Git & GitHub avec Visual Studio Code

[1 Quelques raccourcis 2](#_Toc199862479)

[2 Mise en place Git 2](#_Toc199862480)

[3 Création d’un projet sans versionning 3](#_Toc199862481)

[3-1 En local 3](#_Toc199862482)

[3-2 Lier à GitHub 3](#_Toc199862483)

[3-2-1 si erreur a la création alors faire les commandes : 4](#_Toc199862484)

[3-3 Pour récupérer les changements : 4](#_Toc199862485)

[4 Création d’un projet avec versionning 4](#_Toc199862486)

[4-1 En local 4](#_Toc199862487)

[4-2 Créer des branches (donc des versions) 6](#_Toc199862488)

[4-3 Lier à GitHub 7](#_Toc199862489)

[5 Petit recap et autre 7](#_Toc199862490)

Une documentation très complète de Git se trouve ici :

<https://git-scm.com/book/fr/v2/>

Git :

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

Extension GitHub GitHub Pull Requests

<https://marketplace.visualstudio.com/items?itemName=GitHub.vscode-pull-request-github>

Vérifier version de Git avec la commande dans Git Bash (ou un terminal) :

git –version

# 1 Quelques raccourcis

Dans VS Code :

* Ouvrir un terminal

CTRL + J

* Ouvrir direct éditeur

CTRL + Shift + P

* Menu Git

Ctrl + Shift + G

* Commit rapide

Ctrl + Shift + P puis "Git Commit"

* Rechercher fichier

Ctrl + P

# 2 Mise en place Git

Commandes avec Git Bash ou terminal

git config --global user.name "ton nom"

git config --global user.email "ton-email@example.com"

git config –list

Faire en sorte que tous les nouveaux dépôts soient en main au lieu de master :

git config --global init.defaultBranch main

# 3 Création d’un projet sans versionning

## 3-1 En local

* Un dossier pour un projet

Structure classique :

* + css
  + js
  + img
  + font
  + index.html

A la racine du projet un fichier .gitignore sans extension de de nom. Ce fichier permettra d’ignorer des éléments.

A partir de là soit commande dans le terminal de VS Code ou click droit dans le dossier du projet ouvrir git bash.

* Créer un nouveau repo

git init

* Vérifier la branche

git branch

* Si master renommer en main

git branch -m main

* Vérifier si ok

git branch

* Vérifie les changements

git status

* Ajouter tous les fichiers

git add .

* Commit

git commit -m « ajout d’une fonctionnalité être bien précis »

## 3-2 Lier à GitHub

* Créer un repo dans GitHub (idéalement avec le même nom que le local)
* Récupérer le lien du repo dans code
* Faire les commandes suivantes

git remote add origin https://github.com/utilisateur/mon-projet.git

git branch -M main

git push -u origin main

git status (pour vérifier)

### 3-2-1 si erreur a la création alors faire les commandes :

git pull --rebase origin main

git push origin main

git push --force origin main

git status (pour vérifier)

## 3-3 Pour récupérer les changements :

git pull origin main

# 4 Création d’un projet avec versionning

## 4-1 En local

* Créer un nouveau repo

git init

* Vérifier la branche

git branch

* Si master renommer en main

git branch -m main

* Vérifier si ok

git branch

* Vérifie les changements

git status

* Ajouter les fichiers à référencer

git add nom du fichier (ou chemin du dossier)

* Vérifie les changements

git status

* Commit

git commit -m « ajout d’une fonctionnalité être bien précis »

* Pour voir le commit

git log

A chaque étape faire les manips de add et de commit tout en faisant des status et des log pour vérifier.

Si erreur sur le nom d’un commit :

git commit –amend -m  « changer le nom du dernier commit »

Pour éviter de référencer des fichiers inutiles un fichier .gitinore (sans extension) à la racine du projet. Avec vsc mettre les fichiers ou dossiers à ignore. Par exemple le dossier img :

Img/

Sauvegarder puis faire un commit.

git commit -m  « Fichier .gitignore mis à jour »

git log

Supprimer le dernier commit

git reset –soft HEAD ~1

Modifier le deuxième commit (par exemple)

git rebase -i HEAD ~2

Aller dans la ligne concernée la modifier puis

Changer

pick

Par

reword

Pour quitter l’éditeur ESC et :wq

Faire un git log pour vérifier si ok.

Si vous modifier un élément et que vous avez committez pour voir la différence entre les 2 versions :

git diff

cela peut etre fait sur un fichier spécifique en ajoutant après diff nom + extension ou chemin.

En faisant :

git commit -am « on cumule commit et add donc gain de temps »

## 4-2 Créer des branches (donc des versions)

On crée une branche par exemple test

git branch test

On bascule vers cette branche

git checkout test

A partir de maintenant tous les commits se feront sur la branche test, la branche main est figée.

Si on fait :

git checkout master

on revient au code figé et dans l’aperçu et dans vsc

et vice versa.

Si la partie test est ok on va fusionner avec la branche main

On se met sur :

git checkout master

Puis

git merge test

On supprime la branche test

git branch -d test

On vérifie

git branch

## 4-3 Lier à GitHub

* Créer un repo dans GitHub (idéalement avec le même nom que le local)
* Récupérer le lien du repo dans code
* Faire les commandes suivantes

git remote add origin https://github.com/utilisateur/mon-projet.git

git branch -M main

git push -u origin main

git status (pour vérifier)

# 5 Petit recap et autre

