

 	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

# Créer un repository avec GIT et GITHUB sous Windows

1. PREREQUIS----- 1

2. CREER UN REPOSITORY EN 8 ETAPES ----- 2

2.1. ETAPE 1→ CREER UN NOUVEAU REPOSITORY SUR GITHUB ----- 2

2.1.1. Le fichier README ----- 4

2.1.2. Le fichier .gitignore ----- 4

2.2. ETAPE 2→ CREER UN NOUVEAU DOSSIER VIDE EN LOCAL ET L’INITIALISER ----- 5

2.2.1. Syntaxe de la commande git init ----- 6

2.3. ETAPE 3→ GENERER DES FICHIERS DANS LE DEPOT LOCAL----- 8

2.4. ETAPE 4 → RENOMMER LA BRANCHE LOCALE DE MASTER EN MAIN ----- 9

2.4.1. Syntaxe de la commande pour renommer la branche master en main----- 9

2.5. ETAPE 5 → INDEXER LES FICHIERS-----10

2.5.1. Syntaxe de la commande git add -----10

2.6. ETAPE 6 → CAPTURER UN INSTANTANE DES CHANGEMENTS ACTUELLEMENT INDEXES -----12

2.6.1. Syntaxe de la commande git commit -----12

2.7. ETAPE 7 → CREER LE LIEN POUR RELIER LE DEPOT LOCAL AU DEPOT DISTANT SITUE SUR GITHUB -----12

2.7.1. Syntaxe de la commande git remote add-----13

2.8. ETAPE 8 → ENVOYER LE COMMIT REALISE EN LOCAL SUR LE DEPOT DISTANT SITUE SUR GITHUB-----13

2.8.1. Syntaxe de la commande git push -----14

3. RESUME SCHEMATIQUE -----15

## 1. Prérequis

Créer un repository avec Git et GitHub est une étape essentielle, dans le processus de développement, pour gérer et partager le code source de ses projets.

Voici un guide étape par étape qui indique les points essentiels.

Avant de pouvoir créer un repository sur GitHub, il faut satisfaire à certaines conditions.

- Créez un compte GitHub si ce n'est pas déjà fait. Inscrivez-vous sur [GitHub](https://github.com)<sup>1</sup>.
- Réaliser l’**installation de Git** sur votre machine pour cela il faut :
  - Disposer des **droits administrateur** sur votre machine.
  - [Téléchargez](https://git-scm.com/downloads)<sup>2</sup> Git.
- **Configurer Git** sur votre machine Windows

Après avoir remplis ces prérequis , on peut rentrer dans le vif du sujet : créer un dépôt sur GitHub pour y déposer son code source.

<sup>1</sup> <https://github.com>

<sup>2</sup> <https://git-scm.com/downloads>

	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

## 2. Créer un repository en 8 étapes

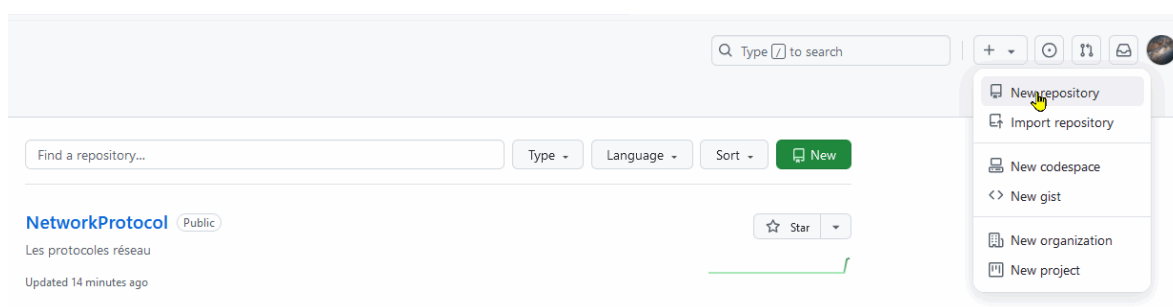
Voyons ces huit étapes :

