

Qu'est-ce que Git ?

Git est un logiciel de versionning créé en 2005 par Linus Torvalds, le créateur de Linux.

Un logiciel de versioning, ou logiciel de gestion de version, est un logiciel qui permet de conserver un historique des modifications effectuées sur un projet afin de pouvoir rapidement identifier les changements effectués et de revenir à une ancienne version en cas de problème.

Les logiciels de gestion de versions sont quasiment incontournables aujourd'hui, car ils facilitent grandement la gestion de projets et parce qu'ils permettent de travailler en équipe de manière beaucoup plus efficace.

Parmi les logiciels de gestion de versions, Git est le leader incontesté et il est donc indispensable pour tout développeur de savoir utiliser Git.

À quoi sert concrètement un système de gestion de version ?

Imaginez que vous possédiez un site web. À chaque fois que vous voulez modifier quelque chose sur le site ou tester une fonctionnalité, vous êtes obligé d'effectuer une sauvegarde du site avant

l'implémentation afin de pouvoir le restaurer si quelque chose se passe mal.

Jusqu'ici, le procédé est contraignant, car vous allez devoir effectuer une sauvegarde complète et restaurer la sauvegarde en cas de problème, mais cela ne semble pas très complexe ni insurmontable.

Imaginez maintenant que vous soyez 10 à travailler sur le même site web, en vous occupant chacun de développer des fonctionnalités différentes, mais qui peuvent être liées entre elles. L'organisation du travail est ici beaucoup plus complexe puisqu'il va falloir s'assurer :

- que les fonctionnalités sur lesquelles travaille un développeur ne rentrent pas en conflit avec les fonctionnalités sur lesquelles travaillent des autres développeurs
- que chaque développeur possède une version actualisée du site pour tester et implémenter ses fonctionnalités

Ici, l'idée la plus logique serait de mettre en place un serveur distant qui contiendrait l'historique des modifications faites par chaque développeur afin que chacun ait accès aux avancées des autres.

Chaque développeur pourrait également copier l'intégralité du contenu du serveur pour travailler en local sur sa machine. On vient ainsi de créer **un système de gestion de version décentralisé**.

Git permet de coordonner le travail entre plusieurs personnes en conservant un historique des changements effectués sur des fichiers.

Git permet à différentes versions d'un même fichier de coexister.

Les développeurs travaillant avec Git ont accès à l'historique des modifications pour l'intégralité du projet et peuvent ainsi savoir quels changements ont été faits par rapport à leur version des fichiers, qui a fait ces changements, etc.

Dans le langage des systèmes de gestion de version, la copie de l'intégralité des fichiers d'un projet et de leur version située sur le serveur central est appelé un dépôt. Git appelle également cela "repository" ou "repo" en abrégé.

Qu'est-ce que GitHub ?

GitHub est un service en ligne qui permet d'héberger des dépôts ou repo Git. C'est le plus grand hébergeur de dépôts Git du monde.

Une grande partie des dépôts hébergés sur GitHub sont publics, ce qui signifie que n'importe qui peut télécharger le code de ces dépôts et contribuer à leur développement en proposant de nouvelles fonctionnalités.

Pour récapituler, et afin d'être bien clair sur ce point : Git est un logiciel de gestion de version, tandis que GitHub est un service en ligne d'hébergement de dépôts Git qui fait office de serveur central pour ces dépôts.

Il est totalement gratuit pour des projets ouverts au public, mais il propose également des formules payantes pour activer quelques fonctions avancées.

GitHub propose également de nombreux autres services très intéressants, par exemple :

- partager du code source avec d'autres développeurs.
- signaler et gérer les problèmes ou bugs de votre code source via les issues.
- partager des portions de code via les Gists
- proposer des évolutions pour un projet opensource.

