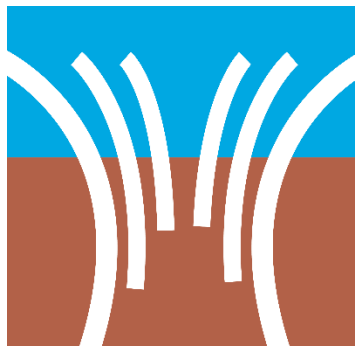


Un peuple-Un but-Une foi



MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE

DIRECTION GENERALE DE L'ENSEIGNEMENT SUPERIEUR



**UNIVERSITE
AMADOU MAHTAR MBOW
DE DAKAR**

Plus qu'une formation, un avenir à construire

COURS DE DEVOPS

Présentation de Git & GitHub



Sommaire:

- I. Qu'est-ce que Git, GitHub ?
- II. Son rôle dans le développement de logiciels
 - a. Outils complémentaires
 - b. Avantage d'utilisation de Git
- III. Installation de git sur windows
- IV. 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.

II. 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.

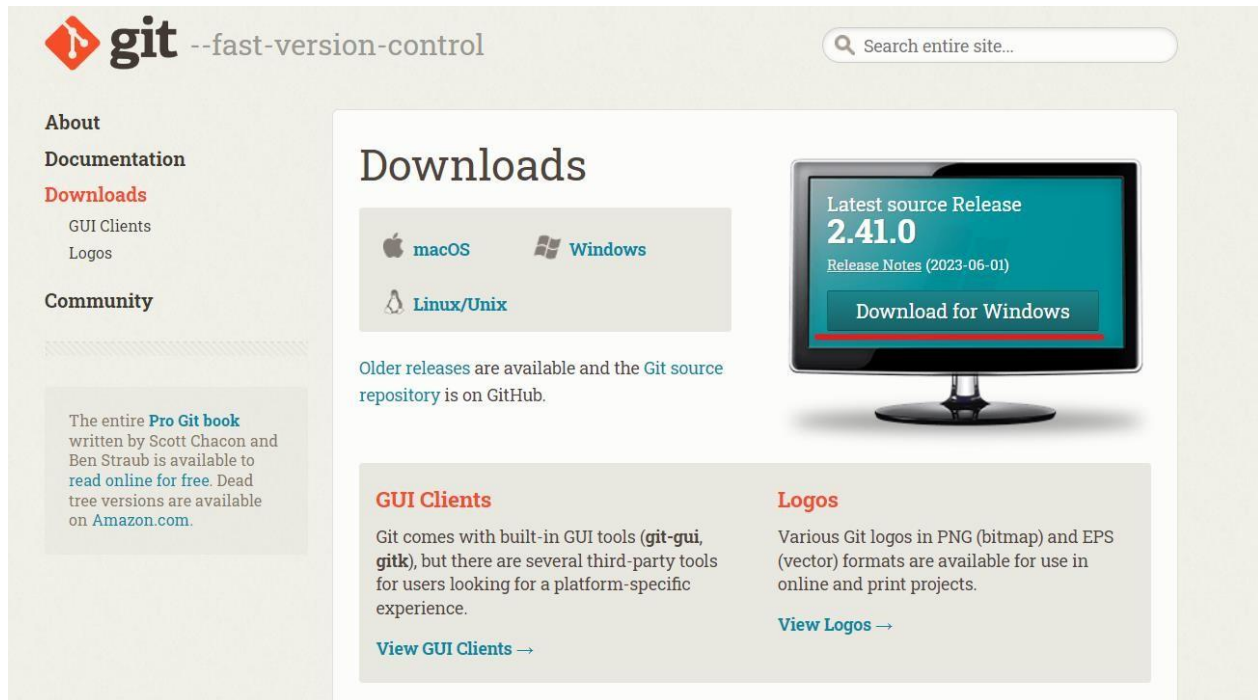
Les principaux sont **Github**, **Gitlab** ou bien **BitBucket**. Ces hébergeurs sont complémentaires à Git et viennent en extension de celui-ci.

