PLAN DE TRAVAIL

INTRODUCTION

I. QUE CE QUE GITHUB
II. QUEL SONT SES FONTIONALITER
III. QUEL SONT SES AVANTAGES
IV. QUEL SONT SES INCONVENIENTS
V. COMMENT S'INSRIRE SUR GITHUB
VI. LES DIFFERENTES COMMANDE UTILISER PAR GITHUB
VII. COMMENT TRAVAILLER SUR GITHUB

CONCLUSION

INTRODUCTION

Tout avant nous ne pouvons parler de GitHub sans avoir parler de git ce serais sauter les étapes comme nous le savon bien git est un logiciel de gestion de version crée en 2005 par Linus torvalds, le créateur de linux ce logiciel permet donc de conserver un historique des modifications effectuer sur un projet afin de pouvoir rapidement identifier les changements effectuer et de revenir à une ancienne version en cas de problème. Donc il est donc incontournable pour une meilleure programmation. Tout au long de notre réflexion nous devrons nous attarder donc sur ce que GitHub, ses différentes fonctionnalités, ses avantages et ses inconvénients.

I. QUE CE QUE GITHUB

GITHUB est donc un service en ligne qui permet d'héberger des dépôts ou repo git .c'est un site web et un service cloud qui aide les développeurs à stocker et a gérer leur codes ,ainsi suivre et contrôler les modification qui lui sont apportées ,c'est ainsi le plus grand des hébergeurs de dépôt git du monde .Au vue de cela une grand partie des dépôts hébergés sont public ce qui signifie que n'importe qui peut télécharger le codes de ses dépôts et contribuer à leurs évolution et mémé en proposant de nouvelle fonctionnalités.

II. Les différentes fonctionnalités de GitHub

GitHub est une plateforme collaborative qui permet aux différentes équipes techniques de stocker et de modifier du code pour tous leurs projets (développement des sites web programmes informatique développement application mobiles etc.). Les développeurs peuvent partager publiquement leurs projets afin que d'autre professionnel puissent contribuer au développement

Elle regorge dont beaucoup de fonctionnalité permis les quels :

- ✓ Le code collaboratif
- ✓ Automatisation
- ✓ Gestion de projet
- ✓ Gestion d'équipe

III. LES DIFFERENTS AVANTAGES DE GITHUB

GitHub nous présente plusieurs avantages tel que :

- Une modification apporter à tous moment
- Une sauvegarde historique
- Un développement en équipe (une bonne gestion de l'équipe)
- Très flexible
- Capable t'interagir avec lui sur votre machine grâce au ligne de commande
- C'est un développement baser sur le tronc (le fait d'avoir un code de base ou un espace pour plusieurs développeuses ou chacun est appeler a coder sur son espace et à la fin ils peuvent fusionner leur travail)

IV. LES INCONVENIENT DU GITHUB

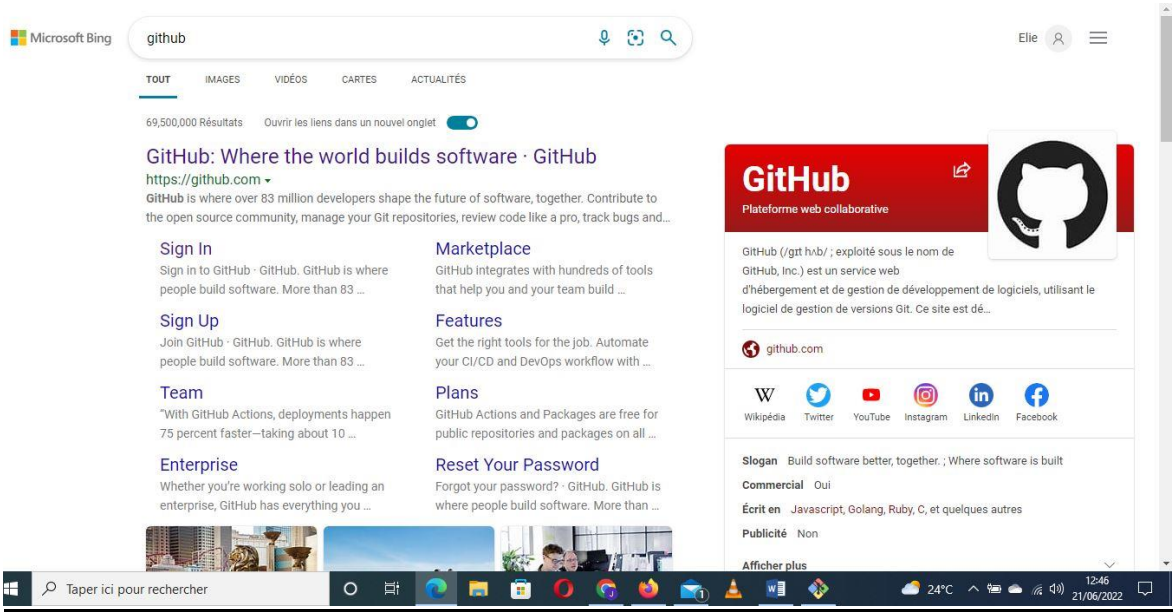
Comme tout logiciel GitHub a aussi des limites car rien n'est parfait sinon pourquoi chercherons toujours à améliorer nous avons donc identifié quelque limite à ce magnifique logiciel tel que :

