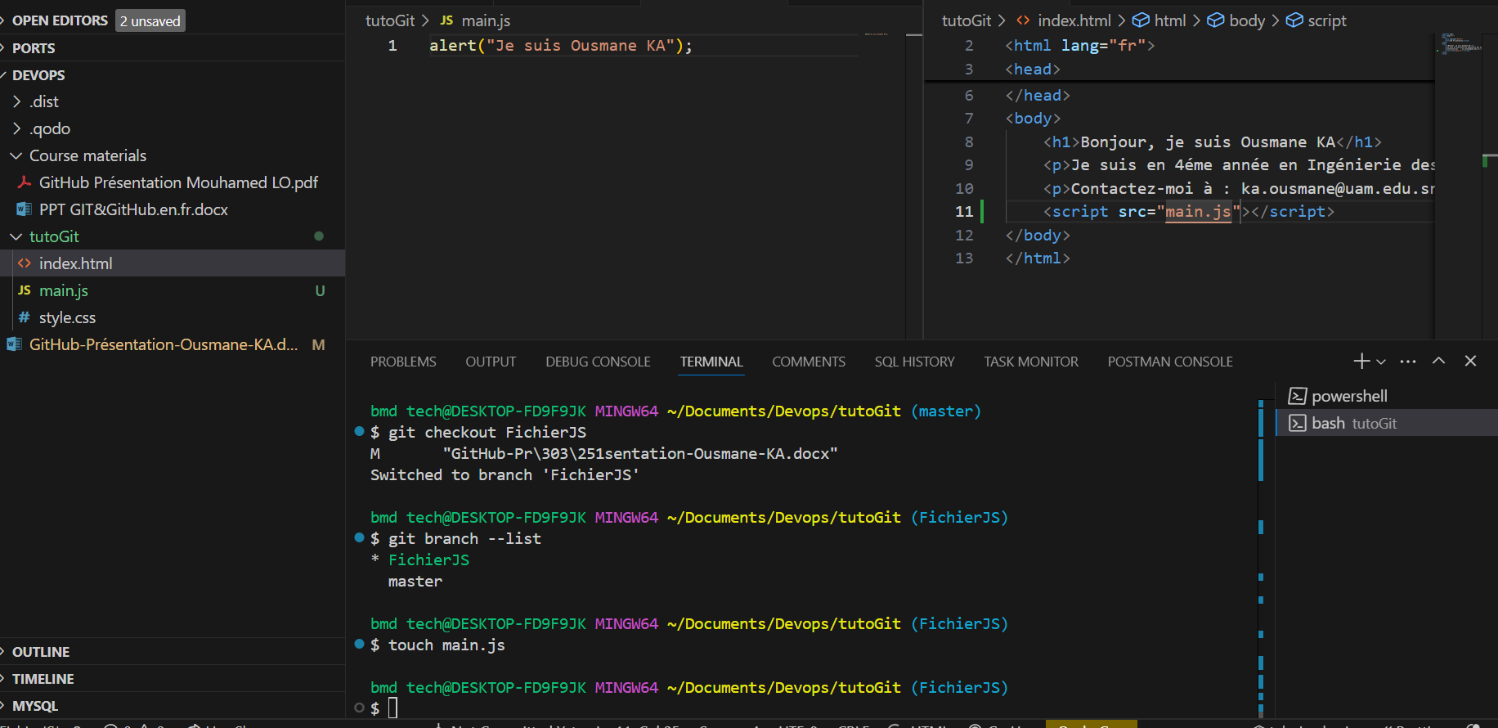
Et pour changer de branche on utilise la commande ***git checkout***

La commande ***git branch --list*** nous permet de voir la liste des branches existantes.