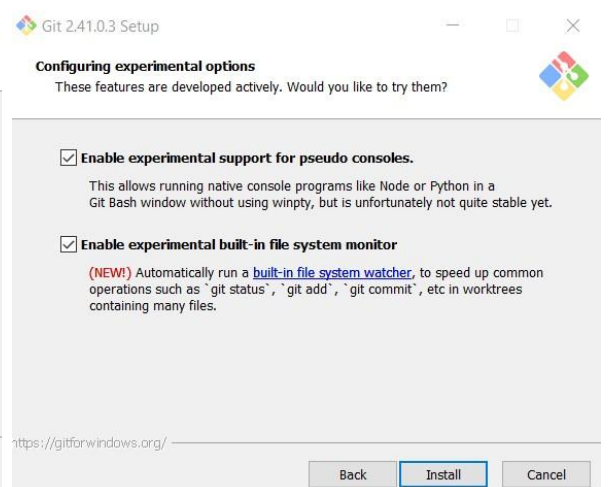
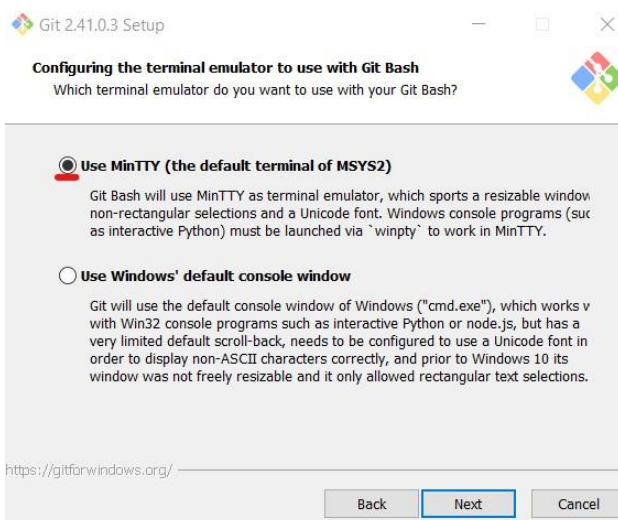
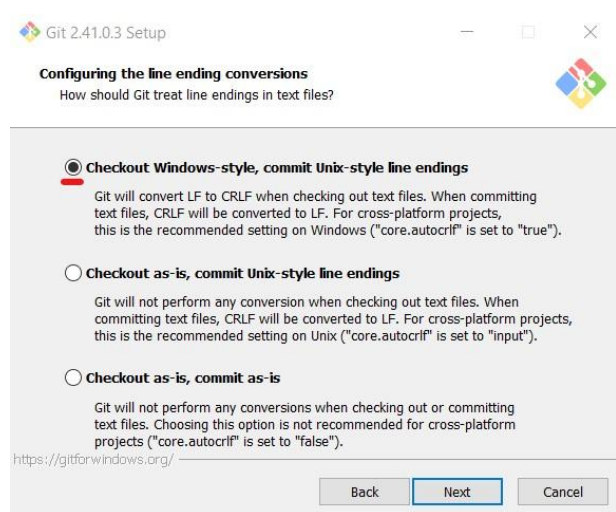
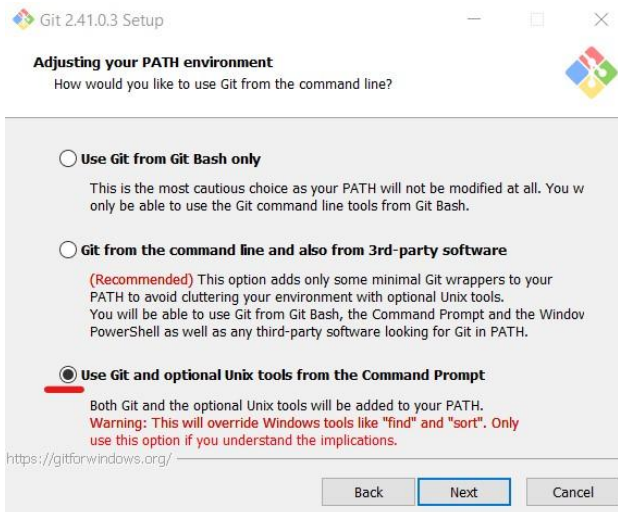
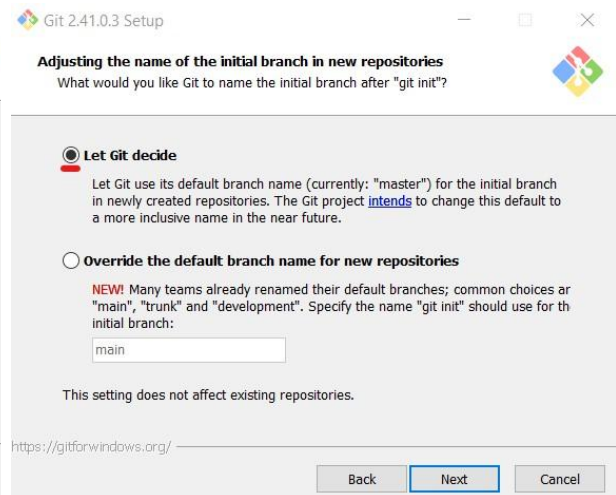
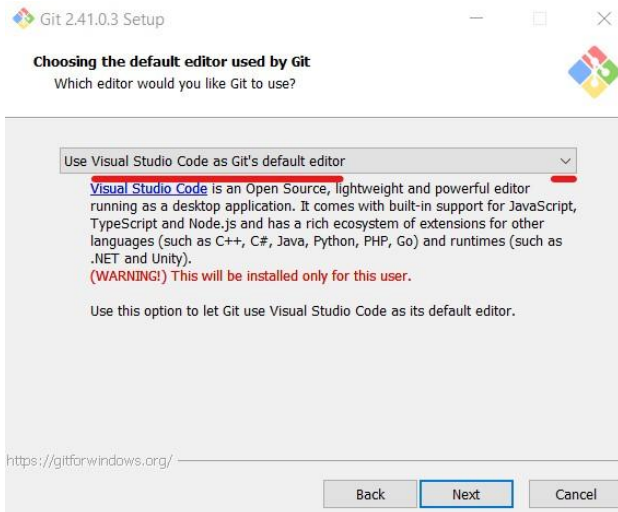
Ils permettent notamment d'héberger le code sur internet et donc de le rendre accessible facilement aux équipes, tout en offrant une interface graphique simple et intuitive aux fonctionnalités prévues par Git.

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.

III. Installation de Git





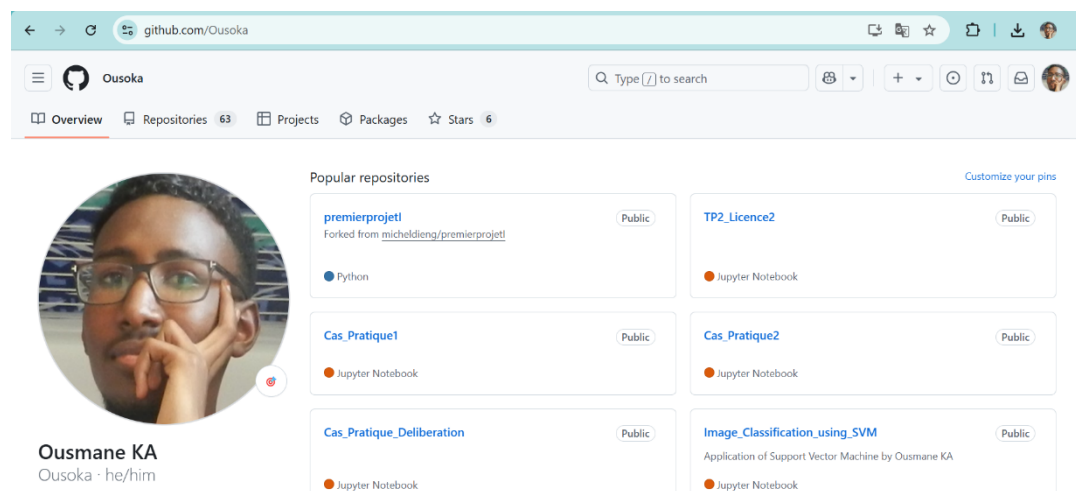
IV. Mise en place de l'environnement de travail et la découverte des commandes.