- ✓ La sécurité (le fait d'être ouvert au grand public met la sécurité des donner souvent en danger tel que le piratage insertion des virus)
- ✓ Le prix cout souvent très élever des Git pour son hébergement

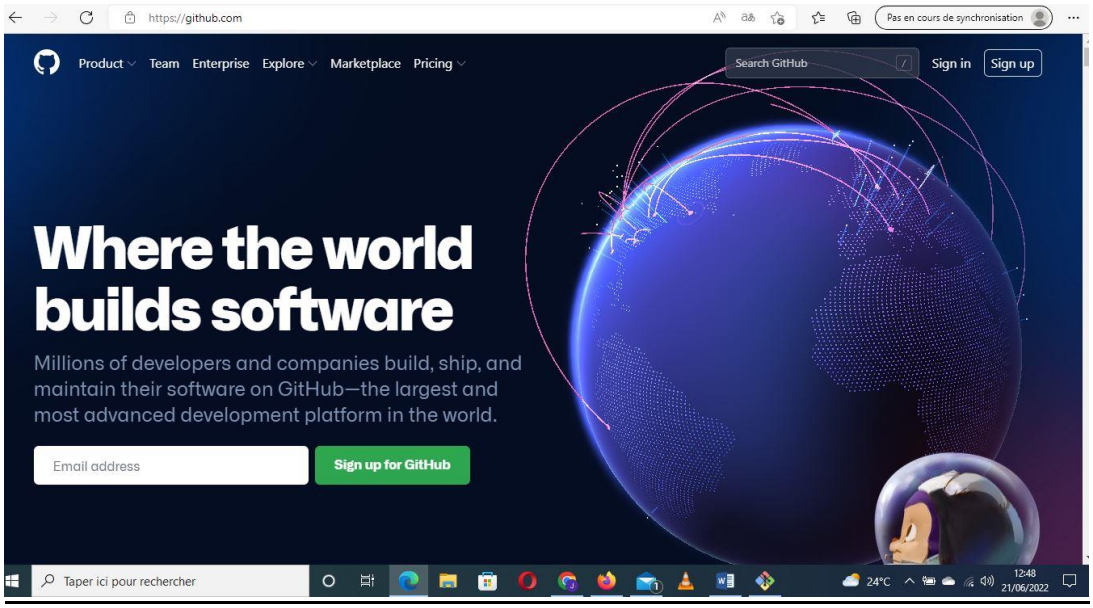
V. COMMENT S'INSRIRE SUR GITHUB

Dans un premier pour pouvoir avoir accès à GITHUB il faut tout d'abord s'inscrire pour cela on se rend sur un moteur de recherche et on tape

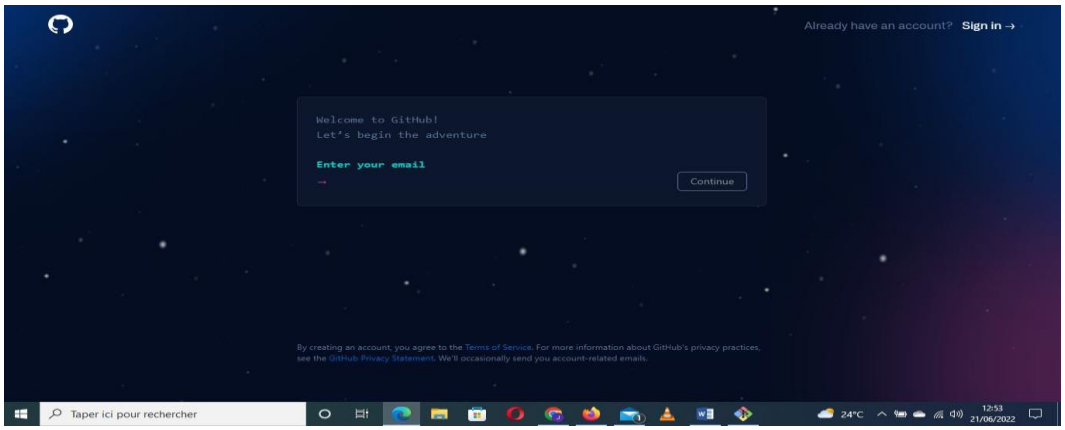
- GITHUB .COM
  Alors vous tomber sur cette page



