Gestion des versions à l'aide de Git

L'objectif de ce TP est de manipuler l'outil de gestion de versions Git, en assurant la maintenance d'un répertoire de travail de manière structurée et ordonnée.

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2.

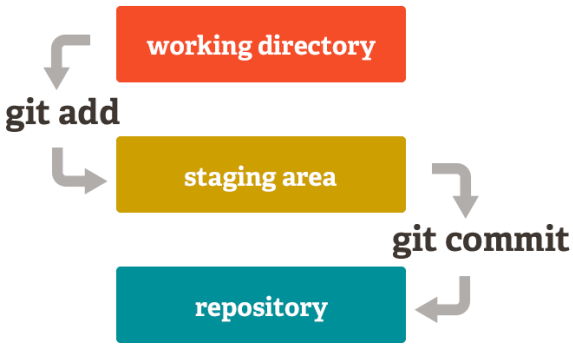
En 2016, il est considéré comme le logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes.

Création d'un nouveau dépôt

Pour créer un nouveau dépôt, on utilise la commande **git init monrepo**. Cette commande initialise un dépôt Git dans le répertoire *monrepo* (celui-ci est créé s'il n'existe pas). Ce répertoire contient alors à la fois une version de travail (dans *monrepo*) et un dépôt Git (dans *monrepo/.git*).

1. Initialiser un nouveau dépôt Git dans un répertoire S*andwich*, et créez le fichier *burger.txt* qui contient la liste des ingrédients d'un burger, un ingrédient par ligne : *steak, salade, tomate, cornichon, fromage*.

Ajout intermédiaire et définitif : Add et Commit



2. Vérifiez avec **git status** l'état dans lequel se trouve votre dépôt. Vos modifications (l'ajout du fichier *burger.txt*) devraient être présentes seulement dans la copie de travail.
3. Préparez *burger.txt* pour le commit avec **git add burger.txt**. Utilisez **git status** à nouveau pour vérifier que les modifications ont bien été placées dans l'index. Puis, utilisez **git diff HEAD** pour observer les différences entre l'index est la dernière version présente dans l'historique de révision (qui est vide).
4. Commitez votre modification avec **git commit -m "Ajout de burger.txt"**. Le message entre guillemets doubles décrira la nature de votre modification.
5. Exécutez à nouveau **git status**, pour vérifier que vos modifications ont bien été commitées.

6. Essayez à présent la commande **git log** pour afficher la liste des changements effectués dans ce dépôt ; combien y en a-t-il ? Quel est le numéro (un hash cryptographique en format SHA1) du dernier commit effectué ?
7. Essayez la commande **git log –-oneline.** Qu'est-ce que vous remarquez ?
8. Créez le fichier *hot_dog.txt* puis commitez-le. Ajoutez-y ses ingrédients : *pain, moutarde, hot-dog* et committez. Créez le fichier *panini.txt* et commitez-le, puis modifiez le contenu en ajoutant : *viande hachée, mayonnaise, fromage* et commitez. Ajoutez au fichier *hot_dog.txt* la ligne : *ketchup* et commitez.
9. Essayez la commande **git log –-oneline.** Qu'est-ce que vous remarquez ?

Naviguer entre versions

Il se peut qu'on ait besoin de charger une version ultérieure temporairement pour effectuer des tests et revenir sur notre travail initial (**git reset**), ou laisser une version stable de votre projet pour continuer le travail sur une partie que vous envisager entamer (créer des branches).

git checkout est une commande très puissante. Elle vous permet de voyager entre différentes branches (voir plus loin) et aussi de revenir temporairement à une version précédente de votre copie de travail.

10. Modifiez le fichier *panini.txt* en rajoutant la ligne : *pain*. Ajoutez la modification à l'index mais ne la commitez pas. Exécutez **git reset panini.txt**. *V*érifiez avec **git status** le résultat.
11. Votre modification a été « retirée » de l'index. Vous pouvez maintenant la jeter à la poubelle avec la commande **git checkout panini.txt**, qui récupère dans l'historique sa version correspondante au tout dernier commit. Essayez cette commande, et vérifiez avec **git status** qu'il n'y a maintenant plus aucune modification à commiter.
12. Regardez l'historique de votre dépôt (avec **git log -oneline**) et choisissez dans la liste le commit où vous avez modifié le fichier *hot_dog.txt* (1er changement où vous avez saisi les ingrédients). Exécutez **git checkout COMMITID** où COMMITID est le numéro de commit que vous avez choisi. Vérifiez que l'état de vos sandwiches est maintenant revenu en arrière, au moment du commit choisi. Que dit maintenant **git status** ?
13. Exécutez la commande **git log –oneline**. Elle ne doit plus afficher les commits postérieurs à l'état actuel, sauf si vous ajoutez l'option **--all**.
14. Vous pouvez retourner à la version plus récente de votre dépôt avec **git checkout master**. Vérifiez que cela est bien le cas. Que dit maintenant **git status** ?

Créer une branche de travail

Une branche dans Git est simplement un pointeur léger et déplaçable vers un de ses commits. La branche par défaut dans Git s'appelle master. Créer une brache dans Git signifie alors diverger de la ligne principale de développement (master) pour continuer à travailler sans se soucier de cette ligne principale. La commande **git branch** sert à créer, à répertorier, à renommer et à supprimer des branches. En revanche, elle ne permet pas de basculer entre les branches ou de reconstituer un historique forké. C'est la raison pour laquelle cette commande est étroitement liée aux commandes **git checkout** (qui permet de se positionner sur une branche) et **git merge** (qui permet de fusionner des branches).

15. Créez une branche « Test » à l'aide de la commande **git branch Test**, et positionnez-vous sur elle (**git checkout Test**).
16. Modifier le fichier *panini.txt* en rajoutant une ligne en fin du fichier *: le pain doit être mi-cuit*. Commitez cette dernière modification. Vérifier l'état du dépôt à l'aide de **git status**. Quel est le résultat de la commande **git log –oneline** ?
17. Revenez sur la branche principale (matser) et vérifiez le contenu du fichier *panini.txt*. Qu'est-ce que vous remarquez ?
18. Pour mettre à jour le contenu du master avec la dernière modification faite sur la branche « Test », il faut utiliser la commande **git merge Test** (au niveau du master). Vérifiez l'état du dépôt et le contenu du fichier *panini.txt.* Que retourne la commande de log ?

Gestion des conflits

19. Assurez-vous que vous êtes sur la branche master (à l'aide de **git status**), puis modifiez maintenant le contenu du fichier *burger.txt* en rajoutant la ligne : *sauce burger* et committez.
20. Mettez-vous ensuite sur la branche Test et modifiez le contenu du fichier *burger.txt* en rajoutant la ligne : *sauce moutarde* puis committez.
21. Revenez sur la branche master et vérifiez le contenu de *burger.txt*. Essayez alors de fusionner le contenu de Test. Qu'est-ce qui se passe ?
22. Ouvrez le fichier *burger.text* et modifiez le pour ne garder pour les lignes de conflits que la ligne : *sauce moutarde*. Commitez la modification.
23. Lister les versions sur master. Qu'est-ce que vous remarquez ?