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.

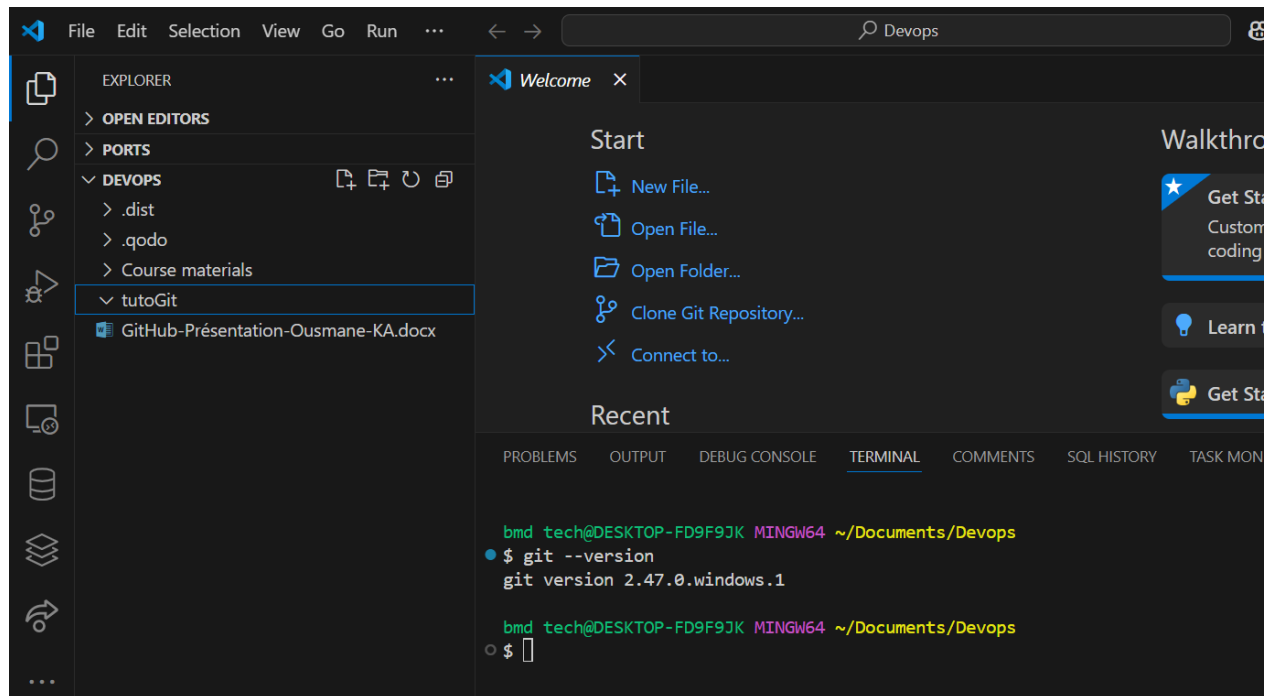
Si vous n'avez pas de compte GitHub vous pouvez en créer un rapidement sur la plateforme.

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

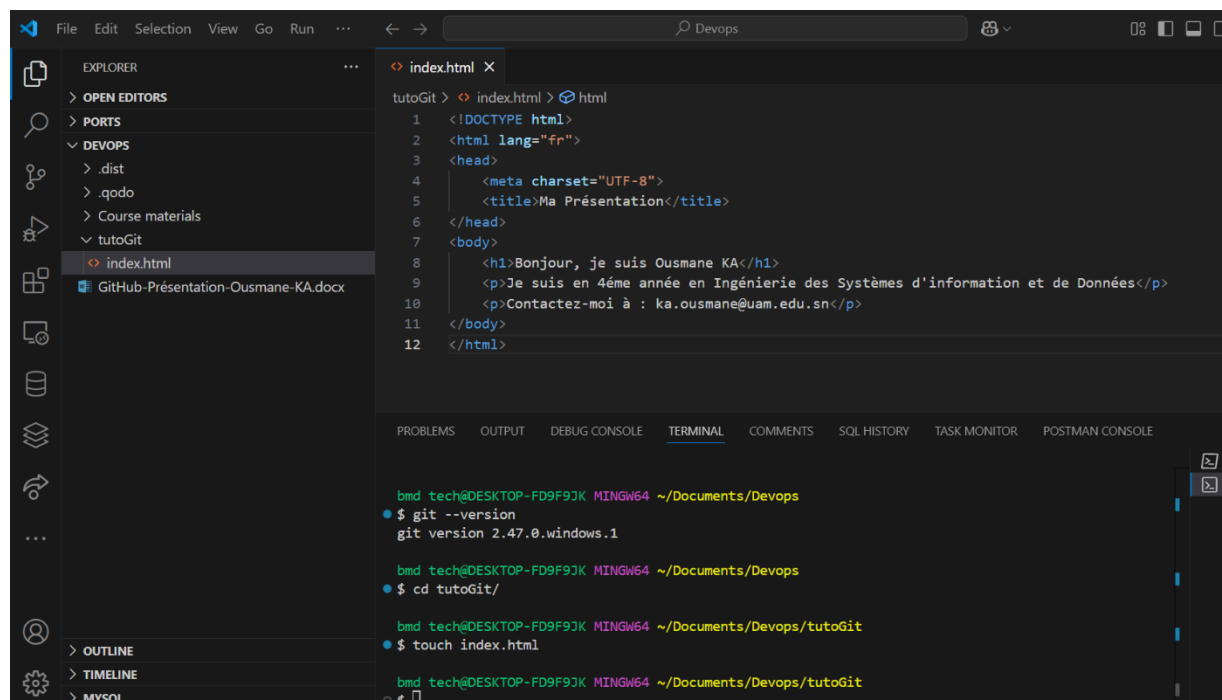


Nous allons créer notre répertoire de projet sur le bureau avec comme nom **tutoGit** puis nous allons l'ouvrir sur notre éditeur de texte **VSCode**.

