**Initiation Repository**

INITIATION GIT :

*Commandes Git pour :*

1- INITIALISER GIT,

2- CREER UN PROJET SOUS GIT,

3- SAUVEGARDER LE STATUT DE VOTRE PROJET GRACE A LA NOTION DE COMMIT,

4- APPORTER DE NOUVELLES FONCTIONNALITES SEREINEMENT AVEC LES BRANCHES,

5- LA CREATION DE DEPOTS DISTANT,

6- L’UTILISATION DE GITHUB POUR UN TRAVAIL COLLABORATIF.

1- INITIALISER GIT dans le projet :

Initialiser GIT pour pouvoir l'utiliser

Taper **git init**

Ce qui donne

*Initialized empty Git repository in C:/Users/clair/landingpage/.git/*

On pourra par la suite demander un status global pour savoir où on en est.

Taper **git status**

Ce qui donne

*On branch master*

*No commits yet*

*nothing to commit (create/copy files and use "git add" to track)*

Il n'y a rien à commiter, c'est normal le fichier est vide.

-----------------------------------------------------------------------------------

2 - Création d'un projet :

Utiliser un éditeur html pour créer un fichier html. (On utilisera VSC)

On créera un squelette de page. On le nommera

**Lessons-GIT-24-07-02**

-----------------------------------------------------------------------------------

3 - Sauvegarder le statut du projet :

On voudra le commiter, c'est à dire, dire à GIT de prendre une photo de tous nos fichiers. Elle se fait toujours en deux étapes.

*- La sélection des fichiers*

Ce qu'on veut enregistrer. Dans l'exemple utilisé il n'y en a qu’un.

Taper **git add nomdufichier**

Ce qui donne à taper

**C:\Users\clair\landingpage>git add Lessons-GIT-24-07-02**

*- La prise du ou des fichiers sélectionnés*

On va pouvoir faire la photo des fichiers sélectionnés.

Taper **git commit -m "first commentaire"**

Ecrire à la suite de -m un petit commentaire explicatif du pourquoi le commiter

Taper **git commit -m "ajout-fichier-Lessons-GIT-24-07-02"**

Ce qui donne

*[master 4997ea1] ajout-fichier-Lessons-GIT-24-07-02*

*1 file changed, 20 insertions(+)*

*create mode 100644 index.html*

On a désormais 2 photos de notre projet prise pour GIT.

On a cette "timeline" qui va nous permettre de voir l'historique de toutes les modifications par ordre chronologique.

Taper **git log**

Ce qui donne l’historique des commits.

On peut voir les modifications (3) avec le nom de la personne qui a fait les

modifications, son email, la date, et l'explication. Tout en dessous on trouve le

premier commit qu'on a réalisé. C'est donc du plus récent au plus ancien.

4 - La notion de branches :

Les branches s'apparentent à une copie de ton projet initial.

On pourra par la suite, modifier ou supprimer sans impact sur le projet initial vers lequel on pourra rapatrier les modifications qui plaisent.

Taper **C:\Users\clair\landingpage>git branch buttonColor**

Le nom donné à la branche doit faire le plus possible faire référence à la fonctionnalité apportée pour s'y retrouver, car on peut en créer une centaine si on le souhaite.

La branche master est la branche initiale, c'est celle qu'on va toujours avoir.

Pour lister les branches

Taper **git branch**

Ce qui donnera toutes les branches les unes à la suite des autres. La liste des branches apparaît.

*buttonColor*

*\* master*

La branche master est d'une autre couleur avec un petit astérix, ce qui signifie qu'on se trouve sur la branche master.

*Si on souhaite changer de branche.*

Taper **git checkout nomdelabranche**

Les commits s'appliqueront uniquement sur la branche active.

Si on veut modifier sur une branche en particulier, il faudra aller dessus.

taper **git checkout buttonColor**

Ce qui donnera

*C:\Users\clair\landingpage>git checkout buttonColor*

*Switched to branch 'buttonColor'*

*D Lessons-GIT-24-07-02.html*

On va pouvoir travailler sur le fichier sereinement et apporter les modifications voulues.

Exemple, on veut modifier la couleur de fond du body :

*- on sélectionne toujours le fichier qu'on va vouloir sélectionner*

taper **git add index.html**

*- puis on prend le fichier qu'on commit avec un petit message pour donner le pourquoi du comment*

taper **git commit -m "addBackgroundColor"**

*- On vérifie si tout s'est bien passé*

taper **git log**

On veut rapatrier les modifications sur la branche principale, pour version définitive du projet.

Comment faire ? 3 étapes à faire.

*- basculer sur la branche principale, la branche master.*

taper **git checkout master**

On peut vérifier le changement de branches

taper **git branch**

*- Demander à GIT de récupérer, de rapatrier le travail de la branch buttonColor pour la fusionner avec la branche master.*

taper **git merge nom delabrancheafusionner**

On vérifie si la fusion a bien été effective

taper **git log**

On doit voir le commit buttonColor sur notre branch master

*- Supprimer cette branche quand le travail dessus est terminé*

taper **git branch -d buttonColor**

(Fortement conseillé de supprimer les branches qui n'ont plus de sens avec l'intitulé qu'on a donné)

On vérifie si la branche a bien été supprimée

taper **git branch**

5- la création de dépôts distants

On va pouvoir déposer tout le projet sur un endroit accessible par d'autres, et que ces personnes puissent récupérer le projet.

Dans le jargon on l'appelle "dépôt distant".

L'environnement sur lequel on vient de travailler s'appelle "dépôt local".

Avec le dépôt distant, l'équipe va pouvoir également pouvoir modifier ce travail et le diffuser à d'autres et nous-mêmes.

On va pouvoir le faire au travers d'un service que l'on appelle GitHub.

C'est une plateforme web qui va nous proposer la création de manière assez simple de ce dépôt-là.

Propose une interface web assez convivial qui permettra de naviguer dans le code, le projet.

*- Créer un compte,*

*- Créer un nouveau repository en lui donnant un nom,*

*- Il faut créer un pont entre notre dépôt distant et local, pour qu'on puisse envoyer notre projet.*

taper **git remote add origin urldepotcréépargithub**

**git remote**

On peut vérifier les dépôts distants du projet.

On obtient le dépôt distant : **origin**

Il ne reste plus qu'à déposer le projet

Taper **git push origin master**

Fait référence à notre branche principale.

Git se charge d'envoyer en automatique le projet

On vérifie si notre projet est bien présent dans le dépôt GitHub.

**https://github.com/Poca23/landingpage OK**

Pour que les collaborateurs puissent récupérer le dépôt distant sur GitHub :

*- Se brancher sur le dépôt distant*

taper **git remote add urldudepotdistantquivientdetrecreer**

*- Pour que les collaborateurs récupèrent le projet*

taper **git pull origin master**

Va rapatrier tout le projet du dépôt distant vers le dépôt local qu'ils ont créés.

Ces personnes doivent être inscrite sur GitHub en ayant un compte, c'est à nous

de leurs donner un accès en écriture au repository distant que l'on vient de

créer.

*- Settings > Collaborators > noms des personnes qui auront l'accès au dépôt distant créé sur GitHub.*

*-Pour qu'ils puissent rapatrier les modifications sur notre dépôt disant.*

**taper git push master**

*- L'ensemble de l'équipe peut récupérer ensuite l'ensemble des modifications*

**taper git pull master.**

On a créé un worflow parfait pour une collaboration.