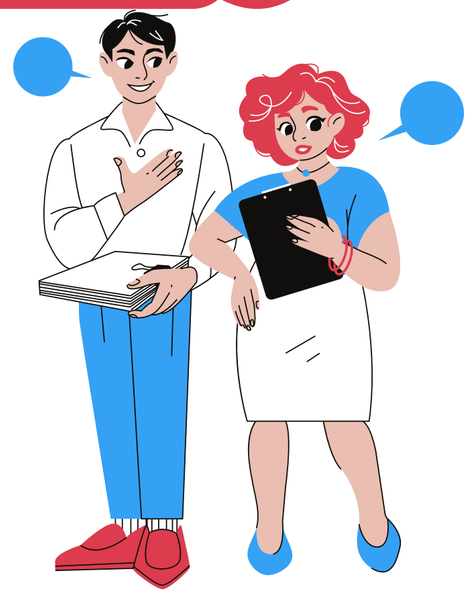
l'outill de versionning , GIT & GITHUB



C'est quoi un outil de versionning ?

Les systèmes de contrôle de version sont une catégorie d'outils logiciels qui permettent de gérer les changements apportés à un code source au fil du temps.

GIT

Git cache un système de versionning décentralisé.
Il existe un dépôt (repository) dans lequel tous les changements sont incorporés et qui permet aux utilisateurs d’y télécharger leurs propres copies de travail. Créé en 2005 par Linus Torvalds.

les commandes utilisées en GIT :

- git config

Usage: git config –global user.name “[name]”
Usage: git config –global user.email “[email address]”
On l’utilise pour configurer les préférences de l’utilisateur

- git init

Usage: git init [repository name]
Cette commande est utilisée pour créer un nouveau dépôt GIT

- git clone

Usage: git clone [url]
La commande git clone est utilisée pour la vérification des dépôts.

- git add

Usage: git add [file]
La commande git add peut être utilisée pour ajouter des fichiers à l’index.

- git commit

Usage: git commit -m “[Type in the commit message]”
La commande git commit permet de valider les modifications apportées au HEAD.

- git branch

Usage: git branch
La commande git branch peut être utilisée pour répertorier, créer ou supprimer des branches.