### 2.1. Etape 1 → Créer un nouveau repository sur GitHub

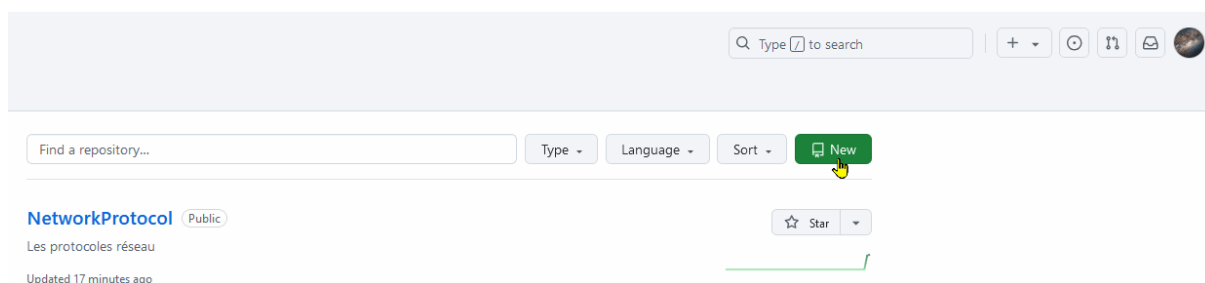
Aller sur [github.com](https://github.com) et la première opération consiste à créer un nouveau repository qui contiendra le code du projet.

Sur la plateforme GitHub, on dispose de deux manières pour créer un nouveau dépôt :

La première :



La seconde :



Ce nouveau dépôt s'appellera : « mySite »

Il sera Public.

Pas de fichier README pour l'instant ; il sera ajouté plus tard.

**⚠** Ne pas ajouter de fichier README ni de .gitignore ! Pour éviter les risques de conflits lors des synchronisations.

  Nombre de pages : 15	<b>Créer un repository avec GIT et GITHUB sous Windows</b>	Réalisé le :	12/11/2024
		Modifié le :	04/12/2024

## 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 \*

 Djamelikra ▾

Repository name \*

/ mySite

✓ mySite is available.

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

Description (optional)

Site exemple



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.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

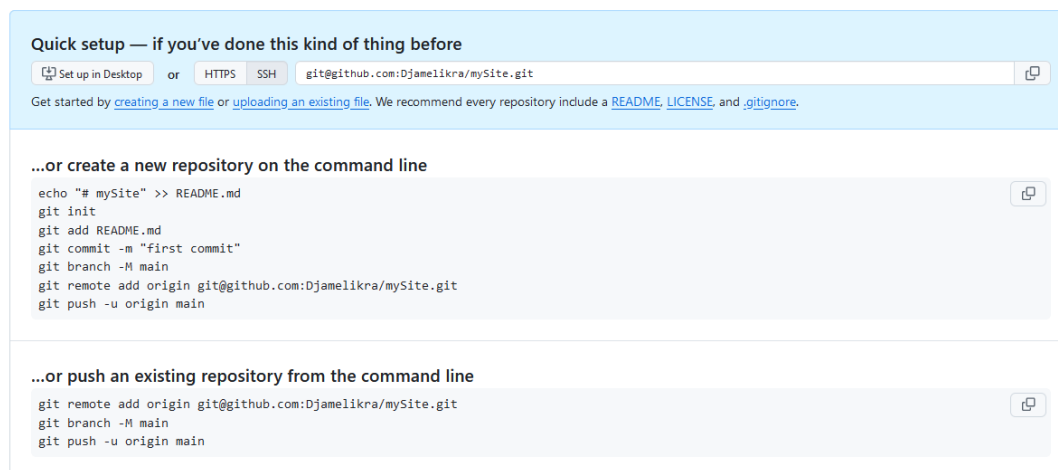
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

ⓘ You are creating a public repository in your personal account.

Create repository

 	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

⚠ Pour l'instant on n'utilise pas les commandes proposées par github .



On nous suggère, ici, de créer les éléments README et .gitignore cela mérite quelques explications:

### 2.1.1. Le fichier README

Dans la sphère GitHub, le fichier README est comme une porte d'entrée vers le projet.

En effet, c'est un document écrit en [Markdown](#)<sup>3</sup>, qui se trouve à la racine du dépôt et qui fournit une description concise et claire du projet.

Le nom "README" i.e. "read me" (lis-moi), résume bien son but : être lu en premier par toute personne qui découvre le projet.

Le fichier README remplit plusieurs fonctions essentielles :

- **Première impression** : C'est souvent la première chose que les gens voient lorsqu'ils visitent votre dépôt. Un README bien écrit donne une bonne impression et incite les autres à s'intéresser à votre projet.
- **Documentation essentielle** : Il fournit une documentation de base indispensable pour comprendre et utiliser votre projet.
- **Collaboration** : Il facilite la collaboration en permettant à d'autres développeurs de comprendre rapidement votre code et de contribuer au projet.
- **Référencement**<sup>4</sup> : Les moteurs de recherche peuvent indexer le contenu de votre README, ce qui peut aider de nouvelles personnes à découvrir votre projet.