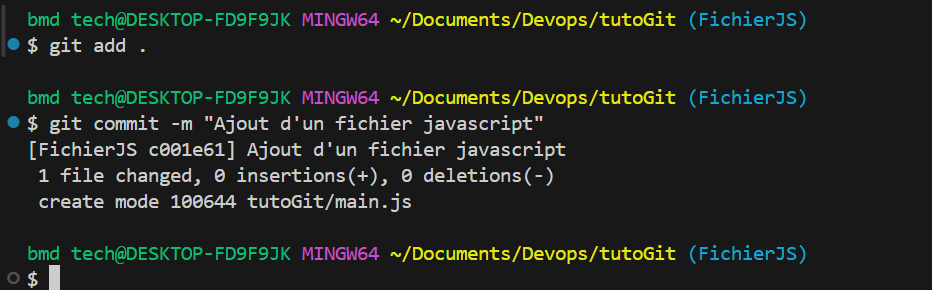


On crée ensuite un fichier *main.js* et on y insère une ligne de code en JavaScript.

Nous allons ensuite le référencer dans notre fichier html à l’aide des balises scripts.

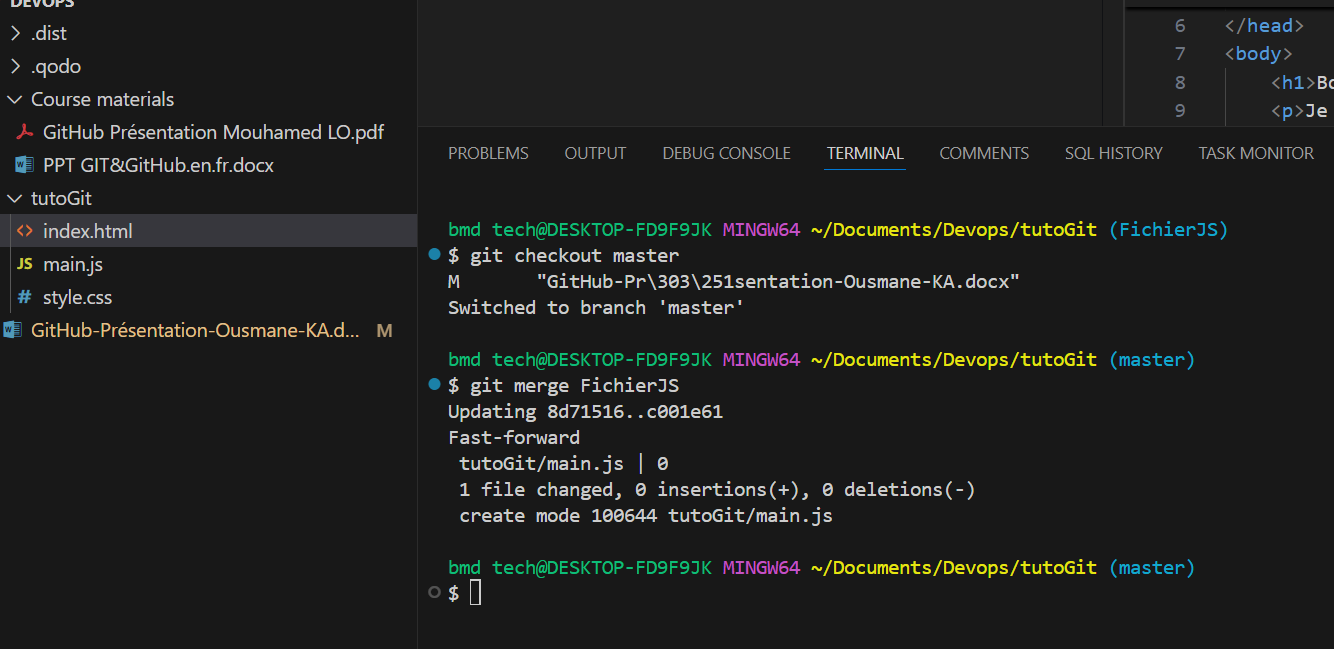


On ajoute à nouveau tout notre dossier avec **git add .** puis on fait un commit avec comme commentaire : Ajout du fichier javascript

