**GIT/GITLAB**

Git permet de suivre les modifications de votre code et d'organiser vos projets de développement. C’est un outil essentiel, que vous travailliez **seul**, en **équipe**, ou même sur **un projet en open source** !

GitHub un site web qui stocke votre code

Git un outil qui permet de stocker plusieurs versions de votre code en utilisant le contrôle de version.

Un gestionnaire de versions est un programme qui permet aux développeurs de conserver un historique des modifications et des versions de tous leurs fichiers.

**Git est un gestionnaire de versions**. Vous l’utiliserez pour créer un dépôt local et gérer les versions de vos fichiers.

**GitHub est un service en ligne** qui va héberger votre dépôt. Dans ce cas, on parle de **dépôt distant** puisqu’il n’est pas stocké sur votre machine.

**Ainsi, les dépôts sont utiles si :**

**Vous souhaitez conserver un historique de votre projet ;**

**Vous travaillez à plusieurs ;**

**Vous souhaitez collaborer à des projets open source ;**

**Vous devez retrouver par qui a été faite chaque modification ;**

**Vous voulez savoir pourquoi chaque modification a eu lieu.**

**GitHub/ GitLab / Bitbucket**

**Cd : commande qui permet de se déplacer dans les dossiers**

**Mkdir : permet de créer un dossier**

**--global : signifie que tous les changements se feront pour tous les projets, si on ne le met pas cela signifie que les changements ne se font que pour le projet actuel**

Git config --global user.name “Zakaria” : permet de configurer le nom

Git config --global user.email “z.lezgham@gmail.cm” : permet de configurer l’email.

**On vient de configurer notre identité, on configure maintenant notre environnement**

Git config –global color.diff auto

Git config –global color.status auto

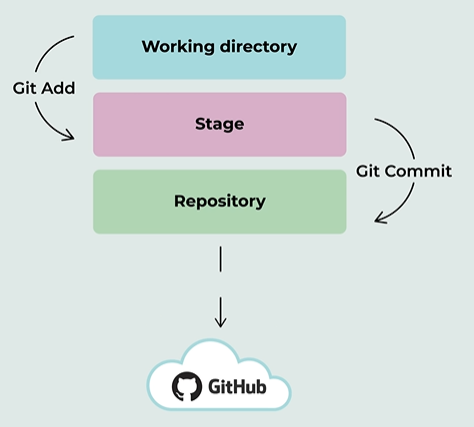
Git config –global color.branch auto: **permet de configurer les couleurs.**

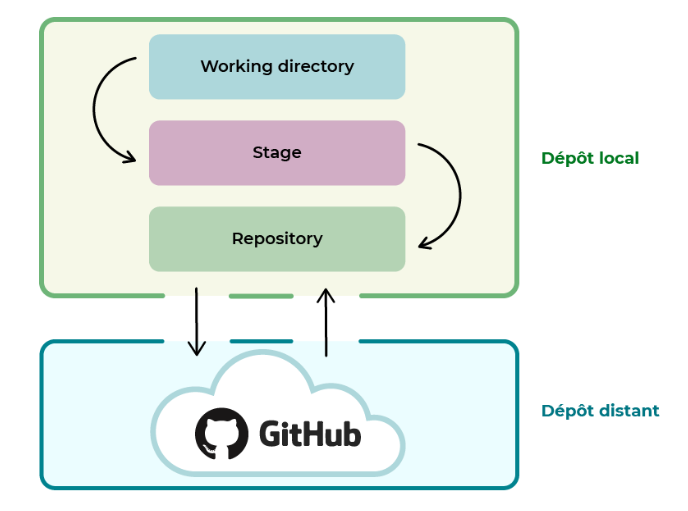
Git config –global core.editor vim (editeur de base de git)

Git config –global merge.tool vimdiff : **permet de configurer l’éditeur.**

Get init : permet d’initialiser notre projet

Git config –list : permet de voir tous les paramètres



  pwd is used to print the 'present working directory'