### 2.1.2. Le fichier .gitignore

Le fichier .gitignore est un fichier texte **placé à la racine d'un dépôt Git**. Il sert à indiquer à Git quels fichiers ou dossiers il doit **ignorer** lors de l'ajout (add) ou du suivi (commit) de modifications.

<sup>3</sup> <https://fr.wikipedia.org/wiki/Markdown>

<sup>4</sup> Le SEO

	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

En d'autres termes, les fichiers listés dans .gitignore ne seront pas inclus dans l'historique de version.

Pourquoi utiliser un fichier .gitignore?

- **Confidentialité** : exclure les fichiers contenant des informations sensibles, comme des mots de passe ou des clés d'API.
- **Optimisation de l'espace** : ignorer les fichiers volumineux ou générés automatiquement qui ne doivent pas être versionnés (comme les fichiers de compilation, les logs, etc.).
- **Propreté du dépôt** : maintenir le dépôt net en n'incluant que le code source et les fichiers nécessaires au projet.

Le fichier .gitignore utilise une syntaxe simple:

- Les lignes commençant par un # sont des commentaires.
- Les lignes vides : Ignorées par Git.

Exemple de fichier .gitignore :

```

*.gitignore.txt - Bloc-notes
Fichier Edition Format Affichage Aide
# Ignorer tous les fichiers qui commencent par
.*
# Ignorer Tous les fichiers d'extensions
.tmp
.docx
.xlsb
.one
# Ignorer les fichiers temporaires
*~
# Ignorer les fichiers générés par le compilateur
*.class
*.o
# Ignorer le contenu des dossiers
/WorkItems
/Temp
Ln 13, Col 34 100% Windows (CRLF) UTF-8

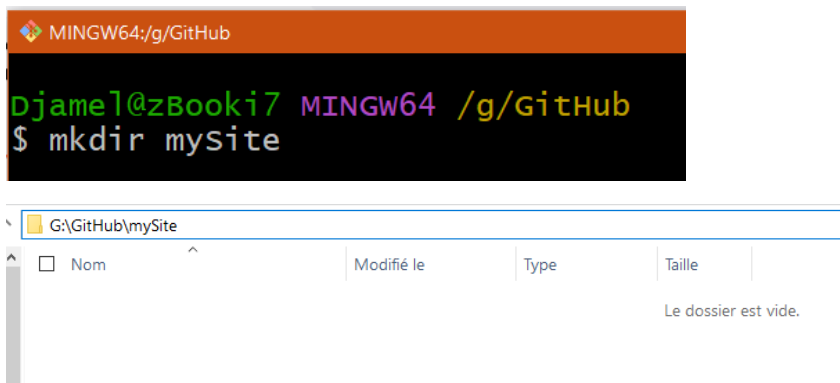
```

Après ces précisions, passons maintenant à la deuxième étape pour créer un dépôt local.

## 2.2. Etape 2→ Créer un nouveau dossier vide en local et l'initialiser

Il est conseillé de mettre le même nom pour le dossier en local et celui distant. Mais ce n'est pas obligatoire.

	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024



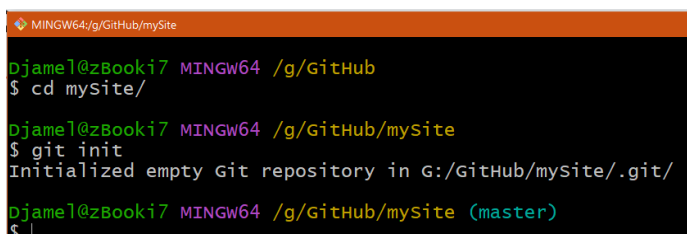
Se déplacer dans le répertoire de travail et l'initialiser avec la commande `git init`.

Une fois exécutée, cette commande va initialiser un nouveau dépôt Git dans ce répertoire. Cela signifie qu'elle va créer un **sous-répertoire caché nommé .git** qui contiendra toutes les informations nécessaires pour gérer le projet sous Git.

### 2.2.1. Syntaxe de la commande `git init`

- Sa syntaxe de base pour initialiser un dépôt est très simple : `git init`
- La syntaxe pour initialiser un dépôt et nommer la branche principale `main` au lieu de `master` est : `git init --initial-branch main`

On va se contenter, ici, de la syntaxe par défaut :



⚠ Il faut vérifier qu'on est bien placé dans le bon dossier avant de faire un `git init`

Une fois le projet initialisé grâce à la commande `git init`, un nouveau **dossier caché .git** apparaît à la **racine dossier** du projet.