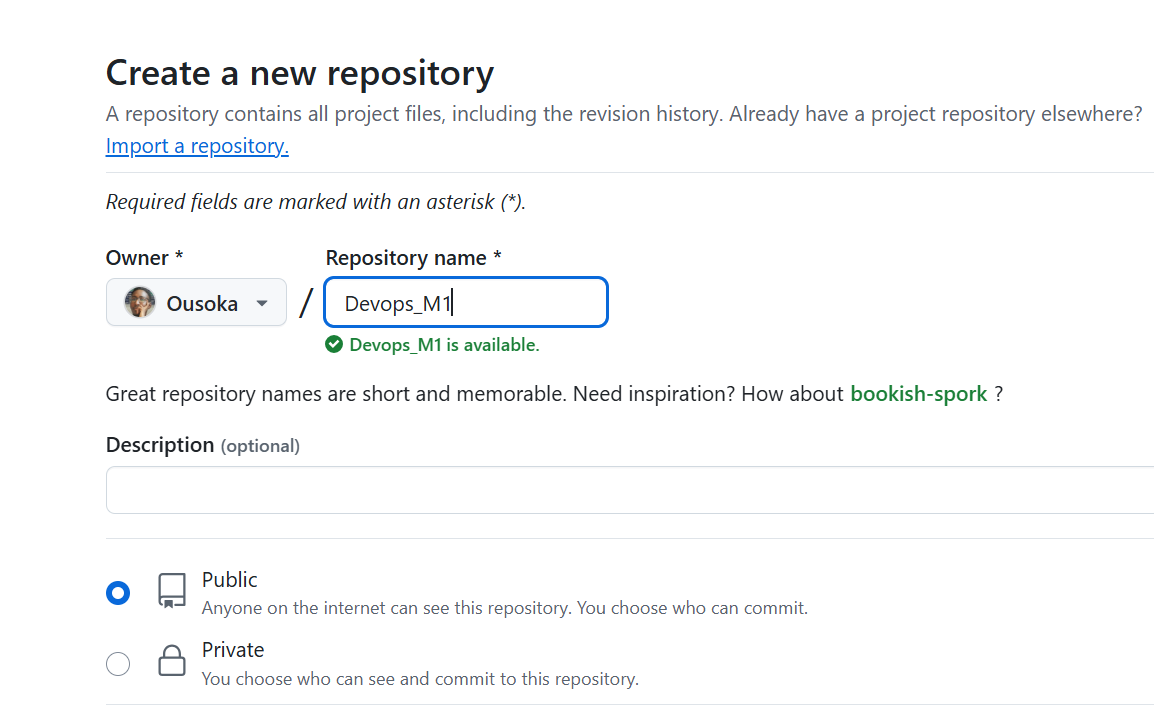


Nous allons maintenant revenir sur la branche master et on remarquera que les modifications fait au niveau de la branche FichierJS sont disparus, on parle alors de développement en parallèle.

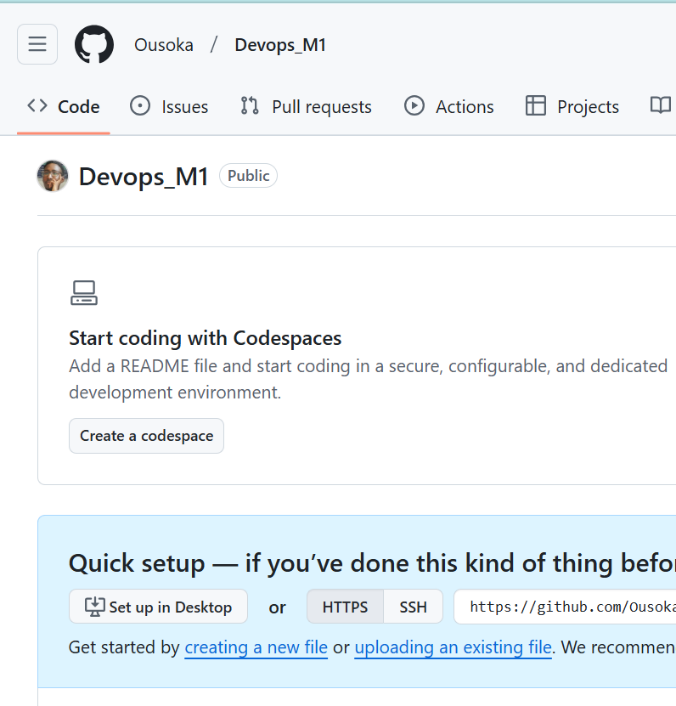
Maintenant que nous avons fini de faire nos changements on fusionne nos 2 branches avec la commande ***git merge***