1 - TÉLÉCHARGEMENT ET INSTALLATION

L'outil GIT est présent sur le site suivant :

Windows :

<https://git-scm.com/download/win>

Mac OS :

<https://www.git-scm.com/download/mac>

Linux :

<https://www.git-scm.com/download/linux>

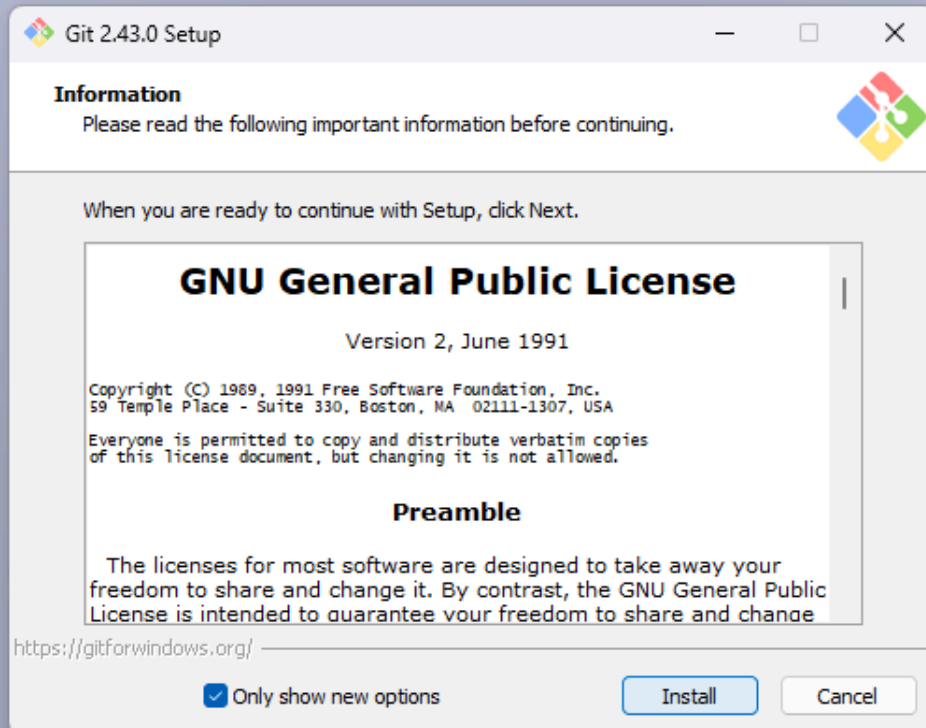




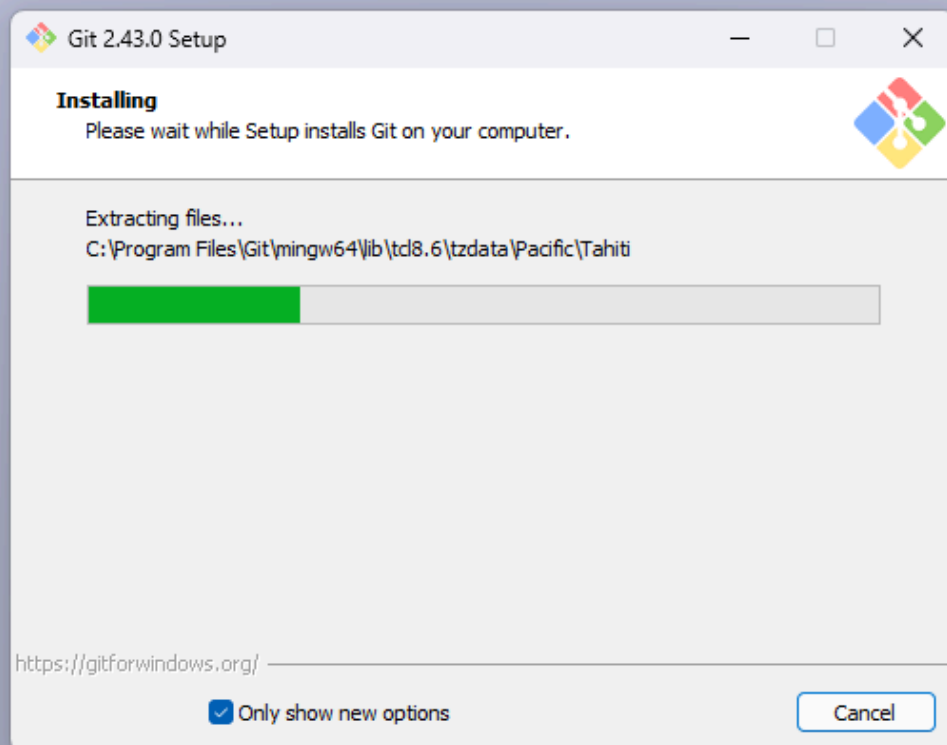
10MENTIONWEB
FORMATION

COMPRENDRE GIT & GITHUB

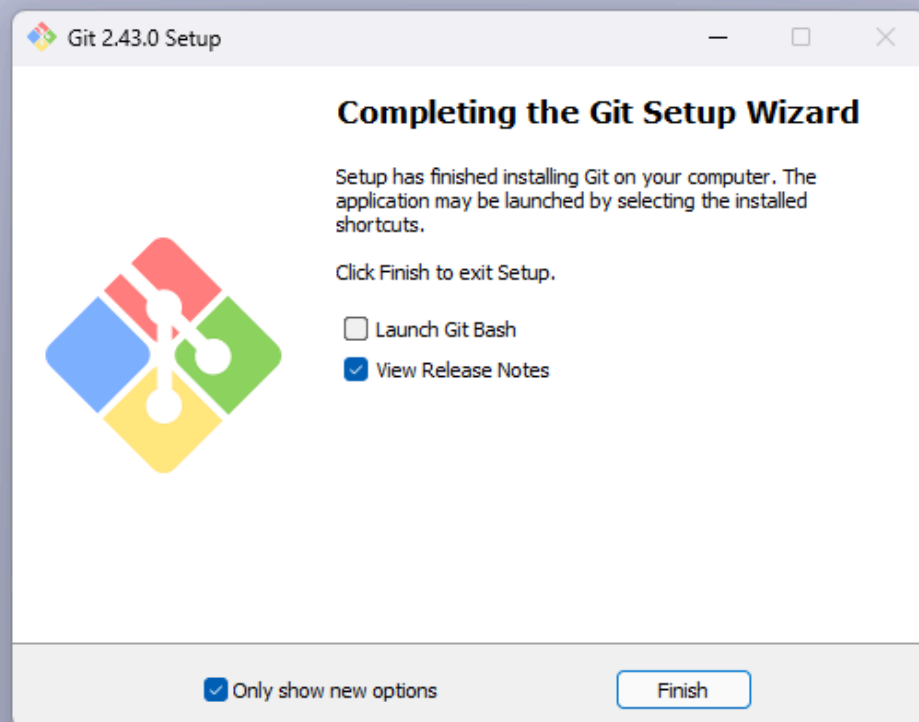
INSTALLATION SOUS WINDOWS



Installation



L'installation est réussie, si vous obtenez l'image suivante :
Il n'est pas nécessaire de lancer le bash git.



2 - VÉRIFICATION

Dans un terminal*, rédigez la commande : **git --version** vous devriez obtenir le numéro de version de votre programme GIT.

* pour ouvrir un terminal : dans la Recherche de votre barre des tâches, écrivez "cmd"

Résultat de commande :

```
C:\Users\sahar>git --version  
git version 2.43.0.windows.1
```

3 - CONFIGURATION

L'outil permet de versionner les fichiers et les modifications et après chaque modification, il va attacher cette dernière à votre identité. GIT propose l'enregistrement de quelques paramètres dans un fichier, l'accès à la configuration se fait à l'aide de commandes, décrites ci-après.

git config --global user.name "nom"

Définit le nom que vous voulez associer à toutes vos opérations de commit.

git config --global user.email "adresse email"