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`

```
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops
• $ git init
Initialized empty Git repository in C:/Users/bmd tech/Documents/Devops/.git/
```

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.

```
Mouhamed L'Oo@MHD0o-PC MINGW64 ~/Desktop/tutoGit (master)
$ git config --global user.name 'plo9'

Mouhamed L'Oo@MHD0o-PC MINGW64 ~/Desktop/tutoGit (master)
$ git config --global user.email 'mhd.plo9@gmail.com'

Mouhamed L'Oo@MHD0o-PC MINGW64 ~/Desktop/tutoGit (master)
$ |
```

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

```

5 <title>Ma Présentation</title>
6 </head>
7 <body>
8 <h1>Bonjour, je suis Ousmane KA</h1>
9 <p>Je suis en 4ème année en Ingénierie des Systèmes d'information
10 <p>Contactez-moi à : ka.ousmane@uam.edu.sn</p>
11 </body>
12 </html>

```

```

• $ cd tutoGit/

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
• $ touch style.css

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
• $ git add .

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
○ $ 

```

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***.

```

2 <html lang="fr">
3 <head>

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
• $ git add .

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
• $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   ../Course materials/GitHub Pr\303\251sentation Mouhamed LO.pdf"
    new file:   ../Course materials/PPT GIT&GitHub.en.fr.docx
    new file:   ../GitHub-Pr\303\251sentation-Ousmane-KA.docx"
    new file:   index.html
    new file:   style.css

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:   ../GitHub-Pr\303\251sentation-Ousmane-KA.docx"

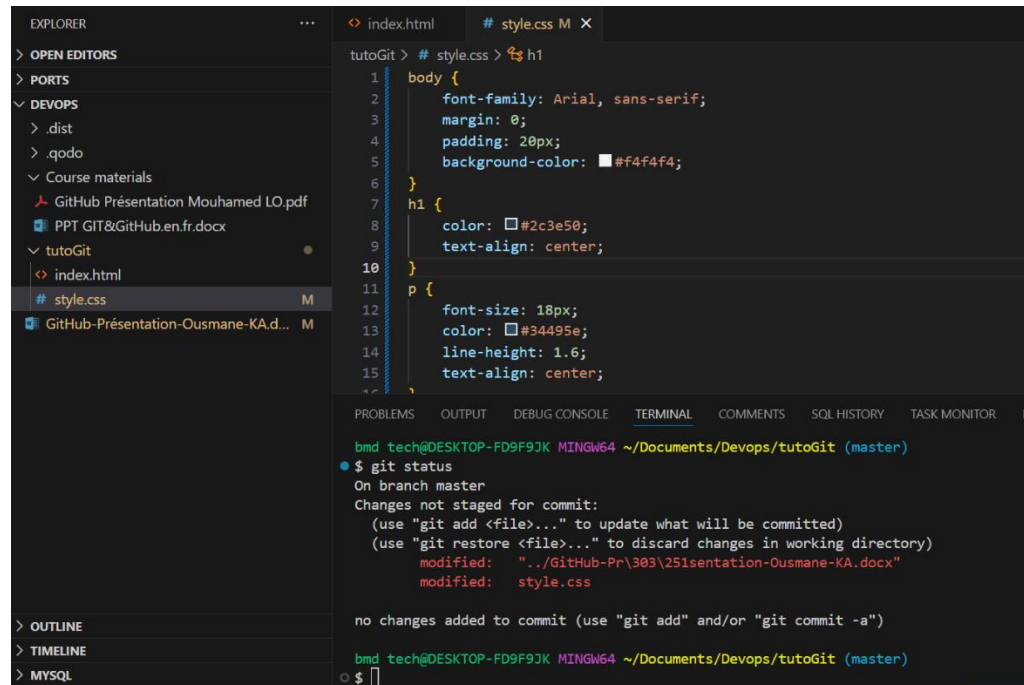
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
○ $ 

```

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*.

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.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'tutoGit' with files 'index.html' and 'style.css'. The main editor displays the content of 'style.css', which includes styles for a body, h1, and p. Below the editor, the TERMINAL panel is open, showing the output of a 'git status' command. The output indicates that 'style.css' has been modified but is not staged for commit. The prompt '\$' is visible at the bottom of the terminal.