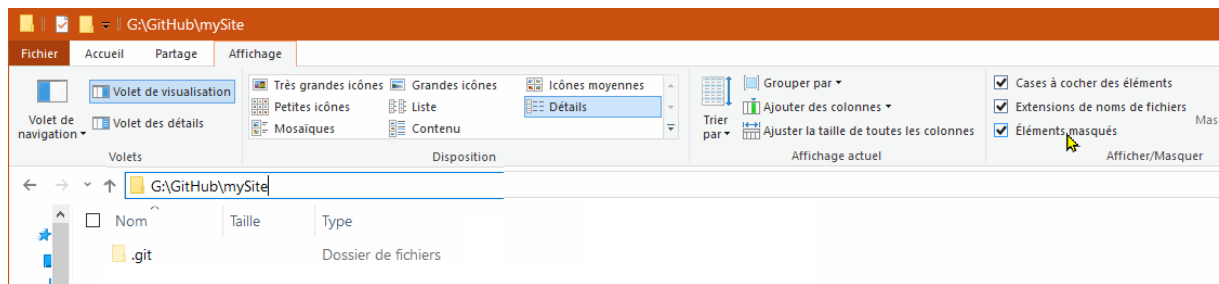
📁 Ce dossier est caché par défaut.



Afficher un dossier caché :











1. Ouvrir l'Explorateur de fichiers à l'endroit du répertoire contenant le projet git
2. Sélectionner Affichage > Éléments masqués.

	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024



Ce répertoire .git contient plusieurs sous-répertoires et fichiers, chacun ayant une fonction bien spécifique :

- **objects**: Ce répertoire contient les objets Git. Un objet Git peut être un fichier, un répertoire, un commit, un tag... Chaque objet est identifié par un hash SHA-1 unique qui représente son contenu.
- **refs**: Ce répertoire contient des pointeurs vers les objets Git. Les branches, les tags et HEAD (qui pointe vers le commit actuel) sont des références.
- **HEAD**: Ce fichier spécial contient le SHA-1 du commit auquel pointe la branche courante.
- **config**: Ce fichier contient les configurations spécifiques à votre dépôt Git.
- **index**: Ce fichier, aussi appelé "staging area", est une zone tampon où vous préparez les modifications que vous souhaitez valider.
- **hooks**: Ce répertoire contient des scripts qui sont exécutés automatiquement à différents moments du cycle de vie d'un dépôt Git (par exemple, avant ou après un commit).

G:\GitHub\mySite\.git			
<input type="checkbox"/> Nom	Type	Taille	
 hooks	Dossier de fichiers		
 info	Dossier de fichiers		
 logs	Dossier de fichiers		
 objects	Dossier de fichiers		
 refs	Dossier de fichiers		
 COMMIT_EDITMSG	Fichier	1 Ko	
 config	Fichier	1 Ko	
 description	Fichier	1 Ko	
 HEAD	Fichier	1 Ko	
 index	Fichier	1 Ko	



**Il ne faut pas modifier le dossier .git manuellement !**

	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

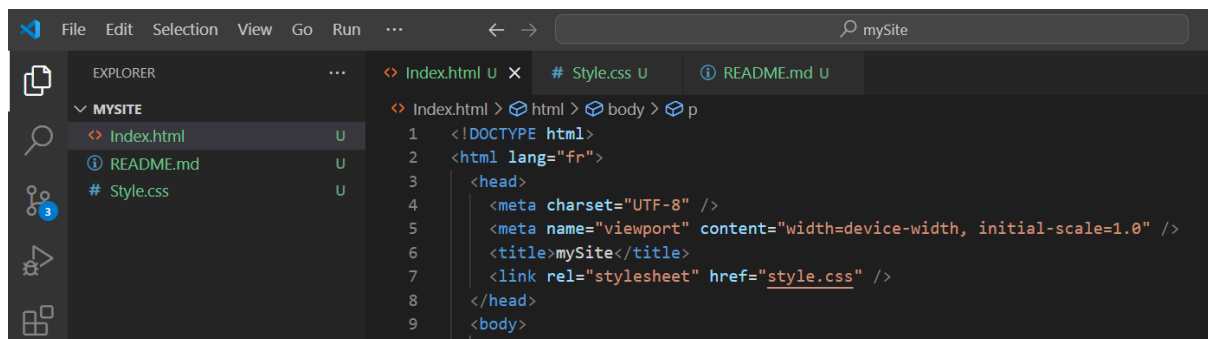
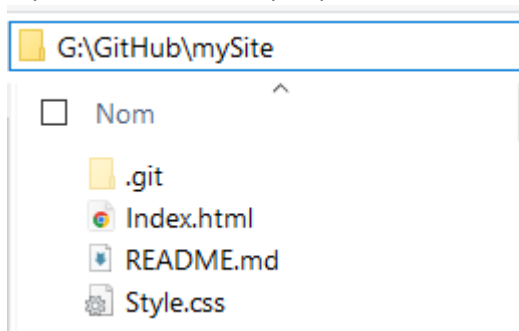
En effet, car le dossier git a une structure complexe et peut facilement être corrompue, toute tentative de modification manuelle peut le détruire. Il donc, est **fortement recommandé** de n'utiliser que les commandes Git pour interagir avec votre dépôt.