Nous allons maintenant sur GitHub et créer un repository ou répertoire qui contiendra notre projet.

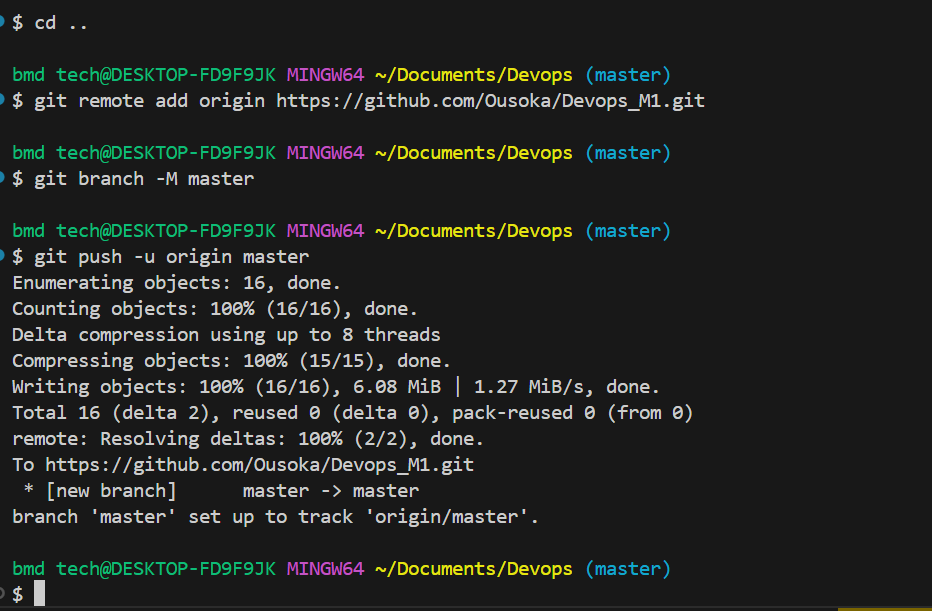


On donne un nom au répertoire et une description

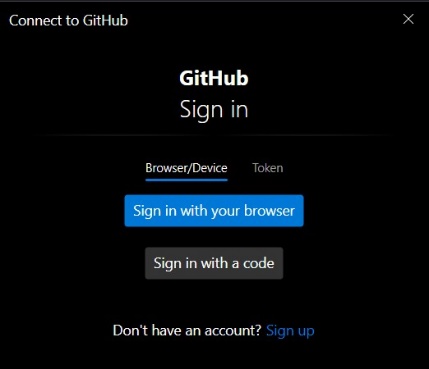
 

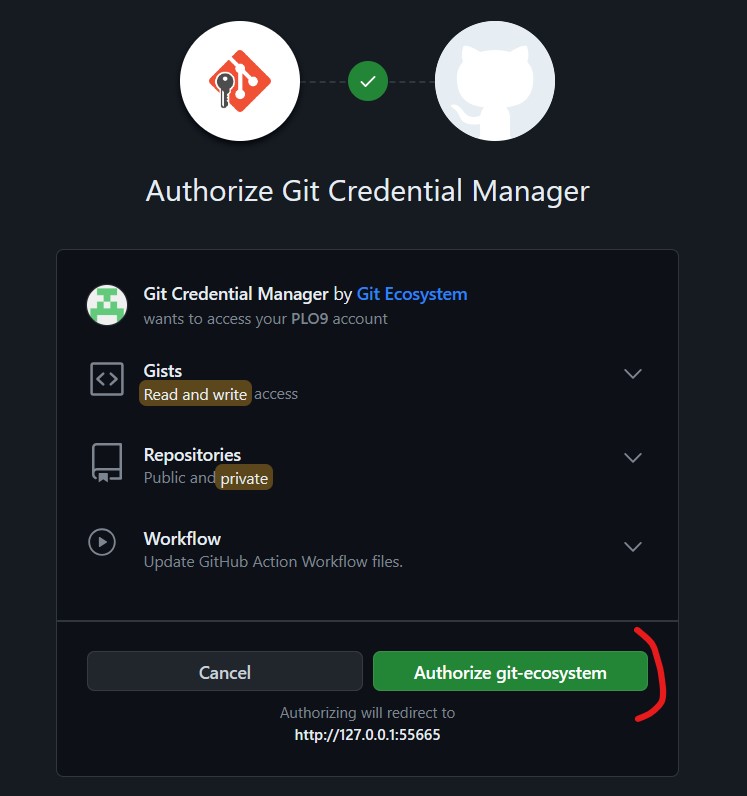
A ce niveau git nous explique comment ajouter notre premier fichier