NOVEI FORMATION

LA MANU

L'ÉCOLE DES MÉTIERS DU NUMÉRIQUE
Manufacture de compétences

SUPPORT APPRENANT

Git et GitHub

CONFIDENTIEL

Git et GitHub

Ce que l'on a vu…

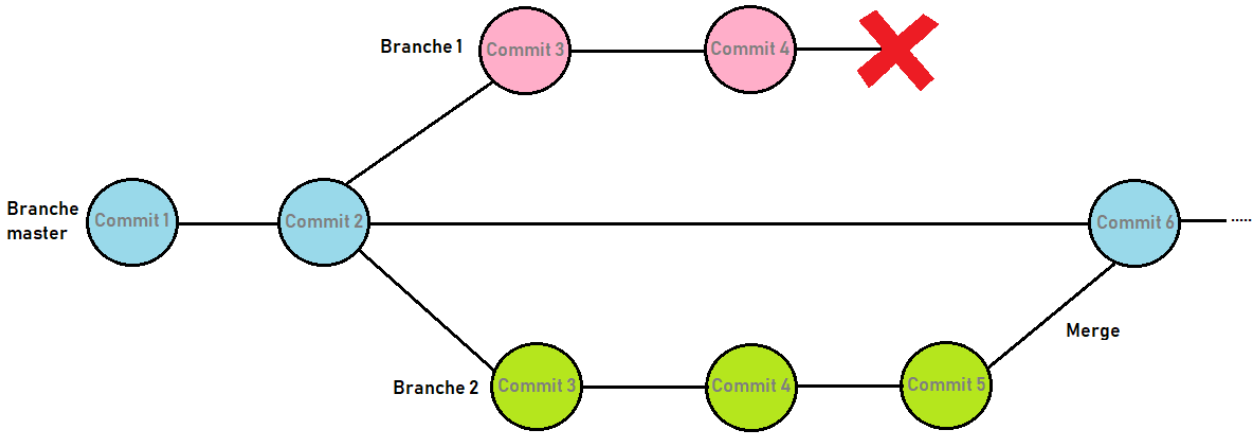
Git

Git est un logiciel de gestion de versions, aussi appelé logiciel de versioning, qui a été créé par Linus TORVALDS en 2005.
Le versioning a plusieurs avantages : il permet d'avoir un historique de toutes les modifications effectuées sur un projet et de revenir sur les versions précédentes, il permet également de travailler à plusieurs sur un même projet ou encore de travailler sur des branches afin de pouvoir tester des modifications sans risquer d'altérer le projet.

Git fait partie des systèmes de versioning dits décentralisés ou distribués, c'est-à-dire que l'historique du projet sera stocké à la fois sur un serveur central et sur l'ordinateur de chaque utilisateur. Ainsi, chaque utilisateur travaille sur le projet en local puis envoie ses modifications sur le serveur central distant. D'autres gestionnaires de versions comme SVN ou CVS, utilisent un système de versioning dit centralisé, c'est-à-dire que le projet est uniquement stocké sur un serveur central.
Le système distribué est plus rapide que le système centralisé, et il est aussi plus sécurisé car une sauvegarde du projet sera présente sur différentes machines contrairement au système centralisé.

Sous Windows, le logiciel Git peut être utilisé de 2 façons, soit via une interface graphique (Git GUI), soit en lignes de commande (Git Bash). Que l'on utilise la ligne de commande ou l'interface graphique, les mêmes commandes vont être exécutées.
Un repository est un répertoire ou dépôt de travail géré par Git. Il est initialisé grâce à la commande **git init** avec Git Bash, celle-ci doit être exécutée à la racine du dossier de projet.
Une fois le repository initialisé il est possible de réaliser un premier commit. Pour cela, avec Git Bash, il faut utiliser la commande **git add** pour ajouter tous les fichiers à suivre, puis **git commit -m** pour ajouter un message descriptif au commit. Ce message doit être clair et précis, il indique les modifications qui ont été apportées au projet.
On va pouvoir faire une sauvegarde sur un nouveau commit à chaque modification importante du projet (ajout/suppression de fichiers, ajout/suppression d'éléments, ajout/suppression de fonctionnalités…). Chaque nouveau commit est enregistré dans l'historique sans écraser les précédents, ainsi chaque commit correspond à une version du projet à un moment donné. Outre les modifications effectuées et le message descriptif, on trouve d'autres informations liées au commit, telles que sa date, son auteur, et son identifiant (SHA). Toutes ces informations peuvent se révéler utiles notamment lorsque l'on travaille à plusieurs sur un même projet ou si l'on souhaite revenir sur une version précédente du projet.

Le logiciel Git permet de faire des tests sur le projet en travaillant sur des branches sans risquer d'endommager le master. En exécutant la commande **git branch** on crée une nouvelle branche qui est une copie de la branche master. On peut ensuite se positionner sur la branche créée avec la commande **git checkout** et faire les tests que l'on souhaite. Il est possible de faire plusieurs branches. Lorsqu'on est satisfait d'un test on peut utiliser la commande **git merge** pour faire fusionner la branche avec la branche master.



