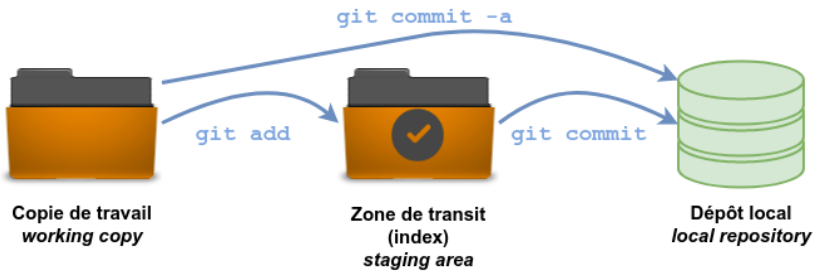
TP GIT

I. Travailler Git en local et gérer les branches



1.1 Créer un projet git en local

1. Créer le répertoire TpGit
2. Dans ce répertoire, créer 2 fichiers index.html et style.css (non vides)
3. Initialiser le dépôt git (git init)
4. Propager en s'assurant que les 2 fichiers sont sur le dépôt (git add, git commit)

1.2 Envoyer le commit sur le dépôt distant Github

1. Sur Github, créer un dépôt intitulé TpGit
2. Lier le dépôt distant au dépôt local
3. Envoyer les commits du dépôt local TpGit vers le dépôt distant sur GitHub
4. Vérifier sur GitHub que les fichiers ont été importés

1.3 Travailler avec git en local

1. Modifier index.html
2. Ajouter readme.txt
3. Propager les modifications (git add)
4. Modifier style.css
5. Propager les modifications (git add)
6. Voir l'historique des modifications (git log)

1.4 Créer une branche idee

1. Créer une branche 'idee'
2. Sur cette branche :
    a. Modifier index.html
    b. Ajouter idee1.txt
    c. Propager (sur cette branche)
    d. Voir l'historique

1.5 Travailler aussi sur la branche master

En parallèle, on travaille aussi sur la branche principale :

1. Modifier style.css
2. Propager
3. Créer idee2.txt
4. Propager
5. Voir l'historique

1.6 Voir l'état de chaque branche

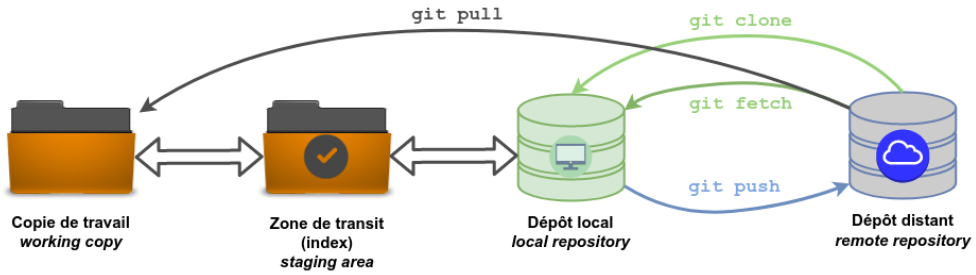
Voir l'état du projet sur chacune des branches, les différences, ...

1.7 Fusionner la branche idee sur la branche master

1. Maintenant, on suppose que ce qui a été développé sur chacune des branches est à conserver, on va donc fusionner la branche idee sur la branche master
2. S'assurer que les modifications réalisées sur les 2 branches ont été conservées
3. Voir l'historique

II. Travailler à plusieurs sur github

Cette partie doit être réalisé à plusieurs (au moins 3).



2.1 Création des projets : init, clone

- user1 :
  1. Créer un projet de votre choix
  2. Initialiser le projet (git init, git add, git commit)
  3. Pusher votre git local sur Github
- user2 :
  1. Cloner le projet de user 1 depuis Gtihub
- user3 :
  1. Cloner le projet de user1 depuis Github

2.2 Chaque développeur travaille sur sa partie

- user1 : apporter des modifications en local
- user2 : ajouter les fichiers de user2 en local (nouvelle branche)
- user3 : ajouter les fichiers de user3 en local (nouvelle branche)

2.3 Partage du travail de l'équipe

1. Chaque utilisateur va propager les modifications sur Github.
2. User2 et user3 devrait envoyer des pull request concernant les modifications qu'ils ont apportées

3. User1 doit tout d'abord valider les changements et les merger par la suite dans la branche racine.

III. Pratiquer et corriger les erreurs

3.1 Supprimer une branche crée par erreur

1. Créer un répertoire test
2. Initialisez un dépôt
3. Créez une branche BrancheTest
4. Apporter des modifications
5. Supprimer la branche BrancheTest

3.2 Annuler des modifications faites sur la branche principale

1. Apporter des modifications à la branche principale
2. Consulter l'état des différentes modifications (git status)
3. Enregistrer les modifications (git stash) et voir les conséquences sur la branche principale.
4. Créer une nouvelle branche
5. Exécuter la ou les commandes enregistrées sur cette nouvelle branche (git stash apply)

3.3 Rectifier les modifications après un commit

1. Ajouter le fichier readme.txt à votre dépôt
2. Vérifier que le fichier a été ajouté (git log)
3. Supprimer le dernier commit (git reset)
4. Créer une nouvelle branche BrancheCommit
5. Appliquer maintenant le commit sur la nouvelle branche (git reset)

3.4 Ajouter un fichier oublié lors d'un commit

1. Ajouter le fichier readme.txt à votre dépôt
2. Ajouter le fichier au commit sans modifier le message (git commit –amend)