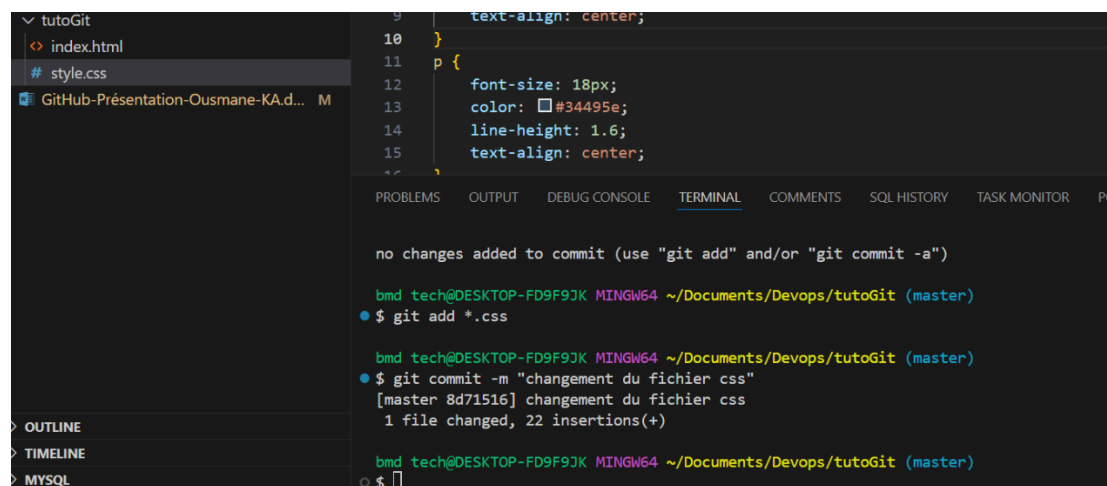
```
tutoGit > # style.css > h1
1  body {
2      font-family: Arial, sans-serif;
3      margin: 0;
4      padding: 20px;
5      background-color: #f4f4f4;
6  }
7  h1 {
8      color: #2c3e50;
9      text-align: center;
10 }
11 p {
12     font-size: 18px;
13     color: #34495e;
14     line-height: 1.6;
15     text-align: center;
16 }
```

```
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (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:   ../GitHub-Pr\303\251sentation-Ousmane-KA.docx
        modified:   style.css

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

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
$
```

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



This screenshot shows the continuation of the terminal session in VS Code. The 'git add *.css' command has been executed successfully. Then, the 'git commit -m' command is used to create a new commit with the message 'changement du fichier css'. The terminal output shows the commit was created with hash 8d71516 and that 1 file was changed with 22 insertions. The prompt '\$' is visible at the bottom.

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

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
$ git add *.css

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
$ git commit -m "changement du fichier css"
[master 8d71516] changement du fichier css
1 file changed, 22 insertions(+)

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
$
```

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.

```
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
• $ git branch FichierJS

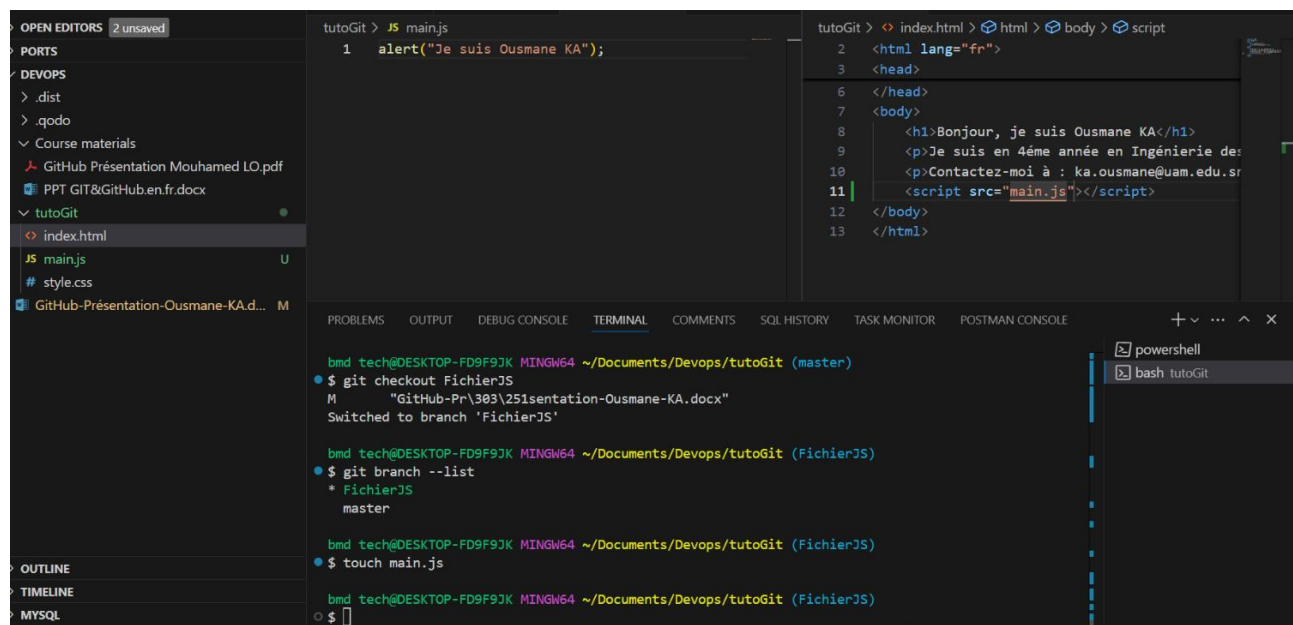
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
• $ git checkout FichierJS
M      "GitHub-Pr\303\251sentation-Ousmane-KA.docx"
Switched to branch 'FichierJS'

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (FichierJS)
• $ git branch --list
* FichierJS
  master