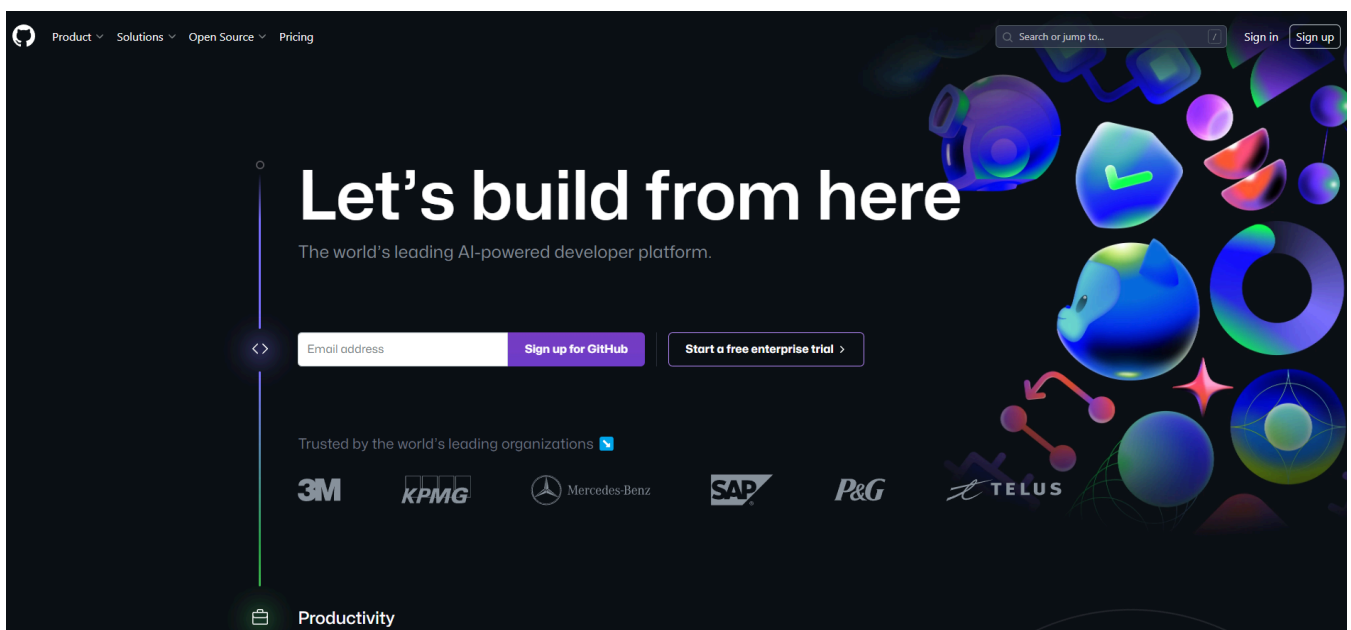
Définit l'email que vous voulez associer à toutes vos opérations de commit

Le paramètre **--global** permet de conserver les informations pour tous les prochains projets.

La commande **git config -l** permet de vérifier l'ensemble des paramètres de configuration.

Création compte GitHub

Créez votre compte GitHub sur le site : github.com



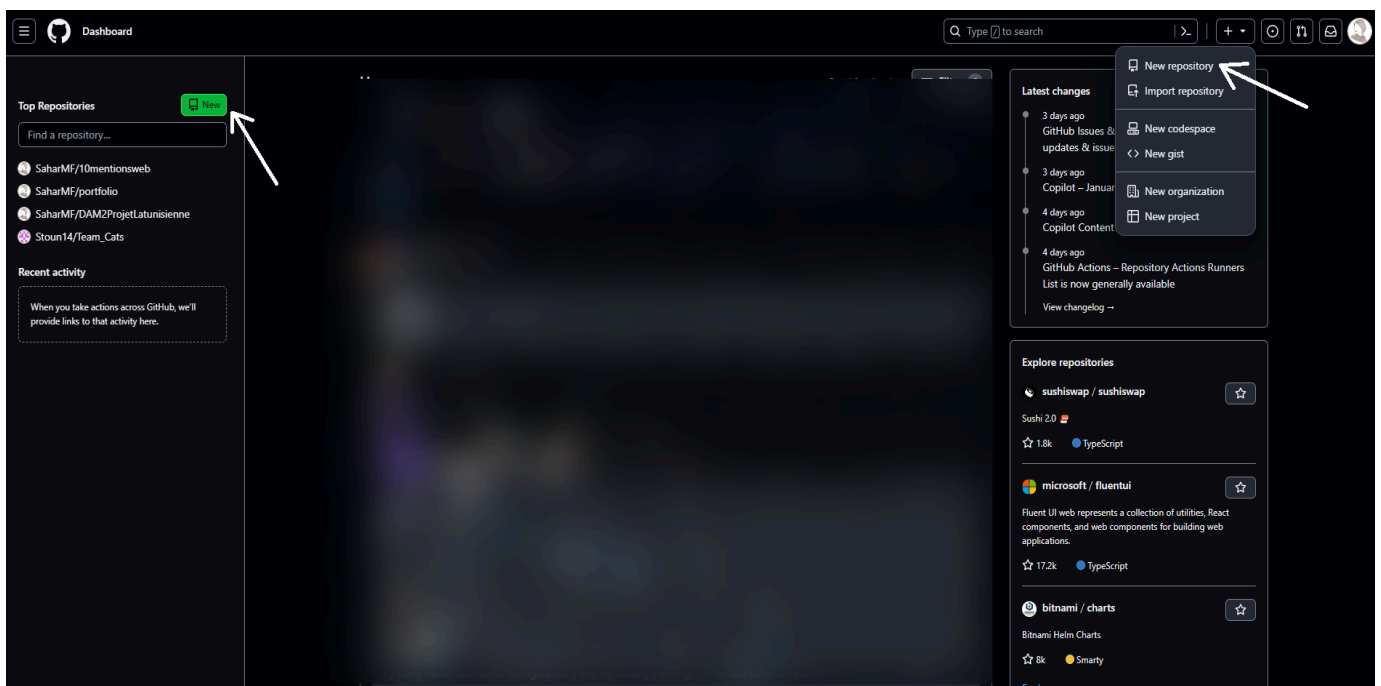
Pour stocker le répertoire de travail (votre code) en ligne, vous devez créer un repo = repository (répertoire ou référentiel).

