**Tp2 CONTRENDUE**

**Objectif du TP :**

Apprendre à collaborer efficacement sur un projet en utilisant Git. Chaque binôme doit construire un site web sur le thème de Git, tout en mettant en pratique les commandes Git pour gérer le travail en équipe.

**Étapes principales :**

**1. Configuration de Git**

Avant de commencer, configurez Git sur votre ordinateur avec vos informations personnelles.  
Commandes à utiliser :

bash

git config --global user.name "Votre nom"

git config --global user.email "votre\_email@example.com"

git config --global push.default "simple"

**2. Création du projet sur GitHub**

* **Collègue A :** Crée un dépôt GitHub (TD4) et ajoute son binôme comme collaborateur.
* **Collègue B :** Clone le dépôt sur son ordinateur, crée un fichier index.html, le committe et le pousse sur GitHub.

**3. Synchronisation et contributions**

* **Collègue A :** Tirez (pull) les modifications de votre binôme, ajoutez du contenu à index.html, commitez, puis poussez.
* **Collègue B :** Créez un fichier commandes.html listant les commandes Git connues, puis suivez les étapes pour tester fetch, status, et pull.

**4. Gestion des conflits avec stash**

Lorsqu'il y a des conflits (modifications simultanées sur le même fichier), utilisez :

* stash pour sauvegarder temporairement vos changements,
* pull pour récupérer les mises à jour,
* stash apply pour réintégrer vos modifications locales.

**5. Branches et publication sur GitHub Pages**

* Créez une branche gh-pages pour publier le contenu du site en ligne sur GitHub Pages.
* Synchronisez les branches avec des commandes comme merge ou push.

**6. Gérer les conflits complexes**

Si des conflits surviennent sur une même section de fichier, ils doivent être résolus manuellement en éditant le fichier concerné. Les délimitations <<<<<<, =======, et >>>>>> aident à identifier les différences.

**7. Collaboration avancée et outils graphiques**

Utilisez des outils comme gitk pour visualiser l’historique des branches et comprendre la structure du projet.

**Résultat attendu :**

À la fin du TP, le binôme doit avoir :

1. Un dépôt GitHub partagé.
2. Un site web fonctionnel publié via GitHub Pages.
3. Une compréhension des outils et commandes Git pour collaborer efficacement.

**Points clés :**

* La communication est essentielle pour éviter les conflits.
* Git conserve toujours un historique, ce qui permet de revenir à des états antérieurs si nécessaire.
* Les branches facilitent le travail en parallèle sans interférences.

Si besoin, je peux encore simplifier certains points ou ajouter des détails.

**Voici une liste des commandes principales utilisées dans le TP, accompagnée d'explications simples pour chacune d'elles :**

**Configuration de Git**

**git config --global user.name "Votre nom"**

Permet de configurer le nom d'utilisateur qui sera associé à vos commits dans Git.

**git config --global user.email "votre@email.com"**

Définit l'adresse e-mail associée à vos commits. Elle peut être fictive si vous travaillez en local.

**git config --global push.default "simple"**

Configure Git pour ne pousser que la branche courante, évitant ainsi des erreurs potentielles.

**Répertoires et collaboration**

**git clone [URL]**

Télécharge un projet Git distant et en crée une copie locale sur votre ordinateur.

**git pull --ff-only**

Met à jour votre copie locale avec les dernières modifications du dépôt distant uniquement si les branches ne divergent pas. C’est une méthode sûre qui évite les conflits.

**git push**

Envoie vos commits locaux vers le dépôt distant, rendant vos modifications visibles par vos collaborateurs.

**Gestion des fichiers et des modifications**

**git status**

Montre l’état des fichiers dans le répertoire : fichiers modifiés, ajoutés ou prêts à être commités.

**git log**

Affiche l’historique des commits pour suivre les modifications passées.

**git add [fichier]**

Ajoute un fichier modifié ou nouveau au prochain commit.

**git commit -m "Message"**

Enregistre vos modifications dans un commit avec un message décrivant ce que vous avez changé.

**Gestion avancée**

**git fetch**

Télécharge les modifications du dépôt distant sans les intégrer à votre copie locale. Vous pouvez analyser les différences avant de fusionner.

**git stash**

Sauvegarde temporairement vos modifications en cours sans les commiter, pour pouvoir travailler sur autre chose.

**git stash apply**

Récupère les modifications précédemment sauvegardées avec stash.

**git reset [SHA]**

Annule les commits récents et revient à un état précédent du projet.

**git revert -n HEAD**

Annule un commit spécifique sans effacer son historique.

**Résolution des conflits et fusion**

**git merge [branche]**

Fusionne une branche spécifiée dans la branche courante.

**git pull**

Fait à la fois un fetch et un merge, intégrant directement les modifications distantes à votre branche locale.

**git checkout -- [fichier]**

Annule les modifications locales d’un fichier pour le remettre dans son état d’origine depuis le dernier commit.

**git branch [nom]**

Crée une nouvelle branche dans votre projet.

**git checkout [branche]**

Change de branche pour travailler sur une autre version du projet.

**gitk --all**

Affiche une vue graphique de l’historique des commits et des branches.