**[Gérez du code avec Git et GitHub](https://openclassrooms.com/fr/courses/7162856-gerez-du-code-avec-git-et-github)**

# 1ère partie

## ***Découvrez la magie du contrôle de versions***

* Un gestionnaire de versions permet aux développeurs de conserver un historique des modifications et des versions de tous leurs fichiers.
* Git est un gestionnaire de versions tandis que GitHub est un service en ligne qui héberge les dépôts Git. On parle alors de dépôt distant.

## ***Saisissez l'utilité des dépôts distants sur GitHub***

* Un dépôt est comme un dossier qui conserve un historique des versions et des modifications d’un projet. Il est essentiel pour travailler en équipe ou collaborer à un projet open source.
* Un dépôt local est l’endroit où l’on stocke, sur sa machine, une copie d’un projet, ses différentes versions et l’historique des modifications.
* Un dépôt distant est une version dématérialisée du dépôt local, que ce soit sur Internet ou sur un réseau. Il permet de centraliser le travail des développeurs dans un projet collectif.
* Il existe plusieurs services en ligne pour héberger un dépôt distant, GitHub étant l’un des plus populaires.

## ***Démarrez votre projet avec GitHub***

* Pour démarrer un projet, vous devez créer votre compte GitHub.
* Vous pouvez suivre vos différents projets facilement grâce au tableau de bord.
* Pour mettre votre projet sur GitHub, vous devez créer un repository.

## ***Installez Git sur votre ordinateur***

* Pour installer Git, vous devez télécharger et configurer Git sur votre ordinateur.
* Pour initialiser un dépôt Git, vous pouvez soit créer un dépôt local vide, soit cloner un dépôt distant.
* git init permet d’initialiser un projet Git.

# 2ème partie

## ***Travaillez depuis votre dépôt local Git***

* Le protocole SSH permet d’authentifier une personne grâce à une clé unique et personnelle.
* Vous pouvez ajouter une clé SSH sur votre compte GitHub afin de vous authentifier plus rapidement.
* git add permet d’ajouter des fichiers dans l’index, qui est une zone intermédiaire dans laquelle stocker les fichiers modifiés.
* git commit permet de créer une nouvelle version avec les fichiers situés dans l’index.
* git commit -m permet de créer une nouvelle version et de préciser le message rattaché au commit.
* git push permet d’envoyer les modifications faites en local vers un dépôt distant.

## ***Appréhendez le système de branches***

* Une branche est une “copie” d’un projet sur laquelle on opère des modifications de code.
* La branche main (ou anciennement master) est la branche principale d’un projet.
* git checkout permet de basculer d’une branche à une autre.
* git merge permet de fusionner deux branches.

## ***Travaillez avec un dépôt distant***

* Sur GitHub, nous pouvons récupérer l'URL d'un dépôt distant.
* git clone permet de copier en local un dépôt distant.
* git remote add permet de lier un dépôt à un "nom court", pour une plus grande facilité d’utilisation.
* git pull permet de dupliquer un dépôt GitHub en local.
* Une Pull Request permet de demander à fusionner votre code sur la branche principale.

# 3ème partie

## ***Pratiquez et corrigez vos erreurs sur un dépôt local***

* git branch -d permet de supprimer une branche.
* git status permet de voir l’état des fichiers.
* git stash enregistre les modifications non indexées pour une utilisation ultérieure.
* git log affiche l'historique des commits réalisés sur la branche courante.
* git reset --hard HEAD^ permet de réinitialiser l'index et le répertoire de travail à l'état du dernier commit.
* git commit --amend permet de sélectionner le dernier commit pour y effectuer des modifications.

## ***Corrigez vos erreurs sur votre dépôt distant***

* git revert HEAD^ permet d'annuler un commit en créant un nouveau commit.
* git reset est une commande puissante. Elle peut être appliquée de 3 façons différentes (--soft; --mixed; --hard).
* La commande git merge produit un conflit si une même ligne a été modifiée plusieurs fois. Dans ce cas, il faut indiquer à Git quelle ligne conserver.
* git reset permet de revenir à l'état précédent sans créer un nouveau commit.
* git revert permet de revenir à l'état précédent en créant un nouveau commit.

## ***Corrigez un commit raté***

* git log affiche l'historique des commits réalisés sur la branche courante.
* git reflog est identique à git log. Cette commande affiche également toutes les actions réalisées en local.
* git checkout un\_identifiant\_SHA-1 permet de revenir à une action donnée.
* git blame permet de savoir qui a réalisé telle modification dans un fichier, à quelle date, ligne par ligne.
* git cherry-pick un\_identifiant\_SHA-1 un\_autre\_identifiant\_SHA-1 permet de sélectionner un commit et de l'appliquer sur la branche actuelle.