Création d'un référentiel (repo)

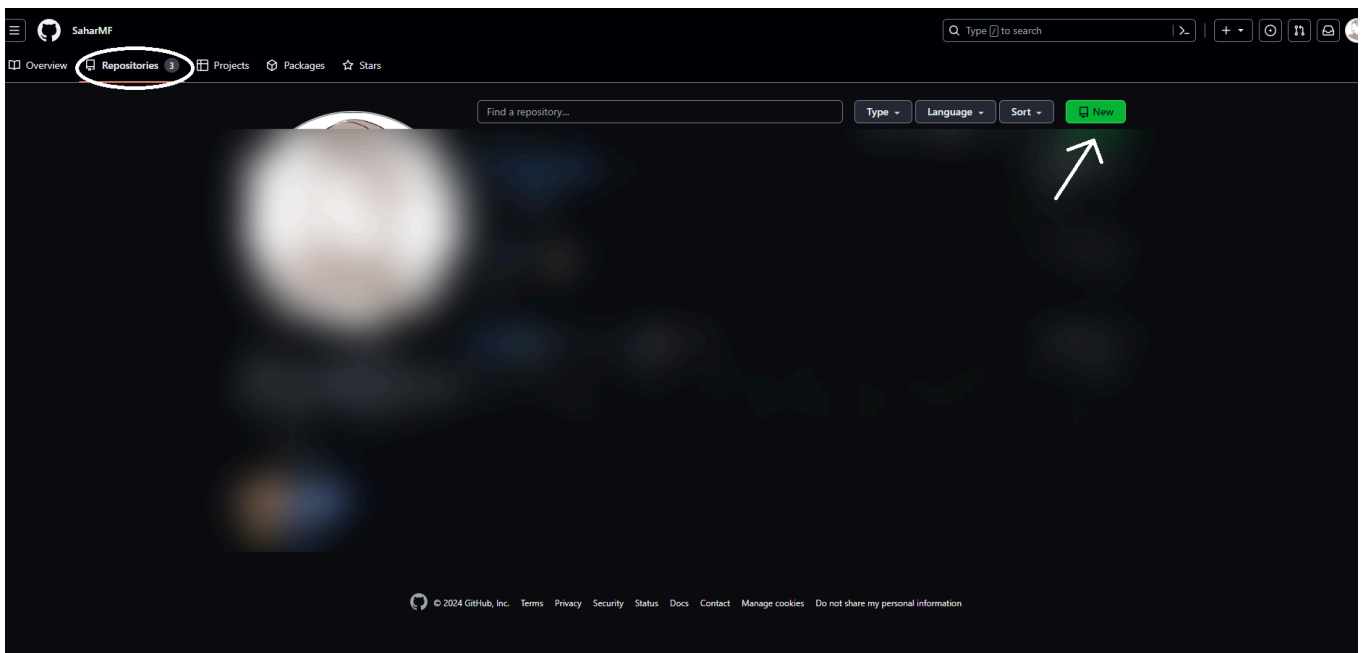
Les étapes :

- 1 - Assurez-vous d'être connecté à votre compte GitHub
- 2 - Accédez à la page d'accueil de GitHub, pour créer un repo, vous avez trois endroits pour le faire :

- Cliquez sur le bouton "+" dans le coin supérieur droit, dans le menu déroulant, choisissez l'option "New repository" pour créer un nouveau repo.
- Cliquez sur le bouton vert "New" qui se trouve à gauche sur votre page d'accueil.