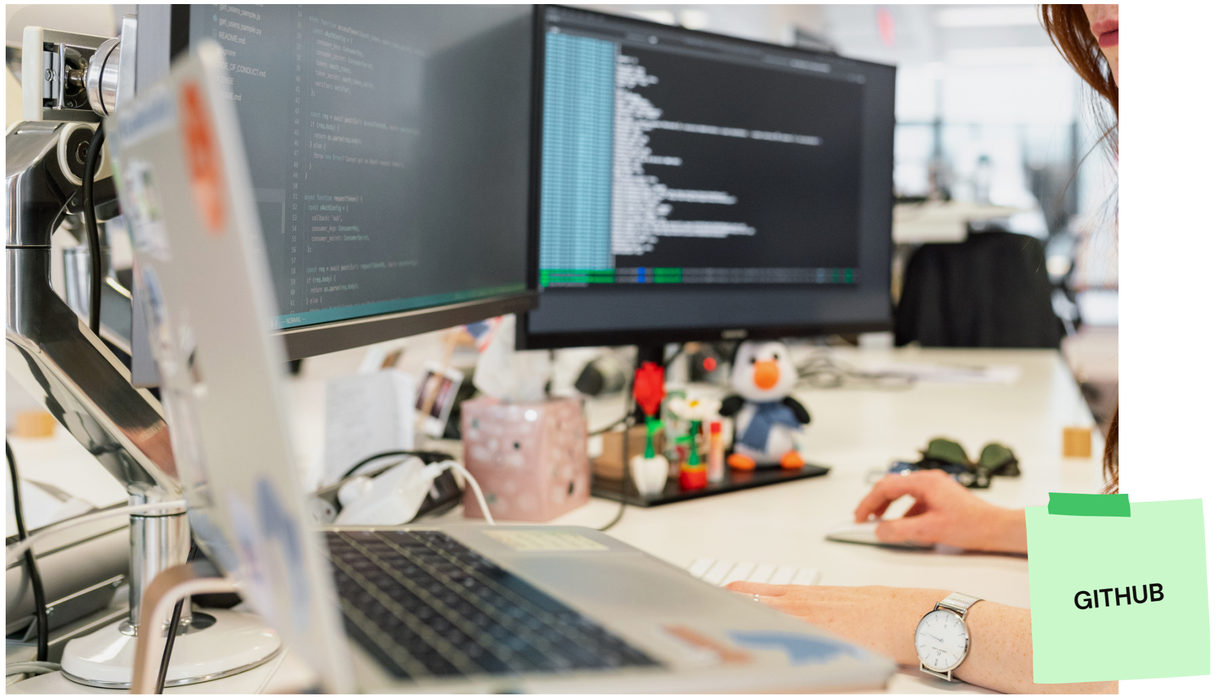
- git push

Usage: git push [variable name] master
Git push est une autre commandes GIT de base. Un simple push envoie les modifications locales apportées à la branche principale associée

- git pull

Usage: git pull [Repository Link]
Pour fusionner toutes les modifications présentes sur le dépôt distant dans le répertoire de travail local

GITHUB



Definition :

GitHub est un site web et un service de cloud qui aide les développeurs à stocker et à gérer leur code

les Avantages de GitHub :

Service gratuit, bien qu'il ait également des services payants.

Recherche très rapide dans la structure des dépôts.

Grande communauté et aide facile à trouver.

Il propose des outils pratiques de coopération et une bonne intégration avec Git.

GIT COMMANDE

1. BRANCH
2. MERGING
3. TAG

Branch c'est le fait de créer une copie du projet principale et la modifier sans toucher la version originale et pour créer une branche il faut simplement taper :

git add brach <name>

Pour l'idée du tag, c'est de faire un hachtag pour votre copie et mentionner la "version" que vous voulez. Pour cela il faut simplement taper

git add tag <message de tag >

Une opération git merge s'effectue en lançant la commande « git merge <nom de la branche à fusionner> ». Lorsque nous effectuons une fusion, git fusionne toujours avec la branche actuelle à partir de laquelle nous effectuons l'opération .

GIT MERGING

COMMANDES IMPORTANTES

Git Checkout Commands

- `git checkout <branch-name>` – **Switch to a different Git branch**.
- `git checkout -b <branch-name>` – **Create a new branch** and switch to it.
- `git checkout -b <branch-name><remote-name>/<branch-name>` – Create a local branch from the remote Git branch and checkout that branch.
- `git checkout <commit hash>` – **Checkout a previous Git commit**.
- `git checkout <tag name>` – Checkout a Git tag in a detached HEAD state.
- `git checkout -b <branch-name><tag-name>` – **Checkout a Git tag** as a branch.

Git Stash Commands

- `git stash` – Create a stash with local modifications and revert back to the head commit.
- `git stash list` – Display a list of all stashes in your repository.
- `git stash show` – View the content of your most recent stash. This will show your stashed changes as a diff between the stashed content and the commit from back when the stash was created.
- `git stash drop <stash>` – Remove a stash from the list of stashes in your repository.
- `git stash pop <stash>` – Apply a stash to the top of the current working tree and remove it from your list of stashes.
- `git stash apply <stash>` – Apply a stash on top of the current working tree. The stash will not be removed from your list of stashes.
- `git stash clear` – Remove all stashes from your repository.

Git Merge Commands

- `git merge` – Combine two or more development histories together. Used in combination with fetch, this will combine the fetched history from a remote branch into the currently checked out local branch.
- `git merge <branch-name>` – **Merge changes from one branch** into the branch you currently have checked out.
- `git merge --abort` – Aborts the merge process and restores the project's state to before the merge was attempted. This works as a failsafe when a conflict occurs.
- `git merge --continue` – Attempt to complete a merge that was stopped due to file conflicts after **resolving the merge conflict**.
- `git merge --squash` – Combine all changes from the branch being merged into a single commit rather than preserving them as individual commits.
- `git merge --no-commit` – Combine branch into the current branch, but

Merci !