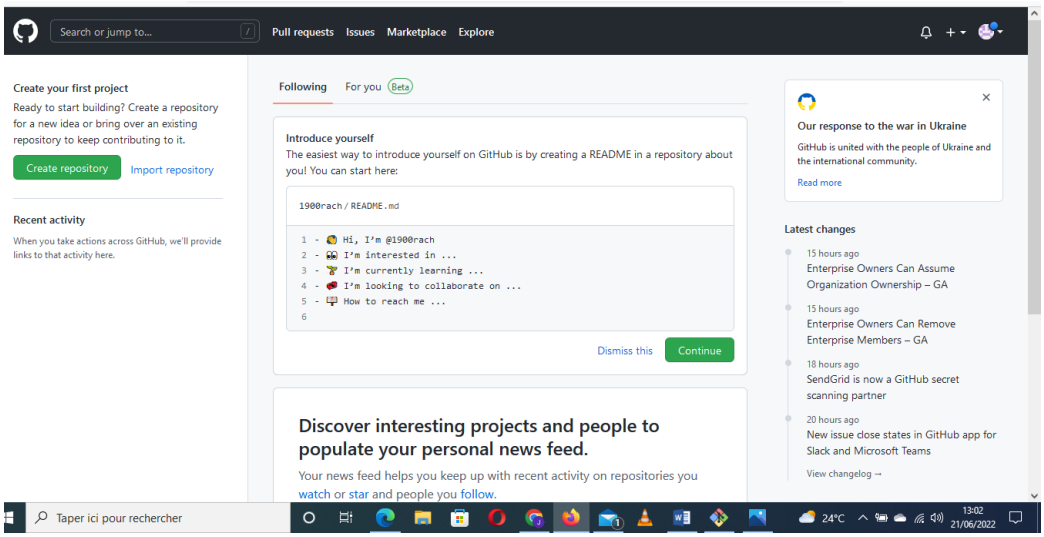
Vous pouvez cliquer sur le premier lien cependant elle vous envoie à cette nouvelle page



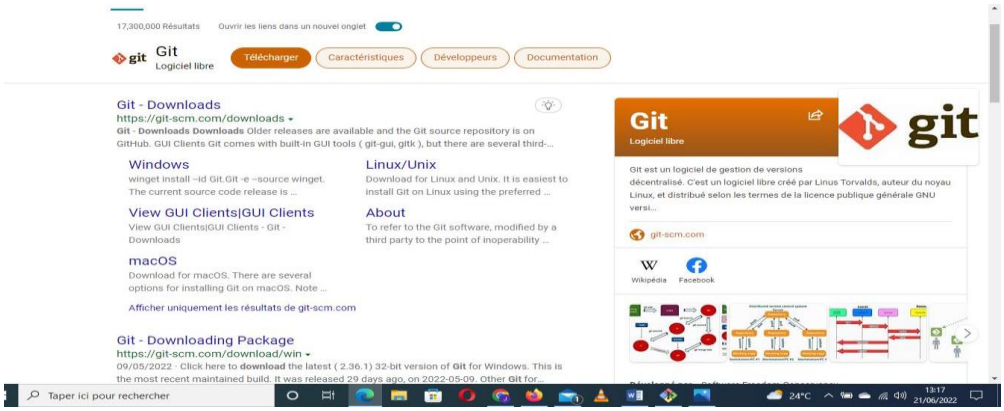
Ici on vous propose de vous enregistrer ou de vous connecter si vous voulez vous connecter clique sur le bouton signe in dans notre cas nous voulons nous enregistrer donc nous cliquons sur sign up on vous demande d'entre votre adresse email après votre mot de passe ensuite vous recevez un message sur votre Gmail



Pour signaler votre inscription dès lors vous êtes inscrit, vous tomber sur cette page bravo vous avez réussi on peut passez à l'étape suivant