Un des intérêts d'utiliser Git est d'avoir une sauvegarde de son projet sur un serveur distant. Cela permet ainsi d'avoir une sauvegarde de sécurité en cas de problème sur son ordinateur et cela permet également de travailler à plusieurs sur le même projet. Pour sauvegarder son projet sur un serveur distant, il faut créer un dépôt distant sur un site dédié tel que GitHub ou Bitbucket et ajouter un remote pour faire la liaison entre le dépôt local et le dépôt distant.

GitHub

GitHub est un site web permettant à ses utilisateurs de créer des repositories distants sur lesquels ils pourront stocker une sauvegarde de leurs projets. GitHub permet également de collaborer à plusieurs sur un même projet. Depuis son rachat par Microsoft en 2018, la plateforme de GitHub permet d'avoir des dépôts privés gratuits. Il est également possible de créer une page web qui sera hébergée par GitHub. Cela peut être pratique pour héberger son CV numérique par exemple.

Avant d'envoyer son commit sur un dépôt distant, il faut d'abord créer un nouveau repository sur GitHub et lui donner un nom, un lien propre à ce repository sera créé. On peut ensuite utiliser ce lien pour créer un nouveau remote dans Git Bash en utilisant la commande **git add remote**. Une fois que la liaison entre le dépôt local et le dépôt distant est établit il ne reste plus qu'à envoyer le commit sur le dépôt distant avec la commande **git push**. L'ajout du remote ne se fait qu'une seule fois, à la création du repository. Pour envoyer un 2ème commit, 3ème commit, etc... avec Git Bash il suffit de reprendre les commandes **git add** pour ajouter les fichiers ou modifications à suivre, **git commit -m** pour ajouter un message au commit et enfin **git push** pour envoyer le commit sur GitHub.

Il est possible de voir l'historique des différents commits sur le site de GitHub, ainsi que les modifications qui ont été apporté au projet. Dans le cas d'un projet commun, les différents collaborateurs peuvent utiliser la commande **git clone** pour faire une copie du projet initial et ensuite travailler sur une branche. Lorsque des modifications sont faites sur le projet il est important que les différents collaborateurs récupèrent ces modifications avant de pusher d'autres changements pour éviter les conflits, pour cela il faut utiliser la commande **git pull**.

Les commandes Git Bash utiles à retenir

Nom de la commande	Description
git init	Initialisation du dépôt. Création d'un dossier caché .git dans le dossier courant, il contiendra tous les fichiers nécessaires au bon fonctionnement du dépôt.
git add *nomFichier*	Ajout des fichiers à suivre.
git commit -m "*description*"	Création d'un commit.
git status	Affichage du statut des fichiers (non suivi, modifié, à jour).
git log	Affichage de l'historique des commits (le plus récent en 1er).
git log --oneline	Affichage condensé de l'historique des commits (le plus récent en 1er).
git branch *nomBranche*	Création d'une branche.
git branch -d *nomBranche*	Suppression d'une branche.
git checkout *nomBranche*	Se déplacer sur une branche.
git checkout *SHADuCommit*	Revenir sur une version précédente.
git help	Affichage de l'aide avec un descriptif des commandes courantes.
git --amend -m *"nouveauMessage"*	Modification du message du dernier commit.
git remote add *nomRemote LienRepository*	Ajout d'un remote pour connecter le dépôt local au dépôt distant.
git push *nomRemote nomBranche*	Envoi des modifications sur le dépôt distant.
git pull *nomRemote nomBranche*	Mise à jour du dépôt local.
git merge *nomBranche*	Fusion de la branche avec le master.
git clone *LienRepository*	Faire une copie d'un dépôt distant.
git config --list	Permet de connaitre l'état de notre configuration.
git config --global user.email *"email"*	Modification de l'adresse mail.
git config --global user.name *"usernameGithub"*	Modification de l'username.

Liens utiles

https://openclassrooms.com/en/courses/2342361-gerez-votre-code-avec-git-et-github
https://openclassrooms.com/en/courses/1233741-gerez-vos-codes-source-avec-git
https://doc.ubuntu-fr.org/git
https://git-scm.com/book/fr/v1/D%C3%A9marrage-rapide
http://rogerdudler.github.io/git-guide/index.fr.html
https://www.youtube.com/watch?v=V6Zo68uQPqE
https://www.grafikart.fr/formations/git
https://fr.atlassian.com/git/tutorials/what-is-git
https://www.hostinger.fr/tutoriels/tuto-git/
https://www.hostinger.fr/tutoriels/commandes-git/
http://try.github.io/
https://learngitbranching.js.org/

Bonnes pratiques

Rappel des principales bonnes pratiques	
☐	Placer le git init à la racine du dossier de projet.
☐	Ne pas utiliser la commande **git add *** (ou **git add .**). Il est important de maîtriser ce que nous mettons à jour.
☐	Mettre un message significatif de la modification apportée dans le commit.
☐	Ne jamais push quelque chose sur le master qui ne serait pas fonctionnel.
☐	Vocabulaire spécifique : • Repo local VS repo distant • Commit • Remote • Branche • Versioning