Touch : créer un fichier

**Touch index.html**

**git status** == ls ?

Une image contenant texte

Description générée automatiquement

En rouge car non indexé

Pour l’indexer **git add index.html**

Une image contenant texte

Description générée automatiquement

**git commit -m** “Ajout des fichiers html et css de base” : ON COMMIT LES NOUVELLES VERSIONS DU CODE

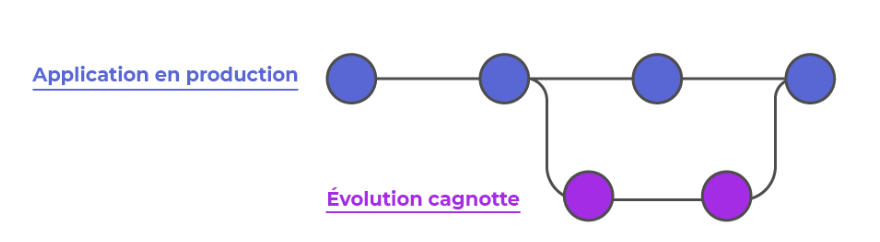
**git remote add origin** https://github.com/EtudiantOC/OpenclassroomsProject

.git

**Git branch -m main**

= lier le dépôt distant au dépôt local

**git push -u origin main** :envoyer des commits du repository vers le dépôt distant GitHub

Concept des branches

**Git branch** : permet de vous montrer tous vos branches

Une image contenant texte, intérieur, sombre

Description générée automatiquementle petit étoile signifie que c’est la branche sur lequel on est situé maintenant

**Git branch test** : créer la branche test

Une image contenant texte

Description générée automatiquement

**Git checkout cagnotte** permet de se situer sur la branche cagnotte

Une image contenant texte

Description générée automatiquement

Fusionner la branche cagnotte avec la branche main :

Get merge cagnotte (mais d’abord il faut basculer vers la branche main)

Une image contenant texte

Description générée automatiquement

**Git clone %url du projet** : cloner ou copier le dépôt distant sur notre machine.(en travail sur le terminal)

Une image contenant texte, écran, noir, capture d’écran

Description générée automatiquement

Ou sur git bash

Une image contenant texte

Description générée automatiquement

**Git remote add OC**: OC représente le nom court que vous utiliserez ensuite pour appeler votre dépôt.





Cette ligne ne permet pas de copier le dépôt, mais permet de dire au dépôt que l’on pointe vers le dépôt distant.

Il se clone dans le dossier



**git pull origin main**

**Une image contenant texte

Description générée automatiquement**

Dans un contexte professionnel, lorsque vous travaillez en équipe sur un repository, la branche principale est souvent bloquée. Vous ne pouvez pas pusher directement votre code sans qu’il soit vérifié. Vous ne pouvez donc pas fusionner vos modifications vous-même.

Pull request : une fonctionnalité de GitHub qui permet de demander aux propriétaires d’un repository l’autorisation de fusionner nos changements sur la branche principale ou toute autre branche sur laquelle on souhaite apporter nos modifications.

‘Change h1 color from red to blue’

**Corriger les erreurs sur le dépôt local :**

Grâce à la ligne de commande **ls -la**  , vous pouvez faire apparaître les dossiers cachés.

**Git branch -d test** : supprime la branche test

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

On a sauvegardé le fichier après modification mais avant de le commit. On ne souhaite pas le changer. On utilise git stash qui sauvegarde nos changements et au même temps enlève ce qu’on devait commit.

La remise, on crée une branche, et c’est là où on veut commit.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Git stash list : Cette commande va vous retourner un "tableau" des stash(des modifications) avec des identifiants.

Git log : analyser les derniers commits, vous allez alors récupérer l'identifiant du commit que l'on appelle couramment le hash.

Par défaut, git log va vous lister par ordre chronologique inversé tous vos commits réalisés.

Git reset --hard HEAD^ : Cette ligne de commande va supprimer de la branche principale votre dernier commit.  Le HEAD^ indique que c'est bien le dernier commit que nous voulons supprimer. L’historique sera changé, les fichiers seront supprimés.