- À partir de la page Repositories de votre profil, cliquez sur le bouton vert "New"



3 - Remplissez les détails du nouveau référentiel :

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 * SaharMF /

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

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.

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

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

Choose a license

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.

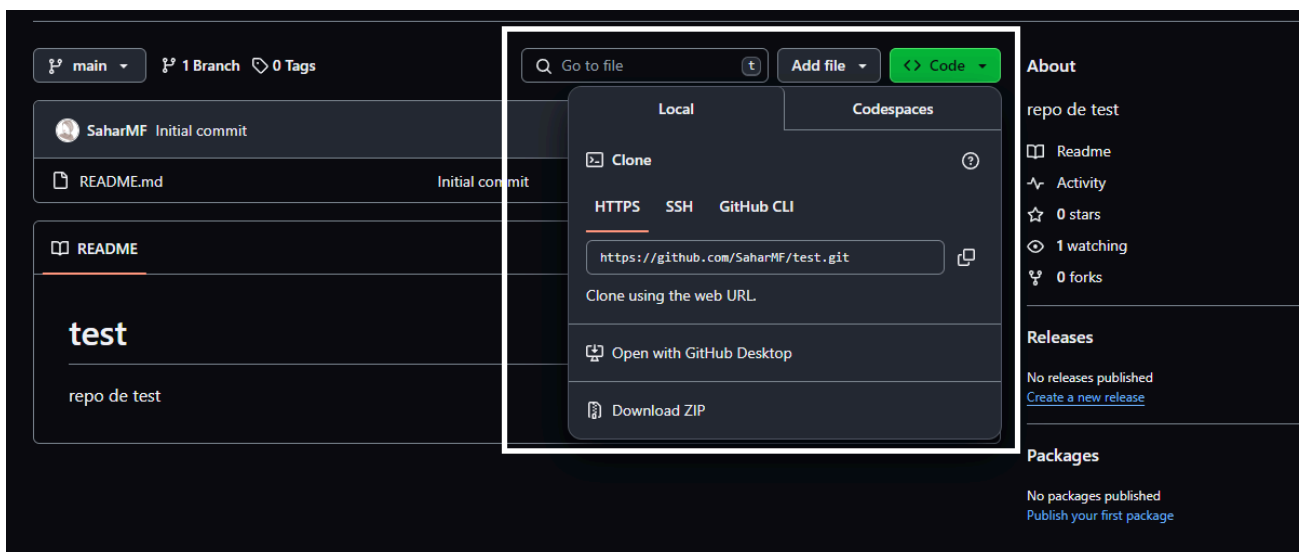
- Public ou Private (Public ou privé) : Choisissez si votre repo sera public (visible par tous) ou privé (accessible uniquement par les collaborateurs autorisés).
- Initialize this repository with a README (Initialiser ce référentiel avec un README) : Vous pouvez cocher cette option si vous souhaitez créer un fichier README dès le début. Cela peut être utile pour fournir des informations sur votre projet.
- Choisissez une licence :
Si vous avez l'intention de partager votre code avec les autres, il est généralement recommandé de choisir une licence open source. GitHub propose une liste de licences que vous pouvez choisir lors de la création du référentiel.
- Cliquez sur le bouton "Create repository" (Créer le référentiel) : Une fois que vous avez rempli tous les détails nécessaires, cliquez sur le bouton vert pour créer votre nouveau référentiel.