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (FichierJS)
```

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

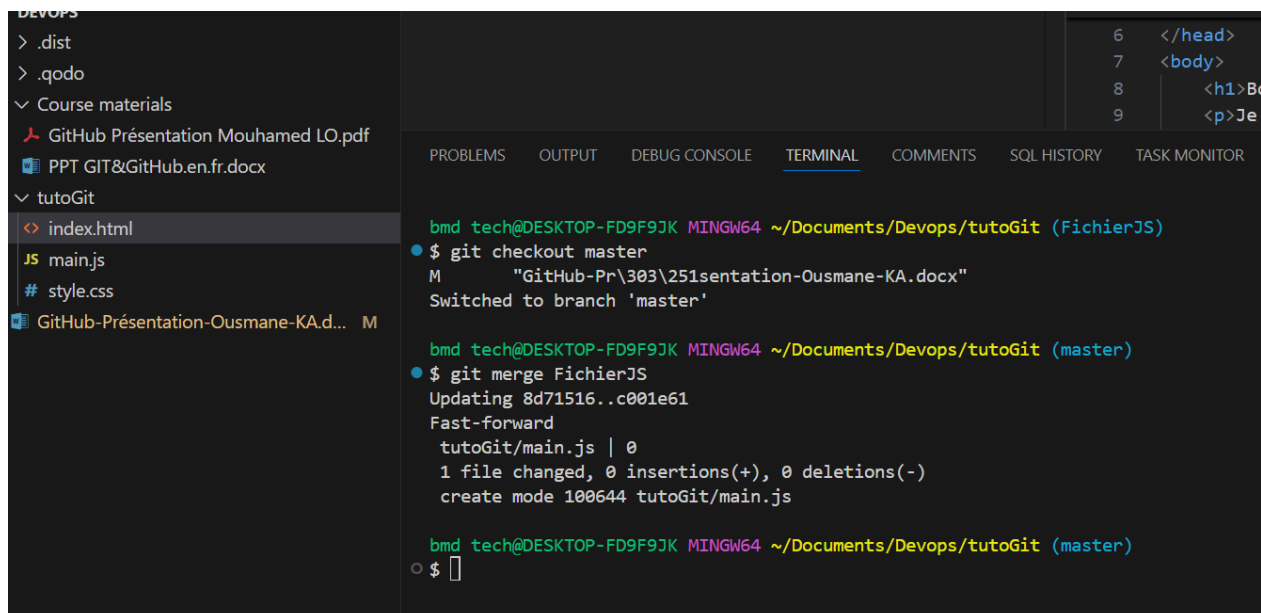
```
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (FichierJS)
• $ git add .

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (FichierJS)
• $ git commit -m "Ajout d'un fichier javascript"
[FichierJS c001e61] Ajout d'un fichier javascript
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 tutoGit/main.js

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (FichierJS)
○ $
```

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**



```
DEVOPS
> .dist
> .qodo
v Course materials
  GitHub Présentation Mouhamed LO.pdf
  PPT GIT&GitHub.en.fr.docx
v tutoGit
  <> index.html
  JS main.js
  # style.css
  GitHub-Présentation-Ousmane-KA.d... M

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS SQL HISTORY TASK MONITOR

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (FichierJS)
• $ git checkout master
M
  "GitHub-Pr\303\251sentation-Ousmane-KA.docx"
Switched to branch 'master'

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
• $ git merge FichierJS
Updating 8d71516..c001e61
Fast-forward
tutoGit/main.js | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 tutoGit/main.js

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops/tutoGit (master)
○ $
```

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

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)



Required fields are marked with an asterisk (*).

Owner *  Ousoka / Repository name *

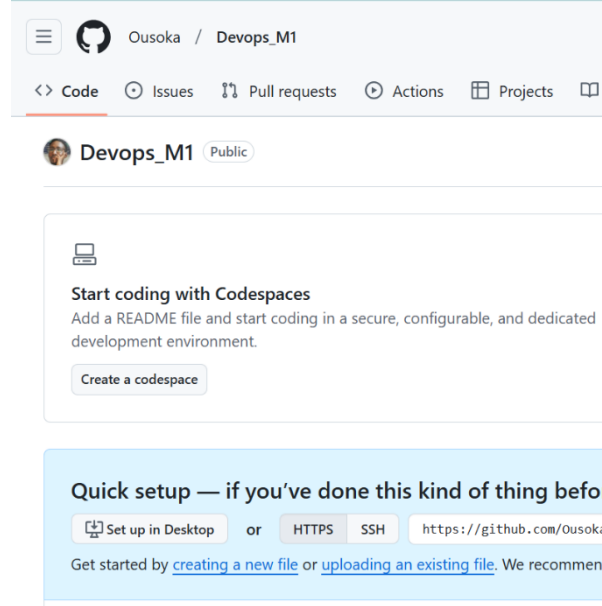
✔ Devops_M1 is available.

Great repository names are short and memorable. Need inspiration? How about [bookish-spork](#) ?

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

On donne un nom au répertoire et une description



The screenshot shows the GitHub repository page for 'Devops_M1' by user 'Ousoka'. The 'Code' tab is selected, showing the repository name and a 'Public' badge. Below this, there's a section titled 'Start coding with Codespaces' with a 'Create a codespace' button. Further down, a 'Quick setup' section offers options to 'Set up in Desktop', 'HTTPS', or 'SSH', with a link to the repository URL: 'https://github.com/Ousoka/Devops_M1'. It also mentions 'Get started by creating a new file or uploading an existing file. We recommend'.

...or create a new repository on the command line

```
echo "# Devops_M1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/Ousoka/Devops_M1.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/Ousoka/Devops_M1.git
git branch -M master
git push -u origin master
```

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.

```
$ cd ..

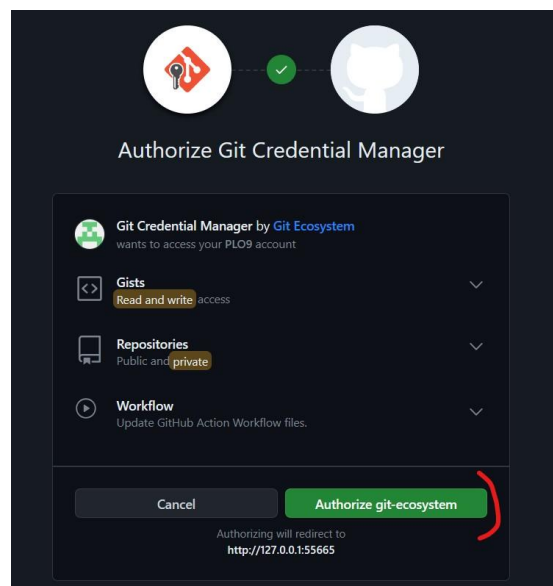
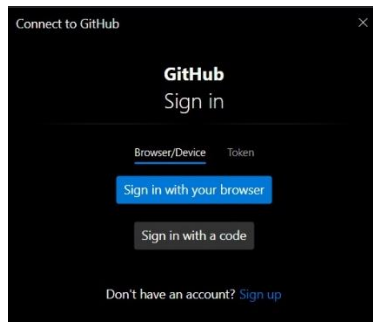
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
$ git remote add origin https://github.com/Ousoka/Devops_M1.git