Une fois le dépôt initialisé , il est temps de le remplir.

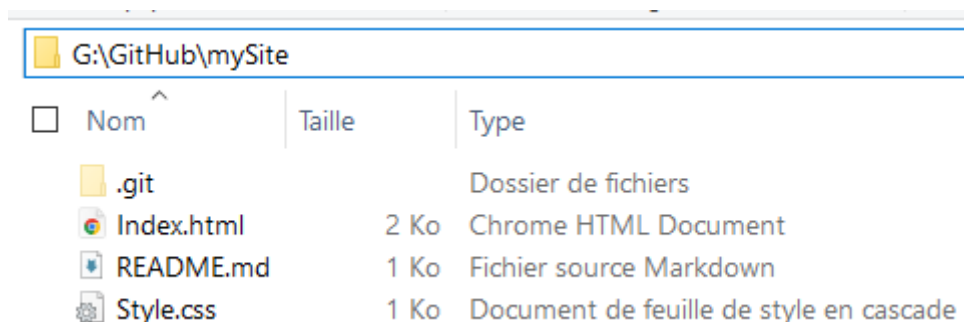
### 2.3. Etape 3→ Générer des fichiers dans le dépôt local

On va ouvrir [VS Code](#)<sup>5</sup> pour générer trois fichiers :

1. README qui porte une extension .md pour : Markdown<sup>6</sup> : qui sert à présenter le projet.
2. Index.html : contenu du site.
3. Style.css : fichier de style qui sera relié au fichier précédent.



Dossier contenant les 3 fichiers :



<sup>5</sup> <https://code.visualstudio.com/>

<sup>6</sup> <https://fr.wikipedia.org/wiki/Markdown>



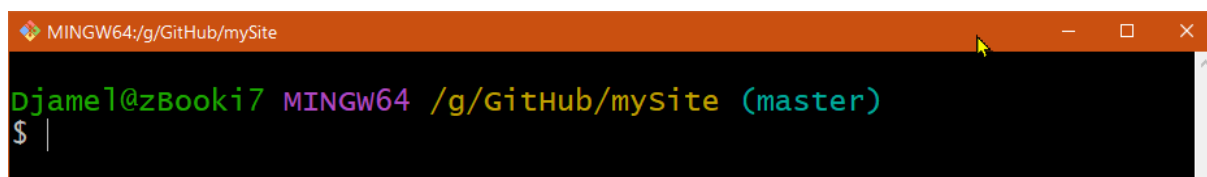
	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

Après avoir ajouté du contenu au dossier , il convient d'adapter le nom de la branche principale.

## 2.4. Etape 4 → Renommer la branche locale de master en main

Dès le [premier octobre 2020](https://github.com/github/renaming)<sup>7</sup>, tous les nouveaux dépôts que vous créerez utiliseront main plutôt que master pour désigner la branche par défaut », annonce GitHub. La mesure cible également les dépôts existants qui s'appuient sur le terme master pour désigner la branche principale.

Nom de la branche locale actuelle : `master`



```

MINGW64:/g/GitHub/mySite
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (master)
$ |

```

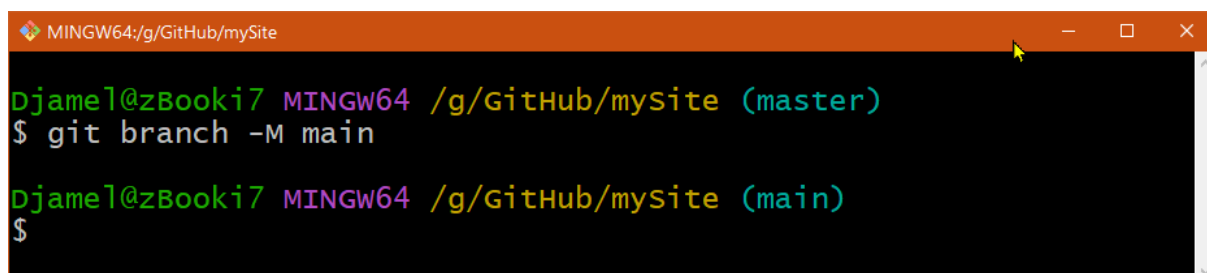
On va donc renommer la branche **master** en **main**.