Ce la supprime le dernier commit mais on garde l’identifiant, on peut créer une nouvelle branch, et faire ce commit la dans cette branch avec la commande :

Une image contenant texte

Description générée automatiquement

Git reset --hard ca83a6df = l’identifiant de la commit

Changer le message dans une commit

Git commit -m « message »

* Git commit –amend -m “nv message”

(-m permet de changer le message)

Si on oublie de commit un fichier on utilise :

Git commit –amend –no-edit

Une image contenant texte

Description générée automatiquement

* 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.

Il est possible d'annuler son commit public avec la commande git revert. L'opération revert annule un commit en créant un nouveau commit. C'est une méthode sûre pour annuler des changements, car elle ne risque pas de réécrire l'historique du commit.

**git revert HEAD^**

**git revert pour une branche public**

**git reset pour une branche privée**

Gardez à l'esprit que   git revert sert à annuler des changements commités, tandis que   git reset HEAD   permet d'annuler des changements non commités

**$ ssh-keygen -t rsa -b 4096 -C "johndoe@example.com" :** permet de créer pair de clé ssh public et privée

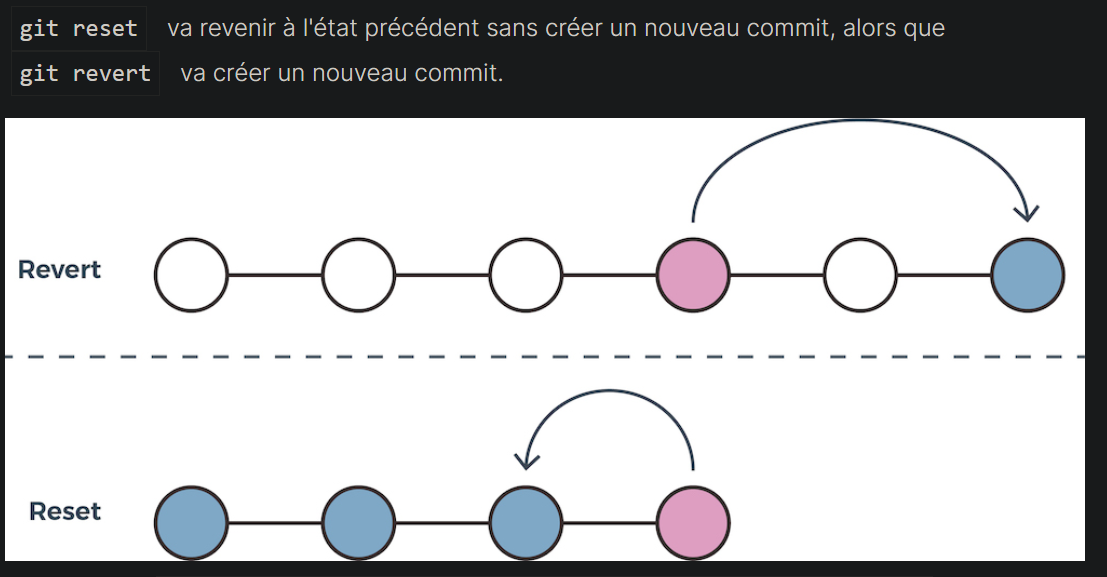
Pour la trouver, il suffit d'aller à l'adresse : C:\Users\VotreNomD'Utilisateur\, et d'afficher les dossiers masqués.

Dans ce dossier, vous avez donc deux fichiers, votre clé publique et votre clé privée.

La clé id\_rsa.txt est votre clé privée alors que la clé id\_rsa.pub est votre clé publique. Ici nous allons utiliser votre clé publique seulement. Vous pouvez copier votre clé publique en l'ouvrant dans un bloc-notes.

Ajouter la clé sur github.

* git revert HEAD^ permet d'annuler un commit en créant un nouveau commit.
* La commande ssh-keygen permet de générer un duo de clés SSH.
* Vous pouvez configurer une nouvelle clé SSH sur GitHub.



* 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.

Git reflog : Git dispose d'un outil encore plus puissant, poussant le journal de logs à l’extrême.

Git checkout ‘id commit’

savoir qui a contribué à quoi.

déterminer où des bugs ont été introduits ;

annuler les changements problématiques.

Une image contenant texte

Description générée automatiquement

En résumé

* 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.