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
$ git branch -M master

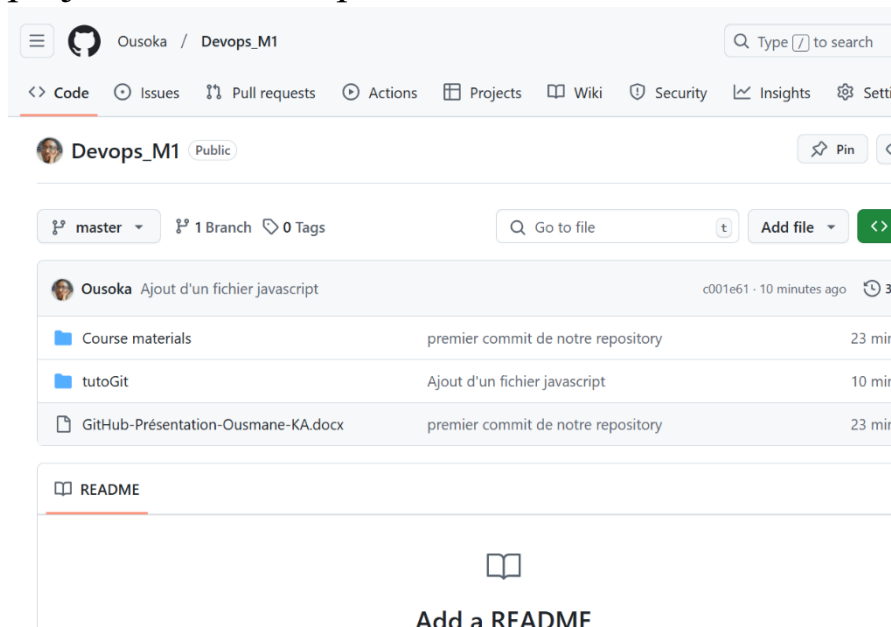
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
$ git push -u origin master
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (16/16), 6.08 MiB | 1.27 MiB/s, done.
Total 16 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Ousoka/Devops_M1.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
$
```

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

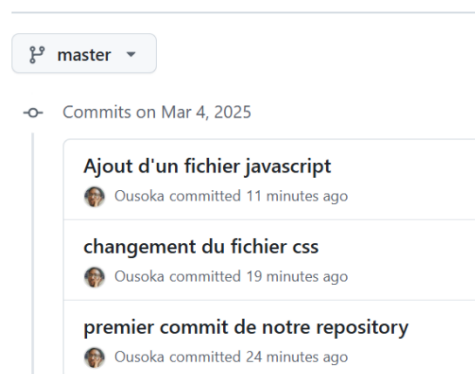


On revient sur GitHub, rafraichissons la page et nous verrons que notre projet a été bien importé.



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

Commits

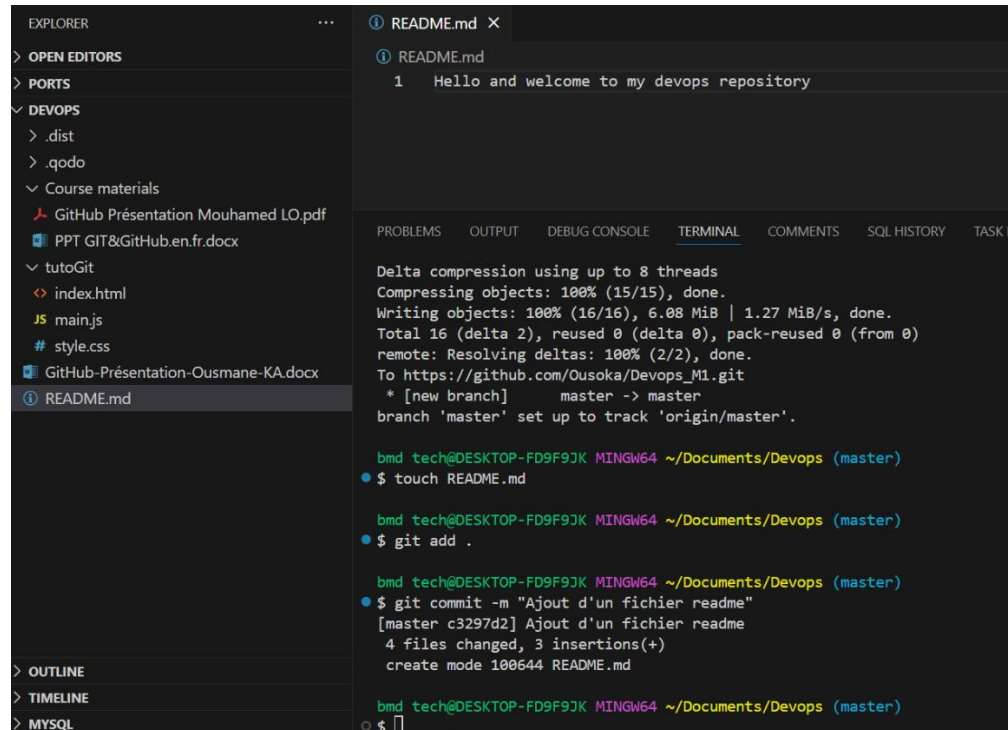


Sur github on voit souvent que certains repository ont une description en dessous. Cela est possible avec le fichier README.

On crée 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*



```
EXPLORER
> OPEN EDITORS
> PORTS
DEVOPS
> .dist
> .qodo
> Course materials
  GitHub Présentation Mouhamed LO.pdf
  PPT GIT&GitHub.en.fr.docx
  tutoGit
    index.html
    JS main.js
    # style.css
  GitHub-Présentation-Ousmane-KA.docx
  README.md

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS SQL HISTORY TASK M...
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (16/16), 6.08 MiB | 1.27 MiB/s, done.
Total 16 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Ousoka/Devops_M1.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
● $ touch README.md

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
● $ git add .