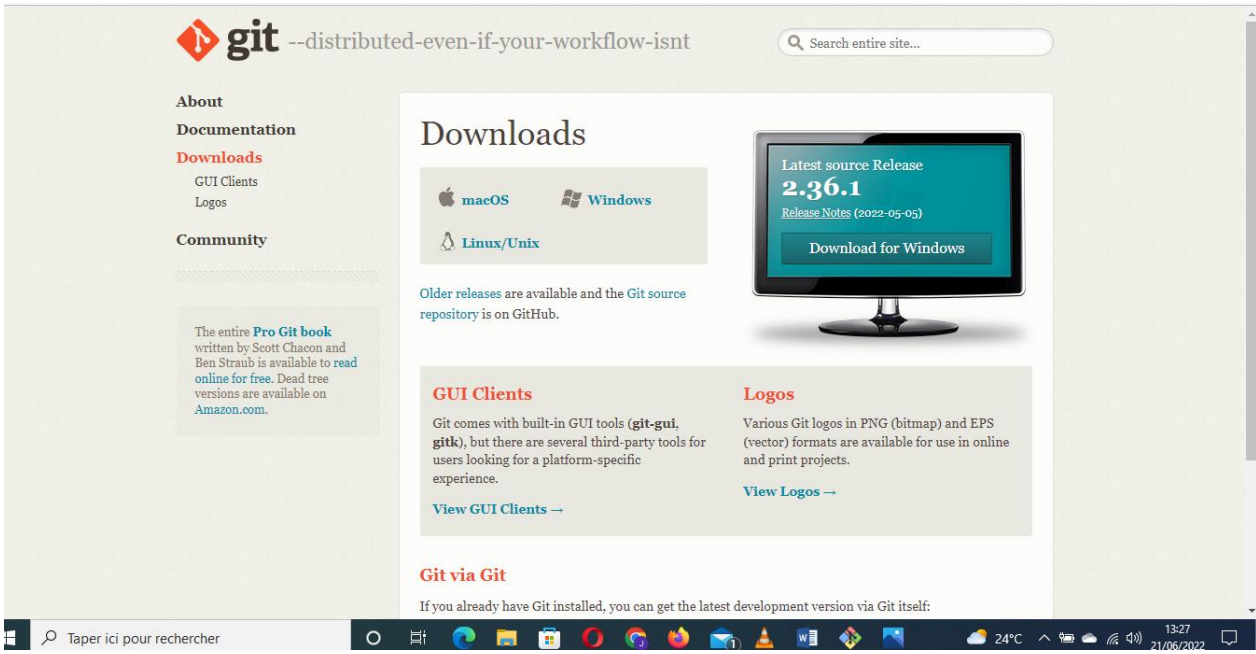


Maintenant vous pouvez passer à l'étape suivante qui le est téléchargement notre invite de commande GIT maintenant je vous invite a taper sur la barre de recherche git download alors vous arriver à cette page

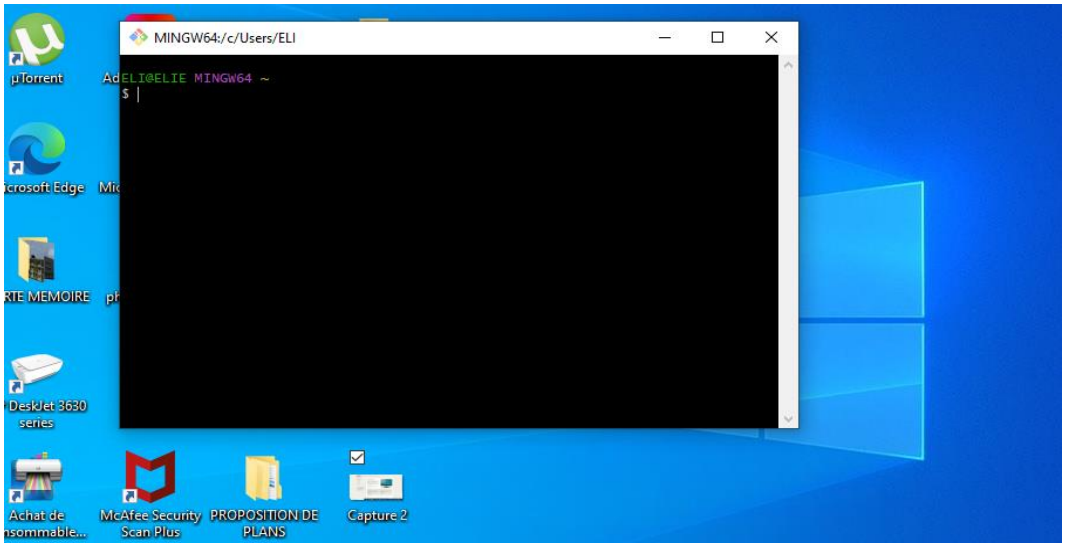


Alors vous pouvez cliquer sur

- downloading package pour choisir sur quel système d'exploitation tu aimerais télécharger



- choisir downloads alors il vous présente le différent système d'exploitation alors tu peux choisir ton système et directement le téléchargement est lancer. après l'installation vous tomber sur une invite de commande comme tel



C'est sur cette espace que nous allons travailler nous aurons besoin de connaitre les différentes commande de GIT

VI. LES DIFFERENTES COMMANDE UTILISER PAR GITHUB

- Git config

L'une des commandes git les plus utilisées est **git config.** On l'utilise pour configurer les préférences de l'utilisateur: son mail, l'algorithme utilisé pour diff, le nom d'utilisateur et le format de fishier etc. Par example, la commande suivante peut être utilisée pour définir le mail d'un utilisateur:

