**Commandes Git Pour :**

* **Création d’un dépôt GitHub local et distant**
* **Mise en place d’un accès pour une Collaboration**

|  |  |  |
| --- | --- | --- |
| **Création d’un dépôt local** | | |
| **Étapes** | **Commandes** | **Fonctions** |
| **0** | * Windows : Barre de recherche/taper « **terminal** » * Mac : Barre de recherche/taper « **Cmd** » | Pour ouvrir le **terminal.** |
| **1** | **git --version** | Vérifier si Git est **correctement installé.** |
| **2** | * **git config --global user.name "Naudin Claire"** * **git config --global user.email "cndweb37@gmail.com "** | **Paramétrage** à faire auprès de git, définir son identité. |
| **3** | **mkdir « nomdurépertoire »** | **Créer un répertoire** directement en ligne de commande : (répertoire = projet). |
| **4** | **cd « nomdurépertoire »** | **Rentrer** dans le répertoire. |
|  | **cd ../ « nomdurépertoire »** | **Revenir** au répertoire supérieur. |
| **5** | **Git init / git status /** créer un fichier ------ | **Initialiser** le projet / Demande un status. |
| **6** | * Ajouter les fichiers souhaités, * **git add « nomdufichier »** | **Sélectionner** des fichiers ajoutés. |
| **7** | **git commit -m "commentaire"** | Faire la **photo des fichiers sélectionnés**, petit commentaire explicatif du pourquoi le commiter ensuite. |
| **8** | **Git push** | **Exporte les commits vers les branches distantes** s’il est créé (git fetch : importe les commits dans les branches locales). |
| **9** | **git log** | **Voir l'historique** de toutes les modifications par ordre chronologique, C'est donc du plus récent au plus ancien. |
| **On a désormais 2 photos de notre projet prise pour GIT.** | | |

|  |  |  |
| --- | --- | --- |
| **Création d’un repository distant et accessible**  *(Tout collaborateur doit créer un compte GitHub)* | | |
| **Étapes** | **Commandes** | **Fonctions** |
| **10** | **echo "***nom du dossier créé* **" README.md** | Créer un fichier **Readme** pour **expliquer le projet** du repository qui sera créé. |
| **11** | **git init** | On crée un **nouveau dépôt Git** (.git) pour la création d’un projet. |
| **12** | **git status** | On affiche le **statut du répertoire** de travail et de la **zone de préparation**. |
| **13** | **git add --all** | On **ajoute** tous les **fichiers** ou les **modifications** au référentiel. |
| **14** | **git status** | On **vérifie** le **statut** et la **validation** des modifications. |
| **15** | **git commit -m "first commit"** | On **capture** l’état du **projet** à l’instant t en **commitant**. |
| **16** | **git log** | On **liste les commits** effectués dans ce dépôt dans **l'ordre chronologique inverse** (du plus récent au plus ancien). |
| **17** | **git branch -M master** | On **sélectionne** **la branche master** principale pour **déposer** notre projet. |
| **18** | **git remote add origin «url du dépôt distant »** | crée un nouvel **enregistrement de connexion à un référentiel distant**. |
| **19** | **git push -u origin master** | le terme **origin master** est utilisé pour **gérer un dépôt distant et une branche master**. Le terme **push origin master** est utilisé pour **pousser les modifications vers le dépôt distant.** |
| **On vérifie si notre projet est bien présent dans le dépôt GitHub :**  **Adresse URL du dépôt distant** | | |

|  |  |  |
| --- | --- | --- |
| **Push d’un repository distant déjà existant et accessible**  *(Tout collaborateur doit créer un compte GitHub)* | | |
| **Étapes** | **Commandes** | **Fonctions** |
| **20** | **git remote add origin «** *url du dépôt déjà existant***»** | **Créer un pont** entre notre **dépôt distant et local.** |
| **21** | **git branch -M master** | Se **mettre** sur la **branche master.** |
| **22** | **git push -u origin master** | **Déposer le projet**, 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 :**  **Adresse URL du dépôt distant** | | |

|  |  |  |
| --- | --- | --- |
| **Collaborations sur un projet distant** | | |
| **Étapes** | **Commandes** | **Fonctions** |
| **31** | **git pull origin master** | **Rapatrier** de notre dépôt **distant vers leurs dépôt local**  *(Ces personnes doivent être inscrite sur GitHub en ayant un compte)* |
| **32** | *Settings* > Collaborators > noms des personnes qui auront l'accès au  dépôt distant créé sur GitHub | **Donner un accès** en écriture au repository distant que l'on vient de créer. |
| **33** | **git push master** | **Rapatrier** **leurs** modifications **sur notre dépôt disant.** |
| **34** | **git pull master** | **L'ensemble de l'équipe** peut récupérer ensuite **l'ensemble des modifications.** |
| **On a créé un worflow parfait pour une collaboration.** | | |

|  |  |  |
| --- | --- | --- |
| **Les Branches** | | |
| **Étapes** | **Commandes** | **Fonctions** |
| **23** | **git branch « nomDeLaBranche »** | **Créer une branche**, faire référence à la fonctionnalité apportée pour son nom. |
| **La branche master est la branche initiale, c'est celle qu'on va toujours avoir.** | | |
| **24** | **git branch** | **Lister les branches**, Ce qui donnera toutes les branches les unes à la suite des autres. |
| **La branche master est d'une autre couleur avec un petit astérix, ce qui signifie qu'on se trouve sur la branche master** | | |
| **25** | **git checkout « nomDeLaBranche »** | **Changer de branche**, Les commits s'appliqueront uniquement sur la branche active. |
| **git checkout « nomDeLaBranche »** | **Modifier** sur une branche en particulier, il faudra aller dessus. |

|  |  |  |
| --- | --- | --- |
| **Rapatrier les modifications sur la branche principale** | | |
| **Étapes** | **Commandes** | **Fonctions** |
| **26** | **git checkout master** | **Basculer sur la branche principale**, la branche master. |
| **27** | **git branch** | **Vérifier** le changement de branches |
| **28** | **git merge « nom delabrancheafusionner »** | **Récupérer pour rapatrier** la branche travaillée pour la **fusionner avec la branche master**. |
| **On doit voir le commit écrit sur notre branch master** | | |
| **29** | **git branch -d « nomBrancheASupprimer »** | **Supprimer cette branche** quand le travail dessus est terminé. |
| **30** | **git branch** | **Vérifier** si la branche a bien été supprimée |