à notre repo ce qui n’est pas notre cas car on a déjà un projet existant donc on choisit la deuxième option qui nous explique comment push notre dossier existant

On copie les trois lignes et on colle sur notre git bash.

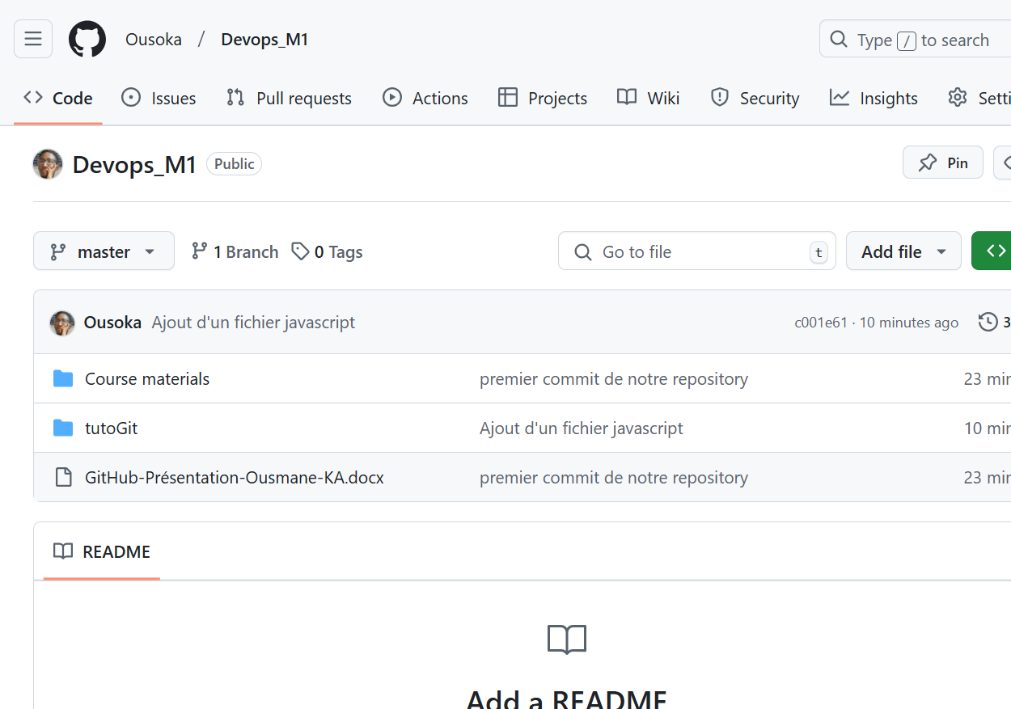


Si c’est notre première connexion git nous demandera de mettre nos identifiants.