Git config --global user.email sam@google.com

- Git init

- Cette commande est utilisée pour créer un nouveau dépôt **GIT** :

  git init

Git add

- La **commande git add** peut être utilisée pour ajouter des fichiers à l'index. Par exemple, la commande suivante ajoutera un fichier nommé temp.txt dans le répertoire local de l'index:

  git add temp.txt

- Clone git

- La **commande git clone** est utilisée pour la verification des dépôts. Si le dépôt se trouve sur un serveur distant, utilisez:

  git clone alex@93.188.160.58:/chemin/vers/dépôt

- Inversement, si une copie de travail d'un dépôt local doit être créée, utilisez:

  git clone /chemin/vers/dépôt

- Git commit

- La **commande git commit** permet de **valider les modifications apportées** au HEAD. Notez que tout commit ne se fera pas dans le dépôt distant.

  git commit –m “Description du commit”

➢ Git status

- La **commande git status** affiche la liste des fichiers modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés. Usage:

  git status

➢ Git push

- **Git push** est une autre commandes **GIT** de base. Un simple push envoie les modifications locales apportées à la branche principale associée :

  git push origin master

➢ Git checkout

- La **commande git checkout** peut être utilisée pour créer des branches ou pour basculer entre elles. Par exemple nous allons créer une branche:

  command git checkout -b <nom-branche>

- Pour passer simplement d’une branche à une autre, utilisez:

  git checkout <nom-branche>

➢ Git remote

- Cette commande **remote** permet à un utilisateur de se connecter à un dépôt distant. La commande suivante répertorie les dépôts distants actuellement configurés:

  git remote –v

- Cette commande permet à l’utilisateur de connecter le dépôt local à un serveur distant:

git remote add origin <93.188.160.58>

➢ Branche git

- La **commande git branch** peut être utilisée pour répertorier, créer ou supprimer des branches. Pour répertorier toutes les branches présentes dans le dépôt, utilisez:

  git branch

- Pour supprimer une branche:

  git branch –d <nom-branche>

➢ Git pull

- Pour fusionner toutes les modifications présentes sur le dépôt distant dans le répertoire de travail local, la commande pull est utilisée. Usage:

  git pull

➢ Git merge

- La **commande git merge** est utilisée pour fusionner une branche dans la branche active. Usage:

  git merge <nom-branche>

➢ Git diff

- La **commande git diff** permet de lister les conflits. Pour visualiser les conflits d’un fichier, utilisez

  git diff --base <nom-fichier>

- La commande suivante est utilisée pour afficher les conflits entre les branches à fusionner avant de les fusionner:

  git diff <branche-source> <branche-cible>

- Pour simplement énumérer tous les conflits actuels, utilisez:

  git diff

- Git tag

- Le marquage est utilisé pour marquer des commits spécifiques avec des poignées simples. Un exemple peut être:

  git tag 1.1.0 <insert-commitID-here>

- Git log

- L' **exécution de** cette commande génère le log d'une branche. Un exemple de sortie :

- commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw
- Author: Alex Hunter <alexh@gmail.com>
  Date: Mon Oct 1 12:56:29 2016 -0600

- Git reset

- Pour réinitialiser l'index et le répertoire de travail à l'état du dernier commit, la **commande git reset** est utilisée :

  git reset --hard HEAD

- Git rm

- **Git rm** peut être utilisé pour supprimer des fichiers de l'index et du répertoire de travail. Usage:

  git rm nomfichier.txt

- Git stash

- L'une des moins connues, **git stash** aide à enregistrer les changements qui ne doivent pas être commit immédiatement. C'est un commit temporaire. Usage:

  git stash

- Git show

- Pour afficher des information sur tout fichier git, utilisez la **commande git show** . Par exemple:

  git show

Git fetch

- **Git fetch** permet à un utilisateur d'extraire tous les fichiers du dépôt distant qui ne sont pas actuellement dans le répertoire de travail local. Exemple d'utilisation:

```
Git fetch origin
```

Git ls-tree

- Pour afficher un fichier arborescent avec le nom et le mode de chaque élément, et la valeur SHA-1 du blob, utilisez la **commande git ls-tree** . Par exemple:

```
git ls-tree HEAD
```

Git cat-file

- À l'aide de la valeur SHA-1, affichez le type d'un fichier à l'aide de la **commande git cat-file** . Par exemple:

```
git cat-file –p d670460b4b4aece5915caf5c68d12f560a9fe3e4
```

Git grep

- **Git grep** permet à un **utilisateur de rechercher** dans les arbres de contenu des expressions et / ou des mots. Par exemple, pour *rechercher www.hostinger.com* dans tous les fichiers, utilisez:

```
git grep "www.hostinger.com"
```