### 2.4.1. Syntaxe de la commande pour renommer la branche master en main

Cette commande admet 2 syntaxes Bash<sup>8</sup>:

- `git branch -m master main` :
  - Renomme simplement la branche locale master en main.
  - Elle ne pousse pas les changements sur le dépôt distant.
  - On l'utilise pour vérifier les changements effectués localement avant de les pousser sur le dépôt distant .
- `git branch -M master main` :
  - Renomme la branche locale **ET** pousse les changements sur le dépôt distant.
  - Elle établit un suivi entre la nouvelle branche locale et la nouvelle branche distante .
  - Elle est, donc, utilisée pour gagner du temps : renommer la branche locale et mettre à jour le dépôt distant **en une seule étape**.

Pour optimiser l'écriture du script, on va utiliser la seconde syntaxe :



```

MINGW64:/g/GitHub/mySite
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (master)
$ git branch -M main
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$

```

<sup>7</sup> <https://github.com/github/renaming>

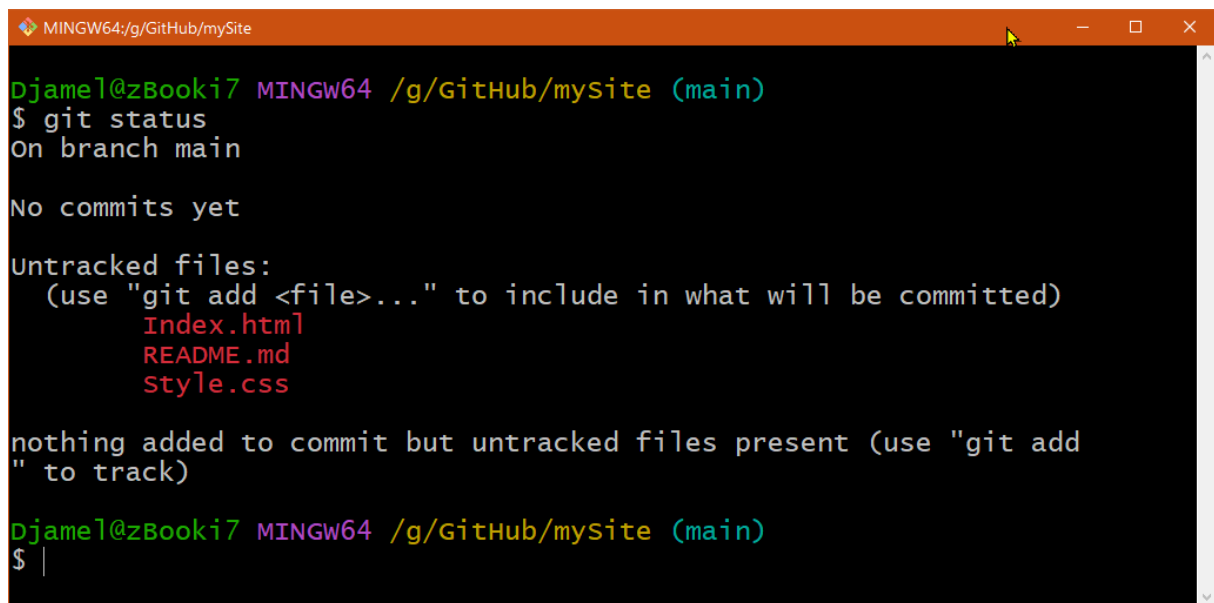
<sup>8</sup> Sensible à la casse

	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

Maintenant que la branche principale a été renommée, on est prêt à ajouter les nouveaux fichiers dans la zone d'index.

## 2.5. Etape 5 → Indexer les fichiers

Au préalable, on va vérifier le contenu du dossier de travail avec la commande : **git status** :



```

MINGW64:/g/GitHub/mySite
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$ git status
On branch main

No commits yet

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

nothing added to commit but untracked files present (use "git add
" to track)
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$ |

```

Nous avons bien les 3 fichiers qui sont « untracked » : donc non encore indexés.