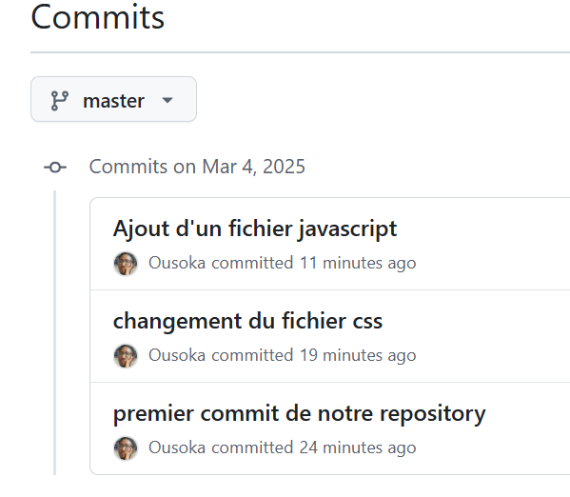




On revient sur GitHub, rafraichissons la page et nous verrons que notre projet a été bien importé.



Sur commit on peut récupérer le code à n’importe quelle version.

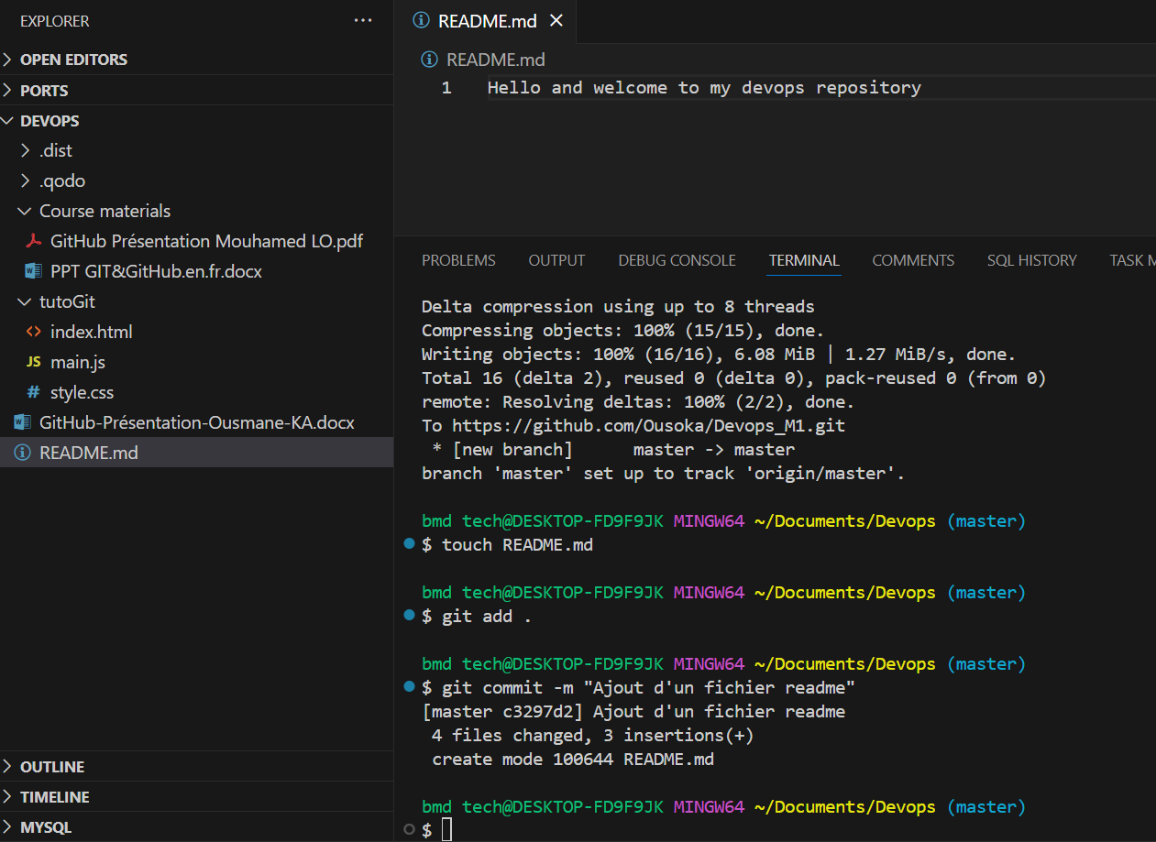


Sur github on voit souvent que certains repository ont une description en dessous. Cela est possible avec le fichier README.