Gitk

- **Gitk** est l'interface graphique du dépôt local. Vous pouvez l'appeler en exécutant:

```
gitk
```

Git instaweb

- Avec la **commande git instaweb**, un serveur Web peut être exécuté par interface avec le dépôt local. Qui redirige directement vers un serveur web. Par exemple:

git instaweb –httpd=webrick

➢ Git gc

- Pour optimiser le dépôt en supprimant les fichiers inutiles et les optimiser, utilisez:

  git gc

➢ git archive

- La **commande git archive** permet à un utilisateur de créer un fichier zip ou tar contenant les composants d'un arbre du dépôt. Par exemple:

  git archive --format=tar master

➢ Git prune

- Via la **commande git prune**, les fichiers qui n'ont pas de pointeurs entrants seront supprimés. Usage:

  Git prune

➢ Git fsck

- Pour effectuer une verification d'intégrité du système de fichiers git, utilisez la commande **git fsck** . Tous les fichiers corrompus seront identifiés: git fsck

➢ Git rebase

- La **commande git rebase** est utilisée pour la reapplication des commits sur une autre branche. Par exemple:

  git rebase master

VII. Comment travailler sur GitHub

Vous pouvez utiliser des dépôts GitHub pour stocker n'importe quel projet logiciel. Mais pour utiliser les pages GitHub et publier un site web (ce qui nous intéresse ici), votre projet devra être structuré comme un site web classique et notamment avec un fichier d'entrée intitulé index.html.

Il faut aussi que le répertoire où le code est stocké soit un « dépôt » Git sur votre ordinateur. Autrement dit, on indique qu'on utilise Git pour gérer les différentes versions des fichiers qui seront stockés dans ce dossier. Pour initialiser un dépôt Git, on suivra ces étapes :

1. Utilisez la ligne de commande pour vous placer dans le répertoire de votre site web (dans cet article, ce répertoire sera appelé test-site, remplacez ce nom avec celui de votre répertoire). Pour ce faire, on utilisera la commande cd (qui signifie « *change directory* » ou « changer de répertoire/dossier » en français). Par exemple, si votre répertoire se situe sur votre bureau et se nomme test-site, vous pourrez taper:

   cd Bureau/test-site

1. Lorsque vous êtes dans le répertoire de votre site web, utilisez la commande suivante. Celle-ci indiquera à Git que le répertoire doit être considéré comme un dépôt Git :
2. git init

Un aparté sur les interfaces en ligne de commande

La meilleure façon d'envoyer votre code sur GitHub est d'utiliser l'interface en ligne de commande de votre ordinateur. La ligne de commande est une fenêtre où on saisit des commandes au clavier pour créer des fichiers, lancer des programmes, plutôt que de cliquer avec la souris en utilisant une interface graphique.

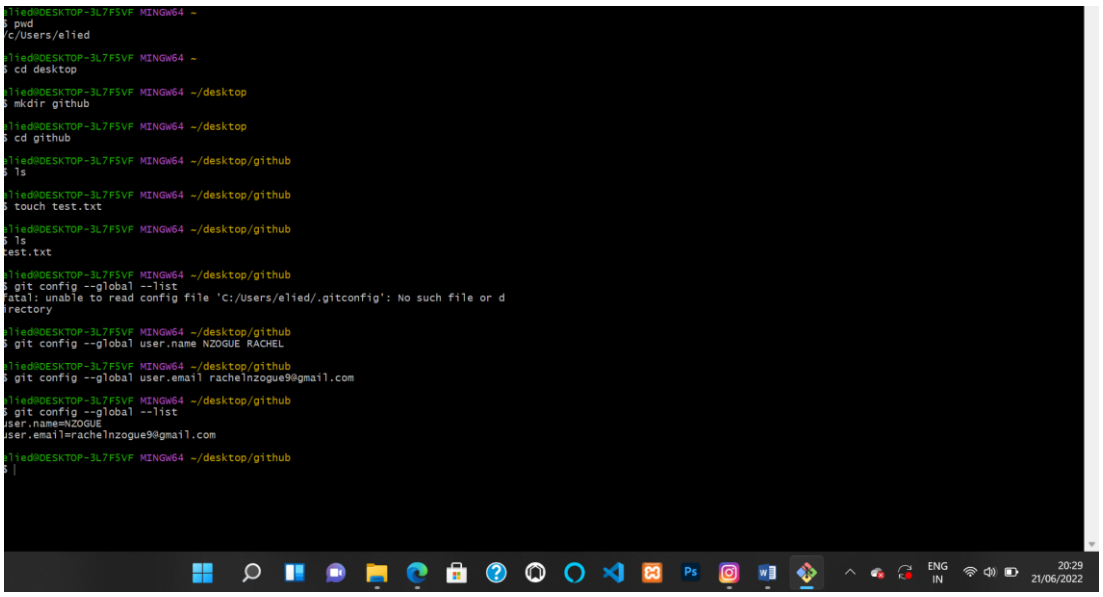
- **COMMENT CONFIGURER GIT**

  Pour un debut c’est vraiment bien de configure git pour cela vous ouvre l’invite de commande et vout taper