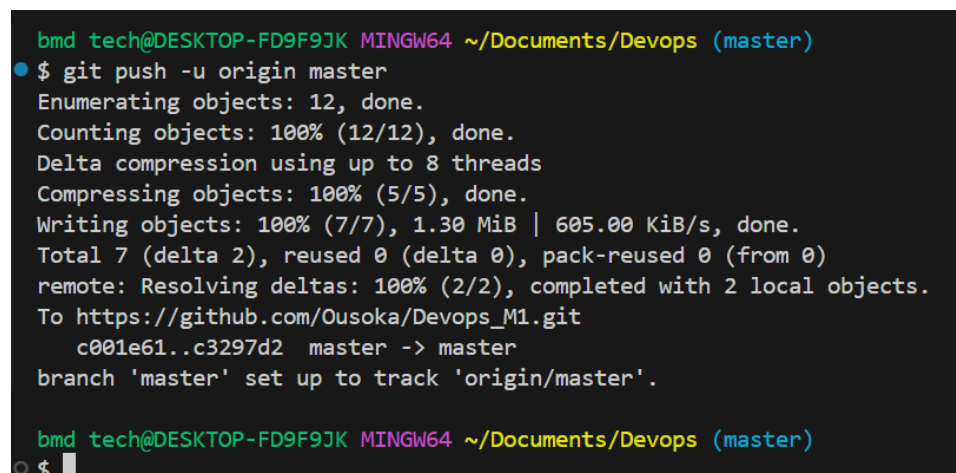
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
● $ git commit -m "Ajout d'un fichier readme"
[master c3297d2] Ajout d'un fichier readme
4 files changed, 3 insertions(+)
create mode 100644 README.md

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
○ $
```

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

Il faut le pousser avec git push

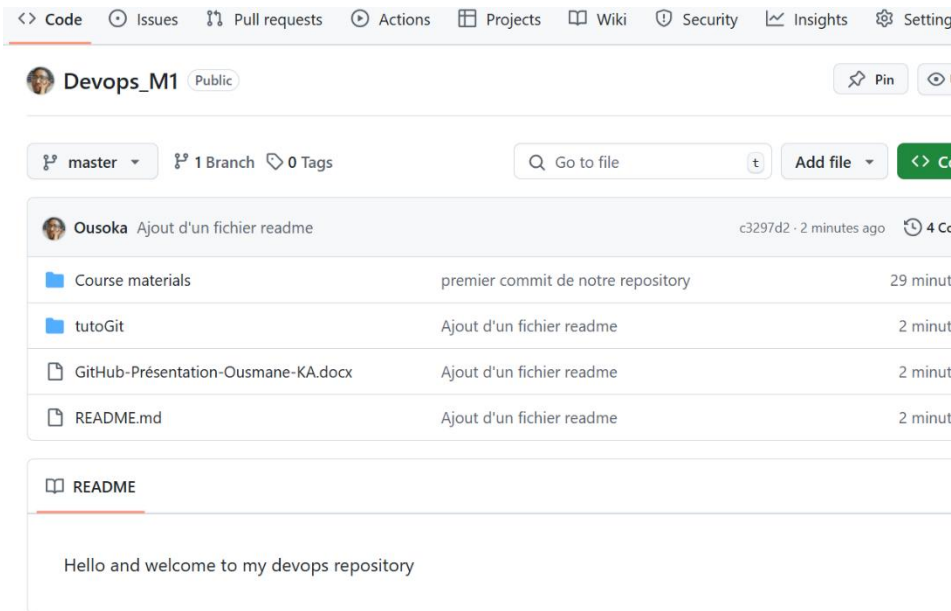
Cela nous permet de mettre tous les changements sur notre repository de github.



```
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
● $ git push -u origin master
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 1.30 MiB | 605.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Ousoka/Devops_M1.git
 c001e61..c3297d2  master -> master
branch 'master' set up to track 'origin/master'.

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
○ $
```

Après rafraichissement on peut maintenant voir

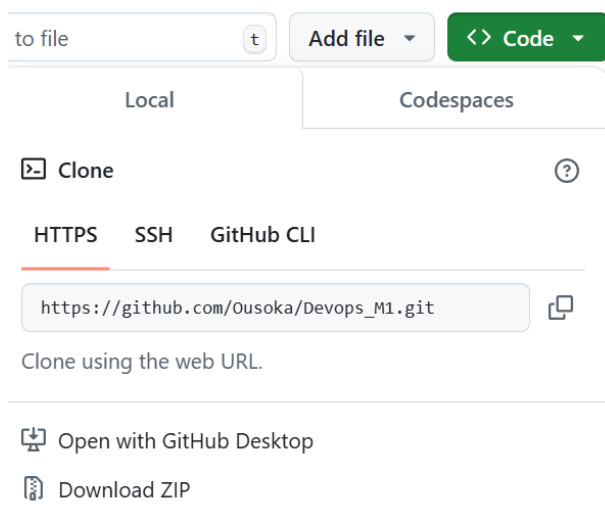


En cas de changement par l'un des membres de l'équipe.

```
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
$ git pull
Already up to date.

bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
```

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



```
bmd tech@DESKTOP-FD9F9JK MINGW64 ~/Documents/Devops (master)
$ git clone https://github.com/Ousoka/Devops_M1.git
```

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.