On créé le fichier README avec la commande *touch* et avec l’extension *.md* sinon cela s’affichera pas correctement sur github.

Ajoutons ensuite la phrase de description avec notre éditeur de texte.

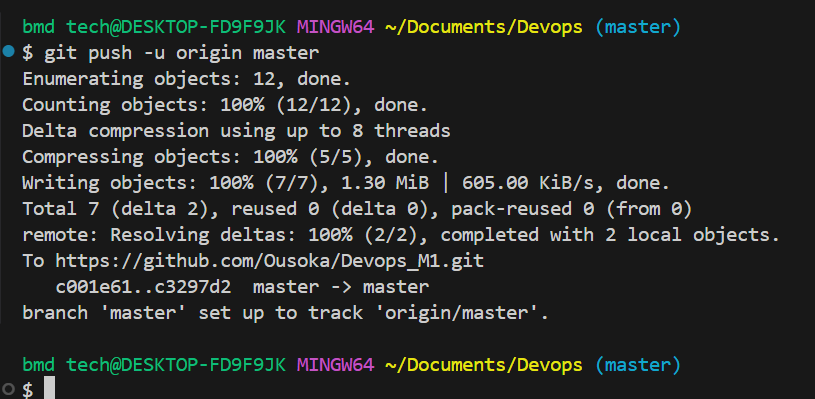
On ajoute ensuite notre README avec un git add puis git commit



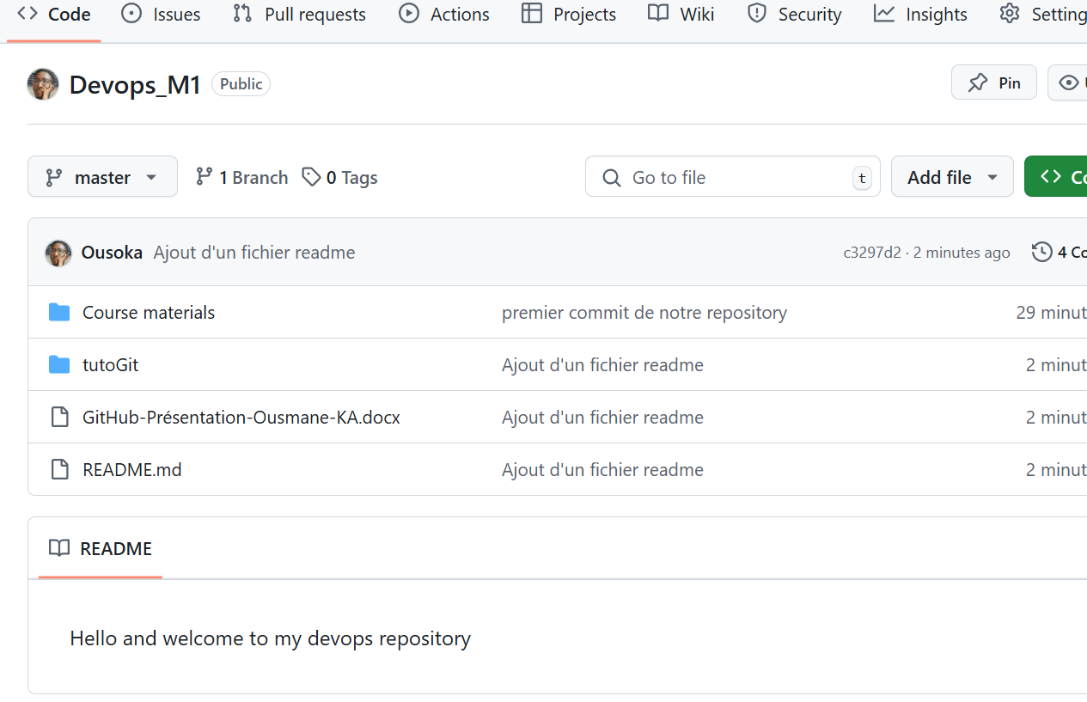
En rafraichissant la page github on ne voit rien, ce qui est normal.

