GIT / GIT HUB

utilisation de git :

1. commencer par installer git
2. ouvrir un terminal (git s'utilise principalement depuis un terminal, exemple cmd de window)
3. vérifier la version avec la commande : git --version
4. définir notre identité avec les commande suivante :
   → git config --global user.name « votre nom »
   → git config --global user.email « votre email »

git est prêt a être utilisé.

Maintenant il faut se mettre dans le dossier de notre projet, pour se déplacer on utilise la commande « cd » suivi du chemin, si vous voulez créer un dossier depuis la ligne de commande il faut utiliser la commande « mkdir » suivi du nom du dossier.

Une fois dans le projet il faut initialiser git avec la commande « **git init** » .

Pour connaître l'état de nos fichier il faut faire un « **git status** ».

Pour enregistrer les fichiers sur git :
1) on commence par sélectionner ce que l'on veut enregistrer avec la commande « **git add** », si on veut sélectionner tous les fichiers on ajoute « ***** » après le add sinon on peut indiquer les fichiers avec leur nom.
2) ensuite il faut valider l'enregistrement de ses fichiers au moyen de la commande « **git commit** », cette commande doit être accompagnée de l'option **'-m « notre commentaire »'** pour indiquer un message qui précise la nature des modifications apportées.

Pour obtenir la liste des commits réalisés au sein de notre git il faut faire un « git log ».
Pour sortir de cet affichage, il faut appuyer sur la touche '**q**'.

git utilise un système de branche, il s'agit d'une copie du projet initial qui permet de le modifier puis de le fusionner au projet sans créer de problème ou de mieux travailler à plusieurs.

On peut créer autant de branches que l'on veut, la branche master est la branche principale.

- Pour voir la liste des branches on fait un « **git branch** ».
- Pour créer une nouvelle branche **git branch** suivie d'un nom.
- Pour changer la branche courante on utilise « **git checkout nom_de_la_branche** »
- Pour supprimer une branche dans git « **git branch -d nom_de_la_branche** ».
- Une fois le travail terminé il faut fusionner les commits sur la branche master pour ce faire on utilise la commande « **git merge** ». On se place sur la branche master est on merge les branches avec « **git merge origin nom_de_la_branche** ».

GIT / GIT HUB

dépôt distant :

Pour pouvoir travailler à plusieurs il faut utiliser un dépôt distant comme git hub ou git lab.

Commencer par créer un projet sur git hub, une fois cela fait il suffit d'utiliser la commande « **git remote add origin adresse_du_projet** » pour l'initialiser, cela va associer le dépôt local au dépôt distant. L'adresse du projet se trouve sur git hub et est sauvegardé.

Pour vérifier les dépôts distants, c'est avec la commande « **git remote** ».

Pour déposer notre projet sur le dépôt distant on fait un « **git push origin master** »,
où origin est le label du dépôt et master est la branche à déposer.
On peut aussi utiliser la commande « **push** » si on se trouve sur la branche sur laquelle on veut envoyer

Pour récupérer le projet il existe deux commandes :
→ **git pull origin master** : qui récupère la nouvelle version du projet en cours d'utilisation (le plus important)
→ **git clone adresse_du_projet** : qui clone tout le projet (à n'utiliser que la première fois)

voir la liste des conflits :

« **git diff --base 'nom_du_fichier'** » : pour visualiser les conflits d'un fichier.
« **git diff 'branche_source' 'branche_cible'** » : pour visualiser les conflits d'une branche à une autre.
« **git diff** » : pour énumérer tout les conflits actuels.

Autres commandes utiles :

- « **git reset** » : pour revenir à la version du dernier commit.
- « **git rm nom_du_fichier** » : supprimer le fichier indiquer avec son nom.
- « **git grep 'expressions et/ou mots'** » : permet de rechercher dans tous les fichiers une expression ou un mot.
- « **gitk** » : est l'interface graphique du dépôt local.
- « **git archive --format=tar master** » : permet de créer un fichier tar contenant les composants d'une branche.
- « **git archive --format=zip master** » : permet de créer un fichier zip contenant les composant d'une branche.
- « **git rebase nom_de_la_branche** » : permet de réappliquer des commits sur une autre branche.
- « **git fsck** » : effectue une vérification des fichiers. Tous les fichier corrompus seront identifiés.