Récupération du repo sen local

Pour récupérer un référentiel (repo) GitHub sur votre machine locale, vous pouvez utiliser la commande **git clone**. Voici les deux étapes principales :

1. Trouvez l'URL du Repo GitHub :

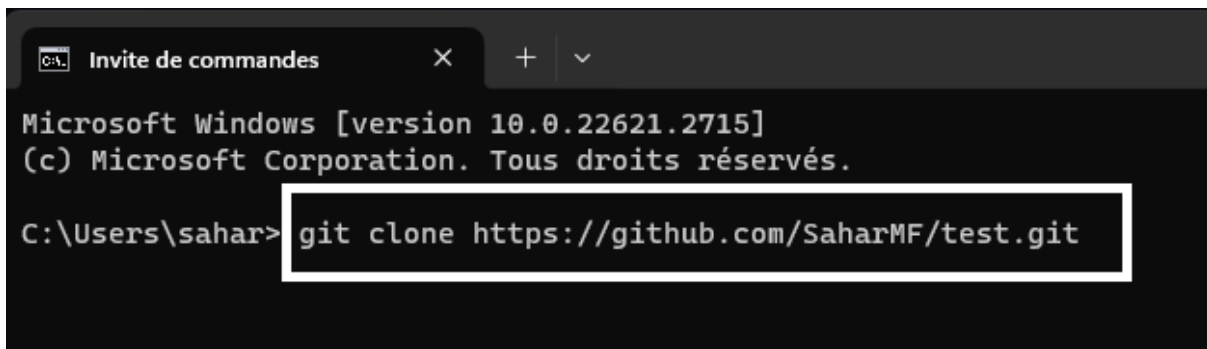
- Sur la page du référentiel GitHub, cliquez sur le bouton vert "Code".
- Copiez l'URL fournie (assurez-vous de sélectionner le protocole que vous souhaitez utiliser, HTTPS ou SSH).



2. Utilisez la commande git clone en local :

- Ouvrez votre terminal sur votre machine.
- Naviguez vers le répertoire dans lequel vous souhaitez cloner le repo en utilisant la commande `cd` (Change Directory).
- Exécutez la commande `git clone` en ajoutant l'URL que vous avez copiée à l'étape 1.

Par exemple, si vous utilisez HTTPS :



```
Microsoft Windows [version 10.0.22621.2715]  
(c) Microsoft Corporation. Tous droits réservés.  
  
C:\Users\sahar> git clone https://github.com/SaharMF/test.git
```

Envoyer un répertoire local en ligne

1. Créez un nouveau référentiel sur GitHub

2. Initialisez un référentiel local en référentiel Git

Ouvrez un terminal, accédez au répertoire de votre projet et exécutez la commande : **git init**

3. Ajoutez les fichiers au suivi de Git :

Utilisez la commande **git add** pour ajouter tous les fichiers à l'index de suivi de Git.

Par exemple, pour ajouter tous les fichiers, utilisez : “**git add .**”
ou pour ajouter un fichier index.html, utilisez : **git add index.html**

4. Effectuez un commit local :

Utilisez la commande **git commit** pour créer un commit local avec les fichiers que vous avez ajoutés à l'index.

Chaque commit a un identifiant unique et contient des informations telles que l'auteur du commit, la date et l'heure de création, ainsi qu'un message descriptif.

Par exemple : **git commit -m "Premier commit - Ajout des fichiers initiaux"**

5. Renommer la branche actuelle en “main” :

La commande **git branch -M main** est utilisée pour renommer la branche actuelle (généralement “master”) en “main”. Cette commande est souvent utilisée lorsque vous souhaitez adopter un nouveau standard de nommage des branches principales en “main” plutôt qu’en “master”

- **git branch**: Cette partie de la commande est utilisée pour manipuler les branches dans Git.
- **-M** : Cette option signifie “move” (déplacer) et “force” (forcer). Elle est utilisée pour renommer une branche existante en spécifiant le nouveau nom.
- **main** : C’est le nouveau nom que vous donnez à la branche principale. Vous pouvez remplacer “main” par le nom de votre choix si vous adoptez un autre standard de nommage.

6. Associez votre référentiel local au référentiel distant sur GitHub :

- Sur la page du référentiel GitHub, copiez l’URL du référentiel (HTTPS ou SSH).
- Utilisez la commande **git remote add** pour ajouter le référentiel distant. Par exemple, avec HTTPS :
git remote add origin
<https://github.com/votre-nom-utilisateur/nom-du-repo.git>

7. Poussez les modifications vers GitHub :

Utilisez la commande **git push** pour pousser les modifications de votre référentiel local vers le référentiel distant sur GitHub. Utilisez -u pour définir la branche upstream (par exemple, main ou master) : **git push -u origin main**.