  - Pwd qui vas vous mettre sur votre ordinateur après vous taper
  - Cd desktop pour te mettre sur le bureau
  - Mkdir nom du dossier pour crée un dossier

- Cd le nom du dossier créer
- Touch nom du fichier pour créer un fichier dans le dossier
- Taper git config - -global user.name (votre nom) taper la touche entre
- Taper git config - -global user.email (votre email)) taper la touche entre
- Taper git config - -global - -list taper la touche entre

  Vous aurez une page comme ceci



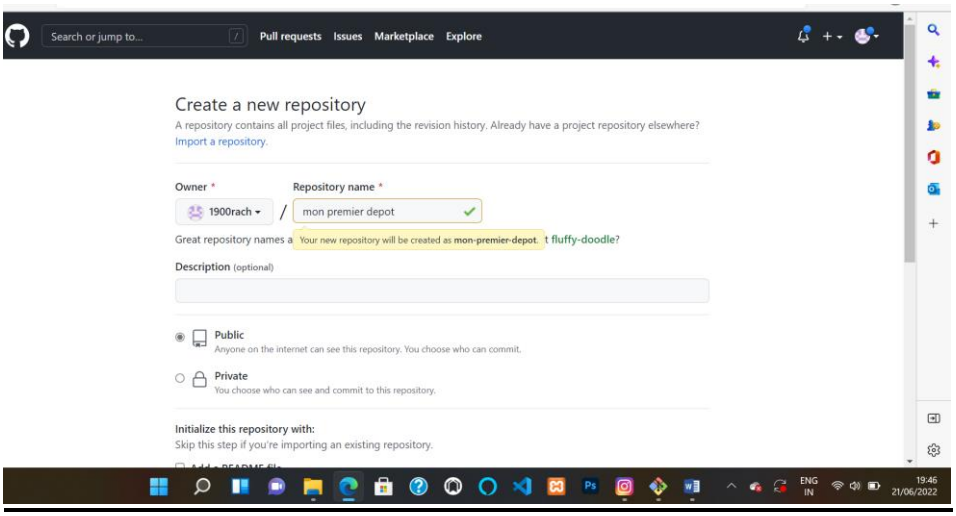
La votre git est configure

❖ COMMENT CREE UN DEPOT EN LIGNE POUR VOTRE CODE

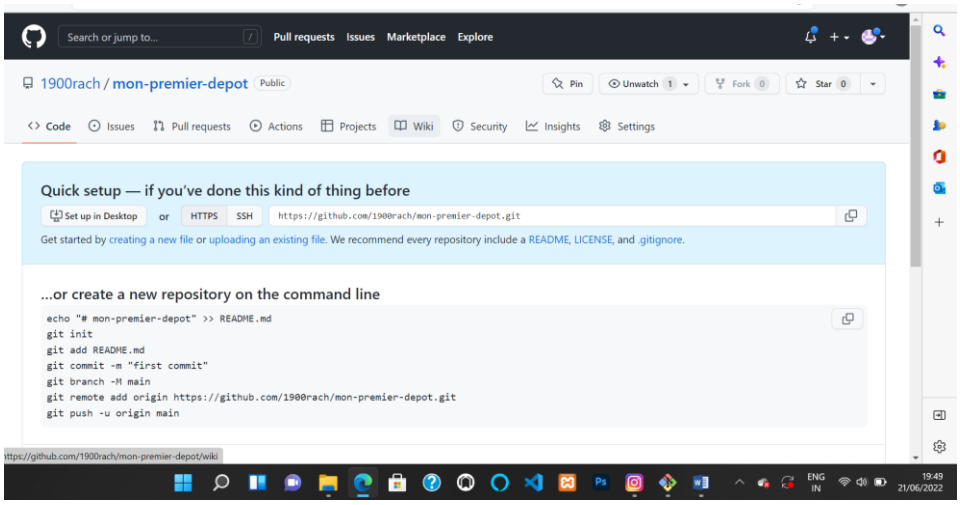
- Ensuite, vous aurez besoin de créer un dépôt GitHub sur lequel envoyer les fichiers de votre site. Quand vous êtes connecté-e sur GitHub, cliquez sur l'icône Plus (+) en haut à droite de la page d'accueil puis sélectionner l'option New Repository (qui signifie « Créer un nouveau dépôt »).