Il faut le pousser avec git push

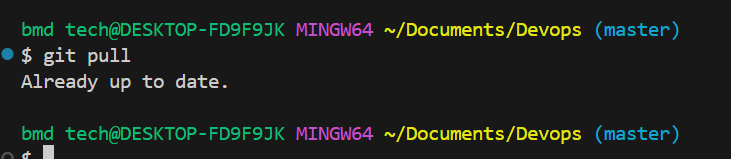
Cela nous permet de mettre tous les changements sur notre repository de github.



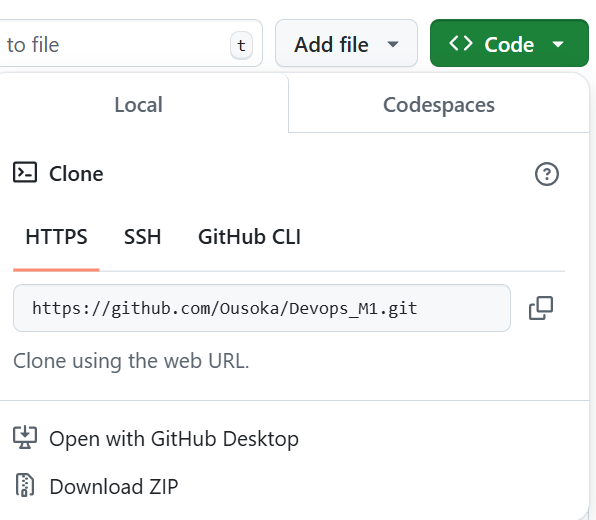
Après rafraichissement on peut maintenant voir

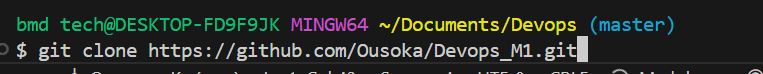


En cas de changement par l’un des membres de l’équipe.



Un dernier cas de figure c’est qu’en cas de perte ou de suppression du dossier du projet on pourra le récupérer à travers GitHub sur un fichier zippé





# Conclusion

GitHub se révèle être bien plus qu'une simple plateforme de gestion de versions. C'est un écosystème dynamique qui favorise la collaboration, l'innovation et la création collective. Grâce à ses fonctionnalités de suivi des modifications, de gestion des problèmes et de partage de code, GitHub facilite le travail d'équipes dispersées géographiquement et permet aux développeurs du monde entier de collaborer efficacement sur des projets communs. Que vous soyez un débutant désireux d'apprendre ou un professionnel chevronné cherchant à accélérer le développement, GitHub offre les outils nécessaires pour transformer des lignes de code en réalisations exceptionnelles.