À ce stade il faut utiliser la commande `git add`.

Cette commande sert à ajouter les fichiers modifiés à la zone de staging ou zone d'index.

Lorsqu'un fichier est modifié, Git considère que ce fichier est amendé localement. Pour inclure ces changements dans le prochain commit, on doit donc utiliser cette commande.

### 2.5.1. Syntaxe de la commande git add

La syntaxe de base pour ajouter :

- un seul fichier modifié : `git add <fichier>`
- plusieurs fichiers modifiés :
  - `git add .`
  - `git add -A`
- un dossier : `git add dossier/`

On va donc taper la commande : `git add .` pour indexer les 3 fichiers à la fois.

Ensuite les noms des fichiers passeront du rouge au vert.

  Nombre de pages : 15	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :  Modifié le :	12/11/2024  04/12/2024
--	--	----------------------------------	------------------------------

```

MINGW64:/g/GitHub/mySite
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Index.html
        README.md
        style.css

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

Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$ git add .

Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$

```

Si on fait un **git status**, on voit que les fichiers passent en vert ce qui confirme leur indexation :

```

MINGW64:/g/GitHub/mySite
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Index.html
        README.md
        style.css

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

Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$ git add .

Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Index.html
        new file:   README.md
        new file:   style.css

Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$

```

	<h2 style="text-align: center;">Créer un repository avec GIT et GITHUB sous Windows</h2>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

Il est temps de faire un commit, c'est à dire prendre une photo instantanée de l'état du projet à un l'instant t.

## 2.6. Etape 6 → Capturer un instantané des changements actuellement indexés

En fait il s'agit de **générer une nouvelle version des fichiers** avec la commande : **`git commit`**.

Elle permet :

- d'enregistrer les modifications effectuées dans le projet Git.
- **Elle crée un instantané de du dépôt à un moment donné,**
- permettant ainsi de revenir en arrière si nécessaire.

En termes plus techniques, un commit est un enregistrement qui contient :

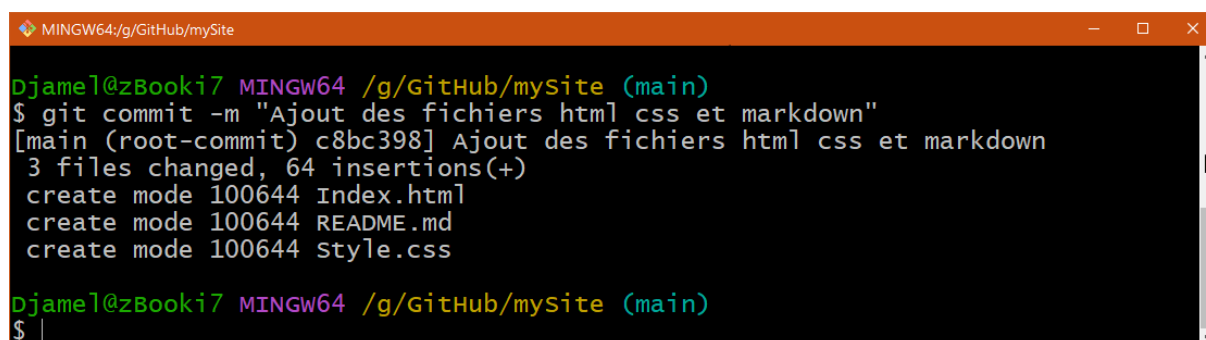
- Un **identifiant** unique (hash) : Une sorte d'empreinte digitale qui identifie de manière unique ce commit.
- Un **message** de commit : Un texte qui décrit les modifications apportées dans ce commit. Ce message est crucial pour comprendre pourquoi et comment ces changements ont été faits.
- Un **pointeur** vers le commit précédent : Cela crée une chaîne de commits, formant ainsi l'historique du projet.

### 2.6.1. Syntaxe de la commande git commit

La syntaxe est :

```
git commit -m "Message descriptif et concis"
```

Avec cette syntaxe l'éditeur de texte ne s'ouvre pas dans le terminal et on intègre le texte du commit directement dans la commande.



```

MINGW64/g/GitHub/mySite
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$ git commit -m "Ajout des fichiers html css et markdown"
[main (root-commit) c8bc398] Ajout des fichiers html css et markdown
3 files changed, 64 insertions(+)
create mode 100644 Index.html
create mode 100644 README.md
create mode 100644 style.css
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$

```

Jusqu'à présent on a travaillé dans le dépôt en local, il convient à ce stade de relier ce dernier au dépôt distant situé sur GitHub.