❖ Sur la page qui s'affiche, dans le champ « Repository name », entrez un nom pour votre dépôt. Vous pouvez par exemple saisir mon-premier-depot.

❖ Il y a également un champ qui décrit le projet qui sera placé dans ce dépôt. Votre écran devrait ressembler à quelque chose comme :



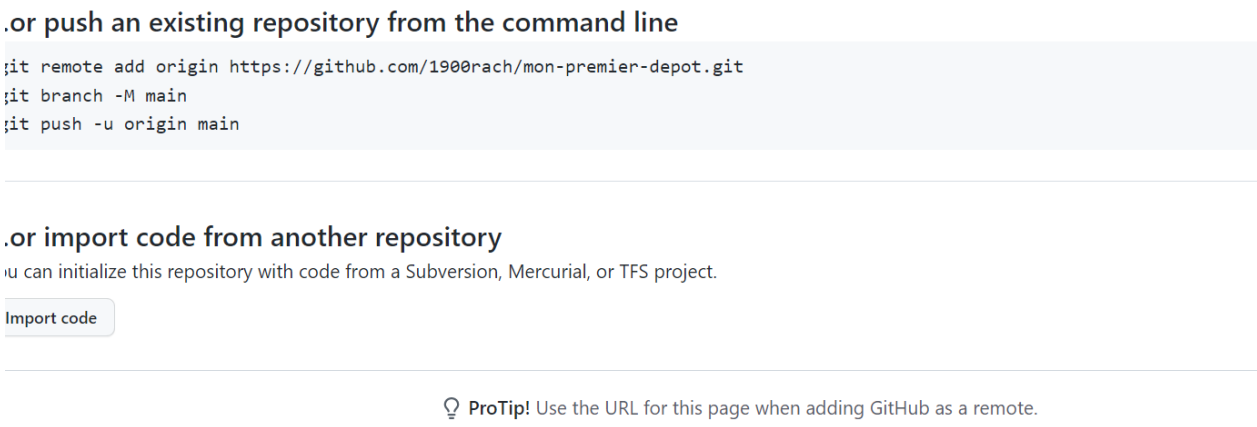
Ensuite, cliquez sur « *Create repository* » (pour créer le dépôt). Vous arrieverez alors sur la page suivante :



- Comment envoyer les fichiers sur github

Sur cette page, une section vous intéresse particulièrement: « *…or push an existing repository from the command line* » (ce qui signifie « ou pousser un dépôt existant grâce à la ligne de commande »). Vous devrez voir deux lignes de codes sous cette section. Copiez la première ligne et collez la dans votre interface

en ligne de commande puis tapez sur Entrée. La commande devrait ressembler à:



git remote add origin https://github.com/chrisdavidmills/mon-premier-depot.git

Ensuite, saisissez ces deux commandes en tapant sur la touche Entrée après chacune. Ces commandes permettent d'indiquer à Git qu'il doit gérer tous les fichiers du dossier et d'enregistrer cette action.

- git add --all
- git commit -m 'ajout des fichiers au dépôt'Enfin, envoyez le code sur GitHub en utilisant la seconde commande affichée sur la section de la page GitHub :
- git push -u origin master

Votre code est publié sur GitHub. Pour avoir une page GitHub, vous devrez créer une *branche* gh-pages sur votre dépôt. Actualisez la page web de votre dépôt, vous devriez obtenir une page semblable à celle présentée ci-dessous. Ensuite, cliquez que le bouton « Branch: **master** » (GitHub indique que vous êtes sur la branche master de votre dépôt). Dans la liste qui s'affiche, saisissez le texte **gh-pages** puis cliquez sur *Create branch: gh-pages* (« créer la branche intitulée gh-pages »). Cela créera

une nouvelle branche pour votre dépôt, cette branche s'appellera gh-pages et sera publiée à un endroit spécifique. L'URL du site sera *nom-utilisateur.github.io/nom-du-depot*. Dans mon exemple, l'URL est donc *https://chrisdavidmills.github.io/my-repository*

Si vous souhaitez modifier votre site et le mettre à jour sur GitHub, modifiez les fichiers comme vous le faisiez auparavant puis utilisez les commandes suivantes pour indiquer les changements à Git et les envoyer sur GitHub (n'oubliez pas d'appuyer sur Entrée entre chaque commande) :

```
git add --all
git commit -m 'Un autre commit'
git push
```

CONGLUSION

Parvenue au terme de notre crépuscule de nos différentes investigations donc il était question de connaitre le logiciel GitHub dans ses profondeurs nous retenons que ce logiciel est très indispensable pour le bon déroulement des différents projets ainsi ils sont pour tous les domaines et s'applique à tout ce pendant il a plusieurs conçurent mais reste le meilleurs le meilleur.