## 2.7. Etape 7 → Créer le lien pour relier le dépôt local au dépôt distant situé sur GitHub

A ce niveau il y a deux types de connexion avec des protocoles différents : https et SSH.

	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

### 2.7.1. Syntaxe de la commande git remote add

On va utiliser, ici, la version en https de la connexion, elle est utilisée par défaut, sa syntaxe est :

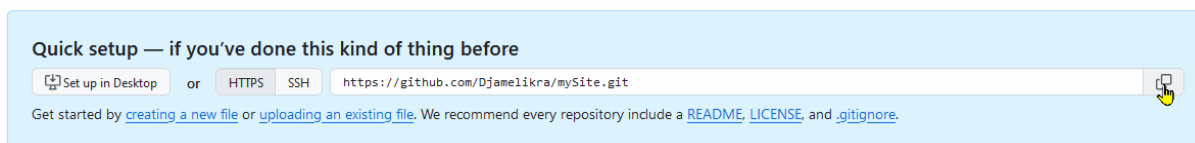
```
git remote add origin <url>
```

Cette commande est une instruction fondamentale en Git qui permet d'établir une connexion entre le dépôt local, celui qui se trouve sur l'ordinateur, et un dépôt distant : celui qui se trouve sur GitHub<sup>9</sup>



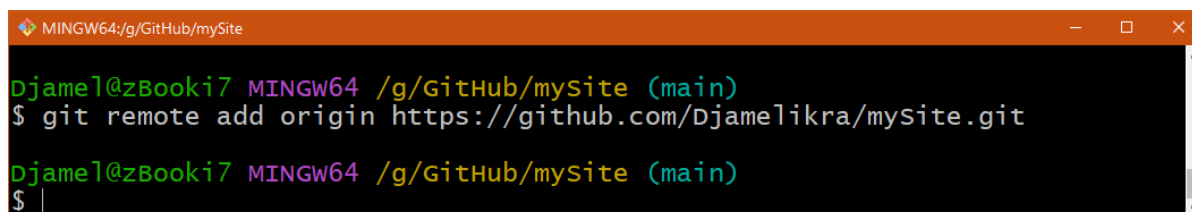
**origin** est un nom par convention que l'on donne généralement au premier dépôt distant que l'on ajoute. Il sert d'**alias** pour **simplifier les commandes ultérieures**. En effet, ainsi il ne sera pas nécessaire de retaper l'URL complète du dépôt distant à chaque fois que l'on souhaite interagir avec lui.

Pour avoir cette url il faut copier le lien, ici en https<sup>10</sup>, distant du dépôt GitHub



et intégrer l'url dans la commande :

```
git remote add origin https://github.com/Djamelikra/mySite.git
```



Maintenant, on est prêt à passer à l'ultime étape : envoyer les modifications de la branche locale `main` vers la branche `main` du dépôt distant `origin`.

## 2.8. Etape 8 → Envoyer le commit réalisé en local sur le dépôt distant situé sur GitHub

Pour cela on utilise la commande `git push`. Elle sert à :

<sup>9</sup>Ou tout autre plateforme d'hébergement de code comme GitLab, BitBucket...

<sup>10</sup> La méthode reste la même en SSH.

	<h1>Créer un repository avec GIT et GITHUB sous Windows</h1>	Réalisé le :	12/11/2024
Nombre de pages : 15		Modifié le :	04/12/2024

- À pousser les modifications de la branche locale `main` vers la branche correspondante sur le dépôt distant nommé `origin`.
- Partager les modifications avec d'autres collaborateurs.
- Sauvegarder les modifications sur un serveur distant.

### 2.8.1. Syntaxe de la commande `git push`

Voici 2 syntaxes de cette commande :

- `git push origin main` : c'est la syntaxe basique.
- `git push -u origin main` : lie la branche locale à la branche distante.
  - `"-u"` remplace le paramètre `"--set-upstream"`.
  - Après cette commande, on peut simplement utiliser `git push` sans spécifier la branche distante et locale.

```

MINGW64:/g/GitHub/mySite
Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.08 KiB | 1.08 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Djamelikra/mySite.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Djamel@zBooki7 MINGW64 /g/GitHub/mySite (main)
$

```

Il ne reste plus qu'à vérifier que tout s'est bien passé en allant sur le dépôt distant ; et en rafraichissant la page on retrouve bien nos 3 fichiers qui ont bien été transférés :

